Drowsiness detection System

Table of Content:

- **Problem statement:**
  Drowsy driving is a significant cause of road accidents worldwide, leading to injuries, fatalities, and property damage. Drivers who operate vehicles while fatigued or drowsy experience impaired reaction times, reduced attention, and decreased cognitive abilities, increasing the risk of accidents. To address this issue and improve road safety, a drowsiness detection system is proposed

Technologies used:

Software Technologies:

Python: Python is a popular programming language for developing machine learning, computer vision, and web applications. It offers a wide range of libraries and frameworks suitable for building a drowsiness detection system.

OpenCV (Open Source Computer Vision Library): OpenCV is widely used for image processing tasks such as face detection, eye tracking, and facial landmark detection, which are essential for monitoring driver behavior.

Dlib: Dlib is one of the most powerful and easy-to-go open-source library consisting of machine learning library/algorithms and various tools for creating software. It helps to analyze the object/face using the functions called HOG (Histogram of oriented gradients) and CNN (Convolutional Neural Networks).

Streamlit: Stream lit is a web application framework for building interactive web applications with Python. It allows you to create user interfaces for data-driven applications, making it suitable for developing a GUI (graphical user interface) for the drowsiness detection system.

NumPy: NumPy is library for numerical computing and data analysis in Python. They provide support for handling arrays, matrices which are useful for data manipulation.

Hardware Technologies:

Web Camera: A built-in or external webcam is required for capturing images or video footage of the driver's face. Microphone (Optional): A microphone may be used to capture audio signals, which can be analyzed for additional cues of drowsiness such as yawning or changes in speech patterns.

Processing Unit: A laptop with a modern processor (CPU) is necessary to perform real-time image processing and analysis tasks. A multi-core processor with adequate processing power is preferable for handling computationally intensive tasks.

Memory (RAM): Sufficient RAM is needed to store and manipulate data during image processing and machine learning tasks. Higher RAM capacity allows for smoother operation and better performance, especially when dealing with large datasets.

Let's now understand how our algorithm works step by step.

Step 1 – Take Image as Input from a Camera

With a webcam, we will take images as input. So to access the webcam, we made an infinite loop that will capture each frame. We use the method provided by OpenCV, to access the camera and set the capture object (cap)

Step 2 – Detect Face in the Image and Create a Region of Interest (ROI)

To detect the face in the image, we need to first convert the image into grayscale as the OpenCV algorithm for object detection takes gray images in the input. We don't need color information to detect the objects. We will be using haar cascade classifier to detect faces. returns an array of detections with x,y coordinates, and height, the width of the boundary box of the object. Now we can iterate over the faces and draw boundary boxes for each face.

Step 3 – Detect the eyes from ROI and feed it to the classifier

The same procedure to detect faces is used to detect eyes. First, we set the cascade classifier for eyes in **leye** and **reye** respectively then detect the eyes using **left_eye**. Now we need to extract only the eyes data from the full image. This can be achieved by extracting the boundary box of the eye and then we can pull out the eye image from the frame . **l_eye** only contains the image data of the eye. This will be fed into our CNN classifier which will predict if eyes are open or closed. Similarly, we will be extracting the right eye into **r_eye**.

Step 4 – Classifier will Categorize whether Eyes are Open or Closed

We are using CNN classifier for predicting the eye status. To feed our image into the model, we need to perform certain operations because the model needs the correct dimensions to start with

Step 5 – Calculate Score to Check whether Person is Drowsy

The score is basically a value we will use to determine how long the person has closed his eyes. So if both eyes are closed, we will keep on increasing score and when eyes are open, we decrease the score. if increase the score it means eyes are closed and score increase. We are drawing the result on the screen. A threshold is defined.for example If the score is greater than the define value This is when we beep the alarm using **sound.play()**

Conclusion:

The Drowsiness Detection System is a valuable tool for enhancing road safety by monitoring driver beahviour and detection signs of drowsiness in real-time. By leaveraging computer vision techniques, machine learning algorithms, and interative user interfaces, the system provides timely alert to prevent accidents caused by drowsy driving