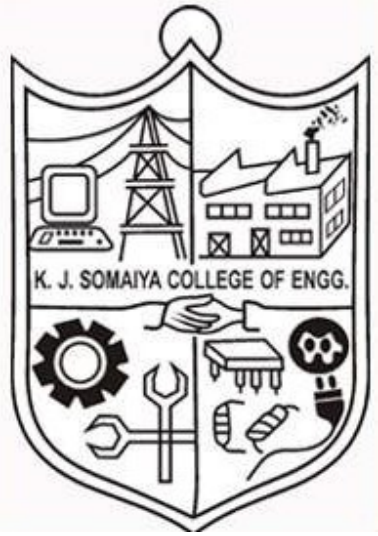
Face Mask Detection

SHRAVANI DHOTE [1812077]

MEET DOSHI [1812078]



Supervisor: Dr. Ninad Mehendale

A Project Report Submitted for the Coursework of
Mini Project

Department of Electronics
K.J. Somaiya College of Engineering
University of Mumbai
Mumbai, India

April 2021

Acceptance Certificate

2UXP-601

Department of Electronics

K. J. Somaiya College of Engineering

The Mini Project Report entitled "Face Mask Detection" submitted by Shravani Dhote (1812077) and Meet Doshi (1812078) of Third Year Electronics under the guidance of Dr. Ninad Mehendale may be accepted for being evaluated.

Date: 30 June 2021

Dr. Ninad Mehendale

Dr. Jagannath Nirmal

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We declare that we have properly and accurately acknowledged all sources used in the production of this report. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: 30th June 2021

Shravani Dhote (1812077)

Meet Doshi (1812078)

Abstract

COVID-19 pandemic caused by novel coronavirus has increased human suffering, undermined the economy, turned the lives of millions of people around the world upside down, and majorly affected the health, economic, environmental and social domains. It initially forced governments worldwide to impose lockdowns to prevent virus transmissions. Vaccines are available now, but the process of vaccination will take time. Till then, at the individual level, we can take precautions like wear mask to restrain the spread of the coronavirus. It's a key measure to suppress transmission. The major cause of the outspread was a lack of proper safeguards. If this task was strictly ensured by machines rather than human error then we would have better control over the outbreak. Here we show that usage of artificial intelligence is very reliable for this purpose and less prone to errors, almost all test datasets had an accuracy of over 95% on validation set, 86% on testing data and about 60% on live RPi feed. This project introduces a face mask detection system using a Raspberry Pi that grants access depending on whether the person has worn a mask or not. It is developed with a Convolutional Neural Network that is simple enough and computationally efficient. Many factors come into the picture in tracking a face mask detector's accuracy, whether the image is incomplete, is the person wearing anything other than a mask, multiple objects in a single image and the list goes on. This project was built keeping in mind lower cost over efficiency and reliable throughput. Some of the major tasks are tackled using Image recognition, Object detection and remote access. The task of controlling appliances or entrance to a locale was achieved with high standards. Face mask detector can be used in crowded places like schools, colleges, bus stops, malls, etc to grant access only to those who wear a mask. Also by observing the position of the face mask on the face, we can ensure that an individual wears it the correct way. It is hoped that our study would be a useful tool to reduce the spread of this communicable disease.

Contents

List of Figures

List of Tables

CHAPTER 1
Introduction

The COVID-19 pandemic is the defining global health crisis of our time. The novel coronavirus causes respiratory illness ranging from the common cold to more severe diseases such as severe acute respiratory syndrome (SARS) and Middle East respiratory syndrome (MERS), according to the WHO. It was first reported in Wuhan, China in December 2019 and has since spread worldwide. The major symptoms of coronavirus by the World Health Organization (WHO) are Fever, dry cough, tiredness, diarrhoea, loss of taste, and smell. People with mild symptoms recover from the disease without needing hospital treatment. They are advised to stay isolated or home quarantined. People affected seriously require oxygen and need intensive care. Many precautionary measures have been taken to fight against coronavirus. Few of them are washing hands regularly, wearing a mask, maintaining a safe distance, refraining from touching eyes, nose, and mouth are the main, where wearing a mask is the simplest one.

Currently, India has been hit by the devastating second wave of the COVID-19, with the death toll increasing daily and many states have not yet issued a complete lockdown. The brutal second wave resulted in a spike in the number of cases leading to a shortage of hospital beds and medical oxygen. Even India's huge vaccination drive is now struggling. Due to this, there arises a severe need to follow all the precautionary measures, face mask being the primary one. Wearing a face mask can help in restricting the transmission of the disease along with the safety of individuals.

1.1 Need of Project

The ongoing COVID-19 pandemic has brought the entire country to a standstill with its second wave resulting in a huge spike in the number of daily cases. The medical infrastructure of the country is not able to support this surge leading to increased mortality. Stopping the transmission of this deadly coronavirus is of utmost importance.

1.2 Aim of the Project

Artificial Intelligence is the key to achieving tasks that cannot be done by static computers even with a mass amount of processing power. Artificial Intelligence is currently gaining its name from its applications in the fields of Image Processing, Recognition and Object Detection. Thus, the field of machine learning came across this global pandemic and measures to reduce its impact is of utmost urgency.

1.3 Objective of the Project

In this project, we provide a cheaper yet efficient remote host face detection model. The objective of this project is to design a model which grants access to a person after detecting whether the person is wearing a mask or not. We modelled it using a Convolutional Neural Network which is simple enough and computationally efficient. To train, validate and test the model, we used a dataset that consisted of 3725 masked faces images and 3828 unmasked faces images. The hardware we have used in this project is a cheap yet fast computer named Raspberry Pi which is assisted with a camera module. The image classification problem is solved only with the help of image classification and neural networks.

Summary:- Thus, by implementing this project, we would be able to mitigate the spread of COVID-19. We have covered previous works and their pros and cons in the literature review section so that the reader can get a better understanding of this project. Even though all image classification requires neural networks, but using them can prove computationally very

expensive, that's why we have designed a network that has just a few hidden layers which make this detection mechanism very fast. We covered the problem of overfitting a model and showed that what decisions must be made while designing such a project with step by step execution of the project in the methodology section followed by Results where we showed that electrical appliances and mechanical parts can be controlled based on whether the face mask is detected or not we have used LEDs and servo motor as an example for each. We showed the reliability of the classifier and the rate of false-positive and true negatives in this section as a metric of correctness. Few tips and suggestions to improve the current model along with its comparison with other models and future scope is also mentioned. Thus, we hope that this would be a useful tool in preventing virus transmission.

CHAPTER 2

Literature Review

Machine learning technique is an exciting branch of artificial intelligence which is all around us in the modern world. It shows us how powerful data is in a totally advanced way. Enabling programs to learn and improve continuously provides smart alternatives to analyzing vast volumes of data. It has its application in image recognition, speech recognition, automatic language translation et cetera. Wuttichai's *et al.*[1] work focuses on performance of machine learning algorithms for face mark detection. KNN, SVM and MobileNet algorithms are used to fine the best suited algorithm for checking who is wearing a mask in a real-time situation. The problem with KNN is that it doesn't learn a discriminative function from the training data but memorizes the training dataset itself. The SVM model is driven by a linear function but it doesn't provide probabilities. It only outputs the class identity. Based on depth wise separable convolution mobileNet algorithm works fast with low latency. Results of the project suggested that mobile net is the best accuracy algorithm because it reduces network size and number of parameters. MobileNet is then followed by SVM and lastly KNN. Camera quality and illumination are the reasons due to which the percentage of the accuracy rate of all the three algorithms drops gradually.

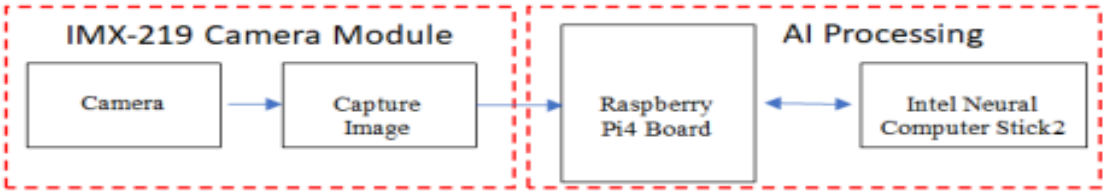


FIGURE 2.1: Block Diagram of the Masked Face Detection model[1].

FIGURE 2.2: The masked face detection system implemented on Raspberry Pi4 broad[1].

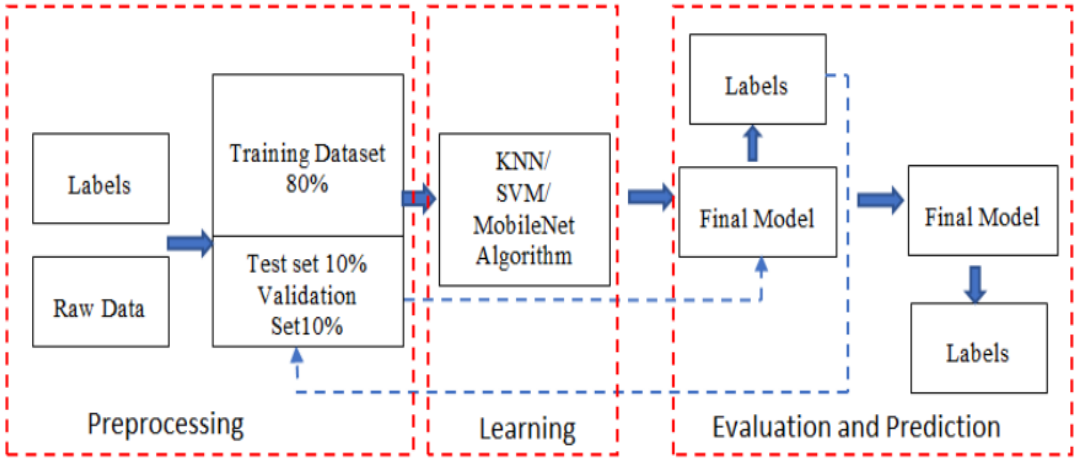


FIGURE 2.3: Model of Machine Learning System whioch detects masks using KNN, SVM and MobileNet[1].

The results of Wuttichai's *et al.*[1] work is supported by Sanjaya' *et al.*[2] MobileNet model of machine Learning. It was implemented in 25 cities with help of various sources of images. From the results, it's found that the percentage of people wearing a mask in the cities have a strong correlation to the vigilance index of COVID-19 which came out to be 0.69. Sanjaya *et al.*[2] used two datasets one taken from Kaggle and public place CCTV,

shop, traffic lights etc. For 25 different cities. The total data used had 1916 data with a mask and 1930 data without a mask. The data was split into 75% for training and 25% for testing. The model was built in six steps which were constructing the training image generator for augmentation, the base model with mobilenetV2, adding model parameters, compiling the model, training the model, and saving the model for the future production process. Finding the percentage of people in different cities was not enough because from the table we can see that two cities in Jawa Timur were categorized into the five lowest percentage of people using a mask. But in reality, they were the hotspots for Covid cases.

