Kaustubh Shirpurkar
427006745
ECEN 649

Project report

Face detection is one of computer vision literature's most studied subjects. The objective of face detection, given an arbitrary image, is to decide whether or not the image has any faces and, if present, return the position and extent of the image of each face. While this seems to be a trivial task for human beings, it is a very challenging task for computers. The facial recognition problem can be due to many differences in size, location, point of view, lighting, occlusions, etc. Although there have been hundreds of reported face detection approaches, if you were asked to name a single face detection algorithm that has the most impact in recent decades This project will study and understand the Viola-Jones algorithm by implementing the entire detection framework and based on implementation, conduct experiments to hopefully further improve performance.

Part 1: Extract Haar Features

The first step of the Viola-Jones face detection algorithm is to turn the input image into an integral image. Integral image, also known as a summed area table, is an algorithm for quickly and efficiently computing the sum of values in a rectangle subset of a pixel grid. The integral image at location x, y contains the sum of the pixels above and to the left of x, y, inclusive:

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

where $i(x,y)$ is the pixel value of the original image and $ii(x,y)$ is the corresponding image integral value. Using the integral image to compute the sum of any rectangular area is extremely efficient, as shown in Figure 1. The sum of pixels in rectangle ABCD can be calculated with only four values from integral image:

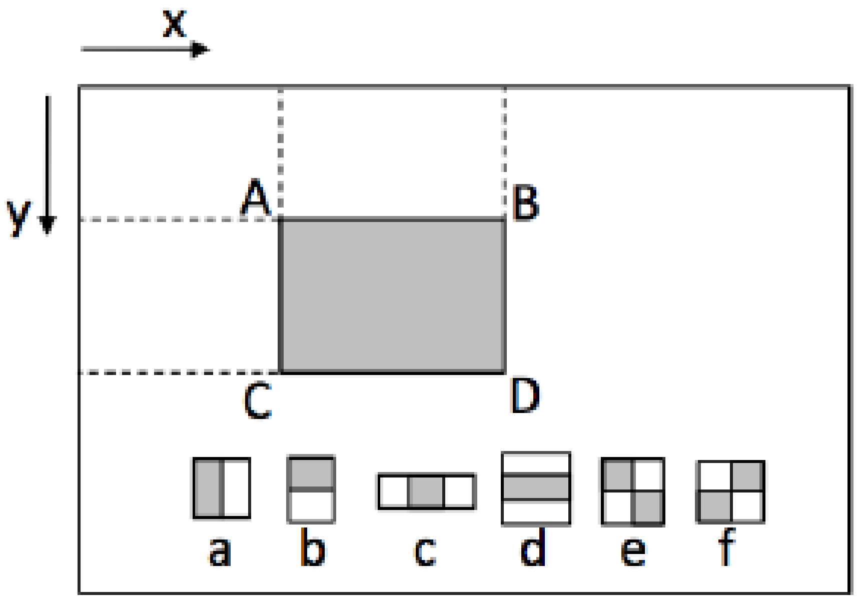

Figure 1: Illustration of the integral image and 6 types of Haar-like rectangle features

$$\sum_{(x,y)\in ABCD} i(x,y) = ii(D) + ii(A) - ii(B) - ii(C)$$

The base resolution of a sub-window in our implementation is 19 by 19 pixels.

In our implementation, for each image sample, we extract five types of Haar-like features:

(a) Two vertical = type (1)

(b) Two horizontal = type (2)

(c) Three vertical = type (3)

(d) Three horizontal = type (4)

(f) Diagonal = type (5)

Given a image input which is 19 by 19 pixels, we compute all possible Haar-like features for all the 5 filters

- The total number of Haar Features is: **63960**
- There are **17100** type (a) or type (1) (two vertical) features.
- There are **17100** type (b) or type (2) (two horizontal) features.
- There are **10830** type (d) or type (3) (three horizontal) features.
- There are **10830** type (c) or type (4) (three vertical) features.
- There are **8100** type (f) or type (5) (diagonal) features.

