Vidyavardhini’s College of Engineering & Technology
Department of Computer Engineering

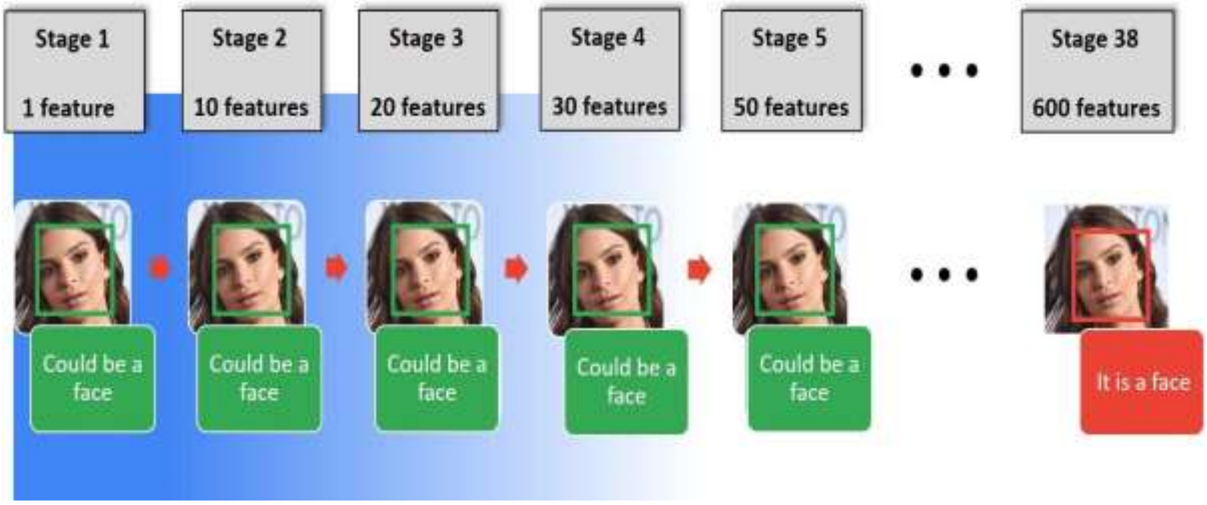
**Aim:** To Study Detecting and Recognizing Faces

**Objective:** To Conceptualizing Haar Cascades Getting Haar cascade data Using Open CV to Perform face detections performing face detection on still images.

## Theory:

1. **Introduction**

   Discover object detection with the Haar Cascade algorithm using OpenCV. Learn how to employ this classic method for detecting objects in images and videos. Explore the underlying principles, step-by-step implementation, and real-world applications. From facial recognition to vehicle detection, grasp the essence of Haar Cascade and OpenCV’s role in revolutionizing computer vision. Whether you’re a novice or an expert, this article will equip you with the skills to harness the potential of object detection in your projects.

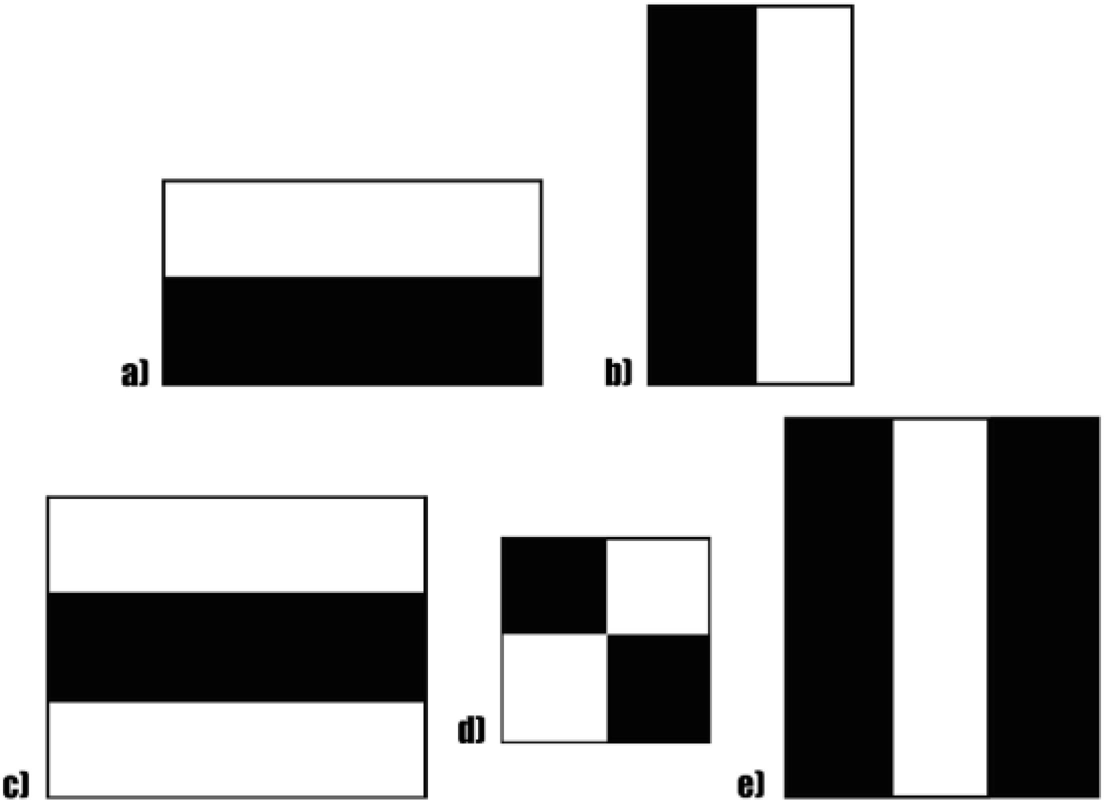


2. **Why Use Haar Cascade Algorithm for Object Detection?**

   Identifying a custom object in an image is known as object detection. This task can be done using several techniques, but we will use the haar cascade, the simplest method to perform object detection in this article.

