Pattern Recognition [ECEN 649]

Project 5 Report

VIOLA JONES FACE DETECTION

Submitted by:

Harshita Chaudhary

UIN: 529005682

Email: harshita@tamu.edu

Objective

The objective of the project is identifying faces and non-faces in images using the Viola-Jones algorithm based on the research paper on **Rapid Object Detection using a Boosted Cascade of Simple Features.**

Introduction

The Viola-Jones face detection scheme is one of the most well-known applications of the AdaBoost algorithm. The detail of the detection scheme has been documented in the original Viola-Jones paper. It is considered one of the state-of-the-art face detectors. The original implementation in 2001/2004 claimed a detection speed of 0.07 seconds per frame of size ~300 x 300 on a standard desktop.

AdaBoost

AdaBoost is an ensemble learning algorithm. A selection of classifiers is required, called poor learners or simple learners (like a rule of thumb).

To generate a good classifier, it combines them. A powerful classifier is one that will yield good results on unseen data. Detection of faces requires a binary classifier (face versus non-face). There are multi-class variants at AdaBoost, but in this project we use the 2-class scenario.

Bad learners (thumb rules) must have an error rate of less than 50%, i.e. they must be marginally better than random guessing at least. It is simpler to find and then combine several basic (weak) rules of thumb than to find one complex (accurate) law.

AdaBoost

input:
training set $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
weak learner WL
number of rounds T
initialize $\mathbf{D}^{(1)} = (\frac{1}{m}, \ldots, \frac{1}{m})$.
for $t = 1, \ldots, T$:
invoke weak learner $h_t = \mathrm{WL}(\mathbf{D}^{(t)}, S)$
compute $\epsilon_t = \sum_{i=1}^{m} D_i^{(t)} \mathbb{1}_{[y_i \neq h_t(\mathbf{x}_i)]}$
let $w_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t} - 1\right)$
update $D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(\mathbf{x}_i))}{\sum_{j=1}^{m} D_j^{(t)} \exp(-w_t y_j h_t(\mathbf{x}_j))}$ for all $i = 1, \ldots, m$
output the hypothesis $h_s(\mathbf{x}) = \mathrm{sign}\left(\sum_{t=1}^{T} w_t h_t(\mathbf{x})\right)$.

Integral Image

A single rectangle feature can be computed using four array references into the integral image.

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

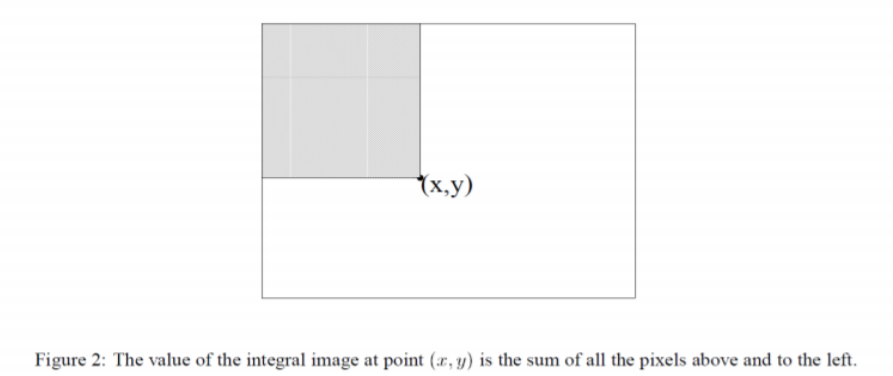


Figure 2: The value of the integral image at point (x, y) is the sum of all the pixels above and to the left.

The integral image can be computed with a single pass over the image, if you use the following recurrences:

$$\begin{aligned} s(x, y) &= s(x, y-1) + i(x, y) \\ ii(x, y) &= ii(x-1, y) + s(x, y) \end{aligned}$$

$s(x, y)$ is the cumulative row sum, $s(x, -1) = 0$

$ii(-1, y) = 0$

A single rectangle feature can be computed using four array references into the integral image.