Among the project files **number_of_feature_vectors.m** includes the code for calculating number of feature vectors of a particular type.

Matlab function **calcHaarVal.m** is used to extract all the feature vectors. It takes integral image of an input image, filter type, starting point of image, length and width of the haar filter and output is intensity differences between white/black region of Haar features. It uses **getCorners.m** as a helper function. This function takes in an integral image and computes the sum of intensities in the area bounded by the four coordinates.

Part 2 : Build Your Adaboost Detector

The Viola- Jones uses a variant of AdaBoost to both select a small set of features and train the classifier. A single AdaBoost classifier consists of a weighted sum of many weak classifiers, where each weak classifier is a threshold on a single Haarlike rectangular feature. The weight associated with a given sample is adjusted based on whether or not the weak classifier correctly classifies the sample.A single weak classifier is defined as:

$$h(x, p, f, \theta) = \begin{cases} 1 & pf(x) < p\theta \\ 0 & \text{otherwise} \end{cases}$$

where f denotes the feature value, θ is the threshold and p is the polarity indicating the direction of the inequality.

AdaBoost learning procedure is as follows:

1. Given training sample images $(x_1, y_1), \dots, (x_n, y_n)$, where $y_i = \{0, 1\}$ for negative and positive examples respectively.
2. Initialize the classifier count t = 0 and the sample weights $w_i = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = \{0, 1\}$ here m and l are the number of negative and positive samples.
3. While the number of negative samples rejected is less 50%:
   (a) Increment t = t + 1.
   (b) Normalize the weights $w_i = \frac{w_i}{\sum_i w_i}$
   (c) Select the best weak classifier with respect to the weighted error

   $$\epsilon_t = \min_{f,p,\theta} \sum_i w_i |h(x_i, p, f, \theta) - y_i|$$

   (d) Define $h_t(x) = h(x, p_t, f_t, \theta_t)$ where f_t, p_t, θ_t are the minimizers of ϵ_t.
   (e) Update the weights as

   $$w_i = w_i \beta_t^{1-\epsilon_t}$$

   where $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$ and $e_i = 0$ if classification is correct and $e_i = 1$ otherwise.
   (f) Compute the strong classifier

   $$H(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \gamma_t \\ 0 & \text{otherwise} \end{cases}$$

   where $\alpha_t = \log\left(\frac{1}{\beta_t}\right)$ and γ_t is chosen such that all positive training samples are correctly classified.
   (g) Evaluate negative samples by the newly computed strong classifier H and update the number of rejected negative samples

Table 1: Performance of the best haar feature over rounds 1 to 5

	Round 1	Round 2	Round 3	Round 4	Round 5
Haar feature type	5	2	1	5	1
Position	(2, 2)	(6, 17)	(17, 5)	(2, 2)	(9, 3)
Length	10	6	2	10	3
Height	8	2	4	8	12
Accuracy on training data	81	77	72	73	74
Alpha	0.75	0.61	0.49	0.51	0.55

Table 2: Performance of the best haar feature over rounds 6 to 10

	Round 6	Round 7	Round 8	Round 9	Round 10
Haar feature type	2	5	2	5	5
Position	(9, 2)	(2, 2)	(9, 2)	(2, 2)	(2, 2)
Length	8	10	8	10	10
Height	2	8	2	8	8
Accuracy on training data	81	75	78	78	75
Alpha	0.74	0.55	0.64	0.66	0.56

