Embedded Linux

Course Project Report

on

Face Mask detection on Raspberry Pi

By:

1. **Prajwal M Patang** USN:01FE18BEC106
2. **Pratiksha Naik** USN:01FE18BEC114
3. **Pratyusha Katti** USN:01FE18BEC115

Semester: VII

Year: 2021

Under the Guidance of

Basawaraj Patil

Prabha Nissimgoudar

ACKNOWLEDGMENT

The feeling of fulfilment that lies within the completion of this project would be unfinished without citing the names of the individuals who made a difference in completing this project as their consistent direction, support and encouragement.
We would like to express our admiration to all those who provided us the possibility to complete this project. We would sincerely like to thank Basawaraj Patil and Prabha Nissimgoudar for their active support and helpful insights while building this project. We offer our most sincere appreciation for the learning opportunities provided to us. We would also like to thank the School of Electronics and Communication Engineering, KLETECH for giving us this great learning opportunity.

-Prajwal, Pratiksha, Pratyusha.

ABSTRACT

In this project, we try to design deep neural network towards Face Mask Detection, a network that tells us whether a person is wearing a mask or not. In the past two years, the pandemic has forced us to wear masks. We need to be masked to enter any public place like universities, shopping malls, etc. An automated mask detection system would be helpful in monitoring the crowd at all times whether there are any defaulters. Deep Neural Networks provide efficient way to design our methodology using its key processes. This can be applicable in crowded places like shopping malls, movie theatres etc.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

- The best prevention at times of a pandemic is to utilize facial coverings. Wearing masks is one such way to reduce the transmission of infections from breathing, speaking, coughing, sneezing, transmission.
- Monitoring large groups of people in public areas who comply with rules set by the governments has become a challenge in the past two years. With machine learning algorithms, building a face mask detection system would assist the automation task and help mitigate the risk.

1.2 Objectives

- Develop an algorithm to perform mask detection.
- Detect masks for multiple people.
- Signal whether mask found or not found.

1.3 Problem statement

Design a methodology to detect whether a person is masked or not.

1.4 Application in Societal Context

- Corporate sector: Face mask detection can be used to maintain a safe environment for everyone. The mask detection system can send alerts or reminders to those not complying.
- Airports: The Face Mask Detection System can be used at airports to detect travelers without masks. Face mask detector can be very effectively used at airports mainly for entrance flow management and monitoring.

- Public Transport: In the view of pandemic, public transport will be needing to use a software to automate the checking process with very little resources needed and with efficiency.

1.5 Organization of the report

Chapter 2 describes the system design with functional block diagram and design alternatives.
Chapter 3 describes the implementation details of the project.
Chapter 4 describes the results obtained in the project.
Chapter 5 describes the conclusion drawn from the project implementation and the future scope is discussed.

Chapter 2

System design

In this chapter, we explore different possibilities to design our framework in the software phase and implement best suited methodology on hardware.

2.1 Functional block diagram

Figure 2.1: Block diagram of face mask detection system

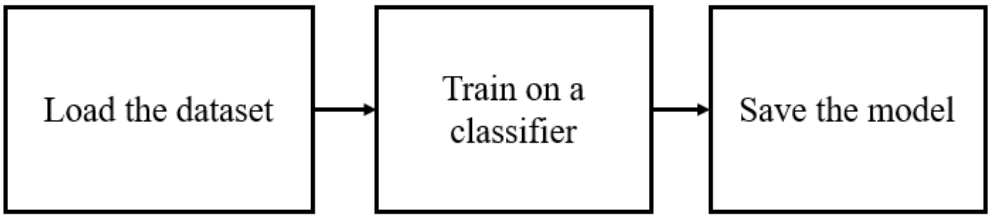


Figure 2.2: Block diagram of training phase

The pre-processing included in training stage consists of obtaining facial features like eyes, mouth, nose, etc which assist in Region of Interest(ROI) extraction. The next step would be overlaying an image of mask onto the cropped image to get masked-images.

