Sap ID: 60004200082
Name: Pratham Bhoir

Eye Drowsiness Detector

- ## Face detection (Pre-processing)

  -locating and localizing one or more faces in an image.
  Locating refers to identifying the position and finding the coordinate of the face in the image. Localizing refers to setting up the boundaries of the image, generally by drawing a rectangle around the region of interest.

  One of the pre-processing steps included for detecting whether the eyes in the dataset images are drowsy or not is to select a classifier to detect a face object in an image. For that purpose we have the following algorithms:

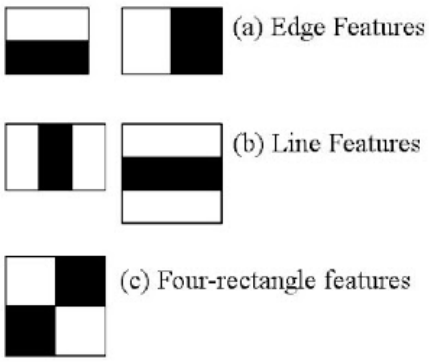
1. HAAR Cascade pre-trained classifiers (OpenCV)
2. HOG + Linear SVM face detector (dlib)
3. Convolutional Neural Network (dlib)

1) HAAR Cascades

HAAR Cascades is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge or line detection features.
The algorithm requires a set of positive samples and a set of negative images. The set of negative samples must be prepared manually, whereas set of positive samples is created using the opencv_createsamples application.

Features:

Haar features are a set of kernels of simple patterns which makes it easy to find out the edges or the lines in the image, or to pick areas where there is a sudden change in the intensities of the pixels.



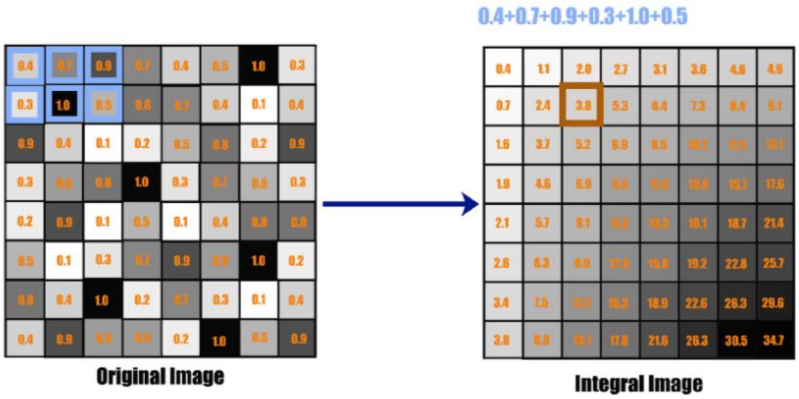
The kernel defines a dark area and a light area with a fixed no. of pixels for each side. The HAAR feature will detect an edge if the value of the expression given below is close to 1:

$$\left\{\frac{Sum\ of\ dark\ pixels}{number\ of\ dark\ pixels} + \frac{Sum\ of\ light\ pixels}{number\ of\ light\ pixels}\right\}$$

The HAAR feature traverses through the image pixel by pixel to detect all the edges. This leads to a large number of calculations which are practically impossible to calculate. To tackle this, we have the concept of **Integral image.**

Integral Image

The integral image is constructed in such a way that each pixel in this is the sum of all the pixels lying in its left and above in the original Image. While using the integral image, only 4 constant value additions are to be performed irrespective of the number of pixels of the image or the kernel size. This makes it more efficient and faster.



Adaboost

Adaboost is a boosting technique that creates weak learners which produce low error rates. Weak learners misclassify only a minimum number of images. They greatly increase the efficiency and performance of the model. With this technique, the final set of features gets reduced from 180000 to 6000.

Attentional Cascades

This technique is proposed to reduce the number of features to be run on each window. Here, all the features are sorted and applied in stages. The simpler ones are applied earlier and if no facial feature is detected, that window is discarded.

