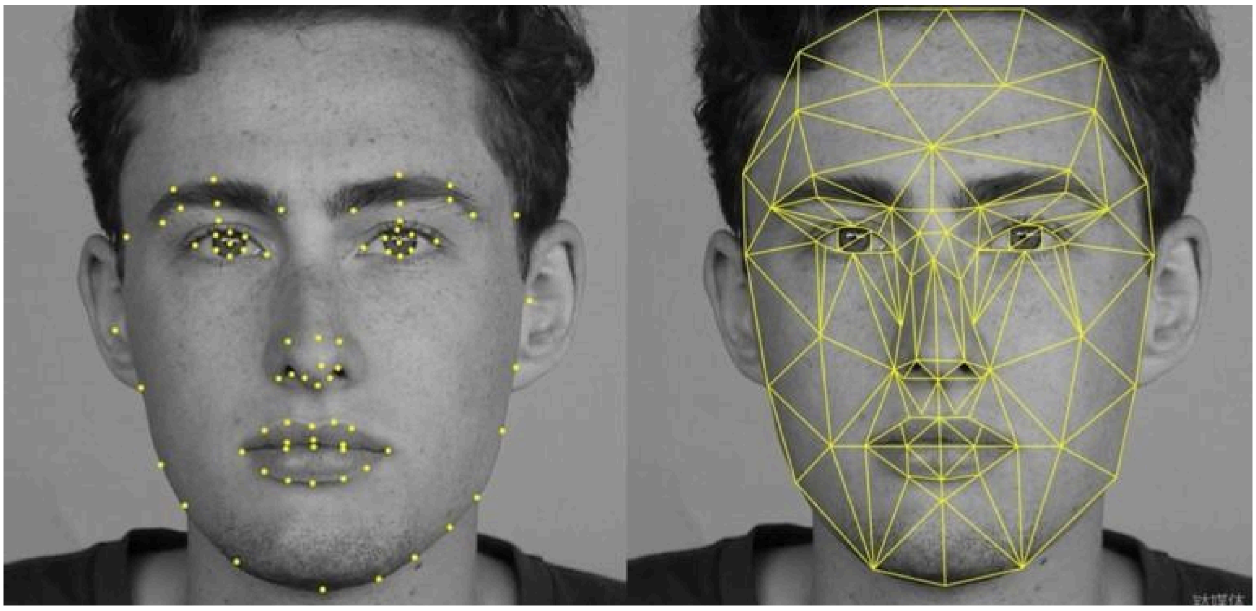
# Drowsiness Detection System Using Python

**By Pranav P**



## INTRODUCTION

This is a program that performs drowsiness detection on individuals and alerts them when necessary. The main modules used in this project are OpenCV and dlib, which are used for facial recognition and processing. This program is mainly focused to be used in vehicles to prevent driver drowsiness

### Facial detection using dlib and OpenCV

Dlib is an open source suite of applications and libraries written in C++ under a permissive Boost license. Dlib offers a wide range of functionality across a number of machine learning sectors, including classification and regression, numerical algorithms such as quadratic program solvers, an array of image processing tools, and diverse networking functionality, among many other facets.

Dlib also features robust tools for object pose estimation, object tracking, face detection (classifying a perceived object as a face) and face recognition (identifying a perceived face). It is an implementation of machine learning.

Though Dlib is a cross-platform resource, many custom workflows involving facial capture and analysis (whether recognition or detection) use the OpenCV library of functions, operating in a Python environment, as in the image below.

A Dlib Face Recognition Network model has 29 convolutional layers, an optimized version of the well-used ResNet-34 network.

Face detection using Haar cascades is a machine learning based approach where a cascade function is trained with a set of input data. OpenCV already contains many pre-trained classifiers for face, eyes, smiles, etc..

## Cascade Classifiers and Haar Features:

Cascade Classifiers and Haar Features are the methods used for Object Detection.