3. **What is Haar Cascade Algorithm?**

   Haar cascade is an algorithm that can detect objects in images, irrespective of their scale in image and location. This algorithm is not so complex and can run in real-time. We can train a haar-cascade detector to detect various objects like cars, bikes, buildings, fruits, etc. Haar cascade uses the cascading window, and it tries to compute features in every window and classify whether it could be an object.



Haar cascade works as a classifier. It classifies positive data points → that are part of our detected object and negative data points → that don’t contain our object.

- Haar cascades are fast and can work well in real-time.
- Haar cascade is not as accurate as modern object detection techniques are.
- Haar cascade has a downside. It predicts many false positives.
- Simple to implement, less computing power required.

### 4. Conceptualizing Haar Cascades:

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

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.

### 5. Getting Haar Cascade Data:

The OpenCV library maintains a repository of pre-trained Haar cascades. Most of these Haar cascades are used for either:

- ➢ Face detection
- ➢ Eye detection
- ➢ Mouth detection
- ➢ Full/partial body detection

**Steps:**

i. Load a pre-trained Haar cascade from disk using the cv2.CascadeClassifer:

   1. detector = cv2.CascadeClassifier(path)

ii. Make predictions with it using the detect MultiScale function:

   1. results = detector.detectMultiScale(
   2. gray, scaleFactor=1.05, minNeighbors=5,
   3. minSize=(30, 30), flags=cv2.CASCADE_SCALE_IMAGE)

iii. The result is a list of bounding boxes that contain the starting x and y coordinates of the bounding box, along with their width (w) and height (h).

6. **Using Open CV to perform Face Detection:**

Using Open CV to perform Face Detection using Haar Cascade include four stages:

**Haar-feature selection:** Dark and light sections make up a Haar-like characteristic. By comparing the difference between the sum of the intensities of the bright and dark zones, it yields a single value. In order to identify an object, valuable elements must be extracted.

**Creation of Integral Images:** In the integral image, each pixel's left and rightmost neighbours are added together to get its value. Integral Images greatly shorten the time required to accomplish the operation of extracting Haar-like features, which requires computing the difference between rectangular dark and bright patches.

**AdaBoost Training:** The best features are chosen by this algorithm out of all features. It creates a "strong classifier" by combining many "weak classifiers" (best characteristics). The "strong classifier" that is formed is just a linear combination of all "weak classifiers."

**Cascade Classifier:** It is a technique for stacking ever-more complicated classifiers, like AdaBoost, that allows non-face input to be swiftly rejected while devoting more processing power to promising or positive face-like regions. It greatly shortens the computation time and improves the effectiveness of the procedure.

7. **Performing Face detection on a still image:**

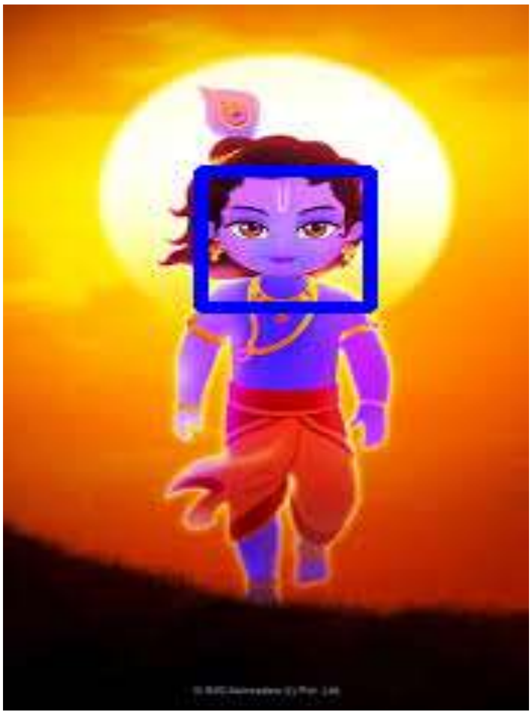
Performing face detection on still image can be done by following steps:

1. Loading and Converting image to grayscale.
2. Loading the required haar-cascade XML classifier file.
3. Applying the face detection method on the grayscale image.
4. Iterating through rectangles of detected faces

**Code:**

```
from imutils import paths

import cv2

from google.colab.patches import cv2_imshow

detector =
cv2.CascadeClassifier("/content/sample_data/haarcascade_frontalface_default.xml")

img = cv2.imread("/content/sample_data/i.jpeg")

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = detector.detectMultiScale(gray,1.1,4)

for (x, y, w, h) in faces:

        cv2.rectangle(img, (x, y), (x + w, y + h), (255,0, 0), 3)

cv2_imshow(img)
```

**Output:**



## Conclusion:

Haar Cascade is the simplest method to perform object detection. Haar cascade is an algorithm that can detect objects in images, irrespective of their scale in image and location. it tries to compute features in every window and classify whether it could be an object.

In this experiment we successfully implemented haar cascade method for object detection.