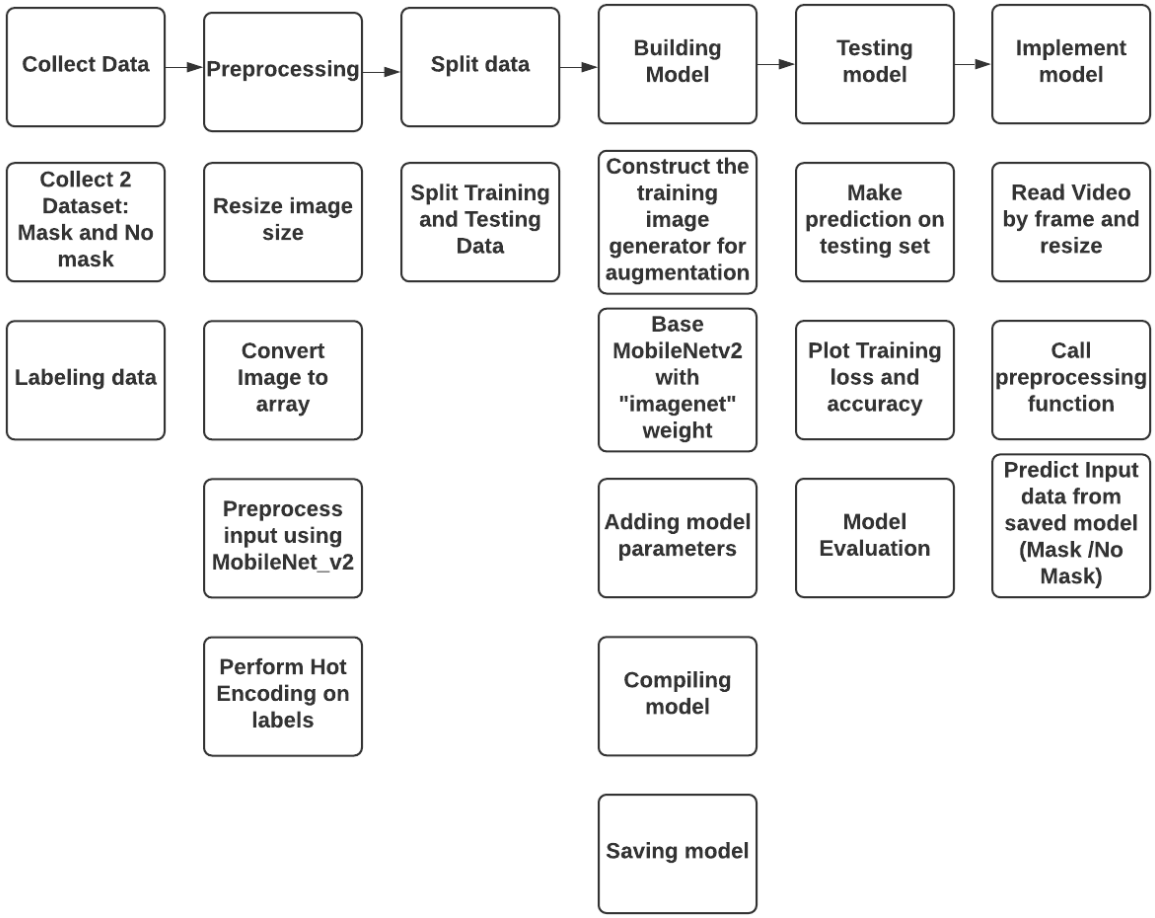


FIGURE 2.4: Steps of Building the Model using MobileNetV2 to find correlation to the vigilance index in 25 citiesof Indonesia [2].

Similarly Rahman *et al.*[3] implemented facial mask detection model for smart city network,where just like Sanjaya *et al.*[2] work, all the public places are monitored with CCTV.

His work would be a useful tool to reduce the spread of Covid for many countries as the corresponding authority is informed through the city network when a person without a mask is detected. For this CNN was used for feature extraction from the image, later these features are learnt by multiple hidden layers. It was found that the training accuracy increases but the testing accuracy decrease because further training results cause overfitting on the training data which occurs when a model learns the unwanted patterns of the training sample train. The training model showed 98.7% accuracy and an AUC of 0.98 on unseen data. One of the major challenges faced by the system is classifying a face covered by a hand since it almost looks like the person wearing a mask. While any individual without a face mask is going on any vehicle, this system can't find that individual effectively. The system fails when there is a network issue because the information about the violator is sent via SMS.

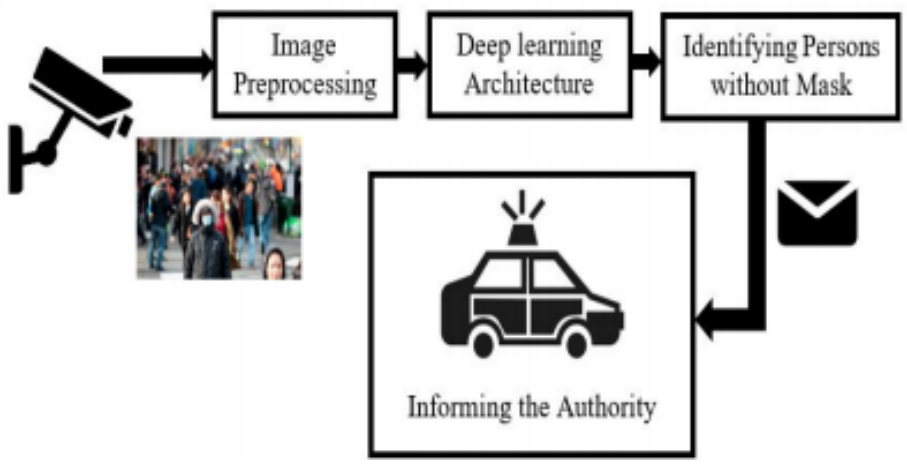


FIGURE 2.5: Block Diagram of facial mask detection model using CNN for smart city network.
Where in the authority is informed through the city network when a person without a mask is detected [3]

After considering entire cities, need for monitoring specific places arose. Due to the surge in cases, the government was forced to impose lockdown which resulted in the shutdown of all economic activity and also the production at manufacturing plants across most sectors were halted. While there is an urgency to continue production, there is a considerably more need to guarantee the well being of the labour force at the factory. So, Khandelwal *et al.*[4] had expressed in his work about using computer vision to monitor the workforce's activity

using CCTV feeds which were already present at the site because this provided adequate coverage of the factory, as well as minimum hardware, was required at the plants. It's a robust social distancing measurement algorithm using a mix of modern-day deep learning and standard projective geometry methods. From a total of 6589 people, 80% of the data was for training and the remaining 20% for validation were utilized in the preparation of the MobileNetV2 model. For Face detection 20% of the total data as a validation set with 380 as mask and 460 images as no mask was considered. The AUROC of the model was 97.6%. A couple of limitations were seen in the model like it couldn't accurately group face images where the face is partially hidden by a person in the front. Moreover, the model couldn't identify faces if the camera height is more than 10 feet. This model has been successfully tested at manufacturing plants of Aditya Birla Group.

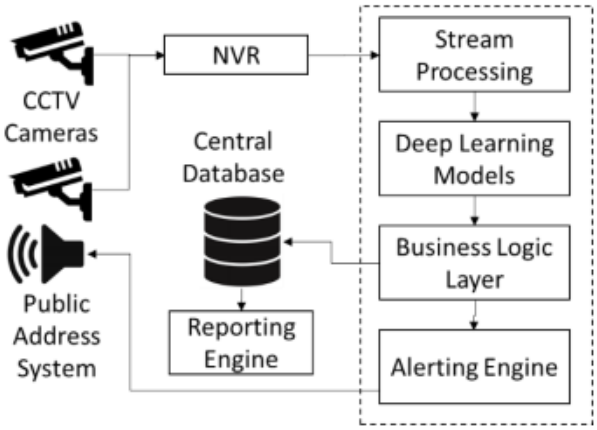


FIGURE 2.6: Block Diagram of the System Using Computer Vision to enhance Safety of Workforce [4].

RetinaFaceMask, a high-accuracy and efficient face mask detector proposed by Jiang *et al.*[5], was able to overcome the above issues. It's a one-stage detector consisting of a feature pyramid network to fuse high-level semantic information with multiple feature maps to focus on detecting face masks. The models used were ResNet and MobileNet. To reject predictions with low confidences and the high intersection of union, a cross-class object removal algorithm was implemented. The Face Mask Dataset contains 7959 images which were split into a train, validation and test set with 4906, 1226 and 1839 images respectively. Transfer learning was proposed to transfer learned knowledge from a source task to a related

target task due to the limited size of the face mask dataset. It is 2.3% and 1.5% higher than the baseline result in the face and masks detection precision individually, and 11.0% and 5.9% higher than baseline.

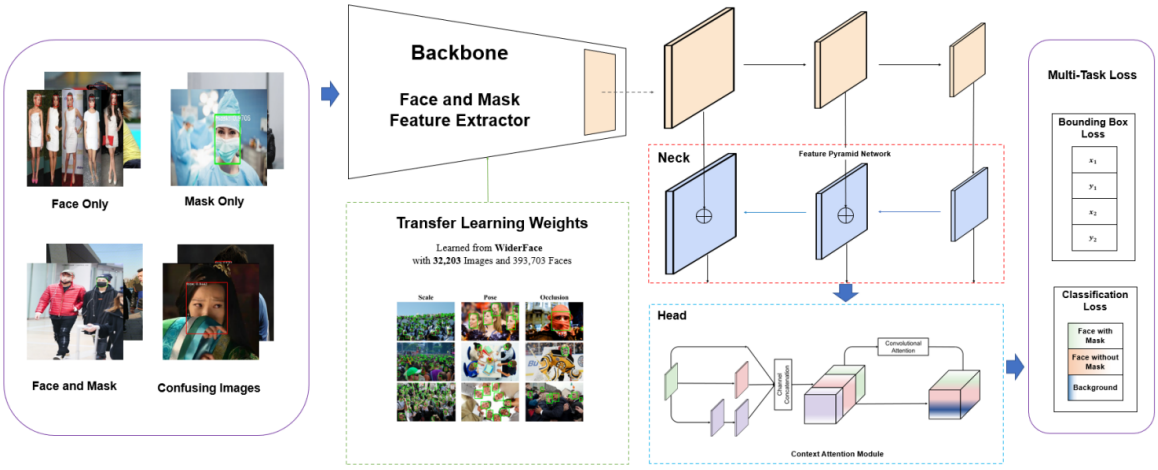


FIGURE 2.7: Architecture of RetinaFaceMask.
It is a one-stage detector consisting of a feature pyramid network. Models used are ResNet and MobileNet [5].

A different approach by Nagrath's *et al.*[6] whose work focuses on SSDMNV2 which has Single Shot Multibox Detector as a face detector and MobilenetV2 architecture as a framework for the classifier which is similar to previous models. SSDMNV2 performs competently in differentiating images having frontal faces with masks from images having frontal faces without masks. The dataset consisted of a total of 5521 images. To increase the precision of mask detection without being too resource-heavy, the DNN module was used from OpenCV, which includes a 'Single Shot Multibox Detector (SSD) object detection model with ResNet-10 as its backbone architecture. This method helps in detecting faces in real-time, even on embedded devices like Raspberry Pi. A pre-trained model MobileNetV2 was used for the predictions. The accuracy of the model is about 0.9264.

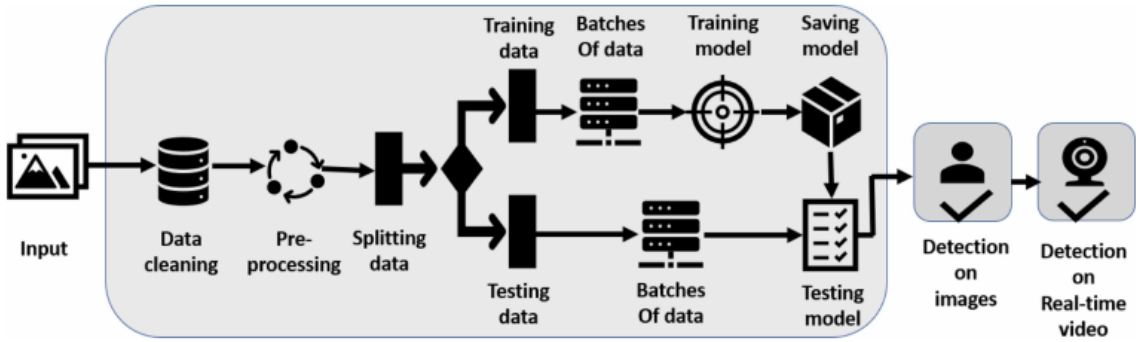


FIGURE 2.8: Flow Diagram of SSDMNV2.
It has Single Shot Multibox Detector as a face detector and MobilenetV2 architecture as a framework for the classifier [6].