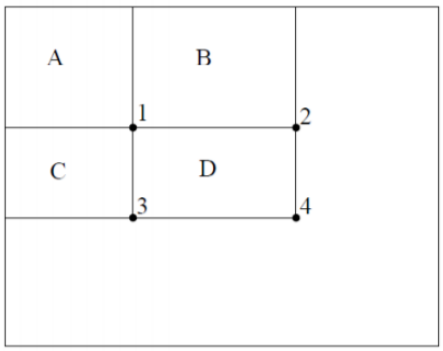
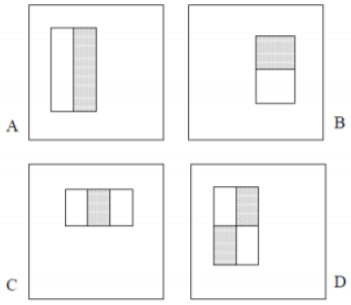


Figure 3: The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is $A + B$, at location 3 is $A + C$, and at location 4 is $A + B + C + D$. The sum within D can be computed as $4 + 1 - (2 + 3)$.

Haar Features

The Viola-Jones detection algorithm uses simple sums and differences of rectangles – these features are computationally very efficient and are called Haar-like features.

Image taken from: P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision, 57(2):137–154, 2004.*



We can find the value easily in any rectangle using the computed integral images. For instance, the value in Figure 2 is essentially the sum that can be computed as (4+1) - (2+3) within D. This is effective computing.

Types of Haar Features used in this project:

5 basic types of Haar-like features are used:

- Horizontal feature with two rectangles
- Vertical feature with two rectangles
- Vertical feature with three rectangles
- Horizontal feature with three rectangles
- Diagonal feature with four rectangles

Number of Haar Features of each type used in this project:

Two Vertical: 17100

Two Horizontal: 17100

Three Vertical: 10830

Three Horizontal: 10830

Four Diagonal: 8100

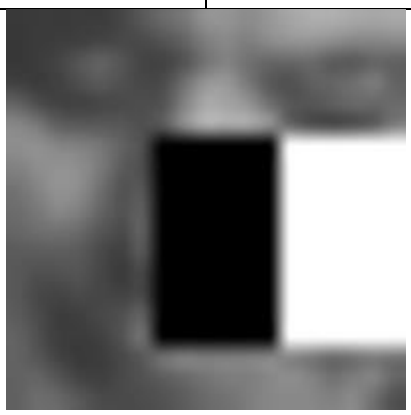
Total number of features: 63960

PART 1

The top 10 features selected by AdaBoost:

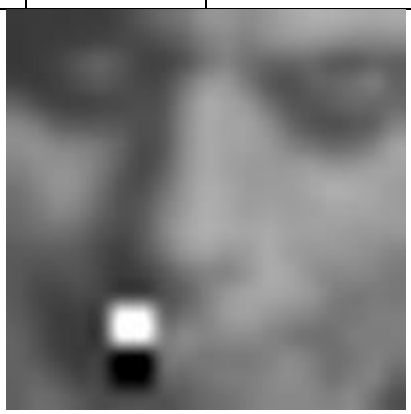
- Feature 1

Position	Feature Type	Width	Height	Threshold	Polarity
(7,6)	Two Vertical	12	10	0.00025	-1



- Feature 2

Position	Feature Type	Width	Height	Threshold	Polarity
(5, 14)	Two Horizontal	2	4	0.00016	1

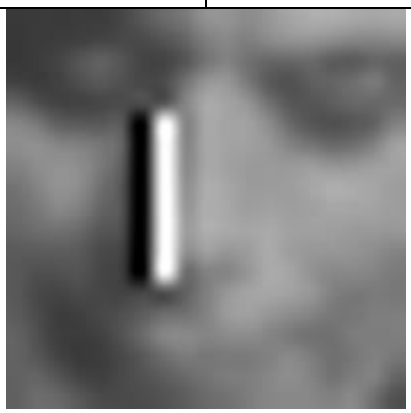


- Feature 3

Position	Feature Type	Width	Height	Threshold	Polarity
(5,2)	Three Vertical	9	1	0.001517	-1

