FACE MASK DETECTION

Final Project Report

SUMMARY

Applying deep learning algorithms to detect faces wearing mask in real-time videos and photos.

Resendiz Robles, Ismael

CS7323 – Image Processing & Comp Vision

May 2021

TABLE OF CONTENTS

introduction ***2***

Data preparation ***2***

CNN Design ***3***

CNNs Training ***4***

CNNs Evaluation ***5***

Filter and Refine Approach ***6***

Adaboost **7**

Results **11**

conclusions ***13***

Works Cited ***14***

May 2021

INTRODUCTION

Face masks are a simple barrier to help prevent a person's respiratory droplets from reaching others. Studies show that mask reduces the spray of droplets when worn over the nose and mouth. During COVID-19 pandemic, the World Health Organization (WHO) issued guidelines to wear mask as a normal way of being around other people to help preventing the spread of this disease. While wearing masks alone can't guarantee full proof from COVID-19 exposure, the combination of physical distance, avoiding crews and cleaning hands frequently lower the changes of contagion.

Due to these recommendations and new practices, all in-door public places like grocery stores, shopping malls, movie theaters, etc. require their customers to wear facial mask to enter. However, enforcing to wear mask in large amounts of customers may be a complicated task for a human to do. Considering that most public places count with surveillant systems with cameras, these systems can be repurposed to automate face mask detection in the customers.

The objective of this project is to apply deep learning algorithms to perform face mask detection on real-time videos and photos. Specifically, this project experiments with few Convolutional Neural Networks (CNN) that perform object classification. These networks will be trained using the face mask dataset downloaded from Kaggle. To obtain better performance during the face mask detection, application of the filter-and-refine technique combining with skin detection and cascading. Finally, after training and refining a network that can correctly classify a person wearing a face mask, this will be tested on video and photos.

Previous experiences working with CNNs have showed me significantly slow performance when detecting faces in large photos. Intuitively, I assumed that the predicting model need be enhanced with a filter-refined technique to reduce the number of sub-windows or patches to be processed. Additionally, photos or video frames capture faces within different distances and different orientations, so the model need to consider scaling and/or orientation changes.

One challenge with this problem is the amount of data that is available related to face mask detection is limited or not labelled. So, training a CNN from scratch with insufficient data may not be as robust to work with all kinds of photos and less accurate than expected. In this project, I will explore transfer learning of pre-trained networks to evaluate the accuracy against a simple CCN designed specifically for this problem.

The rest of this document is divided in data preparation section, where I explain the process of extracting faces and face mask from the chosen dataset. The CCN design section talks about the three different CNNs I selected to implement face mask prediction, and a brief summary of their advantages. Followed by the training and testing sections where I show the steps I followed and parameters I used to perform CNN training and evaluation. Next section is about the filter and refine techniques I implemented to help improve the performance, and finally I describe my experiment results and conclusions.

DATA PREPARATION

The dataset was downloaded from the Kaggle website [1]. It contains a set of 25876 pictures of multiple people wearing and not wearing masks. Each picture on the dataset contains a corresponding xml which labels each person face area and the bounding boxes coordinates. There are 3 different classes: wearing masks, not wearing mask and wearing mask incorrectly. For the purpose of this project the third class (wearing mask incorrectly) will be ignored, since the project objective is only to identify if the person is wearing face mask or not.

The first step is to extract the facial patches into a new image from the dataset pictures (dataset author provides a Matlab script to extract the face area). These are organized in separate folders to identify the class the belong to. Folder "1" represents the face class, and folder "2" the face without mask class. These new extracted images have different sizes from 100x100 to 240X240 pixels. The next step is to standardize the image size of the new images to 100x100, so it's consistent with the simple CNN input layer. However, the input layer in the pretrained CNNs vary from 224x224 to 227x227. Fortunately, Matlab has a data augmentation tool to take care of the resizing adequately.
Finally, the total of number of images extracted are 2000 on each class, and 1000 of each class saved for evaluation.

May 2021

CNN DESIGN

The considerations taken to design the CNN were that this should be lightweight to reduce processing overhead, and to have a high accuracy to classify between faces with mask and without mask.

Initially, I designed a simple CNN with a 100x100 input size and with 3 levels. First level starts with a convolutional layer with 8 filters (3x3) followed by a normalization, a max pooling and a ReLU layers. The next 2nd and 3rd levels are composed the same way, except that their convolutional layer has doubled the number of filters, 16 and 32 correspondingly. The CNN continues with a 3 feed forward connected layers and a ReLU layer of sizes 32, 16 and 2. Finally It ends with a Softmax and a classification layer.

FIGURE 1 - SIMPLE CNN MODEL

Sequentially, I selected two pre-trained networks available to use in MatLab: GoogLeNet [2] and SqueezeNet [3] to compare the accuracy and performance against the simple CNN.

GoogLeNet [2] is a 22 layers network that was trained with 1.2 million images and classifies them into 1000 categories. SqueezeNet [3] is a 18 layers network which focuses on improving accuracy with less parameters and less convolutional layers and training with the ImageNet [4] dataset. These two networks highly differ by size and robustness; also, they were trainned to accept input images of 224x224 and 227x227 pixel size correspondingly to classify 1000 different categories. Another purpose of selecting these two networks which highly differ in size, is to compare experiment if they can achieve the same accuracy with less computational resources.

To use transfer learning in Matlab, one can take advantage of the deep neural network designer tool [5]. So, to reuse the training weights on these networks, I used this it to modify the model and accommodate 2 output classes instead of 1000. Basically, the last feedforward and classification layers need to be replaced with a new one. In the case of SqueezeNet [3], the last convolutional layer also needs to be replaced to learn from the different data.

