Haar Cascades

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features” in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

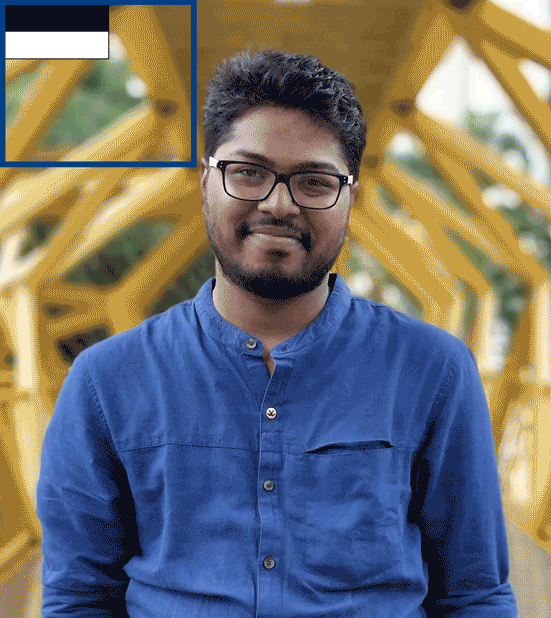
Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, haar features shown in below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

The haar feature continuously traverses from the top left of the image to the bottom right to search for the particular feature. This is just a representation of the whole concept of the haar feature traversal. In its actual work, the haar feature would traverse pixel by pixel in the image. Also all possible sizes of the haar features will be applied.



Depending on the feature each one is looking for, these are broadly classified into three categories. The first set of **two rectangle features** are responsible for finding out the edges in a horizontal or in a vertical direction (as shown above). The second set of **three rectangle features** are responsible for finding out if there is a lighter region surrounded by darker regions on either side or vice-versa. The third set of **four rectangle features** are responsible for finding out change of pixel intensities across diagonals.

A majority of these features won’t work well or will be irrelevant to the facial features, as they will be too random to find anything. So here they needed a Feature Selection technique, to select a subset of features from the huge set which would not only select features performing better than the others, but also will eliminate the irrelevant ones. They used a **Boosting Technique** called[**AdaBoost**](https://blog.paperspace.com/adaboost-optimizer/#:~:text=AdaBoost%20is%20an%20ensemble%20learning,turn%20them%20into%20strong%20ones.), in which each of these 180,000 features were applied to the images separately to create **Weak Learners**. Some of them produced low error rates as they separated the Positive images from the Negative images better than the others, while some didn’t. These weak learners are designed in such a way that they would misclassify only a minimum number of images. They can perform better than only a random guess. With this technique, their final set of features got reduced to a total of 6000 of them.



This is how the detection of features takes place in stages. You can notice that, when the window is at a non-face region, only the first stage with two rectangle features are running, and as they discard the window before the second stage starts. Only one window which actually contains a face, runs both the stages and detects the face.

Face detection using Cascade Classifier using OpenCV-Python

One of the popular algorithms for facial detection is “haarcascade”. It is computationally less expensive, a fast algorithm, and gives high accuracy.

It works in four stages:

1. Haar-feature selection: A Haar-like feature consists of dark regions and light regions. It produces a single value by taking the difference of the sum of the intensities of the dark regions and the sum of the intensities of light regions. It is done to extract useful elements necessary for identifying an object.
2. Creation of Integral Images: A given pixel in the integral image is the sum of all the pixels on the left and all the pixels above it. Since the process of extracting Haar-like features involves calculating the difference of dark and light rectangular regions, the introduction of Integral Images reduces the time needed to complete this task significantly.
3. AdaBoost Training: This algorithm selects the best features from all features. It combines multiple “weak classifiers” (best features) into one “strong classifier”. The generated “strong classifier” is basically the linear combination of all “weak classifiers”.
4. Cascade Classifier: It is a method for combining increasingly more complex classifiers like AdaBoost in a cascade which allows negative input (non-face) to be quickly discarded while spending more computation on promising or positive face-like regions. It significantly reduces the computation time and makes the process more efficient.