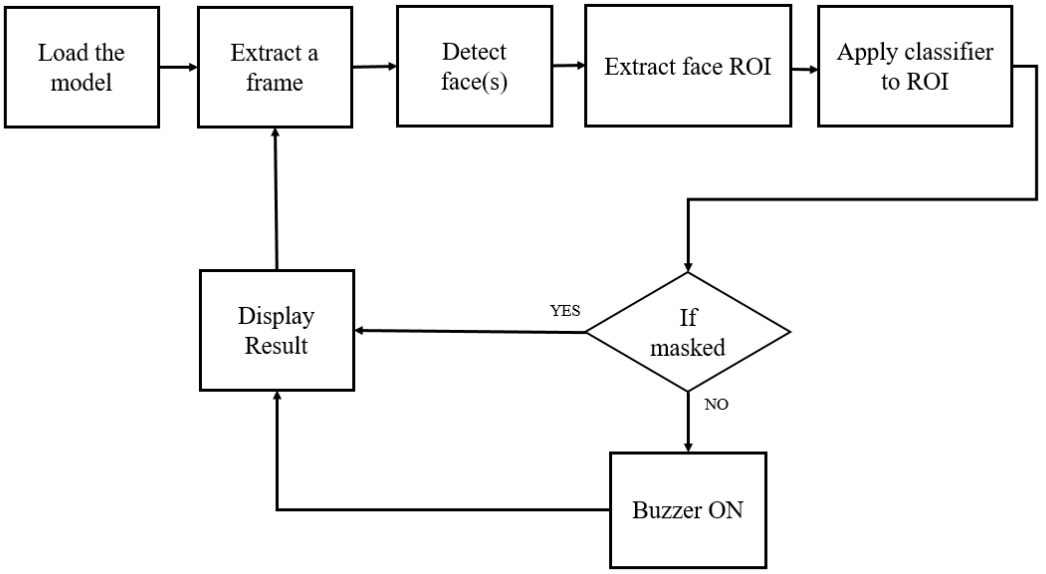


Figure 2.3: Block diagram of testing phase

A real-time video stream is taken as input. Each frame goes through a process of detecting faces, obtaining Region of Interest (ROI), classifying if the person is masked or un-masked. A piezo buzzer is integrated which turns ON when an image of a person with no mask is detected. The accuracy of presence of mask is displayed along with detected bounding box.

2.2 Classifier architecture

MobileNetV2, developed by Google has two types of blocks. One is residual block with stride of 1. Another one is block with stride of 2 for downsizing. There are 3 layers for both types of blocks. This time, the first layer is 1×1 convolution with ReLU6. The second layer is the depthwise convolution. The third layer is another 1×1 convolution but without any non-linearity. It is claimed that if ReLU is used again, the deep networks only have the power of a linear classifier on the non-zero volume part of the output domain.

And there is an expansion factor t. And t=6 for all main experiments. If the input got 64 channels, the internal output would get 64×t=64×6=384 channels. Adding Linear bottlenecks between the layers prevents non-linearities from destroying too much information.

MobileNetV2 improves the state-of-the-art performance of mobile models on multiple tasks and benchmarks. It is a very effective feature extractor for object detection and segmentation. It is a highly effective feature extractor for object detection and segmentation.

here t: expansion factor, c: number of output channels, n: repeating number, s: stride. 3×3 kernels are used for spatial convolution.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 2.4: MobileNetV2 : Each line describes a sequence of 1 or more identical layers repeated n times

Chapter 3

Implementation details

In this chapter, we discuss the specifications and final system architecture with its algorithm and flowchart.
MobileNetV2 architecture is computationally efficient, thus making it easier to deploy the model to the selected embedded system that is Raspberry Pi 4.

3.1 Software

3.1.1 Dataset

Real World Masked Face Dataset The dataset contains 1376 images collected from the Real-world Masked Face Detection (RMFD) dataset. They are divided into two classes:

1. with-mask: 690 images
2. without-mask: 686 images

Out of 1376, 80% which equals to 1100 images were used for training. And 20% that is, 276 images were considered for validation. The model is trained for 20 epochs on this data-set.

3.1.2 Libraries

OpenCV to visualize the results.
Keras/TensorFlow to build and validate model.

3.2 Hardware

The entire hardware is implemented using a Raspberry Pi 4 along with a 5MP camera and a piezo speaker on breadboard.

