PROJECT REPORT
PROJECT 2

Anurag Pant
UID: 705085298

Introduction

In this project, we perform face detection on an image using boosting technique. Boosting is a general method for improving the accuracy of any given learning algorithm. To perform this technique, we select Haar filters which we convert to weak classifiers. These weak classifiers are then combined to form a strong classifier which can be used for the purpose of face detection in the image. There are 2 types of boosting that are used in this project:

1. AdaBoost
2. RealBoost

1. Converting Haar Filters into Classifiers

We load each generated Haar filter and calculate the activation values over the integrated images for each Haar filter using the function to calculate training activations. These activation values are then loaded into objects of the weak classifier class (each Haar filter is loaded into a different weak classifier object). Now using the train function, we access each weak classifier and calculate the minimum and maximum of the its activation values. After generating the range of the activation values, we use it to determine the threshold for each weak classifier by taking a specified number of equidistant points within the range and trying to find the best threshold value (picking the one that gives the minimum weighted error) from among them. If any threshold value generates an error of greater than 0.5, we can subtract the error from 1 and take the polarity at that particular threshold to be -1, else we take the polarity of that threshold to be 1. By doing so we can pick the weak classifier that has the minimum error and obtain its polarity and threshold as well.

2. Implementing AdaBoost

We start off by generating a number of Haar filters and calculating the activations of the integral images over these filters, as we had discussed above. Then we load the training dataset which consists of face and non-faces. We also create corresponding labels to go along with this training dataset that will inform us whether a particular image is a face or a non-face.

Now we have to train the AdaBoost model. This basically involves choosing a bunch of weak classifiers to form a strong classifier that will be able to accurately detect faces within the test image. To choose the weak classifiers, we have to first set the data weights of each data point (each image in the training dataset). Initially, the data weights are set to equal normalized values. Then we call the calculate error function that converts each Haar filter at each step to a weak classifier by determining its minimum error, threshold and polarity (which was discussed

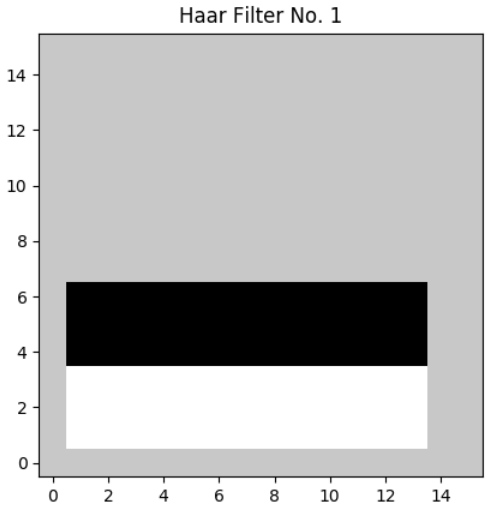
above). Since, the calculate error function is called numerous times throughout the training procedure and since, it itself takes time to find the suitable threshold value for each Haar filter within each step, we need to minimize the overall time complexity. In order to accomplish this, we vectorize the entire calculation that takes place within the calculate error function, eliminating the need for any looping. Moreover, we make use of parallel computing to call this function from within the train function to further speed up the process. Once we have the minimum error for each of the weak classifiers, we find the best weak classifier within that particular step which will help us to classify the images in the best possible way. Now we use this minimum error from among all the weak classifiers (that corresponds to the best weak classifier) to find the value of alpha at this particular iteration. The value of alpha along with the predictions made by the weak classifier and the data labels are used to update the data weights at each iteration.

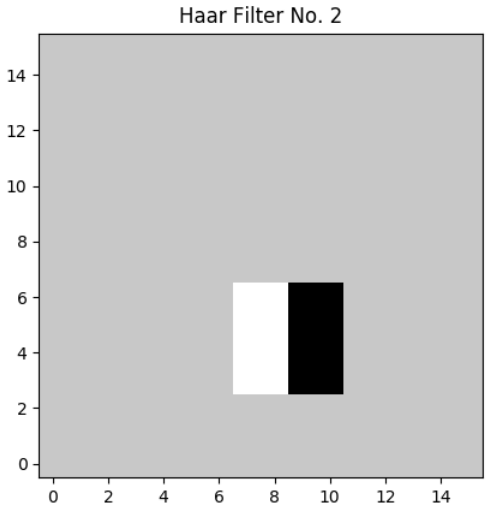
Finally, we append the value of alpha and the chosen weak classifier to the list of chosen weak classifiers. We continue repeating this process of choosing weak classifiers until we obtain the required number of weak classifiers to form the strong classifier.

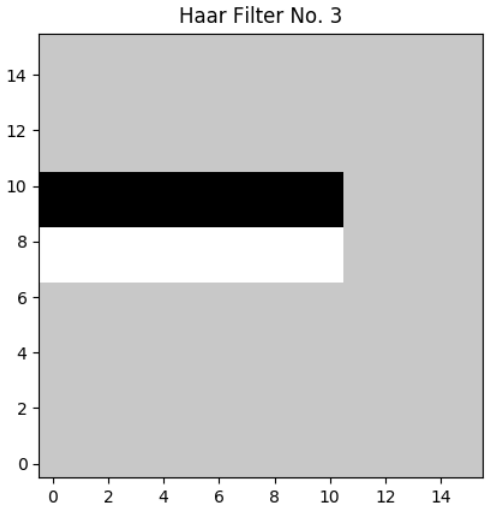
Now we make use of the strong classifier to find the faces in the test image. We call the face detection function which breaks the test image up into patches of different sizes. The strong classifier function is called over all these patches to detect patches that look similar to faces. After detection of faces is complete, we run non-maximum suppression to get rid of detected patches that have an overlap area which is greater than the specified overlap threshold, which is 0.01 (only the patch with the higher score is retained while the other patch is deleted). I also make use of a score threshold, which ensures that only detected patches which have a score of more than 30% of the maximum score are returned.

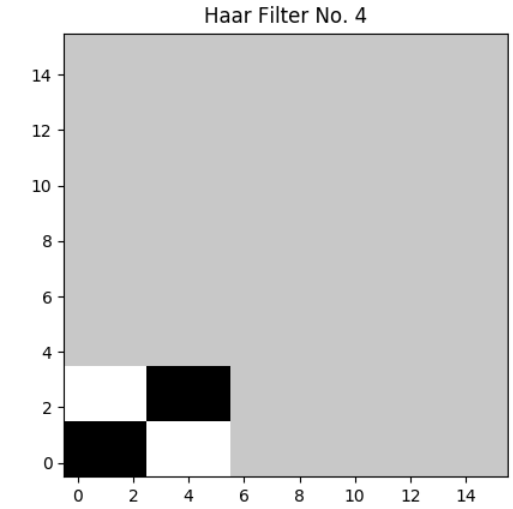
Now we can see the patches that have been detected as faces within the test image. However, this image contains too many detected patches that are actually non-faces. To get rid of these extra detected patches, we perform hard negative mining on the non-face testing images. We repeat the process of face detection on these non-face test images. Any detected patches within these images are actually non-faces. We add these detected patches as non-faces to the training data after converting them into the 16x16 training data size. We then repeat the entire process of activation calculation and training of the AdaBoost model. Now we run face detection again on the test image. The final result turns out to be much more accurate and only detects the actual faces in the test image.