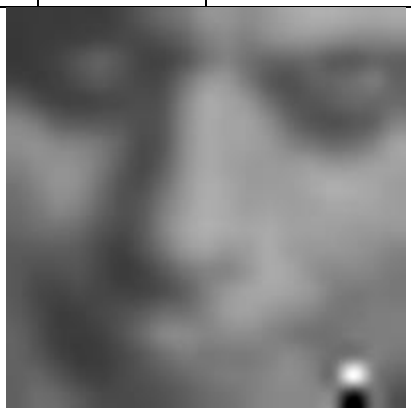
- Feature 4

Position	Feature Type	Width	Height	Threshold	Polarity
(6,5)	Two Vertical	2	8	0.00032	1



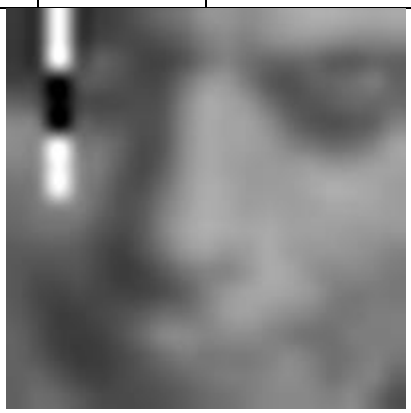
- Feature 5

Position	Feature Type	Width	Height	Threshold	Polarity
(16,17)	Two Horizontal	1	2	5.5183	-1



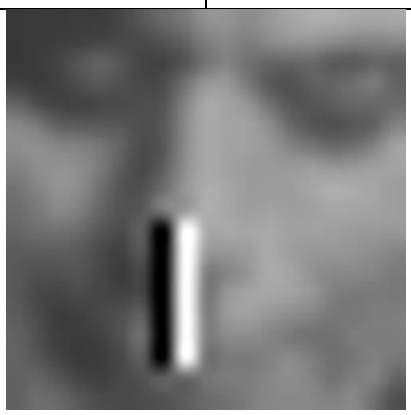
- Feature 6

Position	Feature Type	Width	Height	Threshold	Polarity
(2,0)	Three Horizontal	1	9	0.002577	-1



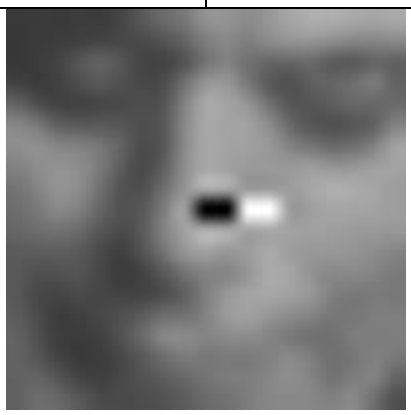
- Feature 7

Position	Feature Type	Width	Height	Threshold	Polarity
(7,10)	Two Vertical	2	7	2.8908	-1



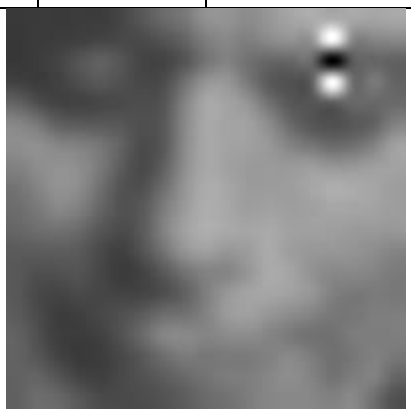
- Feature 8

Position	Feature Type	Width	Height	Threshold	Polarity
(9,9)	Two Vertical	4	1	1.97678	1



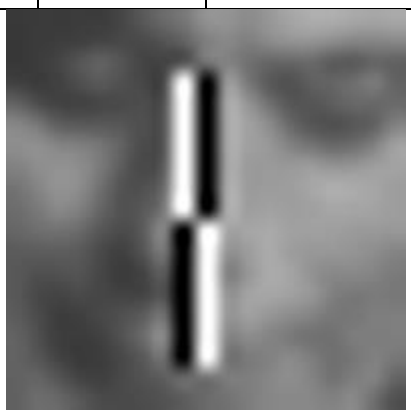
- Feature 9

Position	Feature Type	Width	Height	Threshold	Polarity
(15,1)	Three Horizontal	1	3	0.00227	-1



- Feature 10

Position	Feature Type	Width	Height	Threshold	Polarity
(8,3)	Four Diagonal	2	14	1.01348	-1



Key Observations:

1. AdaBoost selects features that differentiate the high illumination regions of the face to low illumination ones.
2. It detects edges, like around the eyebrows, lips, and nose.