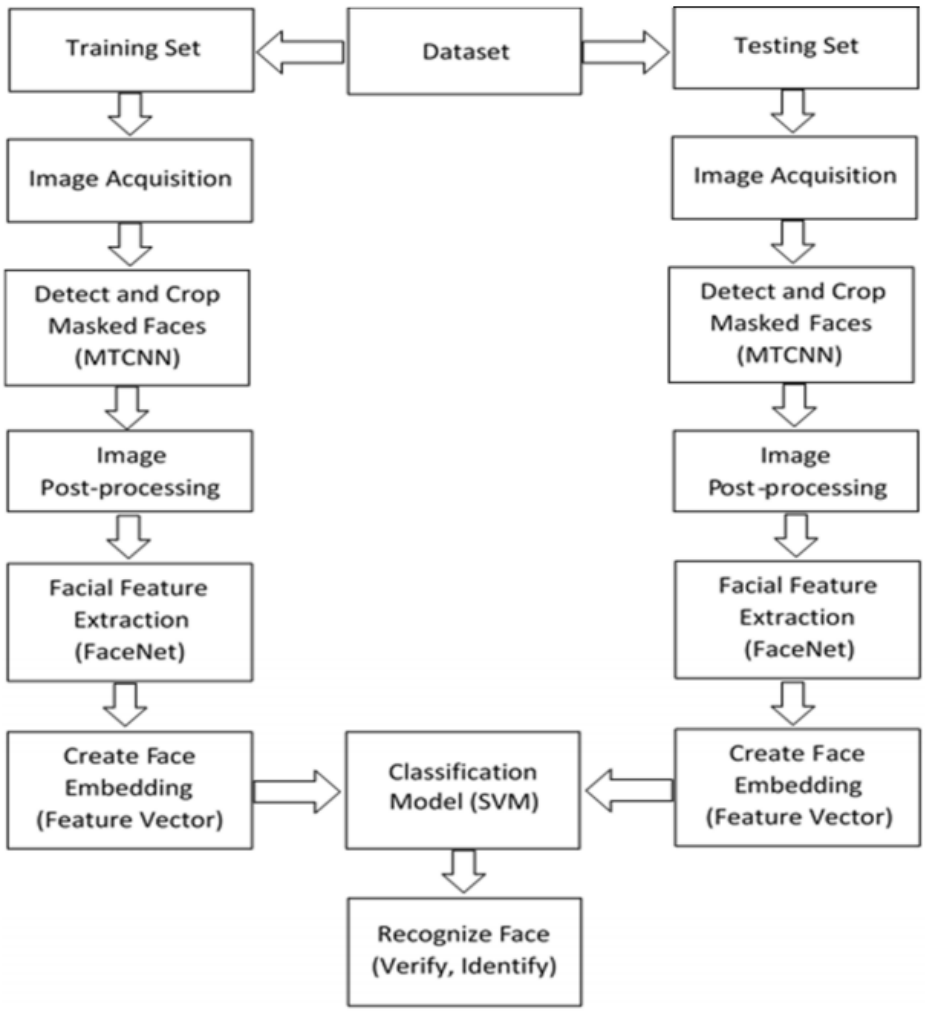


FIGURE 2.9: Flow Chart of Masked Face Detection using Multi-Task Cascaded Convolutional Neural Network (MTCNN) [7]

The accuracy of face recognition decreases because of conditions like changing pose or illumination like in Wuttichai's *et al*.[1] work, degraded images etc. Plenty of researches have been done to overcome these but difficulties created by masks are usually disregarded. Ejaz's *et al*.[7] work focuses on this problem. The problem has been approached using Multi-Task Cascaded Convolutional Neural Network (MTCNN). Google FaceNet embedding model performs the extraction of facial features. FaceNet developed on 22 deep convolutional network layers. And finally, the Support Vector Machine (SVM) does the task of classification. The dataset was divided into 70% train and 30% test data. Different combinations of masked and non-masked face images are made to find out the best recognition accuracy. This method may not be appeasement for all types of masks. A more accurate and sophisticated approach may be needed.

To grant access, Baluprithviraj *et al*.[8] proposed a smart device which was built using raspberry Pi with AI model and Camera. It's integrated with a mobile app that sends an alert message after identifying whether a person is wearing a mask. This smart device automatically opens the door only if people wear face mask. It finds its application at numerous places like schools, colleges, corporate offices etc. MIT App Inventor was used to develop the mobile app. TensorFlow was used in the Convolutional Neural Network (CNN) model which includes Keras library and OpenCV to identify the face mask. The dataset consists of total 1376 images out of which 690 images containing people with face masks and 686 without face masks images. One of the limitations of this system is that during night time the image won't be clearly visible.

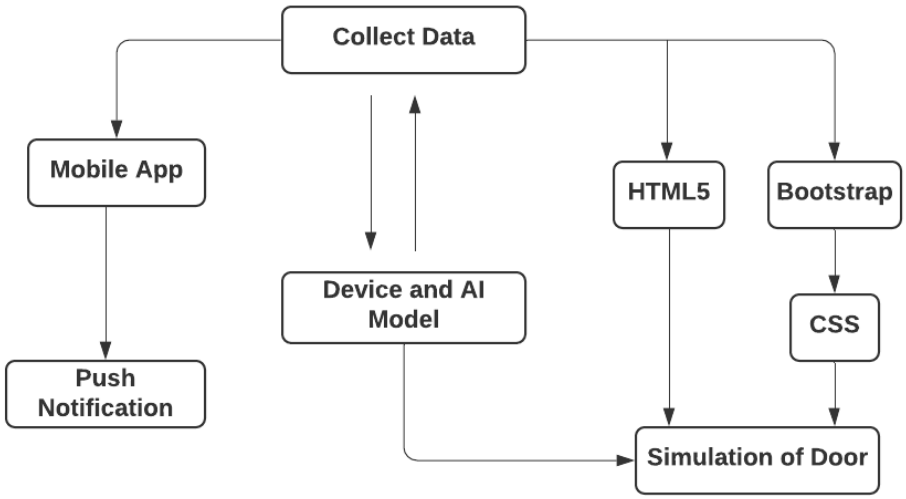


FIGURE 2.10: Block diagram for Smart Door.
It automatically opens only if people wear face mask using CNN and Tensorflow on Raspberry Pi[8].

2.1 Comparison of Different Implementations

Paper	Author	*Method*	*Accuracy (%)*
1	Wuttichai *et al.*[1]	*SVM*	*78.1*
		KNN	*72.5*
		MobileNet	*88.7*
2	Sanjaya *et al.*[2]	*MobileNetV2*	*92*
3	Rahman *et al.*[3]	*CNN*	*98.7*
4	Khandelwal *et al.*[4]	*MobileNetV2*	*97*
5	Jiang *et al.*[5]	*MobileNet*	*82.3*
		ResNet	*93.4*
6	Nagrath *et al.*[6]	*SSDMNV2, MobilenetV2*	*94*
7	Ejaz *et al.*[7]	*MTCNN*	*98.5*
8	Baluprithviraj *et al.*[8]	*CNN*	-

TABLE 2.1: Accuracy Metric Comparison for referred works

Thus, we studied and compared the work of other authors who implemented a similar project which helped us to get a rough idea about the approach on how to implement our project which is step by step described in the following chapter.

