ECEN 642, Fall 2019

Texas A&M University

Electrical and Computer Engineering Department

Dr. Raffaella Righetti

FINAL PROJECT REPORT

PROJECT (2) Face detector

Name: ANUSHKA WAINGANKAR

UIN: 230002195

Date: 12/10/2019

PROBLEM STATEMENT:

Implement a method to detect faces from 10 images. To construct the image set you can choose photos from different daily life scenes and with varied complexity (with a single face vs multiple faces). Upon completion of the method, you need to come up with some performance metric to evaluate your method. You can refer to some publications and try to reproduce the desired results.

ABSTRACT: Viola-jones Algorithm has been implemented and the same has been employed to detect faces from daily life scenes. The algorithm is able to process images rapidly and has high detection rates. The mentioned algorithm can be used in the domain of face detection and this has been implemented in this project. Here 5 different feature types are extracted from 19x19 images.

INTRODUCTION: Viola-Jones Algorithm is an ensemble method using which a series of weak classifiers to make a strong classifier. The algorithm of Viola-Jones introduces new concepts like that of integral images for feature evaluation and Adaboost which is a method using which a classifier is formed from just the important features. In order to incorporate the important features at every boosting stage a weak learner is taken as a feature selected. Then there by combining the complex classifiers in the cascaded structure the speed of the detector is seen to increase. Then implement the Adaboost algorithm on the features to perform the classification of the image. Finally, the classifier error definition is changed to consider only the case of False-negative and False positives

ALGORITHM:

Feature extraction:

- Classification of images is based on simple features as feature based systems are faster than pixel-based systems.

Integral Image:

- Integral image at any location can be calculated using the formula given and can be used to compute the original image in one pass.

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

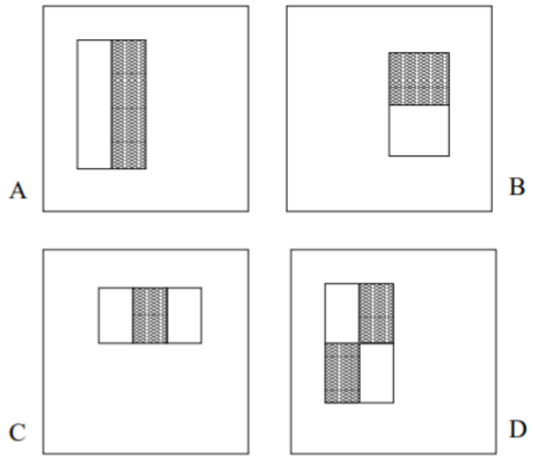
$where\ ii(x,y)$: Integral Image & $i(x,y)$: Original Image

$ii(x,y) = ii(x-1,y) + s(x,y\ )$ $\quad and \quad$ $s(x,y) = s(x,y-1) + i(x,y)$

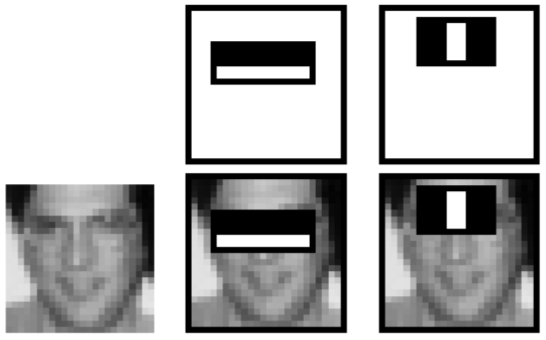
The feature size was limited by two constraints, one that area of the two black and white regions must be equal and the max size of the filter is 8 by 8. The computations were performed using an integral image to decrease computation time.

