Advanced Topics in Communication Electronics, MSCE, TUM, Winter 2021.

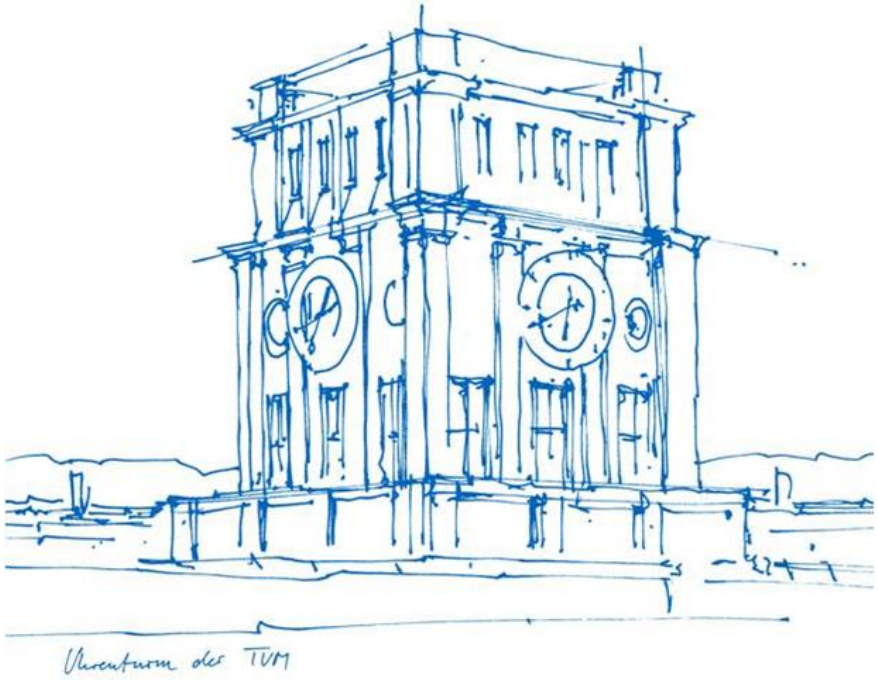
Final Project Presentation

FACE MASK DETECTION

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: 04.02.2022

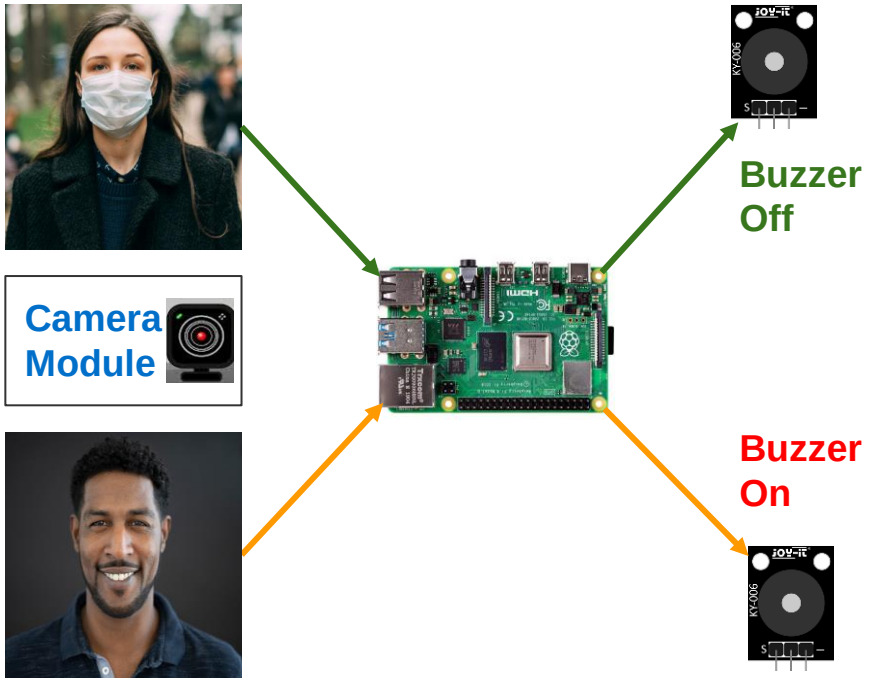


Structure

- Introduction
- Technical Challenges
- Detailed Design
- Performance Evaluation
- Conclusion
- Individual team member’s contribution
- References
- Demo Video

Introduction

- Deep Learning based facemask detector
- Video captured by camera module is used to determine if the person wears facemask or not
- If facemask detected, buzzer does not ring and if not, buzzer rings.