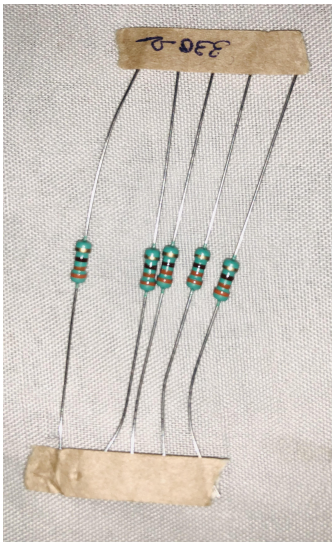
CHAPTER 3

Methodology

3.1 Components and Required Libraries

3.1.1 Hardware

(a) Copper Clad

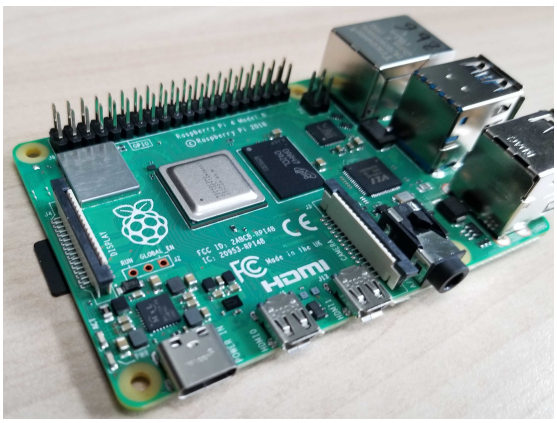


(b) 330 Ω Resistors

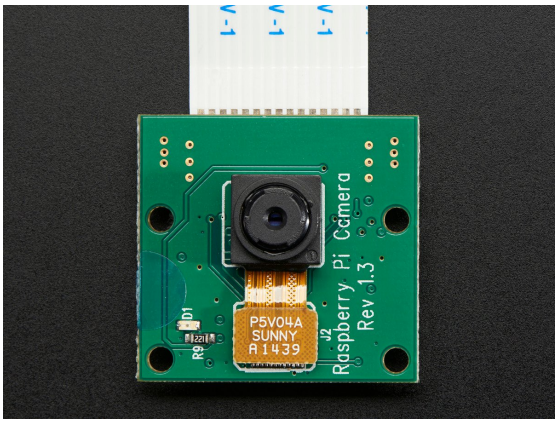
(c) Solder Iron

(d) Solder Wire

(e) 32GB SD Card



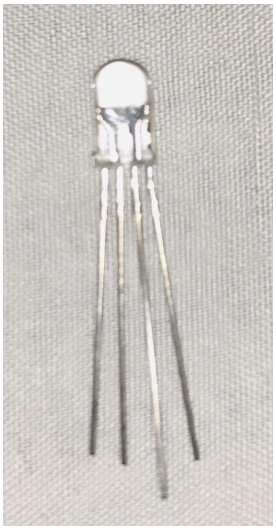
(f) Raspberry Pi 4B



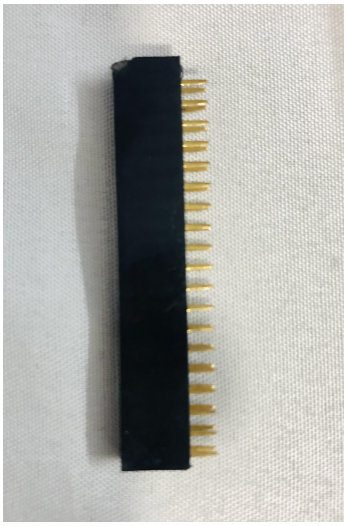
(g) RPi Cam V1.3

(h) 15W Power Adapter

(i) Servo Motor SG90



(j) RGB LED CC



(k) Female Header 20 x 2

FIGURE 3.1: Hardware Components

3.2 Project Flowchart

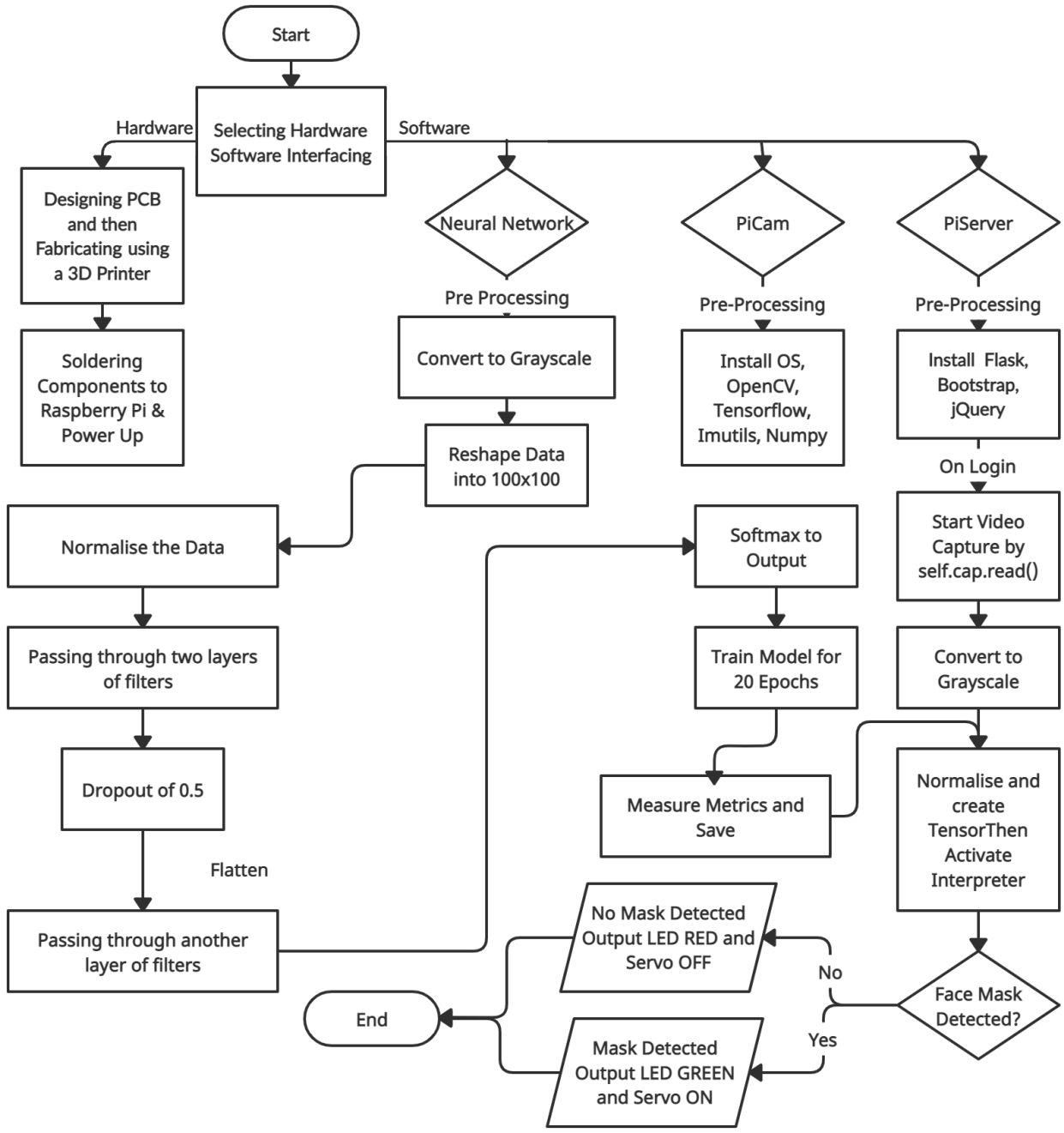


FIGURE 3.2: Project Flowchart

The project was started with assembling the hardware first. This way we can decide which pins on Raspberry pi will be interfaced. The software part is divided into three components- CNN model, Pi camera OpenCV object detection & hosting the video feed to a remote server.

3.2.1 Software

Execute the following commands on your shell to import required libraries.

I Raspberry Pi OS
- https://www.raspberrypi.org/software/operating-systems/

II Etcher
- https://www.balena.io/etcher/

III Tflite
- pip3 install https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp37-cp37m-linux_armv7l.whl

IV OpenCV Dependencies
- sudo apt-get update && sudo apt-get upgrade
- sudo apt-get install build-essential cmake unzip pkg-config
- sudo apt-get install libjpeg-dev libpng-dev libtiff-dev
- sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
- sudo apt-get install libxvidcore-dev libx264-dev
- sudo apt-get install libgtk-3-dev
- sudo apt-get install libcanberra-gtk*
- sudo apt-get install libatlas-base-dev gfortran
- sudo apt-get install python3-dev

V OpenCV 4
- cd ∼
- wget -O opencv.zip https://github.com/opencv/opencv/archive/4.3.0.zip
- wget -O opencv_contrib.zip
- https://github.com/opencv/opencv_contrib/archive/4.3.0.zip unzip opencv.zip
- unzip opencv_contrib.zip
- mv opencv-4.0.0 opencv
- mv opencv_contrib-4.0.0 opencv_contrib

VI Numpy
- pip3 install numpy

VII CMake and compile OpenCV 4

-
- cd ∼/opencv
- mkdir build
- cd build
- cmake -D CMAKE_BUILD_TYPE=RELEASE \-D CMAKE_INSTALL_PREFIX =/usr/local \
  -D OPENCV_EXTRA_MODULES_PATH=∼/opencv_contrib/modules \
  -D ENABLE_NEON=ON \
  -D ENABLE_VFPV3=ON \
  -D BUILD_TESTS=OFF \
  -D OPENCV_ENABLE_NONFREE=ON \
  -D INSTALL_PYTHON_EXAMPLES=OFF \
  -D BUILD_EXAMPLES=OFF ..
- make -j4
- sudo make install
- sudo ldconfig

VIII Face Mask Detector

- git clone git@github.com:meetdoshi90/Face-Mask-Detector.git
- cd rpi_based_mask_detector_new
- pip3 install -r requirements.txt

3.3 Costing

Sr No.	*Component*	*Cost in INR*	*Quantity*	*Link*
1	Raspberry Pi 4B 4GB RAM	*5074/-*	*1*	*RPi 4*
2	RPi Cam V1.3	*350/-*	*1*	*RPi Cam V1.3*
3	Servo SG90	*100/-*	*1*	*SG 90*
4	15W Power Adapter (Owned)	*800/-*	*1*	*Adapter*
5	Female Header 20 x 2	*30/-*	*1*	*Header*
6	VGA to Micro HDMI	*150/-*	*1*	*VGA To Micro HDMI*
7	Copper Clad	*15/-*	*1*	*Copper Clad*
8	Solder Iron (Owned)	*150/-*	*1*	*Solder Iron*
9	Solder Wire (Owned)	*40/-*	*1*	*Solder Wire*
10	SD Card 32GB (Owned)	*400/-*	*1*	*SD Card*
11	Resistors 330 Ω	*3/-*	*3*	*Resistors*
12	RGB LED	*7/-*	*1*	*RGB LED*
13	*Total Cost*	*5729/-*	*14*	-

TABLE 3.1: Costing Sheet

We have stated all the components which one needs to buy to assemble the current project. Additionally you need to have a keyboard, mouse, a VGA monitor, power socket, a computer with VNC viewer installed & an internet connection. The total sum which one might need if it does not have any of the above mentioned components is 7119/-. We excluded the costs of the components owned by us which are power adapter, solder iron, wire, and an SD card.

3.4 Specifications

Sr. No	*Component*	*Specifications*
1	Raspberry Pi 4B	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 4GB LPDDR4-3200 SDRAM 2.4GHz & 5.0GHz IEEE 802.11ac wireless, Bluetooth 5.0 Gigabit Ethernet 2 USB 3.0 ports; 2 USB 2.0 ports Raspberry Pi standard 40 pin GPIO header 2 x micro-HDMI ports 2-lane MIPI DSI display port 2-lane MIPI CSI camera port 4-pole stereo audio and composite video port 5V DC via USB-C connector(minimum 3A) 5V DC via GPIO header(minimum 3A) Power over Ethernet (PoE) enabled Operating Temperature: 0-50 degrees C ambient
2	RPi Cam V1.3	Fully Compatible with all Raspberry Pi models A & B 5MP Omnivision 5647 Camera Module Still Picture Resolution: 2592 x 1944 Video: 1080p@ 30fps, 720p @60fps and 640x480p 60/90 15-pin MIPI Camera Serial Interface Size: 20 x 25 x 9mm Weight 3g
3	Servo SG90	Torque: 4.8V 25.00 oz-in Speed: 4.8V: 0.12 sec/60Bachelors of Technology Weight: 0.32 oz (9.0 g) Motor Type: 3-pole Rotation/Support: Bushing Pulse Width: 500-2400μ
4	15W Power Adapter	5.1V/ 3.0A DC output 96-264V AC operating input range 1.5m 18 AWG captive cable with USB-C output connector
5	RGB LED CC	Package: 5mm, T-1 3/4 Luminous Intensity: 5000mcd, 9000mcd, 6000mcd Typical Forward Voltage: 2V, 3.2V, 3.2V Typical Forward Current 30 mA Pins: 4
6	Resistors 330Ω	Resistance (Ω): 330 Power (Watts): 0.25W Tolerance: $\pm$5%

TABLE 3.2: Specifications of Hardware Components

3.5 Hardware

3.5.1 Design Flowchart

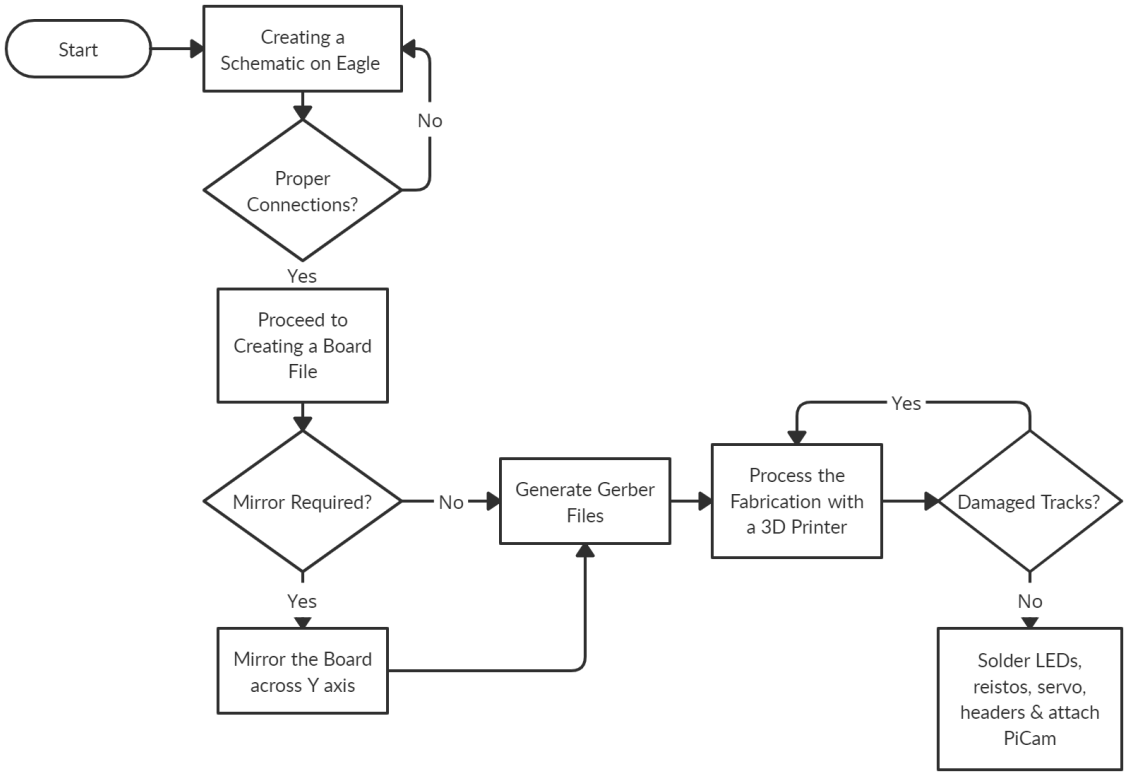


FIGURE 3.3: Hardware Flowchart

The hardware requirement for this project is minimal except for an Raspberry pi. The software used for designing and fabricating the PCB are EAGLE©and CopperCAM©. The PCB track width is set to 60. 9mm drill was used for holes. LEDs can be either 3 single color LEDs or an RGB LED with common cathode.