Haar-Like Features:

- Four features are used namely, there are horizontal and vertical two-rectangle features where the values are the difference between the sum of the pixels in two rectangular regions. A three-rectangle feature where the sum within two outside rectangles subtracted from the sum in the center rectangle and a four-rectangle feature computes the difference between diagonal pairs of rectangles

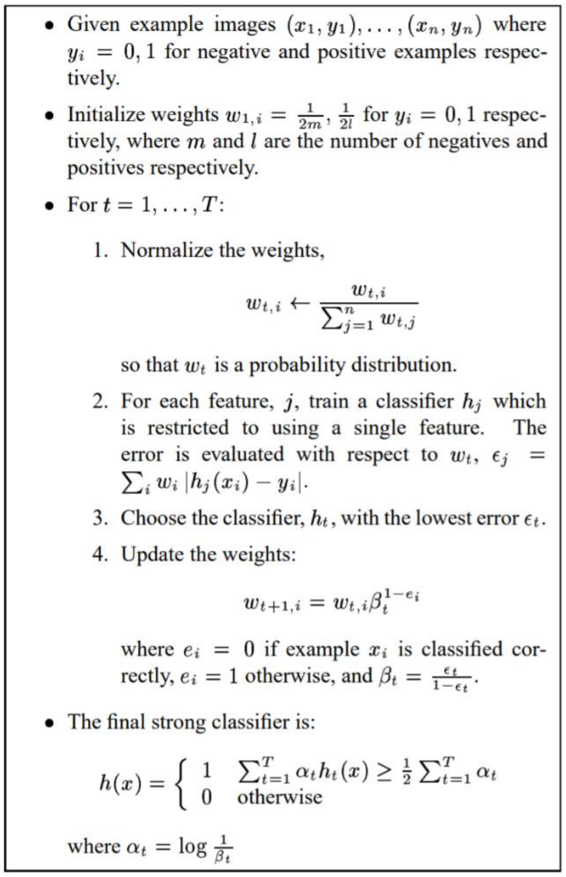


- The two features shown in the right figure tend to focus on the fact that the eye region tends to be darker than the bridge of the nose or the cheek region. Theis combination of features alone can give 100% detection rate and a 50% false positive rate



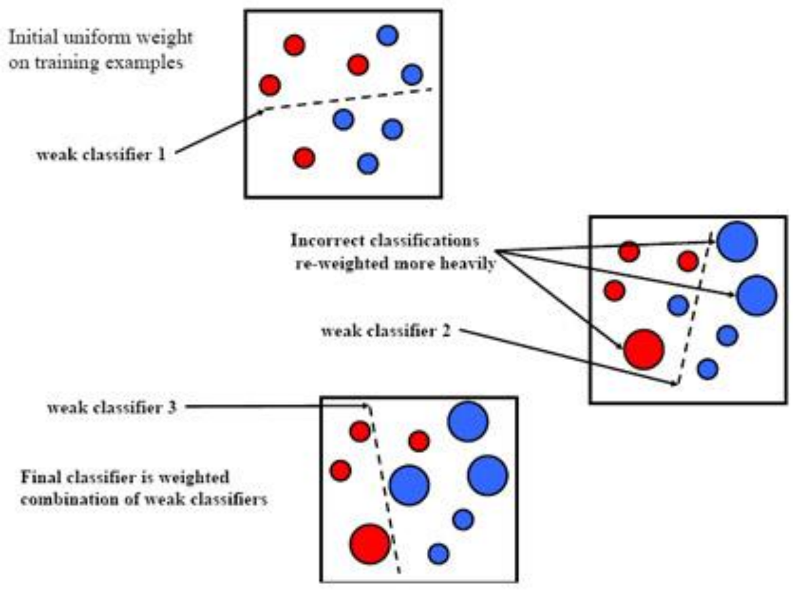
AdaBoost Classification Function:

- There are certain features which give consistently high values on faces and are to include these features, adaboost is used to select and train some features from the classifier. AdaBoost is used to boost the classification performance of a weak classifier.
- The only hyper-parameter used is the number of weak classifiers. First step is to initialize uniform weights for positive and negative images. Then the best feature is found which gives minimal error and this feature is assigned to the weak classifier in strong classifier. Each weak classifier by itself cannot accurately fulfill the classification task. series of weak classifiers and weights their results together to produce the final classification



- Given example images $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \dots, T$:
  1. Normalize the weights,
     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
     so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
  $$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
  where $\alpha_t = \log \frac{1}{\beta_t}$

- The explaination of how adaboost is used such that the weak classifiers can become strong classifiers. Basically the AdaBoost learninf algorithm chooses the most effieicnt weak classifier whose weight will be increased so that the next weak classifier will focus more on the hard ones, and finally we get a combination of straing classifiers.


