HAAR CASCADES ON FACE MASK DETECTION

Chinmay Patil

College of Computing and Digital Media, DePaul University

Abstract

Coronavirus disease 2019 has affected the world seriously. One major protection method for people is to wear masks in public areas. Furthermore, many public service providers require customers to use the service only if they wear masks correctly. However, there are only a few research studies about face mask detection based on image analysis. Object detection is an important feature of computer science. The benefits of object detection are however not limited to someone with a Doctor of Informatics. Instead, object detection is growing deeper and deeper into the common parts of the information society, lending a helping hand wherever needed. This paper will address one such possibility , namely the help of a Haar-cascade classifier.

1.Introduction

The situation reports 96 of world health organization (WHO) presented that coronavirus disease 2019 (COVID-19) has globally infected over 2.7 million people and caused over 180,000 deaths. In addition, there are several similar large scale serious respiratory diseases, such as severe acute respiratory syndrome (SARS) and the Middle East respiratory syndrome (MERS), which occurred in the past few years. Therefore, more and more people are concerned about their health, and public health is considered as the top priority for governments. Now, WHO recommends that people should wear face masks if they have respiratory symptoms, or they are taking care of the people with symptoms . Furthermore, many public service providers require customers to use the service only if they wear masks . Therefore, face mask detection has become a crucial computer vision task to help the global society, but research related to face mask detection is limited. Face mask detection refers to detect whether a person wearing a mask or not and what is the location of the face . The problem is closely related to general object detection to detect the classes of objects and face detection is to detect a particular class of objects.

A more sophisticated method is therefore required. One such method would be the detection of objects from images using features or specific structures of the object in question. However, there was a problem. Working with only image intensities, meaning the RGB pixel values in every single pixel in the image, made feature calculation rather computationally expensive and therefore slow on most platforms. This problem was addressed by the so called Haar-like features, developed by Viola and Jones based on the proposal by Papa Georgiou et. al in 1998. A Haar-like feature considers neighboring rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. This difference is then used to categorize subsections of an image. An example of this would be the detection of human faces. Commonly, the areas around the eyes are darker than the areas on the cheeks. One example of a Haar-like feature for face detection is therefore a set of two neighboring rectangular areas above the eye and cheek regions.

Fig -1 : Images in Facemask Dataset

In this paper , I proposed a face mask detector based on haar-features, which can detect face masks and contribute to public health care. The system detects masks by moving window over the image. Each Stage classifier labels the specific region defined by the current location of the window as either positive or negative – positive meaning that an object was found or negative means that the specified object was not found in the image. If the labelling yields a negative result, then the classification of this specific region is hereby complete, and the location of the window is moved to the next location. If the labelling gives a positive result, then the region moves of to the next stage of classification. The classifier yields a final verdict of positive, when all the stages, including the last one, yield a result, saying that the object is found in the image. A true positive means that the object in question is indeed in the image and the classifier labels it as such – a positive result. A false positive means that the labelling process falsely determines that the object is in the image, although it is not. A false negative occurs when the classifier is unable to detect the actual object from the image and a true negative means that a nonobject was correctly classifier as not being the object in question. In order to work well, each stage of the cascade must have a low false negative rate, because if the actual object is classified as a non-object, then the classification of that branch stops, with no way to correct the mistake made. However, each stage can have a relatively high false positive rate, because even if the n-th stage classifies the non-object as actually being the object, then this mistake can be fixed in n+1-th and subsequent stages of the classifier

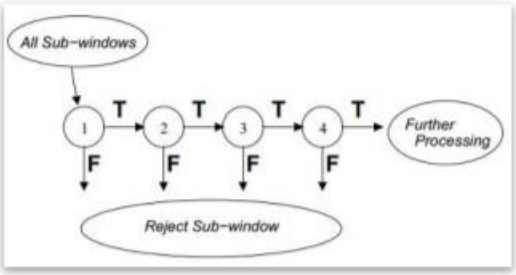


Fig 2: Stages of the cascade Classifier

2.Related Work

2.1 Feature Detection

2.1.1 Harris-Corner Detection

Feature selection is one of the major steps in the object detection approach. In this paper I used three techniques to detect features. First of them is Harris-corner detection. t basically finds the difference in intensity for a displacement of (u,v) in all directions. This is expressed as below:

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} \underbrace{[I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}]^2$$

Fig 3

Fig2. Harris Corner Detection

Harris corner detected the corner features like the thread of mask and the outer boundary of the mask whereas other detected points can be referred as false positive or noise for our application result.

