**Face detection** – the ability to detect and locate human faces in a photograph.

Face detection is the first step in face recognition.

**Face Detection Using Camera** –

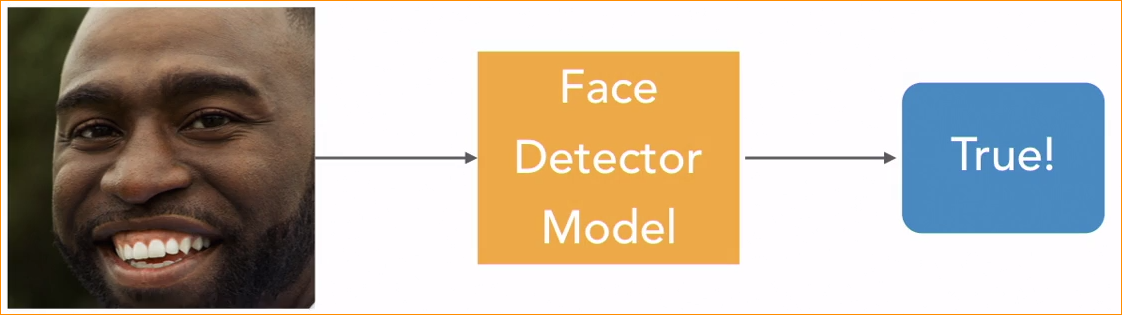
For that, the camera can focus before it takes the picture.

Easiest way to locate the object is to build a sliding window classifier.

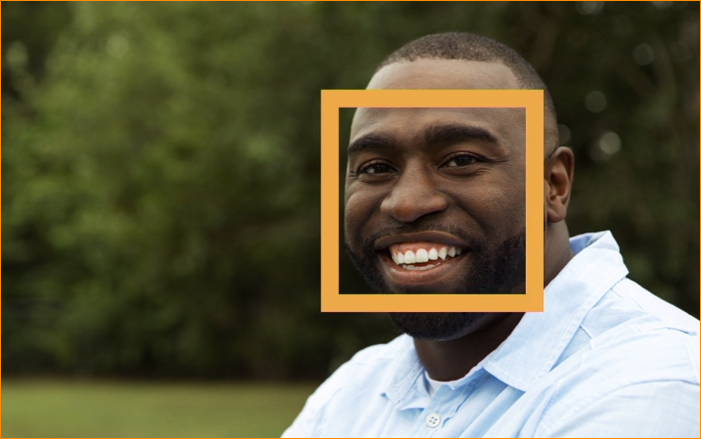
**Step 01 – Sliding window classifier.**

It has 2 steps –

1. Build a simple face detector model using machine learning. – all this model can so is look at a small image and tell it is face or not.



1. Slides the simple face detector across a larger image- each part of the image will check if a face is detected. When the face is detected we will record the location of the face. This is called a sliding window classifier. Because looking at one area or window of at a time.



2 – face

1 – no face

There are several classification algorithms we can use to detect if a region of an image contains a face.

We are gonna look at three of most common algorithms.

Face detection algorithms.



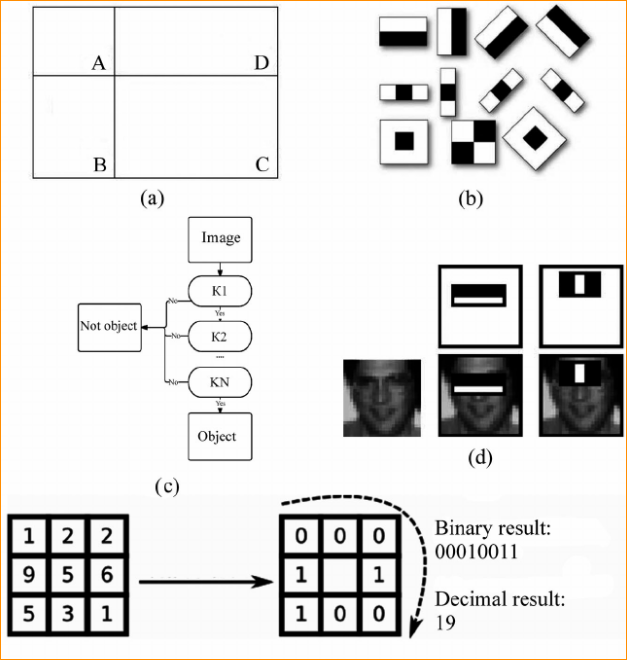
High accuracy

Low accuracy

Can use these 3 but it depend what is the application.

