# TABLE OF CONTENTS

# INTRODUCTION

According to the World Health Organization (WHO), more than 25% of road accidents occur each year. Technological advancements provide opportunities to introduce advanced solutions to improve our standard of living. The National Highway Traffic Safety Administration (NHTSA) reports around 100,000 crashes per year involving drowsy driving, though the actual figure is likely higher. Facial expressions can offer valuable insights into various physiological conditions of the body.

There are numerous algorithms and techniques available for face detection, which is the fundamental step in the process. Drowsiness in humans is characterized by specific movements and facial expressions, such as the eyes beginning to shut. To address this global problem, a solution is proposed to track the driver's eyes, detect drowsiness, and classify the driver as drowsy. In real-time applications, a camera is mounted on the car's dashboard to capture the driver's face.

The proposed drowsy driver detection system utilizes the Dlib model, which is trained to detect 68 facial landmarks. The system extracts drowsiness features from these landmarks and alerts the driver if drowsiness is detected. The implementation is done using Python and the Dlib model. Dlib's shape detector is used to map the coordinates of facial landmarks in the input video, and drowsiness is detected by monitoring the aspect ratios of the eyes.

## ABSTRACT

A significant number of reported accidents are caused by driver inattention or drowsiness. Fatigue and microsleeping while driving often lead to major accidents. Detecting and indicating driver fatigue at an early stage can help prevent severe scenarios. Most conventional methods for detecting sleepiness are based on behavioral factors, some of which are intrusive, distracting to drivers, or require expensive sensors. This paper presents a Driver Drowsiness Detection System using Python and Dlib models, aiming to reduce the number of road accidents. The proposed system is non-intrusive, as it does not require physical contact with the driver, making it easy to implement.

The system detects facial landmarks and computes the Eye Aspect Ratio (EAR) to determine driver drowsiness based on adaptive thresholding. Python, particularly OpenCV, is utilized to access webcam-related functions, allowing the application to detect faces through the webcam and analyze the drowsiness status of the person in the camera feed.

## PROBLEM STATEMENT

The main objective of this project is to develop an application capable of detecting faces using the webcam and determining the sleepiness of the person using the Eye Aspect Ratio (EAR) obtained from Dlib's 68 facial landmarks detection.

Python is chosen as the programming language due to its suitability for machine learning applications.

OpenCV is utilized to access webcam functions.

## MOTIVATION

The main motivation for this project was to enhance understanding of Python programming and its practical application in real-life scenarios.

Exploring new areas of computing not covered in lectures was also appealing.

The opportunity to automate tasks such as face detection, drowsiness detection using machine learning, and utilizing databases and web scraping further fueled the motivation to undertake this project.

Machine learning is a rapidly evolving field, offering numerous solutions, algorithms, and unexplored applications.

The concept of machine learning is to allow computer systems to constantly improve their performance in certain tasks through the use of data and statistical techniques. Rather than programming a computer specifically to improve performance.

# TOOLS AND SOFTWARE USED

The tools and software used to accomplish this project includes:

- Programming Language:

Python: Python's popularity and extensive usage in various fields stem from its versatility, simplicity, and strong community support. Its wide range of libraries and frameworks, combined with its readability and beginner-friendly nature, make Python an excellent choice for both beginners and experienced programmers.

- Coding Environment:

Visual Studio Code:Visual Studio Code, also known as VS Code, is a widely used source-code editor developed by Microsoft for Windows, Linux, and macOS. It offers a range of features and functionalities that enhance the coding experience, including debugging support, syntax highlighting, code completion, and Git integration. With its customizable interface, users can personalize their coding environment by choosing themes, configuring keyboard shortcuts, and installing extensions.

- Modules Used:

TKINTER: Tkinter is a widely used Python framework for creating Graphical User Interfaces (GUIs). It is included in all standard Python distributions and serves as the primary GUI framework in the Python standard library. Tkinter provides an object-oriented interface to the Tk toolkit, which consists of a collection of cross-platform graphical control elements or widgets for building application interfaces.