Initial uniform weight on training examples
weak classifier 1
Incorrect classifications re-weighted more heavily
weak classifier 2
weak classifier 3
Final classifier is weighted combination of weak classifiers

$$H(x) = sign(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$$

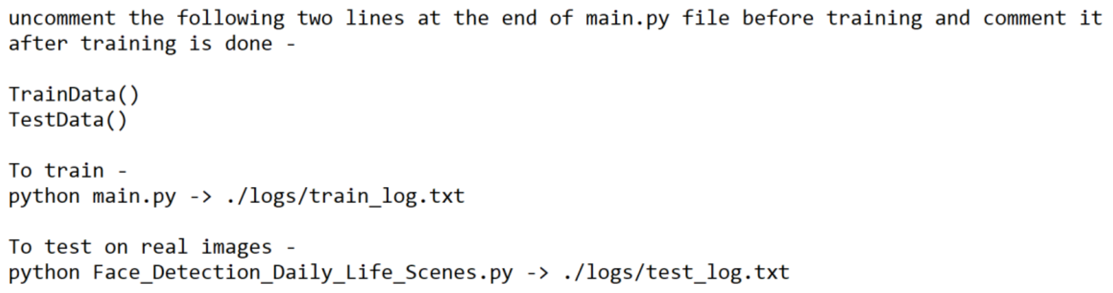
IMPLEMENTATION AND RESULTS:

→ The implementation of the Viola-Jones Algorithm was done on Python and using various packages such as NumPy, mathplot, OpenCV, pickle, SciKit-Learn package and many more.

→ In order to reduce the computation time, using pickle we can save and load models from a file and using SelectPercentile class from Scikit-Learn package helps to reduce the number of features which will result in less number of weak classifiers.

→ For training and testing an online dataset was used which consisted of trainset and test-set. Trainset had 499 faces and 2000 non-face and test-set has 472 faces and 2000 non-faces. Here each image is of 19x19 pixels as opposed to 24x24 mentioned in the paper.

→ To run the implemented algorithm first the dataset is trained and its results are stored in train_log,txt file which has both the results and the performance metrics on on the train and test data.



```
uncomment the following two lines at the end of main.py file before training and comment it
after training is done -

TrainData()
TestData()

To train -
python main.py -> ./logs/train_log.txt

To test on real images -
python Face_Detection_Daily_Life_Scenes.py -> ./logs/test_log.txt
```

→ The performance metrics considered are the accuracy and the training accuracy and false positive and false negative rates on the training and testing data.

→ Further in order to implement the algorithm on daily life images scenes the following can be used and the images with the faces detected are directly stored in DailyLifeScenes-result. The test_log file contains all the face detected locations.

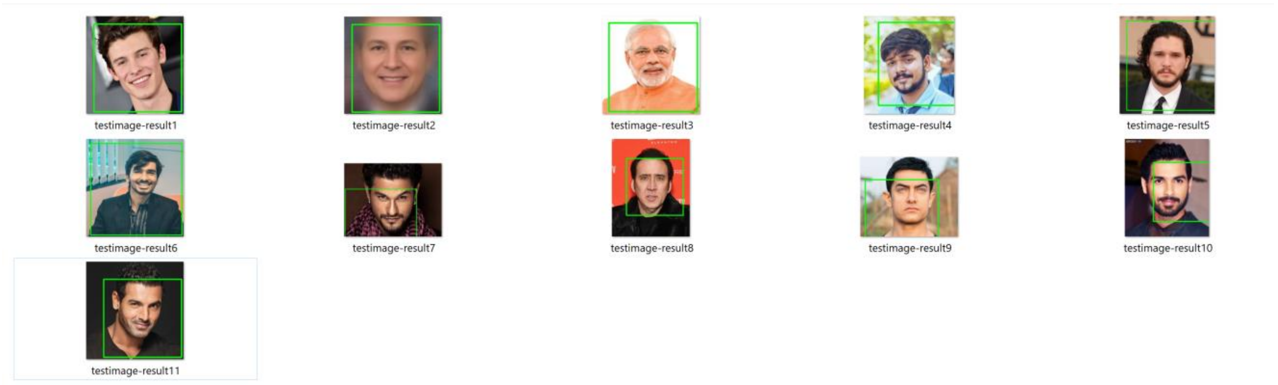
→ The threshold value I have set it to 0.6 with the intention of reducing the false positives in the prediction and separating positive and negative images. So, the images above 0.6 will be classified as positive.

