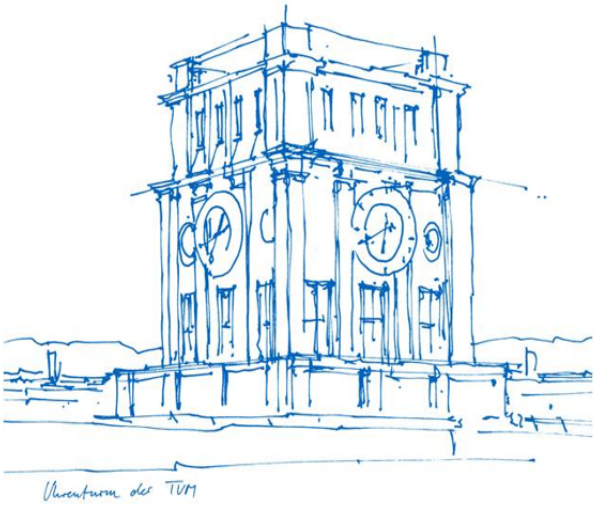
Face Mask Detector

Advanced Topics in Communication Electronics



Prepared by:

Abhishek Sengupta , 03736060 , ge79car.sengupta@tum.de
Priyadharshini Ponraj , 03741605 , priya.ponraj@tum.de
Smriti Nandy , 03737798 , ge59wof@mytum.de
Vinay Shankar Kulkarni , 03738026 , vinay.kulkarni@tum.de

DATE: 17.01.2022

Abstract

COVID-19 is causing a health crisis worldwide. In this current pandemic situation, wearing a face mask is very important as it is one of the necessary safety measures to prevent infections. To ensure the use of face masks by the people in crowded places or larger gatherings, a face mask detection system can be designed. This would help reduce the possibility of spreading the virus. With the use of deep learning detection algorithms and embedded systems, the algorithm can be incorporated in the security systems of a room or any enclosed space. This system would open the gates only if masks are worn, to ensure the safety of all. Using an embedded system would reduce the cost of manufacturing face mask detection systems. A use case for the system would be a classroom, where the doors will unlock only if the incoming person wears a mask, otherwise it will notify the person to wear a mask.

Table of Contents

Components

- Raspberry Pi 4 Model B (4GB RAM)
  The Raspberry Pi 4 Model B [1] with 4GB RAM is being used with key specification of 1.5GHz quad-core Broadcom BCM2711 CPU. Since the current available hardware is a 4GB Raspberry Pi 4, the face detection model is implemented on this hardware. At a later stage, the model will be implemented on a Raspberry Pi 4 Model B 2GB RAM version.

- Raspberry Pi OS (Debian-based)
  The latest Debian version 11 (Bullseye) is installed for the Raspberry Pi 4 hardware.

- USB Camera
  The specification of the USB camere is as follows: 720P HD camera with a 120° wide-angle for real-time image and video transmission.

Implementation

The Face Mask Detection model is based on the MobileNetV2 architecture with the default MobileNetV2 weights and ADAM optimizer.

- Libraries used for the script to deploy the face detection model on Raspberry Pi:
  - opencv-python
  - numpy
  - keras
  - math
  - tensorflow/tensorflowLite

- Parameters for the different layers of the training model shown in Fig.1:
  - Learning rate = 0.0005
  - No. of epochs= 20
  - Batch size= 32
  - For MobileNetV2 : apha = 0.35
    Weights: Default MobileNetV2 weights

