# Driver Drowsiness Detection System

**Team ID:CU\_CP\_TEAM\_6264**

## Introduction:

This project focuses on building a Driver Drowsiness Detection System that monitors a driver's eye status using a webcam and alerts them if they appear drowsy. We utilize OpenCV for image capture and preprocessing, while a Convolutional Neural Network (CNN) model classifies whether the driver's eyes are 'Open' or 'Closed.' If drowsiness is detected, an alarm is triggered to alert the driver.

## Project Overview:

### Steps in the Detection Process:

* Image Capture: Capture the image using a webcam.
* Face Detection: Detect the face in the captured image and create a Region of Interest (ROI).
* Eye Detection: Detect the eyes from the ROI and feed them into the classifier.
* Eye Classification: The classifier categorizes whether the eyes are open or closed.
* Drowsiness Score Calculation: Calculate a score to determine if the driver is drowsy based on how long their eyes remain closed.

## CNN Model:

The Convolutional Neural Network (CNN) architecture consists of the following layers:  
- Convolutional Layers:  
 - 32 nodes, kernel size 3  
 - 32 nodes, kernel size 3  
 - 64 nodes, kernel size 3  
- Fully Connected Layers:  
 - 128 nodes  
 - Output layer: 2 nodes (with Softmax activation for classification)

### Activation Function:

- ReLU: Used in all layers except the output layer.  
- Softmax: Used in the output layer to classify the eyes as either 'Open' or 'Closed.'

## Project Prerequisites:

### Required Hardware:

A webcam for image capture.

### Required Libraries:

Ensure Python (version 3.6 recommended) is installed on your system. Then, install the following libraries using pip:  
```  
pip install opencv-python  
pip install tensorflow  
pip install keras  
pip install pygame  
```

### Other Project Files:

* Haar Cascade Files: Located in the 'haar cascade files' folder, these XML files are necessary for detecting faces and eyes.
* Model File: The 'models' folder contains the pre-trained CNN model cnnCat2.h5.
* Alarm Sound: The audio clip alarm.wav will play when drowsiness is detected.
* Python Files:
* - Model.py: The file used to build and train the CNN model.
* - Drowsiness detection.py: The main file that executes the driver drowsiness detection system.

## How the Algorithm Works:

1. Step 1 – Image Capture: The webcam captures images in real-time using cv2.VideoCapture(0) and processes each frame.
2. Step 2 – Face Detection: The image is converted to grayscale for face detection using a Haar Cascade Classifier. The faces are detected using detectMultiScale(), and boundary boxes are drawn around the detected faces.
3. Step 3 – Eye Detection: Similar to face detection, eyes are detected within the ROI using another cascade classifier. The eye images are extracted and passed to the CNN model for classification.
4. Step 4 – Eye Classification: The extracted eye images are preprocessed by resizing to 24x24 pixels, normalizing the values, and then passed into the CNN model (cnnCat2.h5). The model predicts whether the eyes are open or closed.
5. Step 5 – Drowsiness Detection: A score is calculated based on the status of both eyes. If both eyes are closed for an extended period, the score increases, indicating drowsiness. If the score exceeds a threshold, an alarm is triggered using the Pygame library.

