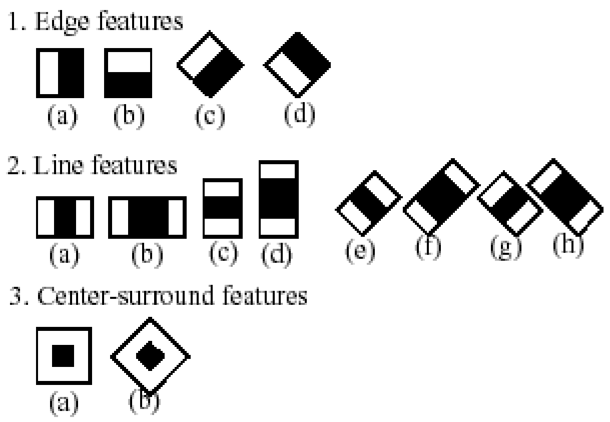
Haar Cascades for Object Detection

- Haar Cascade classifier is based on the Haar Wavelet technique to analyze pixels in the image into squares by function. This uses “integral image” concepts to compute the “features” detected.
- Haar Cascades uses the Ada-boost learning algorithm which selects a small number of important features from a large set to give an efficient result of classifiers then use cascading techniques to detect the face in an image.
- Object Detection is a computer technology related to computer vision, image processing and deep learning that deals with detecting instances of objects in images and videos. We will do object detection in this article using something known as haar cascades.
- This framework can be trained to detect almost any object, but this primarily solves the problem of face detection in real-time. This algorithm has four steps.

```
1. Haar Feature Selection
2. Integral Image Representation
3. Adaboost Training
4. Cascade Classifier Architecture
```

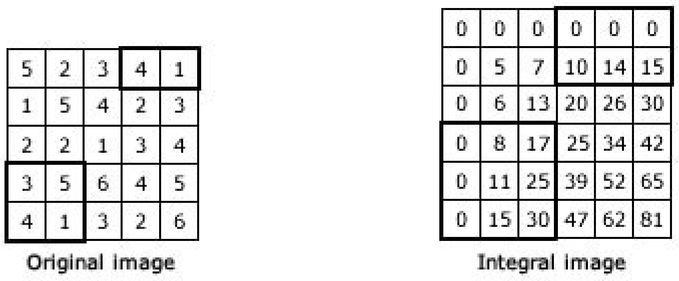
1. Haar Feature Selection

- Objects are classified on very simple features as a feature to encode ad-hoc domain knowledge and operate much faster than pixel system.
- The feature is similar to haar filters, hence the name ‘Haar’.
- An example of these features is a 2-rectangle feature, defined as the difference of the sum of pixels of area inside the rectangle, which can be any position and scale within the original image. 3-rectangle and 4-rectangle features are also used here.



2. Integral Image Representation

- The Value of any point in an Integral Image, is the sum of all the pixels above and left of that point. An Integral Image can be calculated efficiently in one pass over the image.



