CS 276A: Project 2 Report

Human Face Detection using Boosting

Boosting is a general method for improving the accuracy of any given learning algorithm. Specifically, one can use it to combine weak learners, each performing only slightly better than random guess, to form an arbitrarily good hypothesis. In this project, we are required to implement AdaBoost and RealBoost algorithms for frontal human face detection.

We are given 2 sets of data with faces and non-faces respectively.

The data is loaded with the corresponding labels as +1 for faces and -1 for non-faces. A total of 37194 images of size 16*16 are loaded with 11,838 faces and 25,356 non-faces in the bitmap format.

Two algorithms are implements: Adaboost and Realboost.

1) Adaboost

The images are sent to an integrated images function where the integral image is calculated based on the pixel values at each point of the image. The data is normalised. A set of Haar Filters which form our weak classifiers pool, are loaded. Each Haar Filter is applied to the integral images thereby producing responses called activations. These activations are stored in a pickle file. Then the model is trained to select the best performing Haar filters. The weights are initialised to a uniform distribution (1/m).

The training is done for 110 weak classifiers. The weighted error for each weak classifier is calculated and the classifier with minimum error is selected.

The error is calculated by finding a suitable threshold. The minimum and maximum values of the activations is computed and a sample of 50 threshold is selected using the linspace() function. The error for each threshold is calculated using the formula,

$$\epsilon_t(h) = \sum_{i=1}^{m} D_t(x_i) 1(h(x_i) \neq y_i) \quad \forall h \in \Delta$$

If the error is greater than 0.5 (random guess) then the polarity is switched, i.e. error is saved as (1-error) and polarity is made -1. The error are calculated for all the sample thresholds, and the one with least error is selected and returned. The object variables threshold, polarity and error are set to the least weighted error value respectively.

Based on the error for each weak classifier, the alpha value is calculated using the formula,

$$\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t(h_t)}{\epsilon_t(h_t)}$$

and the weights are updated iteratively, using the formula,

$$D_{t+1}(x_i) = \frac{1}{Z_t} D_t(x_i)\exp\{-\alpha_t S(h_t(x_i) = y_i)\}$$

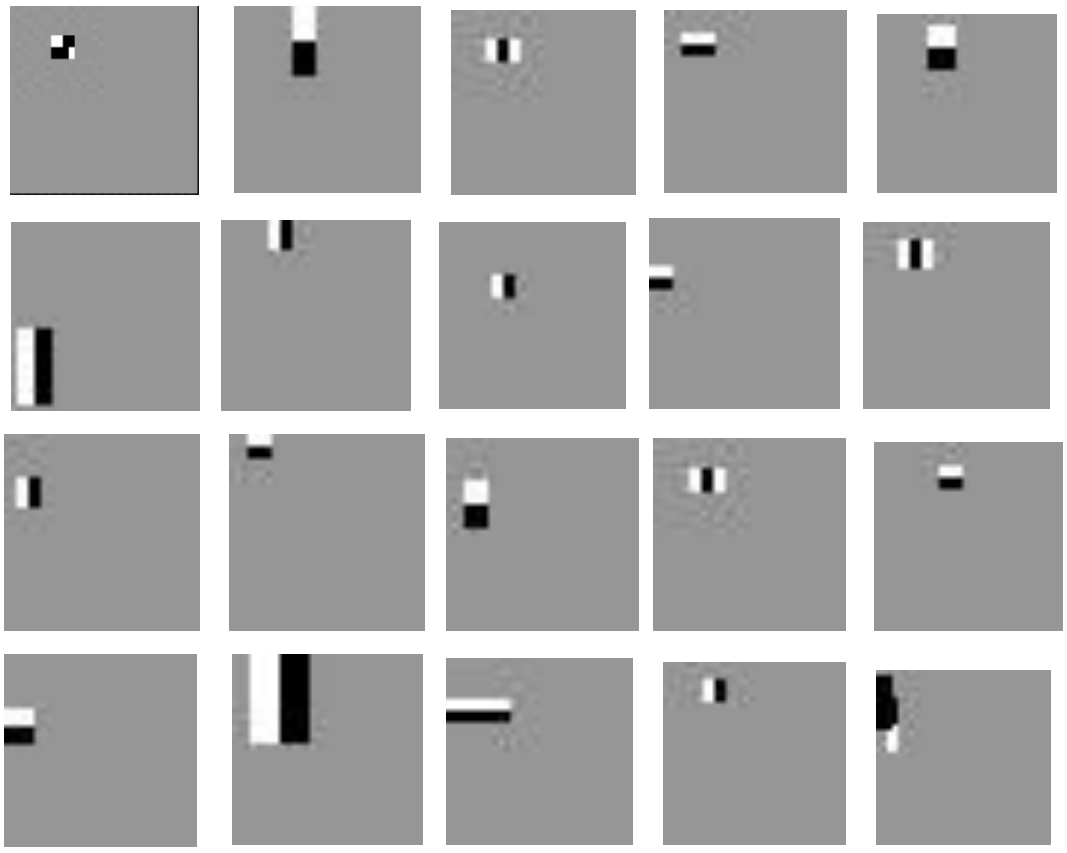
The value of the errors are in the range 0.1 - 0.3 with corresponding alpha values ranging from 0.08-0.41.

After that the top 110 weak classifiers along with their alpha values are appended into the chosen_wcs variable.

