ABSTRACT

Corona Virus Disease (COVID-19) pandemic has caused a health crisis. One of the effective methods to prevent the spread of its deadly virus is to wear mask. Face Mask detection can be used to identify people who wear mask. The model is developed with a machine learning algorithm through the image classification method: MobileNetV2. The steps for building the model are collecting the data, pre-processing, split the data, testing the model, and implement the model. The built model can detect people who are wearing a face mask and who are not wearing it

TABLE OF CONTENTS

List of Figures

1 Introduction

The coronavirus disease 2019 (COVID-19) is an ongoing global pandemic caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The virus can spread from an infected person's mouth or nose in small liquid particles when they cough, sneeze, speak, sing or breathe heavily. Wearing mask along with other basic sanitary measures are very important to keep the spread of the Covid-19 as slow as possible. To slow the spread of Covid-19, face masks are mandatory in public places, but monitoring a large population is quite difficult. One of the solutions is to use a automated face mask recognition system to quickly recognize the offenders and advise them to wear a face mask as quickly as possible. Also, it can alert the authorities. Face identification categorically deals with distinguishing a specific group of entities i.e. Face. It has numerous applications, such as autonomous driving, education, surveillance, and so on Face mask detection involves in detecting the location of the face and then determining whether it has a mask on it or not. The problem is similar to general object detection to detect the classes of objects. This project presents a simplified approach to serve the above purpose using raspberry pi with USB camera to create a prototype for face mask detection systems.

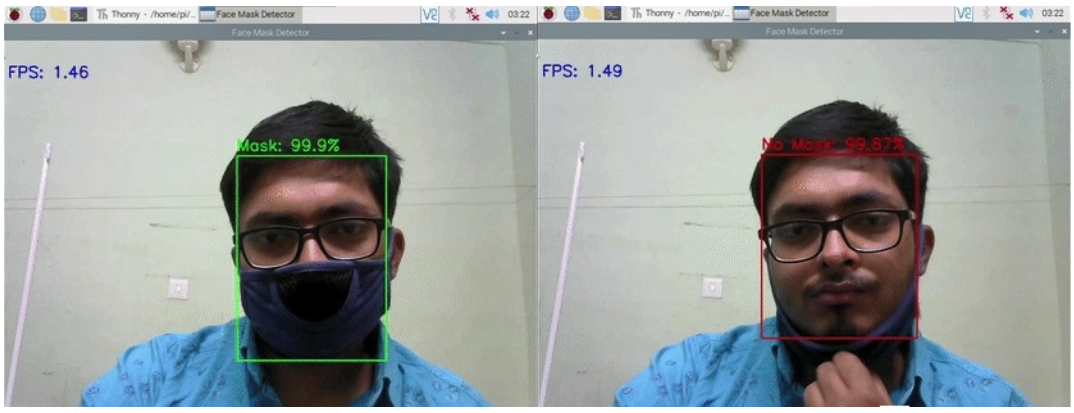

Figure 1: Face Mask detection

2 Background

In face detection method, a face is detected from an image that has several attributes in it. Given an image, the challenge is to identify the face from the picture. Face detection is difficult because the faces change in size, shape, color, etc. and they are not immutable. Recently, Convolutional Neural Network (CNNs) has demonstrated very promising results for face recognition, but it comes with a strict constraint regarding the size of the input image. In this project, the main challenge of the task is to detect the face from the image correctly and then identify if it has a mask on it or not. In order to perform surveillance tasks, the proposed method [1] should also detect a face along with a mask in motion.

2.1 Experiment Setup

Raspberry pi 4 with 4GB RAM is used as the main processing for this image processing technique. A USB Camera is connected to Raspberry pi which is used to capture the frames and send it to the Raspberry pi for processing. OpenCV software is used for processing the frames which detects faces using Haar Cascade algorithm and uses a pre-trained model based on MobileNetV2 to detect mask on the faces.