Related Background

- Convolutional Neural Networks have been used in face recognition tasks.
- Squeezenet, Shufflenet, MobileNet are different deep neural network models built previously.
- MobileNetV2 is used here because of residual connections and bottlenecks in its architecture.

Technical Challenges

- Deciding Hardware Board:
    - Arduino - microcontroller
    - **Raspberry Pi - Microprocessor - Python, 1.2GHz clock**
- Limitations of Rpi and requirements:
    - Dependencies and Libraries - Tensorflow
    - Not suitable for training - Limited processing power
    - Memory Constraints
- Data set:
    - Size?
    - Edited Images - Fake face mask. Proceed?

Source: www.kaggle.com

Technical Challenges

- Implementation methods:

1. Training on Raspberry Pi - Tensorflow installation problem

   > .whl is not a supported wheel on this platform

2. Converting model to tf lite - using Python APIs

3. **Edge Impulse -**

   - TensorFlow ecosystem for training, optimizing, and deploying on edge devices
   - Cloud training and optimized trained models for deployment. (foo.eim format)

Issues: Training time exceeded - Optimize Batch Size and epochs

Model Quantization options : Float32(unoptimized) and **int8(quantized)** - int8 uses 8-bit integers instead of floating-point numbers. Smaller and less accurate, but faster.

https://blog.tensorflow.org/2021/06/how-tensorflow-helps-edge-impulse-make-ml-accessible.html

Technical Challenges

- Bounding boxes:
    - Boxes in dataset or tensorflow requirement for face detection.
    - Output - Buzzer only!!
- Multiple face detection:
    - What will be output signal?
    - Ruled out!
- SDK for Live classification:
    - Python script for classifying Live video using the model.eim file - Implementing camera and buzzer

