Re-thinking Face Detection

Oleksiy Sharov
Computer Science
Data Science and Engineering
Visual Computing
University of Genoa

Ilkay Albayrak
Computer Science
Data Science and Engineering
Artificial Intelligence
University of Genoa

Somya Mittal
Computer Science
Data Science and Engineering
Visual Computing
University of Genoa

***Abstract*—Face detection and recognition are popular and widely used topics in our lives. From our mobile phones to surveillance cameras, they both used in many businesses and public areas. However, the forced lifestyle change caused by COVID-19 pandemic also forces us to re-think our approach to face detection and recognition. In this paper, we will explain how we implemented a basic system to detect faces and label them if they are wearing masks or not. The object of our project is to provide an experimental analysis discussing how our previous methods scale against today's standards where we have to wear a mask daily.**

I. Introduction

Over the last few decades face detection became an intriguing topic for researchers. With the technological advancement of today, it never stopped continuously reaching new heights, and became available for everybody to use. Since, face detection evolved into something we depend a lot in our day-to-day lives over the years, it affected a lot from the necessity of wearing a mask due to COVID-19. Hence, we will use a MTCNN based face detector [1] and create a model for mask detection to identify how a robust face detector perform while detecting faces with masks or in other words, faces with less facial details.

A. Overview of the Workflow

Our workflow consists of two main phases as can be seen in Figure 1. Firstly, training a binary classifier to identify if a face has a mask or has no mask. Secondly, combining the trained mask detection model with the face detection model.

1) First Phase:

- Load a dataset that has samples of faces with masks and faces without masks. Preferably real data.
- Define a model using Keras/Tensorflow [2] to be trained on the chosen dataset, and train the classifier.
- Save the model with now configured weights for using later in the second phase.

2) Second Phase:

- Load and initiate the mask / no mask classifier.
- Load and initiate the chosen face detector, in our case it is MTCNN(Multi-task Cascaded Convolutional Networks). Because, it achieves superior accuracy over the state-of-the-art models for face detection [1].
- Extract the ROI(Region of Interest) of each face on the given image by using the output of the face detector.
- Pass the extracted face to the trained classifier to determine if the it has a mask or not.
- Finally, show the results on the given image.

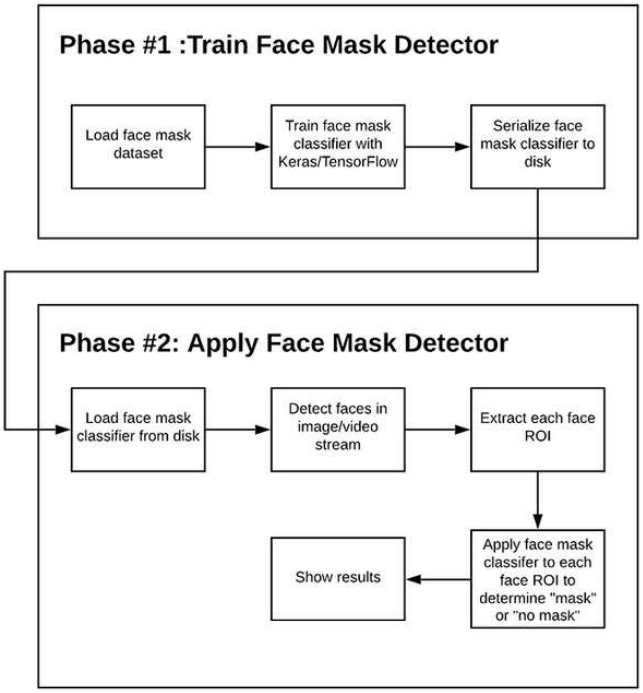


Fig. 1: Workflow for combined face-mask detector [6]

B. The Dataset

The dataset we used to train our mask detection model is "Masked Face Recognition DataSet", from the github user X-zhangyang's "Real-World-Masked-Face-Dataset" repository [5]. It's a public dataset that contains 90,000 face images without masks, and 2203 face images with masks. The dataset also offers a wide range of image sizes, illumination and face positions. Figure 2 shows a pair of face images. (a) a face image without mask. (b) a face image with mask.

