Priyansh Shah
TE-Computer
Div-A
60004200048

DJS Synapse

Synopsis for drowsiness detection
Using Machine Learning:
COMPUTER VISION
(Task 2-ML Developer)

COMPUTER VISION

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image.

Computer vision trains machines to perform these functions, but it has to do it in much less time with cameras, data and algorithms rather than retinas, optic nerves and a visual cortex. Because a system trained to inspect products or watch a production asset can analyze thousands of products or processes a minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities.

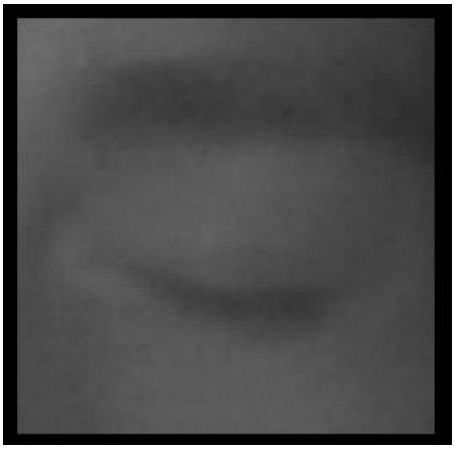
PROBLEM STATEMENT:

- Given a dataset of human face images taken from a dashboard of a car, detect if the driver is drowsy or not. Note that a driver can be classified as drowsy if their eyes are closed more often than open.
    - Describe the preprocessing steps required on the image (if any).
    - Describe techniques you would use to detect the eyes in the entire image.
    - Describe techniques you would use to detect drowsiness in the eyes.
    - End the synopsis with a conclusion which should contain information about why you selected a particular algorithm over other existing solutions.

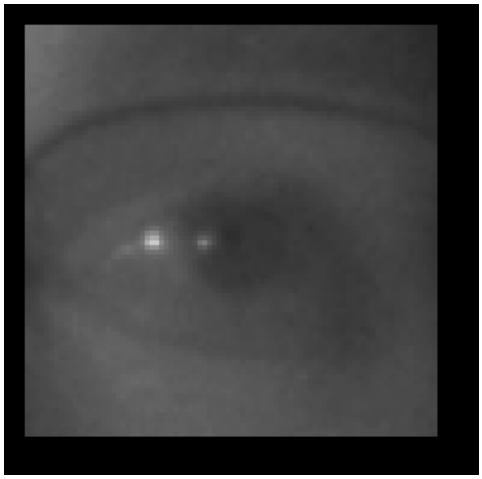
    Note : All modelling techniques must be described in detail. Mathematical explanations would fetch you brownie points! (code for the same is optional).

    Consider this as a supervised learning problem where you have been given images of open and closed eyes for training.

    Dataset contains images like the ones provided below :



closed eye



open eye

Solution:

Techniques for detecting eyes from the entire image:

There are 2 steps to detect eyes from the images:

1. use dlib and OpenCV to detect facial landmarks in an image
2. use our detected facial landmarks to help us label and extract face regions, including:

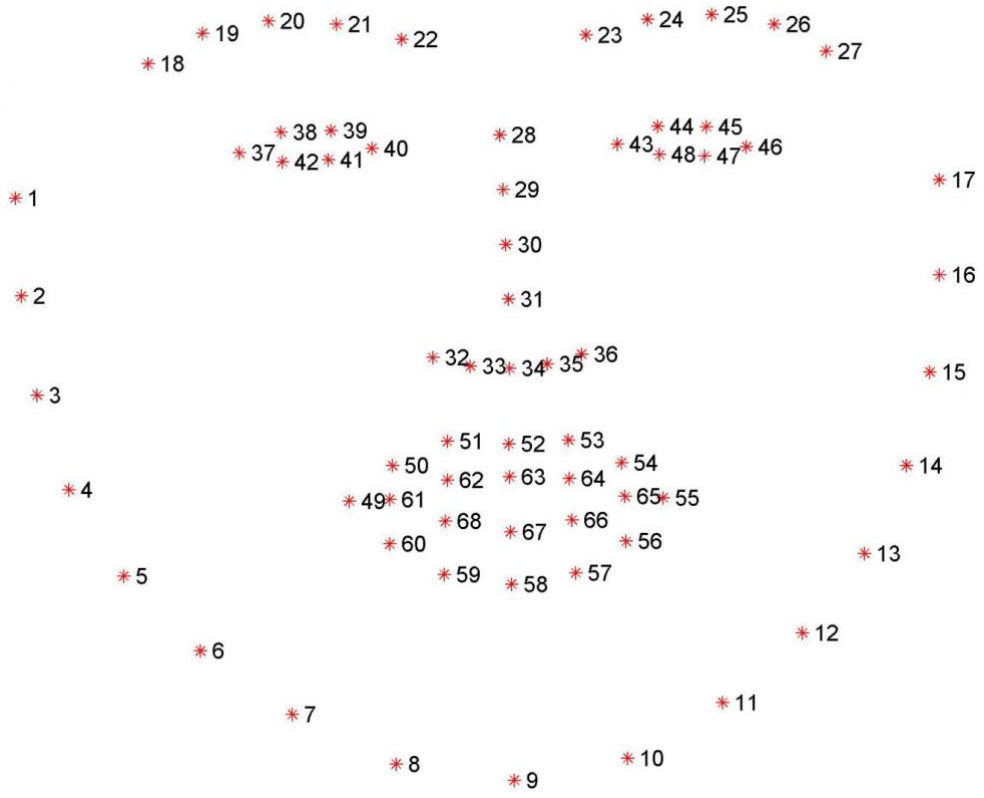
    - Mouth
    - Right eyebrow
    - Left eyebrow
    - Right eye
    - Left eye
    - Nose
    - Jaw

Step 1:

Facial landmarks have been successfully applied to face alignment, head pose estimation, face swapping, blink detection and much more. Detecting facial landmarks is a subset of the shape prediction problem.

The facial landmark detector implemented inside dlib produces 68 (x, y)-coordinates that map to specific facial structures. These 68 point mappings were obtained by training a shape predictor on the labeled iBUG 300-W dataset.

Below we can visualize what each of these 68 coordinates map to:



Step 2:

The next step is to use our detected facial landmarks to help us label and extract face regions.

Examining the image, we can see that facial regions can be accessed via simple Python indexing (assuming zero-indexing with Python since the image above is one-indexed):

- The mouth can be accessed through points [48, 68].
- The right eyebrow through points [17, 22].
- The left eyebrow through points [22, 27].
- The right eye using [36, 42].
- The left eye with [42, 48].
- The nose using [27, 35].
- And the jaw via [0, 17].

Using this dictionary, we can easily extract the indexes into the facial landmarks array and extract various facial features simply by supplying a string as a key.

Techniques you would use to detect drowsiness in the eyes:

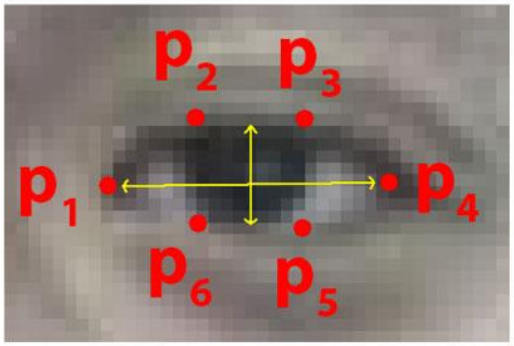
After performing facial landmark detection in real-time in video streams, we are going to build upon this knowledge and develop a computer vision application that is capable of detecting and counting blinks in video streams using facial landmarks and OpenCV.

To build our blink detector, we'll be computing a metric called the eye aspect ratio (EAR), introduced by Soukupová and Čech in their 2016 paper, Real-Time Eye Blink Detection Using Facial Landmarks.

From the previous section we know that we can extract specific facial structures by knowing the indexes of the particular face parts.

In terms of blink detection, we are only interested in two sets of facial structures — the eyes:

Each eye is represented by 6 *(x, y)*-coordinates, starting at the left-corner of the eye (as if you were looking at the person), and then working clockwise around the remainder of the region:



Based on this image, we should take away on key point:

There is a relation between the width and the height of these coordinates.