→ I have followed the algorithm given in the paper and have implemented the technique of non-maximal suppression as well to remove the overlapping windows or square boxes from appearing.

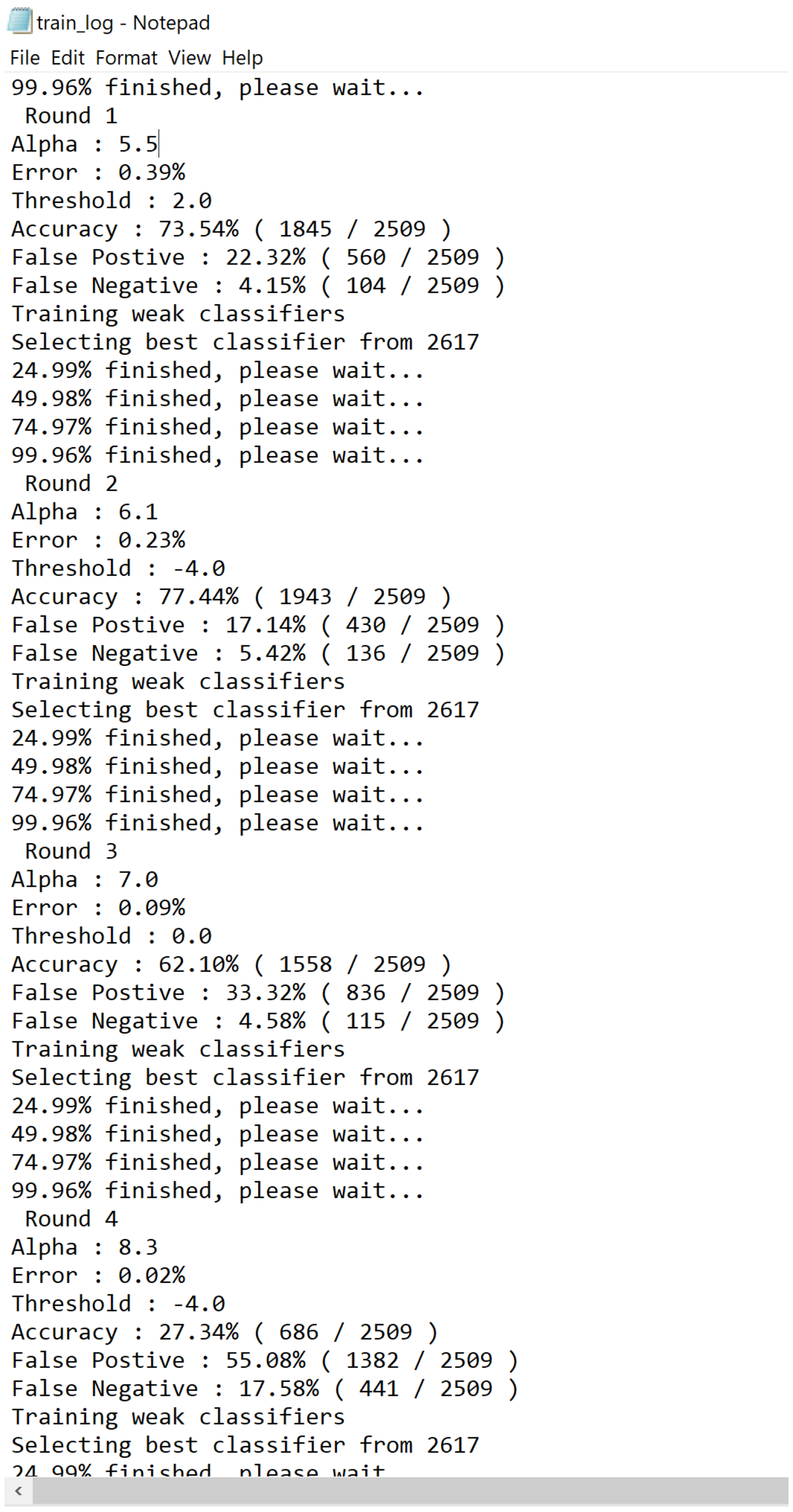
→ The performance metrics for the algorithm are that in each round of the Adaboost classifier both for the training and the testing data present in dataset the the training accuracy and error, false positive and false negative in each case. Using this the analysis of the model can be made. By changing the threshold, the false positive and negative percentages can be changed according to our requirement. The challenge in face detection using Adaboost is the number of false-positive results that accompany actual faces detected on a complex background