2.1.2 Good Features to Track.

This is another method proposed by Jinabo Shi and Carlo Tomasi. The scoring function is given by

$$R = min(\lambda_1, \lambda_2)$$

Fig 4

If it is a greater than a threshold value, it is considered as a corner. In Image you can see the detected features are shown by red dots which covers the boundaries of the face mask as well as the folds and some noise. This produces better result than Harris-corner detection.

2.1.3 ORB (Oriented FAST and Rotated BRIEF) feature detection.

It computes the intensity weighted centroid of the patch with located corner at center. The direction of the vector from this corner point to centroid gives the orientation. To improve the rotation invariance, moments are computed with x and y which should be in a circular region of radius r, where r is the size of the patch. In this application it gives following result.

Fig 5

This is ORB Feature detector with 50 key-points tracked, as it detects all the facial mask corners and the boundaries as well as the eyes which is considered as noise.

Here is the comparison of speeds of all the feature detection approaches used and other available.

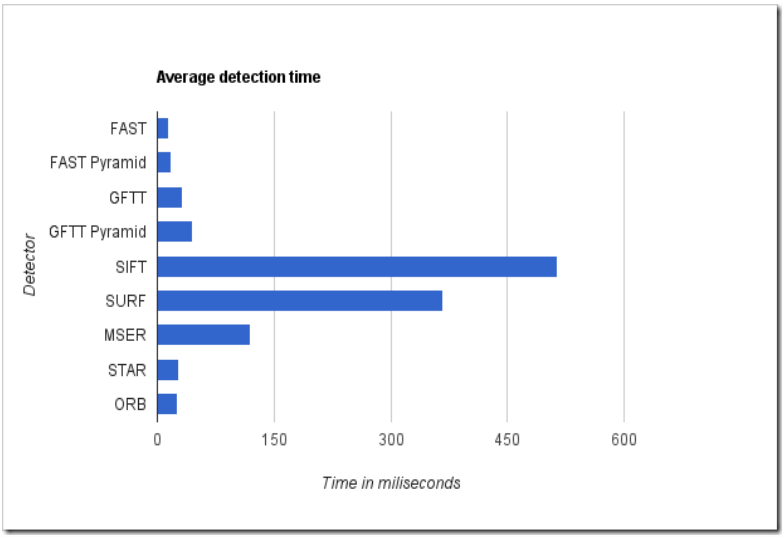


Fig 6

3. Methodology

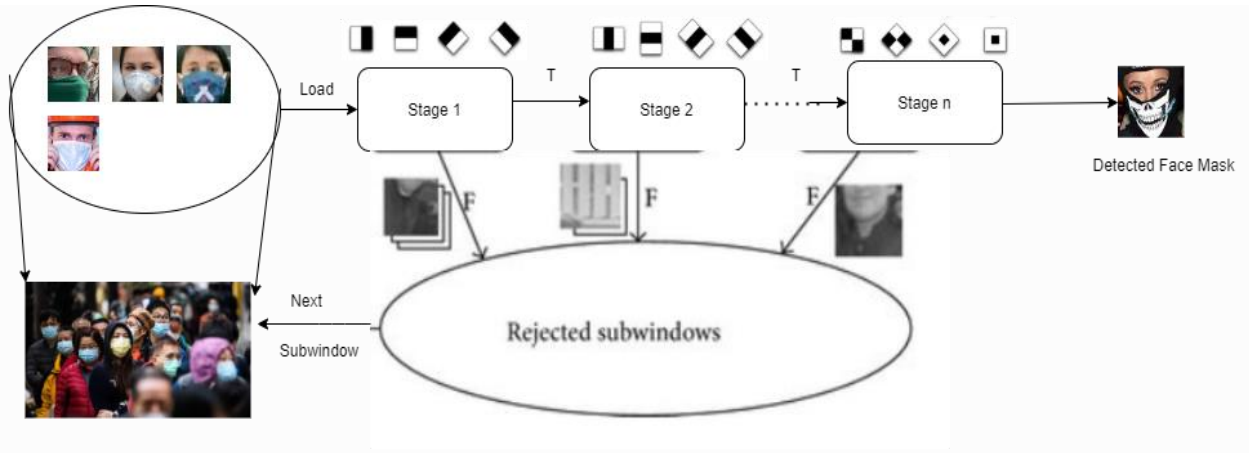


Fig 7:Architecture

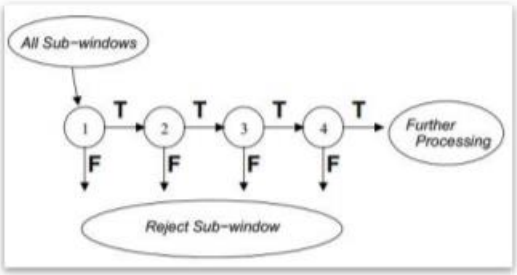


Fig 8 :Stages of Cascade Classifier

The training of cascade proved to be no easy task. The first necessary bit was to gather the images, then create samples based on them and finally starting training process. The opencv traincascade utility is an improvement over its predecessor in several aspects, one of them being that traincascade allows the training process to be multithreaded, which reduces the time it takes to finish the training of the classifier. This multithreaded approach is only applied during the pre-calculation step however, so the overall time to train is still quite significant, resulting in hours, days, and weeks of training time. Since the training process needs a lot of positive and negative input images, which may not always be present, then a way to circumvent this is to use a tool for the creation of such positive images. OpenCV built in mode allows to create more positive images with distorting the original positive image and applying a background image. However, it does not

allow to do this for multiple images. Another important aspect to consider is the number of positives and negatives. When executing the command to start training, it is required to enter the number of positive and negative images 9 that will be used. Special care should be taken with these variables, since the number of positive images here denotes the number of positive images to be used on each step of the classifier training, which means that if one were to specify to use all images on every step, then at