The training accuracy is reported as 0.9410 (94.10%)

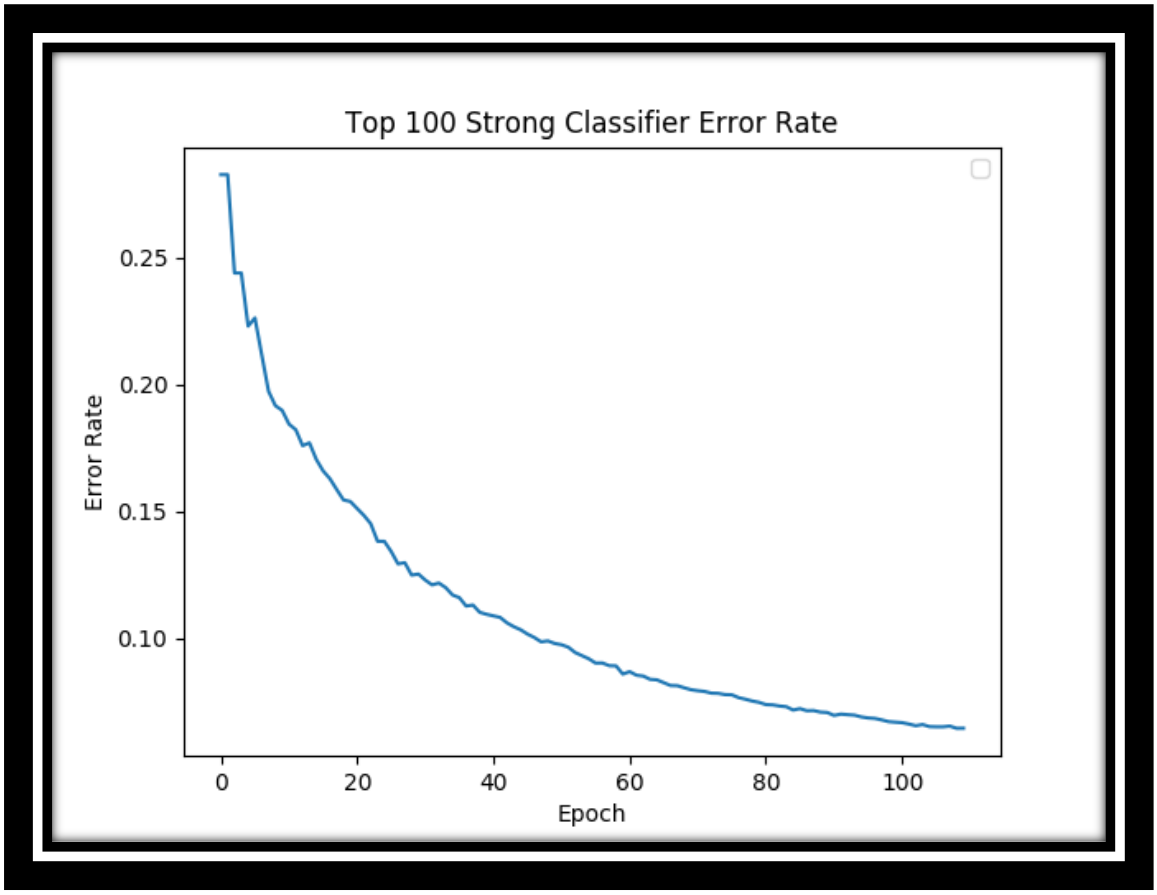
At the end of training, the top 20 best performing weak classifiers are plotted as shown below:

a) Top 20 Haar Filters:

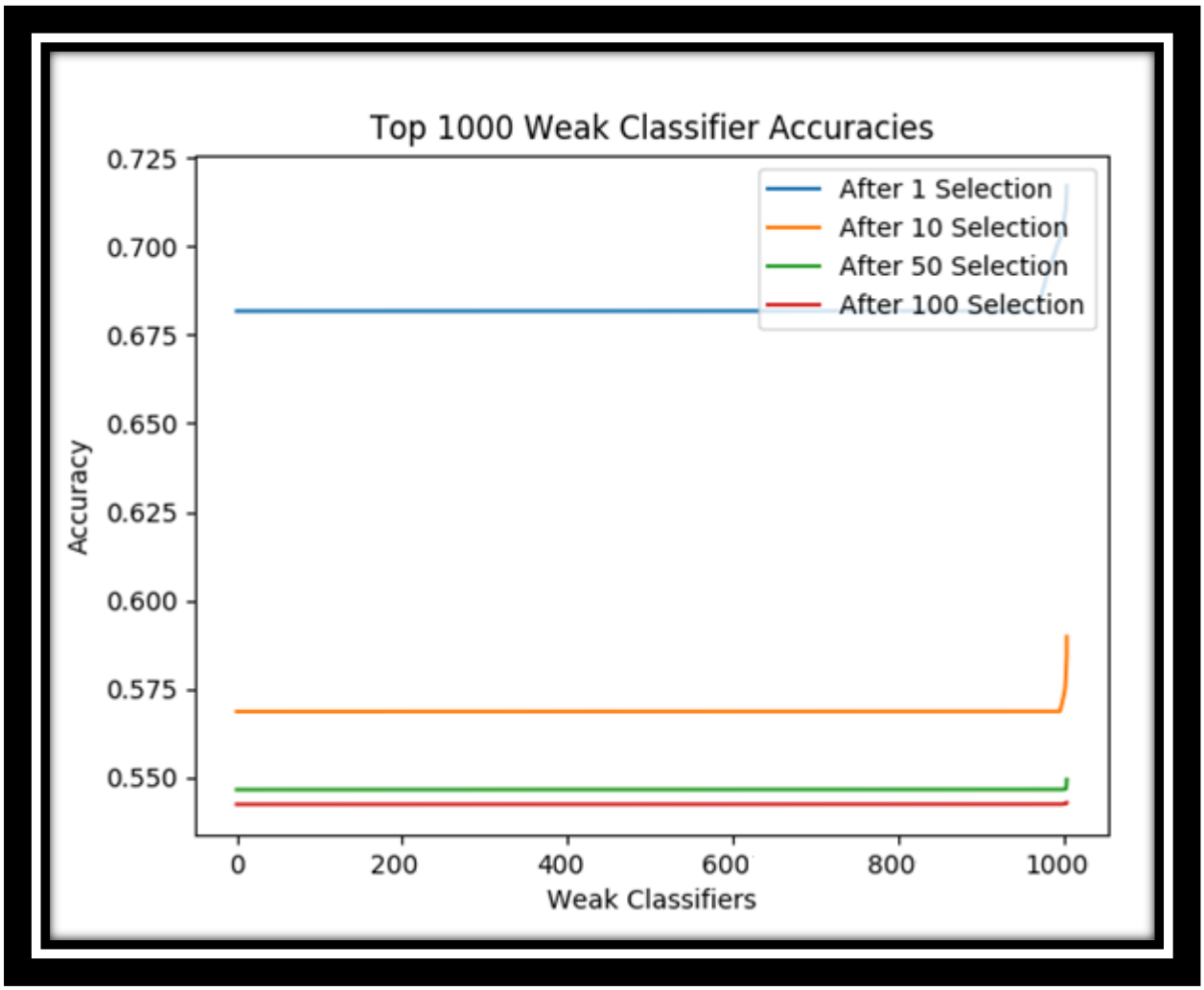


b) The strong classifier error rate is computed by comparing the sign of the score, with the sign of the labels and calculating the mean of that and subtracting it from 1. (1-accuracy) over 110 classifiers.

The resultant graph shows a downward slope as the error keeps decreasing for stronger classifiers.