Part 2

The combined classifiers after running 1, 3, 5, and 10 boosting rounds.

Boosting Rounds: 1

Position	Feature Type	Width	Height	Threshold	Polarity	Alpha
(7,6)	Two Vertical	12	10	0.00025	-1	1.2165

Training

Accuracy: 0.6618

False Positive Rate: 0.411

False Negative Rate: 0.0460

Testing

Accuracy: 0.4268

False Positive Rate: 0.5781

False Negative Rate: 0.3665

Boosting Rounds: 3

Position	Feature Type	Width	Height	Threshold	Polarity	Alpha
(7,6)	Two Vertical	12	10	0.00025	-1	1.2165
(5, 14)	Two Horizontal	2	4	0.00016	1	0.95505
(5,2)	Three Vertical	9	1	0.001517	-1	0.8468

Training

Accuracy: 0.8231

False Positive Rate: 0.173

False Negative Rate: 0.1923

Testing

Accuracy: 0.7011

False Positive Rate: 0.2876

False Negative Rate: 0.7669

Boosting Rounds: 5

Position	Feature Type	Width	Height	Threshold	Polarity	Alpha
(7,6)	Two Vertical	12	10	0.00025	-1	1.2165
(5, 14)	Two Horizontal	2	4	0.00016	1	0.95505
(5,2)	Three Vertical	9	1	0.001517	-1	0.8468
(6,5)	Two Vertical	2	8	0.00032	1	0.9754
(16,17)	Two Horizontal	1	2	5.5183	-1	0.7989

Training

Accuracy: 0.8355

False Positive Rate: 0.1965

False Negative Rate: 0.0360

Testing

Accuracy: 0.6645

False Positive Rate: 0.3251

False Negative Rate: 0.7605

Boosting Rounds: 10

Position	Feature Type	Width	Height	Threshold	Polarity	Alpha
(7,6)	Two Vertical	12	10	0.00025	-1	1.2165
(5, 14)	Two Horizontal	2	4	0.00016	1	0.95505
(5,2)	Three Vertical	9	1	0.001517	-1	0.8468
(6,5)	Two Vertical	2	8	0.00032	1	0.9754
(16,17)	Two Horizontal	1	2	5.5183	-1	0.7989
(2,0)	Three Horizontal	1	9	0.002577	-1	0.8088
(7,10)	Two Vertical	2	7	2.8908	-1	1.0391
(9,9)	Two Vertical	4	1	1.97678	1	0.6813
(15,1)	Three Horizontal	1	3	0.00227	-1	0.9789
(8,3)	Four Diagonal	2	14	1.01348	-1	0.9847