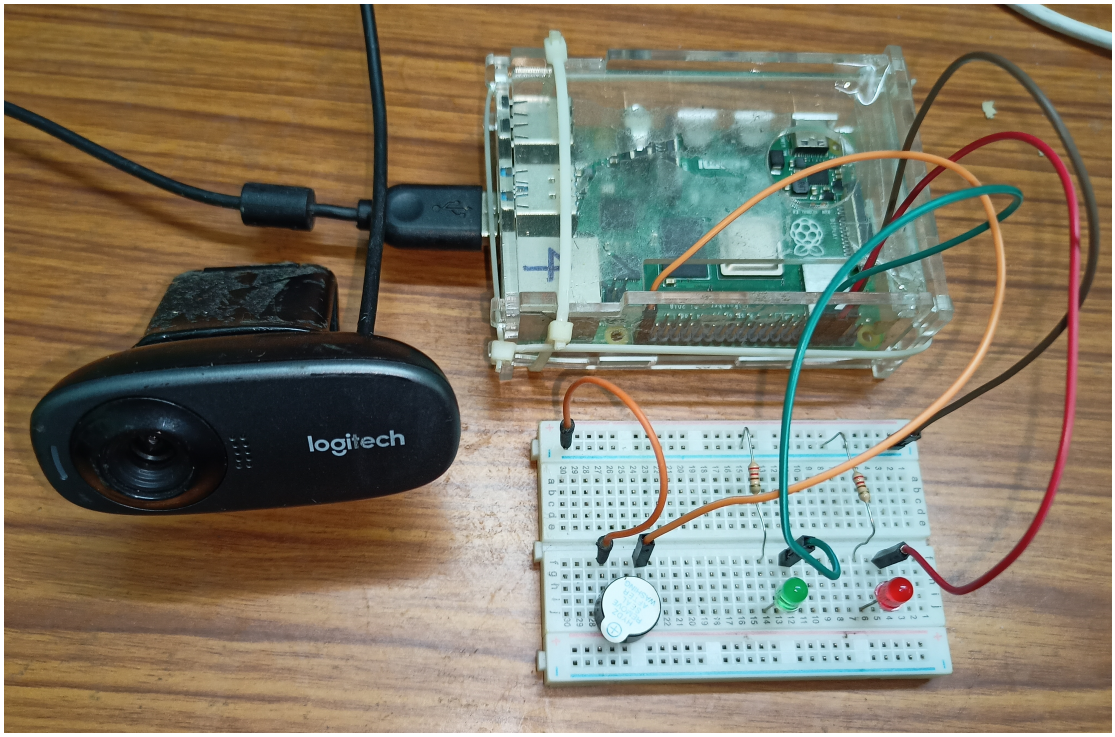

Figure 2: Experiment setup

3 Components Used

3.1 Raspberry pi 4

Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation which provides desktop performance comparable to entry-level x86 PC systems. Raspberry Pi 4 Model B has 1.5 GHz 64-bit quad core ARM Cortex-A72 processor

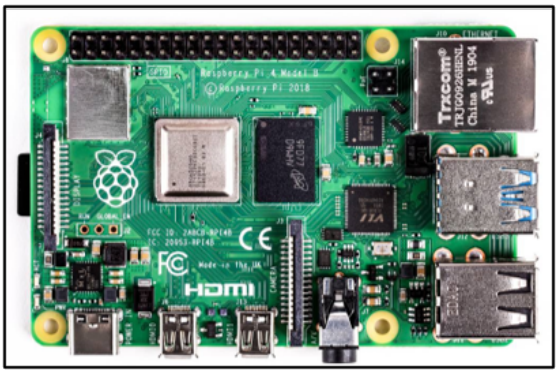

Figure 3: Raspberry pi 4

3.2 OpenCV

OpenCV [2] (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision, originally developed by Intel. The library is cross-platform and free for use under the open-source Apache 2 License. It supports many popular deep learning frameworks like Tensorflow, Pytorch and Caffe

3.3 MobileNetV2

MobileNetV2 [3] is a convolutional neural network (CNN) architecture developed by Google for mobile and embedded vision application. MobileNets are small, low-latency, low-power models parameterized to meet the resource constraints of a variety of use cases. It builds upon the ideas from MobileNet [4], using depth-wise separable convolutions as efficient building blocks. It contains two new features - Linear bottlenecks between the layers & Shortcut connections between the bottlenecks. Its architecture contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers.