Implementation

We can use cv.cascadeclassifier() module available in OpenCV package to make use of HAAR cascades. We can download a pre-trained model named as 'haarcascade_frontalface_default.xml' from OpenCV GitHub project.

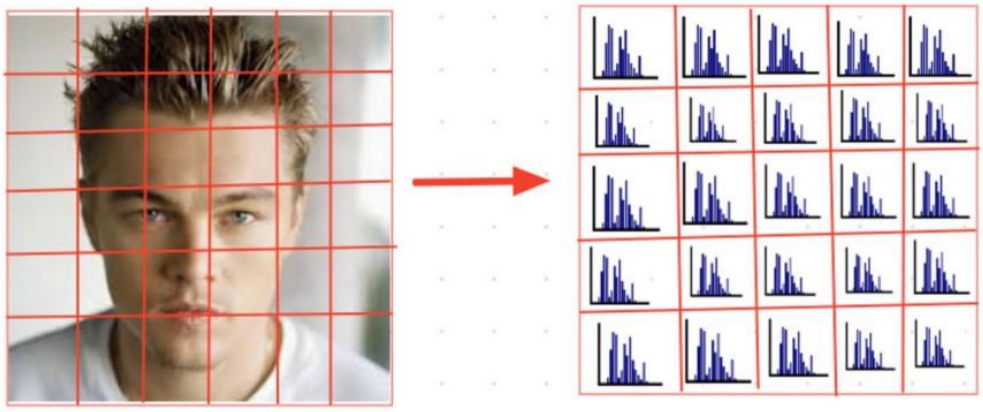
2) Dlib library

Dlib is a deep learning-based module that supports face detection and is one of the most utilized packages for face detection and recognition.

There are two face detection methods available in the dlib library:

1. ### HOG + Linear SVM face detector

   This method is quite accurate and more computationally efficient than HAAR cascade. HOG is a simple and powerful feature descriptor. HOG is robust for object detection because object shape is characterized using the local intensity gradient distribution and edge direction.



The way HOG works is that it divides the image into small connected cells and then computes a histogram with respect to each cell based on the local intensity gradient. Later on, it combines all the histograms and forms a single histogram vector which is unique for each face.

The only disadvantage of HOG is it is unable to detect faces which are at an angle, though it is very useful to detect straight and front faces like in a document. It is, therefore, not recommended to use it for real-time video.

We can use the "dlib.get_frontal_face_detector()" function available in dlib library to implement this method.
The "get_frontal_face_detector" function does not accept any parameters. A call to it returns the pre-trained HOG + Linear SVM face detector included in the dlib library.

2. Convolutional Neural Network (CNN)

CNN is a class of deep networking, it works really well for non-frontal faces at odd angles where HOG based detector is not good at it. Dlib supports CNN based face detection models which can be found in GitHub repositories. They are stored in a ".dat" file format. A Max-Margin (MMOD) CNN face detector that is both highly accurate and very robust, capable of detecting faces from varying viewing angles, lighting conditions, and occlusion.
The MMOD face detector can run on a CPU, but works best and extremely fast with a GPU.

Advantages:

- CNN based face detection is so accurate, it can find faces at odd angles and orientations.
- It works on CPU but much faster on GPU
- Very easy to use

Dis-Advantages:

- Works slow on CPU
- Doesn't detect small faces (you can notice that CNN missed small faces from the output)

We can implement this technique by calling the method "dlib.cnn_face_detection_model_v1(modelPath)". This method accepts a single argument modelPath which is the path of the pretrained CNN model "mmod_human_face_detector.dat".

Choice of Model:

According to me the HOG + Linear SVM face detector method seems to be the most appropriate choice because:

a) It is easier to use.
b) It is not as outdated as HAAR cascades.
c) Doesn't require a powerful GPU to function efficiently and quickly.
d) As we are working on a dataset of images taken from the dashboard of the car, the faces would be straight and front facing.

