**Task 2: Supervised Learning - Synopsis**

**Computer Vision - Drowsy or Not?**

Performance should be measured according to accuracy of prediction

## Pre-processing steps required on the dataset:

- Resizing the images to a standard size so that all images have uniformity.
    - Some images captured by a camera and fed to our AI algorithm vary in size, therefore, we should establish a base size for all images fed into our AI algorithm for better accuracy.
- Segmentation/Thresholding:
    - Removing background from the image to better detect the face.
- Image Filtering:  Low pass filter
    - Basically blurring the image to remove noise
        - Averaging
        - Gaussian Blur
        - Median Blurring
        - Bilateral Filtering
- Making the image grayscale for easier edge detection.

## Techniques to detect the eye from the images

We start by capturing images at regular intervals and then use that data to start detecting whether the driver is drowsy or not.

Step 1: Detecting the face from the given image.

- Viola and Jones approach - Haar Cascade Classifier
- CNN based approaches
- **Other processes:**
    - Circular Hough transform
    - Iris Detection using Entropy An Automatic Eye Detection Method for Gray Intensity Facial Images

Step 2 : Using facial landmark localization to detect the eyes in the face.

Once we obtain the face bounding box (i.e., the (x, y)-coordinates of the face in the image), we can go on to detect the eyes in the face.

This can be done by using the facial landmark detector included in the dlib library or using mediapipe.

The method involving dlib library starts by using:

1. A training set of labeled facial landmarks on an image. These images are manually labeled, specifying specific (x, y)-coordinates of regions surrounding each facial structure.
2. Priors, of more specifically, the probability on distance between pairs of input pixels.

**Techniques to detect drowsiness from the image(Classification into Open/Closed):**

- **Utilizing the given dataset:**

  We can infer the state of the eye opening from a single image of the situation by correlation matching with open and closed eye templates i.e. using our labeled datasets to train the model.

- **Conventional Process: (By checking whether the white region disappears)**

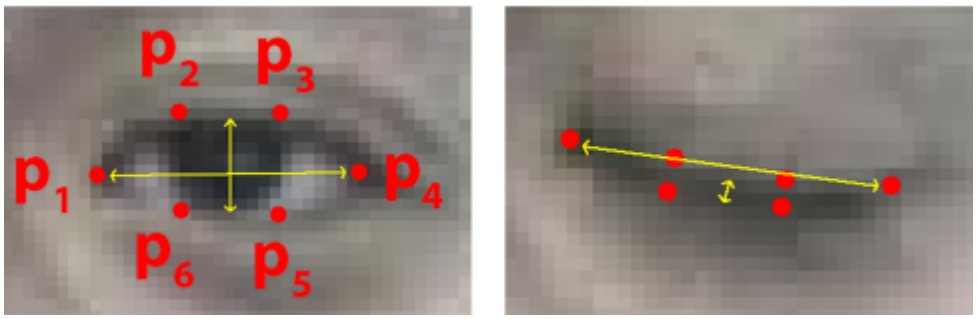
  Step 1: Localize the eye (as mentioned above)

  Step 2: Threshold the image to find the white region of the eye.

  Step 3: Then determining whether the white part is present, if not present then the eye is closed.

- **Using Eye Aspect Ratio (EAR):**

From the landmarks detected in the image, we derive the eye aspect ratio (EAR) that is used as an estimate of the eye opening state. Since the perframe EAR may not necessarily recognize the eye blinks correctly, a classifier that takes a larger temporal window of a frame into account is trained.




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

where p1, . . . , p6 are the 2D landmark locations depicted above.

The eye aspect ratio is approximately constant while the eye is open, but will rapidly fall to zero when a blink is taking place.

Advantages:

- Head pose insensitive
- Partially person insensitive.
- Works real time.

**Conclusion:**

**Related Work:**

Answer the following questions for each piece of related work that addresses the same or a similar problem. What is their problem and method? How is your problem and method different? Why is your problem and method better?

**Bibliography:**


- An Introductory Guide to Computer Vision
- OpenCV: Smoothing Images
- OpenCV: Cascade Classifier
- Real-Time Eye Blink Detection using Facial Landmarks
- Face Landmark Detection
- Detect eyes, nose, lips, and jaw with dlib, OpenCV, and Python