Published in Towards Data Science

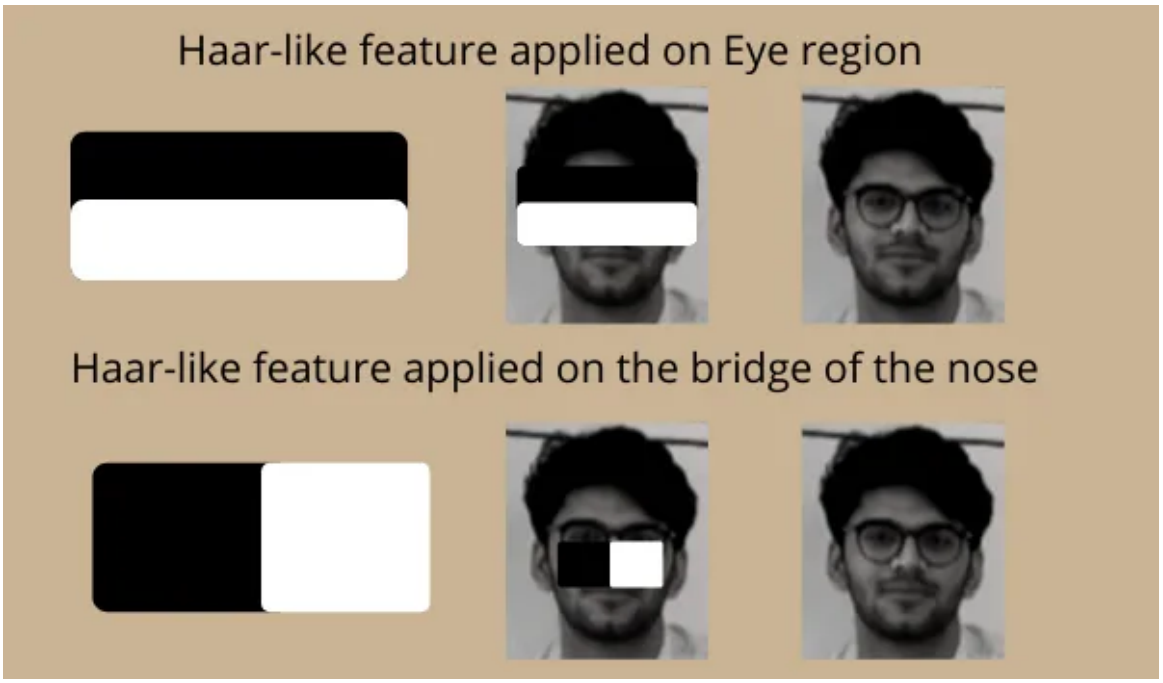
Mrinal Tyagi Follow

Jul 13, 2021 · 6 min read · Listen

Viola Jones Algorithm and Haar Cascade Classifier

Complete explanation and mathematics for beginners

Viola Jones is a novel approach to rapid detection of objects with running capabilities of 15 frames per second. It was the first to achieve real time object detection.



(Image by author)

In this article, I talk about “Viola Jones Algorithm” and it includes the following

subtopics :

1. Viola Jones Detector
2. What are Haar like features?
3. What is an integral image?
4. Calculation of Haar like features with Integral Image
5. Boosting and AdaBoost algorithm
6. Deep Dive into AdaBoost Algorithm mathematics
7. Cascade Filter
8. Implementation using OpenCV Library