**Viola- Jones Algorithm**

* Invented by Paul Viola and Michael Jones in the early 2000s
* Uses decision trees to detect faces based on light and dark areas
* Pros- Very fast and great for low-powered devices
* Cons- Not very accurate



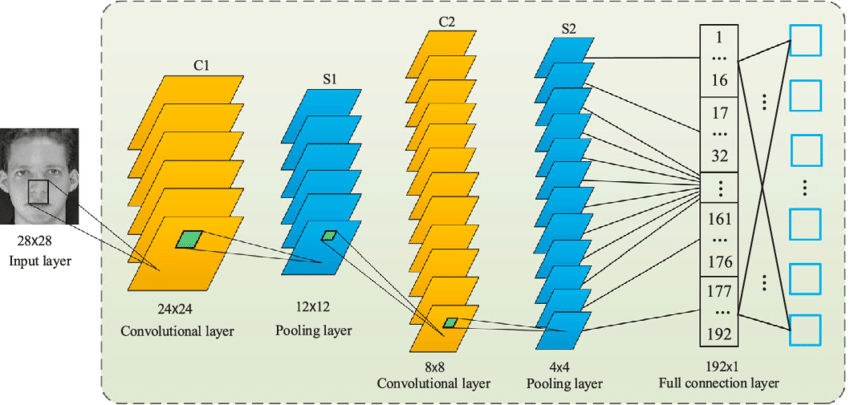
**Histogram of Oriented Gradients (HOG)**

* Invented in 2005
* Looks for the shifts from light to dark areas in an image
* Slower than Viola-Jones, but more accurate
* Runs well on normal computers without special hardware



**Convolutional Neural Network (CNN)**

* Uses a deep neural network to detect faces
* Very accurate, but requires a lot of training data
* Runs best on computers with dedicated GPUs



**Analyzing an Image as a Histogram of Oriented Gradients**

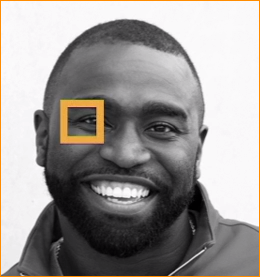


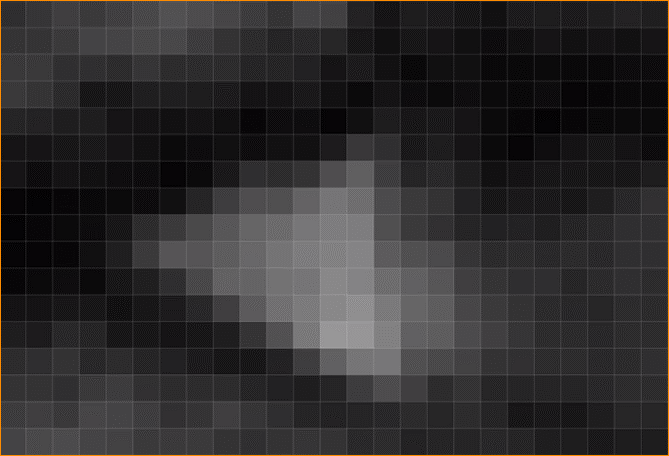
Here image that we want to check for faces.

1 Step – Convert image into black and white



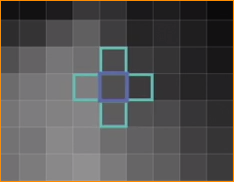
The HOG algorithm only looks at changes between light and dark areas in images.

