PROJECT REPORT

Real Time Implementation of AI-Face Mask Detector with MobileNetV2

“Statistic under AI and its application to engineering sciences”

Rauzan Sumara

Introduction

The spread of COVID-19 is becoming increasingly concerning for everyone throughout the world. This virus can spread from person to person via droplets and airborne particles. According to WHO, in order to minimize the spread of COVID-19, everyone should wear a face mask, practice social distance, avoid crowds, and constantly keep a healthy immune system. As a result, in order to protect one another, everyone should wear a face mask when going outside. However, most selfish individuals will not properly use the face mask for a variety of reasons.

Therefore, this report is trying to create a robust face mask detection based on convolution neural network. In order to detect a face mask, we implement the object detection algorithm which is MobileNetV2 architecture. The rest of this report will be organized as follows: section two illustrates the datasets. Section three describes the face mask detection algorithm and section four presents the experimental results in real-time data. This work will be closed by the conclusion in section five.

Datasets

The dataset used for this task was created by Prajna Bhandary (https://github.com/prajnasb/observations). The dataset contains 1,376 images belonging to two classes, with mask: 690 images and without mask: 686 images. images without mask are actually artificial face mask datasets based on applying facial landmarks. Facial landmarks allow us to automatically infer the location of facial structures, including, eyes, eyebrows, nose, mouth, jawline, etc. The size of the training set chosen for this task is 80% of the images and the test set consists of remaining 20% of the images.

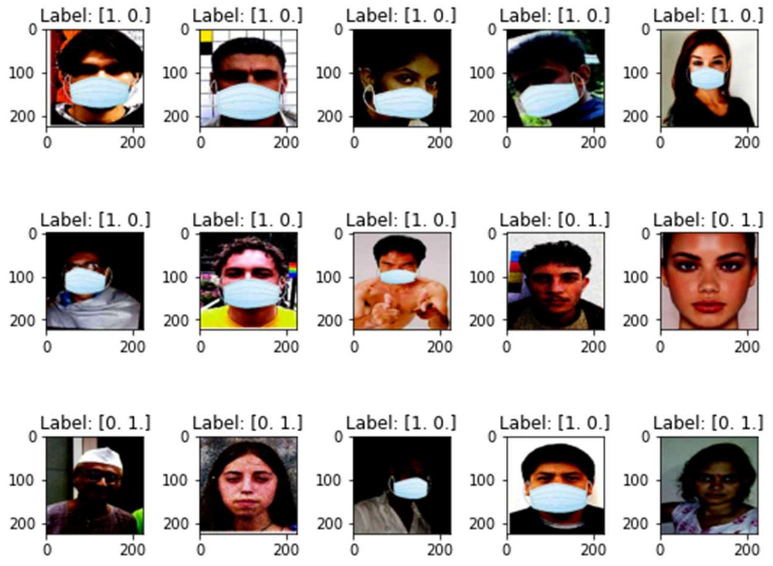


Figure 1. Selected instances of data from the images

The MobileNetV2 Architecture

As for the face mask detector method, this work implemented the convolutional neural network known as MobileNetV2 architecture. According to Sandler, et al. (2018), MobileNetV2 is a convolutional neural network architecture that seeks to perform well on mobile devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depth wise convolutions to filter features as a source of non-linearity. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers. By reflecting these results, it is suitable to implement the method into the real-time face mask detector where high detection accuracy is needed.

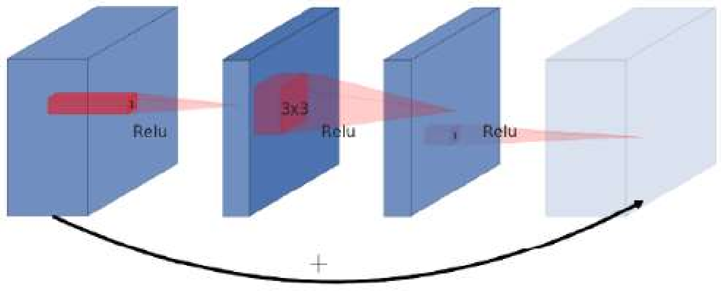


Figure 2. A residual block connects wide layers with a skip connection while layers in between are narrow

