STATS418 Group Project - Real Time Face Mask Detection - Report

Krishna Dave, Sasha Farzin-Nia, Yuxiu Zheng, Ziyao Li

Introduction

Since the beginning of SARS-CoV-2, also known as, COVID-19 at the end of 2019, over 6 million deaths have been reported world wide [1]. The two things that have statistically proven to be mitigating factors in hospitalisation and death from COVID-19 have been vaccinations (such as Pfizer, Moderna, Johnson and Johnson, etc…), and the use of facemasks. In this paper we will be taking a look at how the use of Convolutional Neural Networks (CNN) can help businesses such as restaurants allow the slow and gradual increase of normality in these unprecedented times, in a safe yet effective manner.

Project Scope

During the rise of SARS-CoV-19, the world seemed to fall apart with a lack of planning. The two things that seemed to have been mitigating factors in deaths were the vaccinations (along with the booster) and the use of facemasks. According to the Center for Disease Control and Prevention (CDC), the use of a respirator (such as an N95/K95), lowered the odds of testing positive for COVID-19 by 83%. With this being said, the goal for our project was to create a web-application in which a CNN model is used in real-time through a webcam to detect whether someone is wearing a facemask. We are hopeful that this could attract a target audience of shopping centres and restaurants, where they use this model architecture at the entrance of the buildings, where you can accurately detect a lack of compliance in mitigating the risks of this pandemic.

Using a model found from GitHub as a starting point, we locally trained the CNN on our machine, and using the Python Library, Flask, embedded the Deep Learning Model with the use of HTML and CSS to improve the user interface of the website.

Model Architecture

The dataset which we used was called ImageNet, one which contains over 14 million hand annotated images.

First, we split our data into the training set which will contain the images on which the CNN model will be trained and the *test set* with the images on which our model will be tested. We took *test_size =0.2*, which means that 80% of the total images will go to the *training set* and the remaining 20% of the images will go to the *test set. Then* we constructed the training image generator for data augmentation.

Next, we built a neural network model using the MobileNets2 from *TensorFlow* with Keras library. MobileNets models are based on a streamlined architecture that uses depth-wise separable convolutions to build light weight deep neural networks MobileNetV2 is very similar to the original MobileNet, except that it uses inverted residual blocks with bottlenecking features. It has a drastically lower parameter count than the original MobileNet. MobileNets support any input size greater than 32 x 32, with larger image sizes offering better performance.

For the model, we trained for 20 epochs (iterations). We see that at the end of 12th epoch, our model has an accuracy of 99.5% with the training set and an accuracy of 98.3% with the test set.

Reflection

All in all, the use of a CNN using MobileNets2, proved to perform extremely well. Looking at training the model with 20 epochs didn't prove to perform much better than training the model with 12 epochs, however by doing this, we reduced the computational time by tens of minutes, a substantial decrease. The most challenging part of this whole project was to implement the Deep Learning Model using Flask - we learnt that separately writing code for a Neural Network model, especially with the use of helpful forums such as StackOverflow (which is not dissimilar to how things are done in industry), is not the most difficult part of creating an end-to-end product, such as a Web Application, but it is the putting everything together, similar to putting a 1000 piece jigsaw puzzle, can be very confusing at some points. This project has allowed us collectively to deal with complex scenarios in our future endeavours.

Future Work

Model Architecture

A future improvement we could make to this CNN model is to look into how to optimise the Neural Network, this could be done by fine tuning the hyperparameters that are used, for example, the use of the Adam compared to a Stochastic Gradient Descent Optimizer. Or take a look at the learning rate, and see if there are any improvements to be made on that end. To create a deep learning algorithm isn't difficult per se, the art comes from the ability to finetune the model to minimise computational power and time, yet achieve the best results as possible.

Application

We would like to make some updates to our current application, the biggest change being to allow it to be used on any machine, and not just locally. We can do this by deploying our Deep Learning model and application to a cloud-based service such as Amazon Web Services, Google Cloud, or Microsoft Azure - this would allow the businesses we want benefiting from this to be

able to use it for themselves. Some other features we would like to implement on our website, would be to create a Real Time detection of vaccination cards, which could hopefully detect fraudulent vaccination cards.

References

[1] Kaggle Face Mask Dataset

[2] ImageNet Data Set

[3] TensorFlow and OpenCV Resources

[4] YouTube Tutorial - Face Mask Detection using Python, Keras, OpenCV and MobileNet

[5] MobileNet Model Keras API

[6] Research Paper on MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

[7] Building and Running a Docker Container For ML Model

[8] Deploying Machine Learning Model using Flask

[9] Template for the Facemask Detection Web App