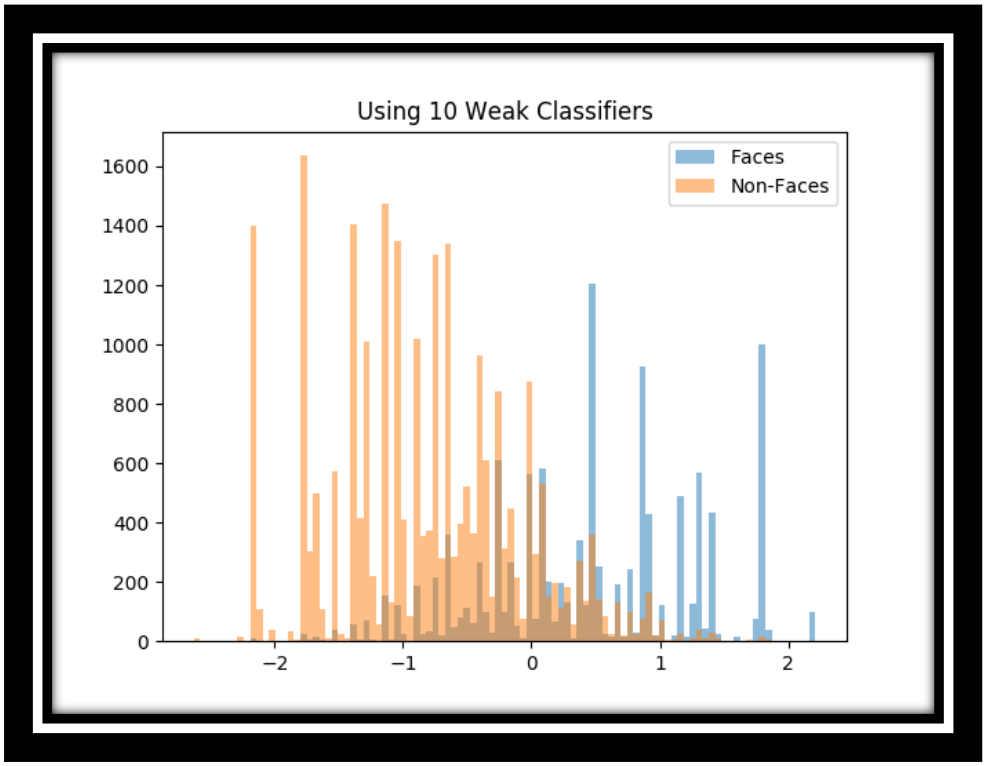


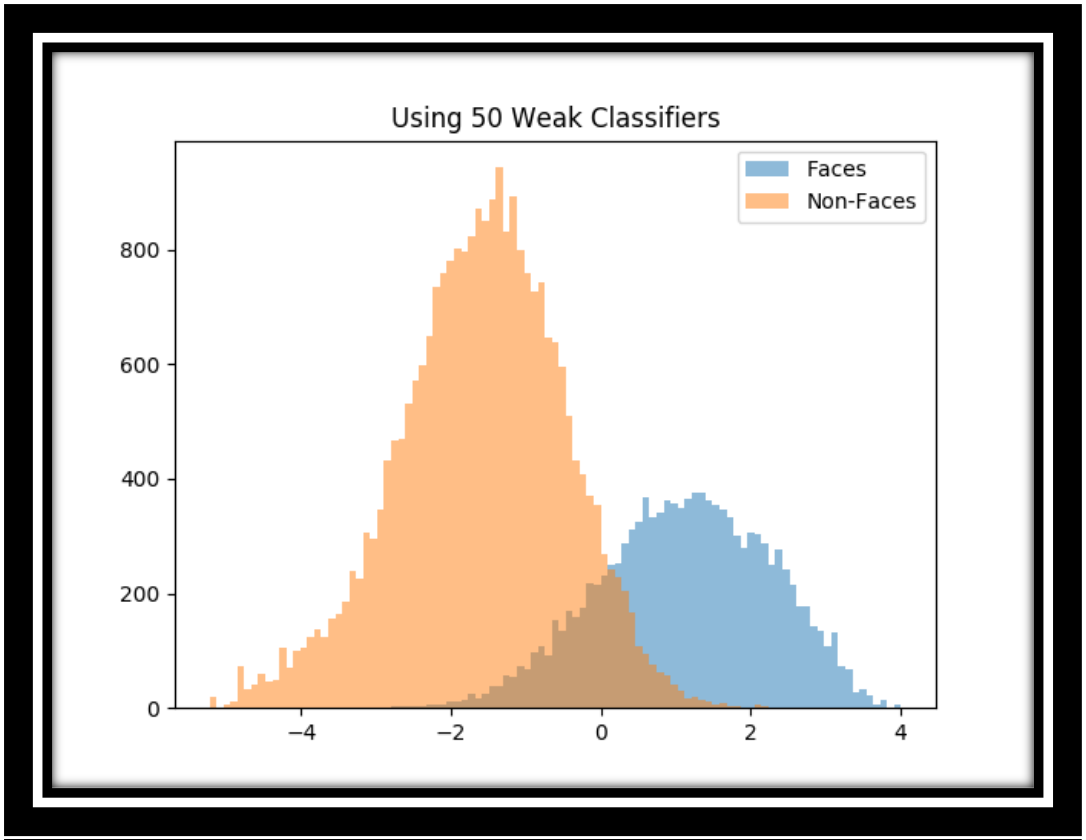
c) The weak classifier accuracies are computed by (1-error) as calculated by the error function. These are also plotted for 1 , 10, 50 and 100 selections respectively.

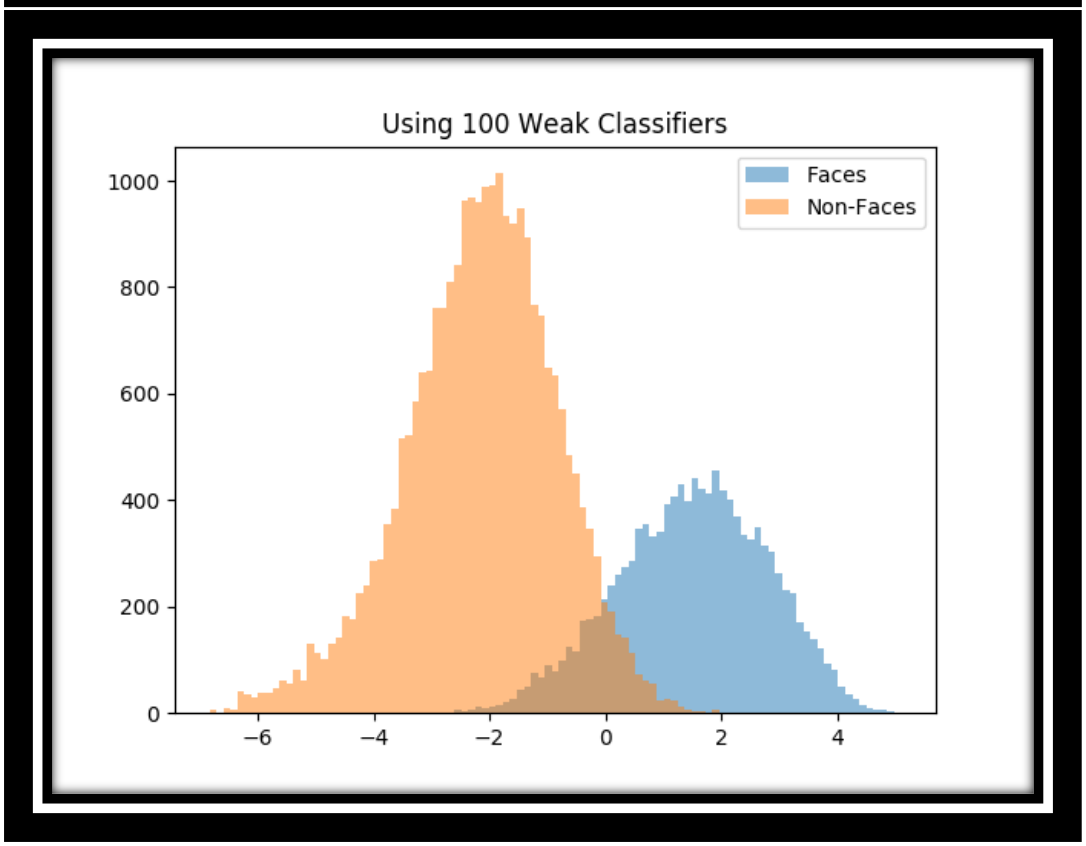


d) For each weak classifier the score is calculated which is appended to the strong_classifier_score variable. The histograms using the score (predicted faces) is plotted at different levels for 10, 50 and 100 weak classifiers respectively and are shown below. As the number of classifiers increase, the distinction from face and noon-face also increases.