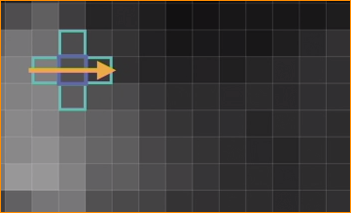
Now look every single pixel on an image once a time. Example – Zoom one area



There are pixels-

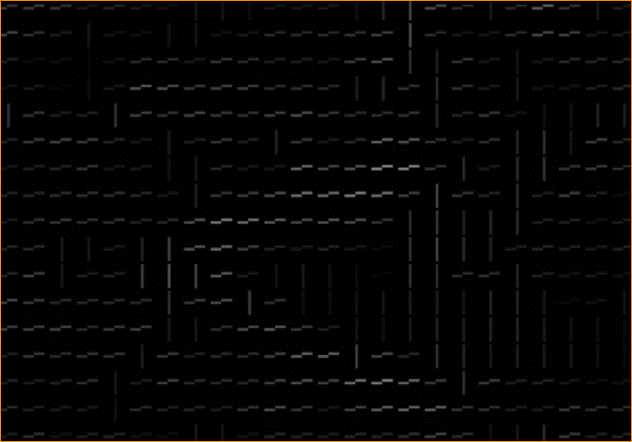
Now look directly next to the pictures

Our goal is – how dark is this pixel , compare to the pixels surrounding it, and then find the direction that where the biggest change happen.



In here we can see the pixels are darker than the pixels in this direction. That extract point the image transitions from a light area to a darker area. Base on this we are draw an arrow on the top of this pixel that points from left to right.

If we repeat every single pixel of image, the image turn into map of transition from light to dark areas.



This line call gradients.

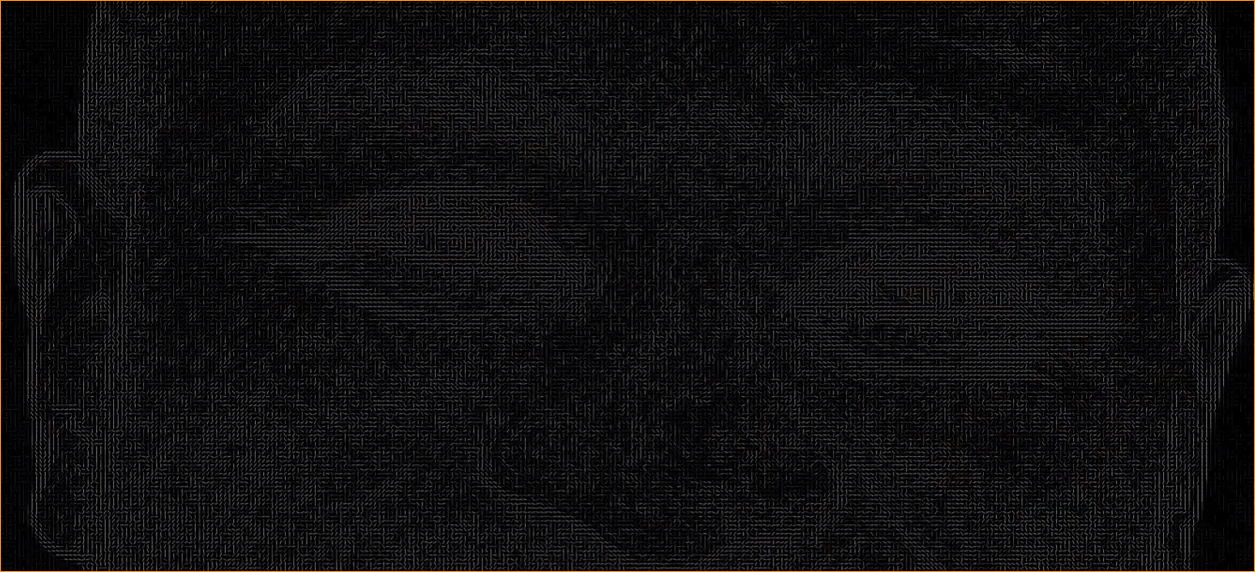
Gradient shows how the image flows from the light area to a dark at the point.