3.5.2 Assembling

We will go through step by step procedure to design hardware for this Face Mask Detector.

- First selecting which pins on GPIO will be used to drive out external components. Here is a pinout of the Raspberry Pi we have used

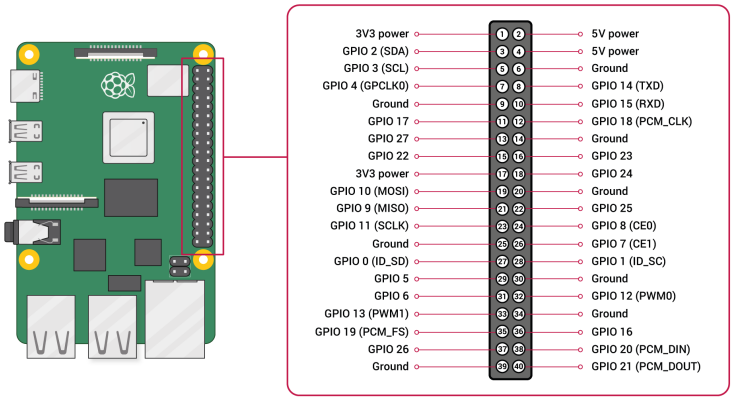


FIGURE 3.4: GPIO Pins

We need to drive 3 LEDs and a Servo motor via our GPIO. For LEDs we can use any GPIO pins available. To drive a servo motor we need a PWM signal, Raspberry Pi's GPIO pins can generate software generated PWM, but it also has hardware generated PWM pins; namely GPIO 12/13. For this project we choose pins 11,13,15 to drive our LEDs and choose GPIO 12 to drive our servo motor.

- Now after selecting our interfacing pins, we can move to EAGLE©for our schematic design. Find the components from the list and connect using wires so that we get the following schematic. Ensure proper connections on each port.

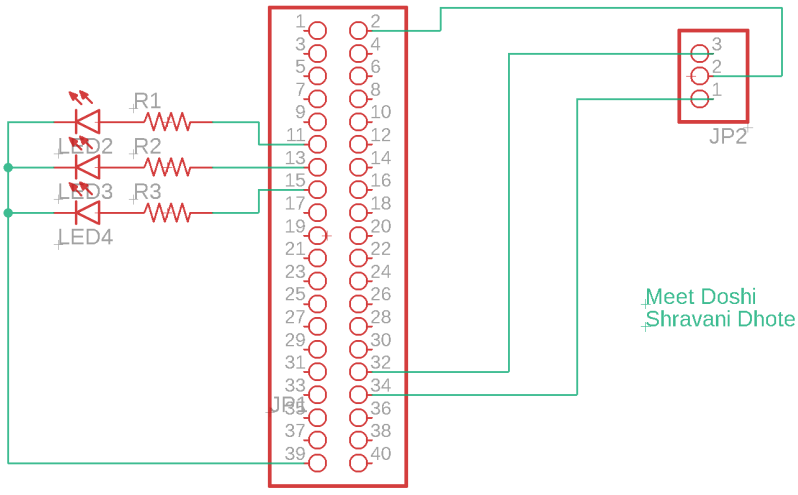


FIGURE 3.5: PCB Schematic

After the schematic is finalised, click on create board from the options bar. Now you need to place the wires and components physically on the virtual board so that you can finalise your PCB diagram.

- The board file is crucial to position the components on the PCB, this way we need to design the PCB such that it does not interfere with the on board Raspberry Pi camera. We decided to position the layout in the following manner.

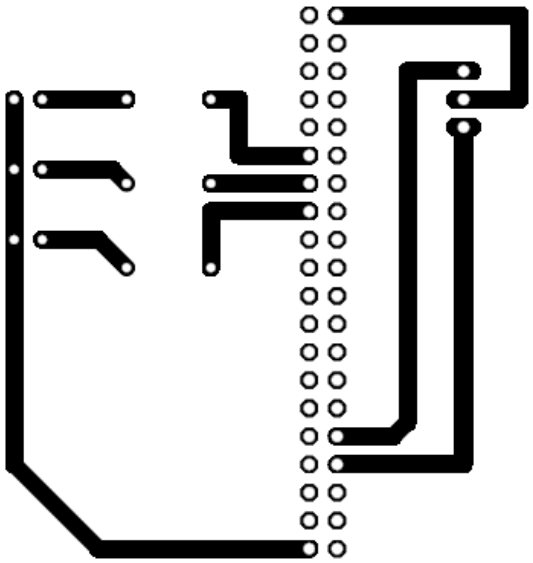


FIGURE 3.6: PCB Board
We have used track thickness as 60 to have plenty of room for soldering components. Now proceed to CopperCAM©to fabricate the design.

- CopperCAM©is a complex software and fabrication via 3D printer was done under the guidance of laboratory assistant Mr. Raghunath Patil. We generated the gerber files from the software and aligned the drill holes with the design. After that we have to just assign the job to the printer and it will do the rest of the fabrication process for us. Make sure to scrub the copper clad before putting it for fabrication. Check for tract damages or wrongly drilled holes.
- Now we move on to the soldering part. Avoid soldering the wire directly to the component, instead use the iron to heat up the legs and then make contact with the wire.

  Now all your hardware is ready and test it for working.

3.5.3 Testing

We performed two types of testing on the hardware.

I Apply high output to GPIO pins via GPIO.zero library available on python. Use a multimeter and check for any leakage current on the PCB and also test if any two ports are getting short circuited.

II Now after we verify the functioning of raspberry pi and PCB, we can now test our servo motor and LEDs. Apply high output to the GPIO pins to LEDs and check if they are glowing. Similarly check if our servo is working on the PWM pin.

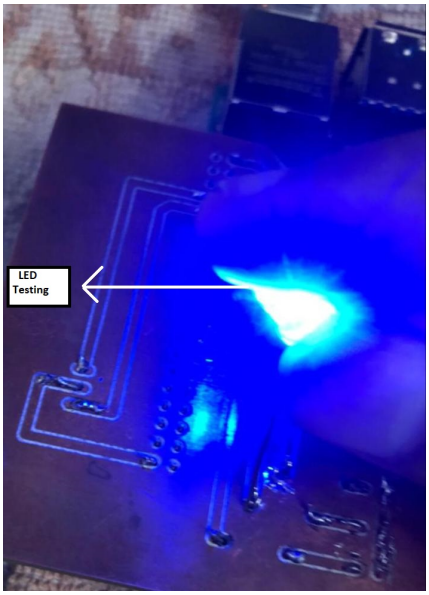


FIGURE 3.7: LED Testing

Both the above points are tested and we are now moving to work out the software.

3.6 Software

The code and all files are available on GitHub. Execute the following command to clone the files to your machine.

git clone git@github.com:meetdoshi90/Face-Mask-Detector.git

3.6.1 Flowchart

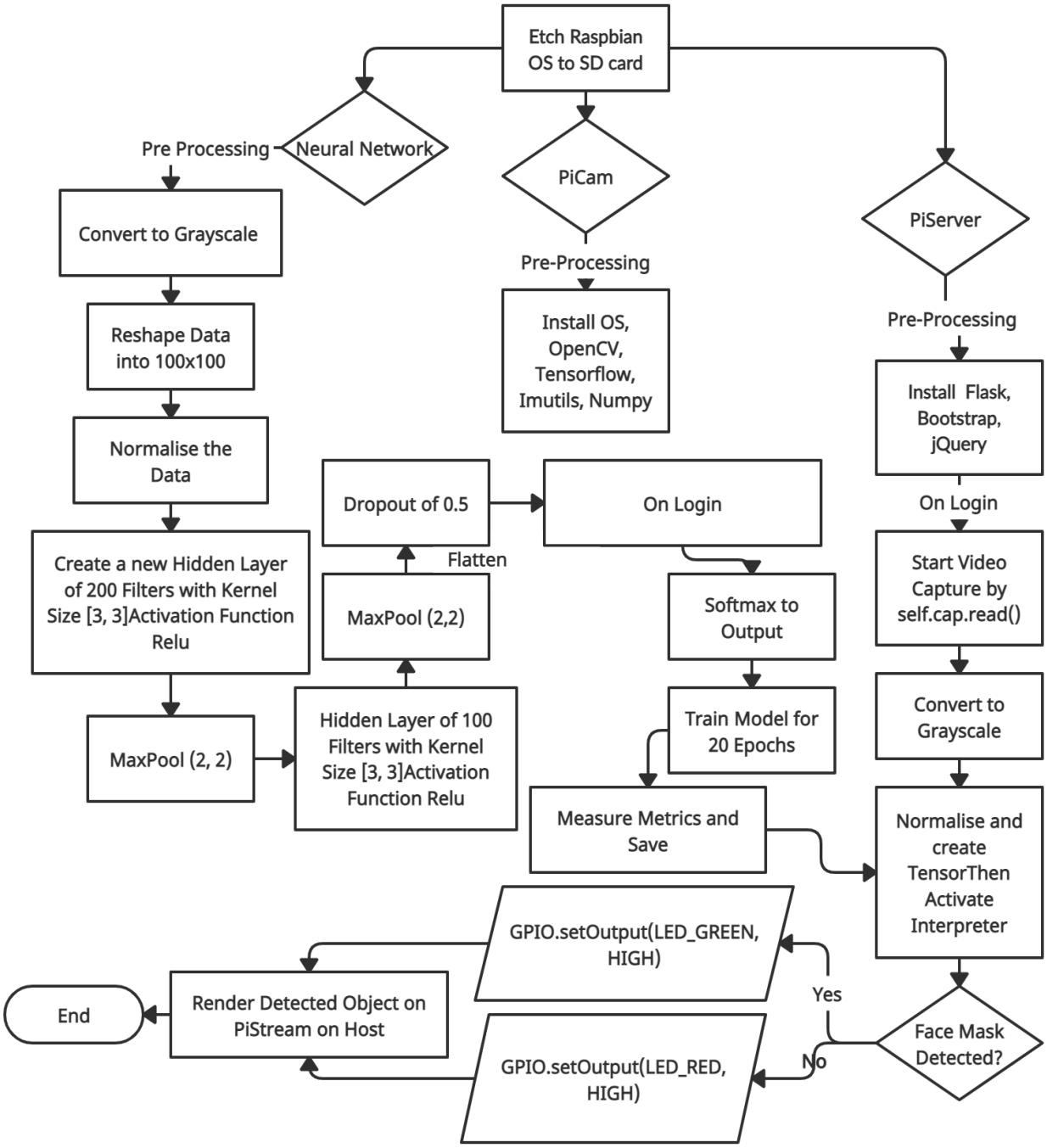


FIGURE 3.8: Software Flowchart

First download the Raspberry Pi OS on your local PC. Now use an etcher tool to etch the disk image file to your SD card. After loading the OS and boot program, connect your raspberry pi to a monitor via HDMI 0 and power up the Raspberry Pi. After the OS loads, run the following two commands on your terminal to upgrade your Raspberry Pi to latest updates.