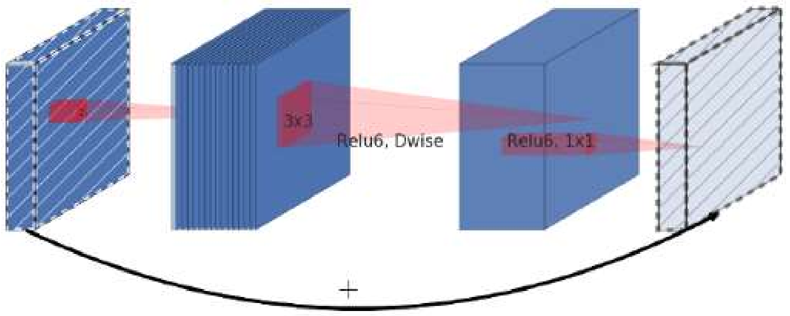


Figure 3. An inverted residual block connects narrow layers with a skip connection while layers in between are wide

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	-	

Table 1. Architecture of MobileNetV2

Now that we understand the building block of MobileNetV2 we can take a look at the entire architecture. In the table you can see how the bottleneck blocks are arranged. *t* stands for expansion rate of the channels. As you can see, they used a factor of 6 opposed to the 4 in our example. *c* represents the number of input channels and n how often the block is repeated. Lastly s tells us whether the first repetition of a block used a stride of 2 for the down sampling process. All in all, it’s a very simple and common assembly of convolutional blocks.

Experimental Results

We conducted our experiment using python programming language. The code and results have been posted on my GitHub (*https://github.com/rauzansumara/face-mask-detector-based-on-mobilenetv2*). You can see that the validation loss and validation accuracy both are in sync with the training loss and training accuracy. The training and validation loss are decreasing and there is not much gap between training and validation accuracy. It shows that our model is not overfitting.

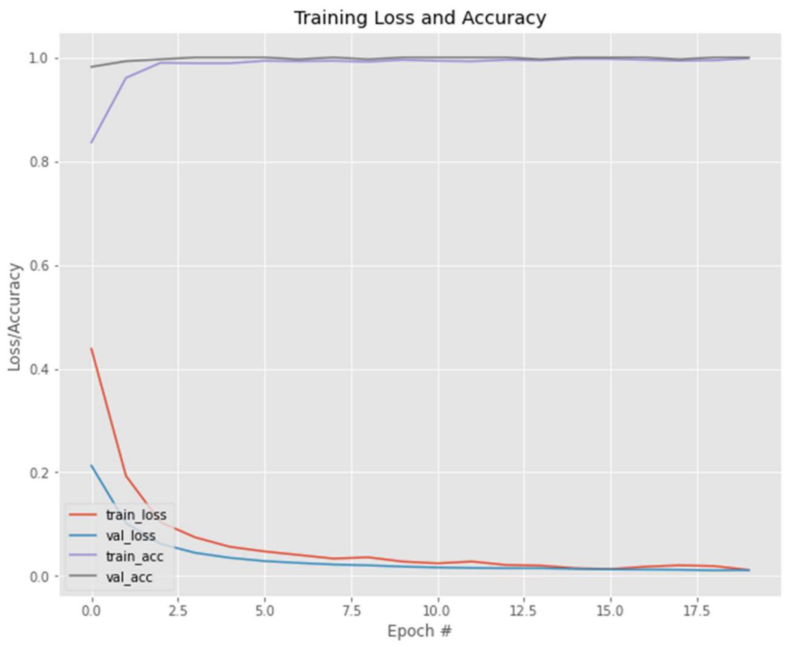


Figure 4. Accuracy and Loss on Training and Testing Sets

Implement Model on Selected instances of Images

we provided two examples of images to evaluate the model. After running the model following results were obtained:

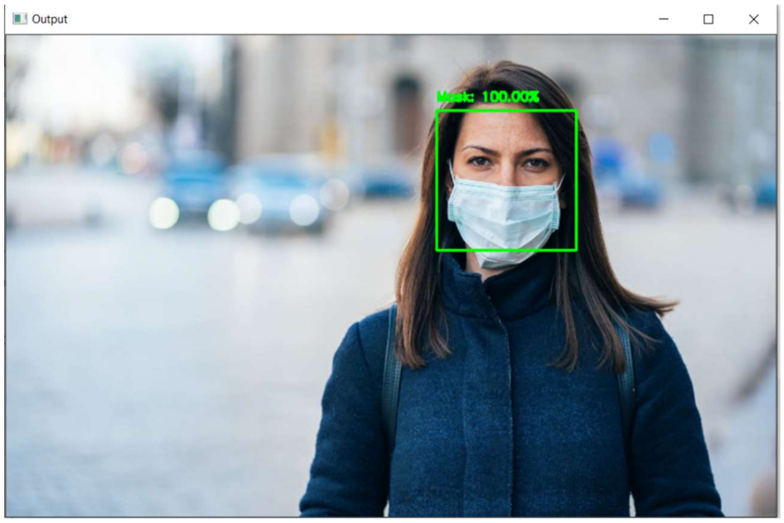


Figure 5. Selected instances of data from the first images

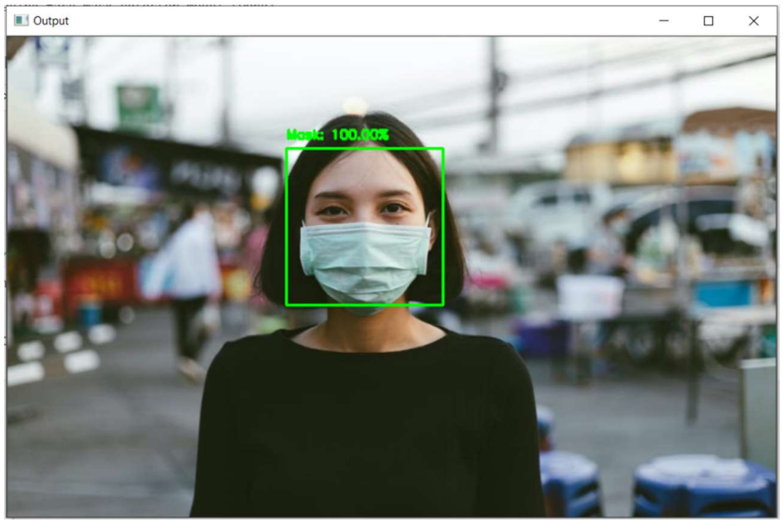


Figure 6. Selected instances of data from the second images

Implement Model on Real-Time Video

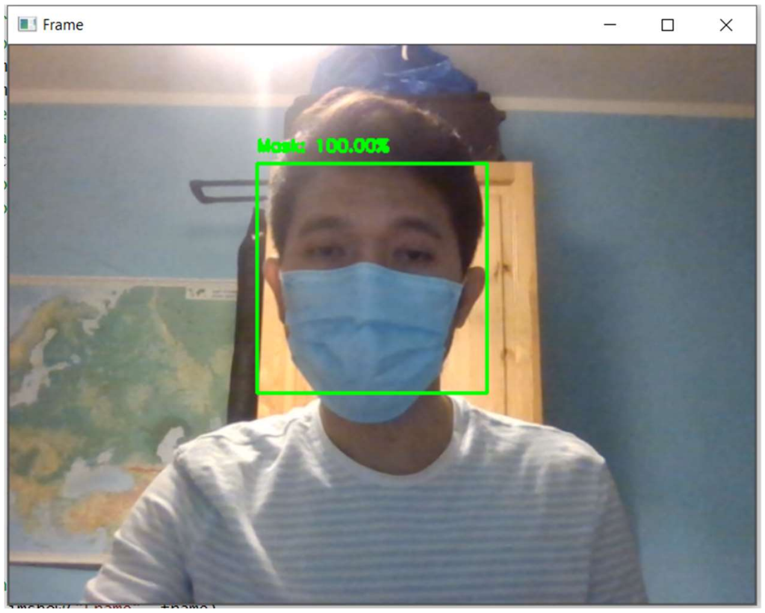


Figure 7. Real-Time Video using Laptop Webcam with Mask

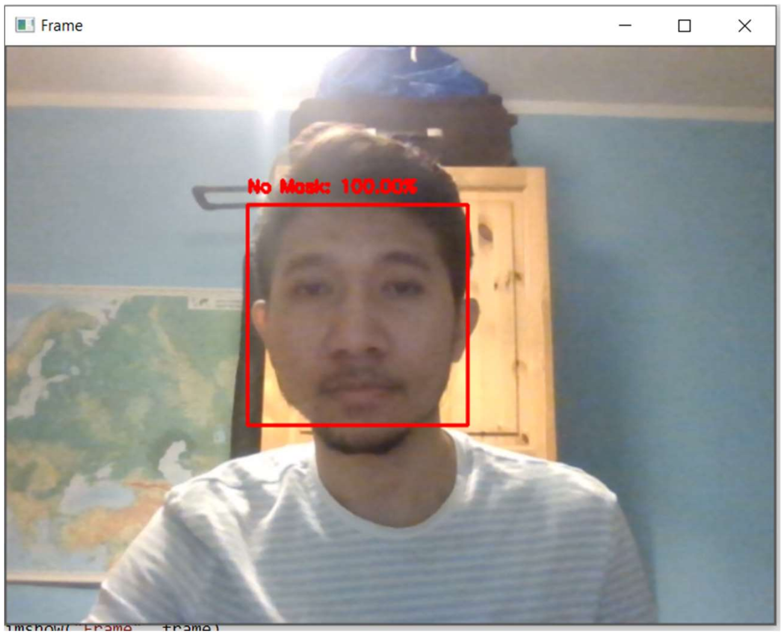
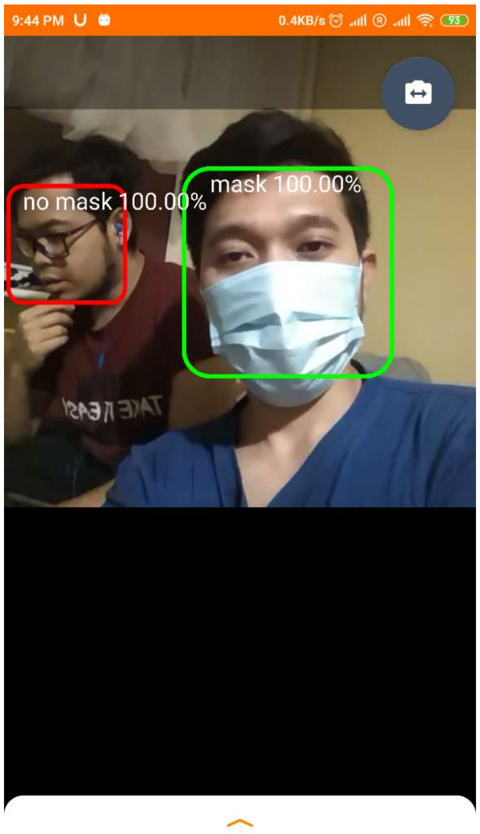


