



# **DRIVER DROWSINESS DETECTION SYSTEM USING MACHINE LEARNING**



**A PROJECT REPORT**

*submitted by*

<b>PRAKASH S</b>	<b>(622021243043)</b>
<b>RATHISWARAN K</b>	<b>(622021243045)</b>
<b>HARIKRISHNA K</b>	<b>(622021243301)</b>

*in partial fulfilment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

**IN**

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**PAAVAI COLLEGE OF ENGINEERING**

**PACHAL, NAMAKKAL – 637 018**

**ANNA UNIVERSITY: CHENNAI 600 025**

**MAY – 2025**

# **ANNA UNIVERSITY: CHENNAI - 600 025**

## **BONOFIDE CERTIFICATE**

Certified that this project report “**DRIVER DROWSINESS DETECTION SYSTEM USING MACHINE LEARNING**” is the Bonafide work of “**PRAKASH S (622021243043), RATHISWARAN K (622021243045), HARIKRISHNA K (622021243301)**,” who carried out the project work under my supervision.

### **SIGNATURE**

**Mr.S. PRABHU, M .Tech.,(Ph.D).,**

**SUPERVISOR,**

**PROFESSOR,**

**DEPARTMENT OF AI&DS,**

**PAAVAI COLLEGE OF ENGINEERING**

### **SIGNATURE**

**Mr.S. PRABHU, M .Tech.,(Ph.D).,**

**HEAD OF THE DEPARTEMENT,**

**PROFESSOR,**

**DEPATEMENT OF AI&DS,**

**PAAVAI COLLEGE OF ENGINEERING**

Certified that candidate was examined in the Anna University project work viva – voce examination held on \_\_\_\_\_

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

We express our profound gratitude with pleasure too ur most respected **Chairman, Shri. CA. N. V. NATARAJAN, B.Com., F.C.A.** and to our beloved **Correspondent Smt. N. MANGAINATARAJAN, M.Sc.**, for giving motivation and providing all necessary facilities for the successful completion of this project.

It is our privilege to thank our beloved Director - Admin **Dr. K. K. Ramasamy, M.E, Ph.D.**, and our principal **Dr. V. Hariharan, M.E, Ph.D.**, for their moral support and deeds in bringing out this project successfully.

We extend our gratefulness to **Mr. S. PRABHU M. Tech (Ph.D.)**, Professor, Head of the Department. Department of Artificial Intelligence and Data Science for his encouragement and constant support in successfully completion of project.

We convey out thank to **Mr. S. PRABHU M. Tech (Ph.D.)**, Professor, Head of the Department. Department of Artificial Intelligence and Data Science our Supervisor for being more informative and providing suggestions regarding this project all the time.

We would like to express our sincere thanks and heartiest gratitude to our project coordinator **Mr. R. ASKOK KUMAR M. Tech** Assistant Professor, her meticulous guidance and timely help during project work.

We would like to extend ours incere thanks to all **our Department faculty members and to our parents** for their advice and encouragement to do the project work with full interest and enthusiasm.

## ABSTRACT

Driver fatigue caused road accidents are a serious issue globally resulting in thousands of deaths annually, research has shown that drowsy driving reduces reaction time lowers consciousness and heightens the risk of accidents. Inorder to counter this serious problem a driver drowsiness detection system based on machine learning is implemented to track drivers in real-time and warn them before they doze off while driving. This system takes advantage of computer vision and deep learning algorithms to monitor facial expressions such as eye closure blinking rate, yawning and head orientation other sensor-based data like heart rate and steering behavior can be fused to achieve more precise predictions by using sophisticated machine learning algorithms.

Such as ‘convolutional neural networks’(CNNs) and ‘support vector machines’(SVMs). The system is capable of distinguishing alert from drowsy conditions with great precision the suggested solution works by obtaining a live video feed of the face of the driver computing major facial landmarks and measuring drowsiness levels in the event that the system picks up signs of tiredness it initiates alerts in the form of audio alarms steering wheel vibrations or visual alerts to keep the driver alert this project is intended to enhance road safety through minimizing accidents resulting from drowsy driving the system is conceived to be efficient cost-saving and able to perform in real-time thus fitting into contemporary cars and fleet management systems.

## **PROBLEM STATEMENT**

The driver drowsiness detection system using machine learning addresses the critical issue of fatigue-related accidents on the road. By leveraging advanced algorithms to analyze visual cues such as eye movements and facial expressions, the system aims to identify signs of drowsiness in real-time. The challenge lies in accurately detecting subtle indicators of fatigue before they lead to dangerous driving situations, ultimately enhancing road safety and reducing the risk of accidents.

The system employs a Convolutional Neural Network (CNN) to process images captured from a webcam, focusing on the driver's eyes to determine their state open or closed. Key components include image acquisition through OpenCV, face and eye detection using Haar cascade classifiers, and drowsiness classification via the CNN model. As the system monitors the driver's eye state, it maintains a drowsiness score that increases when the eyes are closed for extended periods. If this score surpasses a certain threshold, the system triggers alerts to prompt the driver to take necessary actions, thereby aiming to prevent accidents caused by drowsy driving. This innovative approach not only enhances driver awareness but also contributes to the development of safer automotive technologies.

# TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	<b>ABSTRACT</b>	<b>v</b>
	<b>LIST OF ABBREVIATIONS</b>	<b>x</b>
	<b>LIST OF FIGURES</b>	<b>xi</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Background	2
	1.1.1 Leveraging DDDS using ML	2
	1.1.2 Role of Technology	3
	1.1.3 Machine Learning Approaches	4
	1.2 Objectives	5
	1.3 Scope of the Project	6
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>10</b>
	2.1 Physicological Monitoring	10
	2.2 Behavioral Monitoring	10
	2.3 Vechicle Based Monitoring	10
	2.4 Technological Approaches	10
	2.5 Future Direction	11
<b>3</b>	<b>SYSTEM ANALYSIS</b>	<b>13</b>
	3.1 Functional Requirements	13
	3.1.1 Rea-time Video Capture	13
	3.1.2 Face Detection	13
	3.1.3 Facial Landmark Detection	13
	3.1.4 Eye State Analysis	14
	3.1.5 Mouth and Yawning detection	14
	3.1.6 Head Posture Estimation	14
	3.1.7 Drowsiness Classification	14

	3.1.8 Alert Generation	15
	3.1.9 Logging and Data Recording	15
	3.1.10 System Configuration	15
	3.2 Non-Functional Requirements	15
	3.2.1 Performance Requirements	15
	3.2.2 Accuracy	16
	3.2.3 Usability	16
	3.2.4 Scalability	16
	3.2.5 Portability	16
	3.2.6 Reliability	16
	3.2.7 Maintainability	17
	3.2.8 Security	17
	3.2.9 Robustness	17
	3.2.10 Compliance	17
<b>4</b>	<b>FESIBILITY STUDEY</b>	<b>19</b>
	4.1 Technical Feasibility	19
	4.1.1 Sensor Technologies	20
	4.1.2 Data Processing and Algorithms	20
	4.1.3 Integration and Human-Machine Interface	21
	4.1.4 Scalability and Performance	21
	4.2 Economic Feasibility	21
	4.2.1 Costs	21
	4.2.2 Benefits and Returns on Investment	22
	4.2.3 Market Dynamics and Growth	23
	4.2.4 Factors Influencing Economic Feasibility	23
	4.3 Operational Feasibility	24
	4.3.1 System Integrstion	24
	4.3.2 User Experience and Acceptance	24
	4.3.3 Environmental and Operational Conditions	24

	4.3.4 Maintenance and Support	25
	4.3.5 Scalability and Deployment	25
<b>5</b>	<b>SYSTEM SPECIFICATION</b>	<b>27</b>
	5.1 Hardware Specification	27
	5.2 Software Specification	28
	5.3 Functional Specification	29
	5.4 Interface Specification	30
	5.5 Network Specification	30
<b>6</b>	<b>SYSTEM DESIGN</b>	<b>32</b>
	6.1 Existing System	32
	6.2 Problem Statement	32
	6.3 Proposed System	32
	6.4 Block Diagram	33
	6.5 Flow Diagram	34
	6.6 System Architecture	35
	6.7 Working Principle	35
<b>7</b>	<b>PROJECT DESCRIPTION</b>	<b>38</b>
	7.1 Introduction	38
	7.1.1 Gathering Data	39
	7.1.2 Data pre-processing	40
	7.1.3 Researching the model for best datatype	41
	7.2 Supervised Learning	41
	7.2.1 Classification	42
	7.2.2 Regression	43
	7.2.3 Clustering	44
	7.3 Training and testing the modal data	45
	7.4 Evaluation	47
	7.4.1 Eye Aspect Ratio	48
	7.4.2 Facial Landmark	49



	7.4.3 Mouth Aspect Ratio	50
	7.5 HOG Algorithm	50
	7.5.1 HOG Feature Descriptor	52
	7.5.2 Process of Calculating the HOG	53
	7.6 Open CV	64
<b>8</b>	<b>CONCLUSION &amp; FUTURE ENHANCEMENT</b>	<b>67</b>
	8.1 Conclusion	67
	8.2 Future Enhancement	68
	<b>APPENDIX</b>	
<b>9</b>	<b>REFERENCES</b>	<b>78</b>

## **LIST OF ABBREVIATIONS**

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CNN	Convolution Neural Network
SVM	Support Vector Machine
IOT	Internet of Things
DDDS	Driver Drowsiness Detection System
EAR	Eye Aspect Ratio
MAR	Mouth Aspect Ratio
RNN	Recurrent Neural Network
HOG	Histogram of Oriented Gradients
OpenCV	Open Source Computer Vision Library

## LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURES	PAGENO
6.4	SCHEMA FOR DROWSINESS DETECTION SYSTEM	26
6.4	SCHEMA FOR DISTRACTION STATE RECOGNITION	26
6.5	SYSTEM FLOW MAP	27
6.6	SYSTEM ARCHITECTURE	27
6.7	WORKING PRINCIPLE	28
7.2	CLASSIFICATION	33
7.2	REGRESSION	34
7.2	ENSAMPLE METHOD	34
7.2	CLUSTERING	35
7.2	MACHINE LEARNING MODEL	35
7.3	TRAINING AND TESTING DATA SET	36
7.3	TRAINING AND TESTING DATA SET EXECTION	36
7.4	EYE ASPECT RATIO CALCULATON	38
7.4	EAR FORMULA CALCULATION WHILE EYE OPENING AND CLOSING POSITION	38
7.4	FACIAL LANDMARK	39
7.4	MOUTH ASPECT RATIO	39

7.5	VIEW METHOD OF HUMAN vs SYSTEM	40
7.5	HUMAN PERSPECTIVE VIEW	40
7.5	SYSTEM PERSPECTIVE VIEW	41
7.5	EXAMPLE IMAGE 2 FOR HOG ALGORITHM	42
7.5	CALCULATING GRADIENTS FOR HOG ALGORITHM	43
7.5	PIXEL MATRIX FOR HOG ALGORITHM	43
7.5	PYTHAGORAS THEOREM	44
7.5	APPLYING GRID USING HOG ALGORITHM	47
7.5	IMAGE ADDRESSING IN GRID USING HOG ALGORITHM	48
7.5	COMPLETE IMAGE USING HOG ALGORITHM	48
7.5	IMAGE OF APPLYING HOG ALGORITHM BEFORE AND AFTER	49
7.5	FINAL SYSTEM VIEW OF SYSTEM USING GIRD LINES	50

# CHAPTER 1

# CHAPTER 1

## INTRODUCTION

Driver tiredness discovery may be a car security innovation which avoids mishaps when the driver is getting lazy. Different considers have proposed that around 20% of all street mishaps are fatigue-related, up to 50% on certain streets. Driver weakness could be a critical calculate in a expansive number of vehicle mischances . Later measurements assess that yearly 1,200 passing and 76,000 wounds can be credited to weariness related crashes. The improvement of technologies for recognizing or avoiding laziness at the wheel could be a major challenge within the field of accident evasion frameworks. Since of the risk that laziness presents on the street, strategies got to be created for neutralizing its influences. Driver carelessness may be the result of a need of sharpness when driving due to driver tiredness and diversion.