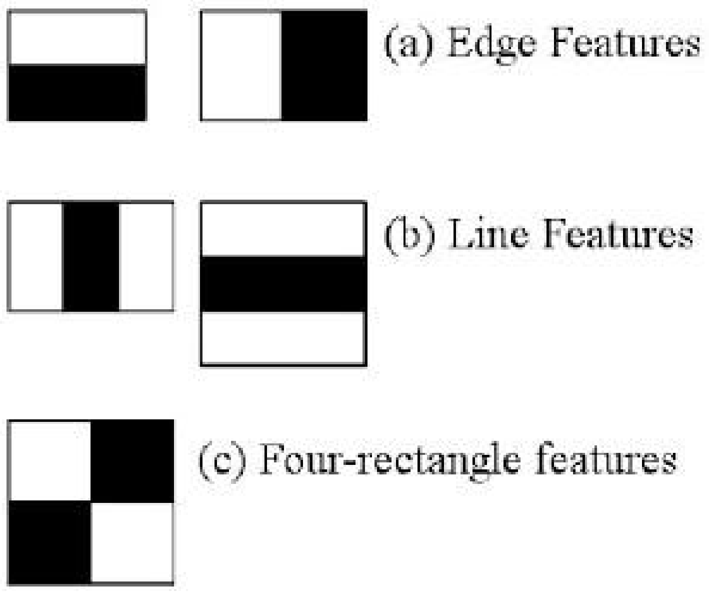
It is a machine learning algorithm where we train a cascade function with tons of images. These images are in two categories: positive images containing the target object and negative images not containing the target object.

There are different types of cascade classifiers according to different target objects. In our project, we will use a classifier that considers the human face to recognize it as the target object.

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.

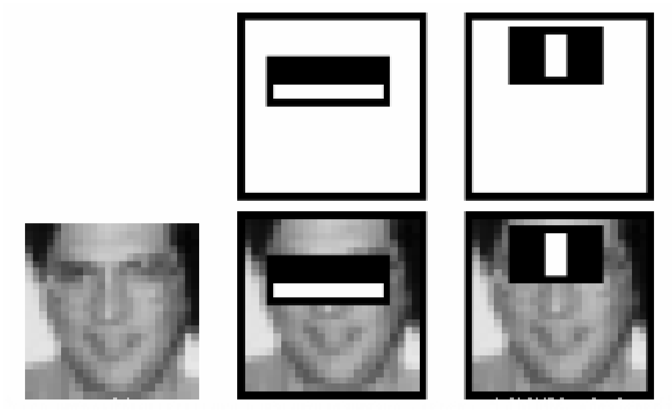
Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function 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 the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle.



Now, all possible sizes and locations of each kernel are used to calculate lots of features.For each feature calculation, we need to find the sum of the pixels under white and black rectangles(Even a 24x24 window results in over 160000 features). To solve this, they introduced the integral image. However large your image, it reduces the calculations for a given pixel to an operation involving just four pixels.But among all these features we calculated, most of them are irrelevant. For example, consider the image below.

The top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applied to cheeks or any other place is irrelevant. So  we select the best features out of 160000+ features using **Adaboost**.



For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. Obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that most accurately classify the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then the same process is done. New error rates are calculated. Also new weights. The process is continued until the required accuracy or error rate is achieved or the required number of features are found).

The final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain)

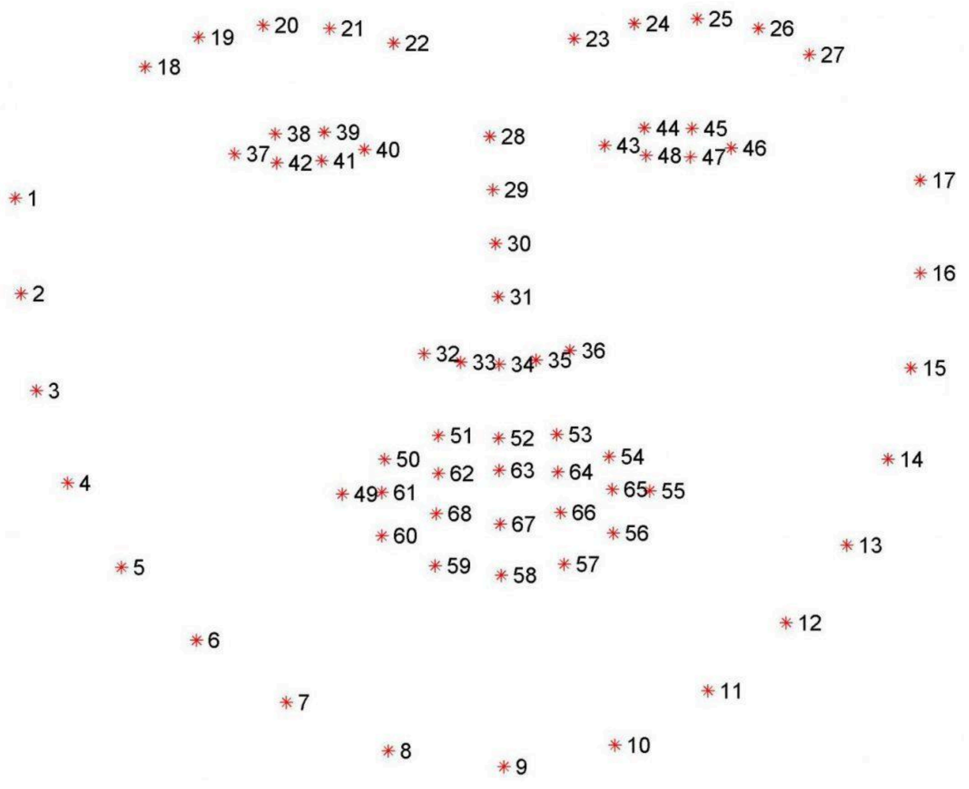
In an image, most of the image is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot, and don't process it again. Instead, focus on regions where there can be a face. This way, we spend more time checking possible face regions.