PIL: (Python Imaging Library) is the go-to image processing package for the Python language. It offers lightweight image processing tools that facilitate tasks such as image editing, creation, and saving. PIL is widely used in various applications that require image manipulation and processing.

OPENCV: OpenCV (Open Source Computer Vision) is a powerful open-source library primarily used for image processing and computer vision tasks. It provides a broad range of functionalities, including face detection, object tracking, landmark detection, and more. OpenCV supports multiple programming languages, including Python, Java, and C++, making it a versatile tool for computer vision applications.

NUMPY: NumPy is a fundamental Python library for performing mathematical operations on arrays. It introduces efficient data structures for working with arrays and matrices, enabling high-performance calculations. NumPy offers a vast collection of high-level mathematical functions designed to operate on arrays and matrices, making it essential for numerical computations.

IMUTILS: Imutils is a convenient library that simplifies basic image processing tasks, such as translation, rotation, resizing, edge detection, contour sorting, and more. It is specifically designed to work seamlessly with OpenCV, enhancing the ease of use and productivity when working with image processing algorithms.

PYGAME: Pygame is a cross-platform set of Python modules designed for game development. It includes graphics and sound libraries tailored for use with the Python programming language. Pygame is leveraged in this project to utilize the mixer library for incorporating an alarm sound during drowsiness detection.

These libraries and frameworks play vital roles in various aspects of the project, enabling GUI creation, image processing, computer vision tasks, mathematical operations, image manipulation, and sound integration.

# METHODOLOGY

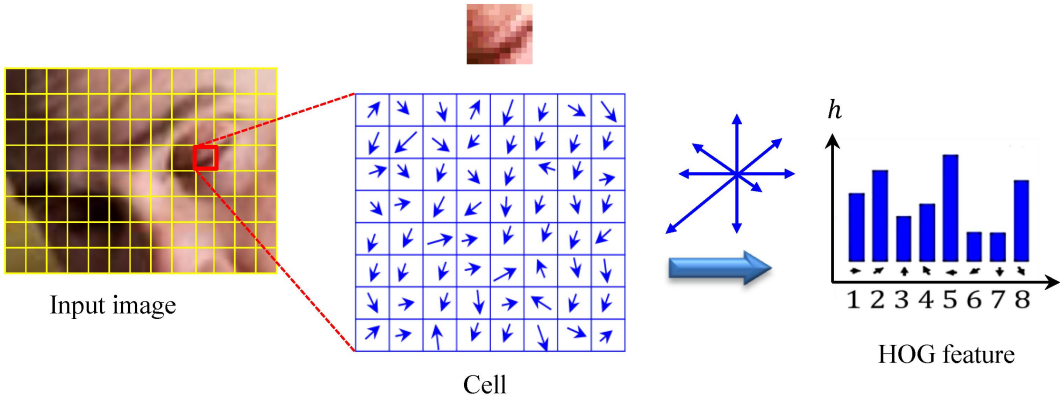
How does this project works:

- FACE DETECTION:

The camera detects and locates faces in images, whether the face is alone or in a crowd. The image can show the person looking straight ahead or in profile.
Face Detection using Dlib HoG

The "get_frontal_face_detector" function in Dlib returns a pre-trained face detector using the HOG (Histogram of Oriented Gradients) + Linear SVM method.
Dlib's HOG + Linear SVM face detector is fast and efficient. However, it is not invariant to changes in rotation and viewing angle due to the nature of the HOG descriptor.