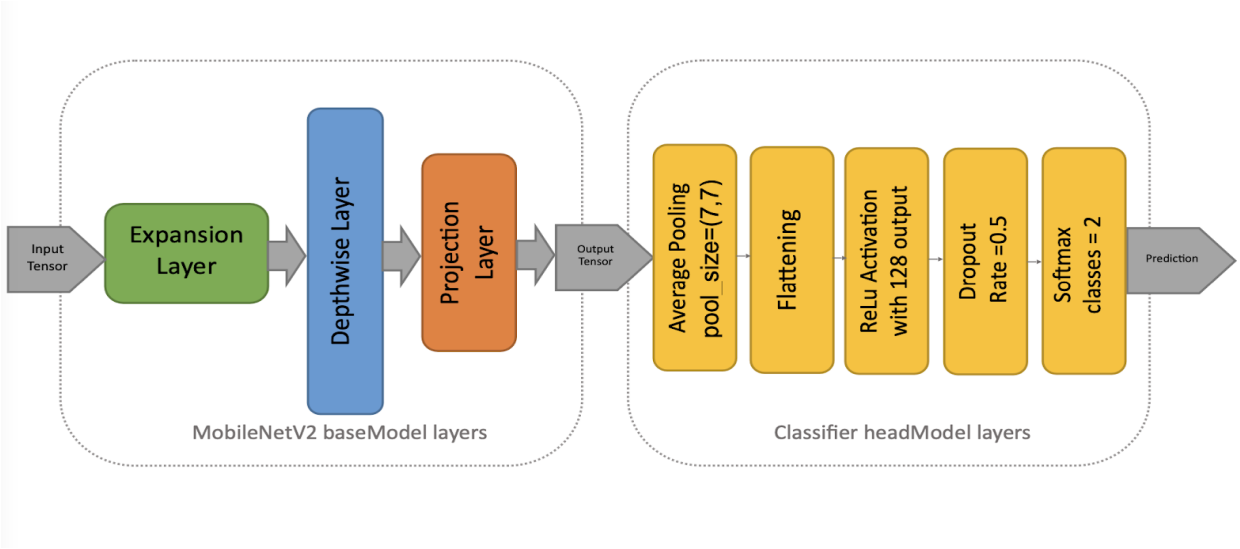


Figure 1. Block diagram of the layers of the Model

Method 1: Using Tensorflow

The model is not being trained on the Raspberry Pi, but instead we used the PC trained model file for the testing phase. Initially, the testing script used on the PC was based on TensorFlow. Therefore, the first step was to install the TensorFlow package on the Raspberry Pi. During the installation of the package, there was a version mismatch of the dependency libraries and installation was quite tedious and unsuccessful.

Method 2: Using TensorflowLite

In order to make installation of the package easier, we have converted the Tensorflow functions to TensorFlow Lite functions for the testing script used on the Raspberry Pi. While exploring other alternatives for the model file using Tensorflow package, we have come across EdgeImpulse.

Method 3: Using EdgeImpulse

Edge impulse[2] was used to design the MobileNetV2 architecture, shown in Fig.1. The model file generated after training is of the format .eim and is completely self-contained with the libraries and dependencies like tensorflow and hence importing them can be avoided. Currently, the unoptimized float 32 model version is deployed on the Pi to achieve better accuracy. For the future work, a quantized int8 version of the model can be implemented that gives low accuracy and higher system performance.

Result

- Table 1 shows the accuracy prediction of the model. We observe a high prediction accuracy with the unoptimized float32 model. F1 score depicts the weighted average of the Precision and Recall.
- The overall training accuracy of the model is 96.3% and loss is 0.12 .
- Memory consumption of the Raspberry Pi is given in Fig.2. The total memory usage of the Raspberry Pi is approximately 267MB.

Data Classification	**Mask (Predicted label)**	**No_Mask (Predicted label)**
Mask (Actual label)	98%	2%
No_Mask(Actual label)	5.1%	94.9%
F1 score	0.96	0.97