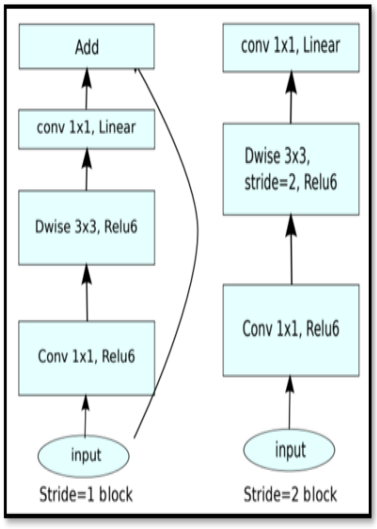

Figure 4: Architecture of MobileNetV2

4 Methodology

4.1 Dataset Preparation

A dataset comprising of around 300 photos is created and labelled as mask or no mask. The data is captured via USB camera connected with raspberry pi and saved in respective folders labelled "Mask" and "No Mask"

4.2 Pre-processing

All the captured images are resized to 224 × 224 pixels, because it becomes easier for the model to be trained with smaller and uniform size. Next, the images are converted into 3D array so that it can be fed to the model sequentially. After that, the image will be used to pre-process input using MobileNetV2. Next, hot encoding is performed on labels. Finally, the dataset is split into 80% training data & 20% testing data.

4.3 Building Model

There are six steps in building the model which are constructing the training image generator for augmentation, configuring the base model with MobileNetV2, adding model parameters, compiling the model, training the model, and finally saving the model for the future prediction process.

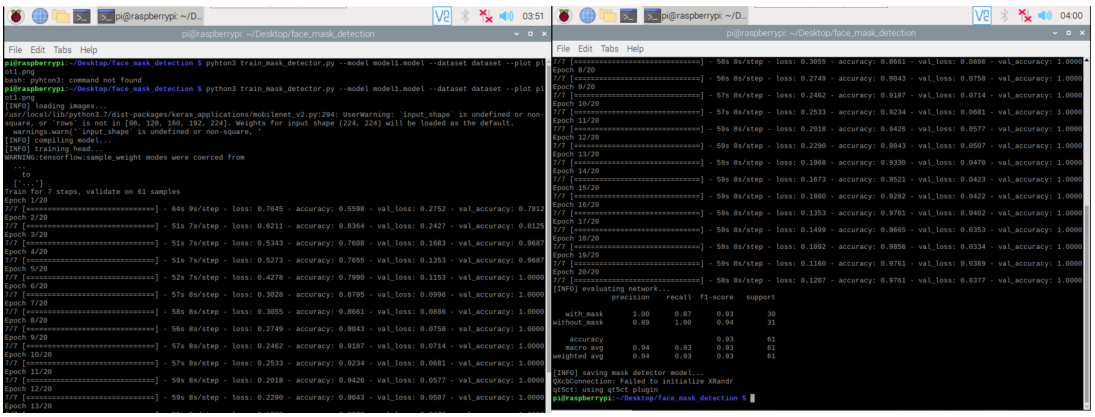

Figure 5: Training the model

4.4 Testing model

The model is tested on training data and a graph of Training loss and accuracy is plotted. After that the model evaluation is done. The metrics selected for evaluation of model is explained below.

$$Accuracy = \frac{Tp + Tn}{Tp + Fp + Fn + Tn} \tag{1}$$

$$Precision = \frac{Tp}{Tp + Fn} \tag{2}$$

$$Recall = \frac{Tp}{Tp + Fn} \tag{3}$$

$$f1score = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{4}$$

where, Tp = True positive, Tn =True negative,

Fp = False positive, Fn = False negative

4.5 Implementation

The above saved model is implemented using Raspberry pi 4 with frames captured by USB Camera. The face detection algorithm detects faces using OpenCV's Haar Cascade [5] algorithm and the pre-trained model for mask detection detects mask on face. If a face with mask is detected, Green LED turns ON, but if no mask is detected, buzzer and Red LED turns ON alerting the authorities about potential breach of Code of conduct.Additionally, if no mask is detected for 10 seconds, an email is sent to the admin.