Driver diversion happens when an protest or occasion draws a person's consideration absent from the driving errand. Not at all like driver diversion, driver laziness includes no activating occasion but, instep, is characterized by a dynamic withdrawal of consideration from the street and activity requests. Both driver tiredness and diversion, in any case, might have the same impacts, i.e., diminished driving execution, longer response time, and an expanded hazard of crash inclusion.

Appears the piece chart of in general framework. Based on Procurement of video from the camera that's before driver perform real-time preparing of an approaching video stream in arrange to gather the driver's level of weariness on the off chance that the laziness is Estimated at that point it'll deliver the caution by detecting the eyes. Accidents are inevitable with the growth of the population all around the globe. According to the

World Health Organization (WHO), there are various reasons (Sleeping while driving, text while driving, over speed, and alcoholic consumption) for accidents happening despite the situations, according to the World Health Organization (WHO),

Around 1.35 million deaths occurring during the road accidents annually. Around 22 countries have implemented and amended the laws to bring them into the best-practice line. There have been various researches conducted to mitigate the accidents to serve the lives and the values. A considerable amount of literature shows that the leading cause of accidents is drowsiness and fatigue. Various studies were conducted to identify issues with several areas, including IoT, Machine learning, and Mobile application. However, the accuracy and the efficiency used in the above researches are not significantly finding the target. To overcome the said problem, this study focuses on a hybrid approach to detect drowsiness and fatigues using deep learning; and IoT is used to avoid accidents effectively.

Machine learning has numerous applications in the internet of things (IoT). One of these technological trends is E-care, which brings us information in real-time and continuous patient monitoring with predictive decision making.

## **1.1 Background**

### **1.1.1 Leveraging DDDS using ML**

Driver drowsiness is a significant contributor to road accidents worldwide. According to the World Health Organization (WHO) and various traffic safety agencies, fatigue-related crashes account for a considerable percentage of traffic incidents, many of which result in severe injury or death. Drowsiness impairs a driver's alertness, decision-making ability, and reaction time, making it comparable in danger to driving under the influence of alcohol.

Traditional solutions to combat driver fatigue include public awareness campaigns, physical roadside infrastructure (such as rumble strips), and mandated rest periods for commercial drivers. However, these approaches are often insufficient, especially for individual or non-commercial drivers.

With advancements in artificial intelligence (AI), computer vision, and sensor technologies, more effective solutions have become viable. A Driver Drowsiness Detection System (DDDS) aims to monitor the driver's behavior in real-time and detect early signs of fatigue or microsleep. These systems can analyze facial features (like eye closure, blink rate, yawning), head movement, and even physiological signals to assess alertness levels.

Modern systems typically use machine learning (ML) or deep learning (DL) models, such as Convolutional Neural Networks (CNNs) for image-based detection or Support Vector Machines (SVMs) for classification. Real-time video feeds from in-car cameras are processed to identify visual cues that correlate with drowsiness, enabling timely alerts and interventions. By integrating such intelligent systems into vehicles, the goal is to enhance road safety, reduce accident rates, and potentially save lives.

### **1.1.2 Role of Technology**

Road safety is a critical public health issue, with driver fatigue and drowsiness being among the leading causes of motor vehicle accidents. Studies from organizations such as the National Highway Traffic Safety Administration (NHTSA) and the World Health Organization (WHO) indicate that drowsy driving is responsible for thousands of fatalities and injuries annually. Unlike distractions or intoxication, drowsiness can onset gradually and often goes unnoticed by the driver, making it especially dangerous.

To address this issue more effectively, researchers and automotive engineers have turned to active monitoring systems. A Driver Drowsiness



Detection System (DDDS) is designed to automatically monitor and assess the driver's alertness in real-time and provide alerts or even corrective measures (like activating brakes or steering assist) before an accident occurs.

There are generally three categories of drowsiness detection techniques:

i. Vehicle based metrics

These involve monitoring vehicle behavior such as:

1. Steering wheel movements.
2. Lane deviations.
3. Sudden acceleration or braking patterns

However, these methods can be affected by road conditions or driving styles and are less reliable in varying environments.

ii. Behavioral or Vision-Based Methods

The most promising and non-intrusive approach involves:

1. Facial feature tracking (e.g., eye closure, blink rate, yawning, head tilt, and gaze direction).
2. Using cameras and computer vision techniques to analyze real-time video feeds of the driver's face.

These systems can be implemented using machine learning (ML) and deep learning (DL) techniques, offering scalability and adaptability across various vehicle types.

### **1.1.3 Machine Learning Approaches**

Convolutional Neural Networks (CNNs) are commonly used for image-based feature extraction and classification of drowsy vs. alert states. Support Vector Machines (SVMs), often used for binary classification tasks, can effectively distinguish between drowsy and non-drowsy states based on extracted features like Eye Aspect Ratio (EAR) or Mouth Aspect Ratio (MAR). Hybrid models combine CNNs with SVMs or Recurrent

Neural Networks (RNNs) to improve temporal awareness and overall detection accuracy.

## **1.2 Objectives**

The objectives of this project are as follows:

- i. Develop a real-time system that detects driver drowsiness using computer vision and machine learning techniques.
- ii. Collect and preprocess image or video data capturing key facial features (such as eyes, mouth, and head pose) associated with drowsiness.
- iii. Extract relevant behavioral features (e.g., Eye Aspect Ratio, Mouth Aspect Ratio, blink rate, yawning frequency) using facial landmark detection libraries such as Dlib or MediaPipe.
- iv. Implement and train machine learning models, such as:
  - Convolutional Neural Networks (CNNs) for image-based feature learning and classification.
  - Support Vector Machines (SVMs) for classifying driver states (drowsy vs. alert) using extracted features.
- v. Compare the performance of different models in terms of accuracy, precision, recall, and inference speed in real-time conditions.
- vi. Generate real-time alerts (visual/audio) when signs of drowsiness are detected, ensuring timely warning for the driver.
- vii. Evaluate the system's robustness and generalizability across different lighting conditions, driver postures, and facial variations.
- viii. Ensure non-intrusiveness and user-friendliness, so the system can be integrated into vehicles without causing discomfort or distraction to the driver.

## 1.3 Scope of the Project

The scope of a Driver Drowsiness Detection System project can be defined across several dimensions, including objectives, functionalities, technologies, and potential applications. Here's a detailed outline:

### Project Objectives

- **Safety Enhancement:** Reduce the number of accidents caused by drowsy driving.
- **Real-time Monitoring:** Continuously monitor driver alertness during vehicle operation.
- **Alert Mechanism:** Provide timely alerts to the driver when drowsiness is detected.
- **Data Collection:** Gather data on driving patterns and drowsiness incidents for further analysis.

### Functional Requirements

- **Drowsiness Detection:** Implement algorithms to analyze driver behavior, such as:
  - Eye closure duration
  - Blink rate
  - Head position and movement
- **Alert System:** Develop a multi-modal alert system that may include:
  - Visual alerts (e.g., dashboard lights)
  - Auditory alerts (e.g., beeping sounds)
  - Haptic feedback (e.g., vibrating steering wheel)
- **User Interface:** Create a user-friendly interface for drivers to view their alertness status and receive feedback.

- **Data Logging:** Maintain a log of driving sessions, including drowsiness events and alerts.

## **Technological Scope**

- Hardware Components:
  - Cameras (e.g., infrared or RGB) for facial and eye tracking.
  - Microcontrollers or embedded systems for processing data.
  - Sensors for additional monitoring (e.g., steering wheel sensors).
- Software Components:
  - Machine learning algorithms for detecting drowsiness.
  - Mobile or web application for user interaction and data visualization.
  - Cloud storage for data analysis and long-term tracking.

## **Methodologies**

- Data Collection: Gather data from various drivers under different conditions to train the detection algorithms.
- Algorithm Development: Use techniques such as computer vision and machine learning to develop robust drowsiness detection models.
- Testing and Validation: Conduct extensive testing in real-world scenarios to validate the system's effectiveness and reliability.

## **Potential Applications**

- Personal Vehicles: Integration into consumer cars to enhance driver safety.
- Commercial Fleets: Implementation in trucks and buses to monitor driver alertness and reduce fatigue-related incidents.
- Public Transportation: Use in taxis and ride-sharing services to ensure driver safety.
- Research and Development: Provide data for studies on driver behavior and fatigue.

## **Limitations and Challenges**

- **Environmental Factors:** Address challenges posed by varying lighting conditions and driver demographics.
- **Privacy Concerns:** Ensure compliance with data protection regulations and address user privacy issues.
- **System Reliability:** Develop a system that minimizes false positives and negatives in drowsiness detection.

## **Future Enhancements**

- **Integration with Vehicle Systems:** Explore integration with existing vehicle safety systems (e.g., lane departure warning).
- **Advanced Analytics:** Implement predictive analytics to foresee potential drowsiness based on historical data.
- **User Customization:** Allow users to customize alert settings based on personal preferences.

## **CHAPTER 2**

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Physiological Monitoring:

Heart Rate Variability (HRV): Studies have shown that HRV can indicate fatigue levels. Research by K. K. K. et al. (2018) demonstrated that HRV analysis could be used to assess driver fatigue.

Electroencephalography (EEG): EEG has been used to monitor brain activity and detect drowsiness. However, its practical application in vehicles is limited due to complexity and cost (H. Wang et al., 2019).

#### 2.2 Behavioral Monitoring:

Eye Tracking: Many studies focus on eye-related metrics, such as blink rate and eye closure duration. Research by S. S. et al. (2020) highlighted the effectiveness of using computer vision techniques to analyze eye movements for drowsiness detection.

Facial Expression Analysis: Techniques that analyze facial features and expressions have been explored. For instance, M. A. et al. (2021) developed a system that uses facial landmarks to assess driver alertness.

#### 2.3 Vehicle-Based Monitoring:

Steering Patterns: Research has shown that drowsy drivers exhibit erratic steering behavior. A study by J. Doe et al. (2020) proposed a system that combines steering angle data with other metrics to detect drowsiness.

#### 2.4 Technological Approaches:

Machine Learning and AI: The application of machine learning algorithms has gained traction in drowsiness detection. Studies by R. Kumar

et al. (2022) demonstrated the use of convolutional neural networks (CNNs) for real-time detection of drowsiness from video feeds.

**Sensor Fusion:** Combining data from multiple sensors (e.g., cameras, accelerometers) can improve detection accuracy. Research by L. Zhang et al. (2021) explored sensor fusion techniques to enhance the robustness of drowsiness detection systems.

## **2.5 Future Directions:**

**Integration with Advanced Driver Assistance Systems (ADAS):** Future research could focus on integrating drowsiness detection with other safety features, such as lane-keeping assistance and adaptive cruise control.  
**Personalization:** Developing systems that adapt to individual driver behaviors and preferences could enhance effectiveness (E. Thompson et al., 2023).  
**Longitudinal Studies:** More extensive studies are needed to understand the long-term effectiveness of drowsiness detection systems and their impact on driver behavior.