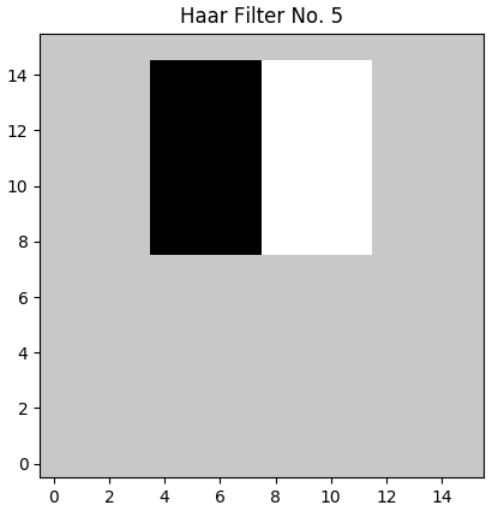
1. Below are the top 20 Haar filters that were used as weak classifiers in the AdaBoost model. They have been named according to their corresponding voting weights:

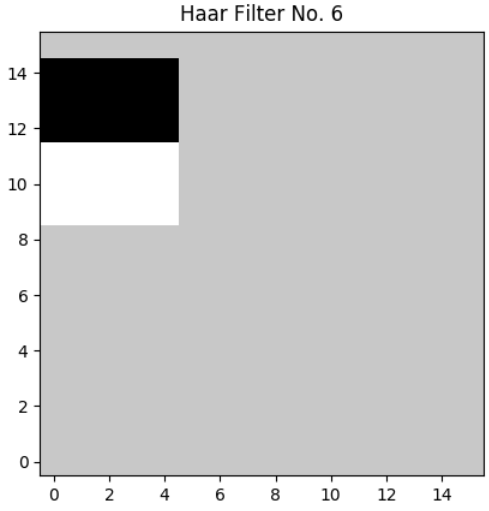

Haar Filter No. 1
0
2
4
6
8
10
12
14

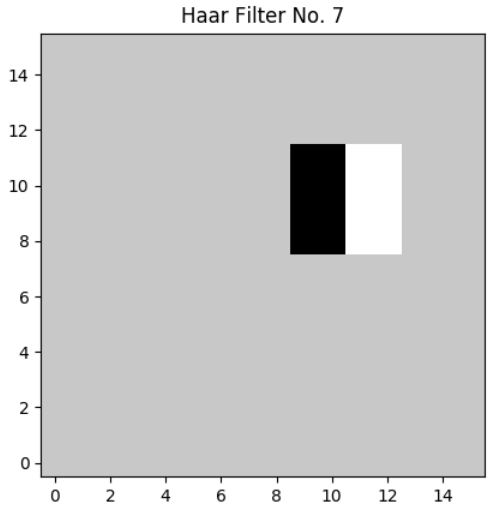

Haar Filter No. 2
0
2
4
6
8
10
12
14

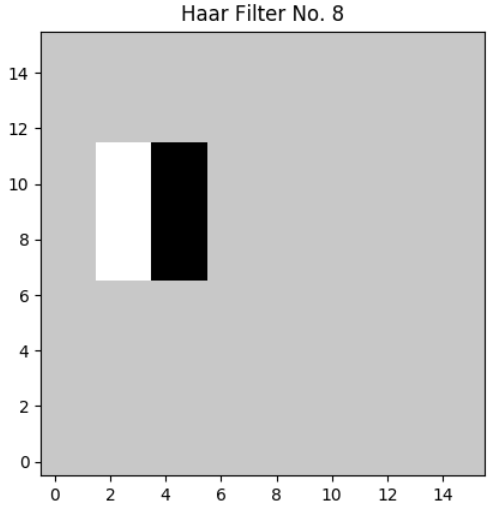

Haar Filter No. 3
0
2
4
6
8
10
12
14

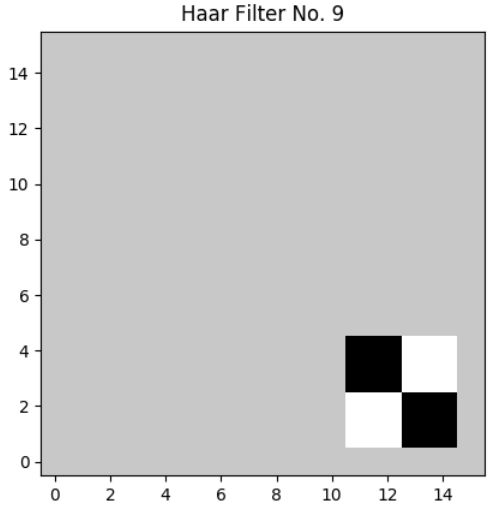

Haar Filter No. 4
0
2
4
6
8
10
12
14

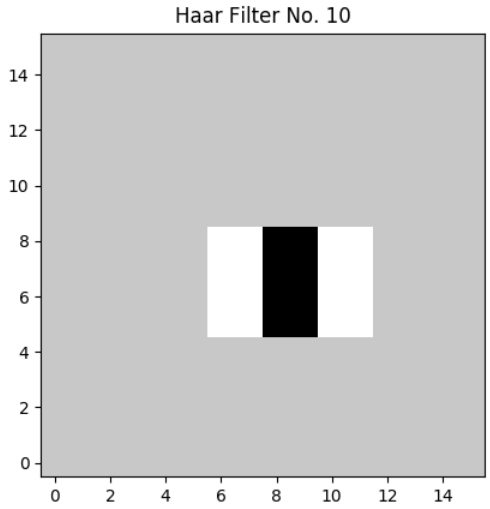

Haar Filter No. 5
0
2
4
6
8
10
12
14

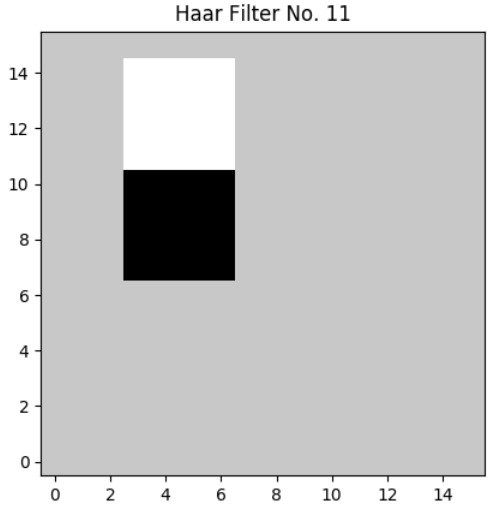

Haar Filter No. 6
0
2
4
6