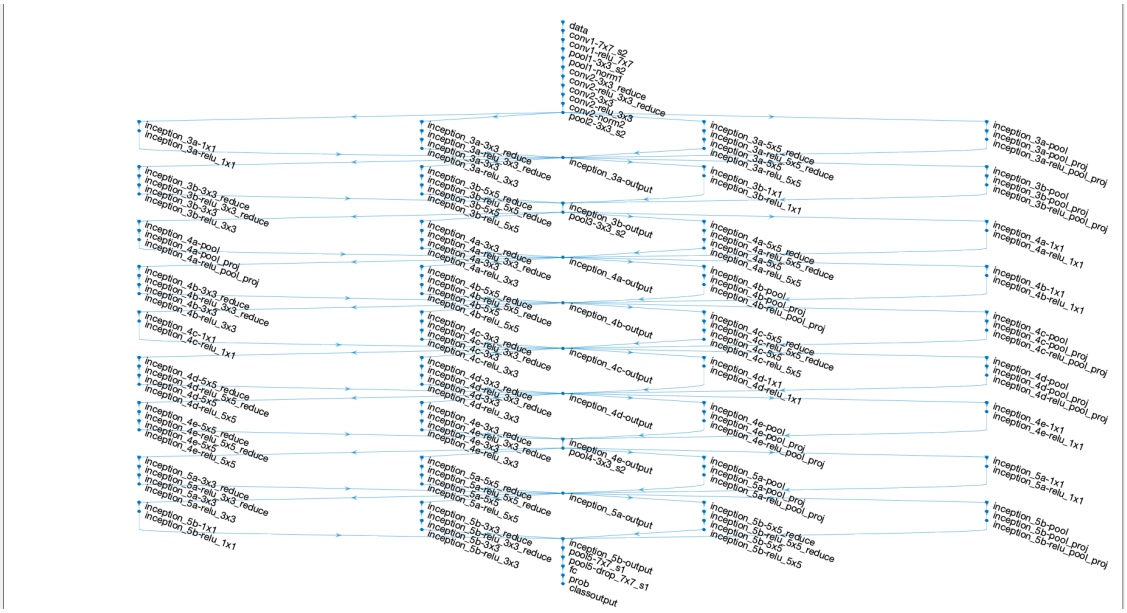


FIGURE 2 - GOOGLENET MODEL

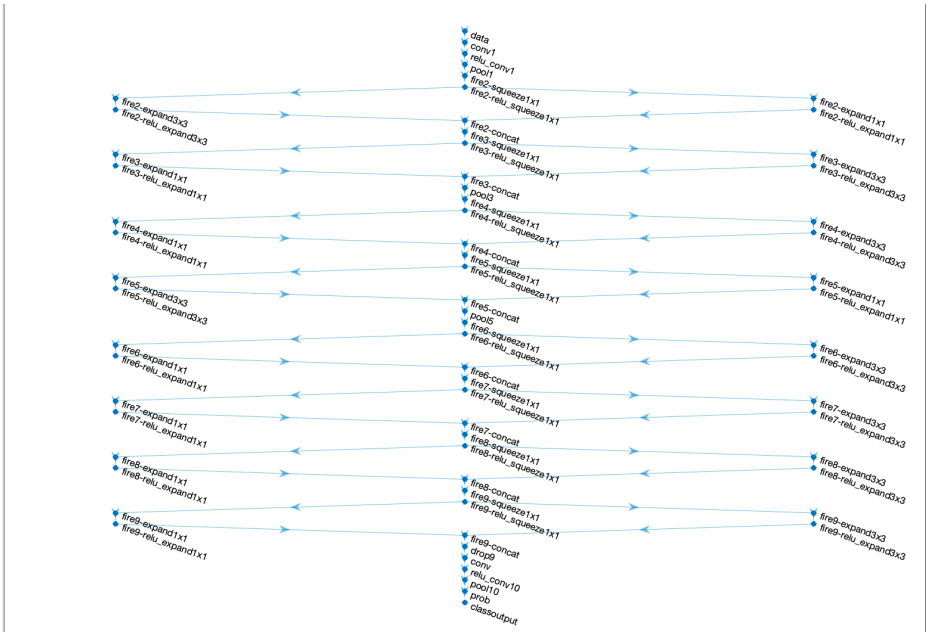


FIGURE 3 - SQUEEZENET MODEL

CNNs Training

All training and evaluations performed in this project were done using a MacBook Pro 2Ghz Intel i7.

To train the simple CNN, I used 2000 images belonging to each class and reserved 1000 per class for testing. Initial learning rate was 0.0001, 30 epochs and 128 samples on each minibatch. The training quicky (within 10 minutes) converged to 99% accuracy.

To re-train GoogLeNet [2] and SqueezeNet [3] models, I used the same dataset and parameters used above for the simple CNN. The training process took around 1 hour for each model and converged to 100% accuracy.

May 2021

CNNs Evaluation

Below are the evaluation results obtained from testing the Simple CNN testing, GoogLeNet [2] and SqueezeNet [3] using the reserved images (1000 per class).

The simple CNN correctly classified 90.3% of the testing images, the largest number of errors came from the false positive in the below confusion matrix (face mask misclassifications as regular face).