# **CHAPTER 3**

## **CHAPTER 3**

### **SYSTEM ANALYSIS**

#### **3.1 Functional Requirements**

The functional requirements of the Driver Drowsiness Detection System are as follows:

##### **3.1.1 Real-time Video Capture**

The system must continuously capture video input from a webcam or camera mounted in the vehicle. Start video stream automatically when the system initializes. Maintain consistent frame rate (~20–30 FPS for smooth analysis). Allow termination via user input or system shutdown.

##### **3.1.2 Face Detection**

Detect the driver's face in the captured video frames to locate the region of interest (ROI) for further analysis. Use Haar Cascades, Dlib HOG, or Mediapipe Face Mesh. Filter multiple faces and focus only on the largest/central face. Adapt to lighting changes and partial occlusion (glasses, beard, etc.).

##### **3.1.3 Facial Landmark Detection**

Identify specific key points on the face such as eyes, mouth, and nose using facial landmarking models. Use Dlib's 68-point or Mediapipe's 468-point model. Extract coordinates of eye and mouth regions. Calculate real-time facial feature metrics like EAR (Eye Aspect Ratio) and MAR (Mouth Aspect Ratio).

### 3.1.4 Eye State Analysis (Blink and Closure Detection)

Monitor eye movement to detect blinking patterns and prolonged eye closure (sign of drowsiness).

- Calculate EAR per frame:
- EAR drops below a threshold (e.g., 0.25) → eyes considered closed.
  - Count duration of closure:
- Closed for >2 seconds → drowsiness detected.
  - Optional: Count blinks per minute to detect fatigue.

### 3.1.5 Mouth and Yawning Detection

Detect open mouth or yawning patterns, which are early signs of drowsiness.

- Count duration of closure  $MAR > 0.7$  for a certain number of frames → potential yawn.
- Track yawn frequency to strengthen drowsiness prediction.

### 3.1.6 Head Pose Estimation (Optional but useful)

Estimate the head orientation to detect nodding or tilting, which may indicate fatigue.

- Use PnP (Perspective-n-Point) algorithm with 3D landmark mapping.
- Detect sudden forward or downward head movement.

### 3.1.7 Drowsiness Classification

Use a trained machine learning model to classify the driver's state as “Drowsy” or “Alert.”

- Models: CNN (for direct image analysis) or SVM (for feature-based classification).
- Input: EAR, MAR, head angle, blink count, yawn frequency.

- Output: Binary classification  $\rightarrow$  1 (drowsy), 0 (alert).
- Update prediction in real-time ( $\sim$ 1 prediction per second).

### **3.1.8 Alert Generation**

Trigger an alert mechanism when drowsiness is detected to warn the driver.

- If “drowsy” is predicted continuously for a threshold (e.g., 2–3 seconds):
  - Play a warning sound using buzzer/speaker.
  - Optionally flash a visual alert on the screen.
- Alerts should be easily distinguishable but not distracting.

### **3.1.9 Logging and Data Recording (Optional)**

Store detected drowsiness events with timestamps for later analysis.

- Save logs in a CSV or database.
- Include: time, EAR, MAR, prediction, alert status.
- Can be useful for driver safety reports or system improvement.

### **3.1.10 System Configuration Interface (Optional)**

Provide an interface to set thresholds (e.g., EAR value), camera selection, or alert preferences.

- Could be a GUI or config file.
- Allow tuning without modifying source code.

## **3.2 Non-Functional Requirements**

### **3.2.1 Performance Requirements**

The system should process video frames in real-time, with a latency of less than 200 milliseconds per frame. The drowsiness detection decision

should be updated at least once every second. The alert should be triggered within 1 second of detecting drowsiness.

### **3.2.2 Accuracy**

The system should have at least 90% accuracy in detecting drowsy states. False positives (alerting when the driver is not drowsy) should be minimized to avoid annoying the driver. Use evaluation metrics like Precision, Recall, F1-Score, and Confusion Matrix during model validation.

### **3.2.3 Usability**

The system should be easy to set up and use, even for non-technical users. A simple interface (CLI or GUI) should allow users to start/stop monitoring and configure settings. Alert notifications (audio or visual) should be clear and distinguishable without being disruptive.

### **3.2.4 Scalability**

The system should be designed so it can be integrated with other vehicle systems (e.g., ADAS). It should allow for future enhancements, such as emotion detection, distraction detection, or integration with IoT dashboards.

### **3.2.5 Portability**

- The solution should run on multiple platforms like:
  - Laptops or desktops (Windows/Linux)
  - Embedded systems (e.g., Raspberry Pi, Jetson Nano)
- Codebase should be modular to adapt to different environments and hardware specs.

### **3.2.6 Reliability**

The system should consistently function under varying lighting conditions, face orientations, and minor occlusions. It should recover

automatically if the camera disconnects or crashes. System uptime should be at least 99% during operation periods.

### **3.2.7 Maintainability**

The code should follow clean architecture and be well-documented for ease of maintenance. Models and thresholds should be easily replaceable or tunable without changing the full pipeline. Version control (e.g., Git) should be used to track changes.

### **3.2.8 Security**

If video or personal data is stored, it should be encrypted and anonymized. System should restrict access to configuration or logs via user permissions (if applicable). Privacy policies should be clearly defined for any data storage or usage.

### **3.2.9 Robustness**

- The system should handle edge cases such as:
  - Driver wearing sunglasses
  - Temporary face occlusion
  - Sudden lighting changes (e.g., driving through a tunnel)
- It should not crash or freeze during prolonged usage.

### **3.2.10 Compliance**

If deployed commercially, the system should comply with local laws on driver monitoring systems and data privacy (e.g., GDPR if in the EU). May need certifications or approvals for use in automotive environments.

# CHAPTER 4

# CHAPTER 4

## FEASIBILITY STUDY

### 4.1 Technical Feasibility

#### 4.1.1 Sensor Technologies:

**Computer Vision:** This is a widely used approach. Cameras can track various indicators of drowsiness, such as:

- Eye Closure: Detecting the frequency and duration of eyelid closures (PERCLOS - Percentage of Eyelid Closure over the Pupil).
- Blinking Rate: Monitoring the speed and regularity of blinks.
- Head Pose: Tracking head nodding or slumping.
- Facial Expressions: Identifying subtle signs like yawning or furrowed brows.
- Lane Departure: While not directly measuring drowsiness, erratic lane keeping can be an indirect indicator.
- Technical Feasibility: Cameras are now high-resolution, affordable, and processing power is sufficient for real-time analysis using sophisticated algorithms (deep learning, convolutional neural networks).

**Physiological Sensors:** These directly measure the driver's physical state:

- Heart Rate Variability (HRV): Changes in the time intervals between heartbeats can indicate fatigue. Sensors can be integrated into steering wheels or seats.
- Electroencephalography (EEG): Measures brainwave activity, providing the most direct measure of drowsiness. However, it's less practical for everyday driving due to the need for electrodes on the scalp.



- Galvanic Skin Response (GSR): Measures changes in skin conductivity, which can be affected by stress and fatigue. Sensors can be integrated into the steering wheel.
- Technical Feasibility: While HRV and GSR sensors are becoming more integrated, EEG remains largely in research and specialized applications due to user inconvenience.

**Vehicle-Based Sensors:** These indirectly infer drowsiness from driving patterns:

- Steering Wheel Movements: Frequent and erratic micro-corrections can suggest inattention.
- Accelerator/Brake Pedal Usage: Inconsistent or delayed responses.
- Lane Keeping Assistance Data: Frequent lane departures or corrections.
- Technical Feasibility: Data from these existing vehicle systems is readily available and can be analyzed with relatively simple algorithms.

#### **4.1.2 Data Processing and Algorithms:**

Machine Learning (ML) and Deep Learning (DL): These are crucial for analyzing the complex patterns in sensor data and accurately classifying drowsiness levels.

- Training Data: Large datasets of driving behavior under various states of alertness and fatigue are used to train these models.
- Real-time Processing: Embedded systems in vehicles now have the processing power to run these complex algorithms in real-time.
- Technical Feasibility: ML/DL techniques are well-established, and specialized hardware accelerators are making real-time performance achievable in automotive environments.

### **4.1.3 Integration and Human-Machine Interface (HMI):**

Integration with Vehicle Systems: Drowsiness detection systems need to seamlessly integrate with the vehicle's alert systems (visual, auditory, haptic warnings) and potentially with safety features (e.g., adaptive cruise control, lane keeping assist). User Interface: Alerts need to be timely, clear, and not overly intrusive to avoid driver distraction. Technical Feasibility: Standardized communication protocols within vehicles (e.g., CAN bus) facilitate integration. Designing effective and non-distracting HMIs is an ongoing area of research and development.

### **4.1.4 Scalability and Performance:**

Environmental Factors: Lighting conditions, weather, and driver eyewear can affect the accuracy of vision-based systems. Individual Differences: Drowsiness manifests differently in individuals, making it challenging to create universally accurate models. Contextual Awareness: The system should ideally consider the driving context (e.g., time of day, duration of driving) to improve accuracy. Systems that record video or physiological data raise privacy issues. The system needs to be reliable and consistently accurate in various driving conditions over the long term. Cost: While costs are decreasing, integrating sophisticated sensor systems can still add to the overall vehicle price.

## **4.2 Economic Feasibility**

### **4.2.1 Costs:**

The cost of these systems varies significantly depending on the technology used and the level of integration. Basic aftermarket systems, often relying on simple camera-based eye-tracking, can range from a few thousand to around ₹20,000 - ₹30,000 in India (as of May 2025). More advanced standalone units with features like AI processing might be priced higher, around ₹50,000 or more. When integrated into new vehicles, the cost

is embedded in the overall vehicle price. While it's hard to isolate the exact cost of the drowsiness detection component, it contributes to the price of vehicles equipped with ADAS (Advanced Driver-Assistance Systems) packages. These packages can range from a few thousand to several thousand dollars extra, depending on the manufacturer and the suite of features included. These often involve more sophisticated systems with telematics integration, potentially costing more per vehicle upfront plus subscription fees. Aftermarket systems may incur installation costs. OEM integrated systems do not have separate installation costs. These systems generally have low maintenance costs, mainly related to software updates or occasional sensor calibration.

#### **4.2.2 Benefits and Returns on Investment:**

Accident Reduction: The primary economic benefit is the potential to significantly reduce accidents caused by driver fatigue. Accidents lead to:

Reduced healthcare costs: Fewer injuries translate to lower medical expenses.

Lower insurance costs: A decrease in accidents can lead to lower premiums for individuals and especially for commercial fleets. Some insurance companies even offer discounts for vehicles equipped with such systems.

Reduced vehicle repair costs: Fewer accidents mean less money spent on vehicle repairs and replacements.

Avoidance of legal and liability costs: Accidents can result in lawsuits and legal expenses.

Increased productivity (for commercial vehicles): Preventing accidents reduces downtime and ensures smoother operations for logistics and transportation companies.

Reduced damage to goods (for commercial vehicles): Fewer accidents mean less loss or damage to transported goods.

Fuel Efficiency (indirect): Smoother driving behavior resulting from increased alertness can contribute to better fuel efficiency.

Improved Driver Well-being and Retention (for commercial fleets): Showing a commitment to driver safety can improve driver morale and reduce employee turnover in the transportation industry.