Adaboost evolution over number of rounds

- Adaboost round: 1

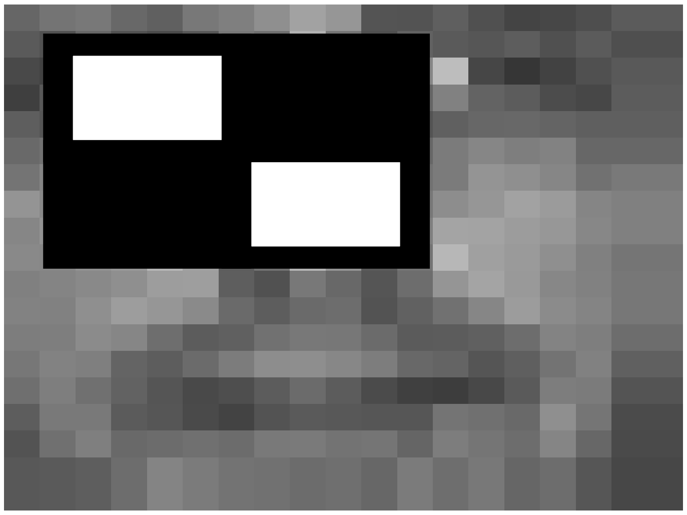

Figure 2: Top 1 feature after 1 rounds Adaboost

- Adaboost round: 2

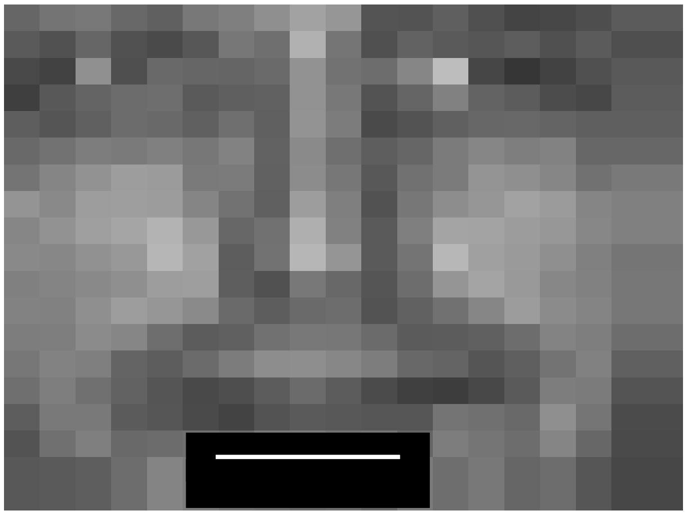

Figure 3: Top 1 feature after 2 rounds Adaboost

- Adaboost round: 3

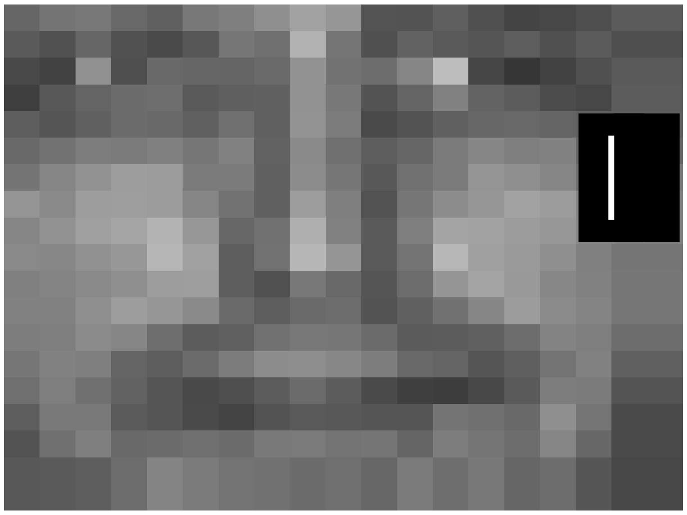

Figure 4: Top 1 feature after 3 rounds Adaboost

- Adaboost round: 4

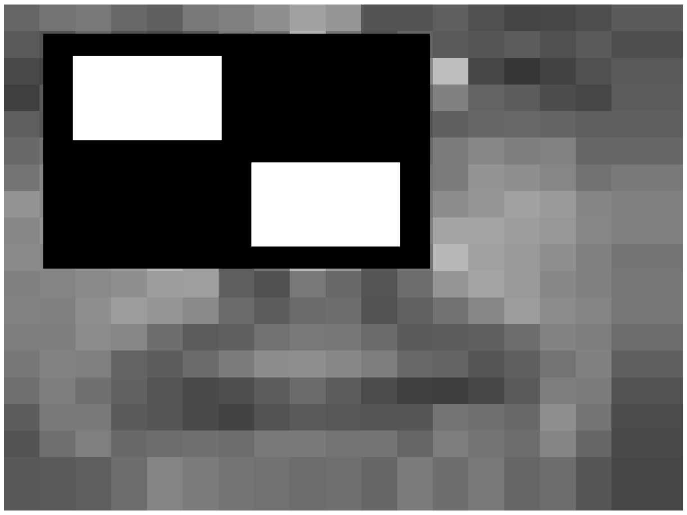

Figure 5: Top 1 feature after 4 rounds Adaboost