For this they introduced the concept of **Cascade of Classifiers**. Instead of applying all 6000 features on a window, the features are grouped into different stages of classifiers and applied one-by-one. (Normally the first few stages will contain very few features). If a window fails the first stage, discard it. We don't consider the remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region.

## How the program works

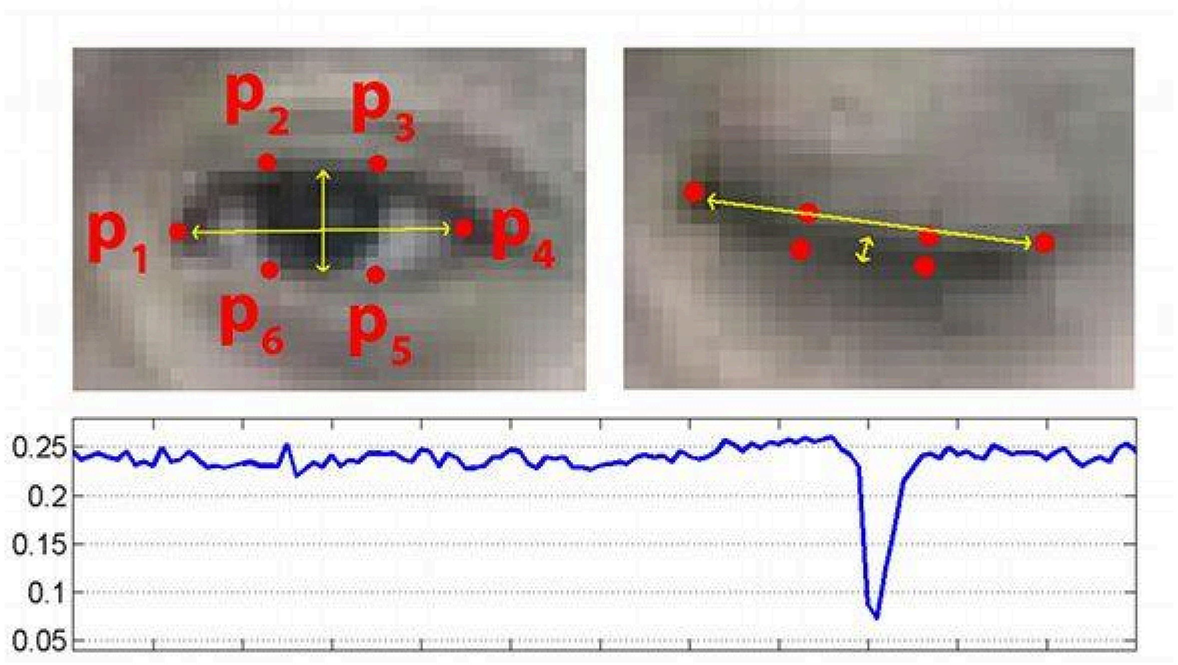
To start, we will apply OpenCV’s Haar cascades to detect the face in an image, which helps to find the bounding box *(x, y)*-coordinates of the face in the frame.

Given the bounding box the face we can apply dlib’s facial landmark predictor to obtain **68 salient points** used to localize the eyes, eyebrows, nose, mouth, and jawline:



dlib’s 68 facial landmarks are *indexable* which enables us to extract the various facial structures using simple Python array slices.

Given the facial landmarks associated with an eye, the *Eye Aspect Ratio (EAR)* algorithm which was introduced by Soukupová and Čech's in their 2017 paper is applied:



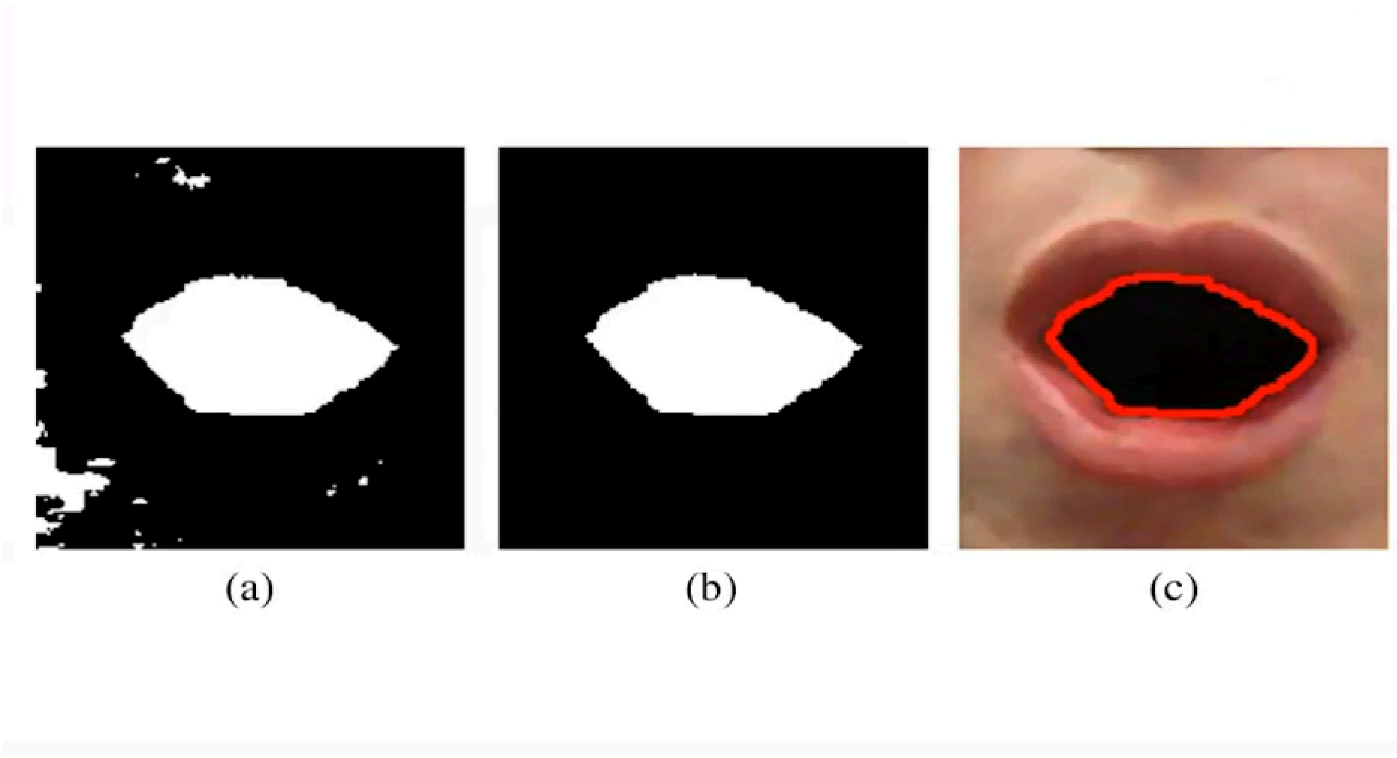
On the *top-left* we have an eye that is fully open and the eye facial landmarks plotted. Then on the *top-right* we have an eye that is closed. The *bottom* then plots the eye aspect ratio over time. As we can see, the eye aspect ratio is constant (indicating that the eye is open), then rapidly drops to close to zero, then increases again, indicating a blink has taken place.

we'll be monitoring the eye aspect ratio to see if the value *falls* but *does not increase again*, thus implying that the driver/user has closed their eyes.

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

The equation given above allows us to calculate the aspect ratio of the eye and set threshold values.

Similarly we monitor the aspect ratio of the mouth by measuring the coordinate points on the mouth from the detections present in the Haar cascade. We are able to construct a dlib.rectangle object corresponding to the bounding box (x, y)-coordinates in the image. This object was fed into dlib's facial landmark predictor which in turn gives us the set of localized facial landmarks on the face this in turn helps to see if the person is yawning or not :



If the EAR value and the YAWN distance ratio value falls below the specified threshold then the program calls the text to speech converter used here called 'espeak' to sound the alarm, which may be programmed to sound any texts to alert the user.

## Conclusion

This is a program based on python used to perform drowsiness detection on individuals and alert them when necessary, the program uses OpenCV and dlib for facial detection and analysis, then the EAR (Eye Aspect Ratio) algorithm is used to find the distance between the points on the eye and similarly the aspect ratio of the mouth is measured to see if the person is yawning, then a text to speech converter is used to sound off the alarm when necessary.

## Resources And Acknowledgements

https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

https://pyimagesearch.com/

https://pyimagesearch.com/2017/10/23/raspberry-pi-facial-landmarks-drowsiness-detection-with-opencv-and-dlib/