3.2.1 Raspberry Pi

Raspberry Pi is a series of small single-board computers. The Raspberry Pi 4 model B was used on this project. It has a 1.5 GHz 64-bit quad-core on board 802.11ac Wi-Fi and Bluetooth 5. It is powered by a USB-C port.
A 16 GB Memory Card for the OS and the model.
Setting up Raspberry Pi:

1. Installing Raspberry Pi OS on microSD card.
2. Enabling SSH to to remotely access the Pi command line.
3. Downloading and installing putty to establish SSH connection.
4. Enabling over VNC using sudo raspi-config command.
5. Running VNC viewer to add new connection.
6. Open the newly added connection and filling set username and password to view the Raspberry Pi desktop on our screen.

3.2.2 Integrating camera and buzzer

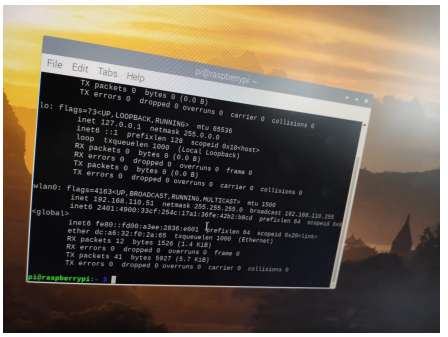
Buzzer:

1. Importing the GPIO and sleep libraries.
2. Initializing pin 23 as an output pin with GPIO.setup( ) function.
3. Beep sound with GPIO.output( ) function.
4. Pause the beep for 0.23 second with sleep( ) function.

Camera:

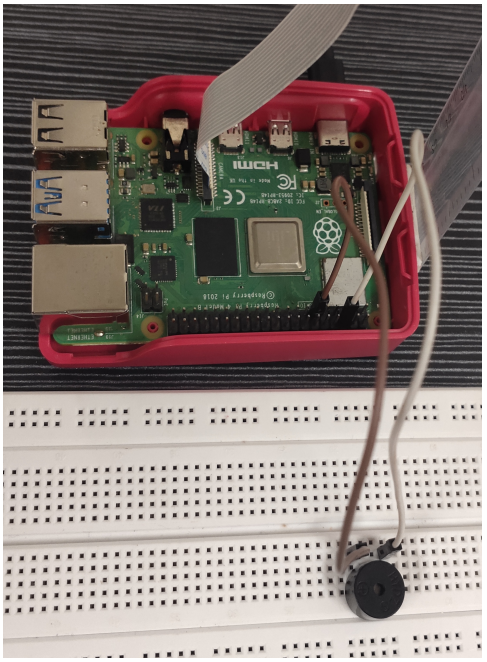
1. Ensuring that the camera is enabled by selecting Interfaces tab in Raspberry Pi Configuration.
2. Importing picamera library to integrate camera module.
3. capture() function is used to get a frame from real-time video which is then fed to the classifier.

(a) Setting up the board.

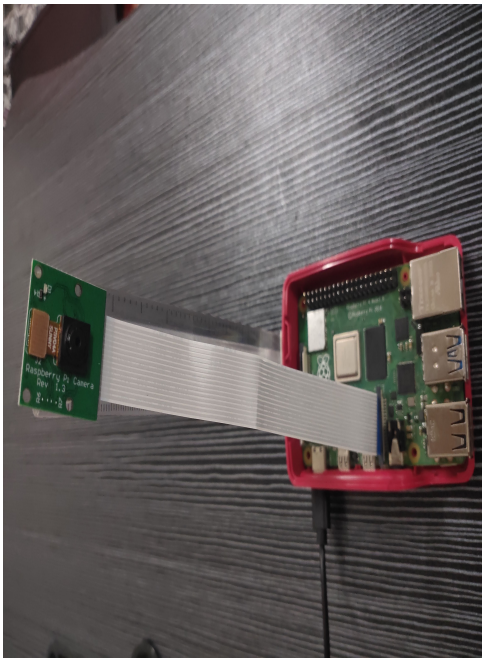


(b) Establishing connection

Figure 3.1: Initial setup



(a) Buzzer



(b) Picamera

Figure 3.2: Peripherals

3.3 Algorithm

Algorithm 1 mask detection

Input: Video stream
Output: Frames with detections

1. **for each frame**
2. Detect face.
3. Predict if the person is wearing mask or not.
4. Display the presence of mask in %
5. If person is not wearing mask:
6. Switch ON the buzzer.
7. Display the results.
8. Continue to next frame.
9. **end for**