- Adaboost round: 5

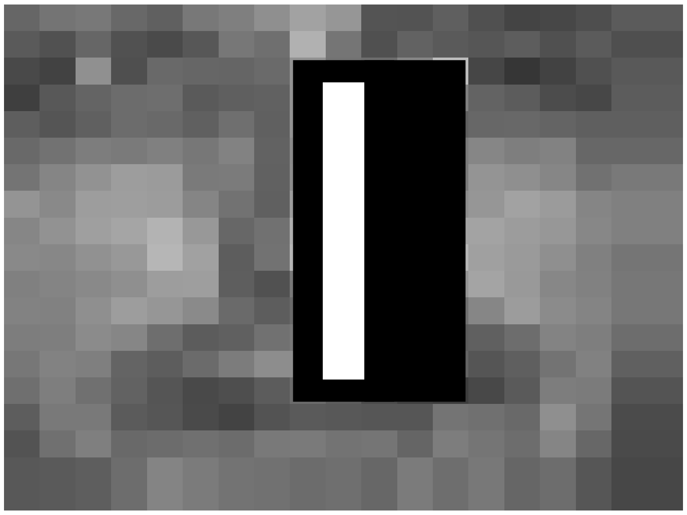

Figure 6: Top 1 feature after 5 rounds Adaboost

- Adaboost round: 6

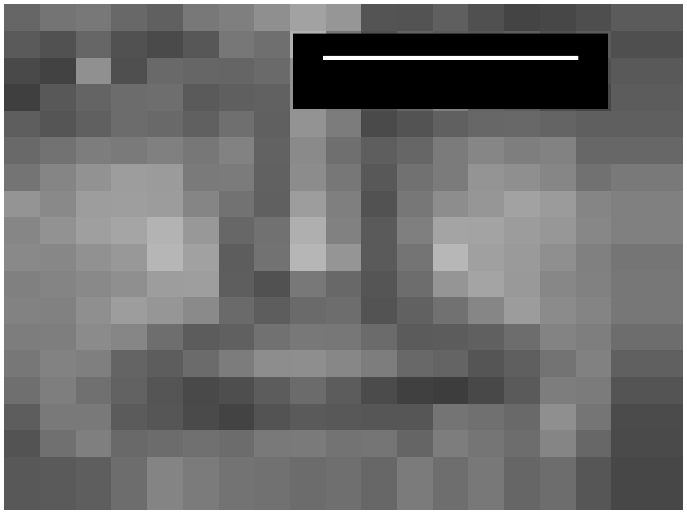

Figure 7: Top 1 feature after 6 rounds Adaboost

- Adaboost round: 7

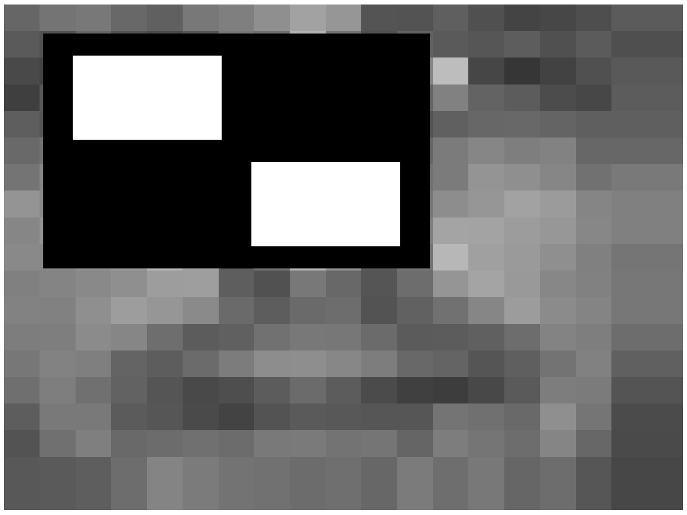

Figure 8: Top 1 feature after 7 rounds Adaboost

- Adaboost round: 8

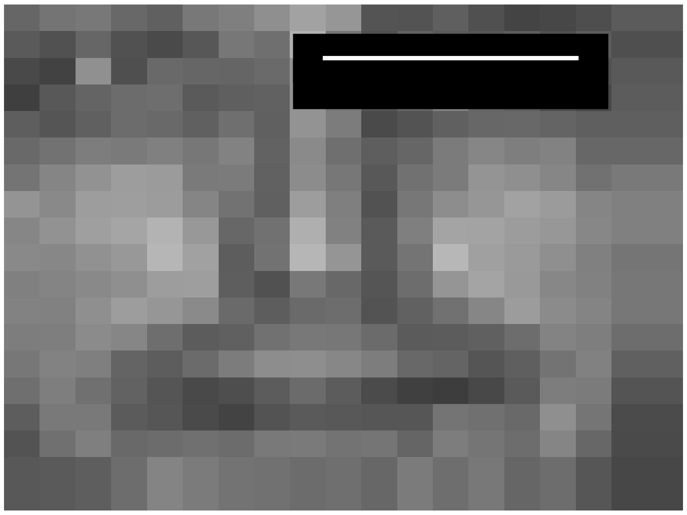

Figure 9: Top 1 feature after 8 rounds Adaboost

- Adaboost round: 9

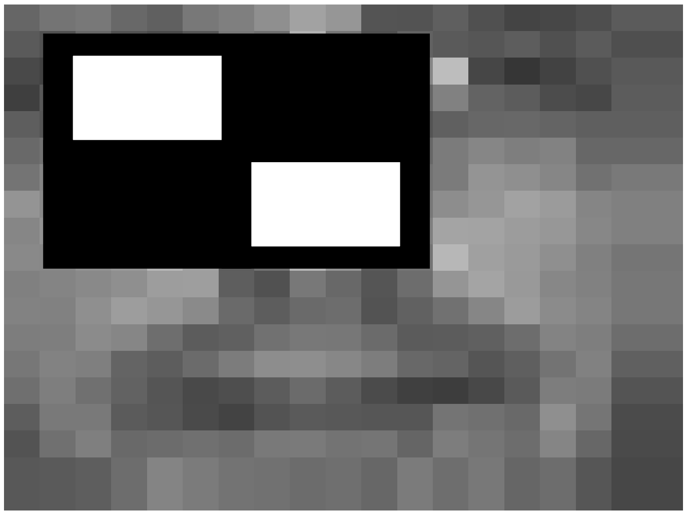