## Execution Instructions:

### Running the Detection System

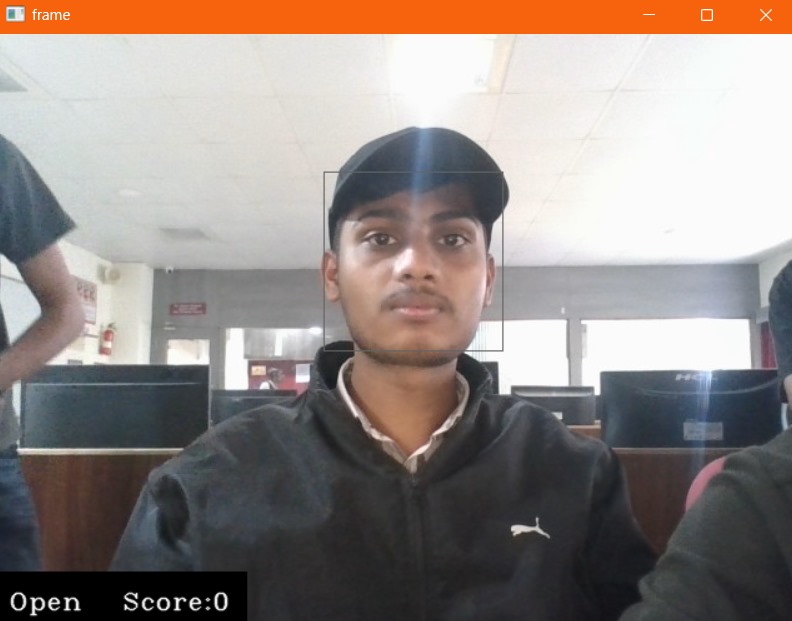
1. Open the command prompt and navigate to the directory where the main file drowsiness detection.py is located.  
2. Run the script using the following command:  
```  
python drowsiness detection.py  
```  
The system will access the webcam and start detecting drowsiness. The real-time status will be displayed on the screen.

## Summary:

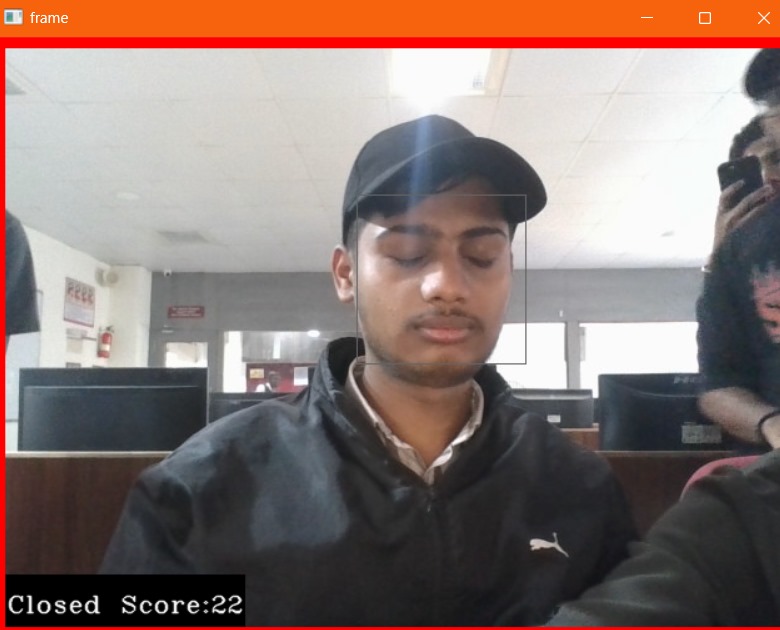
This Python project implements a Driver Drowsiness Detection System using OpenCV and a CNN model to detect whether the driver’s eyes are open or closed. When the eyes are detected as closed for a prolonged time, an alert sound is played to prevent potential accidents. This system can be implemented in vehicles or other applications to enhance driver safety.

## Results:

Set up a camera to monitor a stream for faces. Upon detecting a face, apply facial landmark detection to extract the eye regions.



Compute the eye aspect ratio (EAR) to check if eyes are closed. If the EAR indicates prolonged eye closure, sound an alarm to wake up the driver



## Future Enhancements:

* Improve the detection accuracy by training on a larger dataset.
* Implement real-time monitoring for multiple people.
* Add functionalities to detect other signs of drowsiness like head tilting or yawning.

## Acknowledgments:

- OpenCV Documentation: https://opencv.org/  
- Keras Documentation: https://keras.io/  
- TensorFlow Documentation: https://www.tensorflow.org/