IMPLEMENTATION OF VIOLA-JONES ALGORITHM

DECEMBER 7, 2019

MAYURESH HOOLI

Contents

Introduction

Viola-Jones is most popular algorithm and has great success in face detection and is still used in digital cameras. In the Viola-Jones object detection algorithm, the training process uses AdaBoost to select a subset of features and construct the classifier.

This project, which is submitted as a part of Pattern Recognition course, is implementation of Viola-Jones algorithm for pattern recognition

Objectives of implementation were as follows

- Extract Haar Features
- Build Adaboost Detector
- Adjust the threshold

Optionally it was to build the cascading system

The project is implemented using Anaconda distribution of Python and R with Python 3.6

Implementation Method

For the implementation, several Python packages are used. Following are the major packages used for the implementation.

Please refer the readme document for implementation details.

Testing

Hardware Configuration

- CPU: Intel Core-i7 4750HQ
- Memory: 16GB DDR3 1600 MHz

Test Data

- Training Data:
    - Faces: 499
    - Non-Faces: 2000
- Test Data:
    - Faces: 472
    - Non-Faces: 2000

Adaboost Detector

- Number of samples for the faces and non-faces in the training data set: 499; 2000
- The pictures in the data set are 19×19. While the standard samples in Viola-Jones paper are 24×24, we adjust the window size based on the size of pictures we have
- We use the antialiasing on the image. This step may not be required as it doesn't add much value to the code, but was performed as a test
- We stitch the images together just to make a sense of this.

- We do the same process for the background

Training the Classifier

Extracting Haar Features

Before we can start the training process, we need to define the features in a way they can be evaluated. For this we're going to construct a class for each feature type and then parameterize it according to all possible values, yielding about 160k instances. The best features will be picked later by exhaustive search.

After instantiating the feature detector with all shapes of the feature, the number of features we obtain are as follows

1. Number of feature2h features: 17100
2. Number of feature2v features: 17100
3. Number of feature3h features: 10830
4. Number of feature3v features: 10830
5. Number of feature4 features: 8100
6. Total number of features: 63960

Sampling Data

Online sample data was used for the training. In order to compensate for lighting differences, we variance normalize the training data. For this we obtain a sample mean and standard deviation from the training set:

- Sample mean: 0.48178231716156006
- Standard deviation: 0.23022803664207458

Using normalization, we expect to see approximately zero mean and unit variance

- Example mean: 0.31774526834487915
- Standard deviation: 0.9429342746734619

Evaluation of the Classifier(s)

Image conversion to Integral Image. The average face looked as below using the dataset sample

Antialiased Image

Running the same for background, we get the following

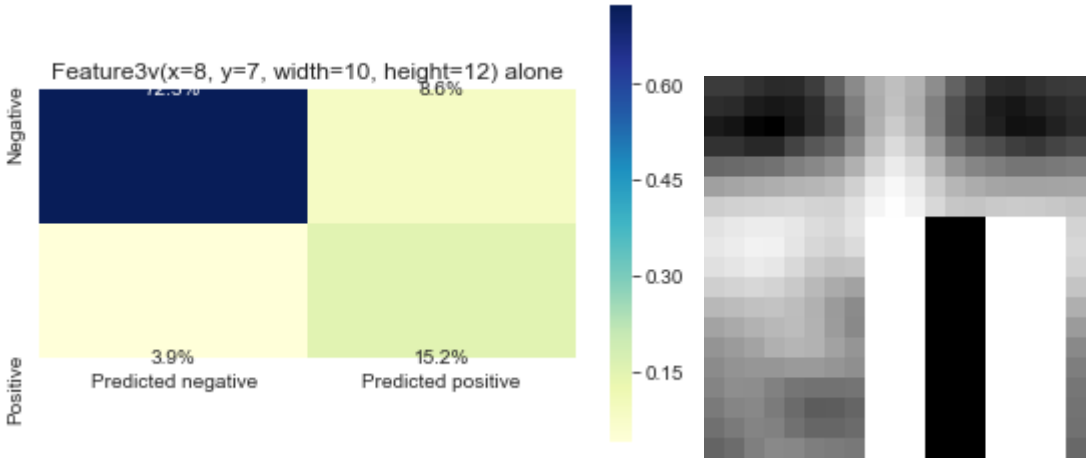
Weak Classifier Results

The weak classifier is run on 1, 3, 5 and 10 rounds and the accuracy has been identified. In addition to this, I have also drawn the haar features over the average faces to give an idea about how the features work.

Adaboost 1 Round

1. Feature 01:

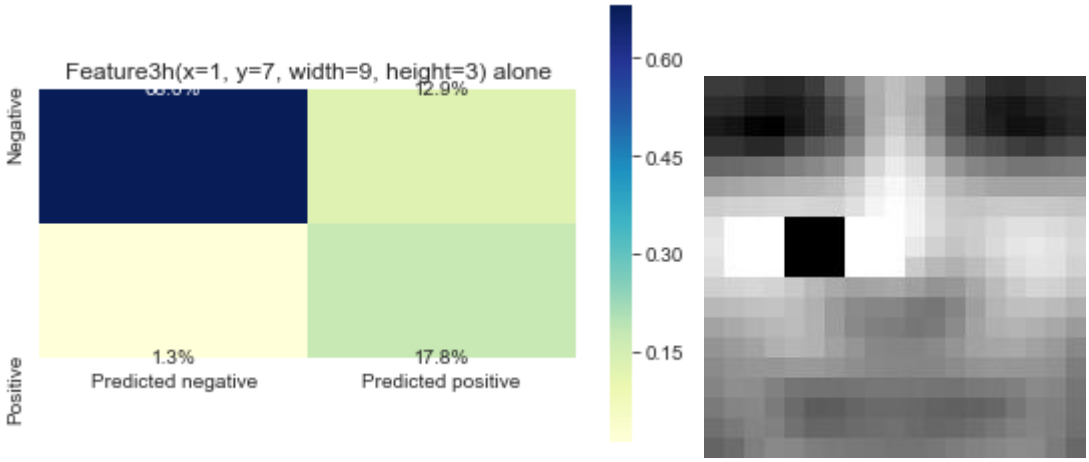
Type	Position	Width	Height	Threshold	Accuracy
3 vertical	(8, 7)	10	12	-18.90	0.87