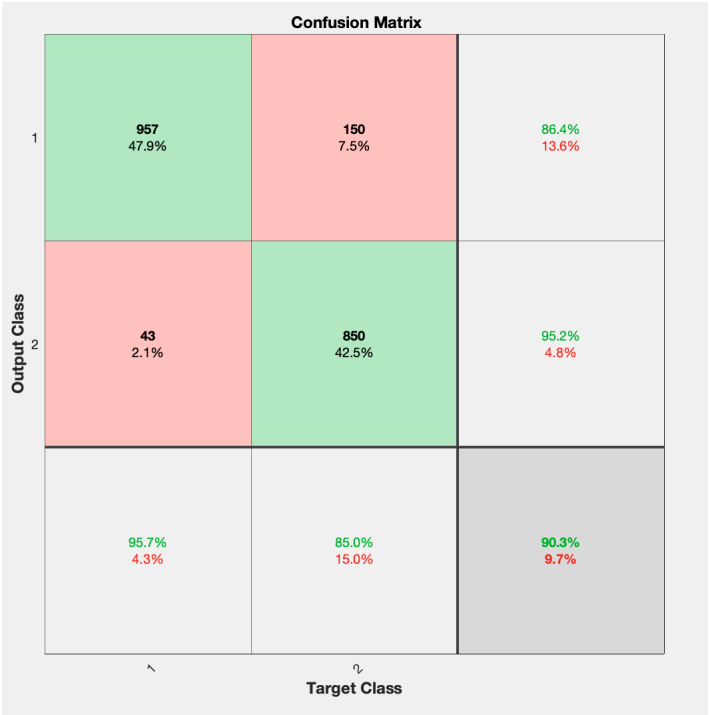


FIGURE 4 - SIMPLE CNN CONFUSION MATRIX

Below figure shows the GoogLetNet [2] confusion matrix, displaying a 99.6% accuracy on the testing images. Clearly outperforms the simple CNN showing a 100% accuracy classifying regular faces and only 8% errors false positive.

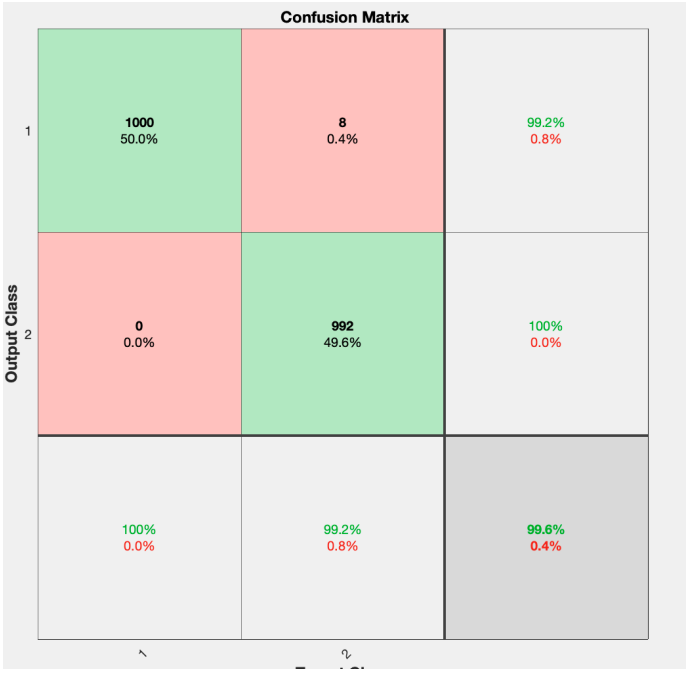


FIGURE 5 - GOOGLENET CONFUSION MATRIX

Below figure shows the SqueezeNet [3] confusion matrix, displaying total accuracy pf 98.2% on the testing images.

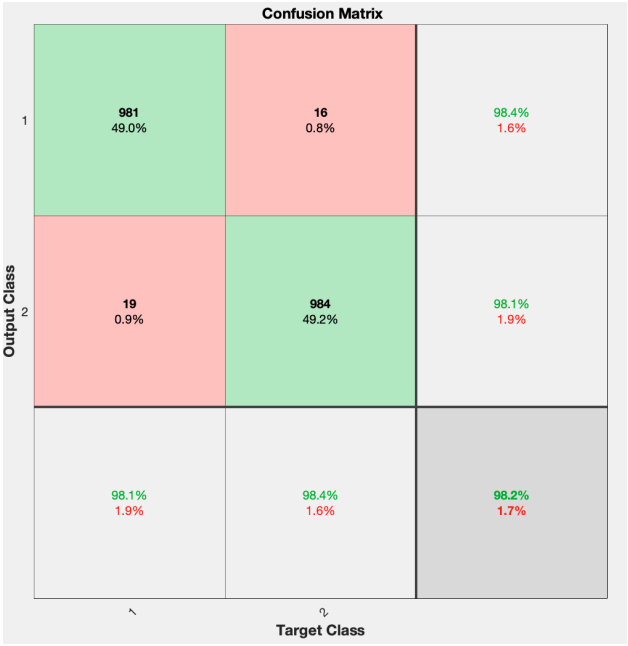


FIGURE 6 - SQUEEZENET CONFUSION MATRIX

Table 1 shows a performance comparison between the three CNNs.