Result of face detection using the implemented Viola-Jones Algorithm

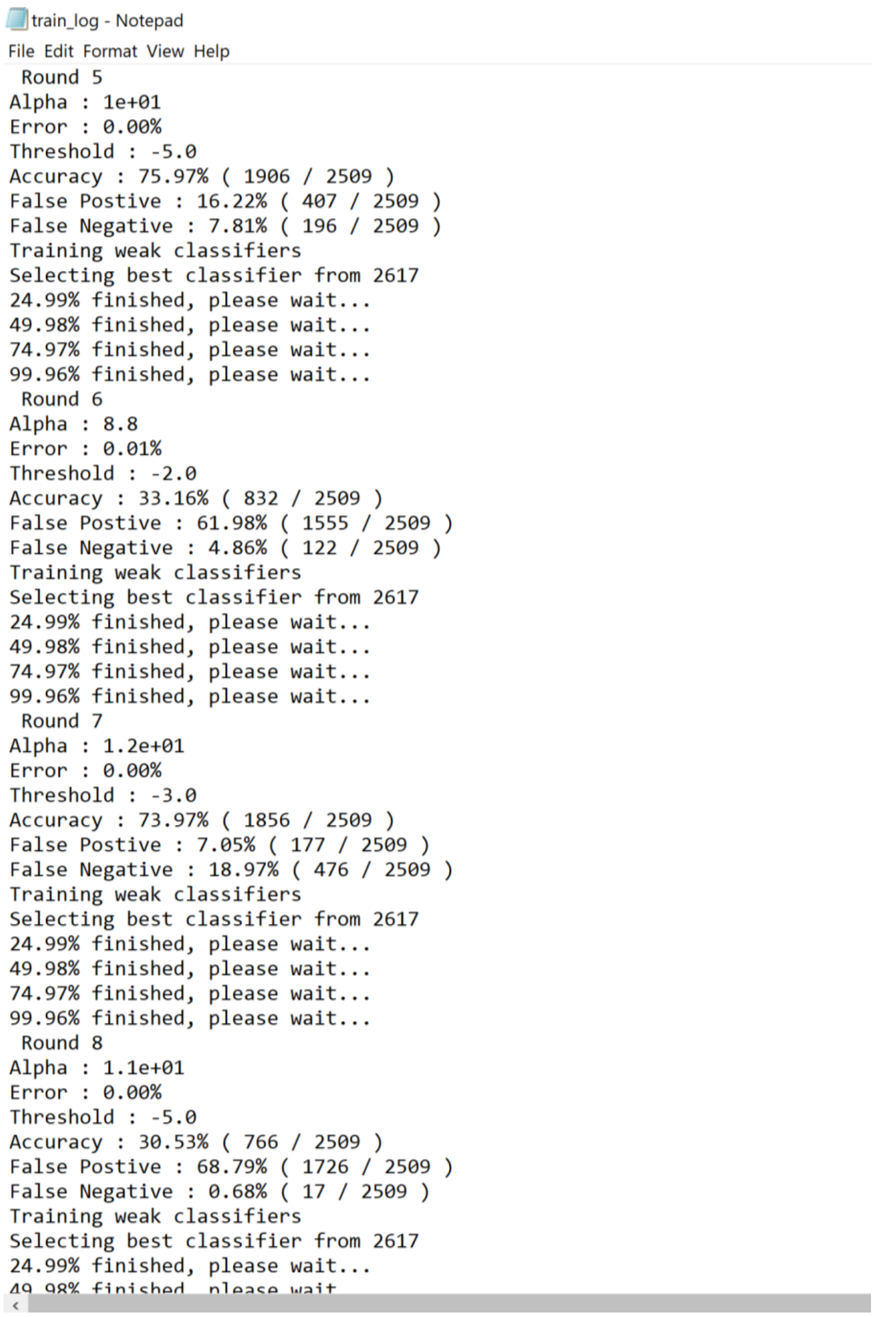
The result of faces detected are stored in the folder are given below