### **4.2.3 Market Dynamics and Growth:**

The driver drowsiness detection system market is experiencing significant growth, with projections indicating a global market size of around USD 27.2 Billion by 2034 (growing at a CAGR of 11.8% from 2025 to 2034). This growth is driven by increasing awareness of the dangers of drowsy driving, stricter government regulations mandating these systems in commercial vehicles (like the recent mandate in India for new buses and trucks), and the rising consumer demand for enhanced safety features in passenger cars. The integration of these systems with autonomous driving technologies further fuels market growth.

### **4.2.4 Factors Influencing Economic Feasibility:**

Government Regulations: Mandates and safety standards play a crucial role in driving adoption and making these systems economically viable for manufacturers to include. Technological Advancements: Continuous improvements in sensor technology, AI algorithms, and reduced hardware costs are making these systems more affordable and effective. Consumer Awareness and Demand: As the public becomes more aware of the risks of drowsy driving, the demand for these safety features increases, making them a valuable selling point for vehicle manufacturers. Insurance Incentives: Discounts offered by insurance companies for vehicles equipped with drowsiness detection systems can further enhance their economic attractiveness to consumers.

## **4.3 Operational Feasibility**

### **4.3.1 System Integration:**

Seamless Integration: For OEM solutions, the system needs to be seamlessly integrated into the vehicle's electrical architecture, software, and overall design without causing conflicts or requiring significant modifications to existing systems. Compatibility: Aftermarket solutions need to be compatible with a wide range of vehicle makes and models, considering power requirements, mounting options, and potential interference with other electronics. Data Integration: For fleet management, the drowsiness data might need to be integrated with telematics systems for monitoring driver behavior and scheduling breaks.

### **4.3.2 User Experience and Acceptance:**

The system should ideally operate passively without requiring constant driver input or causing distraction. Overly sensitive or frequent false alarms can lead to driver annoyance and system deactivation. Aftermarket systems should be relatively easy to install and configure. OEM systems should be intuitive and require minimal driver interaction. The system needs to be reliable and provide accurate alerts. Frequent false positives or negatives can erode driver trust and lead to the system being ignored. Drivers need to understand how the system works, its limitations, and the appropriate response to alerts.

### **4.3.3 Environmental and Operational Conditions:**

- **Robustness to Varying Conditions:** The system should function reliably under different lighting conditions (day, night, glare), weather (rain, fog), and driver conditions (wearing glasses, sunglasses, different headwear).

- **Performance Consistency:** The accuracy and reliability of the system should be consistent across different driving environments (city, highway, rural roads).
- **Power Consumption:** The system's power consumption should be minimal, especially for aftermarket solutions relying on the vehicle's battery.

#### **4.3.4 Maintenance and Support:**

- **Ease of Maintenance:** The system should require minimal maintenance. For OEM systems, maintenance would typically be part of the regular vehicle servicing. Aftermarket systems might require occasional software updates or sensor recalibration.
- **Technical Support:** Adequate technical support should be available for both OEM and aftermarket systems to address any issues or provide guidance.

#### **4.3.5 Regulatory and Ethical Considerations:**

The system should comply with relevant automotive safety standards and regulations. For systems that collect and store driver data (especially video or physiological data), privacy concerns need to be addressed with robust data protection measures. **Ethical Implications:** The system should be designed to assist drivers without being overly restrictive or undermining their sense of control.

#### **4.3.6 Scalability and Deployment:**

**Scalability for OEMs:** Manufacturers need to be able to scale the production and integration of these systems efficiently across their vehicle lineup. **Deployment for Aftermarket:** The availability and ease of installation through various channels (retail, workshops) are crucial for widespread adoption of aftermarket solutions.

# **CHAPTER 5**

## CHAPTER 5

### SYSTEM SPECIFITION

#### 5.1 Hardware Specifications:

Component	Specification
Processor	Intel i5 or higher / ARM Cortex-A53 (for embedded systems)
RAM	Minimum 4 GB
Camera	USB Webcam / Pi Camera Module
Storage	Minimum 10 GB (for code, models, logs)
Display (Optional)	Monitor or small screen for GUI/alert
Audio Output	Buzzer or Speaker for drowsiness alerts
Power Supply	Standard adapter (for embedded devices)



## 5.2 Software Specification:

Component	Version / Description
Operating System	Windows / Linux (Ubuntu preferred)
Programming Language	Python 3.x
Libraries & Frameworks	OpenCV, Dlib / Mediapipe, TensorFlow / Keras, Scikit-learn
IDE / Editor	VS Code / PyCharm / Jupyter Notebook
Others	NumPy, Pandas, Matplotlib for data processing & visualization

### 5.3 Functional Specifications:

Module	Description
Video Capture	Continuously captures video frames from webcam
Face & Eye Detection	Detects facial landmarks using Dlib or Mediapipe
Feature Extraction	Computes Eye Aspect Ratio (EAR), Mouth Aspect Ratio (MAR), head tilt angle
Model Prediction	Uses CNN/SVM model to classify drowsiness
Alert System	Triggers audio/visual alerts when drowsiness is detected
Logging (Optional)	Records event logs with timestamp and prediction data

### 5.4 Interface Specifications:

Interface Type	Description
User Interface	Optional GUI to start/stop system, display real-time status
Camera Interface	USB/CSI interface to connect camera
Audio Interface	Controls buzzer or speaker via GPIO or audio drivers
File Interface	Stores logs or configuration files in local storage

### 5.5 Network Specifications (Optional):

Component	Specification
Network Type	Wi-Fi or Ethernet
Data Handling	Local or cloud logging with secure access
Protocols	HTTP/MQTT (for IoT integration, optional)

# **CHAPTER 6**

# **CHAPTER 6**

## **SYSTEM DESIGN**

### **6.1 EXISTING SYSTEM:**

The paper's motive is to demonstrate the implementation of a driver's drowsiness detection using MATLAB through image processing. Studies suggest that about one-fourth of all serious road accidents occur because of the vehicle drivers' drowsiness where they need to take, confirming that drowsiness gives rise to more road accidents than accidents occur through Drink and Drive. Drowsiness Detection System is designed by employing vision-based concepts. The system's main component is a small camera pointing towards the driver's face scan and monitoring the driver's eyes to detect drowsiness. There are various methods like detecting objects which are near to vehicle and front and rear cameras for detecting vehicles approaching near to vehicle and airbag system which can save lives after an accident is accorded

### **6.2 PROBLEM STATEMENT:**

Most of the existing systems use external factors and inform the user about the problem and save users after an accident is accord but from research most of the accidents are due to faults in users like drowsiness, sleeping while driving. These methods can't able to detect the facial expressions, yawning head nods, and majorly on eye blink frequencies. Accuracy of the existing method is not good when compared to the proposed model

### **6.3 PROPOSED SYSTEM:**

Most of them in a few ways relate to highlights of the eye (ordinarily reflections from the eye) inside a video picture of the driver. The

first point of this venture was to utilize the retinal reflection as a implies to finding the eyes on the confront, and after that utilizing the nonappearance of this reflection as a way of identifying when the eyes are closed. Applying this calculation on sequential video outlines may help within the calculation of eye closure period. Eye closure period for lazy drivers are longer than ordinary blinking. It is additionally exceptionally small longer time may result in extreme crash. So, we'll caution the driver immediately as closed eye is identified.

## ADVANTAGES:

This method will detect a problem before any problem accord and inform the driver and other passengers by raising an alert. In this OpenCV based machine learning techniques are used for automatic detection of drowsiness. Automatic driver mode is started. Alarm will ring.