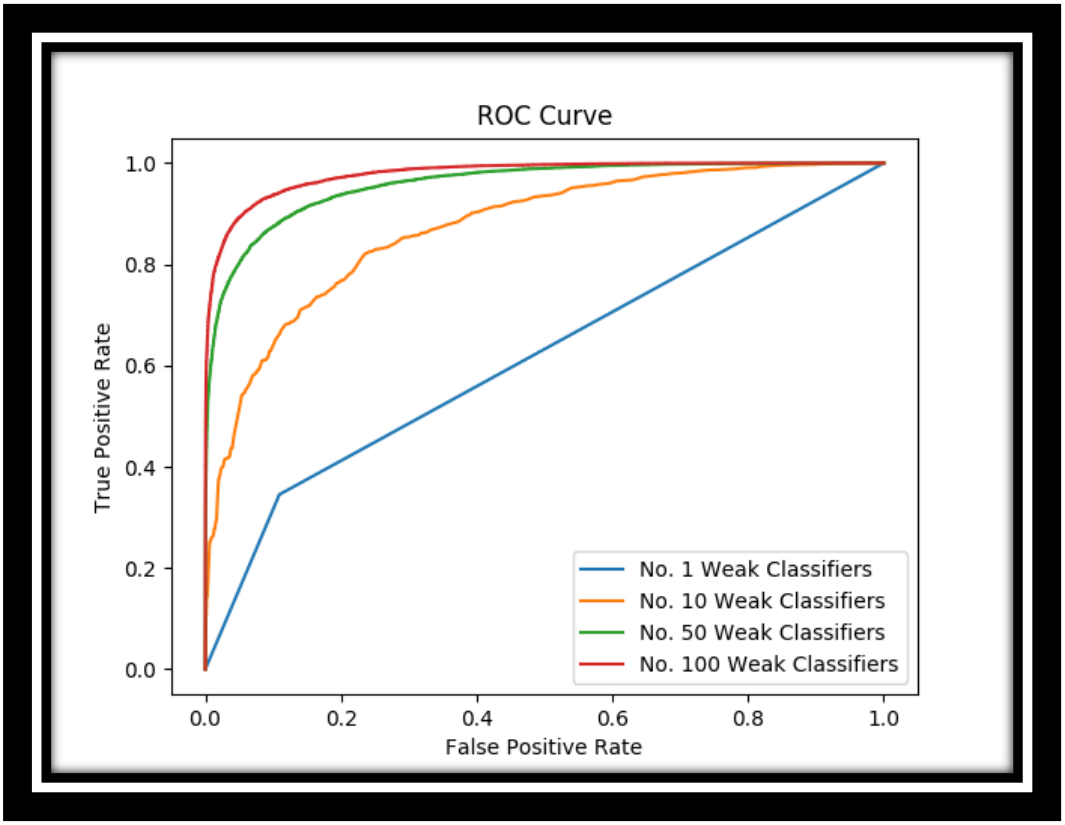
Histograms:




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


Using 100 Weak Classifiers
Faces
Non-Faces
1000
800
600
400
200
0
−6
−4
−2
0
2
4

e) Based on the scores, the ROC curve plotting the True positives (faces that were detected as faces) against the False positives (Non-faces that were detected as faces) is shown below. We see that the ROC curve for 100 weak classifiers is close to one, which is the optimum solution.



f) After training is completed, we are given 3 pictures for face detection.

The images are individually loaded. These images are resized considerable into smaller and smaller window sizes and finally broken down into patches of size 16*16 using the image2patches function. After detecting faces in the image using the trained data, we apply non-maximum suppression to eliminate over-lapping patches. We extract the x and y coordinates of each patch (opposite corners) and sort it by the score. Then we pick the patch with the highest score and compare it to all the other patches and keep eliminating the ones with lower scores. Finally we return only the patches with higher scores as their tendency to be a face is much higher.

The images after face detection look like as follows:


STATE

g) After performing face detection, we perform hard mining of non-faces. We extract all the patches that are detected as faces and append them to the data as non-faces with corresponding labels of -1. There are 2209, 2469 and 2565 patches extracted from each image which are added to the data as non-faces since they were detected incorrectly as faces.

   After loading the new data with the incorporated hard mining data, we retrain our model by doing all the steps once again, to make it more powerful.

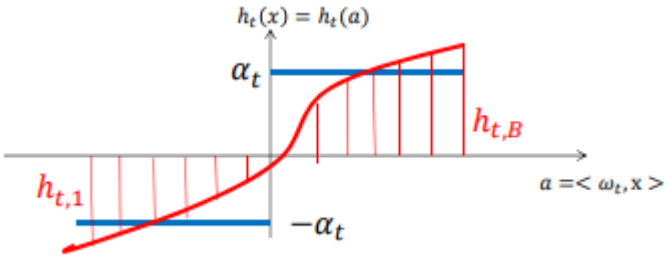
   We see that the number of patches after hardmining are reduced considerably, resulting in better face-detection.

After retraining, the result of the face detection on the same three pictures are as follows:


STATE

2) Realboost

We can extend Adaboost to a more general form by transforming h_t(theta) being a vector of B parameters, where B is the number of bins. We approximate a 1D function by a vector.



This can be represented as the following:

$$F(x;\Theta) = h_{1,b_1(x)} + \ \ldots + h_{t,b_t(x)}$$

where each weak classifier votes by an amount h_t,b which is a real number and is more effective than the binary one, in case of Adaboost. The h_t function absorbs the alpha during error calculation.