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

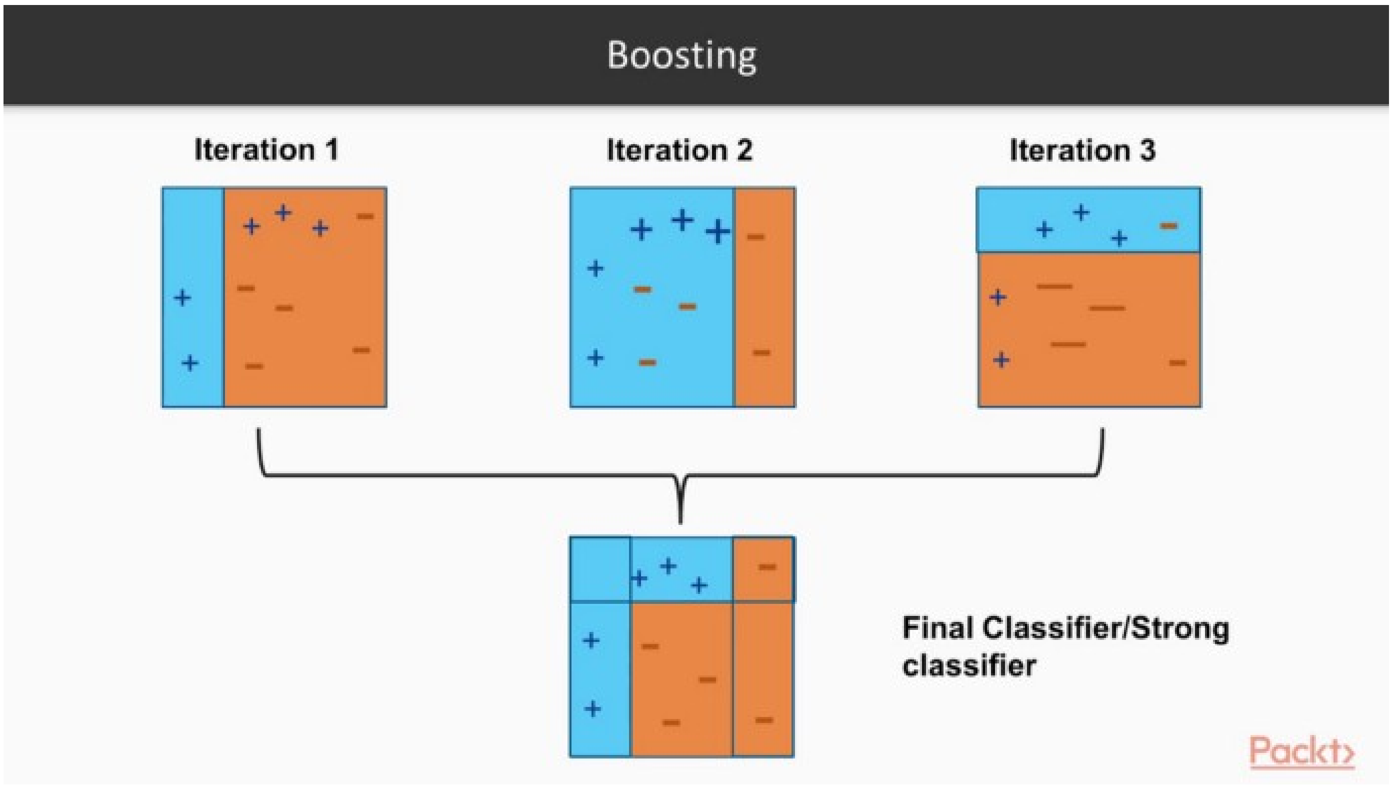
Original image

0	0	0	0	0	0
0	5	7	10	14	15
0	6	13	20	26	30
0	8	17	25	34	42
0	11	25	39	52	65
0	15	30	47	62	81

Integral image

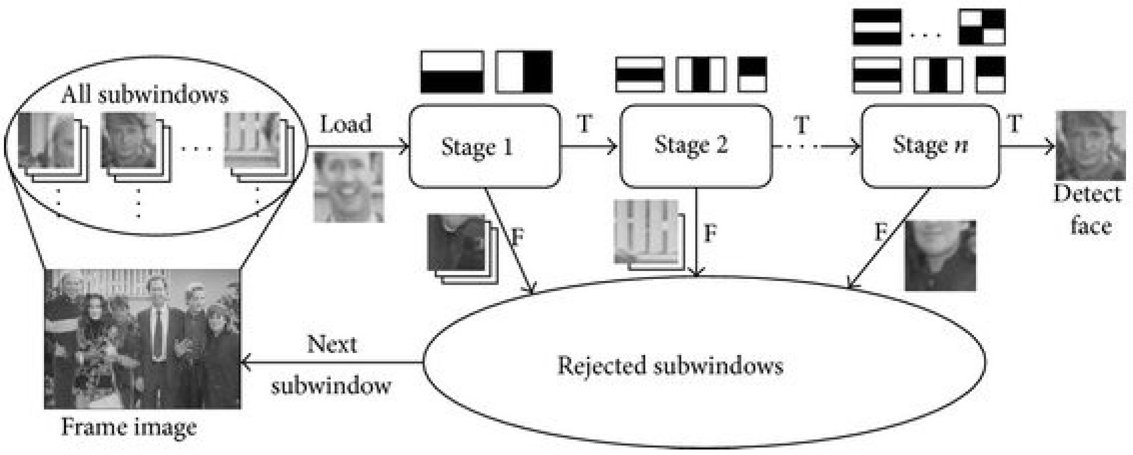
3. Adaboost Training

- For a window of 24x24 pixels, there can be about 162,336 possible features that would be very expensive to evaluate. Hence AdaBoost algorithm is used to train the classifier with only the best features.



4. Cascade Classifier Architecture

- A cascade classifier refers to the concatenation of several classifiers arranged in successive order. It makes large numbers of small decisions as to whether its the object or not. The structure of the cascade classifier is of a degenerate decision tree.



What are Haar Cascades?

- Haar Cascade classifiers are an effective way for object detection. This method was proposed by Paul Viola and Michael Jones in their paper Rapid Object Detection using a Boosted Cascade of Simple Features .Haar Cascade is a machine learning-based approach where a lot of positive and negative images are used to train the classifier.

```
    Positive images – These images contain the images which we want our classifier
to identify.
    Negative Images – Images of everything else, which do not contain the object w
e want to detect.
```

Requirements:

Make sure you have python, Matplotlib and OpenCV installed on your pc (all the latest versions). The haar cascade files can be downloaded from the OpenCV Github repository. All cacscade:- https://github.com/opencv/opencv/tree/master/data/haarcascades (https://github.com/opencv/opencv/tree/master/data/haarcascades)

Implementation

In [1]:

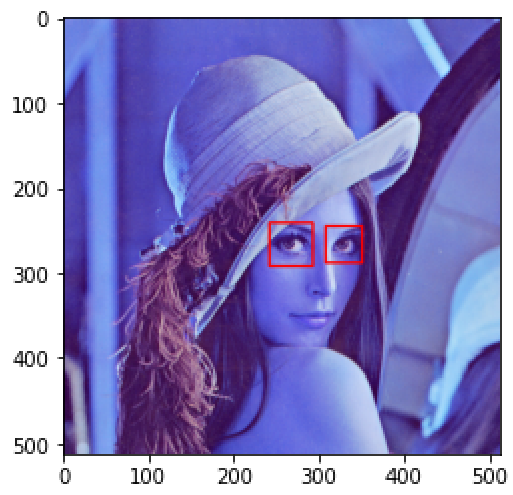
```
# Importing all required packages
import cv2
import numpy as np
import matplotlib.pyplot as plt


# Read in the cascade classifiers for face and eyes
face_cascade = cv2.CascadeClassifier('haarcascades/haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascades/haarcascade_eye.xml')
```

In [2]:

```
img = cv2.imread("lena.png")

# imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

eye = eye_cascade.detectMultiScale(img, 1.1, 4)

for (x,y,w,h) in eye:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

# cv2.imshow("Original",img)

# cv2.waitKey(0)

plt.imshow(img)
```

Out[2]:

```
<matplotlib.image.AxesImage at 0x2a0fcf2d108>
```



In [3]:

```
img = cv2.imread("lena.png")

# imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(img, 1.1, 4)

for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

# cv2.imshow("Original",img)

# cv2.waitKey(0)

plt.imshow(img)
```

Out[3]:

```
<matplotlib.image.AxesImage at 0x2a0ff281cc8>
```