Figure 10: Top 1 feature after 9 rounds Adaboost

- Adaboost round: 10

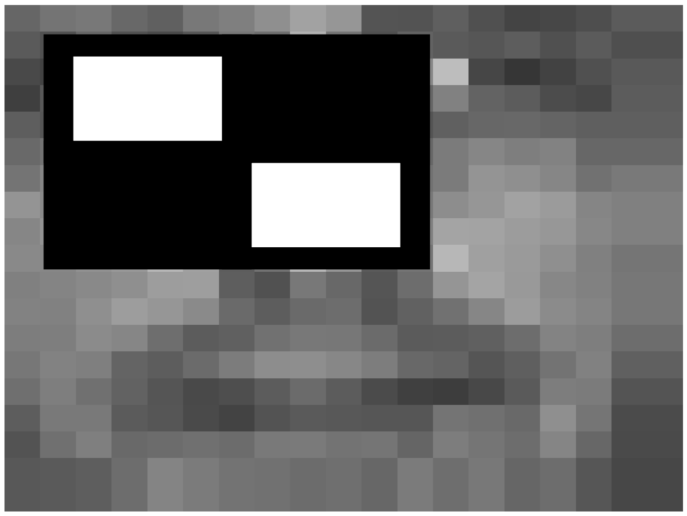

Figure 11: Top 1 feature after 10 rounds Adaboost

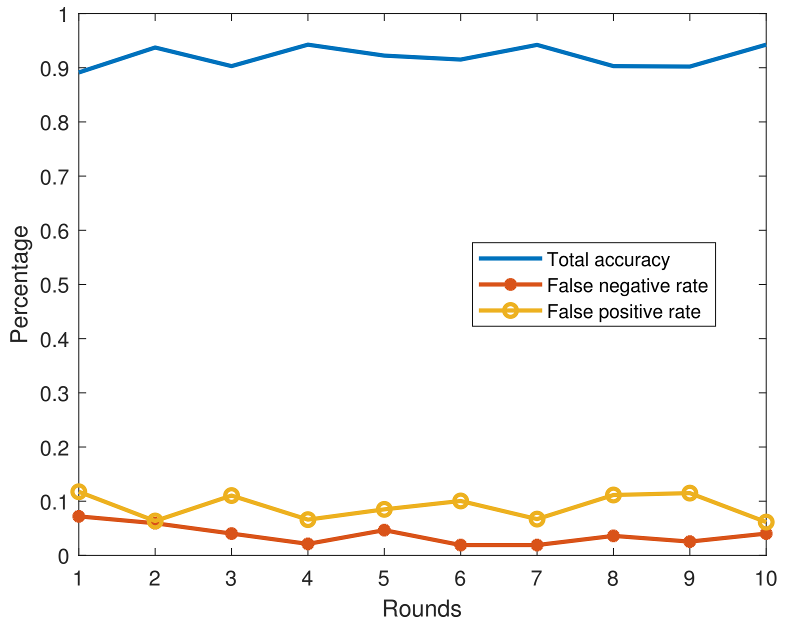

Figure 12: Accuracy,False Positive and False negative rate on test samples vs Rounds

Part 3 : Adjust the threshold

In this part I have recorded performance of the classifiers when False positive rate was used for selecting classifiers. AdaBoost algorithm was trained for 5 rounds.