Training

Accuracy: 0.9003

False Positive Rate: 0.124

False Negative Rate: 0.002

Testing

Accuracy: 0.8145

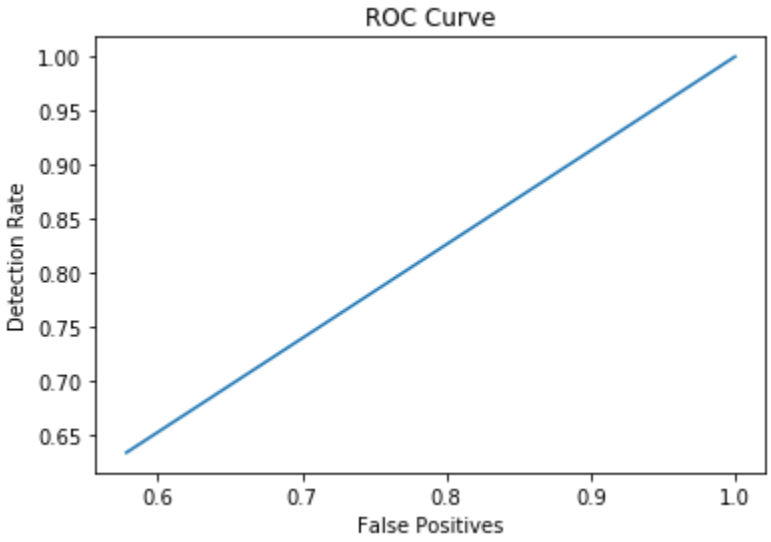
False Positive Rate: 0.1718

False Negative Rate: 0.7521

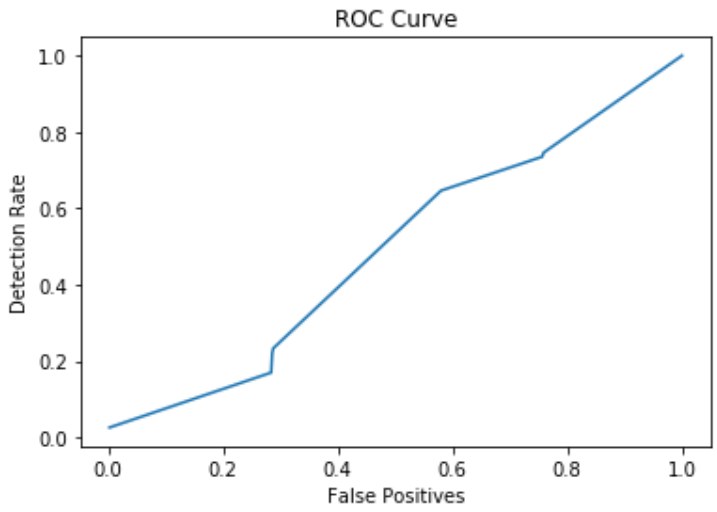
Part3

Following are the ROC curves of the combined classifiers after running 1, 3, 5, and 10 boosting rounds when applied to the test set. Thresholds in the combined classifiers were adjusted and 100 data points were used to obtain this curve for each number of rounds.

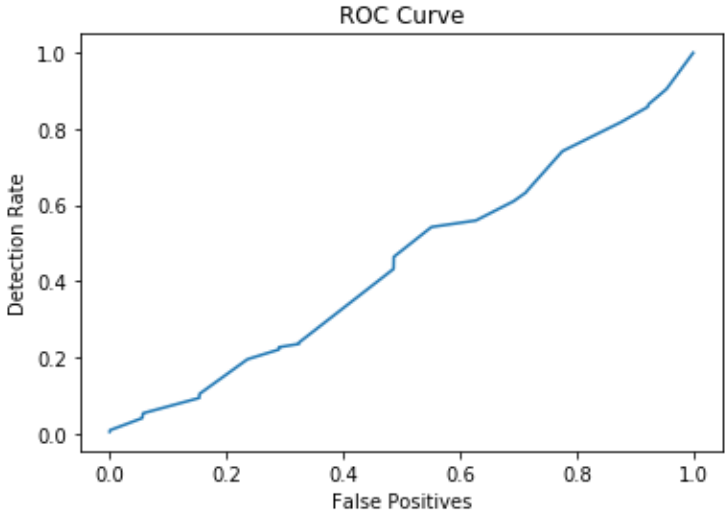
ROC curve for 1 round



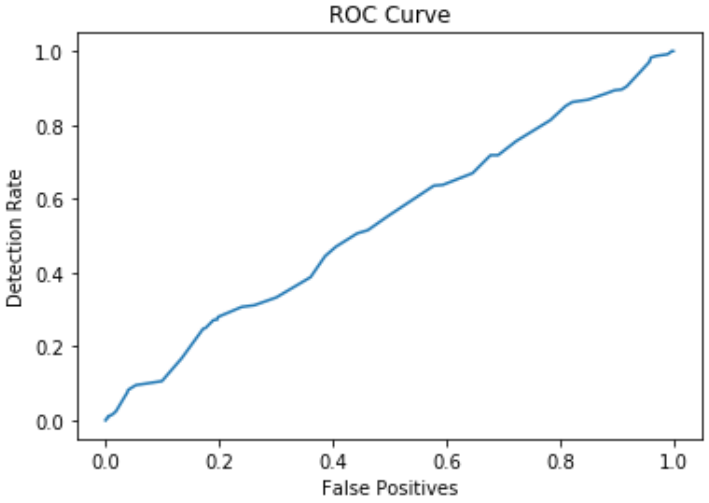
ROC curve for 3 rounds



ROC curve for 5 rounds



ROC curve for 10 rounds



Key Observations:

1. We can see that the detection rate increases with the false positive rate.
2. For low false positive rates, the detection rate is higher for higher number of boosting rounds.
3. The detection rate is better for the higher number of boosting rounds for all value of false positive rates.