For Realboost implementation, the images are loaded as before and the integral image is computed using the nomalised data. The trained activations from Adaboost are used to perform Realboost. Thus instead of chosing from a vast pool, we only train from the top 100 selected classifiers from Adaboost.
After loading the cached trained data we call the error function by creating the realboost object and referencing it with the id variable. The weights are initialised to a uniform distribution of 1/m.
In the error function we calculate the p and q values for each bin using the following formula:

$$p_t(b) = \sum_{i=1}^{m} D(i)\ 1(y_i = +1)1(i \in Bin(b))$$

$$q_t(b) = \sum_{i=1}^{m} D(i)\ 1(y_i = -1)1(i \in Bin(b))$$

where p represent the positive samples and q represents negative samples respectively in each bin. The threshold is assigned to the value of the edges of each bin. The bin_pqs variable saves the p and q values for each bin, and the train_assignment variable stores which bin is each image a part of.
After computing the p and q values we check if any of the values are 0, ie, if the bin was empty and didn't have any positive/negative values. In such a case, we assign a very small value (1e-7) to avoid getting an error during eroor calculation(NAN error).
Using p and q, the error is calculated using the following formula:

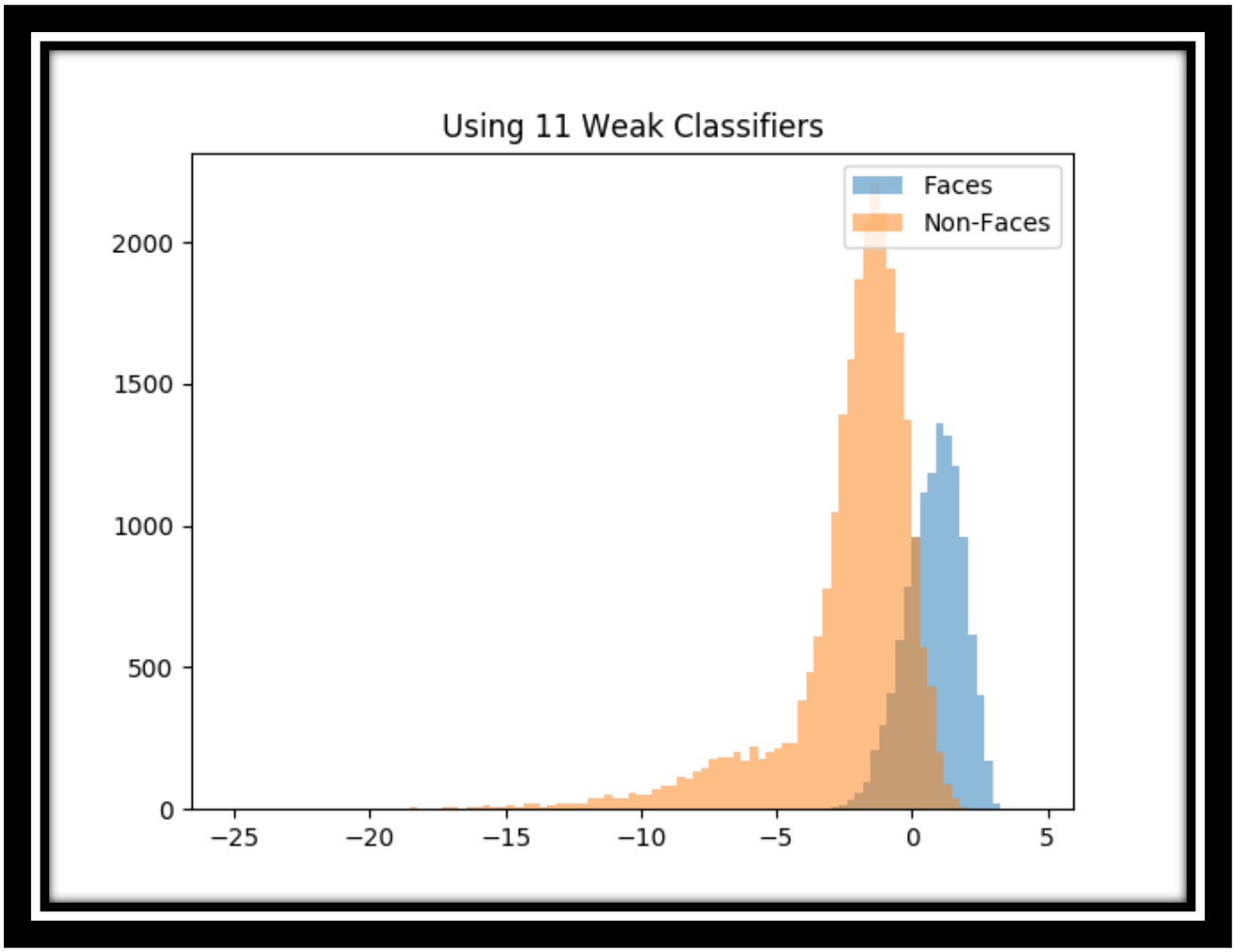
$$Z = 2\sum_{b=1}^{B} \sqrt{p_t(b)q_t(b)}$$