one point the training process would end in an error. This is due to the way the training process is set up, as described in section 1. The process needs to use many different images on every stage of the classification and if one were to give all to the first stage, then there would be no images left over for the second stage, thus resulting in an error message. The training can result in many types of unwanted behavior. Most common of these is either overtraining or undertraining of the classifier. An undertrained classifier will most likely output too many false positives, since the training process has not had time to properly determine which is positive and which is not.

3.1 Training Parameters:

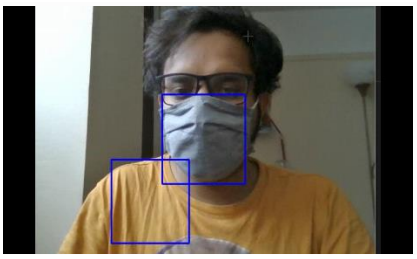
Models are trained using the parameters:

1. Number of positive samples used in training for every classifier stage.
2. Number of negative samples used in training for every classifier stage.
3. Number of cascade stages to be trained.
4. Size of buffer for precalculated feature values (in Mb)
5. Size of buffer for precalculated feature indices (in Mb)
6. Maximum number of threads to use during training. Notice that the actual number of used threads may be lower, depending on your machine and compilation options.
7. This argument is used to determine how precise your model should keep learning and when to stop.
8. Type of stages.
9. Type of features: HAAR - Haar-like features, LBP - local binary patterns.
10. Width of training samples (in pixels). Must have the same value as used during training samples creation
11. Height of training samples (in pixels). Must have the same value as used during training samples creation
12. Minimal desired hit rate for each stage of the classifier
13. Maximal desired false alarm rate for each stage of the classifier
14. Maximal depth of a weak tree.
15. Maximal count of weak trees for every cascade stage

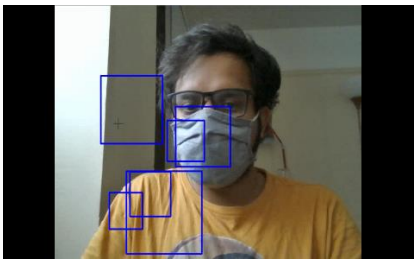
Results:

Using different parameters for each model gives different results. The opposite effect may be observed if too many stages are trained, which could mean that the classification process may determine that even the positive objects in the picture are actually negative ones, resulting in an empty result set Fairly undefined behavior can occur if the number of input images are too low, since the training program cannot get enough information on the actual object to be able to classify it correctly. One of the best results obtained in the course of this work is depicted on result 5. As one can observe, the classifier does detect some masks without any problems, but unfortunately also some false positives are also classified as a mask. The time taken to train the classifier to detect at this level can be measured in days, rather than hours. The results of experiments are shown by the images below.