- Adaboost round: 1

Table 3: Performance of the best haar feature over rounds 1 to 5 (False negative as a classifier selection criterion)

	Round 1	Round 2	Round 3	Round 4	Round 5
Haar feature type	2	2	2	2	2
Position	(8, 4)	(8, 2)	(8, 2)	(7, 2)	(8, 2)
Length	4	4	4	6	4
Height	11	11	5	4	5
Accuracy on training data	66	78	69	75	75
Alpha	0.34	0.63	0.42	0.57	0.56

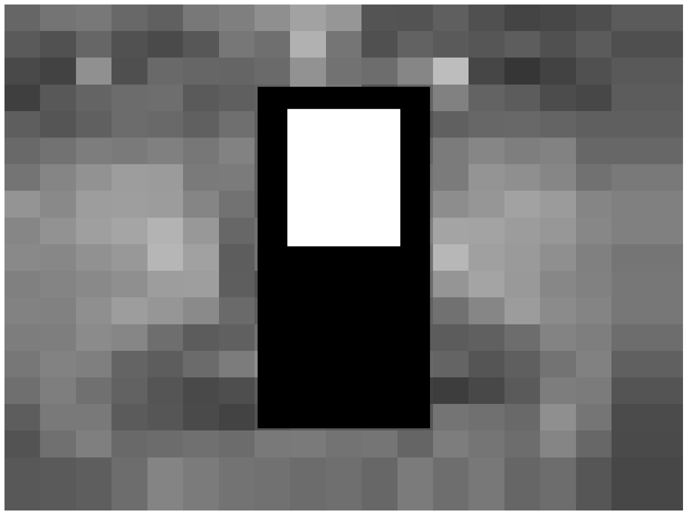

Figure 13: Top 1 feature after 1 rounds Adaboost (False negative as a classifier selection criterion)

