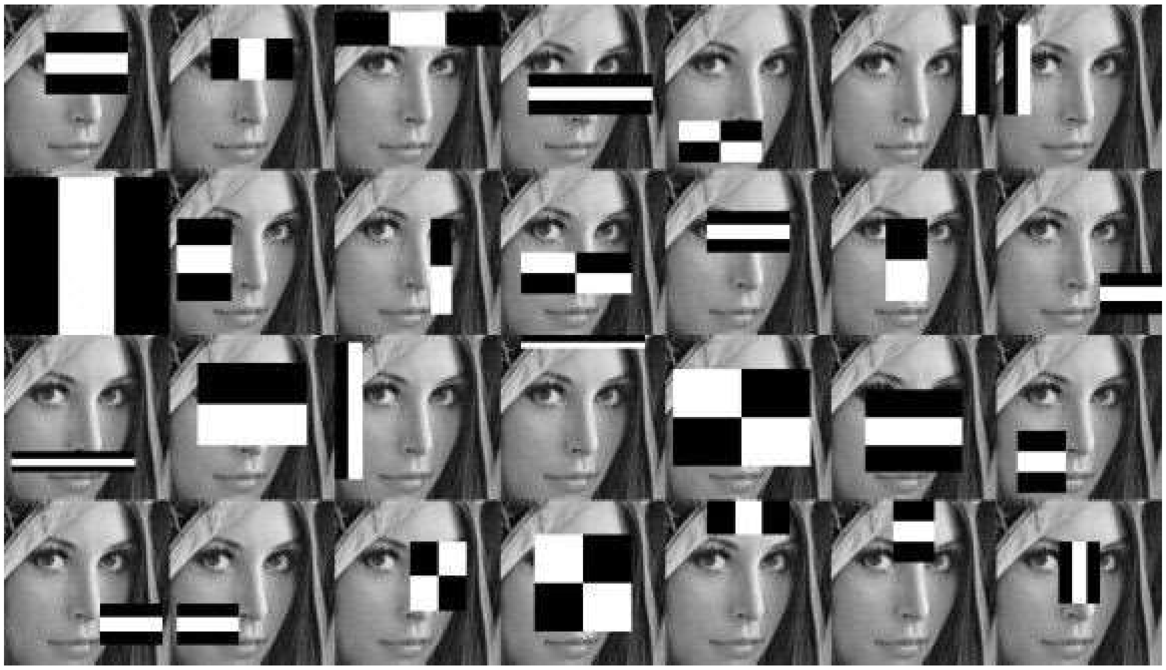
Haar Cascades, Explained

Aditya Mittal Follow
Dec 20, 2020 · 6 min read


A general representation of training a Haar classifer. (Image Source)

You see facial recognition **everywhere**, from the security camera in the front porch of your house to your sensor on your iPhone X. But how exactly does facial recognition work to classify faces, considering the large number of features as input and the striking similarities between humans?

Facial recognition on an iPhone X. (Image Source)

Enter **Haar classifiers,** classifiers that were used in the first real-time face detector. A Haar classifier, or a Haar cascade classifier, is a machine learning object detection program that identifies objects in an image and video.

A detailed description of Haar classifiers can be seen in Paul Viola and Michael Jones's paper "Rapid Object Detection using a Boosted Cascade of Simple Features", linked over here. Note that the article goes into some mathematics, and assumes knowledge of machine learning terminology. If you want a summarized, high-level overview, make sure to keep reading!

Making a Haar Cascade Classifier

Note: This discussion will assume basic knowledge of boosting algorithms and weak vs. strong learners with regards to machine learning. Click here for a quick Adaboost tutorial.

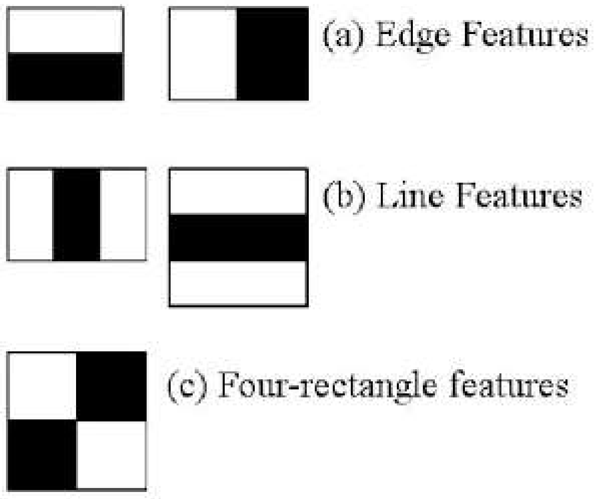
The algorithm can be explained in four stages:

- Calculating Haar Features
- Creating Integral Images
- Using Adaboost
- Implementing Cascading Classifiers

It's important to remember that this algorithm requires a lot of **positive images** of faces and **negative images** of non-faces to train the classifier, similar to other machine learning models.