CNN	Single Window performance	Full picture (216x320) performance	Full picture (216x320) performance including skin detection
Simple CNN	0.003200 seconds	12.661954 seconds	2.923809 seconds
SqueezeNet	0.010004 seconds	29.948079 seconds	9.893502 seconds
GoogLeNet	0.018439 seconds	75.939715 seconds	21.863742 seconds

TABLE 1 - CNNS PERFORMANCE COMPARISON

From the CNN performance comparison, we can note the complexity of each CNN is reflected in the time taken to predict the class on a single patch or in a full picture. Furthermore, adding skin detection helps speed up the process since the number of windows to be consider for prediction are reduced by more than 50%. These results lead me to explore other techniques or algorithms I could use to improve the performance without impacting the accuracy. In the next section I describe them.

Filter and Refine Approach

The filter an refine approach is based on two steps, the first one is to apply a fast but less accurate classifier to shortlist a set of candidates, and the second step is to use a more accurate but slower classifier to work on the initial list of candidates and choose the final results.

The common ground of face mask detection and face detection is that both are looking for facial areas within an image. Face detection is a more generic problem that focuses in detecting any kind of face, whether is partially covered or not. So, one instance of the filter and refine approach would be to use a quick face detector that shortlist the face areas boxes and then apply a more robust CNN classifier on them. A high-level diagram of this approach is shown in Figure 7.

FIGURE 7 – FACE MASK DETECTOR USING FILTER AND REFINE APPROACH

At this point, the face mask classifier task can be performed by any of the three CNNs discussed in previous sections, therefore the only missing piece in the puzzle to find is a quick face detector.

ADABOOST

Adaboost is an algorithm which produces strong classifiers by forming a group of weak classifiers and combining them linearly. This algorithm is quick due to the image representation called integral image, allowing us to perform quick computational operations. Adaboost can be tasked to perform face detection, the output of the algorithm would be a set of bounding boxes that can be feed into the face mask classifier.

In order to train Adaboost, I used the final project dataset option 1). It contains 3047 training face images, and I generated a set of 5004 training nonface images. Additionally, there are 770 testing face images, and I generated another set of 770 of nonface images.

1000 weak classifiers were randomly generated to be optimized using the Adaboost algorithm and to create a strong classifier with the combination of the best boosted weak classifiers. Initially, I selected the 50 best boosted weak classifiers, then I evaluate them using the testing dataset. Below is a graph of the strong classifier accuracy combined from 1 to 50.

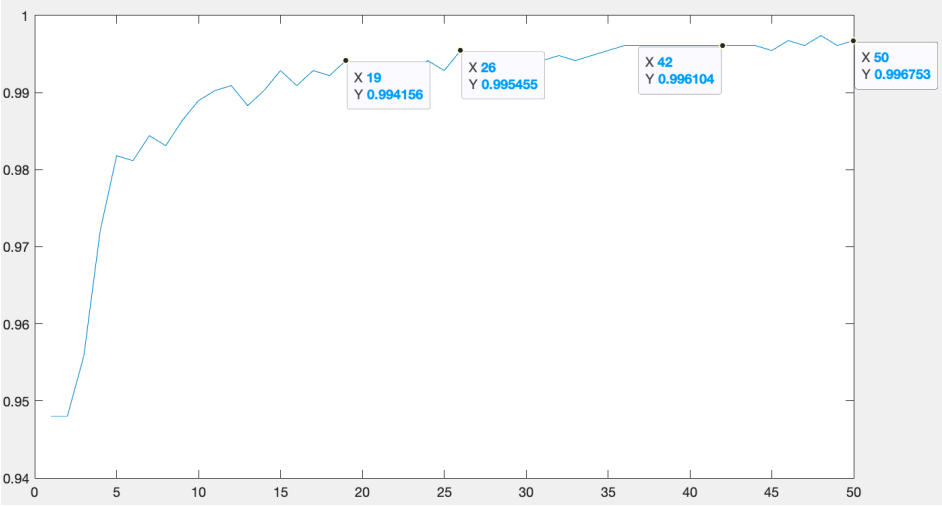


FIGURE 8 - STRONG CLASSIFIER ACCURACY PLOT

As shown in the figure 8, after the 20th boosted classifier, the accuracy oscillates from 0.994 to 0.9967. Overall, I concluded that the from the classifier 19th there is not much difference in accuracy, yielding that a potentially good classifier can be obtained from the first 19 weak boosted classifiers.

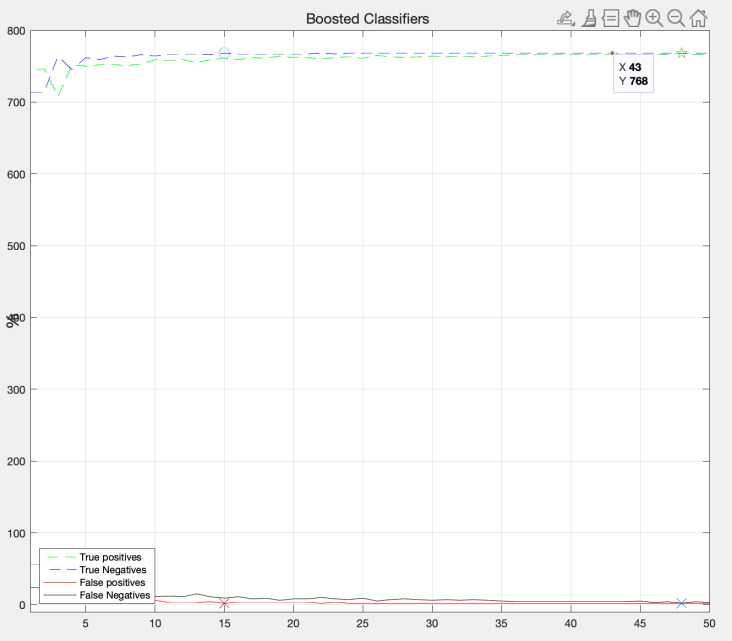


FIGURE 9 - STRONG CLASSIFER CONTINUOS CONFUSION MATRIX PLOT

Another way to view of the strong classifier performance is by looking at the number of errors during the classification. In Figure 9, the plot shows that least number of false positives and max number of true negatives are found by combining the first 15 boosted classifiers.