Viola Jones Detector

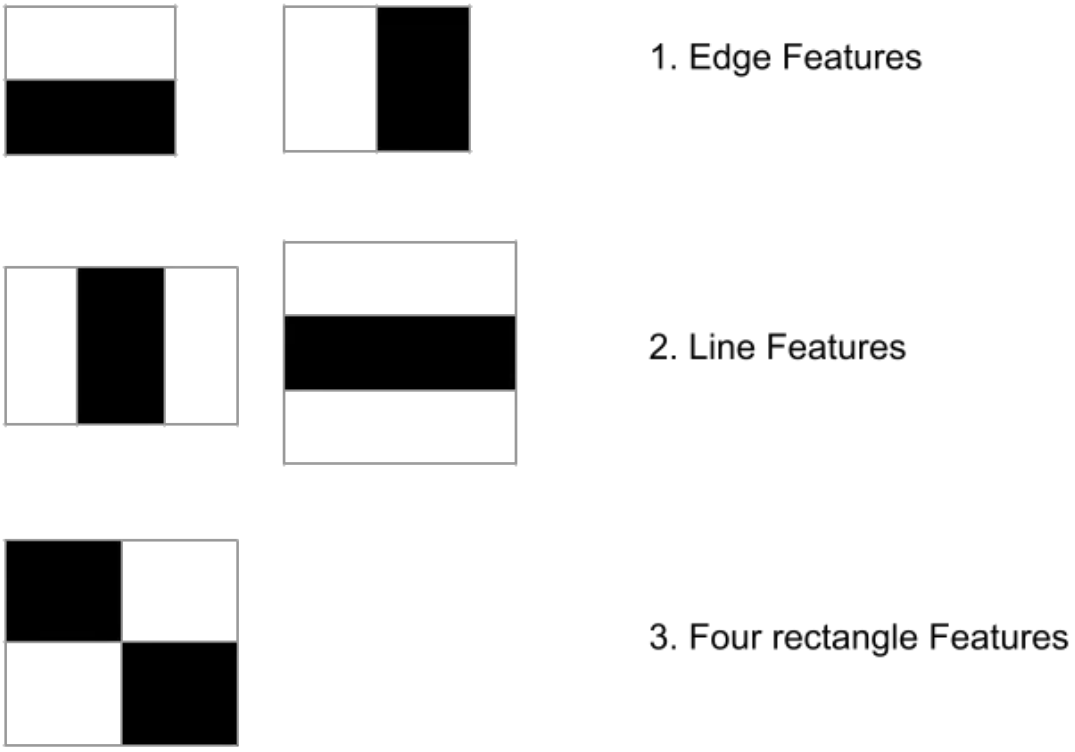
A Viola Jones detector consists of following steps :

1. Calculating Integral Image
2. Calculating Haar like features
3. AdaBoost Learning Algorithm
4. Cascade Filter

What are Haar like Features?

Haar features are the relevant features for face detection. It was proposed by Alfred Haar in 1909. They are like convolutional kernels. There are various types of haar like features but the most dominant features used are :

1. 2 Rectangular Haar features
2. 3 Rectangular Haar features
3. 4 Rectangular Haar features



(Image by author) Inspired (https://docs.opencv.org/3.4/d2/d99/tutorial_js_face_detection.html)

The values of a 2 rectangular feature is the difference between the sum of the pixels within 2 rectangular regions. The regions have same shape and size and are horizontally and vertically adjacent. A three rectangular feature computes the sum in a centre rectangle. Finally a four rectangular feature computes the difference between diagonal pairs of rectangles. Various variations of these regions of different sizes are convolved through the image in order to get multiple filters that will be inputs to the AdaBoost training algorithm. Calculation of these features using the standard technique would require a high computation time. In order to reduce this time, a new approach called the integral image was suggested by the authors of the paper.

What is an integral Image?

Because we have to use haar-like features in all possible sizes and locations which eventually result in around 200k features to calculate which is a really big number.

The problem with novel calculation of haar features is that we have to calculate the average of a given region multiple times and the time complexity of these operations are O(n*n). We can use an integral image approach to achieve O(1) running time. A given pixel in the integral image is the sum of all the pixels on the left and all the pixels above it.