**Stepwise Implementation:**

**Step 1: Loading the image**



**Step 2: Converting the image to grayscale**

Initially, the image is a three-layer image (i.e., RGB), So It is converted to a one-layer image (i.e., grayscale).

A black screen with white text

Description automatically generated

**Step 3: Loading the required haar-cascade XML classifier file**

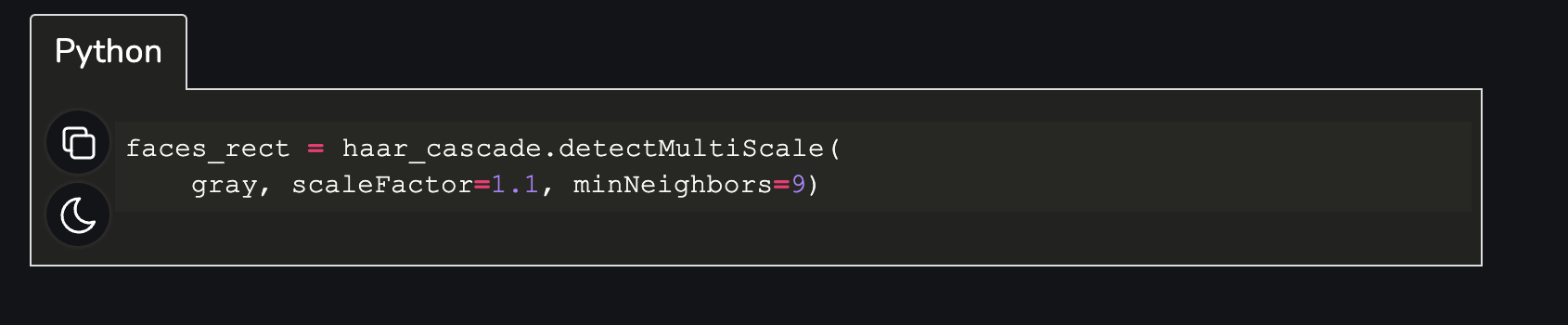
CascadeClassifier method in cv2 module supports the loading of haar-cascade XML files. Here, we need “haarcascade\_frontalface\_default.xml” for face detection.

A black rectangular with yellow and green text

Description automatically generated

**Step 4: Applying the face detection method on the grayscale image**

This is done using the cv2::CascadeClassifier::detectMultiScale method, which returns boundary rectangles for the detected faces (i.e., x, y, w, h). It takes two parameters namely, scaleFactor and minNeighbors. ScaleFactor determines the factor of increase in window size which initially starts at size “minSize”, and after testing all windows of that size, the window is scaled up by the “scaleFactor”, and the window size goes up to “maxSize”. If the “scaleFactor” is large, (e.g., 2.0), there will be fewer steps, so detection will be faster, but we may miss objects whose size is between two tested scales. (default scale factor is 1.3). Higher the values of the “minNeighbors”, less will be the number of false positives, and less error will be in terms of false detection of faces. However, there is a chance of missing some unclear face traces as well.



**Step 5: Iterating through rectangles of detected faces**

Rectangles are drawn around the detected faces by the rectangle method of the cv2 module by iterating over all detected faces.

A screenshot of a computer program

Description automatically generated

**Accuracy calculation**

**True positive (TP): It is an actual object of interest that is correctly identified. The correctly classified faces can be calculated as**

**True positives rate (TPR) = TP/(TP+FP)**

**False-positives (FP): It is a non-object of interest which is falsely identified as the true object.**

**False-negatives (FN): It is an actual object of interest falsely identified as negative.**

[**False negatives**](https://www.sciencedirect.com/topics/engineering/false-negative)**rate**

**(FNR)=FN/(FN+TP)**

**Accuracy = (TP + TN) / (TP + TN + FP + FN)**

**Where,**

**TP: True Positive FP: False Positive**

**TN: True Negative FN: False Negative**

**The Accuracy is obtained for the Haar cascade is 96.24% and for LBP classifier 94.74%.**