8
10
12
14

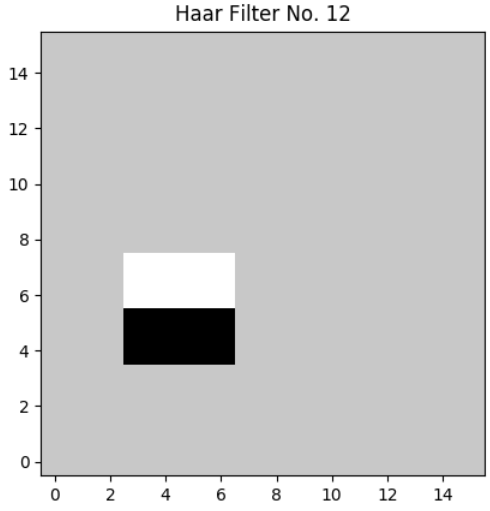

Haar Filter No. 7
0
2
4
6
8
10
12
14

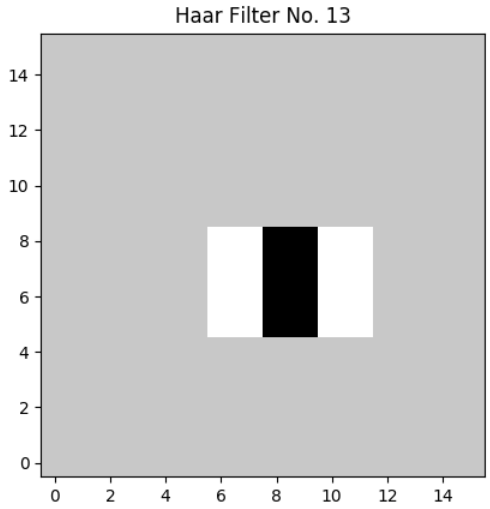

Haar Filter No. 8
0
2
4
6
8
10
12
14

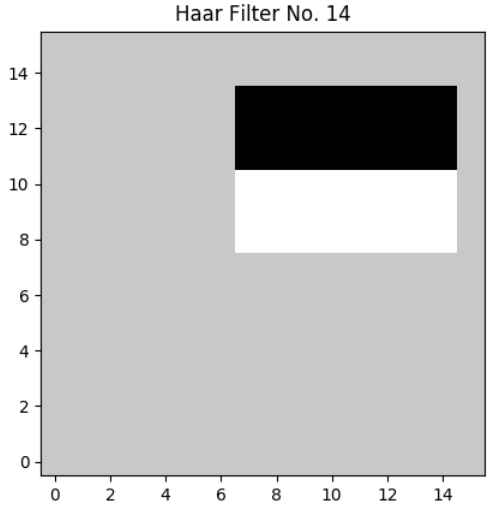

Haar Filter No. 9
0
2
4
6
8
10
12
14

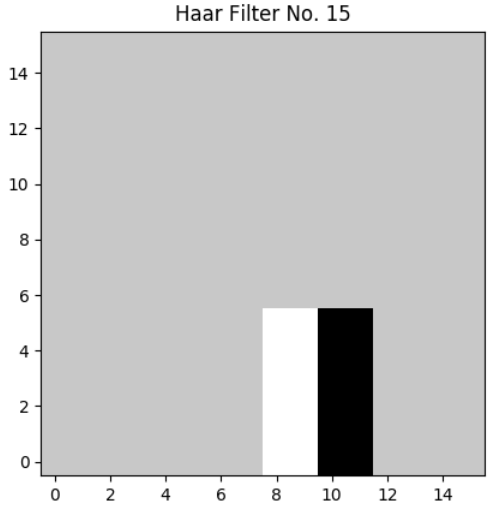

Haar Filter No. 10
0
2
4
6
8
10
12
14

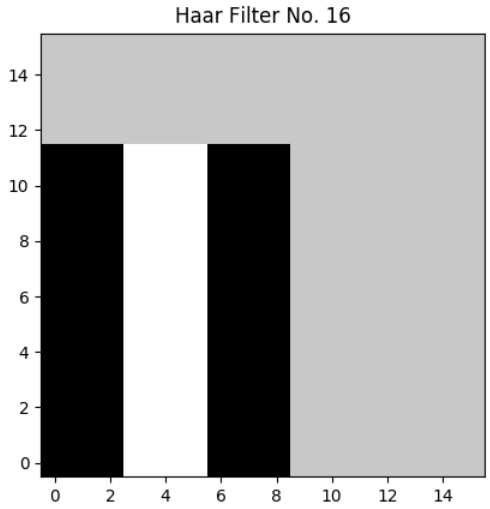

Haar Filter No. 11
0
2
4
6
8
10
12
14

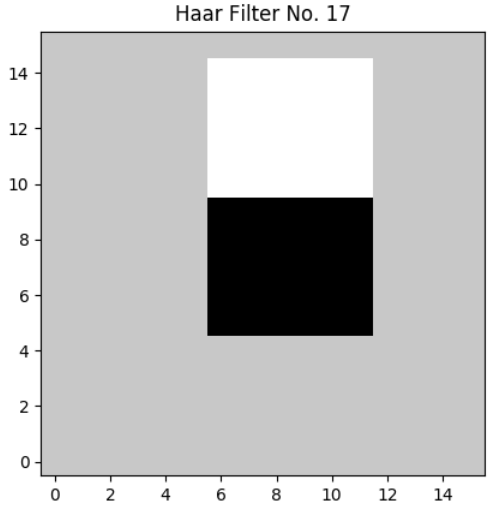

Haar Filter No. 12
0
2
4
6
8
10
12
14

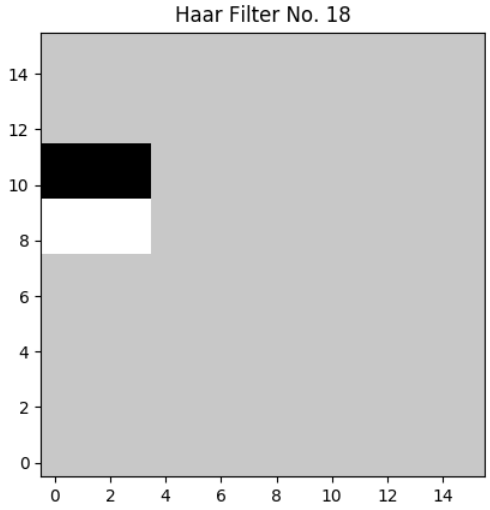

Haar Filter No. 13
0
2
4
6
8
10
12
14

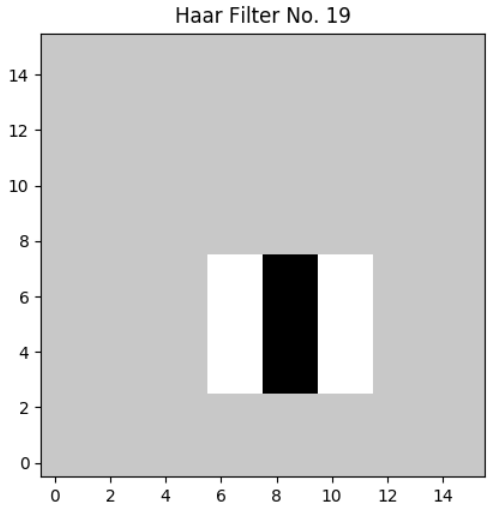

Haar Filter No. 14
0
2
4
6
8
10
12
14