1	2	2	4	1
3	4	1	5	2
2	3	3	2	4
4	1	5	4	6
6	3	2	1	3

Input Image

0	0	0	0	0	0
0	1	3	5	9	10
0	4	10	13	22	25
0	6	15	21	32	39
0	10	20	31	46	59
0	16	29	42	58	74

Integral Image

(Image by author) Inspired (https://www.mathworks.com/help/images/integral-image.html)

64	2	3	61	60	6	7	57
9	55	54	12	13	51	50	16
17	47	46	20	21	43	43	24
40	26	27	37	36	30	31	33
32	34	35	29	28	38	39	25
41	23	22	44	45	19	18	48
49	15	14	52	53	11	10	56
8	58	59	5	4	62	63	1

Original Image

64	66	69	130	190	196	203	260
73	130	187	260	333	390	446	520
90	194	297	390	484	584	683	780
130	260	390	520	650	780	910	1040
162	326	491	650	808	976	1145	1300
203	390	577	780	983	1170	1357	1560
252	454	655	910	1166	1364	1561	1820
260	520	780	1040	1300	1560	1820	2080

Integral Image

(Image by author) Inspired (https://m.blog.naver.com/natalliea/222198638897)

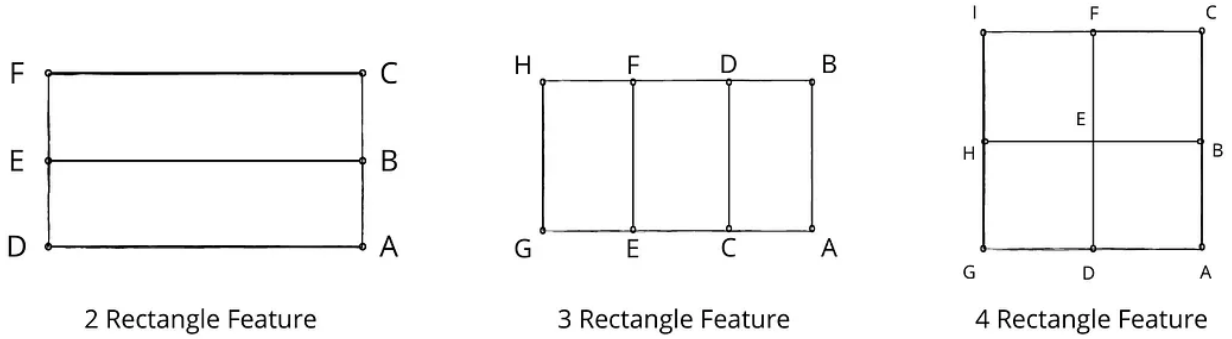
The sum of all purple boxes in the original image is equal to the sum of green boxes in the integral image subtracted by the purple boxes in the integral image.

Calculation of Haar like features with Integral Image

Using integral images we can achieve constant time evaluation of Haar features.

1. Edge Features or 2 Rectangular Features requires only 6 memory lookups
2. Line Features or 3 Rectangular Features requires only 8 memory lookups.
3. Diagonal Features or 4 Rectangular Features requires only 9 memory lookups.

> ***2 Rectangle*** = *A-2B+C-D+2E-F*
>
> ***3 Rectangle*** = *A-B-2C+2D+2E-2F-G+H*
>
> ***4 Rectangle*** = *A-2B+C-2D+4E-2F+H-2I+J*