Calculating Haar Features

The first step is to collect the Haar features. A **Haar feature** is essentially calculations that are performed on adjacent rectangular regions at a specific location in a detection window. The calculation involves summing the pixel intensities in each region and calculating the differences between the sums. Here are some examples of Haar features below.

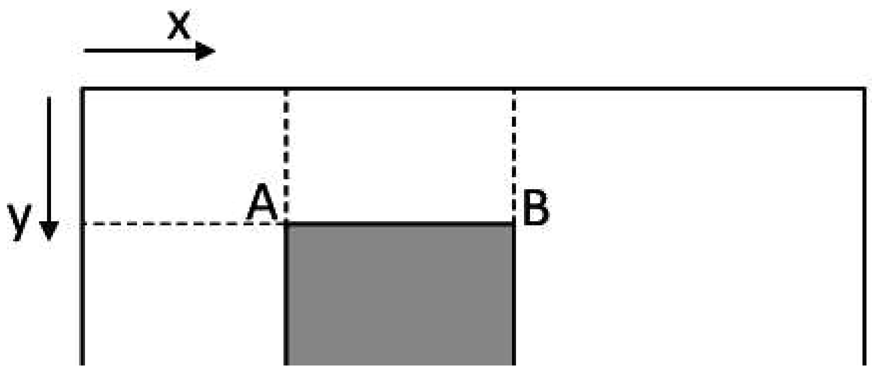


Types of Haar features. (Image Source)

These features can be difficult to determine for a large image. This is where **integral images** come into play.

Creating Integral Images

Without going into too much of the mathematics behind it (check out the paper if you're interested in that), integral images essentially speed up the calculation of these Haar features. Instead of computing at every pixel, it instead creates sub-rectangles and creates array references for each of those sub-rectangles. These are then used to compute the Haar features.