Step1: Divide the image into small cells



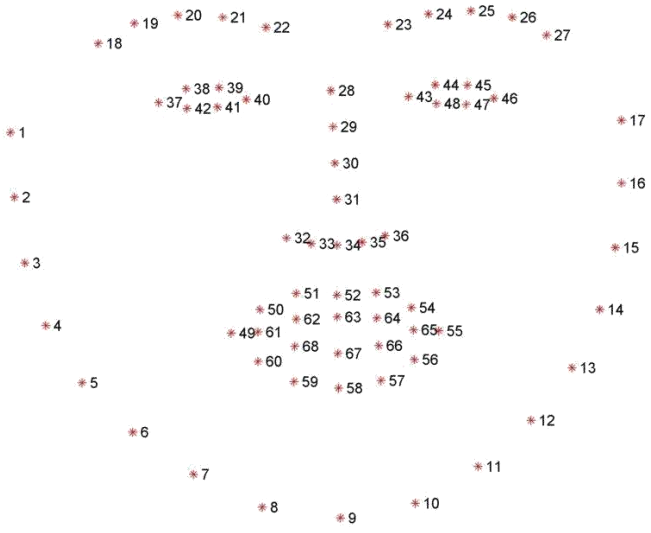

Step2:Histograms are computed for each cell.

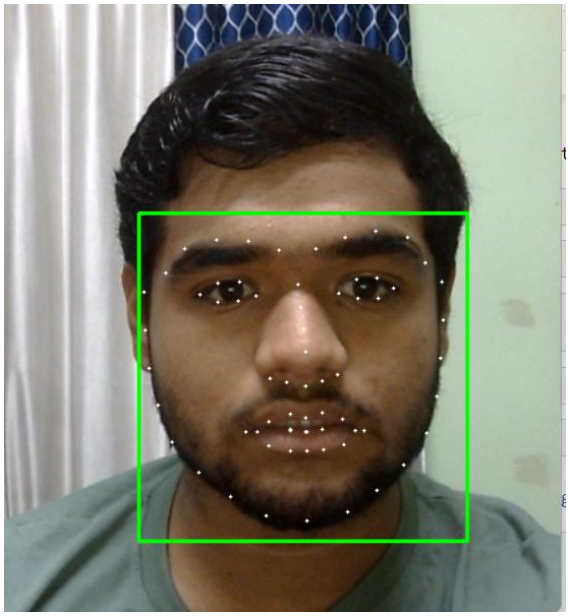
Step3:All histograms are combined to form a feature vector, which is unique for each face.

However, HOG-based face detection is not suitable for faces at odd angles. It works best with straight and front-facing faces. It is commonly used for detecting faces in scanned documents such as driver's licenses and passports but may not be the ideal choice for real-time video applications.

## IMPLEMENTING FACIAL LANDMARKS:

Facial landmark detection is performed using pre-trained models provided by Dlib. The models estimate the location of 68 coordinates (x, y) that represent the facial points on a person's face. These points correspond to regions such as the eyes, nose, and lips.



The facial landmarks are obtained from a pre-trained model trained on the iBUG300W dataset. Facial landmark detection is a crucial computer vision task that involves predicting key points representing different facial features. It has applications in head pose estimation, gaze direction identification, facial gesture detection, and face swapping.

The detected landmarks are then used to calculate the Eye Aspect Ratio (EAR), which indicates whether the eyes are open or closed.

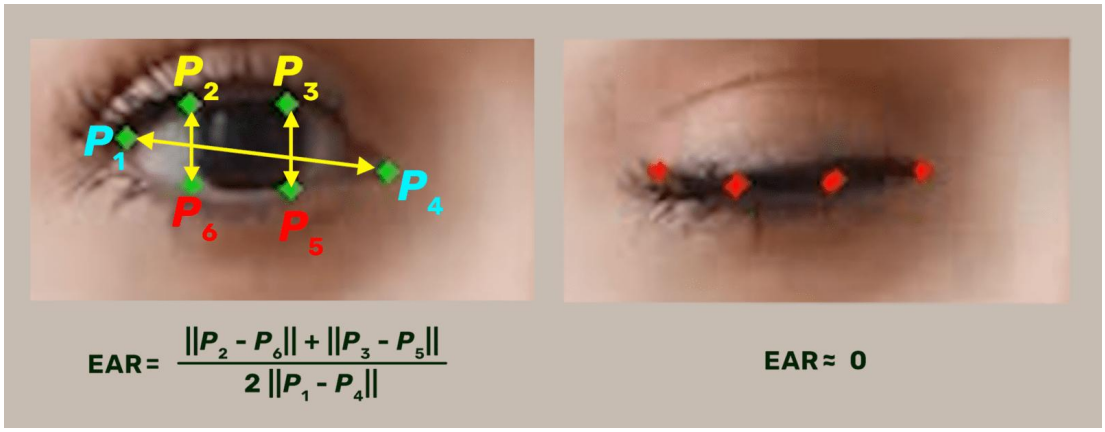
- CALCULATING EYE ASPECT RATIO:

Each eye is represented by 6 landmark points.
The EAR for a single eye is calculated using the formula:

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

EAR = ||p2-p6|| / (0.5 * (||p1-p5|| + ||p3-p4||))
Here, ||p2-p6|| represents the distance between points p2 and p6.

A higher EAR value indicates that the eye is more widely open. A minimum EAR value is decided, and if the calculated EAR is below this threshold (e.g., 0.25), it indicates closed eyes.




The average eye aspect ratio is 0.339 and 0.141 when the eyes opened and closed, respectively. In the project, I have taken the condition that if EAR is  less than 0.25  that means closed eyes else opened eyes.

If the eyes are found closed for a certain number of frames in the webcam feed, it suggests that the person's eyes have been closed for a period of time, indicating drowsiness.

In such cases, an alarm sound is triggered to alert the person. This feature is particularly useful for driving scenarios where drowsiness can be dangerous.

# CONCLUSION

## SUMMARY:

- The Dlib library's pre-trained 68 facial landmark detector is utilized in this project.

- The face detection is based on the Histogram of Oriented Gradients (HOG) method.

- The Eye Aspect Ratio (EAR) is used as a quantitative metric to monitor driver drowsiness.

- Real-time detection results may be affected by lighting conditions.

- The drowsiness detection system effectively differentiates between normal eye blinks and drowsiness, preventing drivers from falling into a state of sleepiness while driving.

- The system performs well regardless of whether the driver wears spectacles and under low-light conditions.

- During monitoring, the system can accurately determine if the eyes are closed or open, issuing a warning signal if the eyes have been closed for an extended period.

## FUTURE WORKS:

The technology can be used for a variety of purposes that includes:

- Real-Time Driver Monitoring: Develop real-time driver monitoring systems that continuously analyze multiple facial and physiological cues to detect drowsiness. This can involve using advanced machine learning algorithms to fuse data from various sensors and provide accurate and timely alerts to drivers.

- Context-Aware Drowsiness Detection: Incorporate contextual information such as road conditions, driving behavior, and time of day to improve the accuracy of drowsiness detection. Difference between genuine cases.

- Data Collection and Analysis: Continuously collect and analyze data from drowsiness detection systems to improve their performance and understand the patterns and risk factors associated with drowsiness-related incidents. This can involve building large-scale datasets and leveraging data analytics techniques to gain valuable insights.

# REFERENCE:


- Websites:

- Drowsiness Detection: A Comprehensive Review. (2018). Advances in Science, Technology, and Engineering Systems Journal, 3(4), 107-113. doi: 10.25046/aj030413

- OpenCV-Python Tutorials: The OpenCV-Python Tutorials on the OpenCV website offer step-by-step explanations and code examples for using OpenCV with Python. The tutorials cover a wide range of topics, from basic image processing to advanced computer vision algorithms. You can find the tutorials at: https://docs.opencv.org/master/d6/d00/tutorial_py_root.html

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. Deep Learning Book. Retrieved from http://www.deeplearningbook.org/. This online resource is an interactive version of the book mentioned

- PythonGuides (https://www.pythonguides.com): PythonGuides is an online platform that offers tutorials, articles, and code examples specifically focused on Python programming.

- Books Referenced:

- 'Python Crash Course' by Eric Matthes.