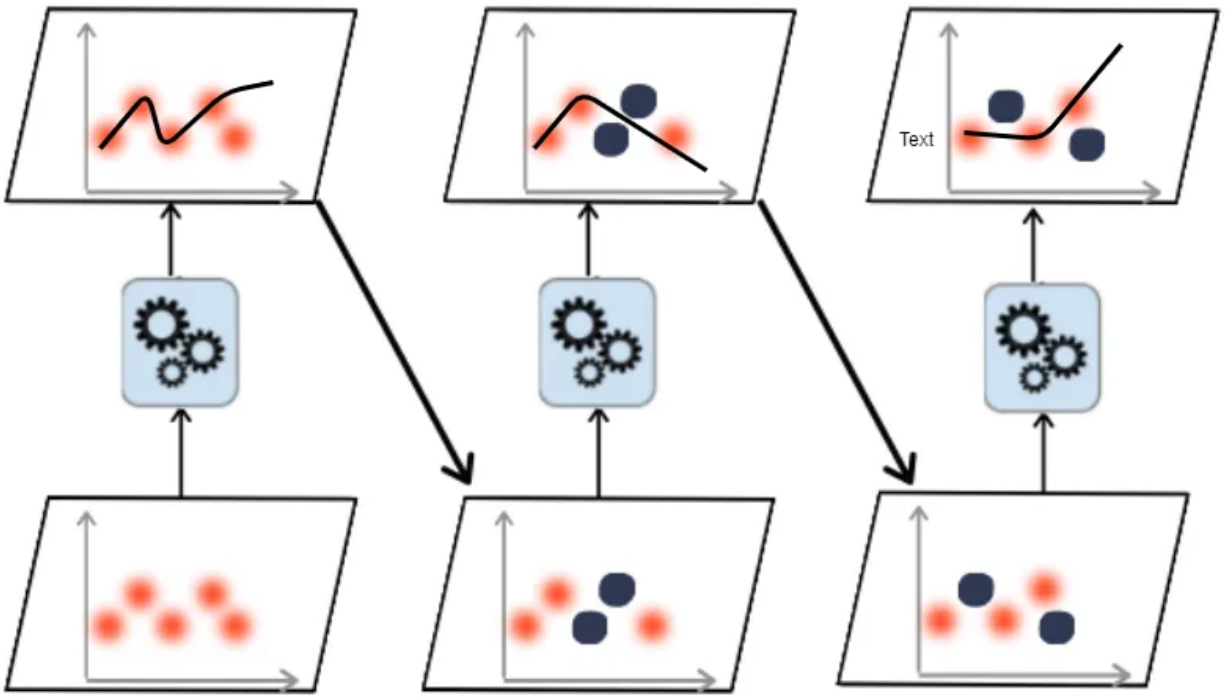
Technique for calculation of sum of regions for calculation of haar like features in a constant amount of time. (Image by author)

Boosting and AdaBoost Algorithm

Boosting refers to any Ensemble method that can combine several weak learners into a strong learner. The general idea of most boosting methods is to train predictors sequentially, each trying to correct its predecessor. AdaBoost also known as Adaptive Boosting is one of the most popular Boosting techniques used.

AdaBoost : One way for a new predictor to correct its predecessor is to pay a bit more

attention to the training instances that the predecessor under-fitted. This results in new predictors focusing more and more on the hard cases. This is known as Adaptive Boosting. For example, to build an Adaptive Boosting classifier, a first base classifier (such a Decision Tree or SVM classifier) is trained and used to make predictions on the training set. The relative weights of the misclassified predictions are altered and increased in order to lay more emphasis on these predictions while making the next predictor. A second classifier is trained using the updated weights and again it makes predictions on the training set, weights are updated and so on. Once all the predictions are trained, the ensemble method makes predictions very much like boosting except the predictors have different weights depending on their overall accuracy on the weighted training set. The drawback of this type of algorithm is that it cannot be parallelized thereby increasing time required. Thus after successfully running AdaBoost on all the features we are left with the most relevant features required for detection. Therefore, this reduces computational time as we don't have to go through all the features and is much more efficient.



(Image by author) Inspired (Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems)

Deep Dive into AdaBoost Algorithm

Let's take a closer look at the AdaBoost algorithm. Each instance weight w(i) is

initially set to 1/m. A first predictor is trained and its weighted error rate r1 is computed on the training set

$$r_j = \sum_{\substack{i=1 \\ \hat{y}_j^{(i)} \neq y^{(i)}}}^{m} w^{(i)}$$

Here we take only the misclassified instances and sum up the weights of those instances to get the weighted error rate (Image by author)

The predictor's weight j is then computed using the formulae given below. The more accurate the predictor is, the higher its weight will be. If it is just guessing randomly, then its weight will be close to zero. However, most often, it is wrong and its weight will be negative.

$$\alpha_j = \eta \log \frac{1 - r_j}{r_j}$$

(Image by author)

Next the instance weights are updated using the formula provided below in order to boost the misclassified instances

$$w^{(i)} \leftarrow \begin{cases} w^{(i)} & if\ \hat{y}_j^{(i)} = y^{(i)} \\ w^{(i)} \exp(\alpha_j) & if\ \hat{y}_j^{(i)} \neq y^{(i)} \end{cases}$$

(Image by author)

Then all the predictors are normalized using the formulae provided below.