We randomly removed 87,800 of the face images without masks to reduce the computational cost, and to be able to train the model without having memory problems. We also didn't use 20 of the face images with masks because they weren't ".jpg, or .png". After this step we had 4383 images in total; 2200 faces with masks, 2183 faces without masks.

Fig. 2: An example of a pair of face images without and with mask. (a) a face image without mask. (b) a face image with mask.

C. Data Augmentatiton

Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without collecting new data [7]. The augmentation techniques create variations of the images that can improve the ability of the fit models to generalize what they have learned to new images. In order to have a better-quality model, we used a data augmentation method called ImageDataGenerator from the Tensorflow/Keras library. It allows us to easily create a wide variety of new data real-time when we are training the model. Therefore, it saves the user from using extra disk space to store all the augmented data.

II. Training and Evaluating the Network

A. Training the Network

We used MobileNetV2 with ImageNet weights as our feature extractor, because it's a light and strong model that can achieve high-level accuracy despite using less computational power and memory [4]. And, since applications such as this one, may have to perform its task using a low power(computational power and memory) device, we decided using MobileNetv2 would be a good idea. Then we stacked dense layers on top the MobileNetV2 according to our classification purposes, this way we formed the model to train our mask detector on as can be seen on Figure 3.

We trained two different models just to see if there is a performance difference between feeding the network with 160x160 and 224x224 image sizes(Both were suggested input sizes on Tensorflow/Keras documentation). As we can see from the Figure 4 they both achieved high accuracy but 224x224 input size performed slightly better.

B. Evaluating the Network

One of our goals is to see how a face detection model that was previously performed better than the state-of art models would scale to the new lifestyle caused by COVID-19. Hence, we decided to use the Joint Face Detection and Alignment model created by using MTCNN(Multi-task Cascaded Convolutional Networks), since it's a robust face detector which performs very well against many other strong competitors [1].

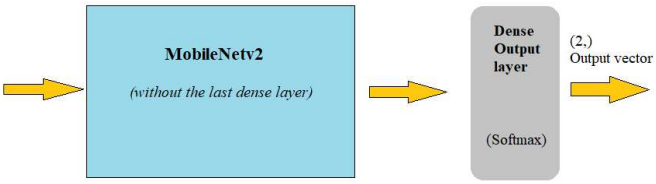
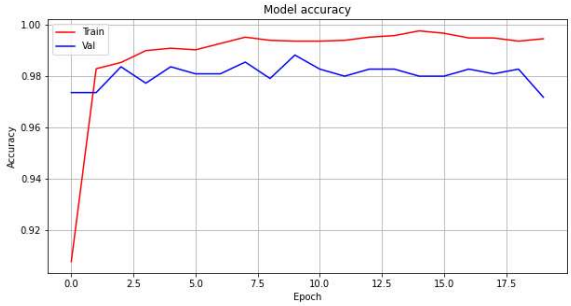
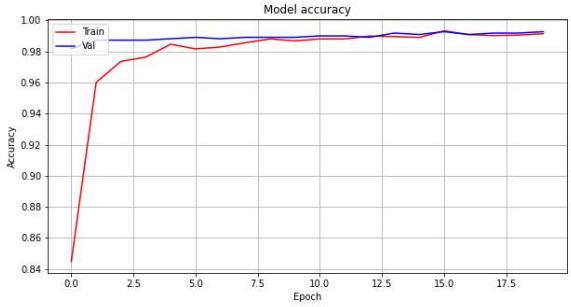


Fig. 3: Simple explanation of connecting pre-trained model with custom classification head