Hardware Setup

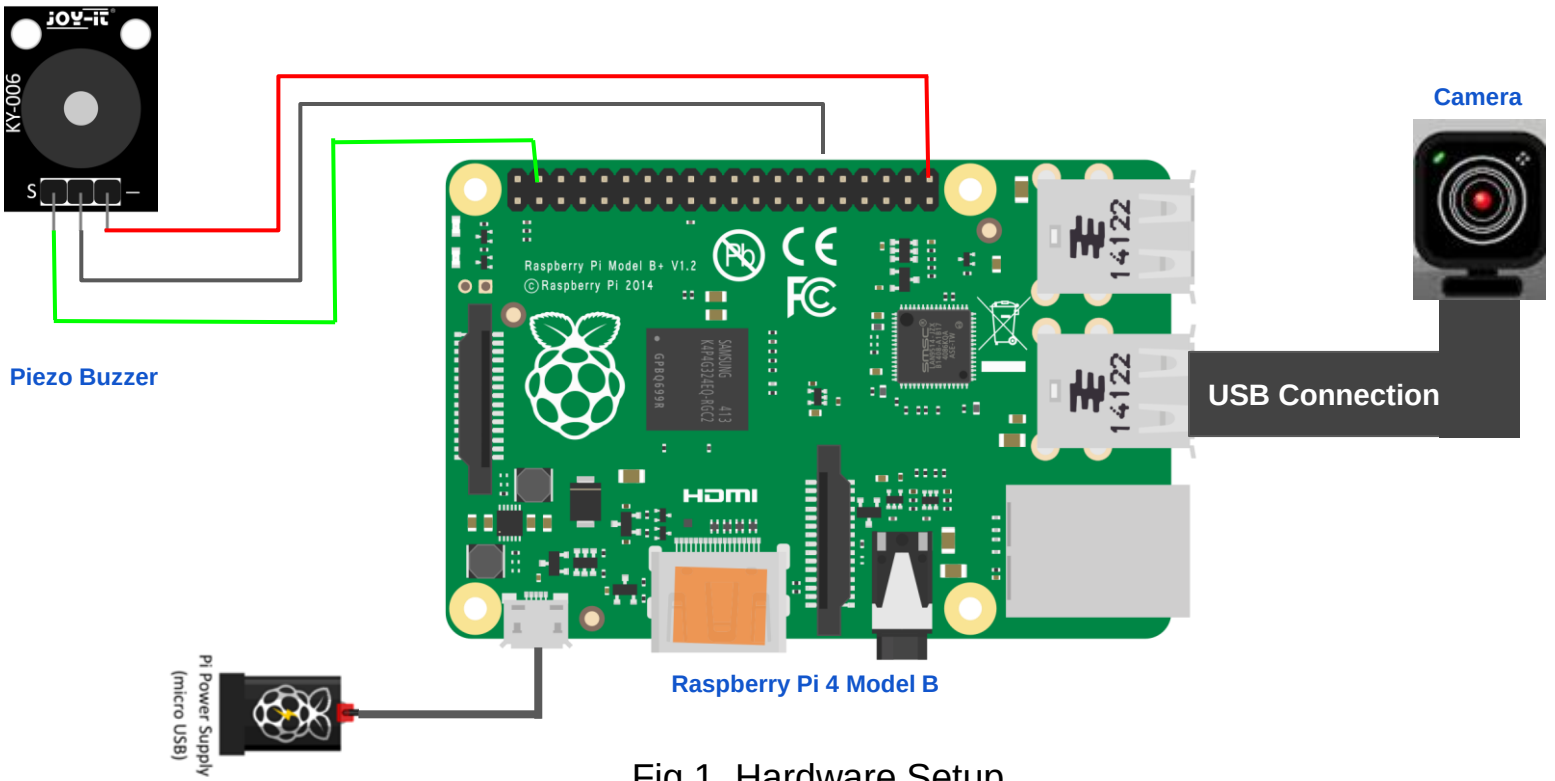


Fig 1. Hardware Setup

Hardware Communication

Communication with camera

```
import cv2

:{
                        }
camera = cv2.VideoCapture(port)
        if camera.isOpened():
            ret = camera.read()

 videoCaptureDeviceId = int(port_ids[0])
⋮                       }
camera = cv2.VideoCapture(videoCaptureDeviceId)

⋮
for res, img in
runner.classifier(videoCaptureDeviceId):
```