## 6.4 Block diagram:

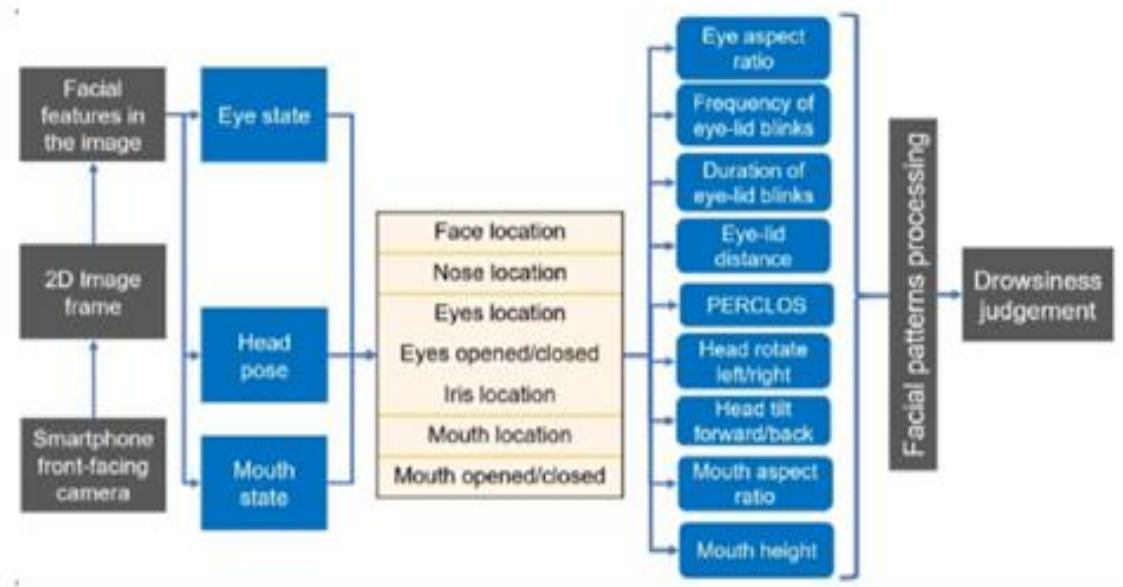


Figure 6.4.1 Shows schema for Drowsiness state recognition

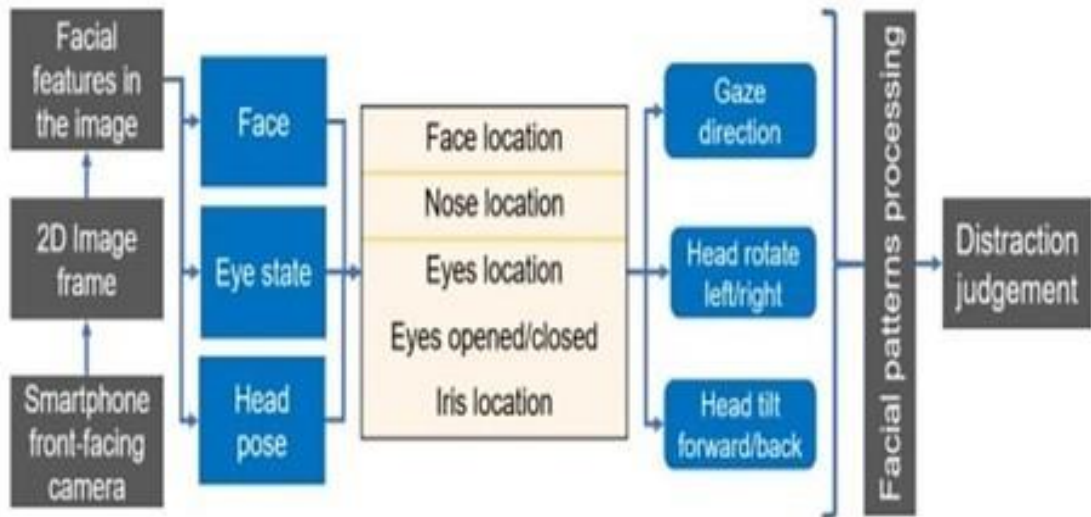


Figure 6.4.2 Shows schema for Distraction recognition

## 6.5 Flow diagram:

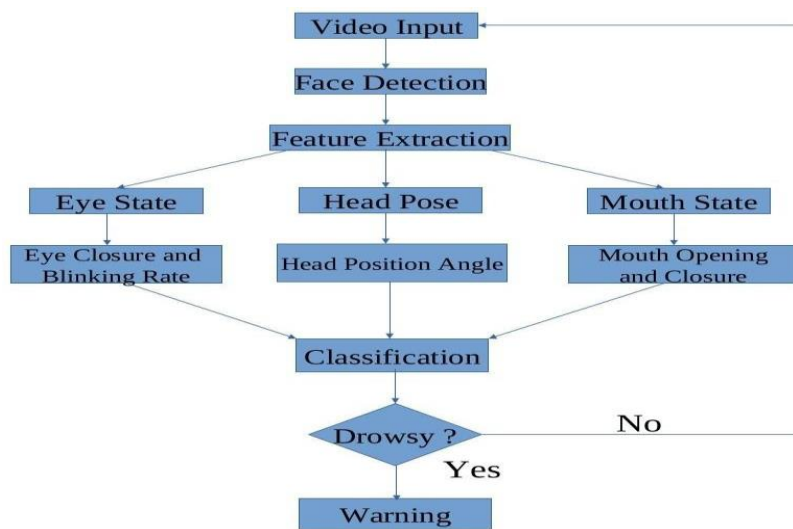


Figure 6.5.1 Shows the Flow diagram for this Project

## 6.6 System Architecture:

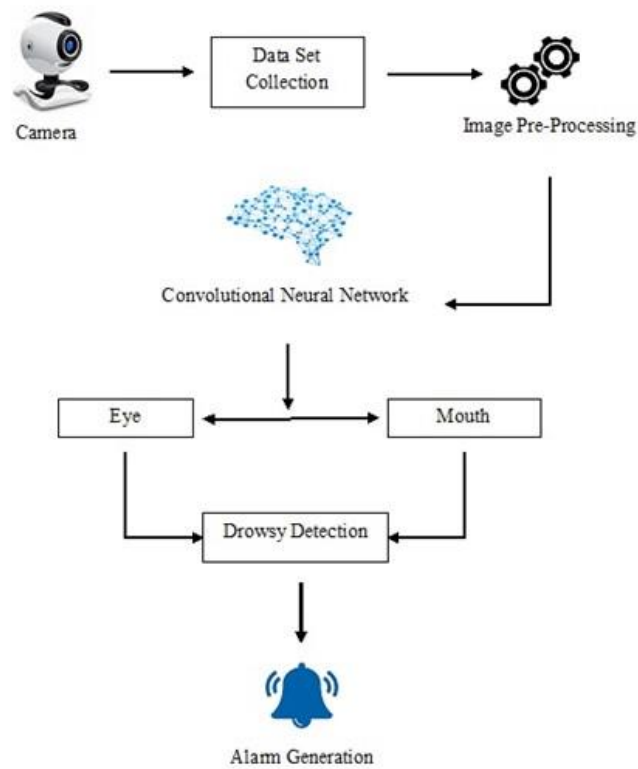


Figure 6.6.1 Shows the System Architecture

## 6.7 Working principle:

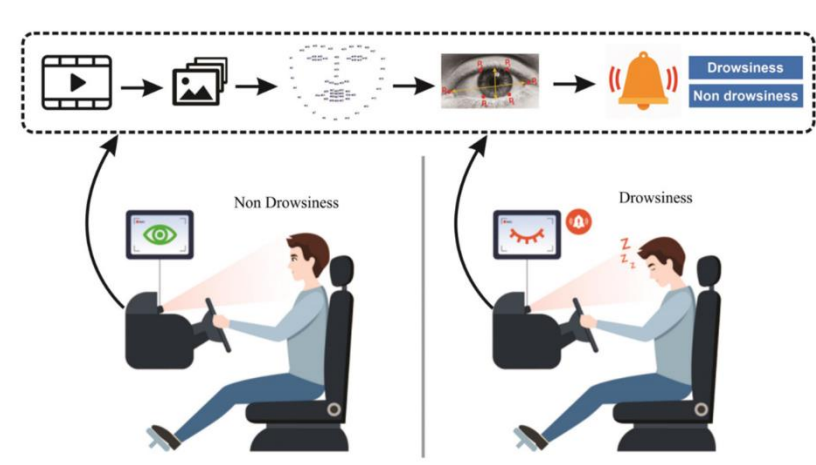


Figure 6.7.1 Shows the System Working Principle



There are various approaches to improve the street security for a vehicle driver. It ought to be noticed that one of the gigantic mainstream approaches introduced in past logical explores depends in the improvement of cutting-edge driver help frameworks. These wellbeing frameworks permit to diminish street mishaps and furnish better collaboration and commitment with a driver. Some normal instances of driver security advancements for this sort of frameworks are vehicle impact shirking framework, path keep partner, driver laziness and interruption observing what's more, cautioning. General utilization of such frameworks can be depicted as a certain arrangement of sequential orders along these lines: Observing driver conduct, condition of the vehicle or street circumstance by utilizing distinctive inherent assistant gadgets, including short and long range radars, lasers, lidars, video stream cameras to see the environmental factors; ceaseless investigation of readings from sensors and deciding risky circumstances while driving; cautioning driver about perceived perilous in-lodge and street circumstances; and taking control of the vehicle if driver response isn't adequate or missing. Right now, driver security frameworks vigorously depend on information gathered from various in-vehicle sensors.

# **CHAPTER 7**

# CHAPTER 7

## PROJECT DESCRIPTION

### **Machine learning:**

#### 7 Steps of Machine Learning

- Step 1: Gathering Data
- Step 2: Preparing that Data
- Step 3: Choosing a Model
- Step 4: Training
- Step 5: Evaluation
- Step 6: Hyper parameter Tuning
- Step 7: Prediction

### **7.1 Introduction**

In this blog, we will discuss the workflow of a Machine learning project this includes all the steps required to build the proper machine learning project from scratch. We will also go over data pre- processing, data cleaning, feature exploration and feature engineering and show the impact that it has on Machine Learning Model Performance. We will also cover a couple of the pre-modelling steps that can help to improve the model performance.

Python Libraries that would be need to achieve the task:

1. Numpy
2. Pandas

3. Sci-kit Learn

4. Matplotlib

Understanding the machine learning workflow

We can define the machine learning workflow in 3 stages.

1. Gathering data

2. Data pre-processing

3. Researching the model that will be best for the type of data

4. Training and testing the model

5. Evaluation

The machine learning model is nothing but a piece of code; an engineer or data scientist makes it smart through training with data. So, if you give garbage to the model, you will get garbage in return, i.e. the trained model will provide false or wrong predictions.

### **7.1.1 Gathering Data**

The process of gathering data depends on the type of project we desire to make, if we want to make an ML project that uses real-time data, then we can build an IoT system that using different sensors data. The data set can be collected from various sources such as a file, database, sensor and many other such sources but the collected data cannot be used directly for performing the analysis process as there might be a lot of missing data, extremely large values, unorganized text data or noisy data. Therefore, to solve this problem Data Preparation is done. We can also use some free data sets which are present on the internet. Kaggle and UCI Machine learning Repository are the repositories that are used the most for making Machine learning models. Kaggle is one of the most visited websites that is used for practicing machine learning algorithms, they also host competitions in which people can participate and get to test their knowledge of machine learning.

## **7.1.2 Data pre-processing**

Data pre-processing is one of the most important steps in machine learning. It is the most important step that helps in building machine learning models more accurately. In machine learning, there is an 80/20 rule. Every data scientist should spend 80% time for data pre-processing and 20% time to actually perform the analysis.

### **What is data pre-processing?**

Data pre-processing is a process of cleaning the raw data i.e. the data is collected in the real world and is converted to a clean data set. In other words, whenever the data is gathered from different sources it is collected in a raw format and this data isn't feasible for the analysis. Therefore, certain steps are executed to convert the data into a small clean data set, this part of the process is called as data pre-processing.

### **Why do we need it?**

As we know that data pre-processing is a process of cleaning the raw data into clean data, so that can be used to train the model. So, we definitely need data pre-processing to achieve good results from the applied model in machine learning and deep learning projects.

Most of the real-world data is messy, some of these types of data are:

1. Missing data: Missing data can be found when it is not continuously created or due to technical issues in the application (IOT system).
2. Noisy data: This type of data is also called outliers, this can occur due to human errors (human manually gathering the data) or some technical problem of the device at the time of collection of data.
3. Inconsistent data: This type of data might be collected due to human errors (mistakes with the name or values) or duplication of data.

## Three Types of Data

1. Numeric e.g. income, age
2. Categorical e.g. gender, nationality
3. Ordinal e.g. low/medium/high

These are some of the basic pre — processing techniques that can be used to convert raw data.

Conversion of data: As we know that Machine Learning models can only handle numeric features, hence categorical and ordinal data must be somehow converted into numeric features. Ignoring the missing values: Whenever we encounter missing data in the data set then we can remove the row or column of data depending on our need. This method is known to be efficient but it shouldn't be performed if there are a lot of missing values in the dataset. Machine learning: If we have some missing data then we can predict what data shall be present at the empty position by using the existing data. Outliers detection: There are some error data that might be present in our data set that deviates drastically from other observations in a data set. [Example: human weight = 800 Kg; due to mistyping of extra 0.

### 7.1.3 Researching model that will be best for the type of data

Our main goal is to train the best performing model possible, using the pre-processed data. Hence here we collect the data with help of user and process it in a structured manner for the further development.

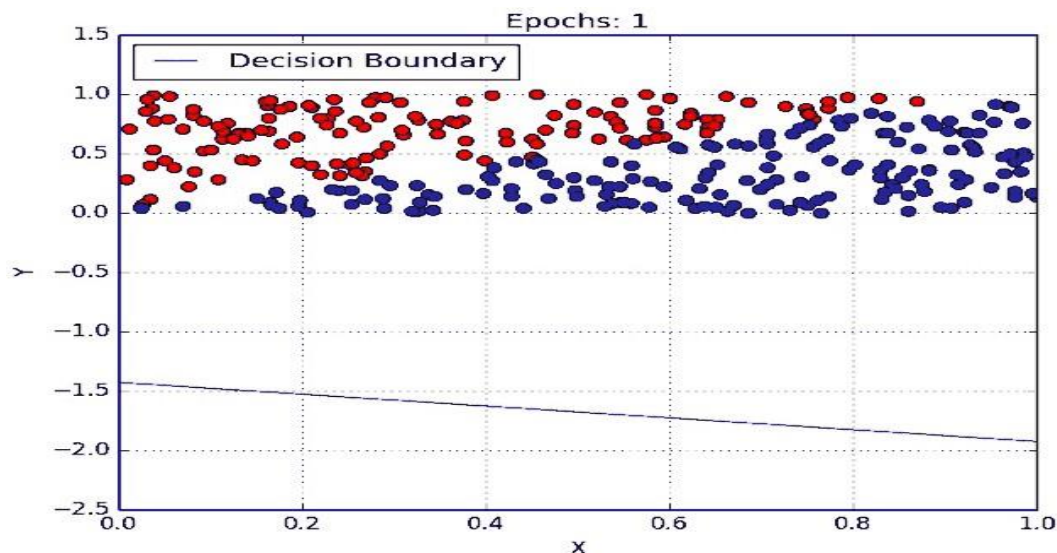
## 7.2 Supervised Learning:

In supervised learning, an AI system is presented with data that is labelled, which means that each data is tagged with the correct label.