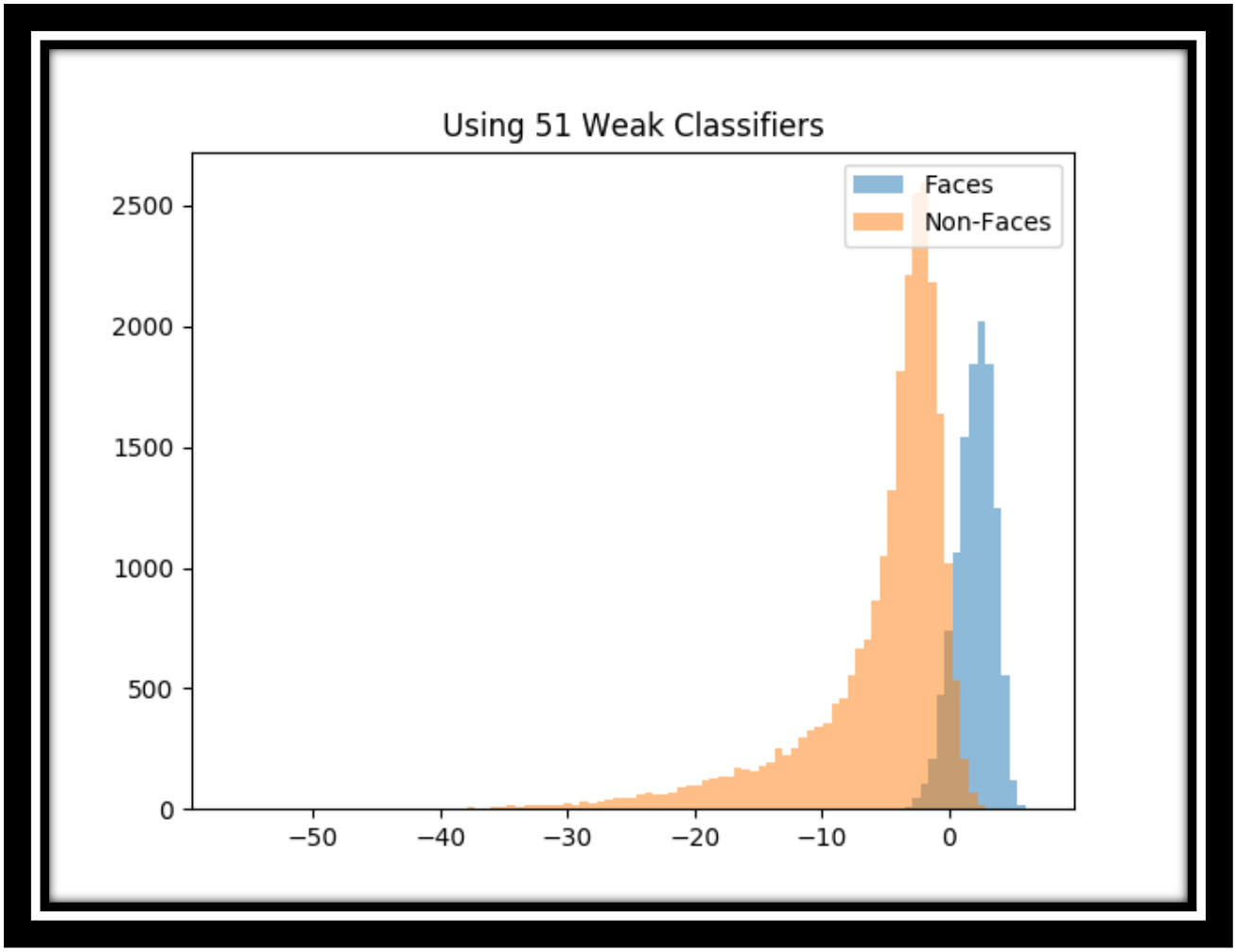
and returned to the training function.
The weights are updated iteratively for each classifier using the formula:

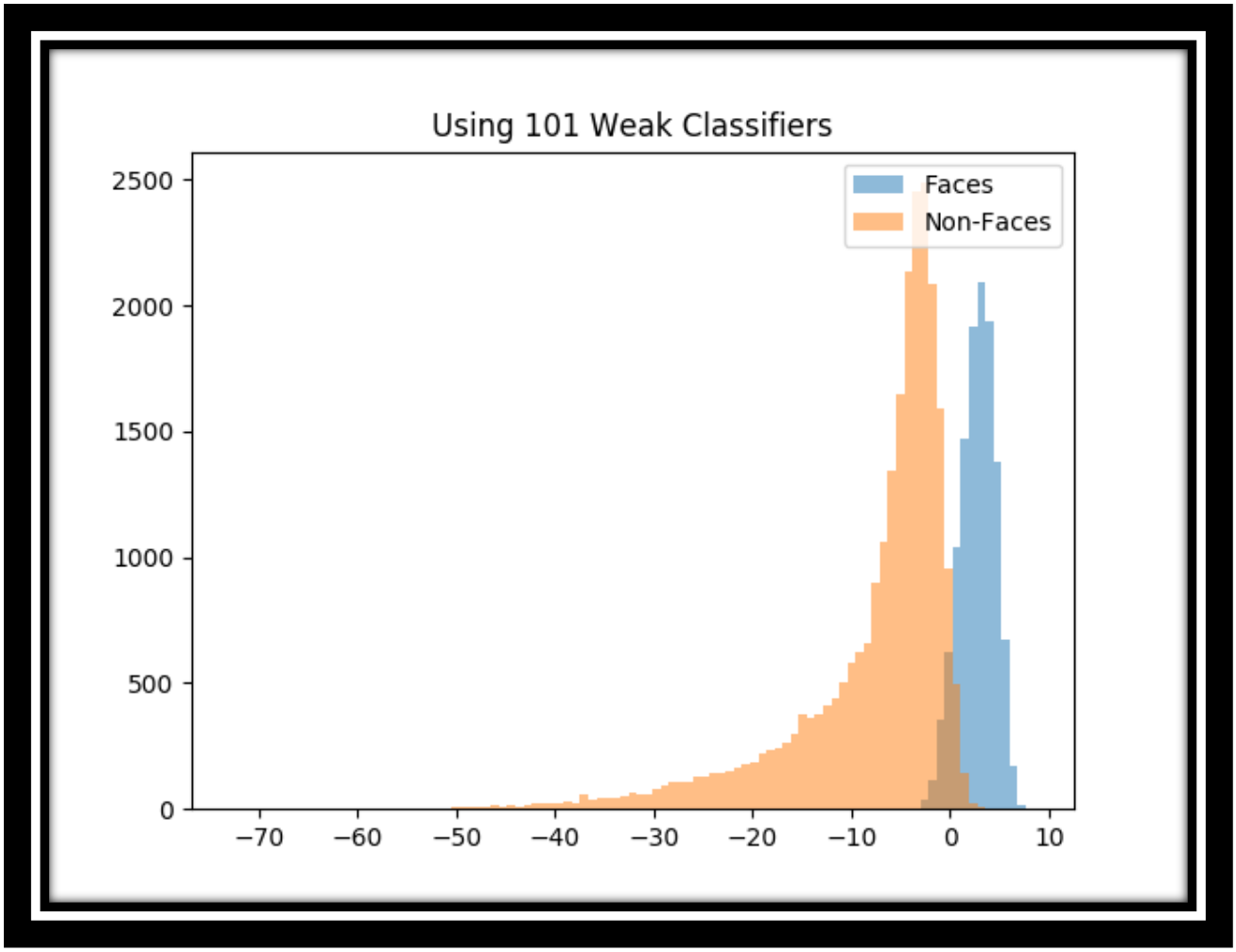
$$D_t(x_i) = \frac{1}{Z_t} D_{t-1}(x_i) e^{-y_i h_{t,b(x_i)}}$$