Now that the candidate to a strong classifier is potentially at the first 19th position, I can seek to optimize the face detection performance by constructing a cascade of classifiers. i.e., a set of strong classifiers that are lined up in stages, each stage is followed by a more complex and accurate classifier that is formed with a greater number of boosted weak classifiers. For example, a cascade based on Figure 8-9 results can be C = [1, 5, 10, 19]. Therefore, the first stage in the cascade would be in charge to discard most of the nonfaces sub-windows, hence in order to reduce the error of rejecting too many possible faces a threshold T_i *(i represents each cascade stage)* must be introduced to control this behavior. To calculate the values of T_i, was done by iterating thought the min and max response values at each individual weak classifier, and by accounting the number of misclassifications at each step. At the end I keep the smallest error and its corresponding threshold. Below is plot show the original boosted classifier accuracy against the cascade classifier applying stage thresholds.

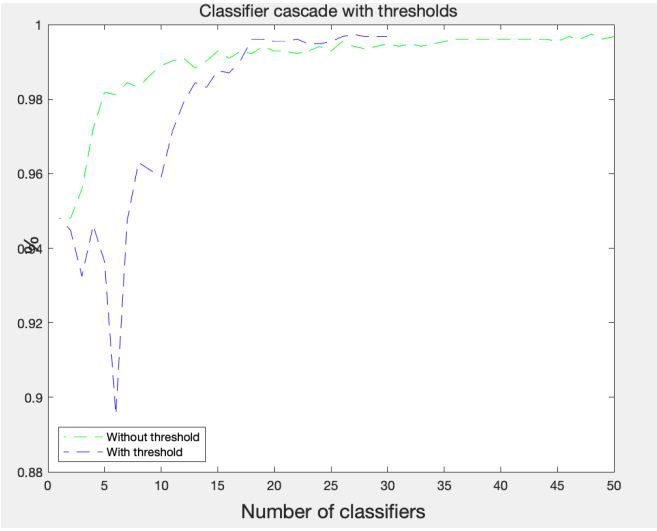


FIGURE 10 - STRONG CLASSIFIER ACCURACY VS CASCADE ACCURACY WITH STAGE THRESHOLD

The effect of the cascade threshold affects the overall accuracy from the 1st to 10th boosted classifiers, however this is expected since each cascade has lowed its criteria to avoid rejecting potential faces. This can be seen in the Figure 11, where I compare a strong

classifier with the first 3 weak boosted classifiers against the against the same classifier applying the threshold. Note, the image on the top accepts more nonfaces (images from 771-1540 are nonfaces) as faces than the figure in the bottom.

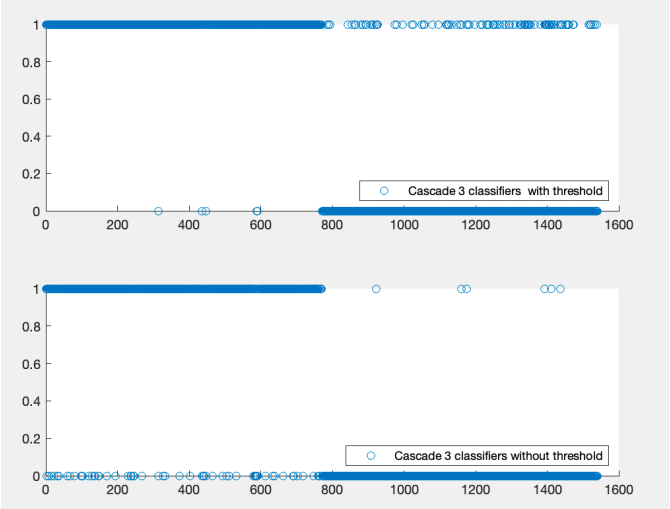


FIGURE 11 - STRONG CLASSIFIER 3 COMPARISON WITH AND WITHOUT THRESHOLD

To evaluate the performance, I compared cascade classifier *C* against a strong classifier composed of 19 weak classifiers. This evaluation was performed on the same photo. Also noted that adding more than 20 classifiers in the cascade, does not improve the accuracy but affects the performance, since there are more calculations to perform.

Classifier	Time Taken in full picture (375x300)
Strong Classifier (first 19)	2.094030 seconds
Cascade (*C*)	0.435493 seconds
Cascade [1, 5, 10, 20]	0.559430 seconds
Cascade [1, 5, 10, 50]	1.090011 seconds

TABLE 2 - STRONG CLASSIFIER VS CASCADE CLASSIFIER PERFORMANCE COMPARISON

Table 2 shows that the inclusion of cascade technique helps positively to speedup the performance. Additionally, further improvement can be incorporated by including skin detection to ignore window patches that does not contain a certain percentage of skin. To determine the percentage amount of skin on each window I used the integral image method to avoid executing summations which can affect the performance.