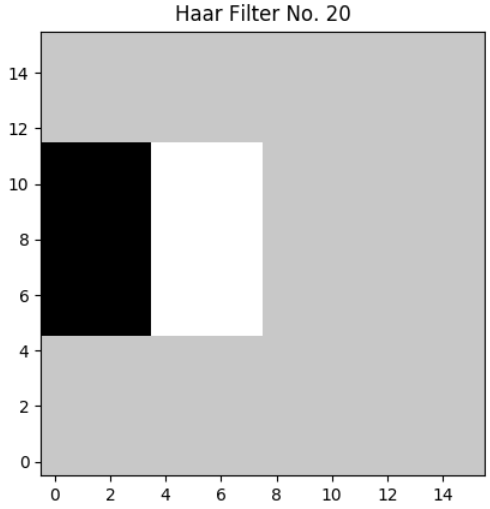

Haar Filter No. 15
0
2
4
6
8
10
12
14


Haar Filter No. 16
0
2
4
6
8
10
12
14

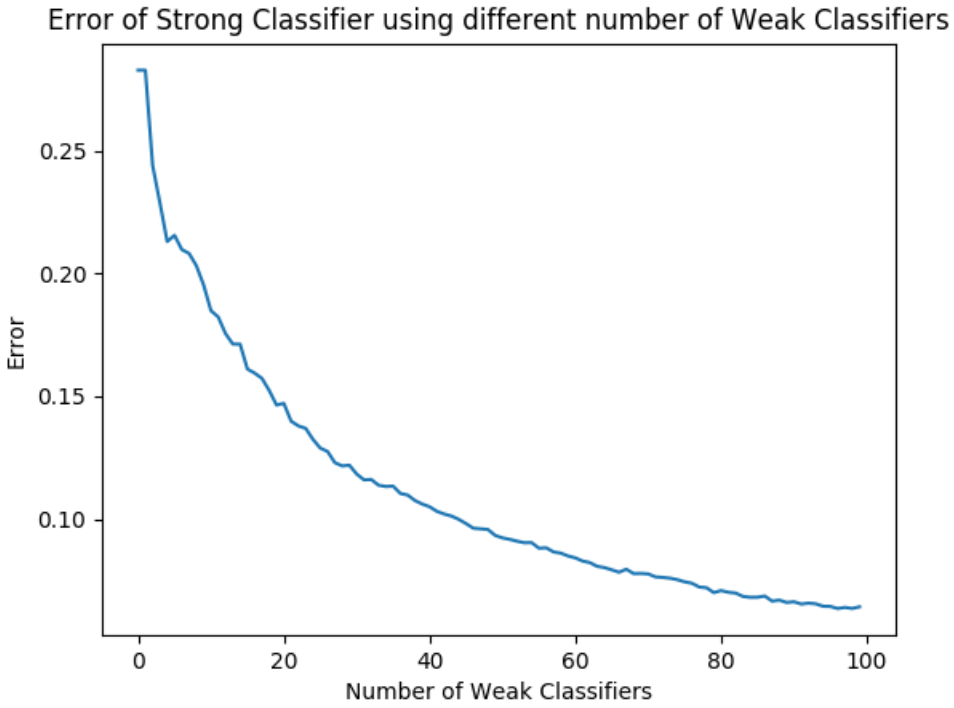

Haar Filter No. 17
0
2
4
6
8
10
12
14


Haar Filter No. 18
0
2
4
6
8
10
12
14

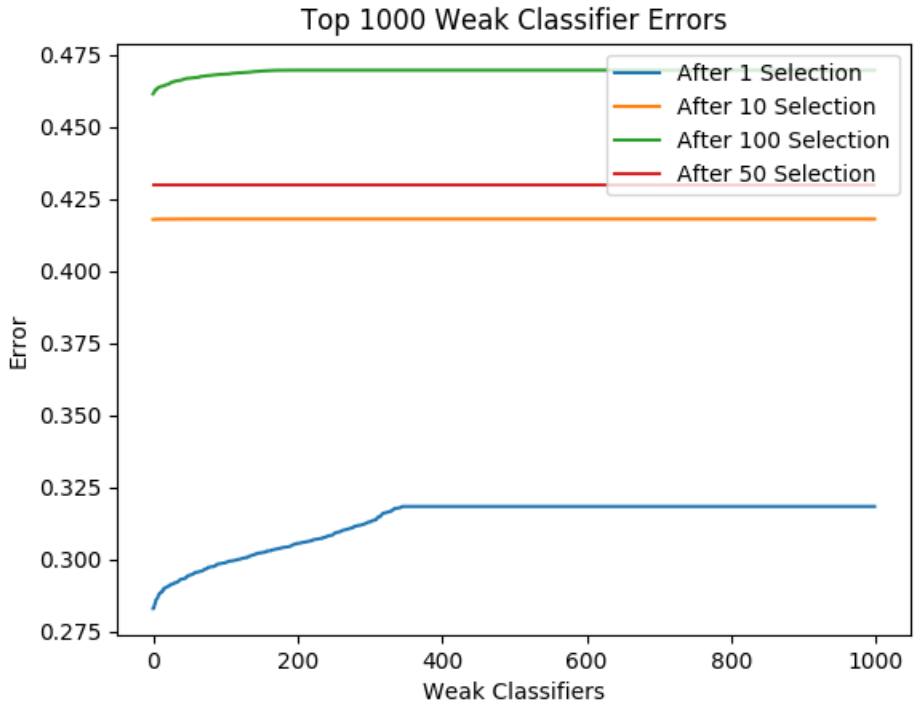




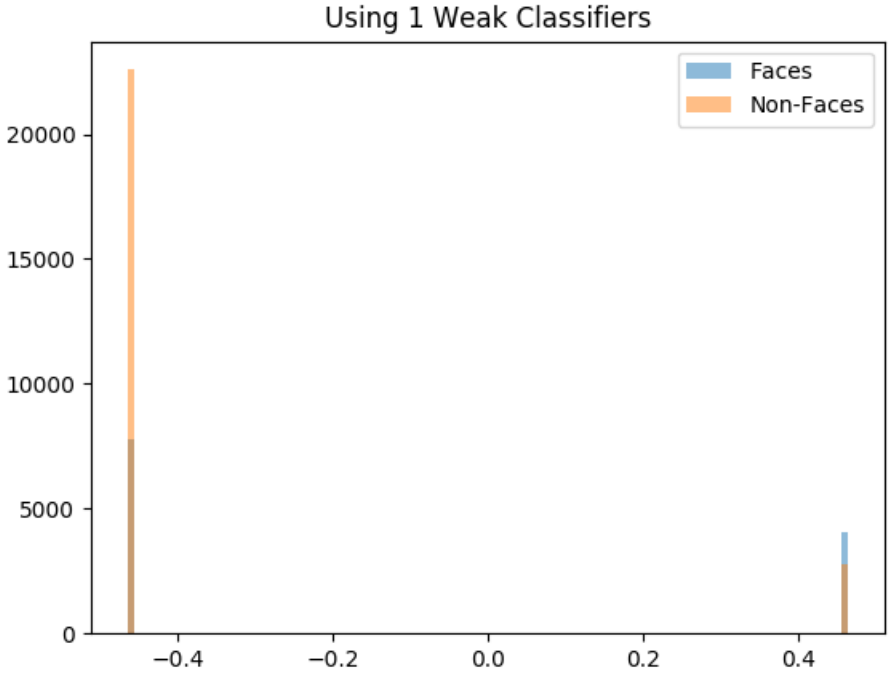
2. Below we have plotted the training error of the strong classifier over the number of weak classifiers as they keep getting added to the strong classifier. As expected, the overall training error of the strong classifier keeps decreasing as we add more and more weak classifiers to the strong classifier. The error slowly approaches towards 0 throughout the graph.