Adaboost 3 Rounds

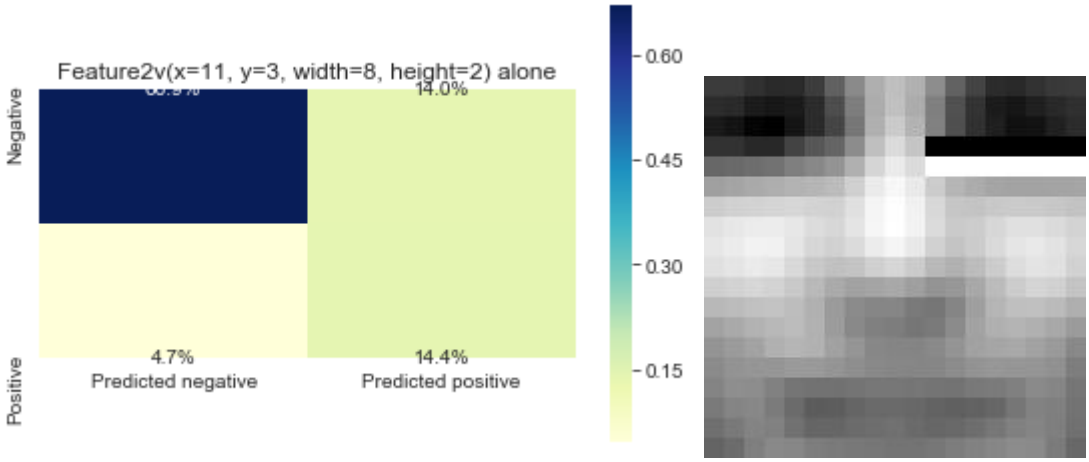
1. **Feature 01:**

Type	Position	Width	Height	Threshold	Accuracy
3 horizontal	(1, 7)	9	3	-2.86	0.86



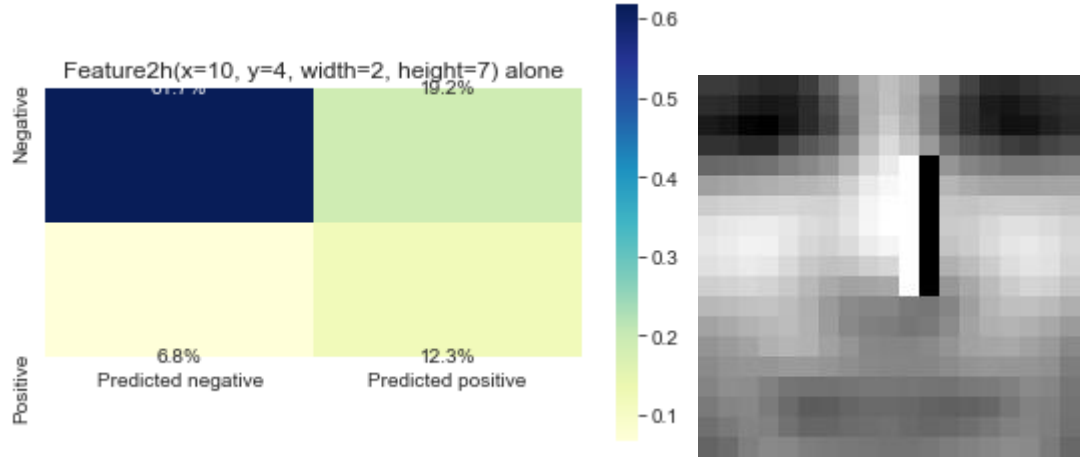
2. **Feature 02:**

Type	Position	Width	Height	Threshold	Accuracy
2 vertical	(11, 3)	8	2	0.79	0.81



3. Feature 03:

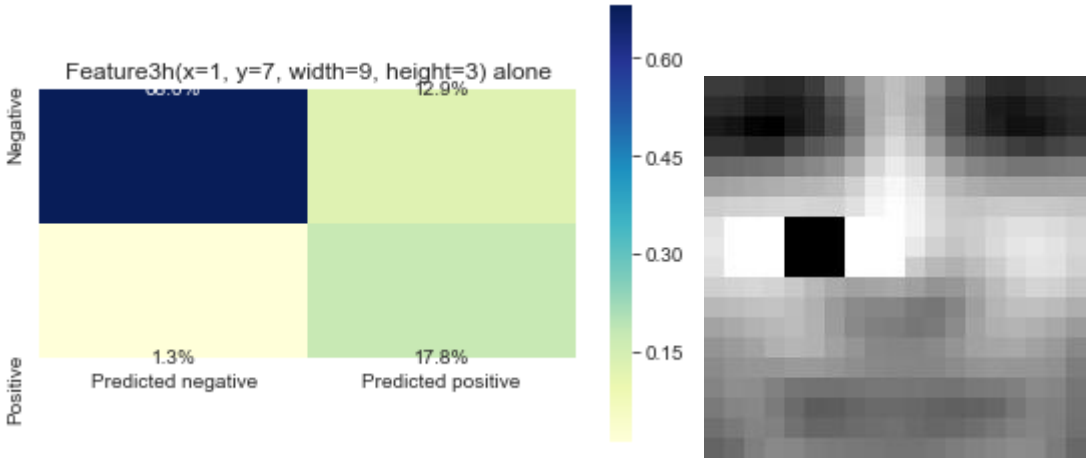
Type	Position	Width	Height	Threshold	Accuracy
2 horizontal	(10, 4)	2	7	0.26	0.74