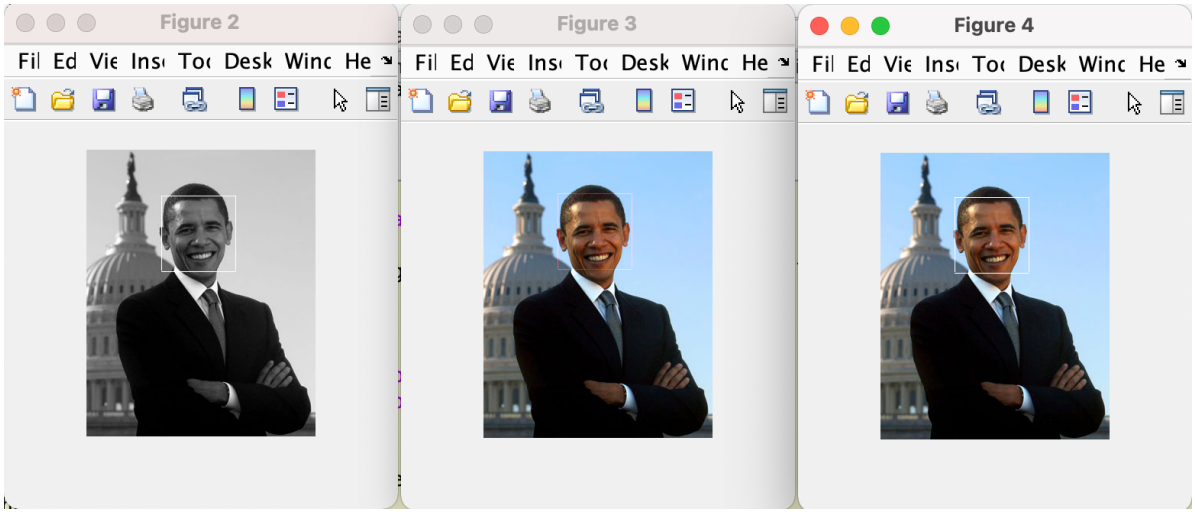


FIGURE 12 - RESULTS OBTAINED USING STRONG CLASSIFER (19 WEAK CLASSIFIERS), CASCADE CLASSIFER(C) AND CASCADE CLASSIFIER WITH SKIN DETECTION.

Figure 12 shows the results of using skin detection using a 39% threshold (far most right picture) does not affect face detection, in fact it helps improve the performance from 0.435493 to 0.032311 seconds. In some scenarios, skin detection helps the face detector to

focus on the windows patches where the skin percentage is higher and by discarding those which have a high score(misclassifications) but don't have at least the skin percentage indicated in the threshold.

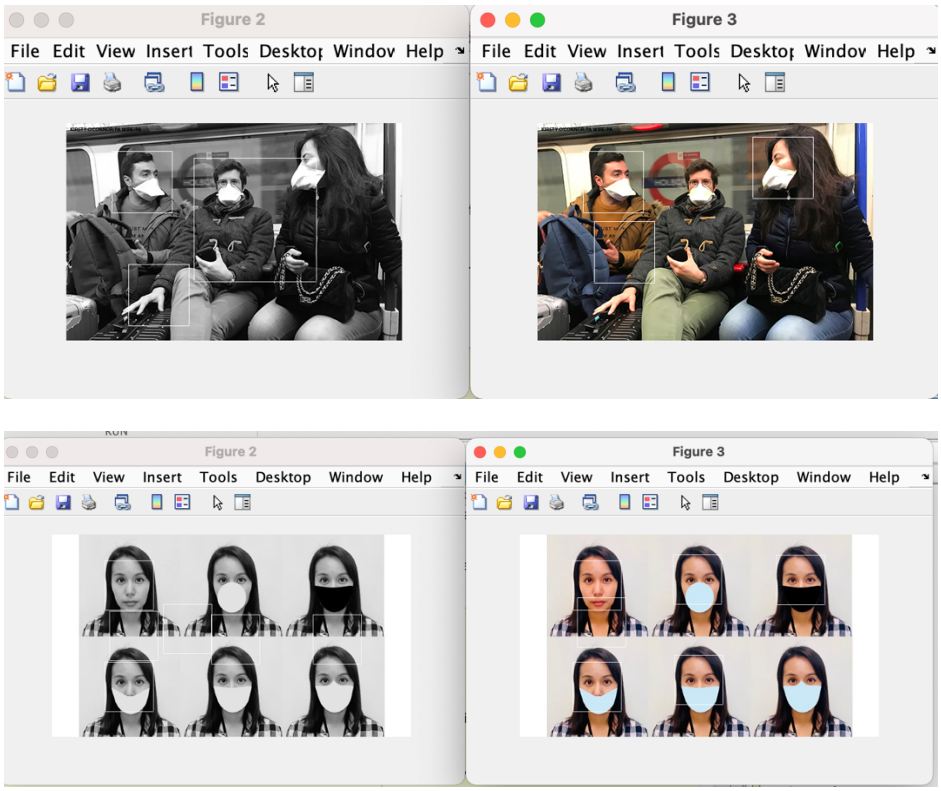


FIGURE 13 - SKIN DETECTION POSITIVELY HELPS THE FACE DETECTOR IN PHOTOS.

There are also some cases where the face detector does not properly identify all the faces in a photo, especially those which have multiple faces, different sizes or there is background noise produced by the sunlight or shades.