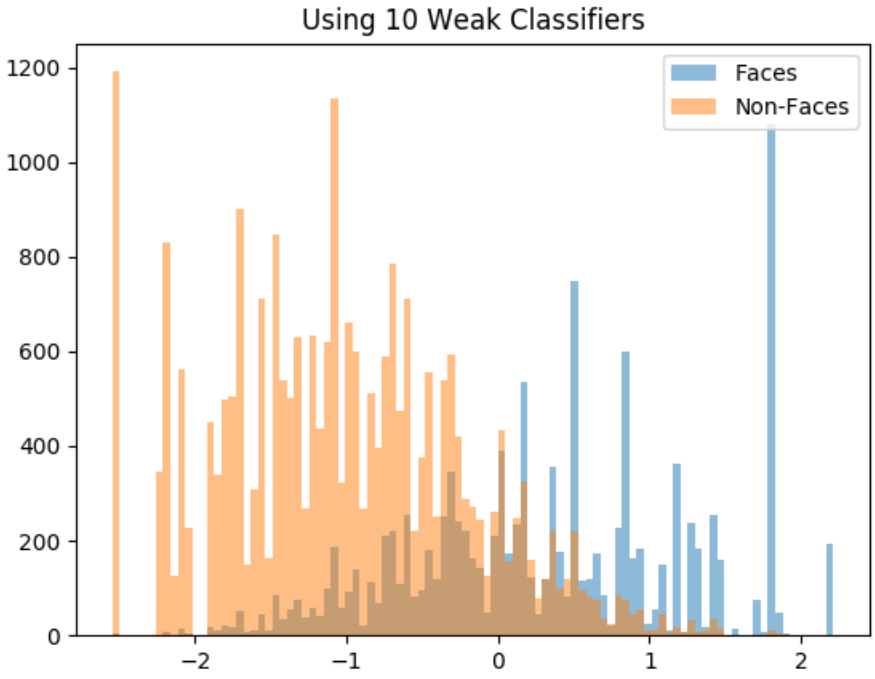


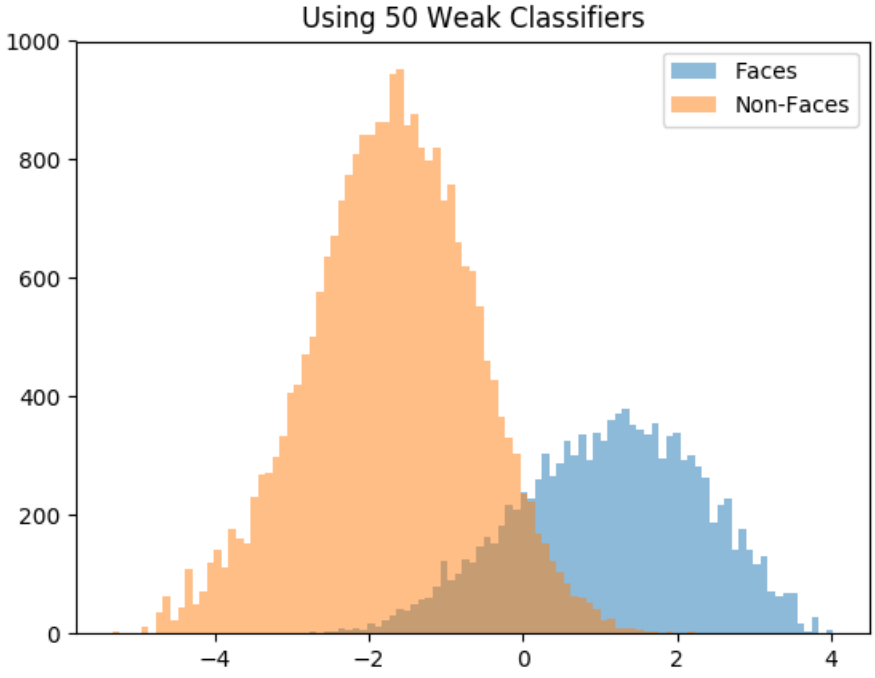
3. Below we have plotted the training error for the top 1000 weak classifiers from among the pool of all the weak classifiers in increasing order of the training errors after selecting 1 weak classifier, 10 weak classifiers, 50 weak classifiers and 100 weak classifiers. Comparing all the curves, we can see that the overall training error evidently increases as the number of weak classifier selections goes up. This happens because the best classifiers have already been selected and only a few data points are still misclassified (this has an effect on the data weights). Along each curve, the training error of the top 1000 weak classifiers increases before finally plateauing. This can be seen clearly when 1 weak classifier is selected and when 100 weak classifiers are selected. However, the training error after selecting 10 and 50 classifiers seems to remain constant. This happens because none of the weak classifiers can have a training error of more than 0.5 and this restricts the range of the overall training error since the best classifiers have already been selected. There is still some variation in the weak classifier error, however it is minute, and we will not be able to distinguish it since the curve has been plotted for a 1000 weak classifiers.

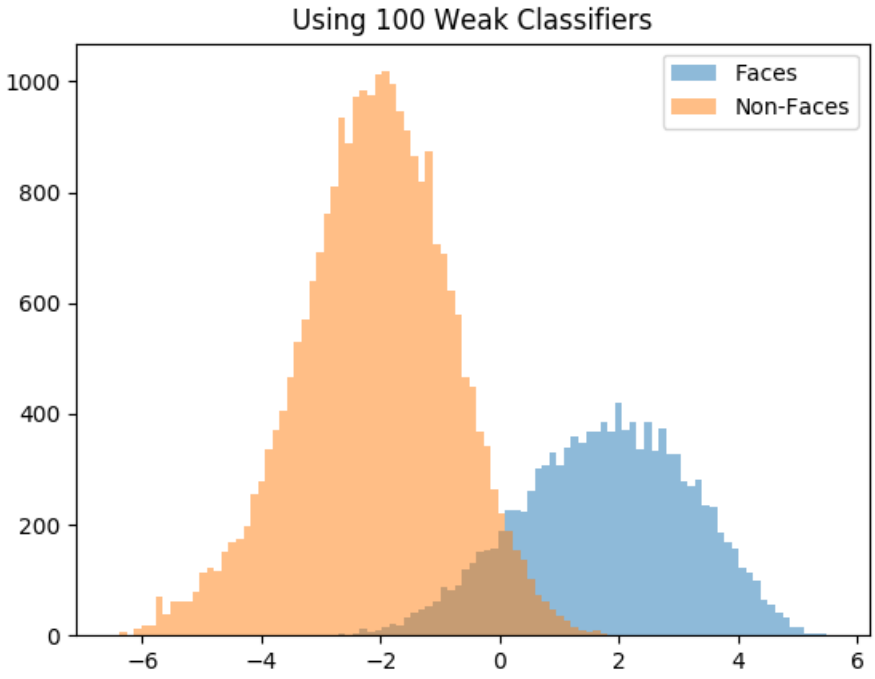


4. The histograms for the positive and negative population after selection of 1 weak classifier, 10 weak classifiers, 50 weak classifiers and 100 weak classifiers have been plotted below. As expected, there is significant overlap in the populations when only 1 weak classifier is used. The overlap decreases, and the populations become more ordered and easily distinguishable as the number of weak classifiers used increases.

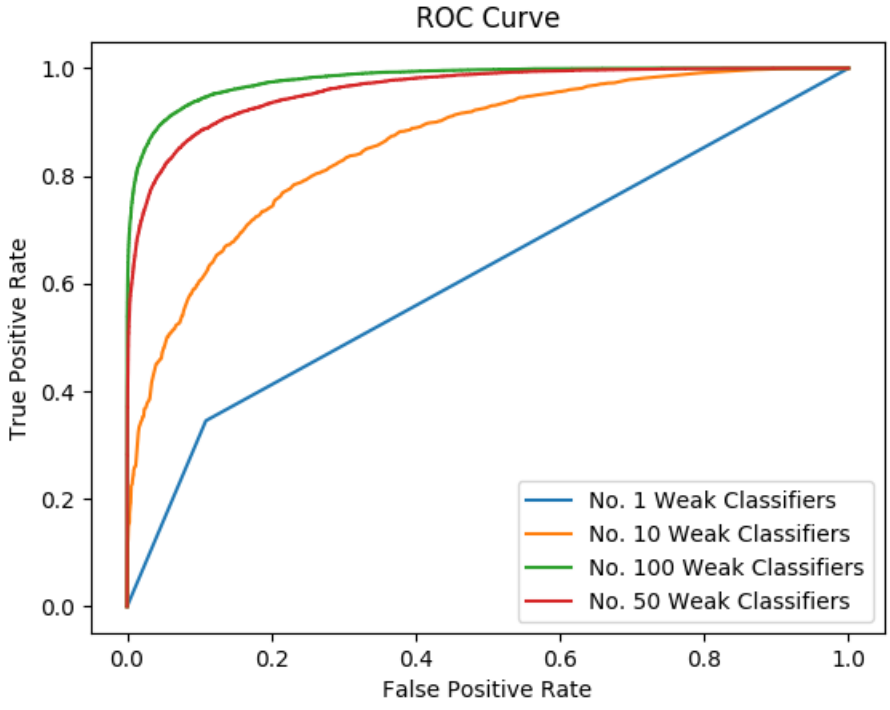



Using 10 Weak Classifiers
Faces
Non-Faces
1200
1000
800
600
400
200
0
−2
−1
0
1
2