• Eye Detection

For eye detection, we can choose between the following two methods:

1. HAAR cascade

The working of this technique is similar to the one explained in the face detection section.
It is implemented by calling the same cv.cascadeclassifier() module but this time, we pass the pretrained haarcascade_eye.xml model which is specifically for the detection of eyes. Alternatively, you can use haarcascade_eye_tree_eyeglasses.xml model if you are dealing with glasses.

2. Dlib Shape Predictor

-For the purpose of eye drowsiness detection, we are going to use this this approach to detect the eye objects in the images.

Understanding facial landmarks

Facial landmarks are used to localize and represent salient regions of the face, such as Eyes, Eyebrows, Nose, Mouth, Jawline. Some of the applications of facial landmarks are face alignment, head pose estimation, face swapping, blink detection, etc. Out of these, we are going to use it for blink detection.
Facial landmarks uses shape prediction methods on faces to detect the various facial structures.

Algorithm

This is a 2-step process:
Step 1: Localize the face in the image.
Step 2: Detect the key facial structures on the face ROI.

As mentioned earlier, we are using HOG + Linear SVM face detector to achieve step 1. This will provide us with the face bounding box (i.e., the (x, y)-coordinates of the face in the image).

We are going to use the facial landmark detector included in the dlib library which is an implementation of the “One Millisecond Face Alignment with an Ensemble of Regression Trees” paper by Kazemi and Sullivan (2014).
This method starts by using:

1. Using a training set of labelled facial landmarks on an image. These images are manually labelled, specifying specific (x, y)-coordinates of regions surrounding each facial structure.

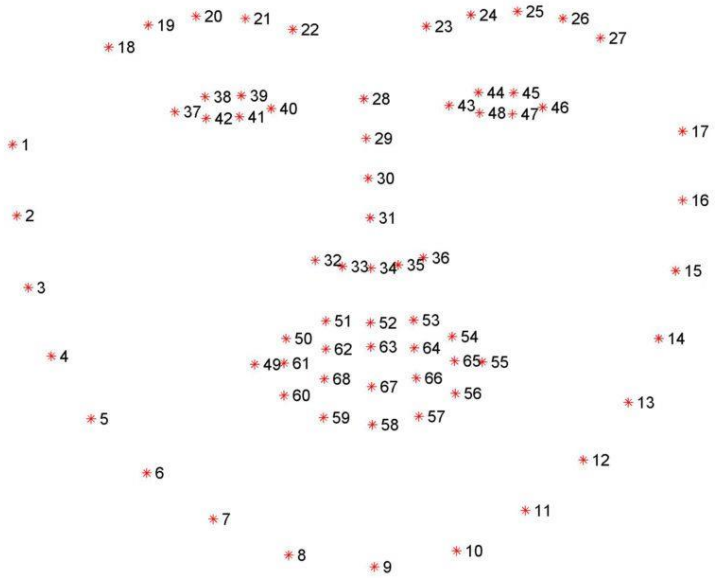
2. The probability of distance between pairs of input pixels (priors).

An ensemble of regression trees are trained to estimate the facial landmark positions directly from the pixel intensities themselves (i.e., no “feature extraction” is taking place). This creates the required facial landmark detector.

There exists 2 types of facial landmark detectors in dlib library:

1. 68-point variant
2. 5-point variant (used for face alignment)

We are going to select the 68-point variant as we will be requiring to detect the eyes, which will be more efficient. It is used to estimate the location of 68 (x, y)-coordinates that map to facial structures on the face.



These annotations are part of the 68 point iBUG 300-W dataset which the dlib facial landmark predictor was trained on.