Part4 (BONUS)

Cascading improves the performance of the AdaBoost predictor. Training with Adaboost is time-consuming as it is important to test a large number of features. Cascading the strong classifiers in such a way that a greater number of processed features are used by each classifier in the cascade than the previous one will lead to good results. In order to produce very low false negative rates, the strong classifier threshold may be lowered (no "face" should be labeled a "nonface"), allowing high false-positive rates (some "non-faces" can be labeled as "face"). The main principle is to reject as many apparent non-faces at an early stage as possible. The next classifier in the cascade and so on is activated by a go-ahead (positive result) from the first classifier. A negative outcome removes the argument immediately.

The cascade definition is motivated by the fact that there will be far more non-face windows in most pictures than face windows. Quickly discarding several non-faces saves a great deal of time during both preparation and testing. In this project, I have introduced a cascading architecture for face detection.

No. of layers in cascade network implemented = 4

No of boosting rounds in 4 layers = 2, 2, 2, 3

The performance of cascaded classifier along with images discarded at each layer are:

Training cascade network:

```
Layer 1: classifier with 2 rounds
Number of non-face images discarded:  1178
Layer 2: classifier with 2 rounds
Number of non-face images discarded:  455
Layer 3: classifier with 2 rounds
Number of non-face images discarded:  366
```

```
Layer 4: classifier with 3 rounds
Number of non-face images discarded:  1
```

Evaluation

1. Training data

```
Training accuracy: 0.80 (2000/2499)
False positives: 0.00 (2/2499)
False negatives: 0.20 (497/2499)
```

2. Testing data

```
Testing accuracy: 0.98 (19568/20044)
False positives: 0.00 (4/20044)
False negatives: 0.02 (472/20044)
```

Observations:

1. The performance increased considerably on test data.
2. Due to time and resource constraints, I had to limit the number of rounds of AdaBoost classifiers at each round. However, the performance can be further improved if we use higher number of boosting rounds for combined classifiers at each level. It would be very interesting to try this out in future experiments.

Source Code and Stored Data

The source code in the form of a Jupyter notebook has been zipped along with this project. The entire code is written in Python.

Note: The pickle files used to store the data at various checkpoints were too big to upload to Google Classroom, hence are not zipped along with. However, they can be found in my Github repository:

https://github.com/harshita-chaudhary/pr-project

Running Training and Testing

1. Download attached zips:
    a. 529005682.zip contains source code without pickle files
    b. vj-face-rec-pickel_files.zip contains pickle files that contained stored data to make verification faster

2. Install all the requirementsx
    a. Pillow
    b. Pickle
    c. Numpy
    d. Scikit-learn
    e. matplotlib
3. Install Jupyter to run the. ipynb file.
4. Copy the pickle files in vj-face-rec-pickel_files.zip that contain saved data for faster execution. Otherwise, the code creates those, which may take considerable time. This step can be skipped at the cost of extra running time.
5. Extract dataset.zip in the current location in the following structure
    a. Dataset/trainset/faces/
    b. Dataset/trainset/non-faces/
    c. Dataset/testset/faces/
    d. Dataset/testset/ non-faces/
6. Run the Jupyter notebook. For testing, only run the testing cells.

References

1. Viola and Jones, “Robust real-time object detection”, IJCV 2001
2. Tutorial by Zhou and Yu: “Adaboost”
3. Tutorial by Freund and Schapire: “A short introduction to boosting”
4. Medium post by Anmol Parande: https://medium.com/datadriveninvestor/understanding-and-implementing-the-viola-jones-image-classification-algorithm-85621f7fe20b