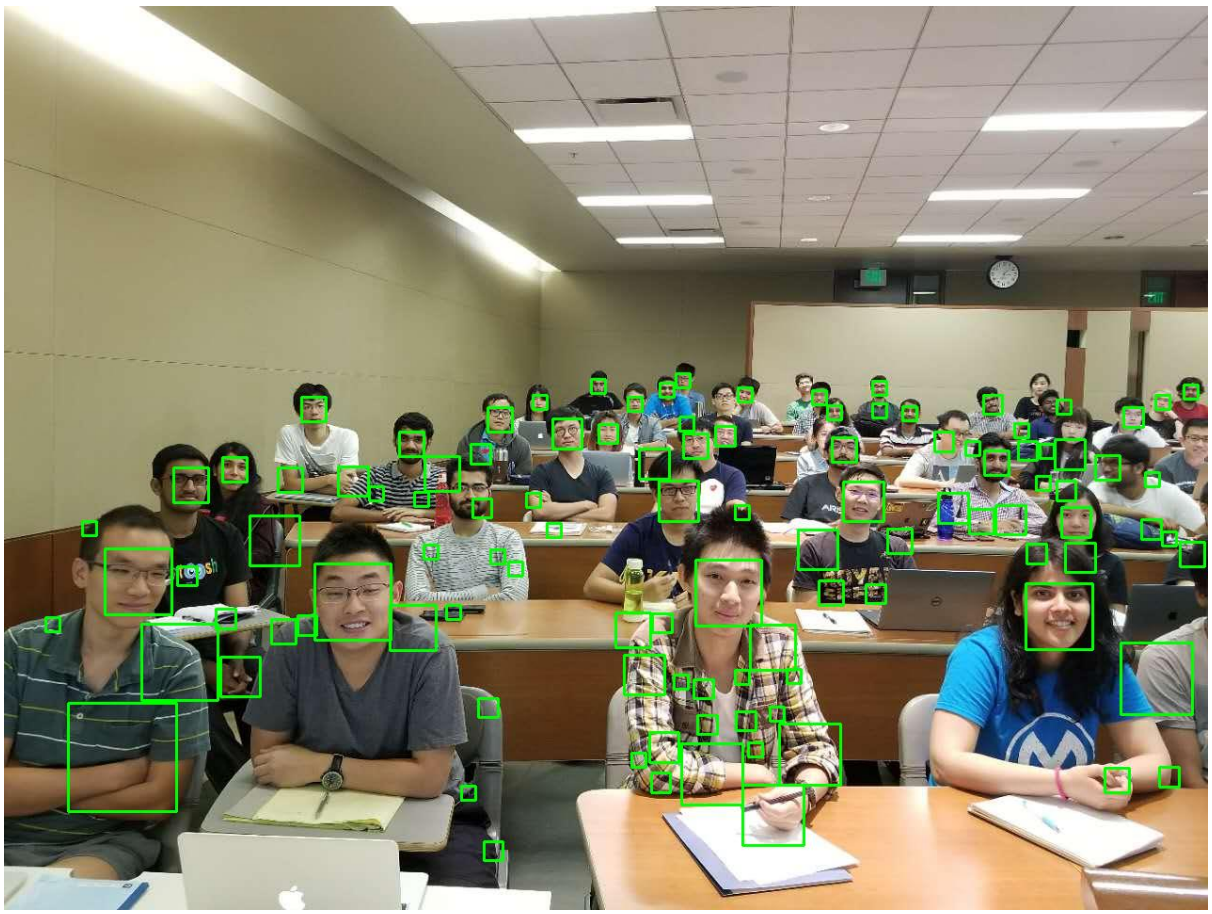

Using 50 Weak Classifiers
Faces
Non-Faces
1000
800
600
400
200
0
−4
−2
0
2
4



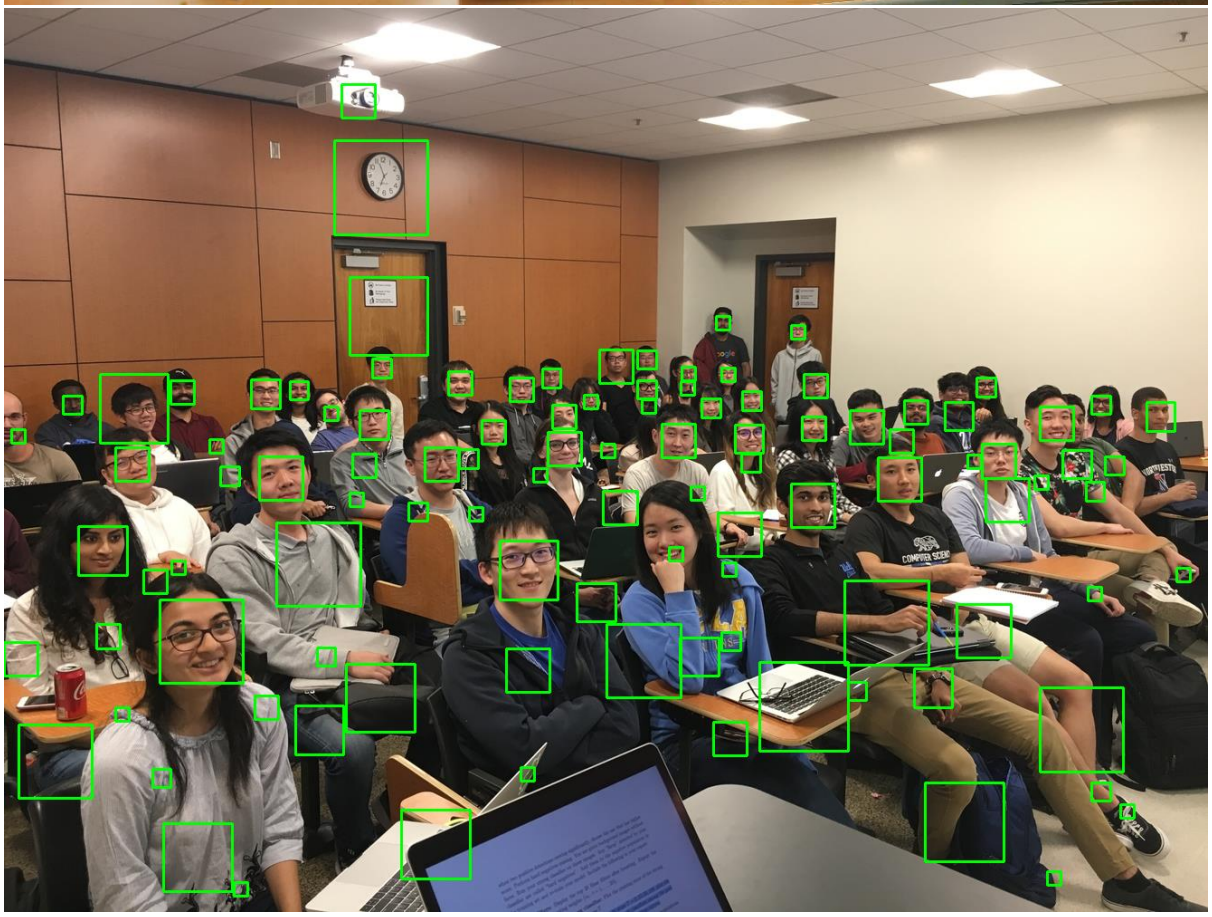
5. The ROC curves for the histograms have been plotted below. As can be observed from the graph, the elbow point of the curves increases as the number of weak classifiers used to form the strong classifier increases. This means that the true positive rate of the strong classifier keeps getting better and the false positive rate keeps decreasing as we use more and more weak classifiers to form the strong classifier (precision improves).