sudo apt-get update

sudo apt-get upgrade

3.6.2 Neural Network

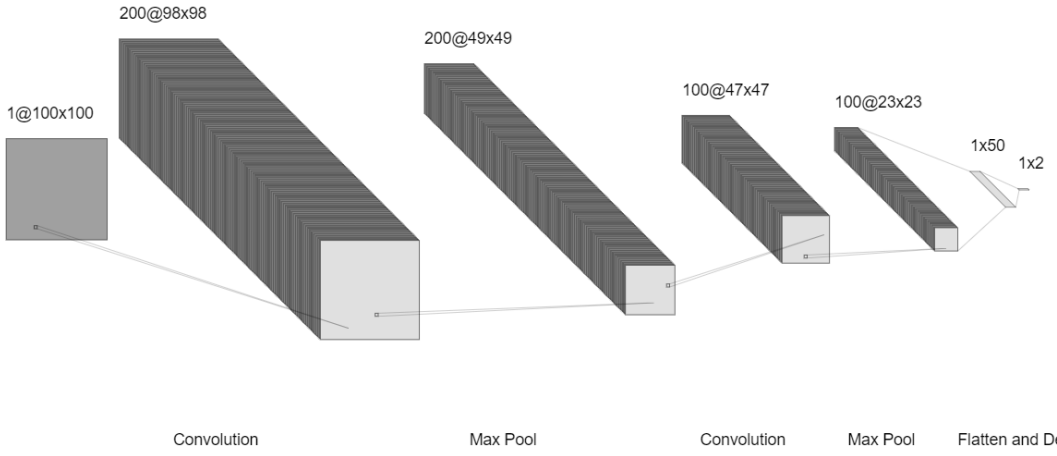


FIGURE 3.9: CNN
The Convolutional Neural Network designed here uses *categorical_crossentropy* loss function and *adam* optimizer for compilation. We also used pooling at each stage of the hidden layer to reduce computations and provide ease for object classification. At the point of flattening the layers we get about 52900 parameters. Output layer is densed to first layer of 50 parameters which in turn return two parameters at the end.

3.6.2.1 Pre-Processing

Parse each image and convert to grayscale via the OpenCV functions cvtColor. Resize the images to 100 x 100 pixel size.

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
resized = cv2.resize(gray,(img_size, img_size))
```

Normalise the dataset by dividing with a factor of 255. Again reshape it to flatten.

```
data = np.array(data)/255.0
data = np.reshape(data, (data.shape[0], img_size, img_size,1))
```

3.6.2.2 Creating the Layers

First import the Sequential class from keras and then create its object and store it inside a variable.

```
from keras.models import Sequential
model = Sequential()
```

Now we have a model with no layers. Adding the first layer with 200 filters and setting the kernel size to parse the image in a [3, 3] matrix. We chose the activation function ReLu because of its non-linearity, performance and easier training.

```
model.add(Conv2D(200, (3,3), input_shape = data.shape[1:]))
model.add(Activation('relu'))
```

After this layer we have about 1920800 parameters. Now we use max pooling to prevent overfitting and also to reduce the complexity of the network.

```
model.add(MaxPooling2D(pool_size=(2,2)))
```

This reduces the parameters to about 480200 after the first layer.

Similarly we add another Conv2D layer with 100 filters and similar kernel size. And after that we again down sample using the MaxPooling2D function. Now we have 2 layers and after the second layer we have about 52900 parameters. We now flatten it for the dense layer. We also use dropout factor of 0.5 to prevent overfitting on the model. Now moving to the dense output layer. We use 50 neurons on the first one and 2 neurons on final classification on the last. The activation function used for the last layer is softmax.

```
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(50, activation='relu'))
model.add(Dense(2, activation='softmax'))
```

Now we compile the model and then pass it for training.

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

We train the model for 20 epochs and monitor value of cost function for cross-validation data. The default learning rate for *adam* optimizer is 0.001. The results for the training and validation is shown in the results section.

3.6.3 Face Detection Algorithm

```
I   VideoStream().start()
II  def func get-mask:
III readCurrentFrame()
IV  convertColor(RGB to Gray)
V   model = trainedModel()
VI  interpreter = model.Interpreter()
VII normalised = resizedDetectedObject / 255.0
VIII reshaped = normalised.reshape(inputShape)
IX  interpreter.invoke(inputDetails, reshaped)
X   result = interpreter.Output()
XI  if result == maskDetected:
XII         output(Mask Detected , GPIO.HIGH)
XIII else:
XIV         output(No Mask, GPIO.LOW)
```

3.6.4 Hosting A Remote Server

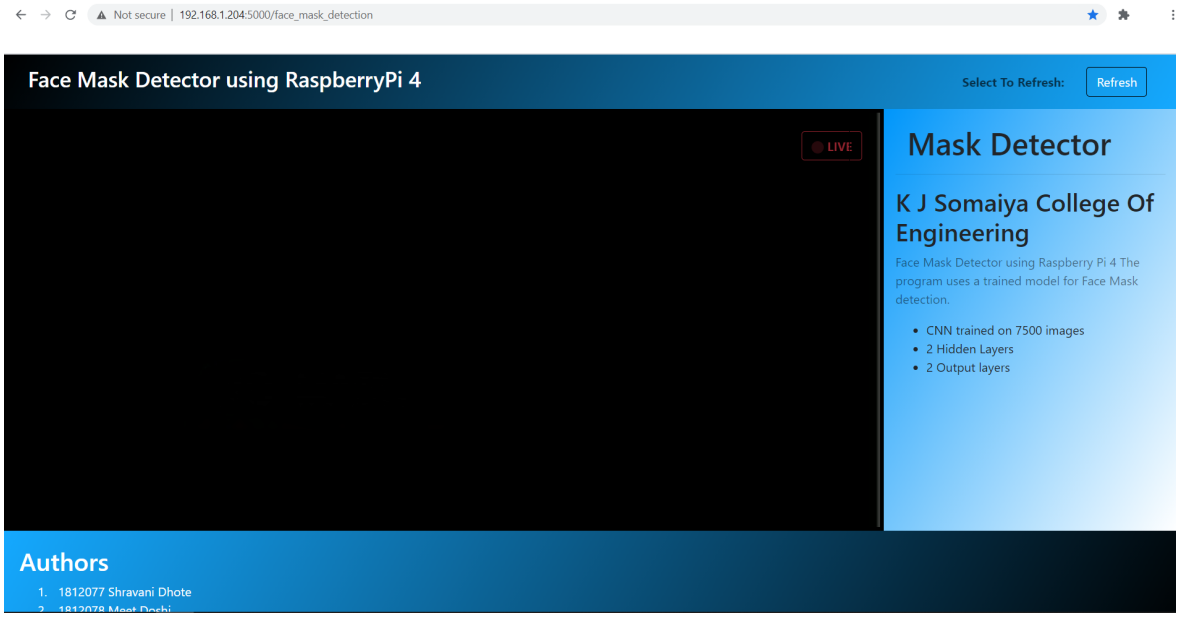


FIGURE 3.10: Website Template
Standard static HTML website with one nav-bar, one body, one footer. JavaScript to invoke refresh button, CSS for styling, live video stream from Pi Camera fed to the body.

To host the server on raspberry pi we used Flask which is a framework for hosting website in python. Showing how to use Flask as a web server is beyond the scope of this report. To run the server just run the following commands of the Raspberry Pi terminal

- cd rpi_based_mask_detector_new
- flask run –host=0.0.0.0

After which your host has started running. Now currently to access the server you need raspberry pi's ip address to make an HTTP request to server. For that you can look into pi's ip address by typing the command in the terminal:

- sudo apt-get install net-tools
- ifconfig

Now connect to the following URL *http://192.168.1.204:5000/face_mask_detection*, now you should see your live mask detector running For login authorization:

Username = password

Password = username

3.6.5 Data Flow Diagram

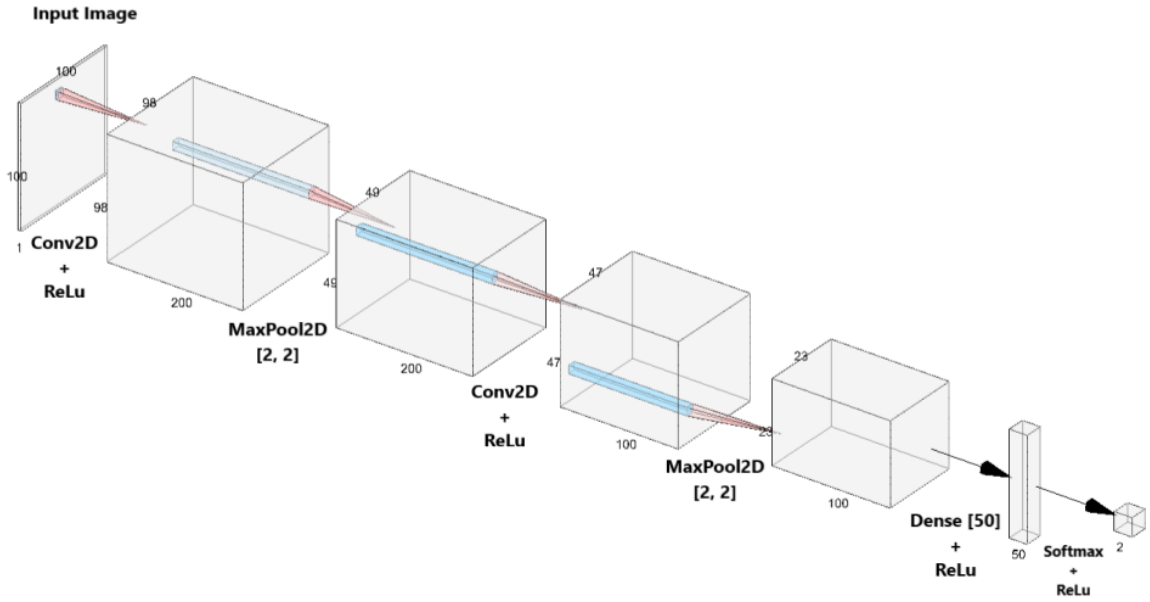


FIGURE 3.11: Data Flow: CNN
With this figure we can see the selection of kernel size and also the parameters of each layer in the network.

After pre-processing the data we will have a input of [1 x 100 x 100] pixel image at the input layer which will then be passed to our first convolutional layer. After our first convolutional layer the data will be in the form of a 3D matrix of shape [200 x 98 x 98]. We used pooling to reduce the number of data points passing through the network increasing computations. After the pooling we will have a down sampled matrix of shape [200 x 49 x 49]. This ends our first hidden layer. We preferred having more filters for our network to avoid biasing of a layer to a particular pattern. We repeat the convolutional layer and pooling to get a shape of [100 x 23 x 23] matrix. Now we inserted a dense layer as an intermediate output layer, but for using dense layers we need to flatten out the matrix. We also use 50% dropout to add non linearity

to the model. After then first dense layer we have the possible combinations to output layer of 50 * 52900 * 0.5 *(dropout rate)* = 1322500, input to the last layer is [50] which then has 50 * 2 = 100 possible combinations. Out of which we take argmax of the output layer to determine which probability is the highest.

3.6.6 Testing

For testing the model we just take a validation split of 20% from the training data and feed it to the model for learning. Here are some of the validation loss and validation accuracy metrics over 20 epochs.