Communication with buzzer

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPActiveBuzzer = 17
{.......   }
 if ( scr1 >= 0.7 ):    print( "Mask Detected" )
 elif ( scr2 >= 0.85 ):   print( "No mask
detected")
GPIO.output(GPActiveBuzzer,GPIO.HIGH)#Buzzer on
time.sleep(0.5) #pause for 0.5 seconds
GPIO.output(GPActiveBuzzer,GPIO.LOW)#Buzzer off
```

Machine Learning Model

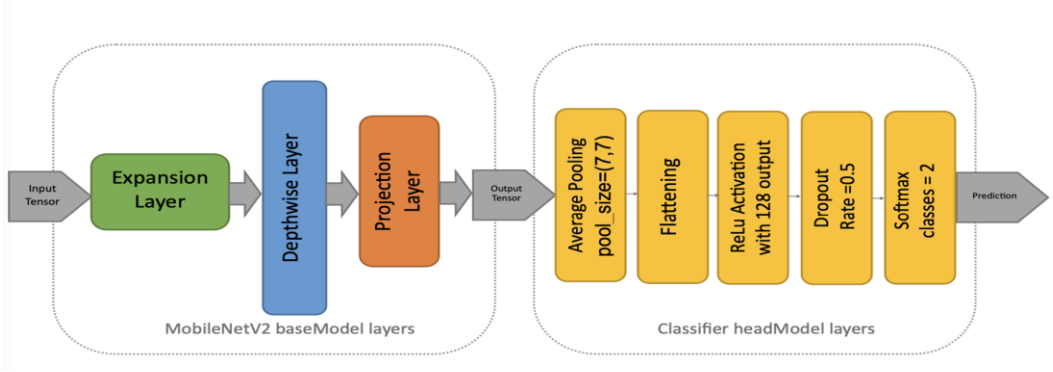


Fig 2 : MobileNetV2 base Model + Classifier Head Model

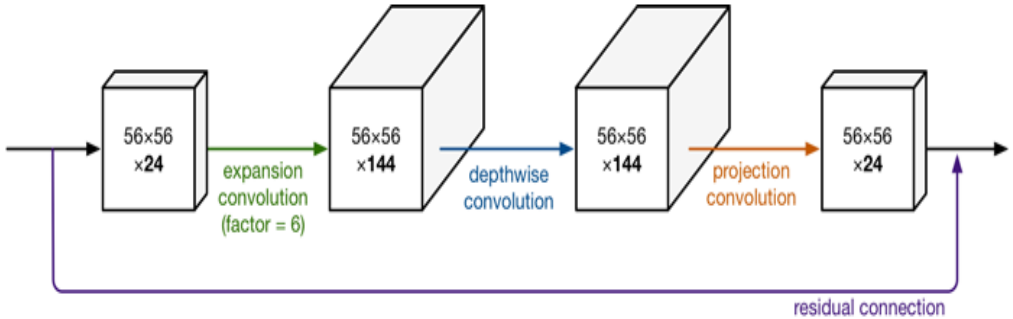


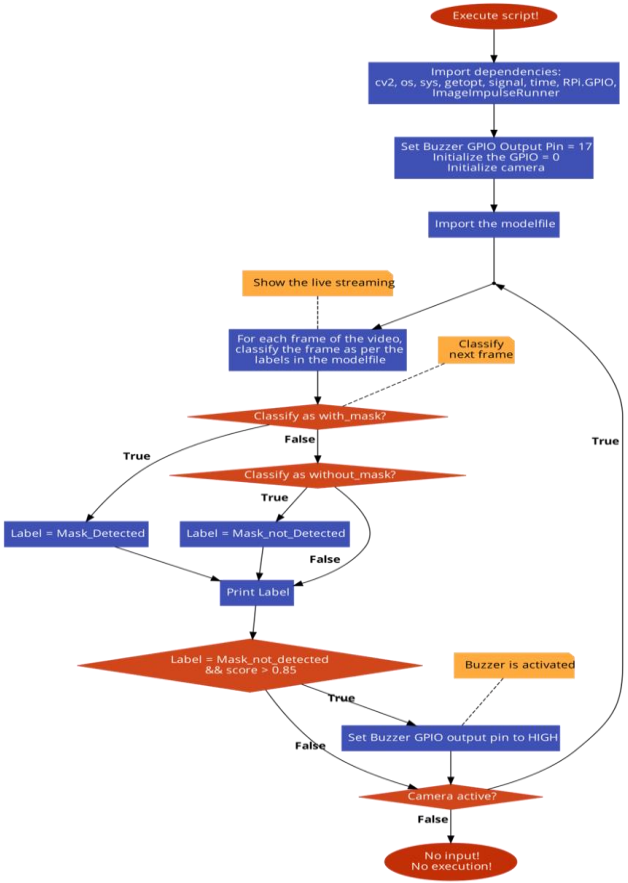
Fig 3 : Working of the MobileNetV2 base Model [1]

Why use the MobileNetV2 Model?

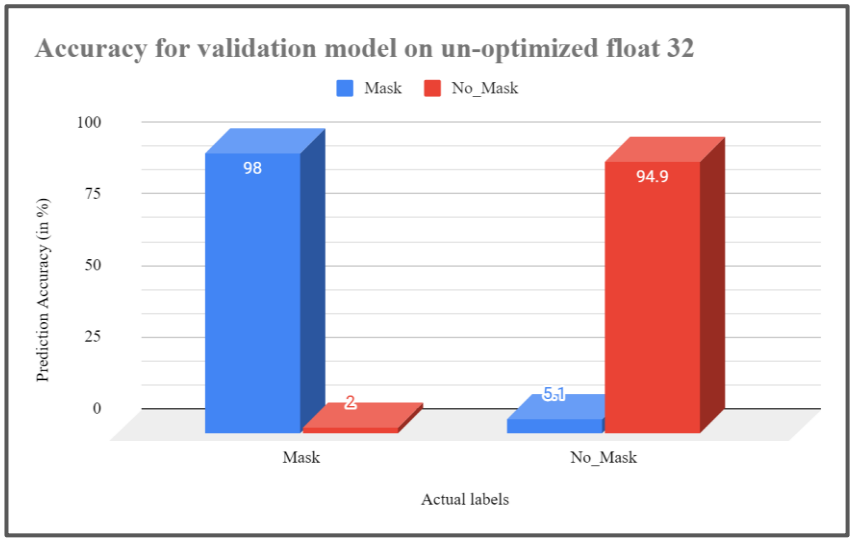
- Decompressed data at the input of the depthwise layer, depthwise convolution more effective.
- Reduced future computations because of low-dimensional tensor output.
- Faster Performance than traditional Machine Learning Models

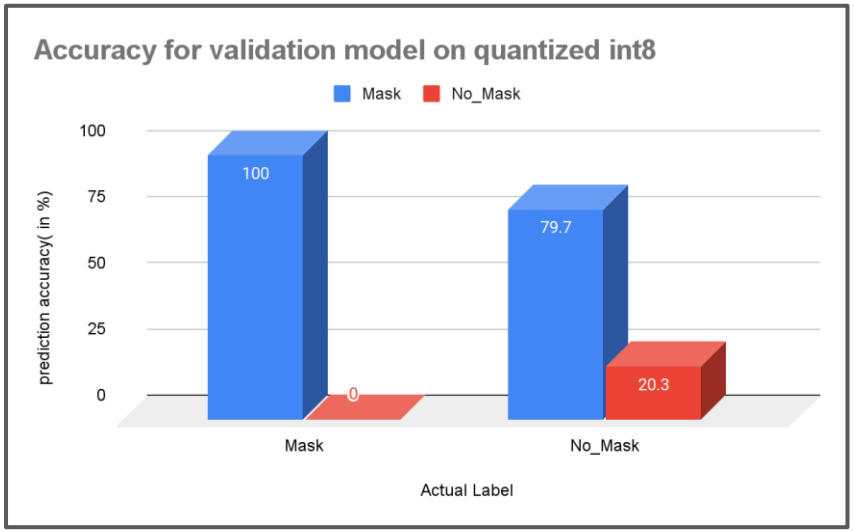
Testing Script

- Initialize camera and buzzer GPIO pins
- Load the pre-trained model file
- Check status of the camera
- Classification of frames for the usb camera video
- Determine probability of the labels
- For label “without_mask” and score > 0.85