6. The following pictures depict the detected faces in the test images before hard negative mining. Even though most of the actual faces have been detected, there are a lot of false-positive detections in each image.

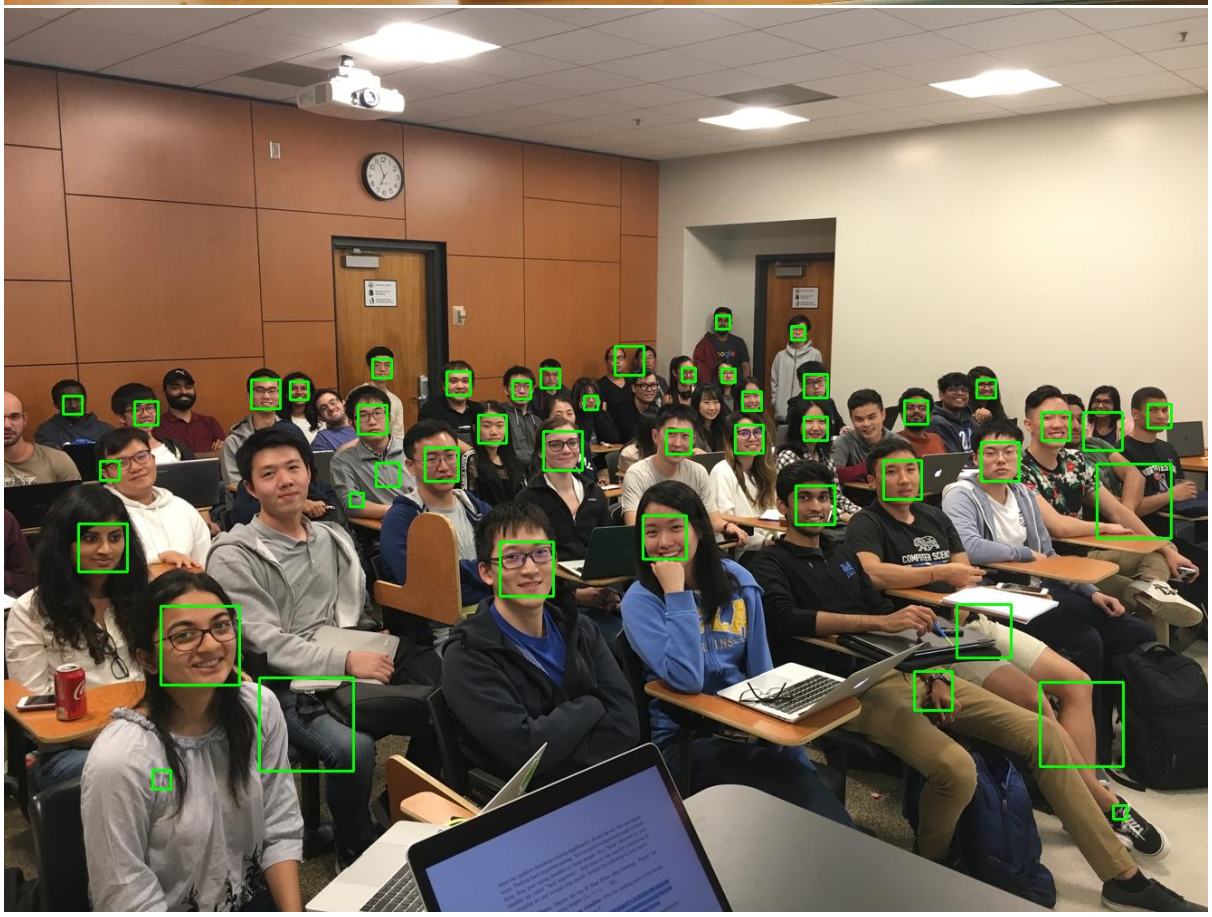



STATE



7. The following pictures depict the detected faces in the test images after hard negative mining. We observe that a few faces detected in the previous images are not properly detected in these images. However, the number of false-positive detections goes down substantially while most of the actual faces are still detected. Therefore, the overall precision improves.