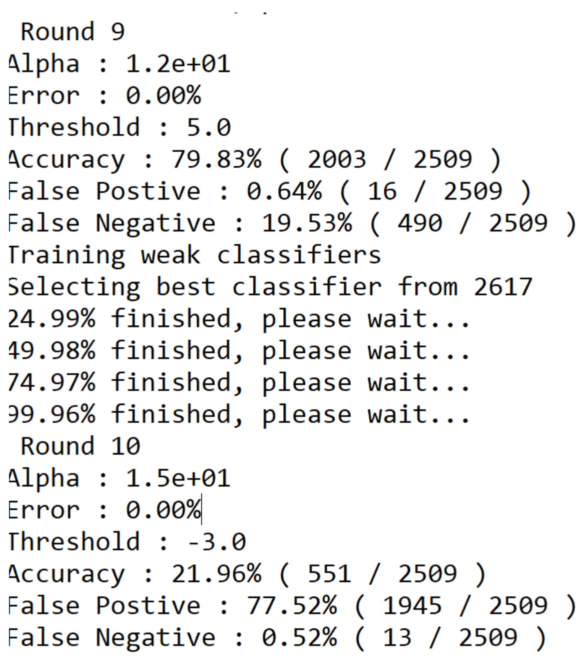
The train_log file shows the below mentioned results



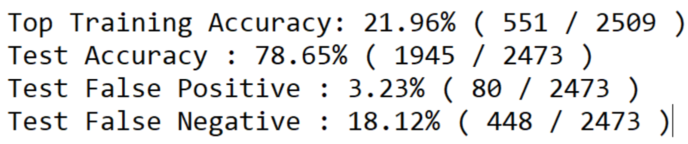
```
99.96% finished, please wait...
 Round 1
Alpha : 5.5
Error : 0.39%
Threshold : 2.0
Accuracy : 73.54% ( 1845 / 2509 )
False Postive : 22.32% ( 560 / 2509 )
False Negative : 4.15% ( 104 / 2509 )
Training weak classifiers
Selecting best classifier from 2617
24.99% finished, please wait...
49.98% finished, please wait...
74.97% finished, please wait...
99.96% finished, please wait...
 Round 2
Alpha : 6.1
Error : 0.23%
Threshold : -4.0
Accuracy : 77.44% ( 1943 / 2509 )
False Postive : 17.14% ( 430 / 2509 )
False Negative : 5.42% ( 136 / 2509 )
Training weak classifiers
Selecting best classifier from 2617
24.99% finished, please wait...
49.98% finished, please wait...
74.97% finished, please wait...
99.96% finished, please wait...
 Round 3
Alpha : 7.0
Error : 0.09%
Threshold : 0.0
Accuracy : 62.10% ( 1558 / 2509 )
False Postive : 33.32% ( 836 / 2509 )
False Negative : 4.58% ( 115 / 2509 )
Training weak classifiers
Selecting best classifier from 2617
24.99% finished, please wait...
49.98% finished, please wait...
74.97% finished, please wait...
99.96% finished, please wait...
 Round 4
Alpha : 8.3
Error : 0.02%
Threshold : -4.0
Accuracy : 27.34% ( 686 / 2509 )
False Postive : 55.08% ( 1382 / 2509 )
False Negative : 17.58% ( 441 / 2509 )
Training weak classifiers
Selecting best classifier from 2617
24.99% finished, please wait
```



```
 Round 5
Alpha : 1e+01
Error : 0.00%
Threshold : -5.0
Accuracy : 75.97% ( 1906 / 2509 )
False Postive : 16.22% ( 407 / 2509 )
False Negative : 7.81% ( 196 / 2509 )
Training weak classifiers
Selecting best classifier from 2617
24.99% finished, please wait...
49.98% finished, please wait...
74.97% finished, please wait...
99.96% finished, please wait...
 Round 6
Alpha : 8.8
Error : 0.01%
Threshold : -2.0
Accuracy : 33.16% ( 832 / 2509 )
False Postive : 61.98% ( 1555 / 2509 )
False Negative : 4.86% ( 122 / 2509 )
Training weak classifiers
Selecting best classifier from 2617
24.99% finished, please wait...
49.98% finished, please wait...
74.97% finished, please wait...
99.96% finished, please wait...
 Round 7
Alpha : 1.2e+01
Error : 0.00%
Threshold : -3.0
Accuracy : 73.97% ( 1856 / 2509 )
False Postive : 7.05% ( 177 / 2509 )
False Negative : 18.97% ( 476 / 2509 )
Training weak classifiers
Selecting best classifier from 2617
24.99% finished, please wait...
49.98% finished, please wait...
74.97% finished, please wait...
99.96% finished, please wait...
 Round 8
Alpha : 1.1e+01
Error : 0.00%
Threshold : -5.0
Accuracy : 30.53% ( 766 / 2509 )
False Postive : 68.79% ( 1726 / 2509 )
False Negative : 0.68% ( 17 / 2509 )
Training weak classifiers
Selecting best classifier from 2617
24.99% finished, please wait...
49.98% finished, please wait...
```