```
31/31 [==============================] - 124s 4s/step - loss: 0.0846 - accuracy: 0.9747 - val_loss: 0.1492 - val_accuracy: 0.9435
Epoch 10/20
31/31 [==============================] - 126s 4s/step - loss: 0.0743 - accuracy: 0.9768 - val_loss: 0.1640 - val_accuracy: 0.9435
Epoch 11/20
31/31 [==============================] - 118s 4s/step - loss: 0.0557 - accuracy: 0.9778 - val_loss: 0.1533 - val_accuracy: 0.9476
Epoch 12/20
31/31 [==============================] - 120s 4s/step - loss: 0.0471 - accuracy: 0.9828 - val_loss: 0.1524 - val_accuracy: 0.9435
Epoch 13/20
31/31 [==============================] - 119s 4s/step - loss: 0.0596 - accuracy: 0.9818 - val_loss: 0.2244 - val_accuracy: 0.9073
Epoch 14/20
31/31 [==============================] - 120s 4s/step - loss: 0.0486 - accuracy: 0.9818 - val_loss: 0.1922 - val_accuracy: 0.9274
Epoch 15/20
31/31 [==============================] - ETA: 0s - loss: 0.0284 - accuracy: 0.9909INFO:tensorflow:Assets written to: model-015.model/asse
ts
31/31 [==============================] - 123s 4s/step - loss: 0.0284 - accuracy: 0.9909 - val_loss: 0.1421 - val_accuracy: 0.9516
Epoch 16/20
31/31 [==============================] - 119s 4s/step - loss: 0.0352 - accuracy: 0.9859 - val_loss: 0.1856 - val_accuracy: 0.9315
Epoch 17/20
31/31 [==============================] - 136s 4s/step - loss: 0.0345 - accuracy: 0.9889 - val_loss: 0.1776 - val_accuracy: 0.9355
Epoch 18/20
31/31 [==============================] - 117s 4s/step - loss: 0.0465 - accuracy: 0.9828 - val_loss: 0.1665 - val_accuracy: 0.9395
Epoch 19/20
31/31 [==============================] - 117s 4s/step - loss: 0.0556 - accuracy: 0.9798 - val_loss: 0.2122 - val_accuracy: 0.9194
Epoch 20/20
31/31 [==============================] - 122s 4s/step - loss: 0.0736 - accuracy: 0.9717 - val_loss: 0.1952 - val_accuracy: 0.9315
```

(a) Validation Loss and Accuracy over last 10 epochs

```
print(model.evaluate(test_data, test_target))
```

```
5/5 [==============================] - 3s 670ms/step - loss: 0.1302 - accuracy: 0.9493
[0.1302407681941986, 0.9492753744125366]
```

(b) Final evaluation over training model

CHAPTER 4

Results and Discussions

Although no design is ever perfect, this design is very powerful given its efficiency to cost ratio. We now discuss the faults this project has and what advantage it has over the other ones discussed in the literature review section. But before that we have to present the results using viable metrics. In this section we go through all the testing results initial and final, for neural network we show the trained and test model accuracy and loss function plot using *matplotlib*. We also go through the testing prediction confusion matrix and plot its results using the *sklearn* and *seaborn* libraries.

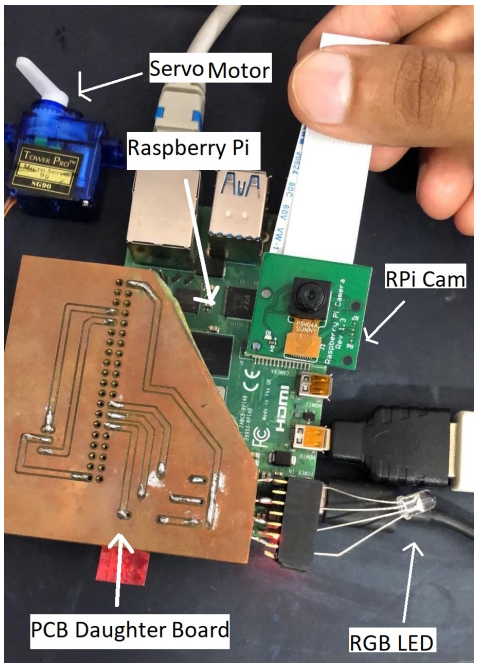
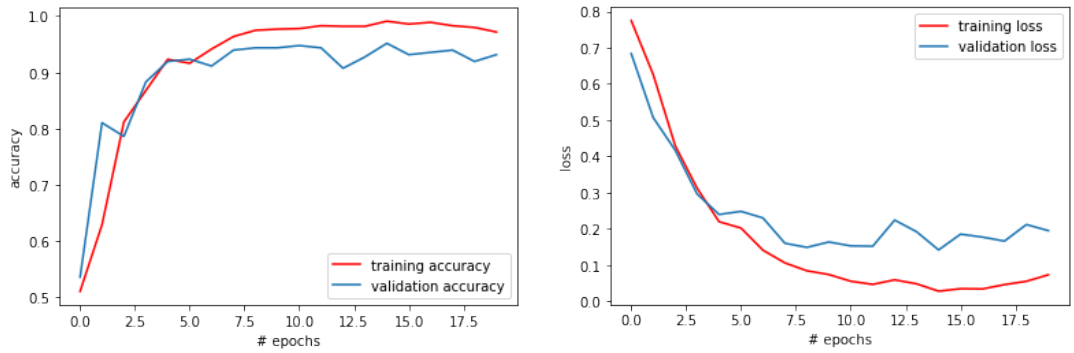


FIGURE 4.1: Final Project Look
An image taken after assembling all components and soldering them. An Ethernet cable is used for network and a VGA monitor is connected to HDMI port 0. RPi cam has also been fixed to its in built slot.

4.1 Trained Model Accuracy



(a) Accuracy over training model

(b) Loss over training model

FIGURE 4.2: Plot of Loss and Cost
Trained model loss and accuracy on training set over 20 epochs and validation set distributed in the ratio of 80:20

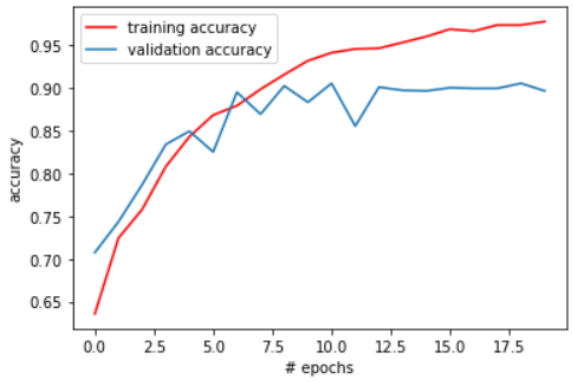
From the plot we can see that our model does not over fit and maintains its accuracy even after numerous epochs. The training accuracy and validation accuracy curves are almost identical and show that our model will work very well on unseen images.

Looking at the loss curve, as we have used *categorical_crossentropy* as our loss function which takes the form:

$$CCE(p,t) = -\sum_{c=1}^{C} t_{o,c} \log\left(p_{o,c}\right)$$

This loss function performed well over *MSE* and *Hinge Loss*. We can see that the loss function has not yet received global minimum but it has started to swing over minimas. Yet it performs well for real life scenarios.

4.2 Test Model Accuracy



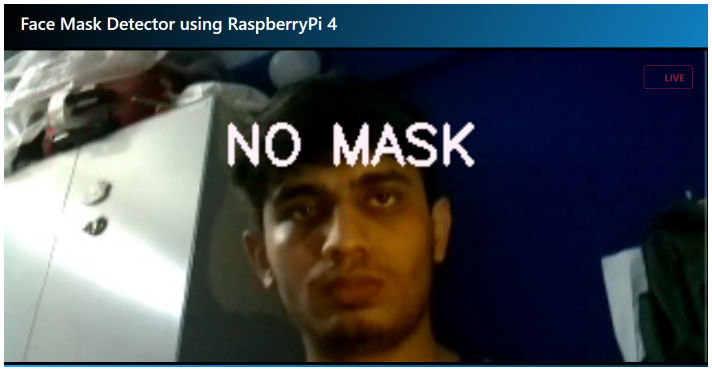