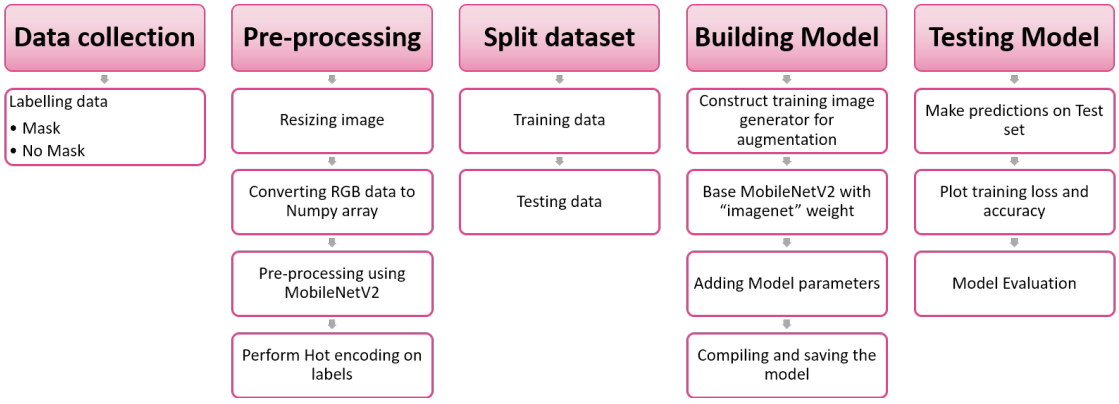

Figure 6: Overall steps in building the model

5 Results and Analysis

5.1 Results

The results obtained on training and evaluating the model is as shown below:

	Precision	Recall	F1-score
With mask	1.00	0.87	0.93
Without Mask	0.89	1.00	0.94
Weighted average accuracy	0.94	0.93	0.93

Figure 7: Evaluation result

The training and Loss accuracy during training process is plotted below

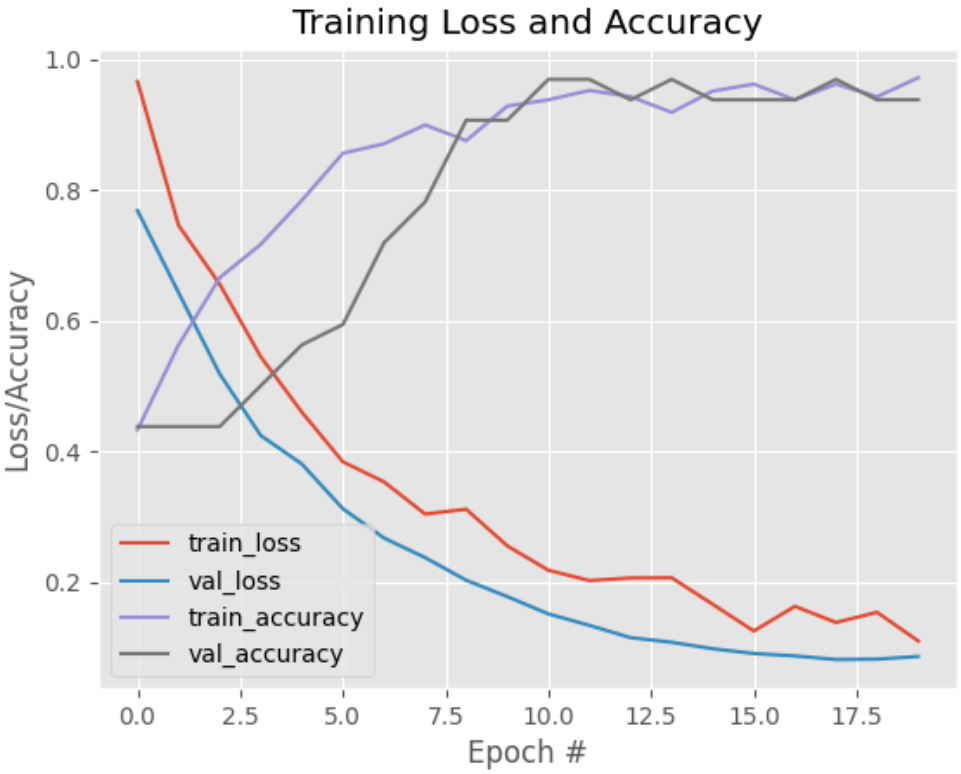

Figure 8: Training Loss and accuracy

5.2 Analysis

We can clearly see that the model is able to predict face with mask with an accuracy of 97% . But, as evident from the evaluation results, the model is over-fitting with precision for "with mask" category and recall for "No mask" category both being equal to 1.0. When the email is being sent, the camera stops capturing frames for around 2 seconds.

6 Conclusion

The model is over-fitting probably because the model is trained on similar kind of data and no enough data is provided. Despite this, the model will work very good when it is trained with more and varied data. Also, The output frames per second is slow, because there is not much computational power present in Raspberry pi, and also many other applications are running in parallel (as the GUI output is also being provided). The model should work fine when only a dedicated system is being used with no GUI output. This experiment is just a prototype to test whether a face mask detector can be made using embedded devices. There is a lot of scope of improvement in this model and also on the system such as image of person not wearing mask for long time can also be sent through email.

References

[1] 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)*, pp. 1–5, 2020.

[2] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[5] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.