We can implement this technique by the function dlib.shape_predictor(ModelPath) which takes the ModelPath as the argument. We are going to pass the path of “shape_predictor_68_face_landmarks.dat” model file as the argument

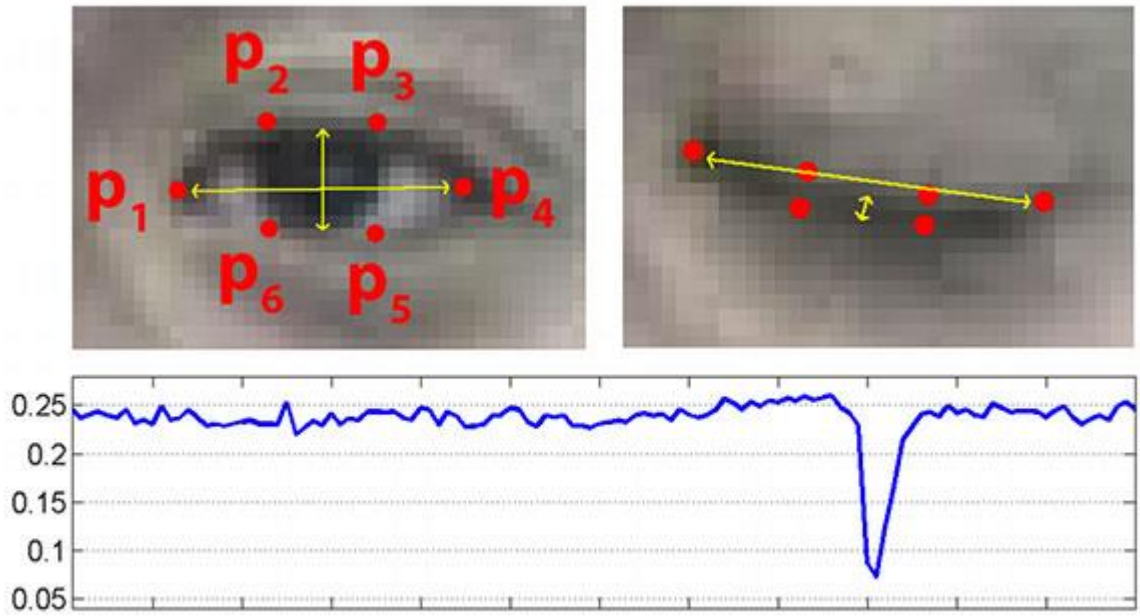
- # Feature Extraction

Now, we have 68 points marked on the face whenever an image of a face is detected. Considering the indexing begins from 0 and ends with 67, we have to focus on the 12 points from index number 36 to 47, as these are the points which are associated with the eyes.
Base on just the eyes, we can use Eye Aspect Ratio (EAR) as the core feature to determine the drowsiness status of the driver.

Eye Aspect Ratio (EAR)

EAR is defined as the ratio of the length of the eyes to the width of the eyes. The length of the eyes is calculated by averaging over two distinct vertical lines across the eyes.
We can calculate the distance between 2 points by using the function np.linalg.norm(ptA - ptB) provided by numpy library. Here we have imported numpy as np. The function finds the Euclidean distance between the given two points passes as the argument.

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



As evident from the image as well, this ratio lies between 0 and 1 as eye width is always greater than the averaged eye length. While training our

model with sample image datasets, we can check the value of the ratio for labelled eyes opened and eyes closed images. Generally, a pair of eyes with ratio value greater than 0.25 is considered to be completely opened . The ratio value less than 0.21 is considered to be closed.

- # Conclusion

We used the HOG + Linear SVM face detector approach for face detection as it is a more accurate and better approach than HAAR Cascade. Even though, CNN approach is more efficient and produces less errors, it can be slower sometimes.
For eye detection, we used the facial landmark method using dlib shape predictor as it is more accurate than HAAR cascade.
Alternatively, one can use the 3D mesh available in the MediaPipe library.
We can train our model to collect data of the EAR for all images. Accordingly, we set the parameters to detect drowsiness.