STATE



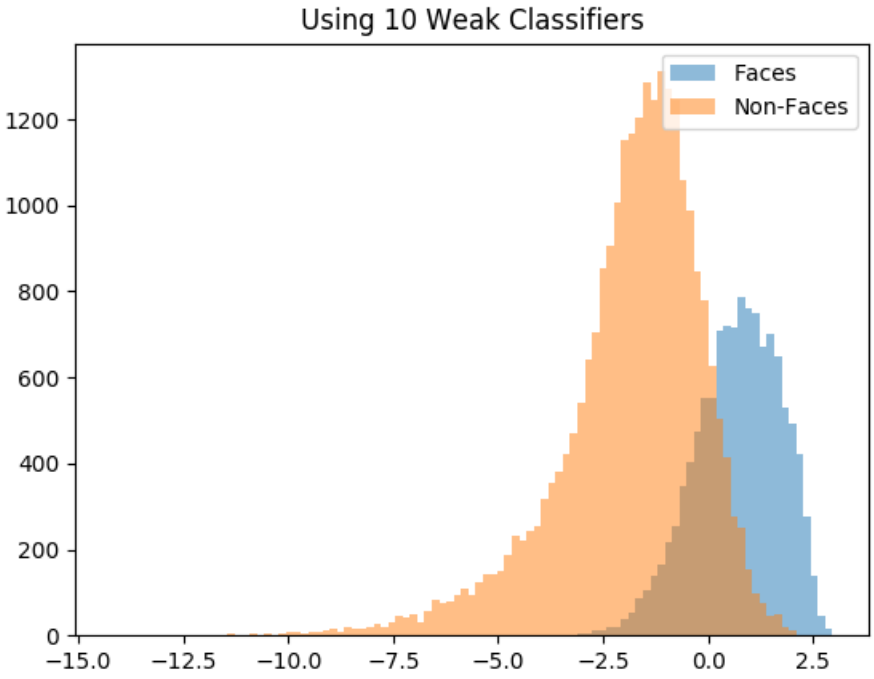
3. Implementing RealBoost

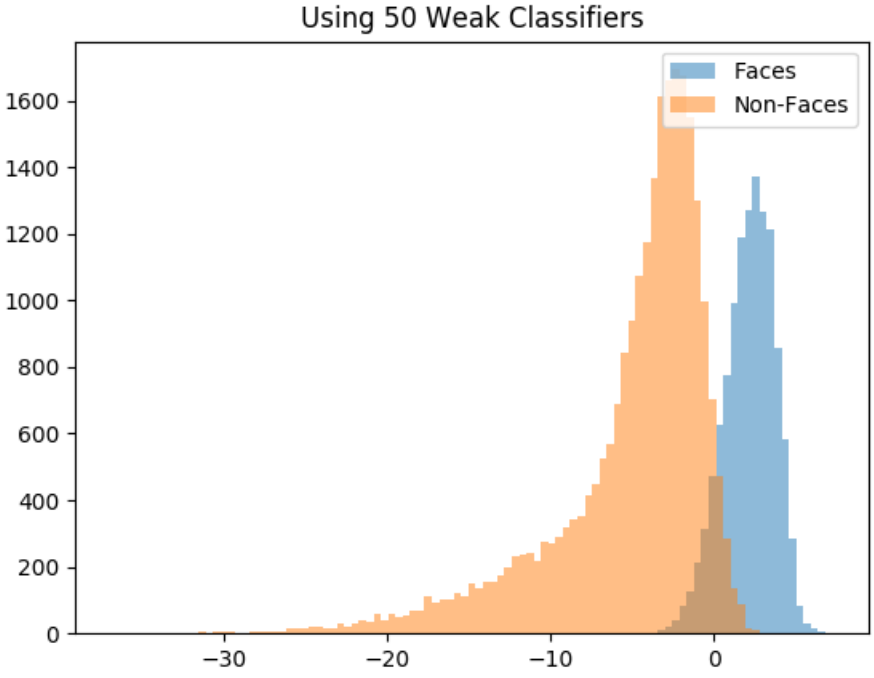
In RealBoost, we break down the range of activations into a number of bins. We assign each data point to a bin during the training period. Based on the number of faces and non-faces within each bin, we assign a value of p and q to each bin in the weak classifier. When we have to classify a data point using this weak classifier, we base our prediction on the p and q value of the bin to which the activation belongs.

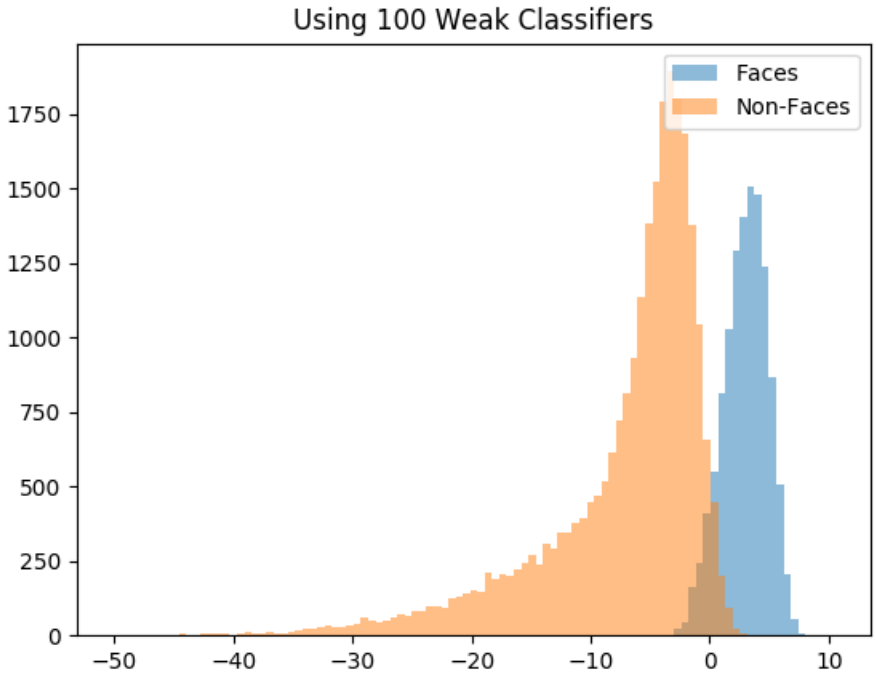
1. The histograms for the positive and negative population after selection of 1 weak classifier, 10 weak classifiers, 50 weak classifiers and 100 weak classifiers have been plotted below. When only 1 weak classifier is selected, there is significant overlap in the positive and negative populations and they are practically indistinguishable. As the number of chosen classifiers increases, the overlap between the populations decreases and they become more easily distinguishable.

   However, in the case of RealBoost, the overlap begins plateau off after 10 weak classifiers have been selected. This is because RealBoost achieves the majority of its overall accuracy using lesser number of classifiers and is unable to improve upon the accuracy after that. In the case of AdaBoost, the overall accuracy is very low when using just 10 weak classifiers and there is significant overlap in the population. The improvement in accuracy and decrease of overlap starts to plateau after 50 weak classifiers have been chosen. Also, the overall accuracy of RealBoost seems to be much better than the overall accuracy of AdaBoost, when comparing the graph formed after selecting 100 classifiers.

   Therefore, we can infer that RealBoost has better performance as compared to AdaBoost.

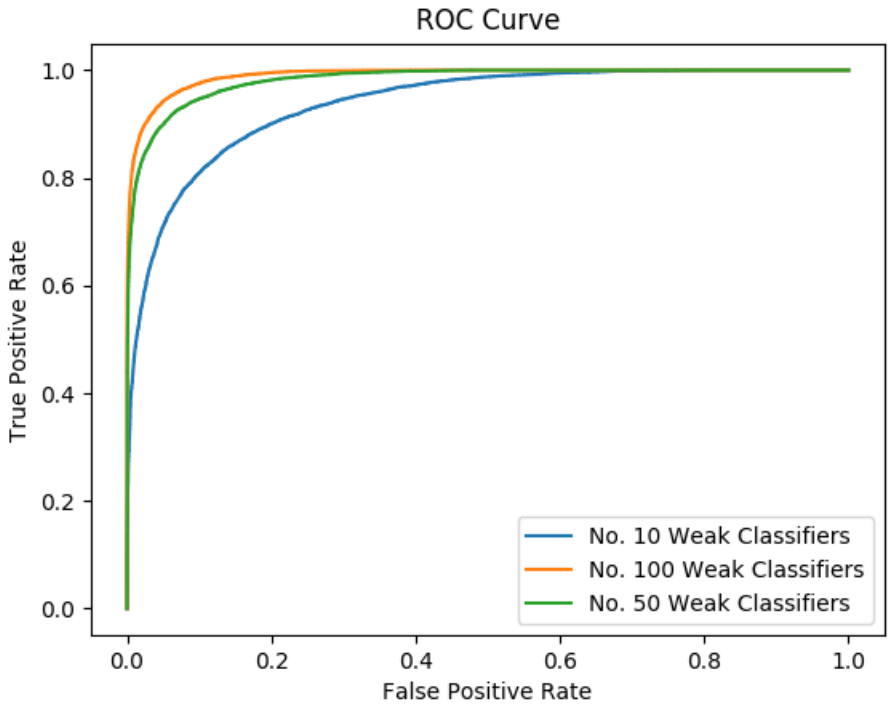

Using 10 Weak Classifiers
Faces
Non-Faces
1200
1000
800
600
400
200
0
−15.0
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5


Using 50 Weak Classifiers
Faces
Non-Faces
1600
1400
1200
1000
800
600
400
200
0
−30
−20
−10
0



2. The ROC curves for the histograms of RealBoost have been plotted below. It is evident from the graph, that the elbow point of the curves increases as the number of weak classifiers used to form the strong classifier increases.

   Compared to the ROC graph formed by AdaBoost, we can see that the ROC graph for RealBoost is much steeper, thereby indicating that RealBoost gives more number of true positives as compared to false positives. Therefore, the precision of the strong classifier produced by RealBoost is much better than the precision of the strong classifier produced by AdaBoost.


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
No. 10 Weak Classifiers
No. 100 Weak Classifiers
No. 50 Weak Classifiers