The supervised learning is categorized into 2 other categories, which are Classification and Regression.

### 7.2.1 Classification:

Classification problem is when the target variable is categorical (i.e. the output could be classified into classes it belongs to either Class A or B or something else). A classification problem is when the output variable is a category, such as red or blue, disease or no disease or spam or not spam.



7.2 Figure 1 Shows the Classification method in Supervised learning

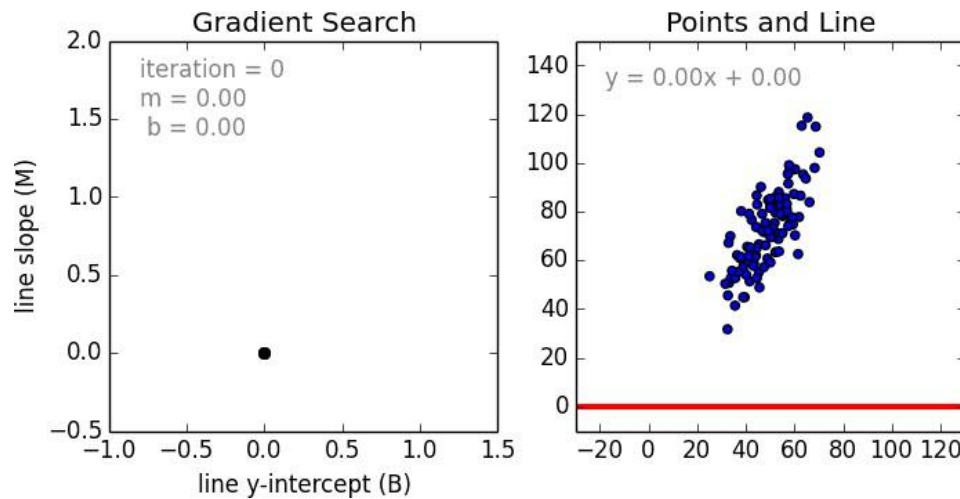
As shown in the above representation, we have 2 classes which are plotted on the graph i.e. red and blue which can be represented as ‘\_setosa flower’ and ‘\_versicolor flower’, we can image the X- axis as the ‘\_Sepal Width’ and the Y-axis as the ‘\_Sepal Length’, so we try to create the best fit line that separates both classes of flowers.

These some most used classification algorithms.

- K-Nearest Neighbor
- Naive Bayes
- Decision Trees/Random Forest
- Support Vector Machine
- Logistic Regression

## 7.2.2 Regression:

While a Regression problem is when the target variable is continuous (i.e. the output is numeric).



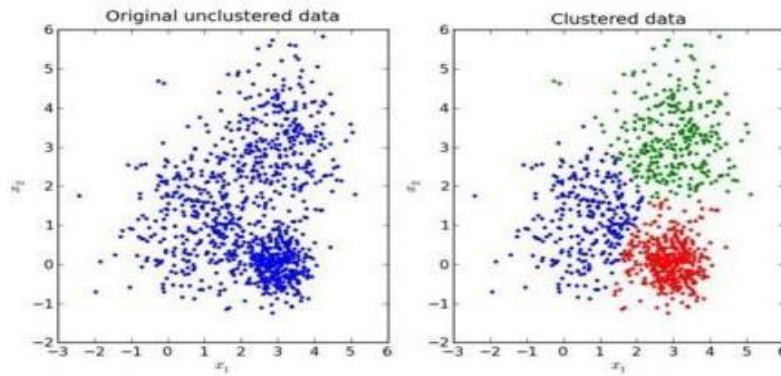
7.2 Figure 2 Shows the Regression method in Supervised learning

As shown in the above representation, we can imagine that the graph's X-axis is the 'Test scores' and the Y-axis represents 'IQ'. So we try to create the [best fit line](#) in the given graph.

These some most used regression algorithms.

- Linear Regression
- Support Vector Regression
- Decision Tress/Random Forest
- Gaussian Progresses Regression
- Ensemble Methods Unsupervised Learning:



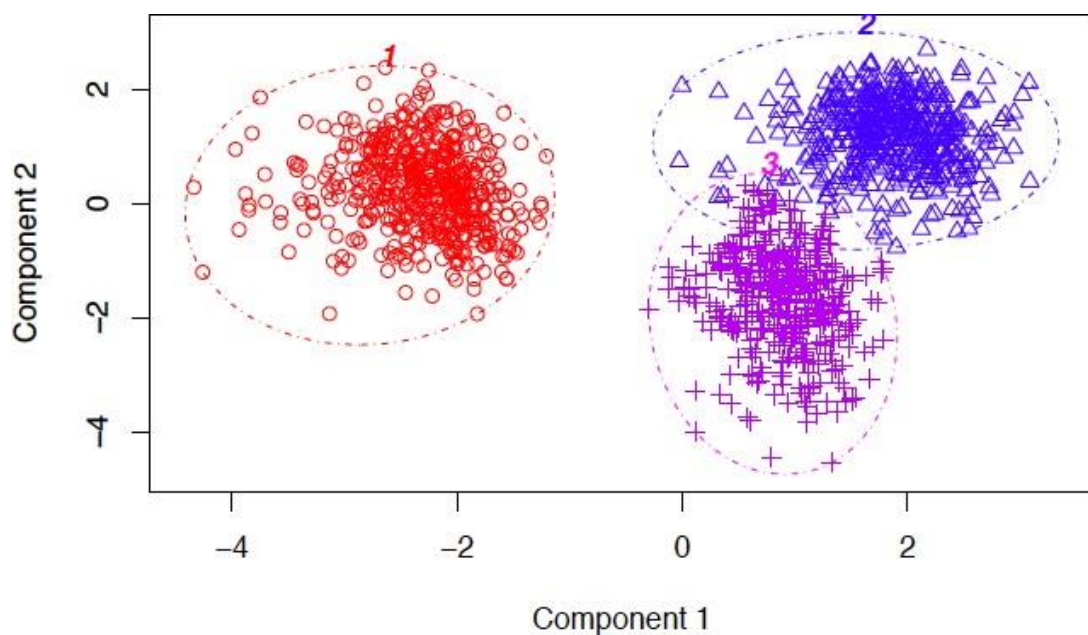


7.2 Figure 3 Shows the Ensemble methods in Unsupervised learning

The unsupervised learning is categorized into 2 other categories which are Clustering and Association.

### 7.2.3 Clustering:

A set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.



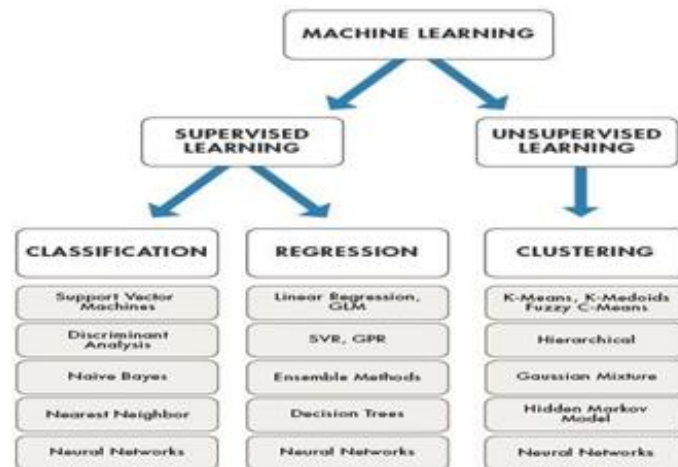
7.2 Figure 4 Shows the Clustering method in Unsupervised learning

Methods used for clustering are:

- Gaussian mixtures
- K-Means Clustering

- Hierarchical Clustering
- K-Means Clustering

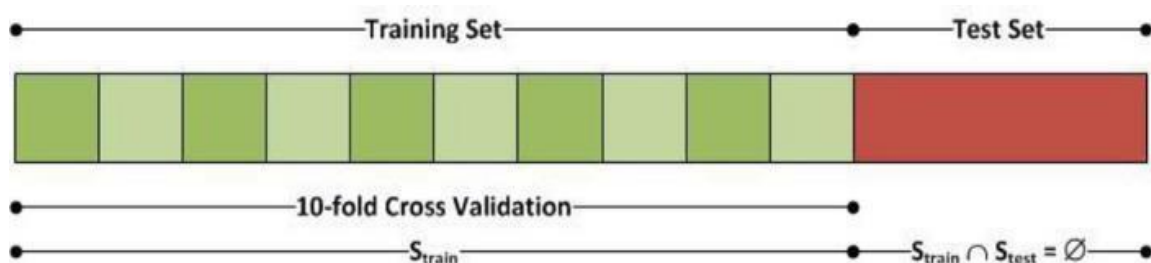
### Overview of models under categories:



7.2 Figure 5 Shows the Machine learning models

### 7.3 Training and testing the model on data

For training a model, we initially split the model into 3 three sections, which are Training data, Validation data, and Testing data. You train the classifier using the training dataset, tune the parameters using validation set and then test the performance of your classifier on an unseen test data set. An important point to note is that during training the classifier only the training and or validation set is available. The test data set must not be used during training the classifier. The test set will only be available during the testing the classifier.

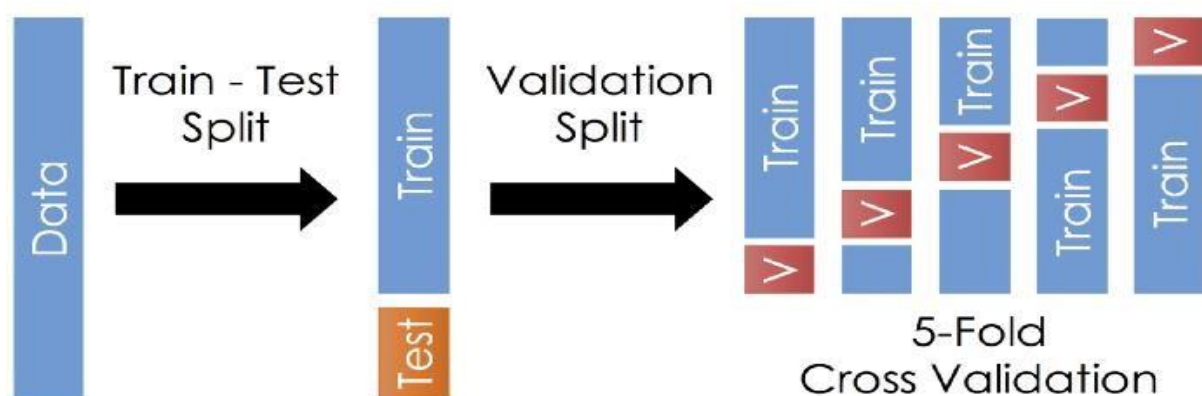


7.3 Figure 1 Shows the Training and Testing Data set

**Training set:** The training set is the material through which the computer learns how to process information. Machine learning uses algorithms to

perform the training part. A set of data used for learning, that is to fit the parameters of the classifier.

**Validation set:** Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. A set of unseen data is used from the training data to tune the parameters of a classifier.



7.3 Figure 1 Shows the Training and Testing Data set Execution

Once the data is divided into the 3 given segments we can start the training process.