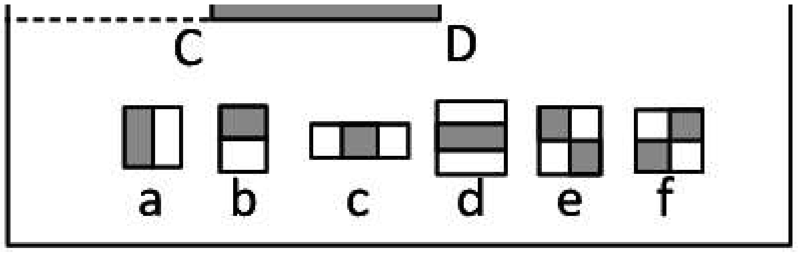


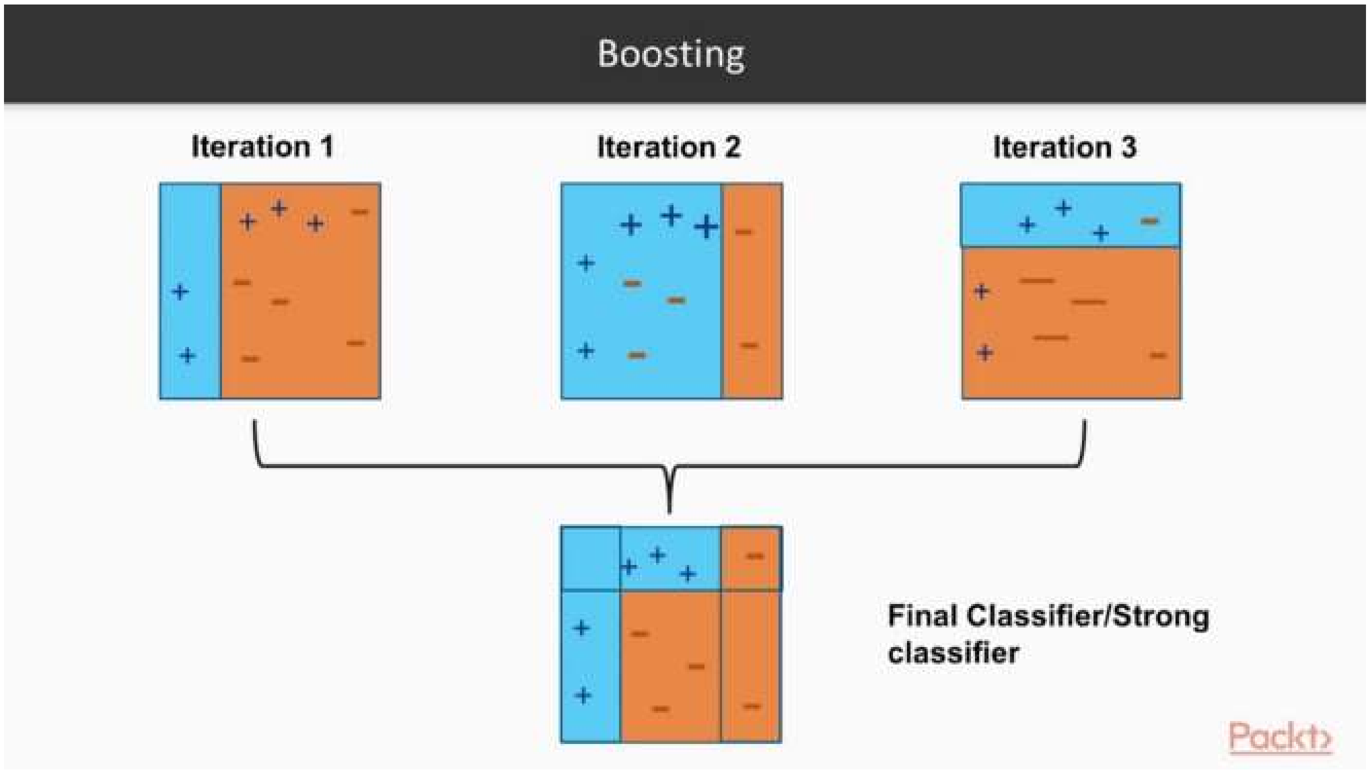
Illustration for how an integral image works. (Image Source)

It's important to note that nearly all of the Haar features will be **irrelevant** when doing object detection, because the only features that are important are those of the object. However, how do we determine the best features that represent an object from the hundreds of thousands of Haar features? This is where **Adaboost** comes into play.

Adaboost Training

Adaboost essentially chooses the best features and trains the classifiers to use them. It uses a combination of **"weak classifiers"** to create a **"strong classifier"** that the algorithm can use to detect objects.

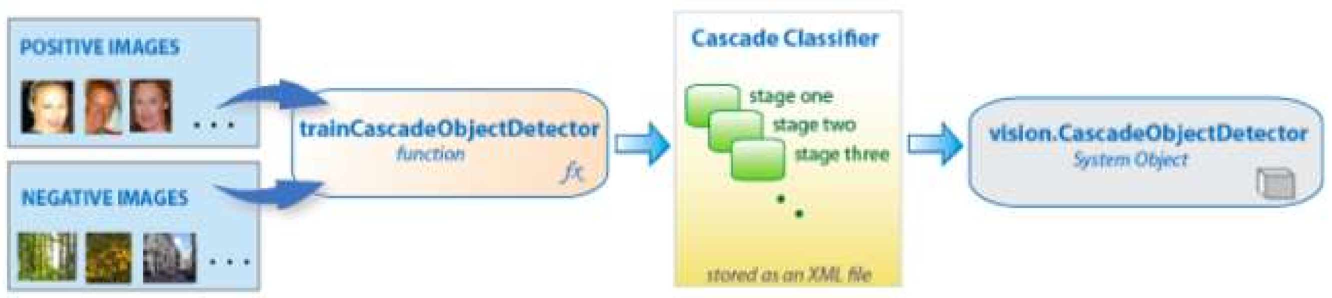
Weak learners are created by moving a window over the input image, and computing Haar features for each subsection of the image. This difference is compared to a learned threshold that separates non-objects from objects. Because these are "weak classifiers," a large number of Haar features is needed for accuracy to form a strong classifier.



Representation of a boosting algorithm. (Image Source)

The last step combines these weak learners into a strong learner using **cascading classifiers**.