    - Buzzer is activated



Prediction Accuracy for the model





F1 score is 0.96 for mask prediction and 0.97 for no-mask prediction

Accuracy : 96.3 %

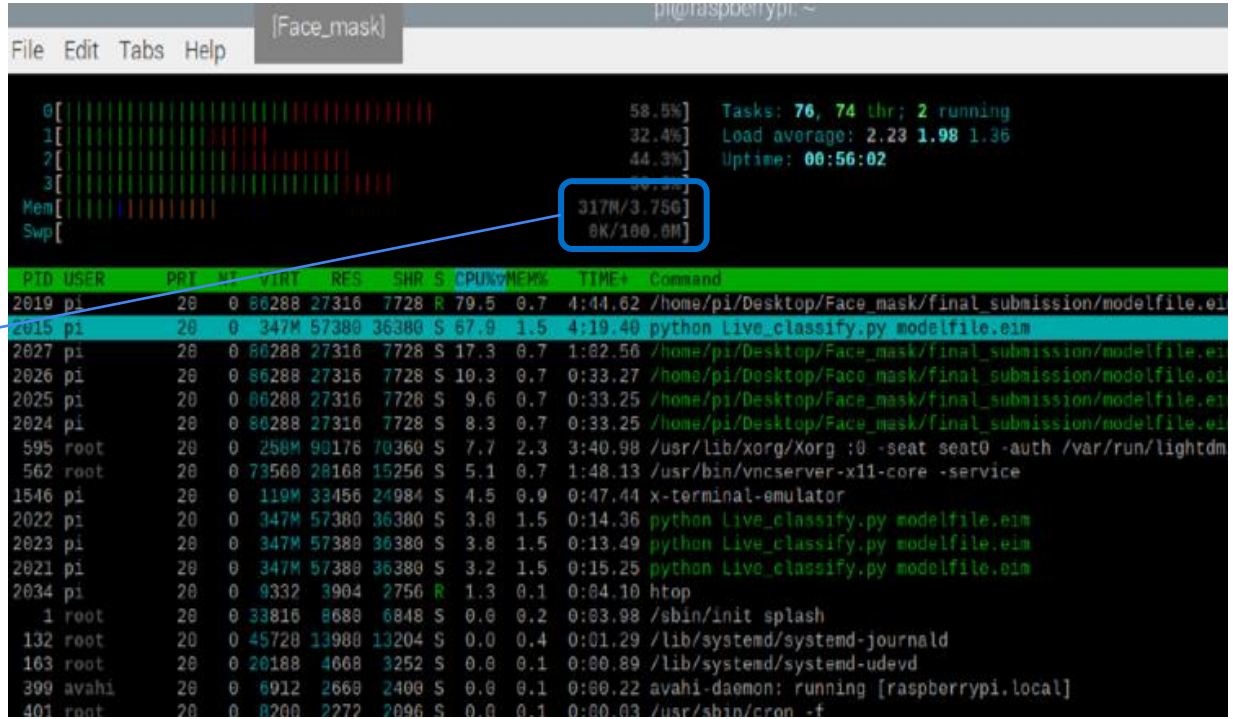
F1 score is 0.68 for mask prediction and 0.34 for no-mask prediction

Accuracy : 56.9%

Resource Utilization

26 MB of memory is used by our model file and the entire python script consumes 56 MB of memory.

The total memory consumed by our model is Approximately 317 MB



Conclusion

Summarizing the project:

- COVID-19 Face Mask Detection model is achieved using the MobileNetV2 CNN architecture.
- Model is trained with ~4000 images of classes with_mask and without_mask.
- Implementation of model on Raspberry Pi 4 with usb webcam and buzzer to indicate the label without_Mask.

Some of the challenges and workarounds:

- Installation of python libraries
  - Use tensorflowLite package
- Compress the tensorflow model
  - Compress the model using fine-tuning
- Achieving good accuracy
  - Unoptimized float32 model

Conclusion

Advantage of the system:

- Easy deployment using EdgeImpulse platform on embedded system
- Fewer computation
- Cost efficient system

Future topics:

- Improve the prediction accuracy with the quantized model.
- Focus on the corner use cases.
- Implement a screen with mask detection for real time applications.

Individual Team Member's Contribution

TOPICS	CONTRIBUTION
Literature survey	Abhishek Sengupta 25% Priyadarshini Ponraj 25% Smriti Nandy 25% Vinay Shankar Kulkarni 25%
Implement detection model on local system	Abhishek Sengupta 25% Priyadarshini Ponraj 25% Smriti Nandy 25% Vinay Shankar Kulkarni 25%
Generate training model for Raspberry Pi4	Abhishek Sengupta 30% Priyadarshini Ponraj 20% Smriti Nandy 30% Vinay Shankar Kulkarni 20%
Testing script and Buzzer Implementation for Raspberry Pi4	Abhishek Sengupta 20% Priyadarshini Ponraj 10% Smriti Nandy 30% Vinay Shankar Kulkarni 40%
Report writing	Abhishek Sengupta 30% Priyadarshini Ponraj 20% Smriti Nandy 25% Vinay Shankar Kulkarni 25%

References

[1] MobileNetV2: https://machinethink.net/blog/mobilenet-v2/

[2] Dataset: https://www.kaggle.com/

[3] Source Code: www.github.com/

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

[5] EdgeImpulse: https://www.edgeimpulse.com/

[6] MobileNetV2 parameters: https://keras.io/api/applications/mobilenet/

Demo Video

Thank You

Questions?