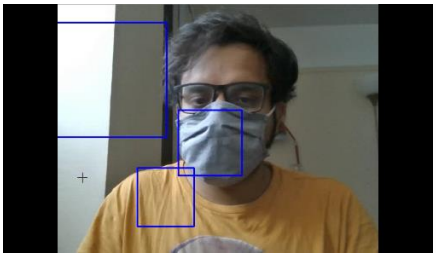
Model results:



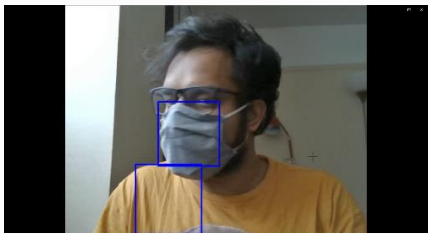
Result : 1



Result: 2



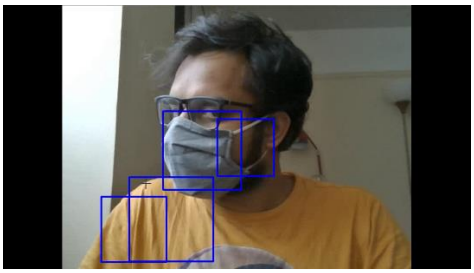
Result :3



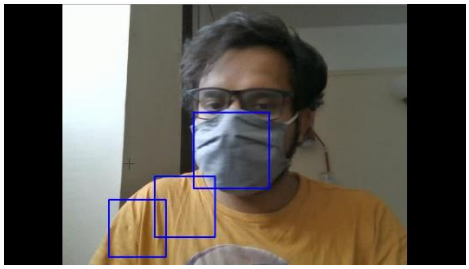
Result:4

Result:5

Result:6

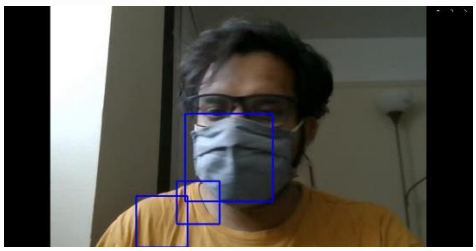


Result:7



Result:8

Result:9



Result:10

Performance Comparison

The performance of the face masks model is compared by applying different parameters. Below table shows the comparison between models and the parameters applied to them

Cascades	Min Neighbour	Scale Factor	Positive Images	Negative Images	Num Stages	Min hit Rate	Max False
Result 1	9	1.1	400	600	10	1	0.1
Result 2	9	1.1	400	600	10	0.999	0.1
Result 3	9	1.1	400	600	12	0.999	0.2
Result 4	9	1.1	400	1000	12	1	0.1
Result 5	9	1.1	400	800	12	1	0.2
Result 6	9	1.1	400	6000	12	1	0.1
Result 7	9	1.1	400	900	10	0.999	0.1
Result 8	9	1.1	400	900	12	1	0.1
Result 9	9	1.1	400	1000	12	1	0.1
Result 10	9	1.1	400	1000	12	1	0.1

Table 1

Conclusion:

In this paper, I proposed a face mask detector using haar-cascades. The architecture of the haar cascades is build upon the number of positive and negative images. The filters are applied to image exactly as the convolutional kernel. Each feature value is obtained by subtracting sum of pixel under white rectangle from the sum of pixel under the black rectangle. Having number of negative images twice the number of positive images gives less possibility of false positives, maximizing the false alarm rate while keeping minimum hit rate to 1 makes bounding box accurate and less possibility of the false positive.

References:

1. Md. Sabbir Ejaz*, And Md. Rabiul Islam “Masked Face Recognition Using Convolutional Neural Network “
2. Amey Kulkarni, Amulya Vishwanath And Chintan Shah “Implementing A Real-Time, Ai-Based, Face Mask Detector Application For Covid-19”.
3. Mohamed Loey, Gunasekaran Manogaran, Mohamed Hamed N. Taha ,Nour Eldeen M. Khalifa “A Hybrid Deep Transfer Learning Model With Machine Learning Methods For Face Mask Detection In The Era Of The Covid-19 Pandemic”
4. Toshanlal Meenpal, Ashutosh Balakrishnan, Amit Verma” Facial Mask Detection Using Semantic Segmentation”
5. Mingjie Jiang , Xinqi Fan , Hong Yan” Retinafacemask: A Facemaskdetector”
6. Shiming Ge, Jia Li, Qiting Ye, Zhao Luo” Detecting Masked Faces In The Wild With Lle-Cnns”
7. Sander Soo” Object Detection Using Haar-Cascade Classifier”
8. Https://Docs.Opencv.Org/3.4/Dc/D88/Tutorial_Traincascade.Html
9. Https://Computer-Vision-Talks.Com/2011-07-13-Comparison-Of-The-Opencv-Feature-Detection-Algorithms/
10. Jianbo Shi , Carlo Tomasi “Good Features To Track”
11. Https://Medium.Com/Datadriveninvestor/Haar-Cascade-Classifiers-237c9193746b