Table 1:Prediction accuracy for the model

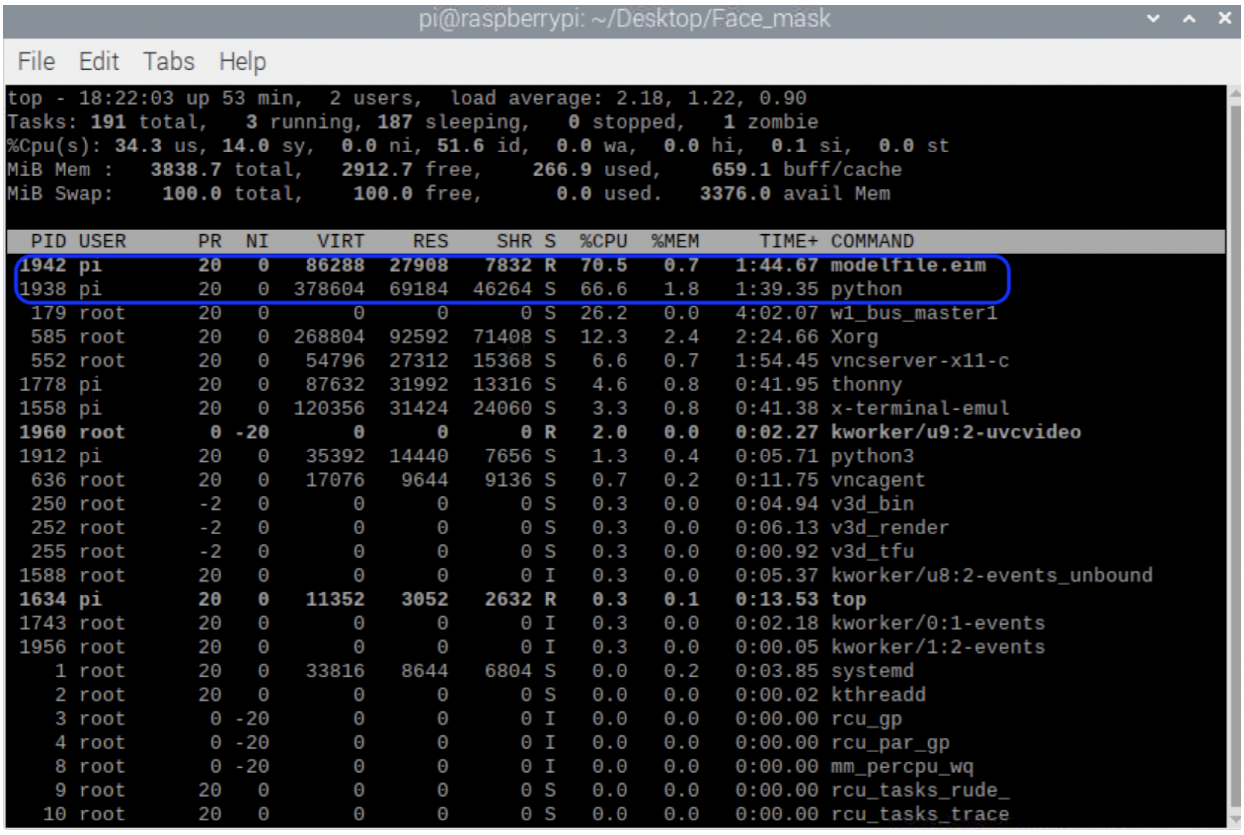


Figure 2. Raspberry Pi memory consumption

Summary

Finally, method 3 is used to implement the model on Raspberry Pi, as the trained model is completely self-contained with the libraries and dependencies. Hence it was feasible to implement a python SDK using openCV and edge_impule_linux for live classification The accuracy observed is 98% with 0.12 loss factor on the validation set. The runtime memory usage of the Raspberry Pi is ~267 MB.

Future work

- The implemented model classifies the background (Live stream without a person) as the label “Mask”. For the later work, a frame without a person should not be classified as “Mask” or “No_Mask” and improve the prediction accuracy with a quantized model.
- Implement buzzer/LED circuit along with the Raspberry Pi 4 hardware.

Bibliography

[1] RaspberryPi 4: https://www.raspberrypi.com/raspberry-pi-4-model-b/
[2] EdgeImpulse: https://www.edgeimpulse.com/

Appendix

Link for Implemented Model on Gitlab: https://gitlab.lrz.de/