Implementing Cascading Classifiers

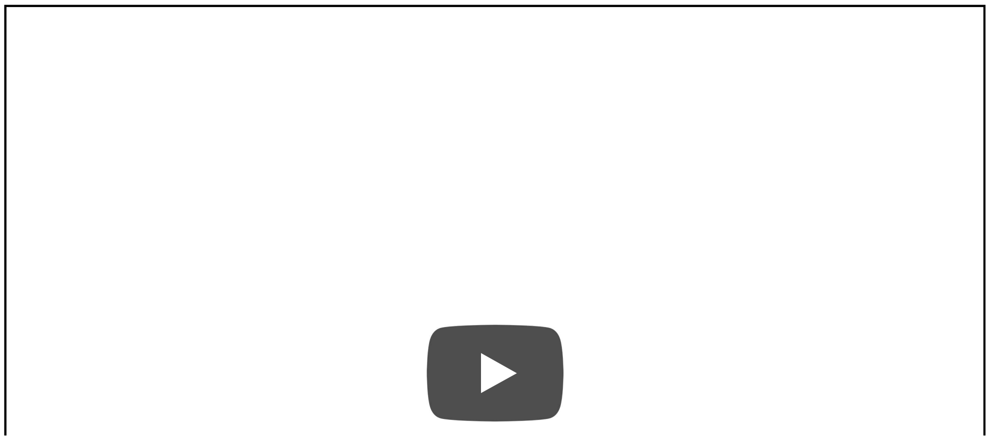


A flowchart of cascade classifiers. (Image Source)

The cascade classifier is made up of a series of stages, where each stage is a collection of weak learners. Weak learners are trained using boosting, which allows for a highly accurate classifier from the mean prediction of all weak learners.

Based on this prediction, the classifier either decides to indicate an object was found (positive) or move on to the next region (negative). Stages are designed to reject negative samples as fast as possible, because a majority of the windows do not contain anything of interest.

It's important to maximize a **low false negative rate**, because classifying an object as a non-object will severely impair your object detection algorithm. A video below shows Haar cascades in action. The red boxes denote "positives" from the weak learners.



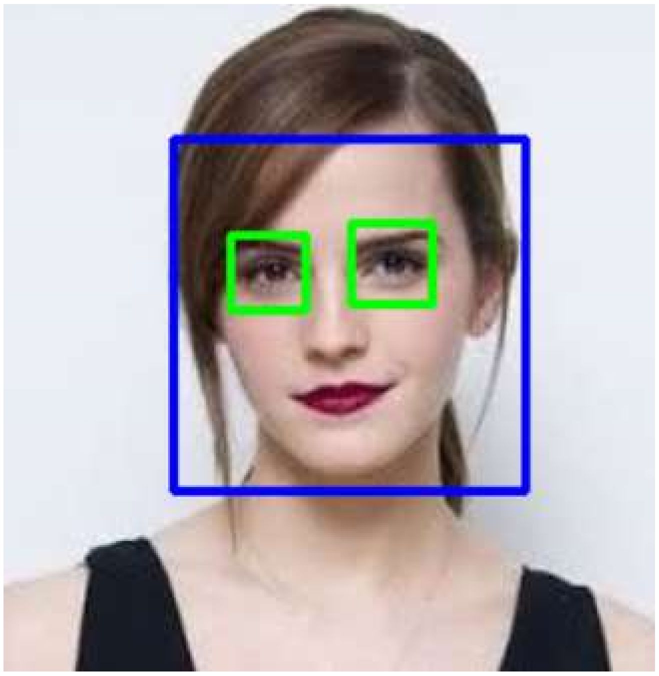
Haar cascades are one of many algorithms that are currently being used for object detection. One thing to note about Haar cascades is that it is very important to **reduce the false negative rate**, so make sure to tune hyperparameters accordingly when training your model.

Haar Cascades in Code

Implementing this into code is surprisingly easy using OpenCV's CascadeClassifier function.

```
import numpy as np
import cv2

f_cascade = cv2.CascadeClassifier("face.xml")
e_cascade = cv2.CascadeClassifier("eye.xml")

image = cv2.imread("actor.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = f_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
    img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    eyes = e_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

cv2.imshow('img',image)
cv2.waitKey(0)
```