Top: Accuracy plot of 160x160 image size



Bottom: Accuracy plot of 224x224 image size

Fig. 4: Accuracy plots of two different input image sizes used for training

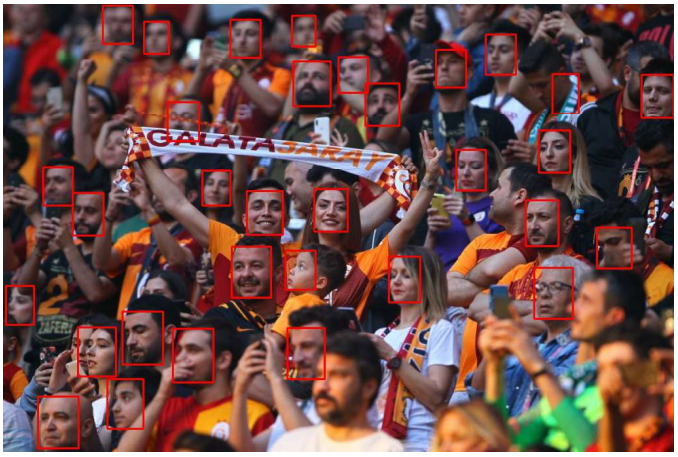
We used a python implementation of MTCNN face detection which comes trained and ready to use as a package [3].

III. Experiments

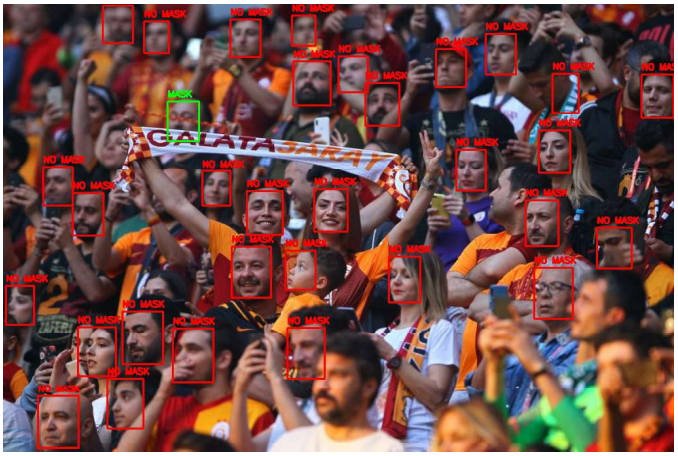
A. Experiments on Images

After initializing the face detector, we tried detecting faces on some example images, and it returned high accuracy results as expected as in the Figure 5:Top. Therefore, we passed to the next stage, which is putting our classification model to the test by extracting the detected faces and sending them to the classifier. As expected, the classifier labeled the faces almost perfectly with just one exception as presented in the Figure 5:Bottom.

1) The Impact of Illumination on Mask Detection: The photo used to create Figure 5:Top and Figure 5:Bottom have been taken under well spread low natural light, and the success



Top: Face detection.



Bottom: Mask detection

Fig. 5: Photo under well-spread, low natural light.

Fig. 6: Photo under well-spread artificial light.

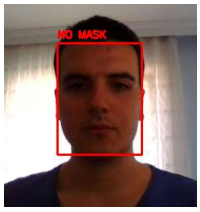
Fig. 7: Photo under bright spotlights.

of the both face detection and mask classifier models are high. The same also can be said for the photo taken under well spread artificial light as we can see in the Figure 6. Successful detection rate was usually high with the images that have similar illumination.

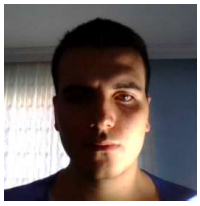
On the other hand, the photo that has been taken under bright spotlights, and also seems to have a slightly altered color palette(i.e. adding filters on certain mobile apps) as you can see in the Figure 7, completely tricked our mask detection model. Most faces have objects in front of them(somebody's arm or shoulder, or the person's own arm, graduation cap etc.) but those objects are not completely blocking the facial features. On top of that, there are faces completely open with slightly tilted heads that are still labeled incorrectly. After encountering many others like this example, we can say that our mask classification model is severely affected by unbalanced illumination.