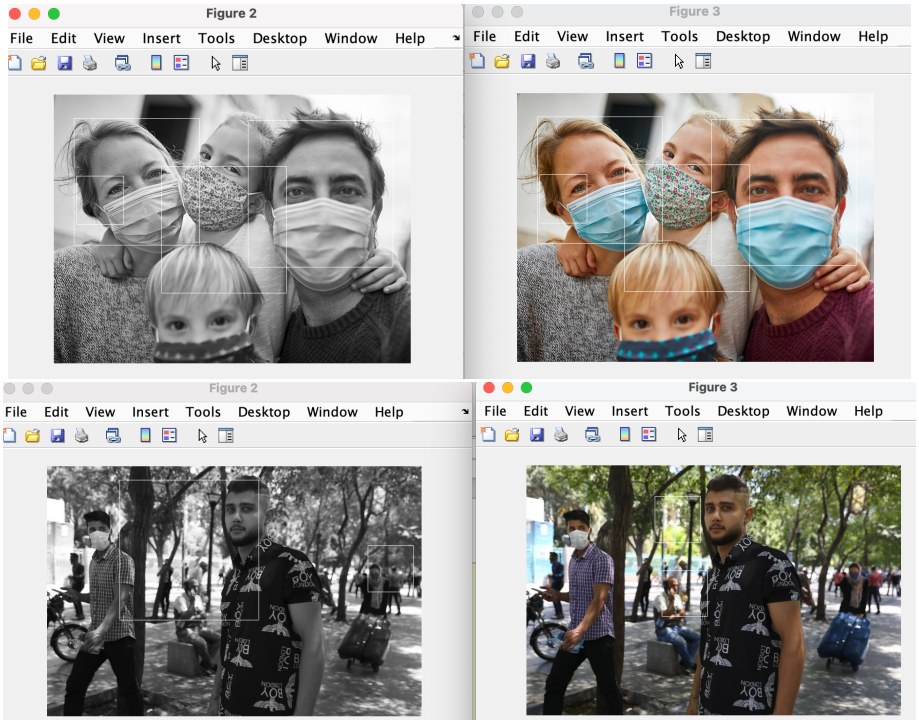


FIGURE 14 - FACE DETECTOR RESULTS ON CROWDED PHOTOS

RESULTS

To evaluate the overall performance of the face mask detector (face detector + face mask classifier), I selected 100 random photos of multiple people wearing a not wearing masks from a different face mask dataset in Kaggle [6]. This is dataset only contains 853 photos and its distribution contains the labels and bounding boxes of the corresponding face mask or regular face.

Face mask detector Type	Number of Faces	Number of face masks
Ground Truth	14	117
Simple CNN	94	37
SqueezeNet	52	79
GoogLeNet	34	97

TABLE 3 - FACEDETECTOR EVALUATION

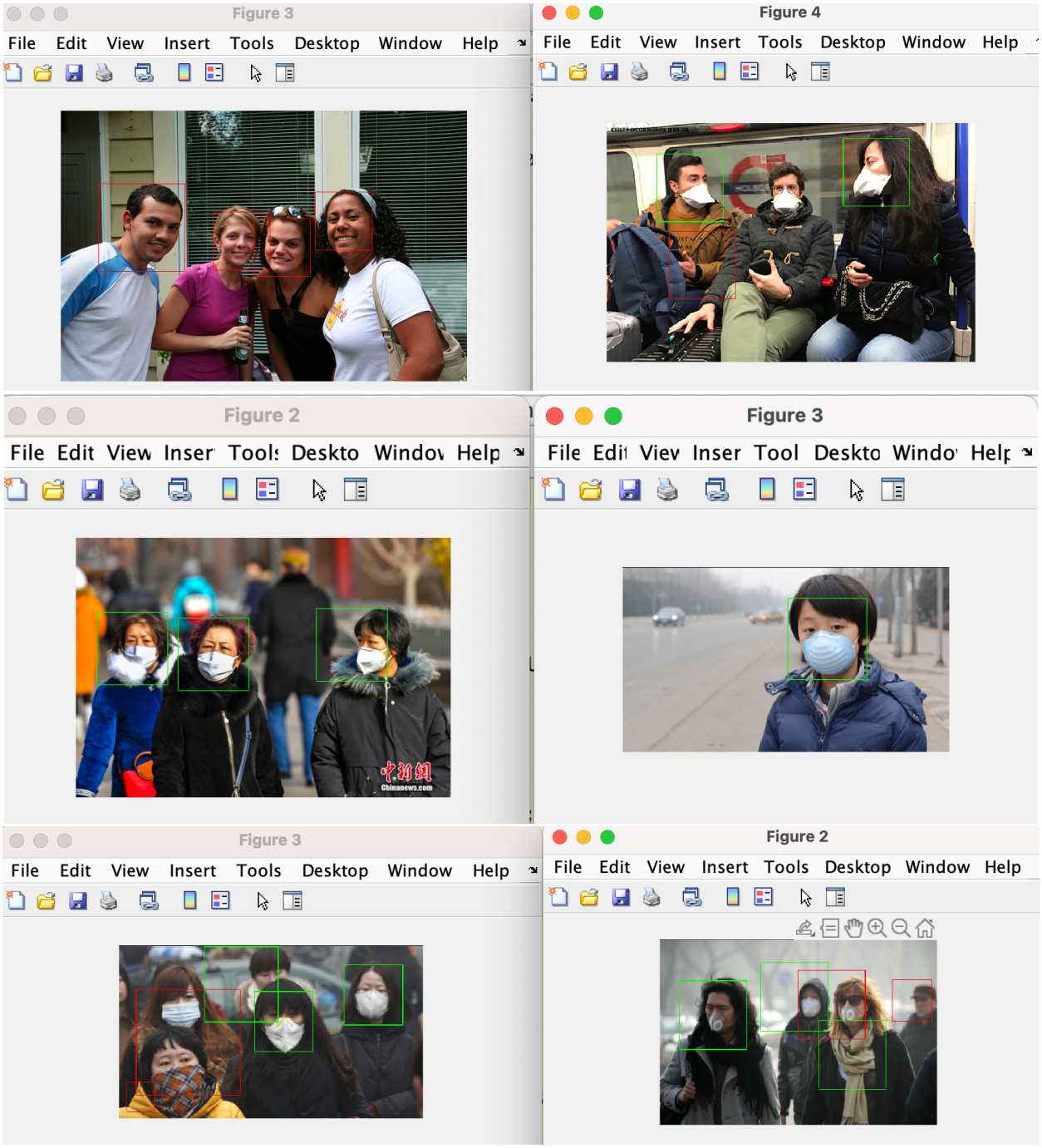
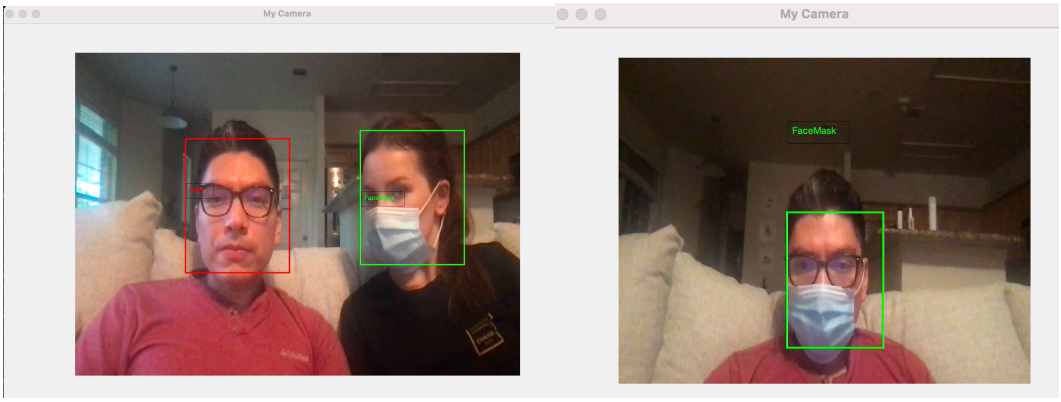