- Adaboost round: 2

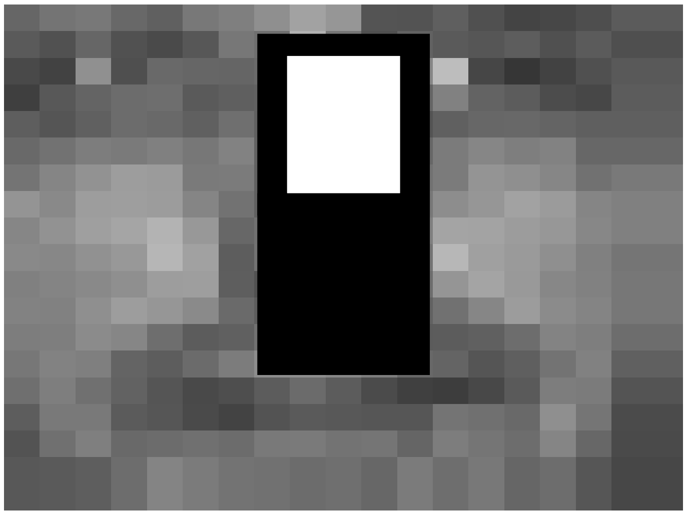

Figure 14: Top 1 feature after 2 rounds Adaboost (False negative as a classifier selection criterion)

- Adaboost round: 3

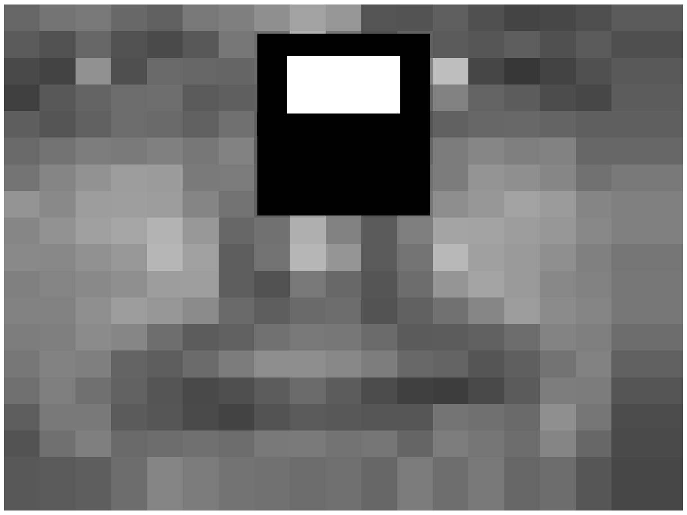

Figure 15: Top 1 feature after 3 rounds Adaboost (False negative as a classifier selection criterion)

- Adaboost round: 4

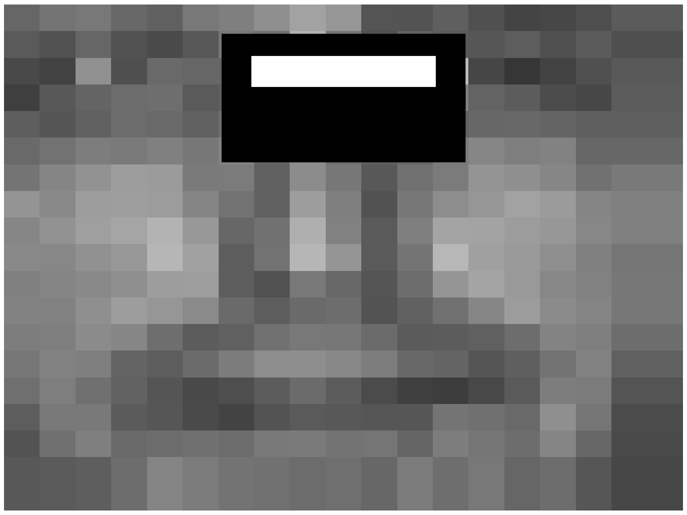

Figure 16: Top 1 feature after 4 rounds Adaboost (False negative as a classifier selection criterion)