2) The Impact of Illumination on Face Detection: The illumination difference was also affected the performance of our MTCNN face detector, slightly more than we anticipated. The effect can easily be seen in the Figure 8. In the Figure 8a, even though the lighting condition is not an ideal one, the illumination difference was not extreme between the facial features. Hence, we can see both our face and mask detectors worked nicely. In the Figure 8b however, the illumination difference is quite strong. Half of the face is significantly darker than the other, especially around the eye, and as we can see that the face detector failed to detect the face. We saw the same trend with other examples of strong illumination difference as well.

3) The Impact of Wearing a Mask on Face Detection: With the introduction of wearing masks daily, the face detectors trained previously started performing terribly, since a big portion of the faces covered by masks. We can see the effect of it clearly in the Figure 9 and other similar examples. If we compare Figure 5 and Figure 9, the difference in performance becomes even more clear. Because, both photos have been taken from a similar angle, however, one with masked and the other without masked faces. Some faces can still be detected while wearing a mask but they require optimal conditions(mostly good illumination and head pose). Hence, there is a need of training new face detection models that can



(a) Low lighting difference between facial features



(b) Extreme lighting diffrence between facial features

Fig. 8: Comparison of face detection under two different lighting settings.

detect a face more accurately from the facial features above the nose(i.e. eyes, eyebrows and forehead) than the previous ones.

Fig. 9: Photo of faces with masks from a side angle. This photo proves that how badly covering a big part of the face can affect a robust face detector's performance.

B. Experiments on Videos

On videos, or more precisely the videos captured live from our computer cameras, we had the same problems which we had on images. When illumination, pose and some other parameters(i.e. visibility of the facial features above the nose) all align perfectly, both face and mask detectors work nicely. However, relying on perfect conditions in a real-world is not acceptable, and only proves we need to enhance our methods to keep up with the new lifestyle introduced by COVID-19.

IV. Conclusions and Further Works

MTCNN based face detector usually achieved high accuracy while detecting faces without masks if we exclude some extreme lighting conditions. However, when started testing it on examples of faces with mask, its accuracy dropped significantly. Most of the time, the face detector required near perfect conditions(i.e. illumination and pose) when detecting faces with masks.

To detect faces with masks with a near perfect accuracy like the faces without masks, we may need to design models that are able to detect faces with less facial features. Also, more faces with masks data can be prepared to train better models.

Our mask detection model on the other hand, performed decently as long as there is a detected face while under moderate lighting conditions. However, when we used our mask detector on images that had bright lights, a lot of shadows or more than one light sources producing different kinds of lights(i.e. standard light, colored lights or lights that looked like ultra-violet) our mask detector got confused, and predicted wrong labels. With more faces with masks data to incorporate to training and with stronger hardware(i.e. more ram, stronger GPUs) mask detector model can be trained to be much more consistent even in illumination conditions which are not ideal.

We also believe, if illumination can be handled better, both detectors can do significantly better.

Acknowledgements

The authors would like to thank all our professors that supplied us with all the fundamental knowledge and ideas, Github user X-zhangyang for the dataset, Tensorflow/Keras team for the tutorials and their well-prepared documentation, Github user Iván de Paz Centeno(a.k.a. "ipazc") for his MTCNN face detector implementation and finally the author of "pyimagesearch.com" Adrian Rosebrock for his clear workflow steps we have followed for completing this project.

References

[1] "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks", arXiv:1604.02878.
[2] "Tensorflow webpage", https://www.tensorflow.org/.
[3] "MTCNN by ipazc github webpage", https://github.com/ipazc/mtcnn.
[4] "MobileNetV2: Inverted Residuals and Linear Bottlenecks", arXiv:1801.04381.
[5] "Real-World-Masked-Dataset by X-zhangyang github webpage", https://github.com/X-zhangyang/Real-World-Masked-Face-Dataset.
[6] "Adrian Rosebrock's computer vision webpage", https://www.pyimagesearch.com/.
[7] "Data augmentation explanation", https://bair.berkeley.edu/blog/2019/06/07/data_aug/.