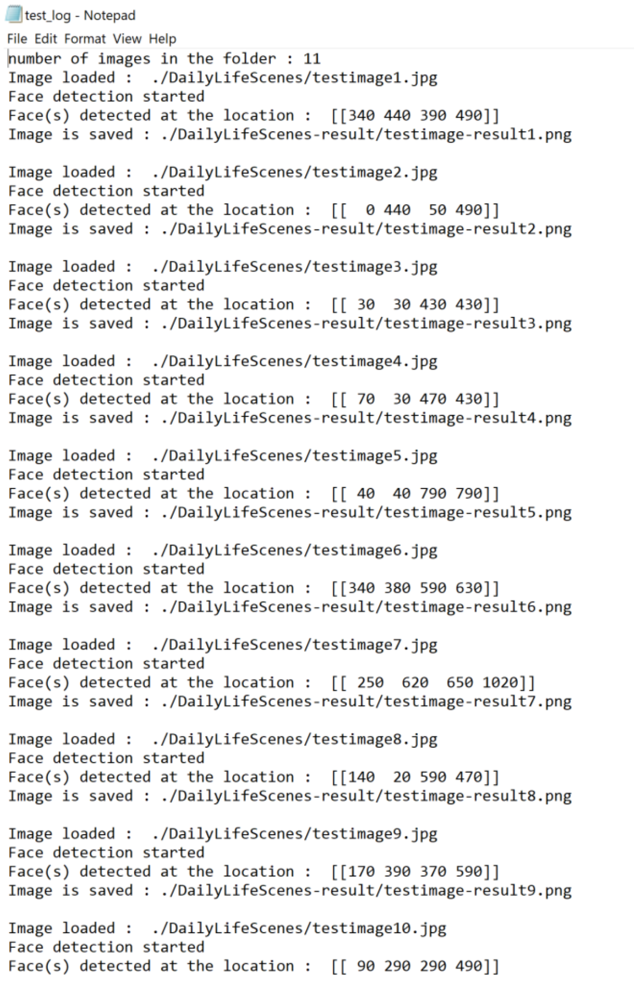


```
 Round 9
Alpha : 1.2e+01
Error : 0.00%
Threshold : 5.0
Accuracy : 79.83% ( 2003 / 2509 )
False Postive : 0.64% ( 16 / 2509 )
False Negative : 19.53% ( 490 / 2509 )
Training weak classifiers
Selecting best classifier from 2617
24.99% finished, please wait...
49.98% finished, please wait...
74.97% finished, please wait...
99.96% finished, please wait...
 Round 10
Alpha : 1.5e+01
Error : 0.00%
Threshold : -3.0
Accuracy : 21.96% ( 551 / 2509 )
False Postive : 77.52% ( 1945 / 2509 )
False Negative : 0.52% ( 13 / 2509 )
```



```
Top Training Accuracy: 21.96% ( 551 / 2509 )
Test Accuracy : 78.65% ( 1945 / 2473 )
Test False Positive : 3.23% ( 80 / 2473 )
Test False Negative : 18.12% ( 448 / 2473 )
```