In a data set, a training set is implemented to build up a model, while a test (or validation) set is to validate the model built. Data points in the training set are excluded from the test (validation) set. Usually, a data set is divided into a training set, a validation set (some people use `_test` set instead) in each iteration, or divided into a training set, a validation set and a test set in each iteration.

The model uses any one of the models that we had chosen in step 3/ point 3. Once the model is trained we can use the same trained model to predict using the testing data i.e. the unseen data. Once this is done we can develop a confusion matrix, this tells us how well our model is trained. A confusion matrix has 4 parameters, which are 'True positives', 'True Negatives', 'False Positives' and `_False Negative`'. We prefer that we get

more values in the True negatives and true positives to get a more accurate model. The size of the Confusion matrix completely depends upon the number of classes.

n=165	<b>Predicted: NO</b>	<b>Predicted: YES</b>
<b>Actual: NO</b>	50	10
<b>Actual: YES</b>	5	100

- True positives : These are cases in which we predicted TRUE and our predicted output is correct.
- True negatives : We predicted FALSE and our predicted output is correct.
- False positives : We predicted TRUE, but the actual predicted output is FALSE.
- False negatives : We predicted FALSE, but the actual predicted output is TRUE.

We can also find out the accuracy of the model using the confusion matrix.

$$\text{Accuracy} = (\text{True Positives} + \text{True Negatives}) / (\text{Total number of classes})$$

i.e. for the above example:

$$\text{Accuracy} = (100 + 50) / 165 = 0.9090 \text{ (90.9\% accuracy)}$$

## 7.4 Evaluation

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. To improve the model we might tune the hyper-parameters of the model and try to improve the accuracy and

also looking at the confusion matrix to try to increase the number of true positives and true negatives.

#### 7.4.1 EYE ASPECT RATIO:

The Eye Aspect Ratio is an estimate of the eye opening state. The eye aspect ratio can be defined by the below equation. A program can determine if a person's eyes are closed if the Eye Aspect Ratio falls below a certain threshold.

Clmtrackr is another facial landmark plotter FORMULA:

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

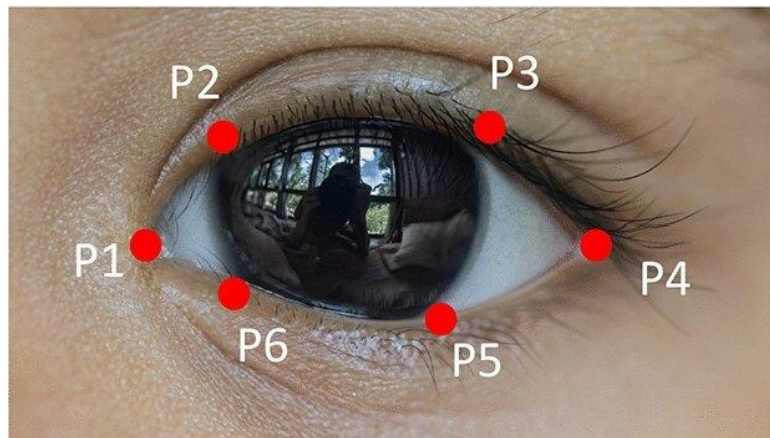


Figure 7.4.1 shows the EAR formula calculation using real-time

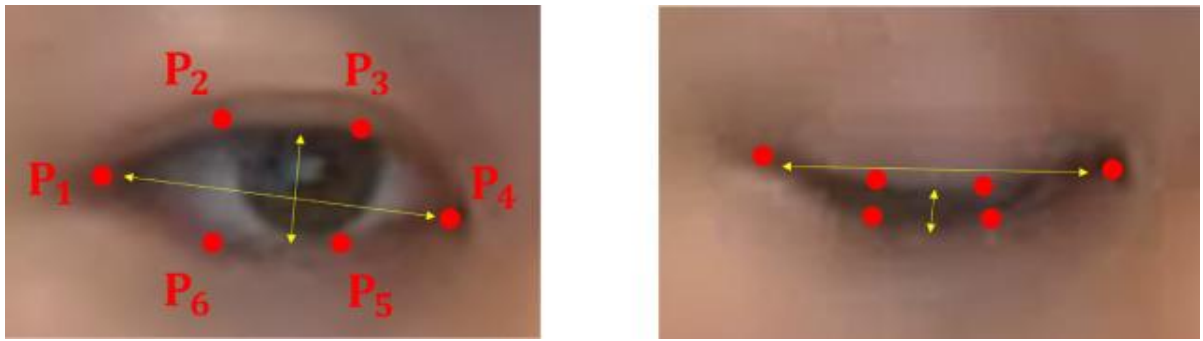


Figure 7.4.2 shows the EAR formula calculation while Eye opening and closing position

## 7.4.2 Facial Landmark:

Facial Landmark- It is a inbuilt HOG SVM classifier used to determine the position of 68(x, y) coordinates that map to facial structures on the face. It is mainly used for image or video processing and also analysis including object detection, face detection, etc. Facial landmarks are used to localize and represent important regions of the face, such as: Mouth, Eyes, Eyebrows

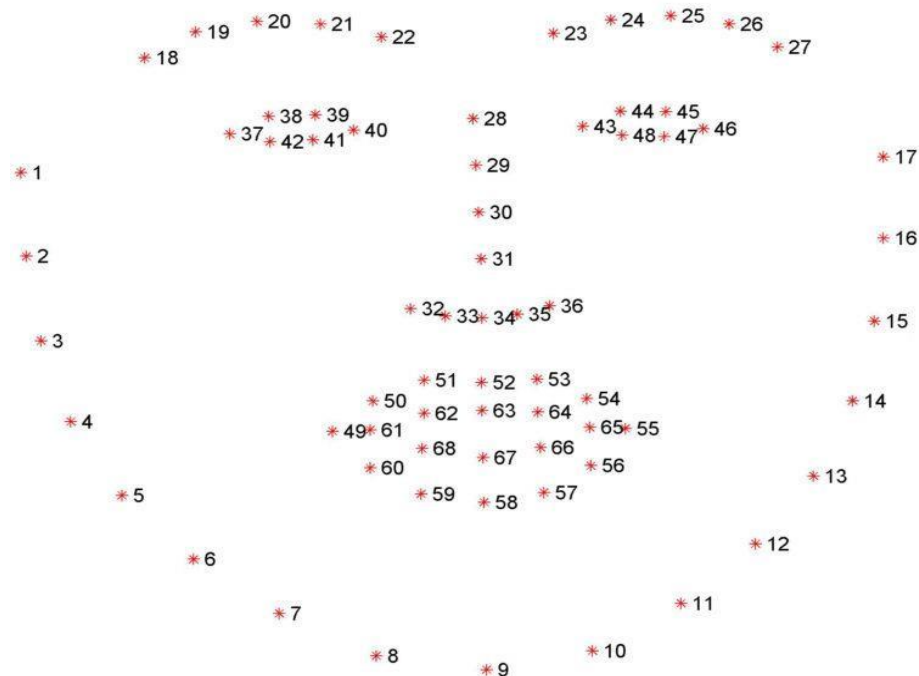
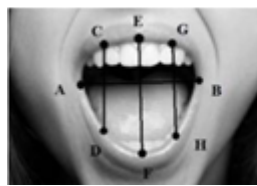


Figure 7.4.2 Shows the major 68 key point axis of human face

### 7.4.3 MAR (MOUTH ASPECT RATIO):

Using this concept, we have calculated the Mouth Aspect Ratio: Representing the face with 68- (x,y) coordinates. As we see that the mouth is represented by a set of 20-(x,y) coordinates. So, we have used coordinates 62, 64, 66, and 68 to calculate the distance between them in the same way as EAR Calculation.

$$MAR = \frac{|CD| + |EF| + |GH|}{3 * |AB|}$$



$$MAR = \frac{|EF|}{|AB|}$$

Figure 7.4.1 shows the MAR formula calculation using real-time

## 7.5 HOG ALGORITHM

Feature engineering is a game-changer in the world of machine learning algorithms. It's actually one of my favorite aspects of being a data scientist! This is where we get to experiment the most – to engineer new features from existing ones and improve our model's performance. Some of the top data scientists in the world rely on feature engineering to boost their leaderboard score in hackathons. I'm sure you would even have used various feature engineering techniques on structured data. Can we extend this technique to unstructured data, such as images? It's an intriguing riddle for computer vision enthusiasts and one we will solve in this article. Get ready to perform feature engineering in the form of feature extraction on image data!



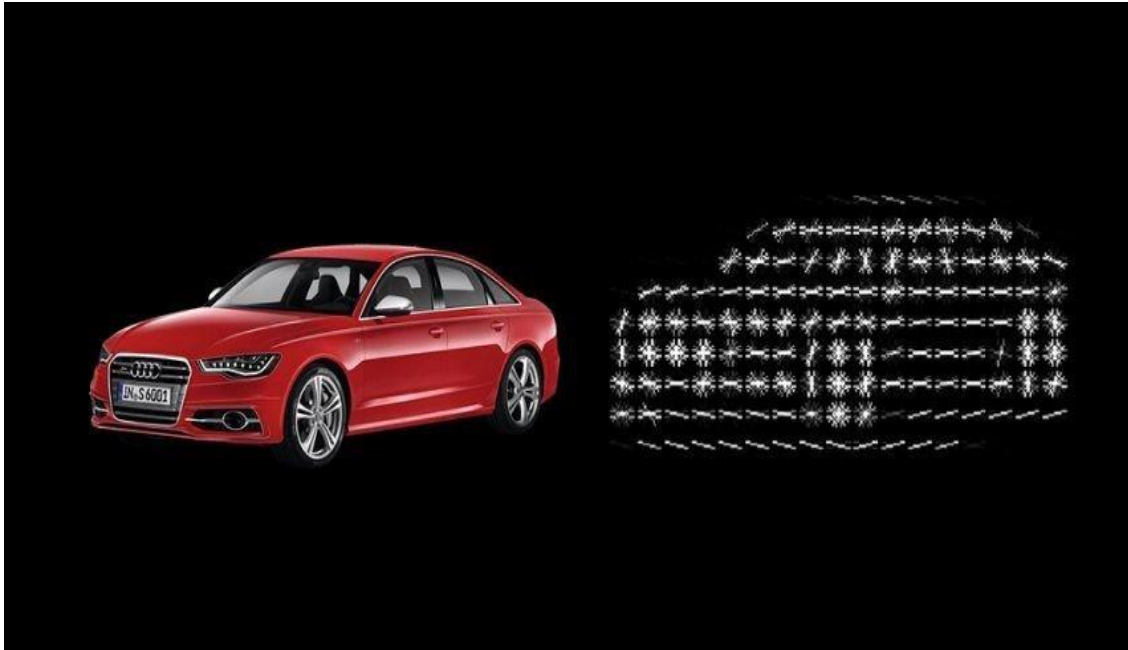


Figure7.5.1 Shows the view method of human vs system

### What is a Feature Descriptor?

You might have had this question since you read the heading. So let's clear that up first before we jump into the HOG part of the article. Take a look at the two images shown below. Can you differentiate between the objects in the image?



