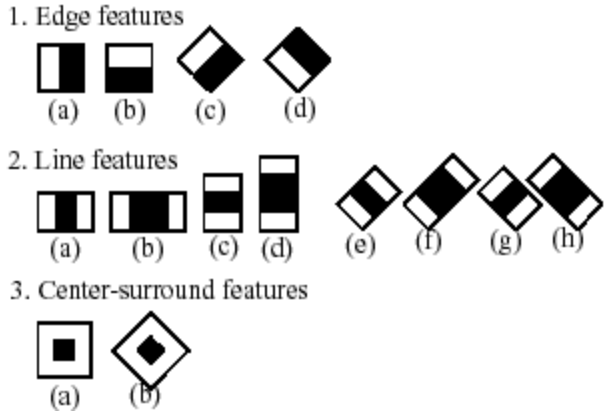
FACE RECOGNITION:

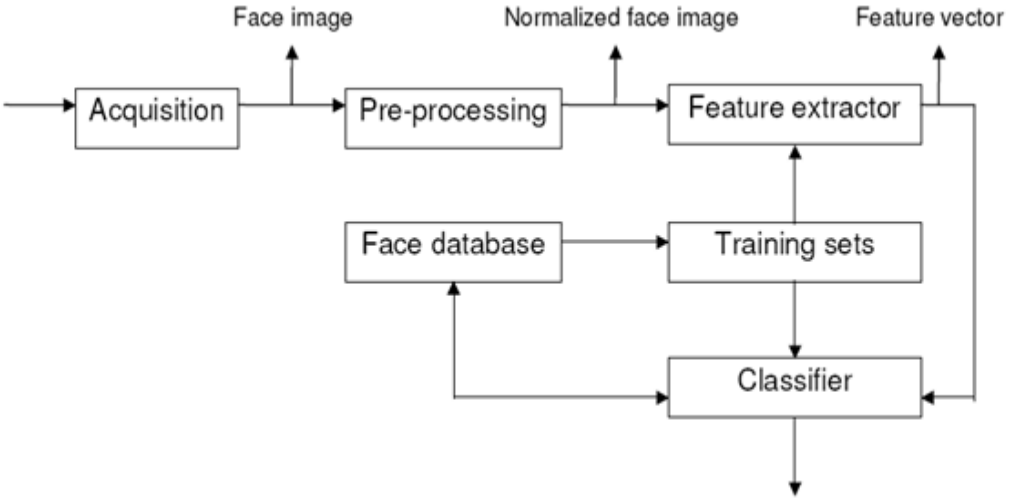
1. Haar Feature selection technique has a target to extract human face features. Haar features are like convolution kernels. These features are different permutations of black and white rectangles. In each feature calculation, we find the sum of pixels under white and black rectangles.
2. The word “cascade” in the classifier name means that the resultant classifier consists of several simpler classifiers (*stages*) that are applied subsequently to a region of interest until at some stage the candidate is rejected or all the stages are passed. The word “boosted” means that the classifiers at every stage of the cascade are complex themselves and they are built out of basic classifiers using one of four different boosting techniques (weighted voting). Currently Discrete Adaboost, Real Adaboost, Gentle Adaboost and Logitboost are supported. The basic classifiers are decision-tree classifiers with at least 2 leaves. Haar-like features are the input to the basic classifiers, and are calculated as described below. The current algorithm uses the following Haar-like features:

CASCADING CLASSIFIERS: HAAR FEATURE-BASED CASCADE CLASSIFIER FOR OBJECT DETECTION

For installing the cascading please follow the link: Cascade classifier

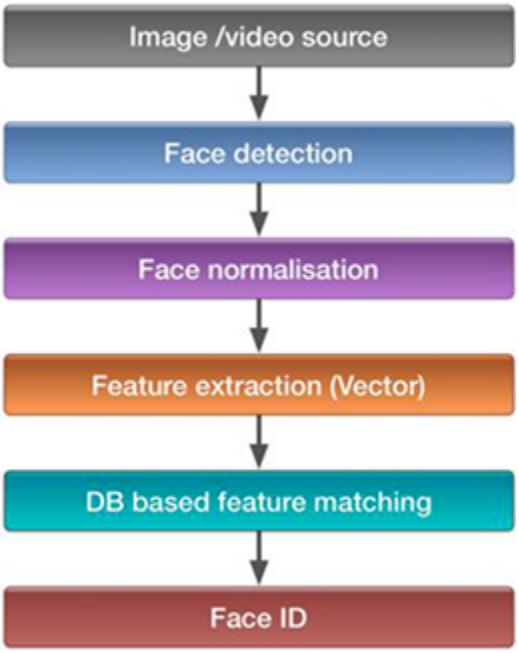
- First, a classifier (namely a *cascade of boosted classifiers working with haar-like features*) is trained with a few hundred sample views of a particular object (i.e., a face or a car), called positive examples, that are scaled to the same size (say, 20x20), and negative examples - arbitrary images of the same size.
- After a classifier is trained, it can be applied to a region of interest (of the same size as used during the training) in an input image. The classifier outputs a “1” if the region is likely to show the object (i.e., face/car), and “0” otherwise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. The classifier is designed so that it can be easily “resized” in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure should be done several times at different scales.





- The feature used in a particular classifier is specified by its shape (1a, 2b etc.), position within the region of interest and the scale (this scale is not the same as the scale used at the detection stage, though these two scales are multiplied). For example, in the case of the third line feature (2c) the response is calculated as the difference between the sum of image pixels under the rectangle covering the whole feature (including the two white stripes and the black stripe in the middle) and the sum of the image pixels under the black stripe multiplied by 3 in order to compensate for the differences in the size of areas. The sums of pixel values over a rectangular regions are calculated rapidly using integral images (see below and the **integral()** description).
- Import the pre-installed libraries or import quickly:

- **“pip install dlib”, “pip install face_recongnition”, “pip install pickle”,**
- The Face Recognition is basically divided into:
    1. Embedding Phase: The images are takes as input
    2. Recognition Phase: Now, we will recognize that particular person from the camera frame.



THE PROCESS IN THE FLOWCHART:

1. Import Libraries
2. Load the data from the data set.
3. Preprocess the data
4. Create the model
5. Train the Basic Model
6. Keep the test images
7. Feature Extraction
8. Output