The test_log file shows the below mentioned results



```
number of images in the folder : 11
Image loaded :  ./DailyLifeScenes/testimage1.jpg
Face detection started
Face(s) detected at the location :  [[340 440 390 490]]
Image is saved : ./DailyLifeScenes-result/testimage-result1.png

Image loaded :  ./DailyLifeScenes/testimage2.jpg
Face detection started
Face(s) detected at the location :  [[  0 440  50 490]]
Image is saved : ./DailyLifeScenes-result/testimage-result2.png

Image loaded :  ./DailyLifeScenes/testimage3.jpg
Face detection started
Face(s) detected at the location :  [[ 30  30 430 430]]
Image is saved : ./DailyLifeScenes-result/testimage-result3.png

Image loaded :  ./DailyLifeScenes/testimage4.jpg
Face detection started
Face(s) detected at the location :  [[ 70  30 470 430]]
Image is saved : ./DailyLifeScenes-result/testimage-result4.png

Image loaded :  ./DailyLifeScenes/testimage5.jpg
Face detection started
Face(s) detected at the location :  [[ 40  40 790 790]]
Image is saved : ./DailyLifeScenes-result/testimage-result5.png

Image loaded :  ./DailyLifeScenes/testimage6.jpg
Face detection started
Face(s) detected at the location :  [[340 380 590 630]]
Image is saved : ./DailyLifeScenes-result/testimage-result6.png

Image loaded :  ./DailyLifeScenes/testimage7.jpg
Face detection started
Face(s) detected at the location :  [[ 250  620  650 1020]]
Image is saved : ./DailyLifeScenes-result/testimage-result7.png

Image loaded :  ./DailyLifeScenes/testimage8.jpg
Face detection started
Face(s) detected at the location :  [[140  20 590 470]]
Image is saved : ./DailyLifeScenes-result/testimage-result8.png

Image loaded :  ./DailyLifeScenes/testimage9.jpg
Face detection started
Face(s) detected at the location :  [[170 390 370 590]]
Image is saved : ./DailyLifeScenes-result/testimage-result9.png

Image loaded :  ./DailyLifeScenes/testimage10.jpg
Face detection started
Face(s) detected at the location :  [[ 90 290 290 490]]
```

SCOPE FOR IMPROVEMENT:

- → The implementation of Viola-jones can be improved further by making use of attentional cascade, where the cascading classifiers are used to improve the speed by early rejection of non-face regions by simple classifiers and can reduce the false positive rates as well.
- → The present implementation is not able to detect multiple faces from an image but can detect single face when present in the image. Multiscale parameters could be incorporated in order to do so.
- → The data-set used for training and testing is less and that is one more reason why the results are not as good as the in-built function of haarcascade present in OpenCV which has been trained on a much larger dataset.

CONCLUSION:

The Viola -Jones algorithm still has its impact face recognition even after years of its discovery and a successful implementation of the Viola-jones algorithm leads to the conclusion that itis a robust detector, but it has a very long training period. The advantages of using it are that it classifies images quickly because each weak classifier requires only a small number of parameters, and with sufficient number of weak classifiers, it has a low rate of false positives. The results observe were not the standard results, because of feature size was limited to only 8x8 and dataset was limited so the disparity between false positive and false negative was quite prominent.

REFERENCES:

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Kauai, HI, USA, 2001, pp. I-I. doi: 10.1109/CVPR.2001.990517

[2] O. H. Jensen. Implementing the Viola-Jones face detection algorithm. PhD thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2008.

[3] M. Pound, “Detecting Faces (Viola Jones Algorithm)”, YouTube -Computerphile, Oct 2018.

[4] https://medium.com/greyatom/a-quick-guide-to-boosting-in-ml-acf7c1585cb5