```
print(model.evaluate(test_data, test_target))
```

```
24/24 [==============================] - 7s 303ms/step - loss: 0.4002 - accuracy: 0.8677
[0.40023109316825867, 0.8677248954772949]
```

FIGURE 4.3: Model Evaluation over Testing Data
After training the model over around 4500 images, we see that at the time of testing on unseen data the model has and accuracy of 86% and we can see overfitting.

To have better testing results we recommend a good data set and plenty of training for the model. We realise that its accuracy can be improved by downsampling the images and shuffling the dataset. Other methods to increase accuracy would be to increase the number of filters by around 25% and adding another convolutional layer to the hidden layers.

4.3 Live detection Accuracy



(a) No Mask

(b) With Mask Example 1

(c) With Mask Example 2

(d) With Mask Example 3

FIGURE 4.4: Results of Mask Detection from a remote host server and output shown to a remote computer.

4.4 Confusion Matrix

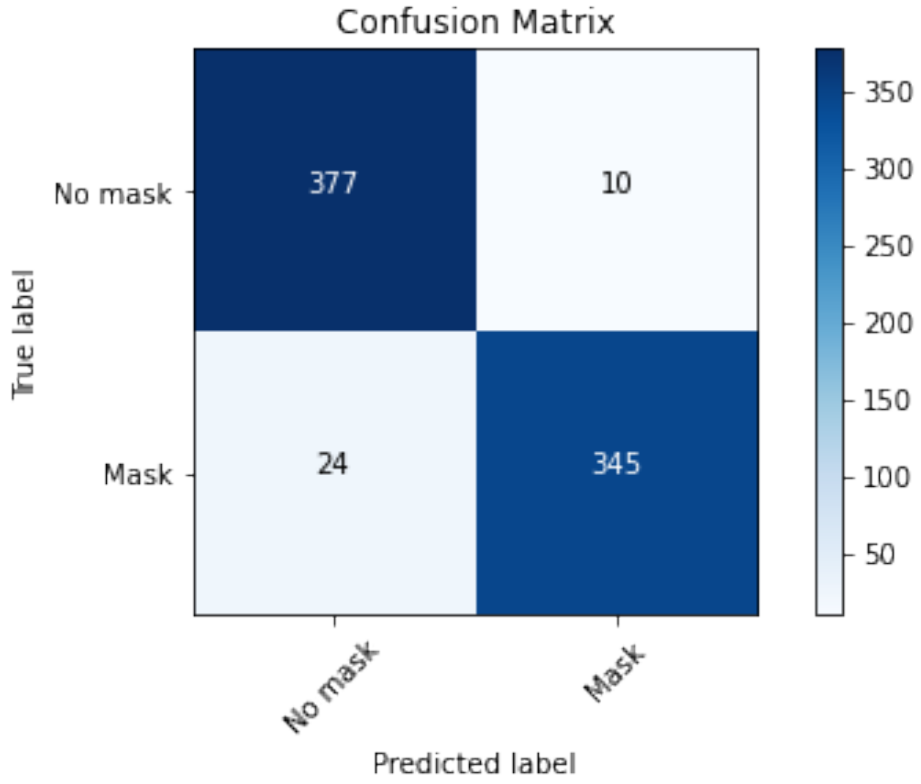
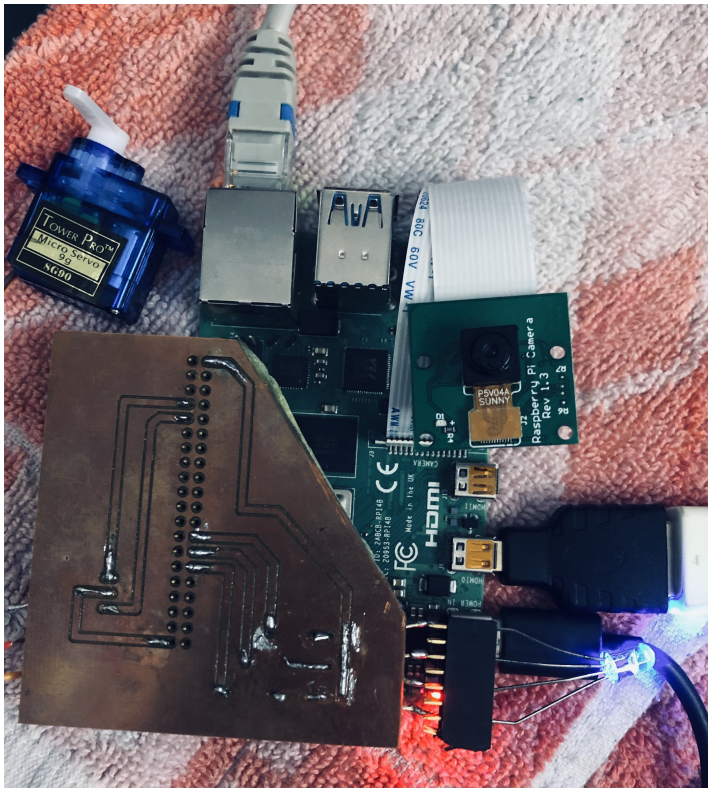


FIGURE 4.5: Confusion Matrix over the tested data
This is the confusion matrix over the model for 756 test images. We can see that we have about 24 false negatives and 10 false positives. Which comes out to be around 95.5% accuracy rate which is up to the mark.

4.5 Hardware Results

The hardware results were tested on how fast the servos and LEDs react to detection of an image. In the program we had inserted a sleep time of 1 sec after every time the model was instantiated therefore the servos and LEDs had more than plenty time to show the output. We came to notice that hardware output was extremely faster as compared to software rendering.

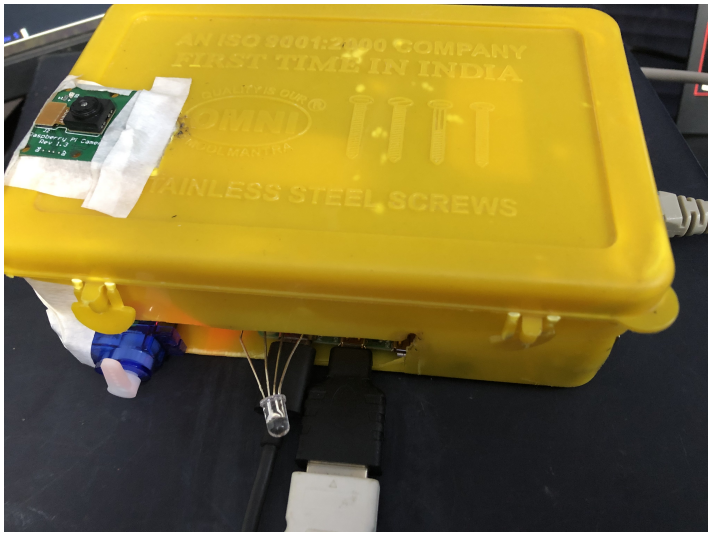


(a) Output LED Blue ON Servo at 0°: No Mask Detected

(b) Output LED Green ON Servo at 180°: Mask Detected

FIGURE 4.6: Results of Mask Detected and Mask Not Detected.

4.6 Enclosure Design



(a) Left Top View of Enclosure

(b) Bottom Top View of Enclosure

FIGURE 4.7: Enclosure Design.

4.7 Discussions

Paper	Author	*Method*	*Accuracy (%)*
1	Wuttichai *et al.*[1]	*SVM*	*78.1*
		KNN	*72.5*
		MobileNet	*88.7*
2	Sanjaya *et al.*[2]	*MobileNetV2*	*92*
3	Rahman *et al.*[3]	*CNN*	*98.7*
4	Khandelwal *et al.*[4]	*MobileNetV2*	*97*
5	Jiang *et al.*[5]	*MobileNet*	*82.3*
		ResNet	*93.4*
6	Nagrath *et al.*[6]	*SSDMNV2, MobilenetV2*	*94*
7	Ejaz *et al.*[7]	*MTCNN*	*98.5*
8	Baluprithviraj *et al.*[8]	*CNN*	-
9	Current Method on Testing Set	*CNN*	*86*

TABLE 4.1: Comparing Current Method with Reffered Works

Although our model is only consisted of four layers in the enite network, the model gives around 95% accuracy on validation set and about 86% on testing set but on live feed it dropped below the expected mark. The accuracy reduced due to latency and other features which the model did not expect, came out to be around 60% on live feed from the RPi. We think the reason for this huge dip in the accuracy was due to dataset being made up of images from people around the globe, their clothing, mask types, flipped, sheared images. This could be improved by training the model with images on which the model is to be tested upon. We aimed to make model faster by using less number of layers and introducing more non-linearity to the network. To achieve a faster network we made a compromise on the number of hidden layers which resulted in lower accuracy. If we use a deeper and better model such as MobileNetV2, Darknet-53, YOLO, etc we can give up speed and achieve accuracy on Raspberry Pi.

4.7.1 Suggestions

Below we list some suggestions and tips which we think can bring some improvements to the current project.

I **Using a better Raspberry Pi**

Having a faster hardware can always give computational advantage over slower hardware

II **Using a better RPi cam**

Although we used a 5 Megapixel camera we could have had improved performance by using a newer and faster version of the Raspberry Pi camera.

III **Improving the training**

Even though we have used over 7500 images for this model, such purpose can benefit more from larger and better datasets. By better we mean more realistic and similar to the environment of deployment.

IV **Not rendering the video feed**

We came to a conclusion that even if we use better hardware and better code base, we still suffer from the problem of network speed limitations. For rendering the live video feed embedded inside a static html webpage still takes a lot of load over the device and gives slower output buffer.

V **Improving and increasing the hidden convolutional layers**

We saw that using more number of filters and a slightly lower kernel size gives better performance in throughput than a dense model with thrice the layers. CNN's are a black box for large computations and must be designed carefully as the throughput can increase significantly if a better model is designed.

VI **Using Multi-class / Multi Object Detection** This design is flawed in a way that it can only detect the output with good accuracy if only there is one person in the view of camera and that persons frontal face is visible clearly. Although this is a flaw, but it is also a feature for our design as it can be used at places where a queued or restricted audience is allowed. This model would be able to detect multiple objects if we use an object detection algorithm beforehand. A recommended algorithm is YOLO *et al.*[9]

CHAPTER 5
Conclusions and Future Scope

5.1 Conclusions

Significant organizations from diverse sectors are turning to Artificial Intelligence and Machine Learning, leveraging technology at the service of humanity amid the pandemic. To curb the spread of the coronavirus, measures must be taken.

In this project, we introduced a face mask detection system using a Raspberry Pi that grants access depending on whether the person has worn a mask or not. We used a Servo motor and an RGB LED as an indicator of the same. We modelled it using a Convolutional Neural Network which is simple enough and computationally efficient. To train, validate and test the model, we used a dataset that consisted of 3725 masked faces images and 3828 unmasked faces images. These images were taken from various sources like Kaggle. The model was inferred on images and live video streams. We designed this project keeping in mind the benefit of a remote server. Using this design we can view the live video stream from any device globally. This system has an accuracy of over 95% on validation set, 86% on testing data and about 60% on live RPi feed.

This study can assist authorities to use more unstructured data resources for more data-based mitigation, prevention, evaluation, and action planning against COVID-19. It can be used in crowded places like schools, colleges, bus stops, malls, etc to grant access only to those who wear a mask. We hope that this would be a useful tool in preventing virus transmission.

5.2 Future outlook

We can improvise this idea by adding few features like alerting the person near to the one not wearing a mask by buzzer, and displaying the violators face on the monitor so that he can maintain a safe distance. Also, identification of the type of mask-like surgical face masks, cloth masks, N95 etc worn by the person could be another feature. Instead of a biometric attendance system, facial recognition can be used as contactless attendance in offices, colleges, factory plants etc to further curb the spread of COVID-19.

As the performance of AI is increasing in today's world, we have seen newly designed algorithms that can learn from millions of gigabytes of data and learn to find so many patterns inside unlabelled data. AlphaGO one such program newly acquired by google's AI company designed to master a board game name Go which has about 10^{360} possible moves, literally a google times more possibilities than the game of Chess.

If such a program can master with expertise then the possibilities for AI are endless. At this rate, we can only imagine that so much data is generated every second over the internet which can be used by algorithms to parse through a tremendous amount of raw data and make sense out of it to make even better programs that are undoubtedly billion times faster than a simple modern day computer. Thus, the performance and accuracy of the proposed method for face mask detection can be improved.

Acknowledgements

We would like to express our sincere gratitude to our project guide Dr. Ninad Mehendale for providing his valuable help, guidance, comments, wholehearted cooperation, practical suggestions, helpful advice and motivating us to work harder throughout the course of the project. We would specially thank Mr. Raghunath Patil for helping us in fabricating the PCB.

We are also grateful to respected Dr. Jagannath H Nirmal (HOD ETRX) and to our respected Principal Dr. Shubha Pandit for permitting us to utilize all the necessary facilities of the institution. We are also thankful to all the other faculty and staff members of the Electronics department for their kind co-operation and help.

During the course, we faced many challenges due to our lack of knowledge and experience but these people helped us in getting over them and in the final compilation of our idea to a shaped sculpture. We are thankful to them for the encouragement they have given us in completing the project. We would also like to thank all the shopkeepers who provided us with the necessary components and materials required to build the project.

Lastly, we would like to express our deep appreciation towards our classmates and our parents who have patiently extended all sorts of help and provided us with moral support and encouragement.

References

[1] W. Vijitkunsawat and P. Chantngarm, "Study of the Performance of Machine Learning Algorithms for Face Mask Detection," *2020 - 5th International Conference on Information Technology (InCIT)*, Chonburi, Thailand, 2020, pp. 39-43, doi: 10.1109/InCIT50588.2020.9310963.

[2] S. A. Sanjaya and S. Adi Rakhmawan, "Face Mask Detection Using MobileNetV2 in The Era of COVID-19 Pandemic," *2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI)*, Sakheer, Bahrain, 2020, pp. 1-5, doi: 10.1109/ICDABI51230.2020.9325631.

[3] M. M. Rahman, M. M. H. Manik, M. M. Islam, S. Mahmud and J. -H. Kim, "An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network," *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, Vancouver, BC, Canada, 2020, pp. 1-5, doi: 10.1109/IEMTRONICS51293.2020.9216386.

[4] Prateek Khandelwal, , Anuj Khandelwal, Snigdha Agarwal, Deep Thomas, Naveen Xavier, and Arun Raghuraman. "*Using Computer Vision to enhance Safety of Workforce in Manufacturing in a Post COVID World." (2020).*

[5] Jiang, Mingjie, et al. "RetinaMask: A Face Mask Detector." *ArXiv:2005.03950 [Cs]*, June 2020. arXiv.org, http://arxiv.org/abs/2005.03950.

[6] Nagrath P, Jain R, Madan A, Arora R, Kataria P, Hemanth J. "SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2". *Sustain Cities Soc. 2021*;66:102692. doi:10.1016/j.scs.2020.102692

[7] Md.Sabbir Ejaz and Md.Rabiul Islam, "Masked Face Recognition using Convolutional Neural Network", *2019 International Conference on Sustainable Technologies for Industry 4.0(STI)*, Dhaka, Bangladesh, 2019,pp. 1-6, doi: 10.1109/STI47673.2019.9068044.

[8] K. N. Baluprithviraj, K. R. Bharathi, S. Chendhuran and P. Lokeshwaran, "Artificial Intelligence based Smart Door with Face Mask Detection," *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, Coimbatore, India, 2021, pp. 543-548, doi: 10.1109/ICAIS50930.2021.939580

[9] Larxel. "Face Mask Detection." Kaggle, Accessed May 22, 2020. https://www.kaggle.com/andrewmvd/face-mask-detection.

[10] “BCM2711.” BCM2711 - Raspberry Pi Documentation. Accessed May 2, 2021. https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/.