Un-zoom

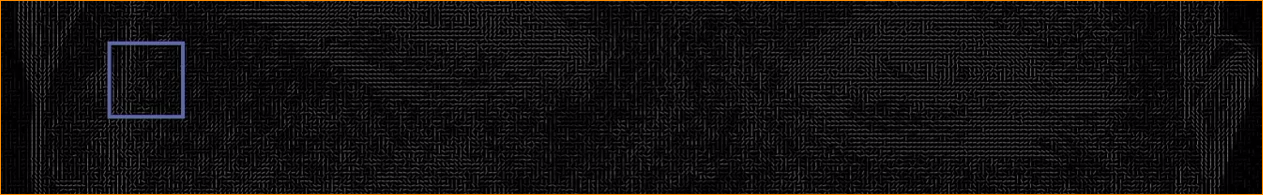


Here look like after replacing every pixel, the gradient map is simplified of the original image.

But it's still pretty complex.



Capturing the gradient for every single pixel is more detail that we need. To detect faces , we realy need the structure of the image. (We can simplify this representation further.) lets take 16x16 (256) pixel area as an example.



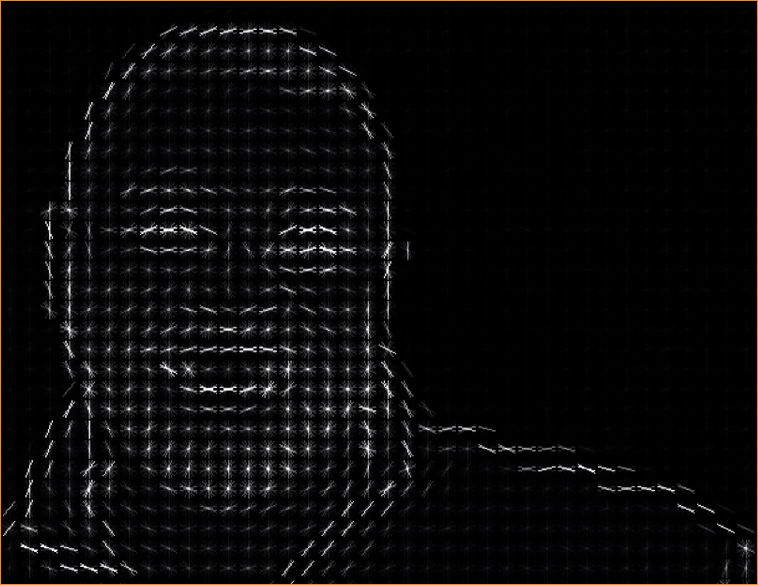
We count how many gradients that in each major direction. How may point, up, right and so on. Instead of keeping track of the 256 separate gradients with in this box just store a count of how many gradient of each direction.





Majority going top to bottom

In this **majority going from top to bottom**, so that’s then strongest factor representing this area of the image. There are also gradients pointing in other directions that we will keep track of. Represent of other directions here as lines that are less bold. Now we can repet this process for a entire image. This is the End result-



The original image is that representation thet capture the basic structure.

We can use this representation easy to the face detection model.

**Face Detection Code –**

* In this we use HOG pre-train HOG face detector to detect all the faces that appear in an image. Because most of the human face structure is same. So there is no new scratch.
* Here Imported PIL library class.



PIL – is Python image library, lets us easily display an image on the screen and draw lines on top of the image.

Import face recognition library.



This is a library that gives access to the face detection model DLIB.

Note –

face recognition is a Python library built on top of DLIB and other machine-learning frameworks. It provides a higher-level interface for face detection, recognition, and manipulation. It's relatively easy to use and powerful for various face-related tasks. Here's a basic example of how to use face recognition for face detection:



This script will use face recognition to detect faces in the input image and draw rectangles around the detected faces. Make sure to install face recognition and OpenCV (pip install face recognition opencv-python) before running the script.

To perform face detection of this library, need load image file. Then run the image file through the model.

Lets mark faces in this image using model –



People.jpg

Then load the image in to memory , we call facerecognition.load\_image\_file