and the classifiers are appended into the chosen_wcs variable. The score for each classifier is computed and appended into the strong_classifier_score variable to compute the histogrm and ROC curves.

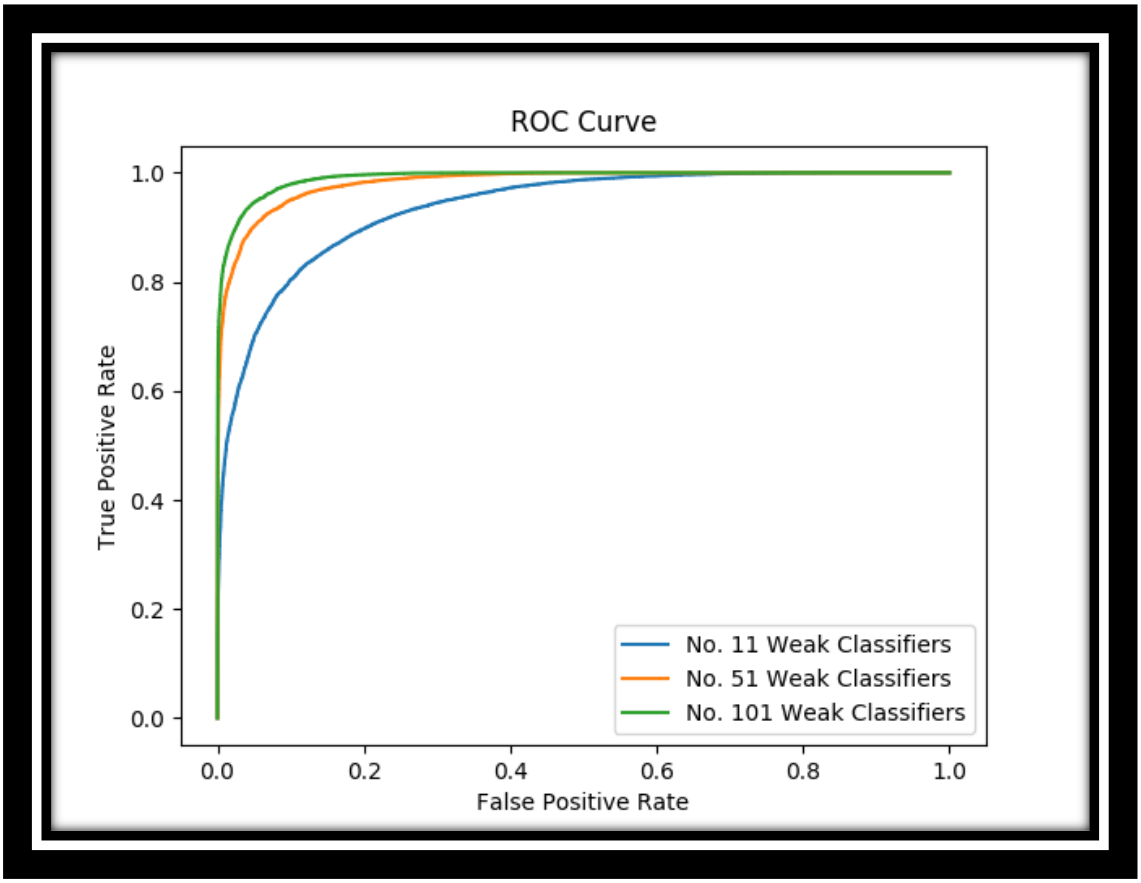
h) The histograms of negative and positive populations are plotted as follows for T = 10,50,100, respectively.




Using 51 Weak Classifiers
Faces
Non-Faces
2500
2000
1500
1000
500
0
−50
−40
−30
−20
−10
0


Using 101 Weak Classifiers
Faces
Non-Faces
2500
2000
1500
1000
500
0
−70
−60
−50
−40
−30
−20
−10
0
10

i) ROC: Based on the histograms the corresponding ROC curves for the top 100 classifiers are plotted.



We see that the ROC curve for RealBoost gives better results than that of AdaBoost (as shown in (e)). The curves are closer to 1 which is the most optimum solution of the True positive against False Positive graph.

Thus we implement both AdaBoost and RealBoost algorithms for frontal human face detection. Boosting is a general method for improving the accuracy of any given learning algorithm. Specifically, one can use it to combine weak learners, each performing only slightly better than random guess, to form an arbitrarily good hypothesis. In this project, we see how weak classifiers together form strong classifiers that give us high accuracy and enable face detection.