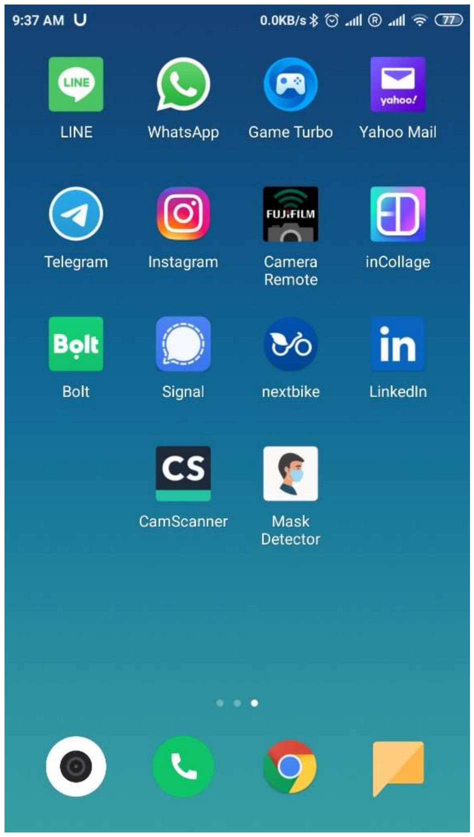
Figure 8. Real-Time Video using Laptop Webcam with Without Mask

Implement Model on Based-Android Application

In this case of implementing our model on based-android app, we need to implement the two steps detection. Most of the work will consist in splitting the detection, first the face detection and second the mask detection. For the face detection step, we are going to use the Google ML kit. For the mask detection, we are going to use our model. First thing to do is using **TocoConverter** python class to migrate from the Keras **.h5** format model to the TensorFlow Lite **.tflite** format model.

It's amazing how easy a high-level deep learning model can be ported to format suitable for mobile, simply by executing one line of code. The model was created in previous section, producing a '.h5' file of about 11.2 MB. After TensorFlow Lite conversion, the resulting file is very light-weight only 9.2 MB, really good for a mobile application. Here is the example of the working app on my Phone (Android Version 6.0.1).


9:44 PM
0.4KB/s
no mask 100.00%
mask 100.00%


9:37 AM
0.0KB/s
LINE
WhatsApp
Game Turbo
Yahoo Mail
Telegram
Instagram
Camera Remote
inCollage
Bolt
Signal
nextbike
LinkedIn
CamScanner
Mask Detector

References

Bhandary, Prajna. 2020. Mask and non-Mask Datasets [https://github.com/prajnasb/observations]

Prove, Paul-Louis. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks [https://towardsdatascience.com/mobilenetv2-inverted-residuals-and-linear-bottlenecks]

Rosebrock, Adrian. 2017. Deep Learning for Computer Vision with Python: Practitioner Bundle. Pyimagesearch

Sandler, Mark, et al. 2018. ChenMobileNetV2: Inverted Residuals and Linear Bottlenecks. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4510-4520 [https://arxiv.org/abs/1801.04381]

Sumara, Rauzan. 2022. AI-Face Mask Detector with MobileNetV2 [https://github.com/rauzansumara/face-mask-detector-based-on-mobilenetv2]