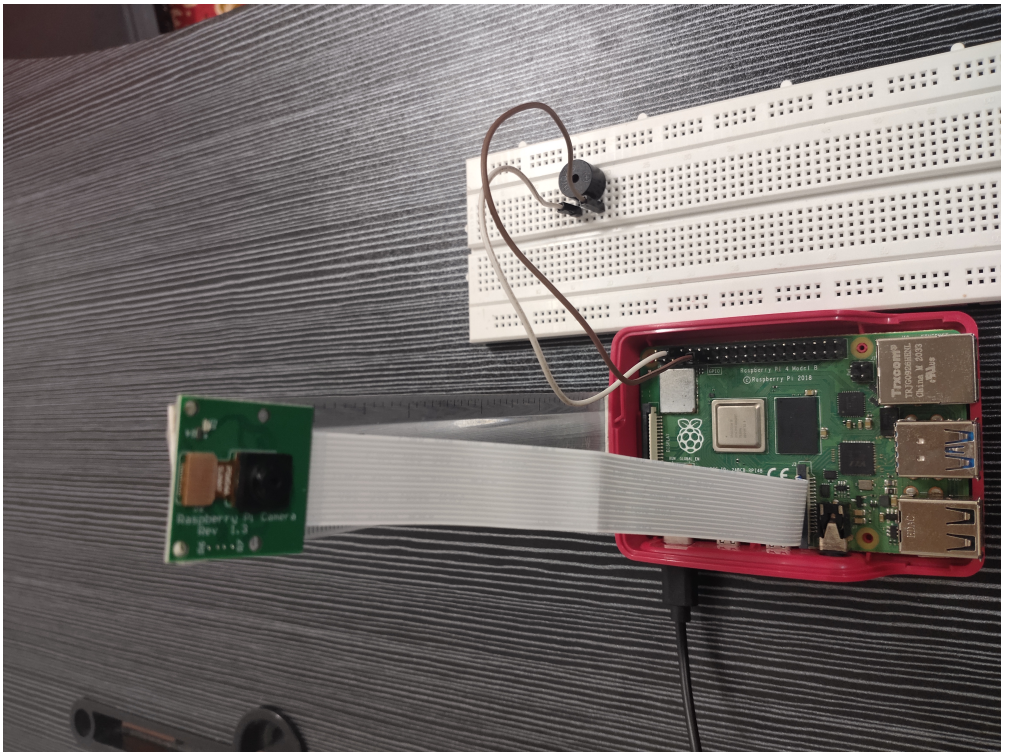


Figure 3.3: All modules integrated

Chapter 4

Results and discussions

4.1 Result Analysis

Precision is:

$$\frac{Tp}{(Tp + Fp)}$$

where Tp is the number of true positives and Fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

Recall is;

$$\frac{Tp}{(Tp + Fn)}$$

where Tp is the number of true positives and Fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0.

Studying all these evaluation metrics and performing the experiment, we arrived at the following results:

Image type	**Precision**	**Recall**	**f1-score**	**Accuracy**
with mask	0.99	1.00	0.99	0.99
without mask	1.00	0.99	0.99	0.99