Based on the work by Soukupová and Čech in their 2016 paper, Real-Time Eye Blink Detection using Facial Landmarks, we can then derive an equation that reflects this relation called the eye aspect ratio (EAR):

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

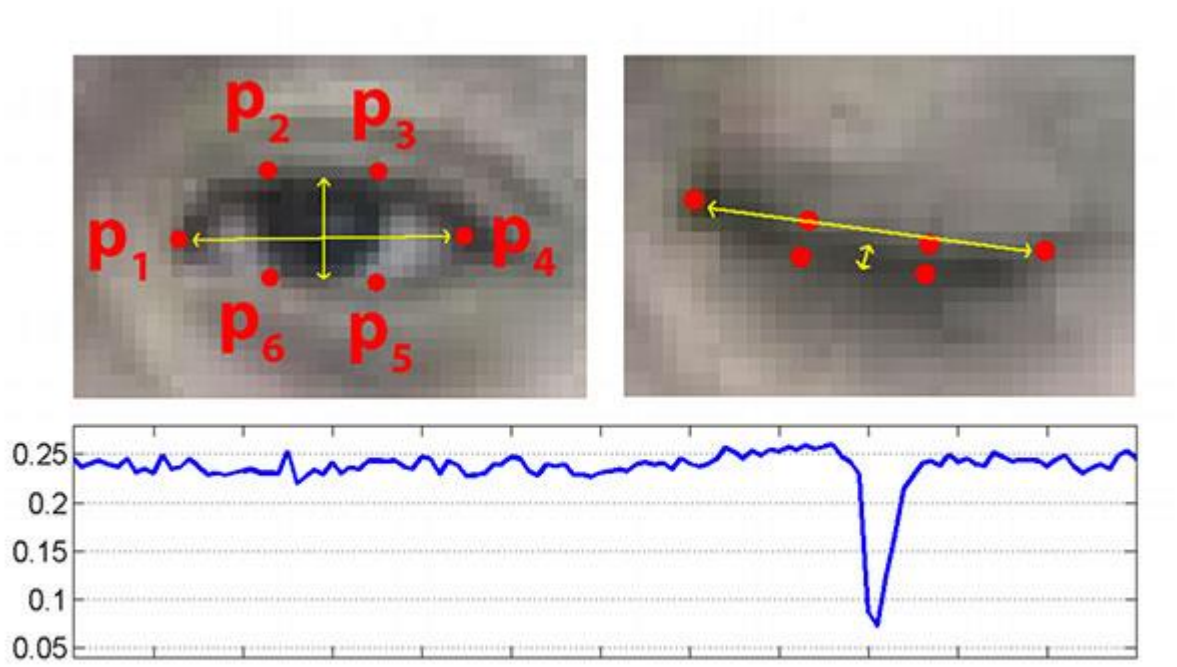
Where p1, …, p6 are 2D facial landmark locations.

The numerator of this equation computes the distance between the vertical eye landmarks while the denominator computes the distance between horizontal eye landmarks, weighting the denominator appropriately since there is only one set of horizontal points but two sets of vertical points.

Well, as we'll find out, the eye aspect ratio is approximately constant while the eye is open, but will rapidly fall to zero when a blink is taking place.

Using this simple equation, we can avoid image processing techniques and simply rely on the ratio of eye landmark distances to determine if a person is blinking.

To make this clearer, consider the following figure from Soukupová and Čech:



On the top-left we have an eye that is fully open — the eye aspect ratio here would be large(r) and relatively constant over time.

However, once the person blinks (top-right) the eye aspect ratio decreases dramatically, approaching zero.

The bottom figure plots a graph of the eye aspect ratio over time for a video clip. As we can see, the eye aspect ratio is constant, then rapidly drops close to zero, then increases again, indicating a single blink has taken place. If the eye aspect ratio falls below this threshold, we'll start counting the number of frames the person has closed their eyes for. If the number of frames the person has closed their eyes in exceeds, we understand that this is more than a blink.

Conclusion:

We could also have used traditional image processing methods for computing blinks which typically involve some combination of:

1. Eye localization.
2. Thresholding to find the whites of the eyes.
3. Determining if the "white" region of the eyes disappears for a period of time (indicating a blink).

Another option could be to use Adaboost Algorithm and Classifier Training. Adaboost algorithm is a kind of boosting algorithm proposed by Freund and Schapire , which selects some weak classifiers and integrates them into strong classifiers automatically.

The eye aspect ratio is instead a much more elegant solution that involves a very simple calculation based on the ratio of distances between facial landmarks of the eyes, thus would be better to use.