$$w^{(i)} = \frac{w^{(i)}}{\sum_{i=1}^{m} w^{(i)}}$$

(Image by author)

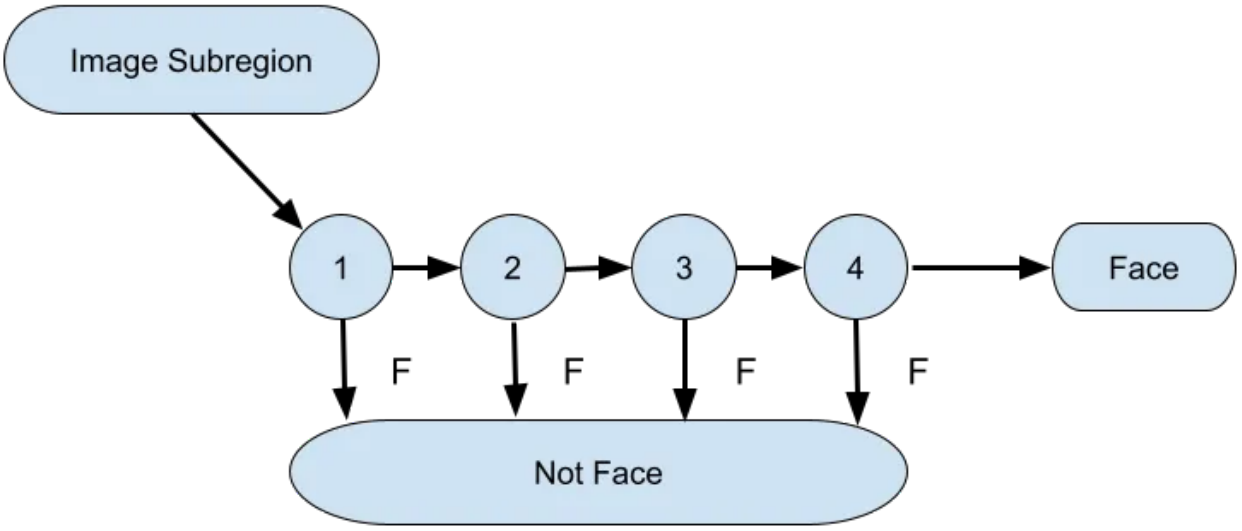
Finally, a new predictor is trained using the updated weights, and the whole process

is repeated until the desired number of predictors is reached which is specified by the user.

During inference, AdaBoost simply computed the predictions of all the predictors and weights then using the predictor weight αj. The predicted class is the one that receives the majority of weighted votes.

Cascade Filter

- Strong features are formed into a binary classifier. : Positive matches are sent along to the next feature. Negative matches are rejected and exit computation.
- Reduces the amount of computation time spent on false windows.
- Threshold values might be adjusted to tune accuracy. Lower threshold yield higher detection rated and more false positives.



(Image by author) Inspired (https://www.researchgate.net/figure/Cascade-classifier-illustration-2_fig2_323057610)

In simple terms, each feature acts as a binary classifier in a cascade filter. If an extracted feature from the image is passed through the classifier and it predicts that the image consists of that feature then it is passed on to the next classifier for next feature existence check otherwise it is discarded and next image is checked. This thereby decreases computation time as we have to check only some features in

windows where the object is not present rather than checking all features. This is the main part of the algorithm that allows it to process videos at a rate of approximately 15 frames per second and enables real time implementation.

Implementation using OpenCV Library

References

1. Rapid Object Detection using a Boosted Cascade of Simple Features : https://web.iitd.ac.in/~sumeet/viola-cvpr-01.pdf
2. Detecting Faces (Viola Jones Algorithm) — Computerphile : https://www.youtube.com/watch?v=uEJ71VlUmMQ
3. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow:

Concepts, Tools, and Techniques to Build Intelligent Systems by Aurelien Geron

Stay tuned for new research papers' explanation like this!

Feel free to connect and give ur suggestions : https://www.linkedin.com/in/mrinal-tyagi-02a1351b1/

Machine Learning Computer Vision Classification Mathematics

Artificial Intelligence