Adaboost 5 Rounds

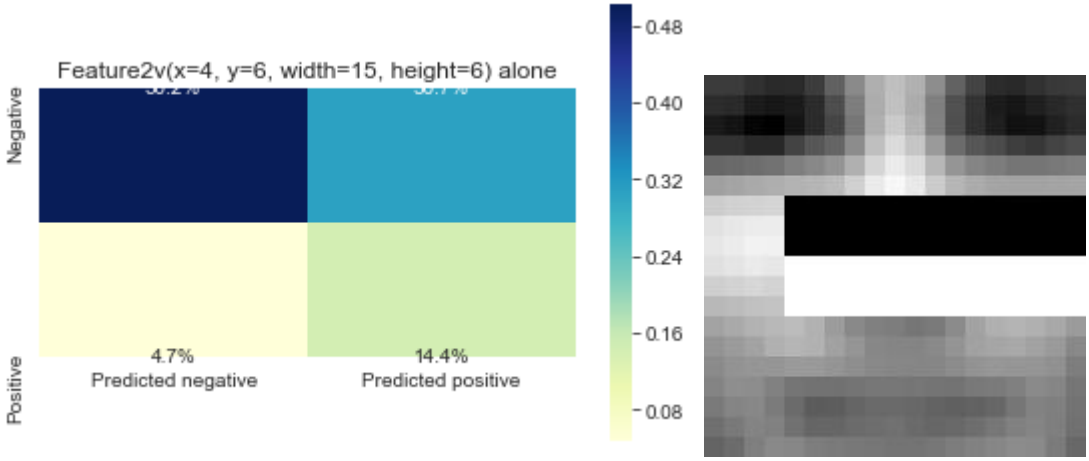
1. Feature 01:

Type	Position	Width	Height	Threshold	Accuracy
3 horizontal	(1, 7)	9	3	-2.86	0.86



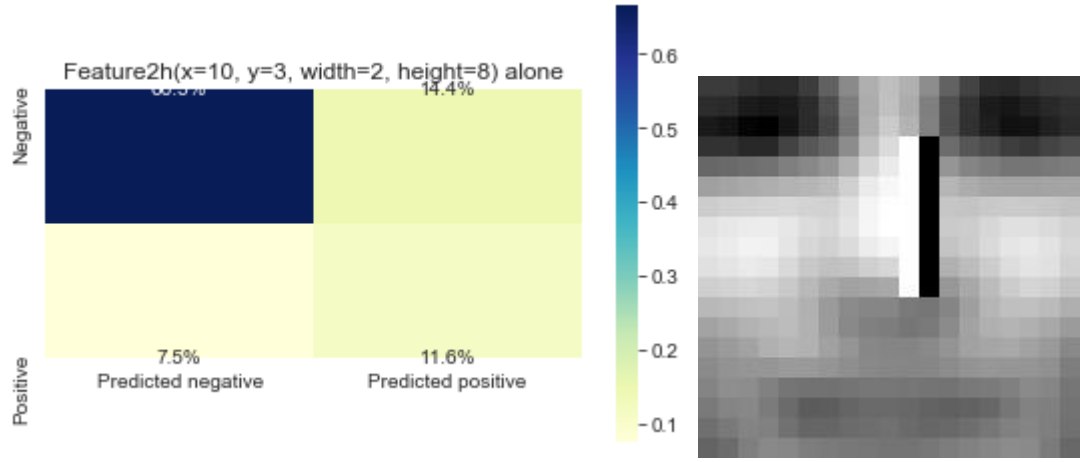
2. Feature 02:

Type	Position	Width	Height	Threshold	Accuracy
2 vertical	(4, 6)	15	6	-1.35	0.65



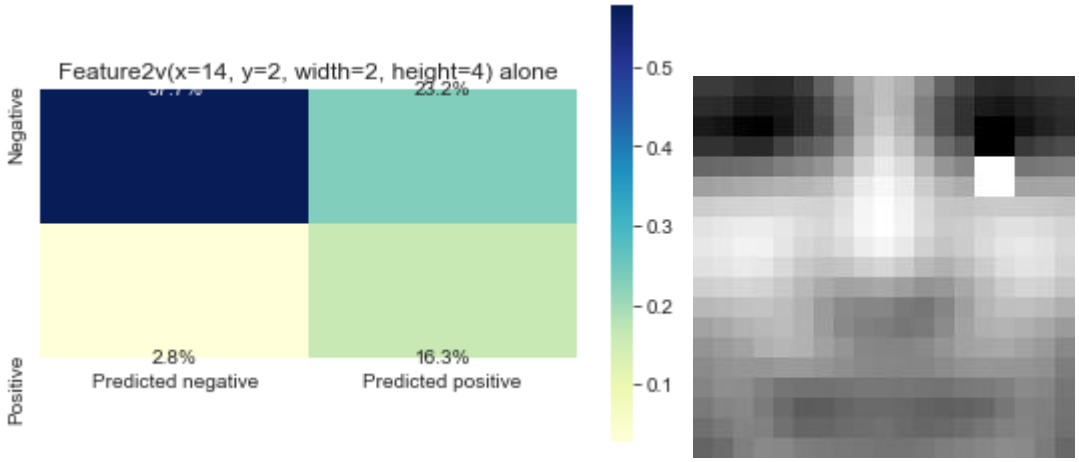
3. Feature 03:

Type	Position	Width	Height	Threshold	Accuracy
2 horizontal	(10, 3)	2	8	0.46	0.78



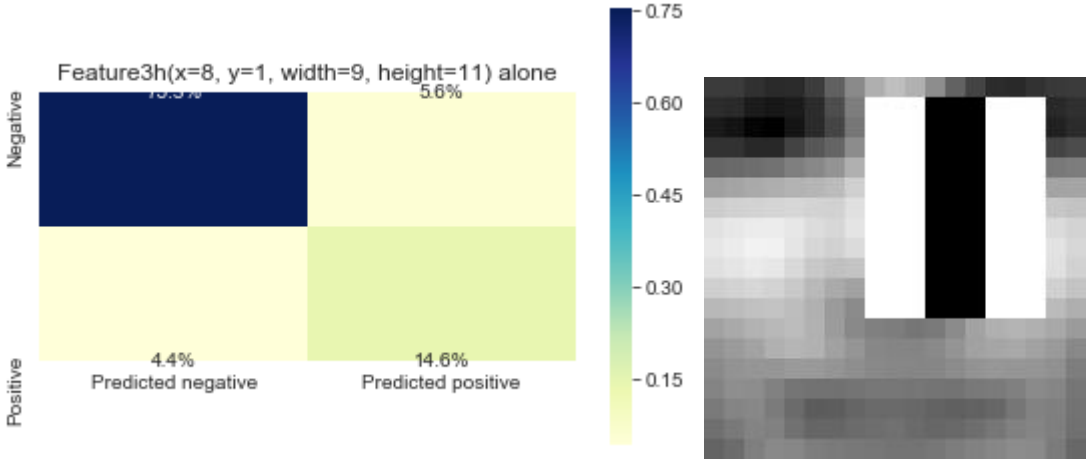
4. Feature 04:

Type	Position	Width	Height	Threshold	Accuracy
2 vertical	(14, 2)	2	4	0.36	0.74



5. Feature 05:

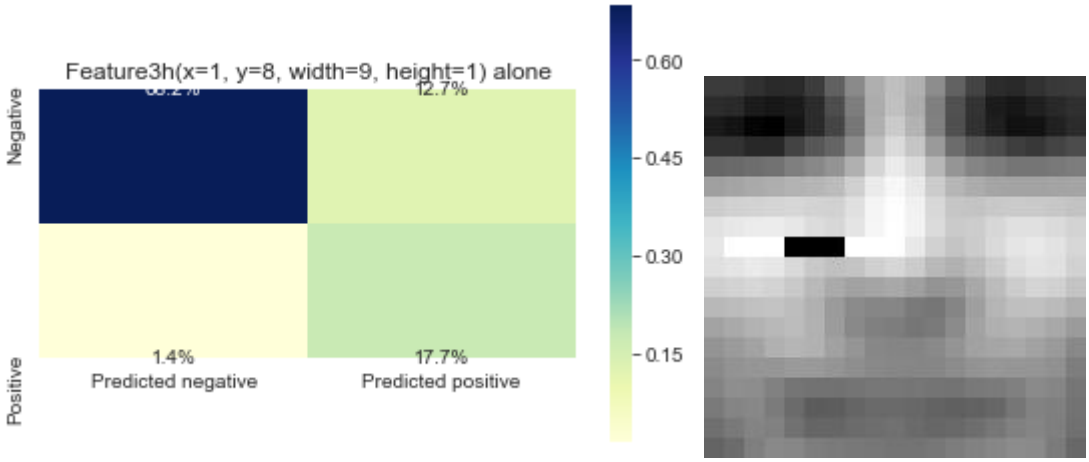
Type	Position	Width	Height	Threshold	Accuracy
3 horizontal	(8, 1)	9	11	-18.69	0.90



Adaboost 10 Rounds

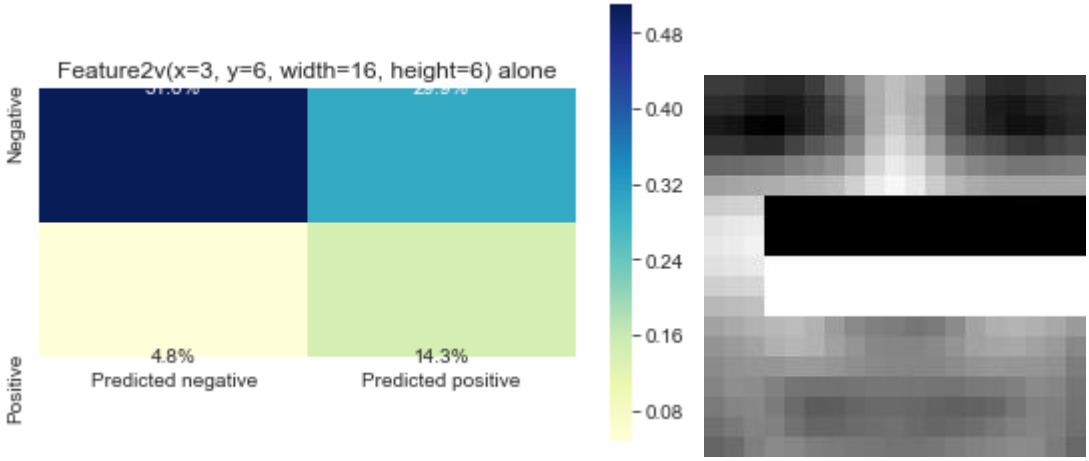
1. Feature 01:

Type	Position	Width	Height	Threshold	Accuracy
3 horizontal	(1, 8)	9	1	-1.08	0.86



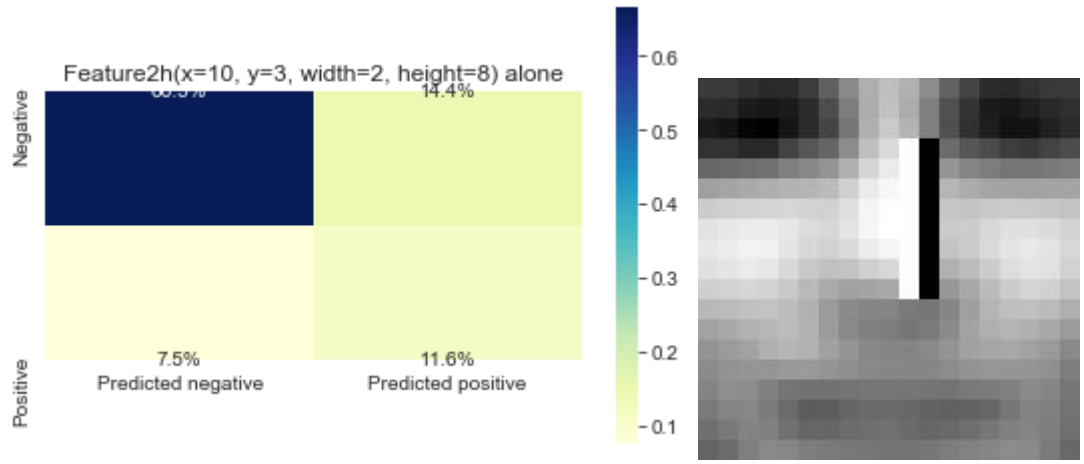
2. Feature 02:

Type	Position	Width	Height	Threshold	Accuracy
2 vertical	(3, 6)	16	6	-1.43	0.65



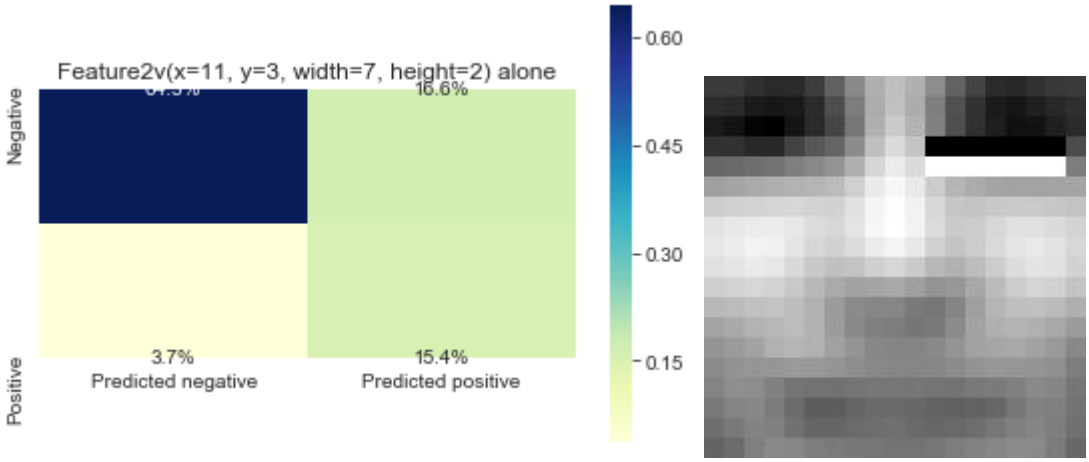
3. Feature 03:

Type	Position	Width	Height	Threshold	Accuracy
2 horizontal	(10, 3)	2	8	0.46	0.78



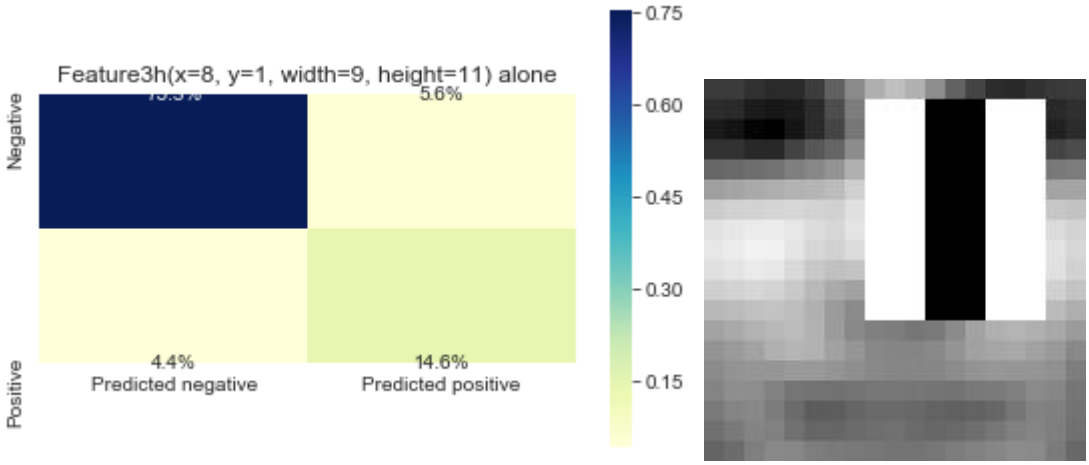
4. Feature 04:

Type	Position	Width	Height	Threshold	Accuracy
2 vertical	(11, 3)	7	2	0.49	0.80



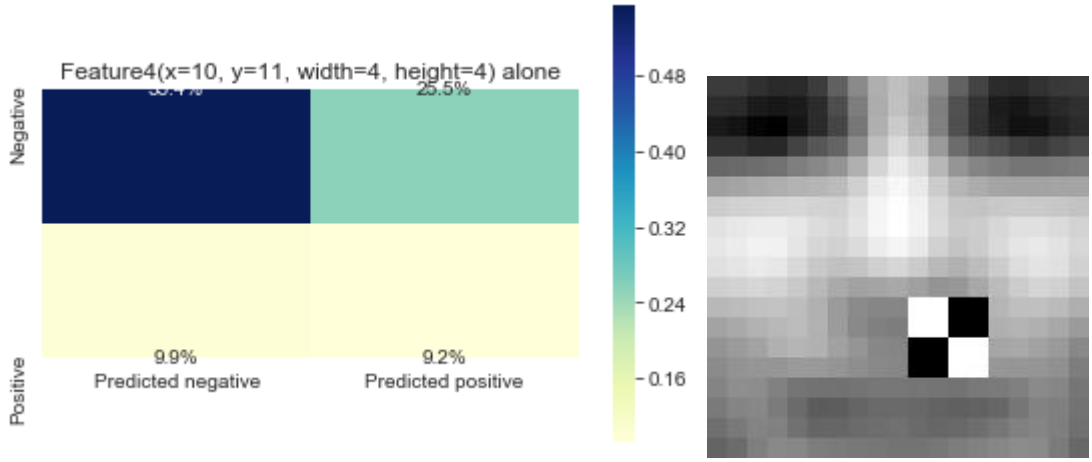
5. Feature 05:

Type	Position	Width	Height	Threshold	Accuracy
3 horizontal	(8, 1)	9	11	-18.69	0.90



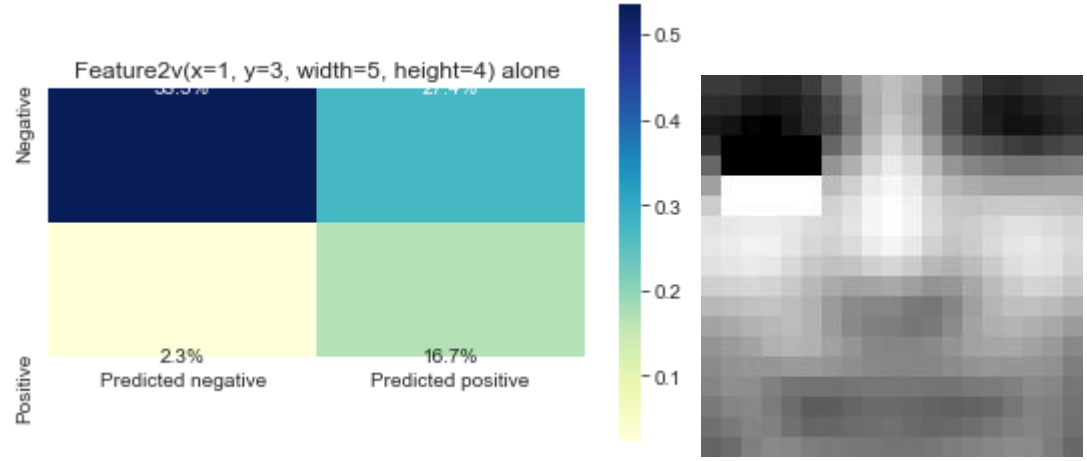
6. Feature 06:

Type	Position	Width	Height	Threshold	Accuracy
4	(10, 11)	4	4	-0.15	0.65



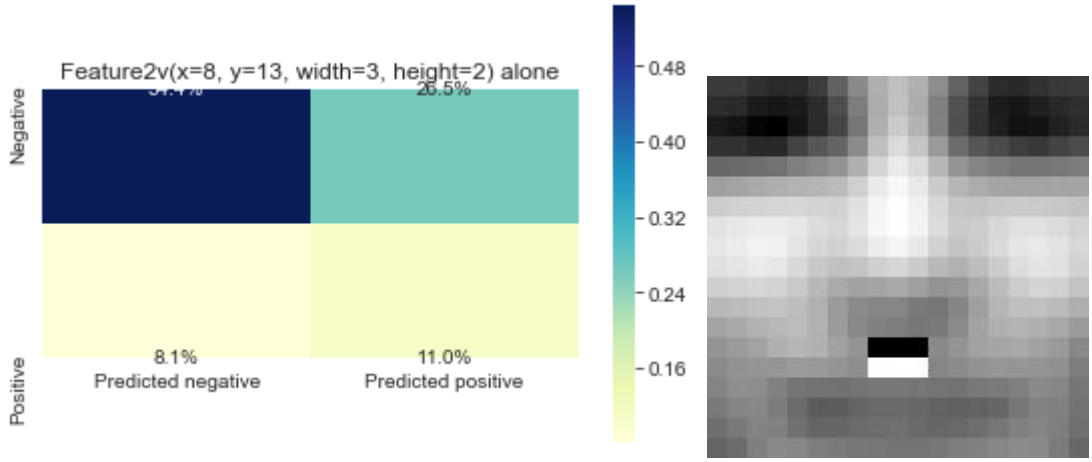
7. Feature 07:

Type	Position	Width	Height	Threshold	Accuracy
2 vertical	(1, 3)	5	4	1.23	0.70



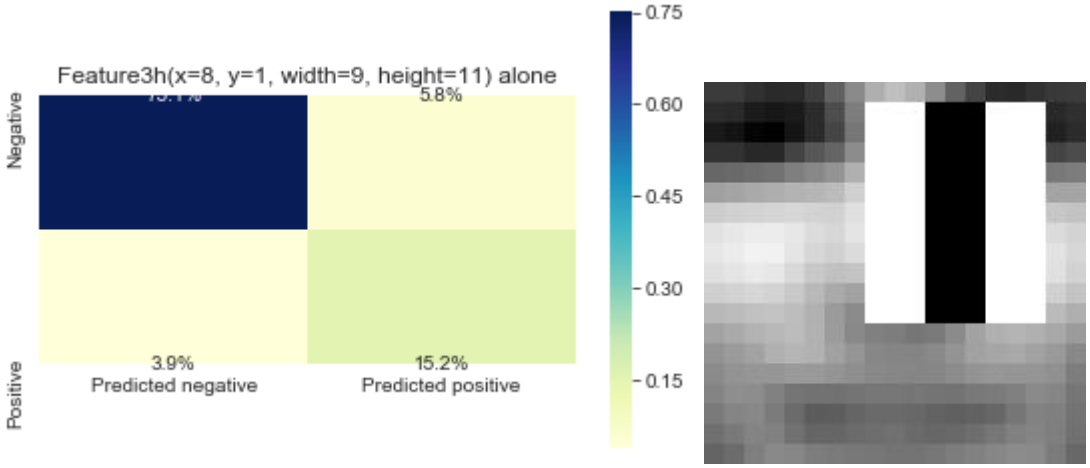
8. Feature 08:

Type	Position	Width	Height	Threshold	Accuracy
2 vertical	(8, 13)	3	2	0.03	0.65



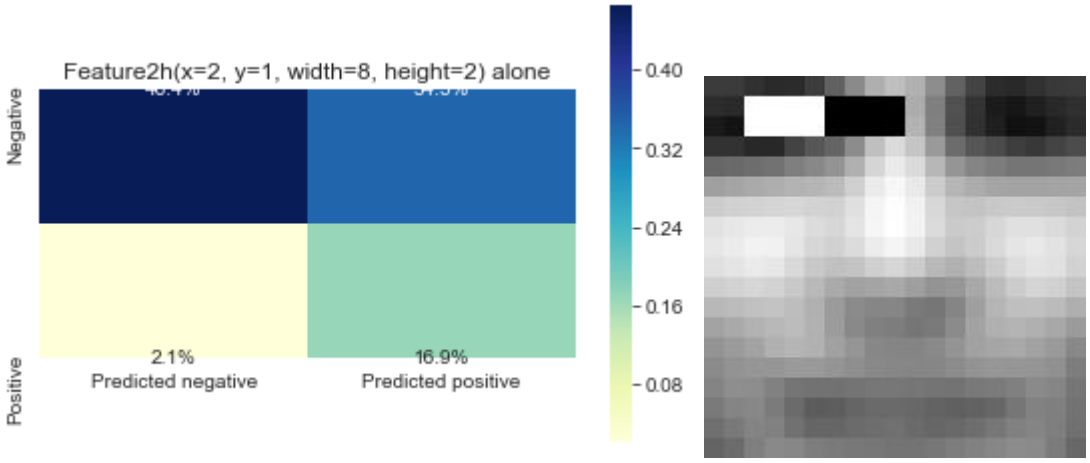
9. Feature 04:

Type	Position	Width	Height	Threshold	Accuracy
3 horizontal	(8, 1)	9	11	-17.95	0.90



10. Feature 05:

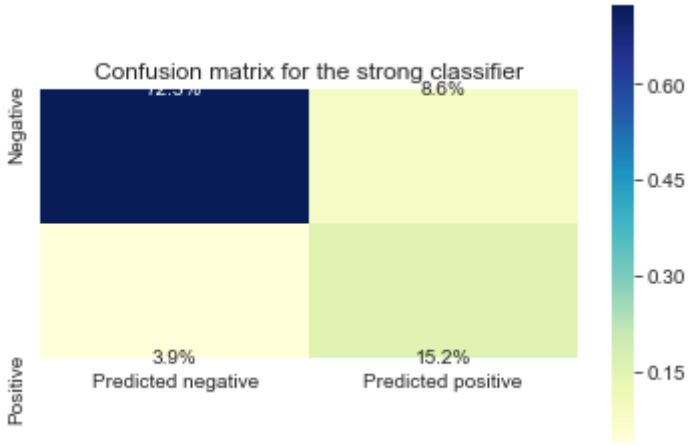
Type	Position	Width	Height	Threshold	Accuracy
2 horizontal	(2, 1)	8	2	-0.53	0.63