- Adaboost round: 5

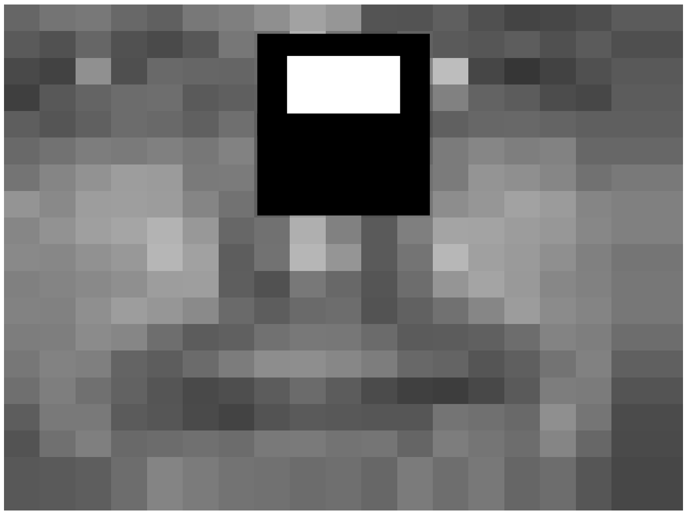

Figure 17: Top 1 feature after 5 rounds Adaboost (False negative as a classifier selection criterion)

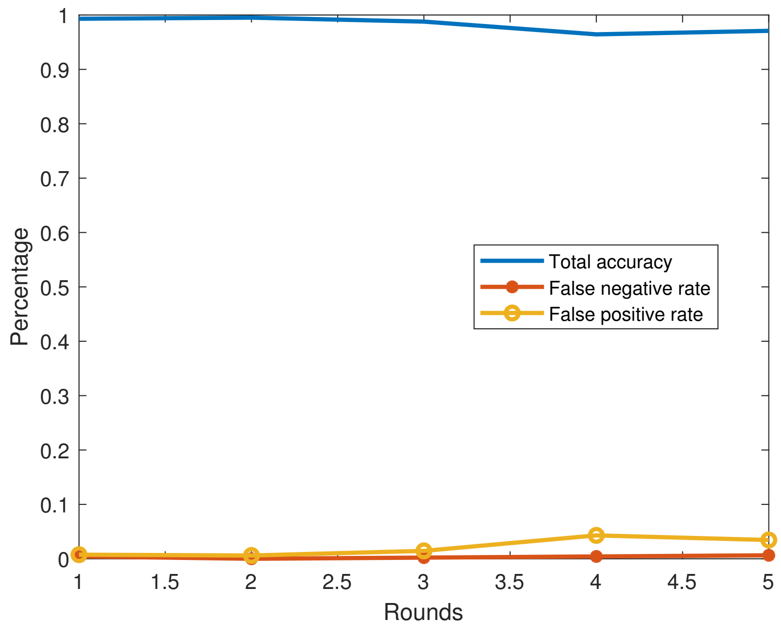

Figure 18: Accuracy,False Positive and False negative rate on test samples vs Rounds (False negative as a classifier selection criterion)

From the graph we can see that as number of rounds increases false positive rate also increase with it which support the fact that this criterion is not very useful in building a system for security reasons as system will recognize faces in non face objects too.

Unfortunately I was not able to train AdaBoost using false positive as a training criterion. But based on previous experience with false positive rate criterion we can say that as number of rounds increase false positive rate will increase.