FIGURE 15 – FACE MASK DETECTOR SAMPLE RESULTS ON PHOTOS (FROM TOP TO BOTTOM: SIMPLE CNN, SQUEEZENET, GOOGLENET)

May 2021

As mentioned in the beginning of this report, another purpose of this project is to experiment real-time video. Since videos produce multiple frames per second the face mask detector must work in a fraction of a second to show results in real-time that does lapse more than the video frame sequences. Figure 15 show some result obtained applying the detector on the webcam.



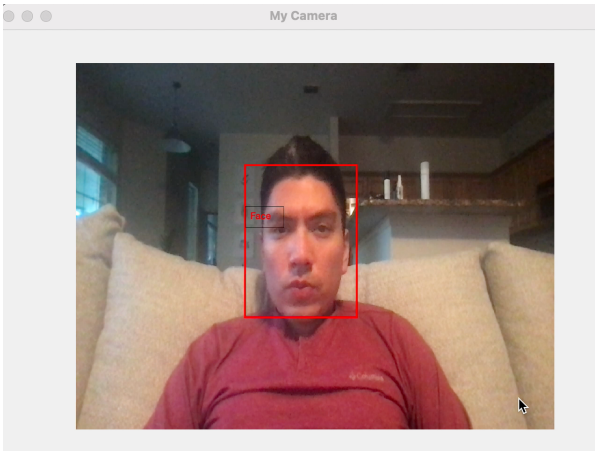


FIGURE 16 – FACE MASK DETECTION APPLIED ON A WEB CAMERA

CONCLUSIONS

In this project I explored variations of the CNN capabilities to construct a face mask detection with a small number of data samples. I showed the advantages of using a pre-trained CNN like GoogLeNet [2] or SqueezeNet [3] are that the classifier accuracy is noticeable higher than a simple CNN when applied with a dataset that size. There are also disadvantages, like the training time and the slower performance. In general, there has been a lot of research time put into those pre-trained networks that we can use to apply to solve multiple computer vision problems. I also explored multiple filter-and-refine techniques to help speed the face detection process. Adaboost algorithm is in fact quick and accurate, that in combination with a cascade of classifiers and skin detection composed a modest face mask detector.

There are multiple areas of improvement in the face mask detection problem. As seen in the experiment results, the face detector has problems classifying multiple faces with different sizes and orientations, also the face mask detector may not work very well in low resolution pictures. Regarding processing speed, I demonstrated that applying multiple computer vision techniques in conjunction with the machine learning algorithms we can improve the performance. However, further improvement is required in order to apply this face mask detector in real-time videos to keep up with the frame bitrate.

May 2021

WORKS CITED

[1] Kaggle, 2020. [Online]. Available: https://www.kaggle.com/rahulmangalampalli/mafa-data.

[2] G. inc, "Going deeper with convolutions," 2014. [Online]. Available: https://arxiv.org/pdf/1409.4842v1.pdf.

[3] S. University, 2014. [Online]. Available: https://arxiv.org/pdf/1602.07360.pdf.

[4] ImageNet. [Online]. Available: https://www.image-net.org.

[5] Matlab, "Network Designer Tool," [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/squeezenet.html.

[6] Kaggle. [Online]. Available: https://www.kaggle.com/andrewmvd/face-mask-detection.