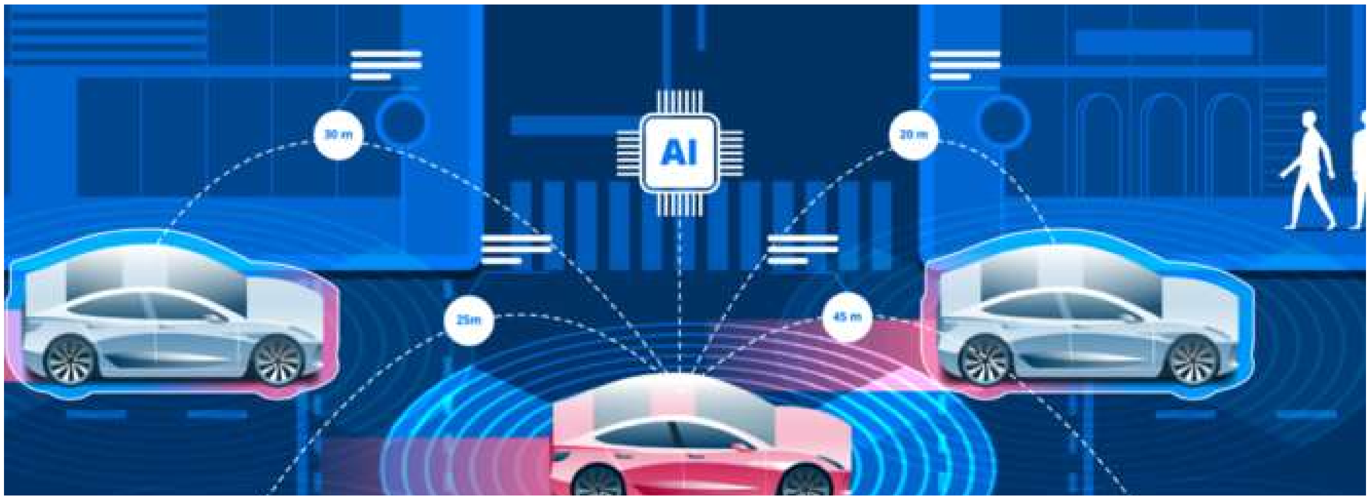
```
    cv2.destroyAllWindows()
```


Output of the above code.

As you can see here, this model is surprisingly accurate in detecting both eyes and faces. What's even more incredible is that the Haar classifier can be used to identify **any object** given enough training images to run upon.

A different type of approach using OpenCV's detectMultiScale function can be seen in GeeksForGeeks's article here. For live streaming face detection, check out this article for code in Java, C++, and Python.

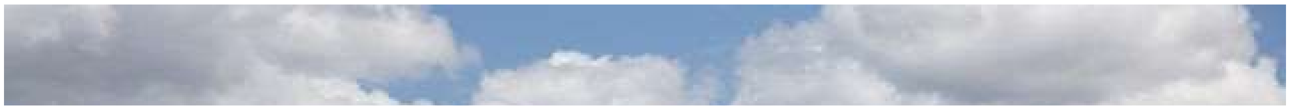
Applications of Haar Cascades



A representation of computer vision being used in autonomous vehicles. (Image Source)

The applications of this technology are enormous in a variety of different fields. A few of the most important applications are listed below:

- **Facial Recognition**: Similar to how iPhone X uses facial recognition, other electronic devices and security protocols can use Haar cascades to determine the validity of the user for secure login.
- **Robotics:** Robotic machines can "see" their surroundings to perform tasks using object recognition. For instance, this can be used to automate manufacturing tasks.
- **Autonomous Vehicles:** Autonomous vehicles require knowledge about their surroundings, and Haar cascades can help identify objects, such as pedestrians, traffic lights, and sidewalks, to produce more informed decisions and increase safety.
- **Image Search and Object Recognition:** Expanding off facial recognition, any variety of objects can be searched for by using a computer vision algorithm, such as Haar cascades.
- **Agriculture:** Haar classifiers can be used to determine whether harmful bugs are flying onto plants, reducing food shortages caused by pests.
- **Industrial Use:** Haar classifiers can be used to allow machines to pick up and recognize certain objects, automating many of the tasks that humans could previously only do.



A representation of computer vision being used in agriculture. (Image Source)

As shown above, Haar cascades and relevant computer vision technology will undoubtedly create a huge impact in the economy and ML world. Because of the versatility of Haar cascades, they can be applied virtually **anywhere**.

TL;DR

- Haar cascades are machine learning object detection algorithms.
- They use use Haar features to determine the likelihood of a certain point being part of an object.
- Boosting algorithms are used to produce a strong prediction out of a combination of “weak” learners.
- Cascading classifiers are used to run boosting algorithms on different subsections of the input image.
- Make sure to optimize against false negatives for Haar cascades.
- Use OpenCV for implementing a Haar cascade model yourself.

Further Reading

- Article: Deep Learning Haar Cascade Explained
- Paper: Rapid Object Detection Using a Boosted Cascade of Simple Features
- Article: Haar-Like Features
- Article: Face Recognition with OpenCV Haar Cascades
- Article: Difference Between Haar Classifiers and CNNs
- Article: Computer Vision with Haar Cascade Classifiers

If you want to talk more about Haar cascades or anything else, schedule a meeting: Calendly! For information about projects that I am currently working on, consider subscribing to my newsletter! Here's the link to subscribe. If you're interested in connecting, follow me on Linkedin, Github, and Medium.

Artificial Intelligence Machine Learning Haar Cascades Image Classification Computer Vision