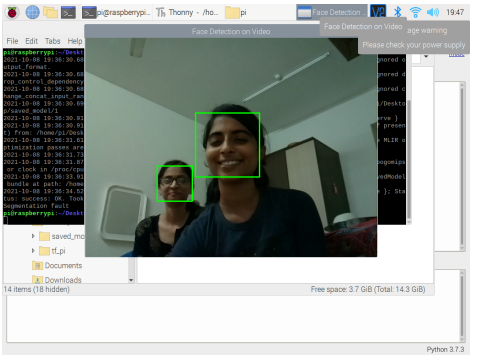


Figure 4.1: Face Detection

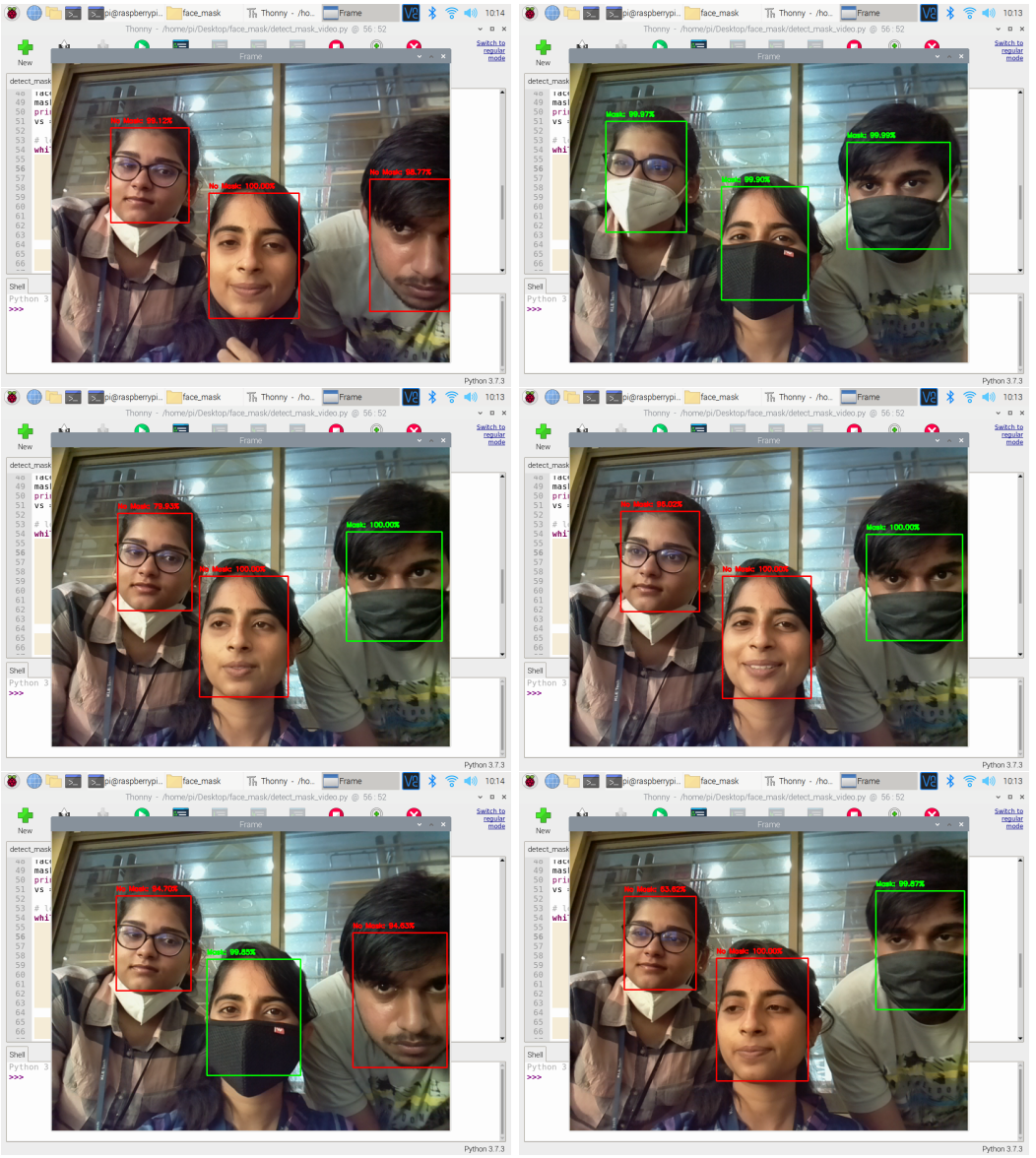


Figure 4.2: Results

Chapter 5

Conclusions and future scope

In this chapter, we brief our conclusion and future scope for mask detection.

5.1 Conclusion

The main focus of the team was to understand the working and use of embedded Linux system and implement a machine learning model on such a computer. We were able to acquire the fundamental knowledge of Raspberry Pi. Along with that, we learned interfacing of two external peripherals to the Raspberry Pi- camera and a piezo speaker. The scope of this project ends at this stage with model building, result validation, hardware implementation. In the next section, we discuss on further optimisations that can be made with time and resources to develop a system that can be used on larger scale in various fields.

5.2 Future scope

The project was implemented in two stages- software and hardware.
In model design and training, there can be more development on acquiring more data, exploring various other architectures to get results in lesser time.
Other System on Chip (SoC) can be surveyed. The same hardware board with resource optimisation would be a field of research to build a system and implement in commercial use.