Strong Classifier Results

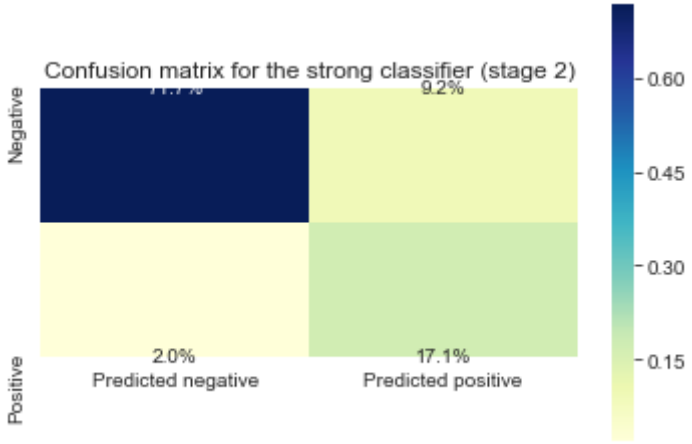
Adaboost 1 Round

	Type	Position	Width	Height	Threshold	Accuracy
Feature 01	3 vertical	(8, 7)	10	12	-18.90	0.87



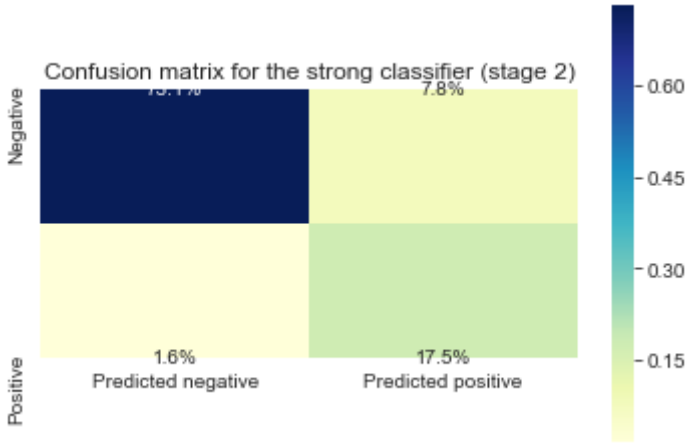
Adaboost 3 Rounds

	Type	Position	Width	Height	Threshold	Accuracy
Feature 01	3 horizontal	(1, 7)	9	3	-2.86	0.86
Feature 02	2 vertical	(11, 3)	8	2	0.79	0.81
Feature 03	2 horizontal	(10, 4)	2	7	0.26	0.74



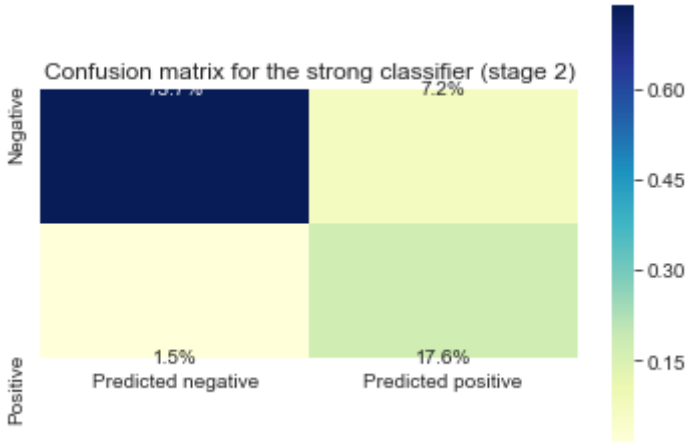
Adaboost 5 Rounds

	Type	Position	Width	Height	Threshold	Accuracy
Feature 01	3 horizontal	(1, 7)	9	3	-2.86	0.86
Feature 02	2 vertical	(4, 6)	15	6	-1.35	0.65
Feature 03	2 horizontal	(10, 3)	2	8	0.46	0.78
Feature 04	2 vertical	(14, 2)	2	4	0.36	0.74
Feature 05	3 horizontal	(8, 1)	9	11	-18.69	0.90



Adaboost 10 Rounds

	Type	Position	Width	Height	Threshold	Accuracy
Feature 01	3 horizontal	(1, 8)	9	1	-1.08	0.86
Feature 02	2 vertical	(3, 6)	16	6	-1.43	0.65
Feature 03	2 horizontal	(10, 3)	2	8	0.46	0.78
Feature 04	2 vertical	(11, 3)	7	2	0.49	0.80
Feature 05	3 horizontal	(8, 1)	9	11	-18.69	0.90
Feature 06	4	(10, 11)	4	4	-0.15	0.65
Feature 07	2 vertical	(1, 3)	5	4	1.23	0.70
Feature 08	2 vertical	(8, 13)	3	2	0.03	0.65
Feature 09	3 horizontal	(8, 1)	9	11	-17.95	0.90
Feature 10	2 horizontal	(2, 1)	8	2	-0.53	0.63



Overall

	Accuracy	Precision	Recall	False Positive	False Negative
Adaboost 1	0.87	0.64	0.79	0.11	0.21
Adaboost 3	0.89	0.65	0.89	0.11	0.11
Adaboost 5	0.91	0.69	0.92	0.10	0.08
Adaboost 10	0.91	0.71	0.92	0.09	0.08

Threshold Results

The threshold has been changed and run on a strong classifier for Adaboost 5. The threshold for sum of weights has been changed as discussed over the email.

Threshold	Accuracy	Precision	Recall	False Positive	False Negative
0.5	0.91	0.69	0.92	0.10	0.08
0.6	0.60	0.83	0.25	0.05	0.75
0.4	0.59	0.88	0.22	0.03	0.78

Cascading Results

The cascading results have been attempted by calculating Adaboost 20.

The Adaboost 20 has been compared with the cascades of Adaboost 1, 3, 5, 10 and 20 to give the following results with the Fusia image:

- Adaboost 20: Found 17164 candidates at stage 1.

- Cascade: Found 52853 candidates at stage 1, 13259 at stage 2, 10469 at stage 3, 8383 at stage 4 and 6300 at stage 5.