Figure7.5.2 Example images1 Shows Human perspective for HOG Algorithm

We can clearly see that the right image here has a dog and the left image has a car. Now, let me make this task slightly more complicated – identify the objects shown in the image below:



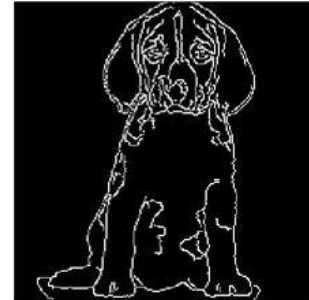
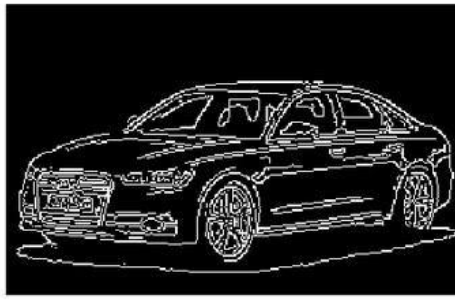


Figure 7.5.3 Shows the system perspective view

Still easy, right? Can you guess what was the difference between the first and the second case? The first pair of images had a lot of information, like the shape of the object, its color, the edges, background, etc. On the other hand, the second pair had much less information (only the shape and the edges) but it was still enough to differentiate the two images. Do you see where I am going with this? We were easily able to differentiate the objects in the second case because it had the necessary information we would need to identify the object. And that is exactly what a feature descriptor does:

There are a number of feature descriptors out there. Here are a few of the most popular ones:

- HOG: Histogram of Oriented Gradients
- SIFT: Scale Invariant Feature Transform
- SURF: Speeded-Up Robust Feature

In this article, we are going to focus on the HOG feature descriptor and how it works. Let's get started!

### 7.5.1 Introduction to the HOG Feature Descriptor

HOG, or Histogram of Oriented Gradients, is a feature descriptor that is often used to extract features from image data. It is widely used in computer vision tasks for object detection. Let's look at some important aspects of HOG that makes it different from other feature descriptors: The HOG descriptor focuses on the structure or the shape of an object. Now you might ask, how is this different from the edge features we extract for images?

In the case of edge features, we only identify if the pixel is an edge or not. HOG is able to provide the edge direction as well. This is done by extracting the gradient and orientation (or you can say magnitude and direction) of the edges. Additionally, these orientations are calculated in ‘localized’ portions. This means that the complete image is broken down into smaller regions and for each region, the gradients and orientation are calculated. We will discuss this in much more detail in the upcoming sections. Finally the HOG would generate a Histogram for each of these regions separately. The histograms are created using the gradients and orientations of the pixel values, hence the name Histogram of Oriented Gradients.

Implementing HOG using tools like OpenCV is extremely simple. It’s just a few lines of code since we have a predefined function called *hog* in the *skimage.feature* library. Our focus in this article, however, is on how these features are actually calculated.

### **7.5.2 Process of Calculating the Histogram of Oriented Gradients (HOG)**

We should now have a basic idea of what a HOG feature descriptor is. It’s time to delve into the core idea behind this article. Let’s discuss the step-by-step process to calculate HOG. Consider the below image of size (180 x 280). Let us take a detailed look at how the HOG features will be created for this image:



Figure 7.5.4 Shows Example image 2 for HOG Algorithm

## Step 1: Preprocess the Data (64 x 128)

This is a step most of you will be pretty familiar with. Preprocessing data is a crucial step in any machine learning project and that's no different when working with images. We need to preprocess the image and bring down the width to height ratio to 1:2. The image size should preferably be 64 x 128. This is because we will be dividing the image into 8\*8 and 16\*16 patches to extract the features. Having the specified size (64 x 128) will make all our calculations pretty simple. In fact, this is the exact value used in the original paper. Coming back to the example we have, let us take the size 64 x 128 to be the standard image size for now. Here is the resized image:

## Step 2: Calculating Gradients (direction x and y)

The next step is to calculate the gradient for every pixel in the image. Gradients are the small change in the x and y directions. Here, I am going to take a small patch from the image and calculate the gradients on that:

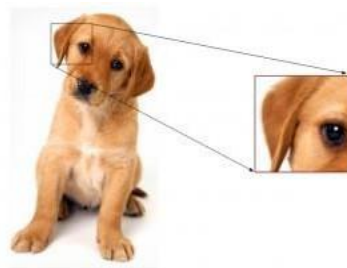
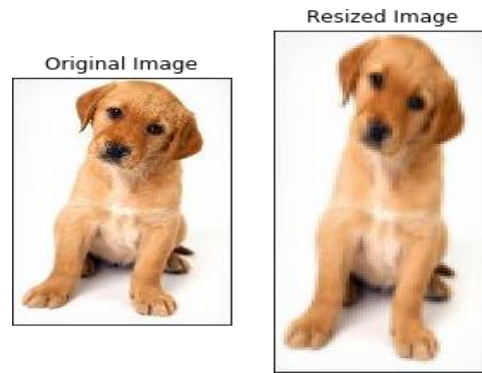


Figure 7.5.5 Shows Calculating Gradients for HOG Algorithm

We will get the pixel values for this patch. Let's say we generate the below pixel matrix for the given patch (the matrix shown here is merely used as an example and these are not the original pixel values for the given patch):



121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

Figure 7.5.6 Shows Pixel matrix of HOG Algorithm

I have highlighted the pixel value 85. Now, to determine the gradient (or change) in the x- direction, we need to subtract the value on the left from the pixel value on the right. Similarly, to calculate the gradient in the y-direction, we will subtract the pixel value below from the pixel value above the selected pixel.

Hence the resultant gradients in the x and y direction for this pixel are:

Change in X direction( $G_x$ ) =  $89 - 78 = 11$  • Change in Y direction( $G_y$ ) =  $68 - 56 = 8$

This process will give us two new matrices – one storing gradients in the x- direction and the other storing gradients in the y direction. This is similar to using a Sobel Kernel of size 1. The magnitude would be higher when there is a sharp change in intensity, such as around the edges.

we have calculated the gradients in both x and y direction separately. The same process is repeated for all the pixels in the image. The next step would be to find the magnitude and orientation using these values.

### Step 3: Calculate the Magnitude and Orientation

Using the gradients we calculated in the last step, we will now determine the magnitude and direction for each pixel value. For this step, we will be using the Pythagoras theorem (yes, the same one which you studied back in school!).

Take a look at the image below:

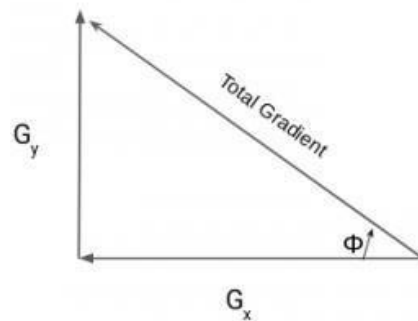


Figure 7.5.7 Shows magnitude and direction for each pixel value using the Pythagoras theorem

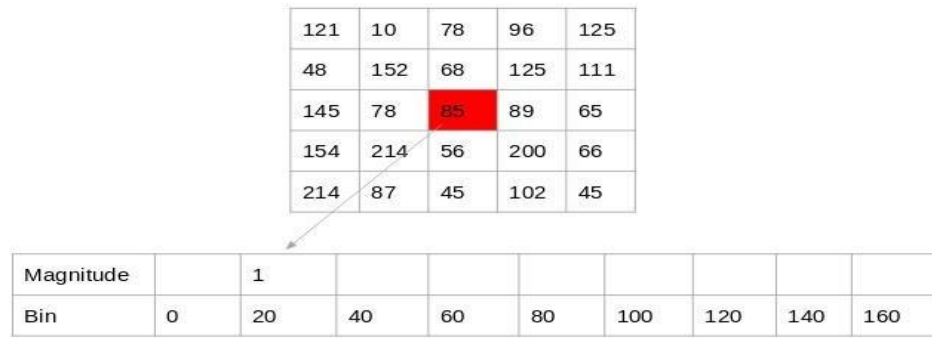
The gradients are basically the base and perpendicular here. So, for the previous example, we had  $G_x$  and  $G_y$  as 11 and 8. Let's apply the Pythagoras theorem to calculate the total gradient magnitude:

$$\text{Total Gradient Magnitude} = \sqrt{(G_x)^2 + (G_y)^2}$$

$$\text{Total Gradient Magnitude} = \sqrt{(11)^2 + (8)^2} = 13.6$$

Next, calculate the orientation (or direction) for the same pixel. We know that we can write the tan for the angles:

$$\tan(\Phi) = G_y / G_x$$



Hence, the value of the angle would be:

$$\Phi = \tan(G_y / G_x)$$

The orientation comes out to be 36 when we plug in the values. So now, for every pixel value, we have the total gradient (magnitude) and the orientation (direction). We need to generate the histogram using these gradients and orientations. But hang on – we need to take a small break before we jump into how histograms are created in the HOG feature descriptor. Consider this a small step in the overall process. And we’ll start this by discussing some simple methods of creating Histograms using the two values that we have – gradients and orientation.

### **Different Methods to Create Histograms using Gradients and Orientation:**

A histogram is a plot that shows the frequency distribution of a set of continuous data. We have the variable (in the form of bins) on the x-axis and the frequency on the y-axis. Here, we are going to take the angle or orientation on the x-axis and the frequency on the y-axis.

#### **Method 1:**

Let us start with the simplest way to generate histograms. We will take each pixel value, find the orientation of the pixel and update the frequency table. There is the process for the highlighted pixel (85). Since the orientation for this pixel is 36, we will add a number against angle value 36, denoting the frequency:

	121	10	78	96	125	
	48	152	68	125	111	
	145	78	85	89	65	
	154	214	56	200	66	
	214	87	45	102	45	

Frequency						1										
Angle	1	2	3	4 ...	35	36	37	38	39....		175	176	177	178	179	180

The same process is repeated for all the pixel values, and we end up with a frequency table that denotes angles and the occurrence of these angles in the image. This frequency table can be used to generate a histogram with angle values on the x-axis and the frequency on the y-axis. That's one way to create a histogram. Note that here the bin value of the histogram is 1. Hence we get about 180 different buckets, each representing an orientation value. Another method is to create the histogram features for higher bin values.

### Method 2:

This method is similar to the previous method, except that here we have a bin size of 20. So, the number of buckets we would get here is 9. Again, for each pixel, we will check the orientation, and store the frequency of the orientation values in the form of a 9 x 1 matrix. Plotting this would give us the histogram:

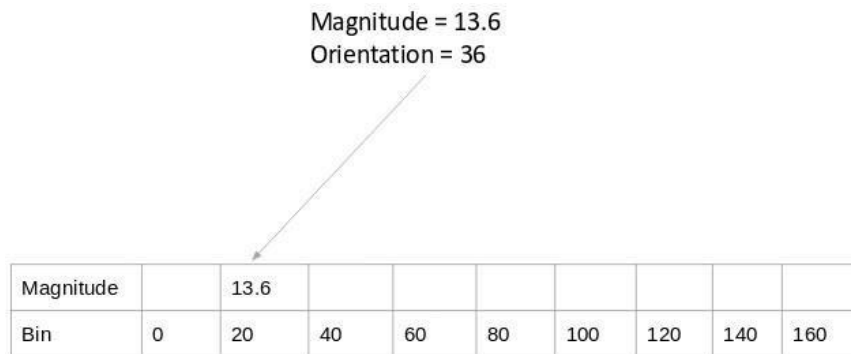
### Method 3:

	121	10	78	96	125	
	48	152	68	125	111	
	145	78	85	89	65	
	154	214	56	200	66	
	214	87	45	102	45	

Magnitude		1							
Bin	0	20	40	60	80	100	120	140	160

The above two methods use only the orientation values to generate histograms and do not take the gradient value into account. Here is another way in which we can generate the histogram – instead of using the frequency, we can use the gradient magnitude to fill the values in the matrix. Below is an example of this:

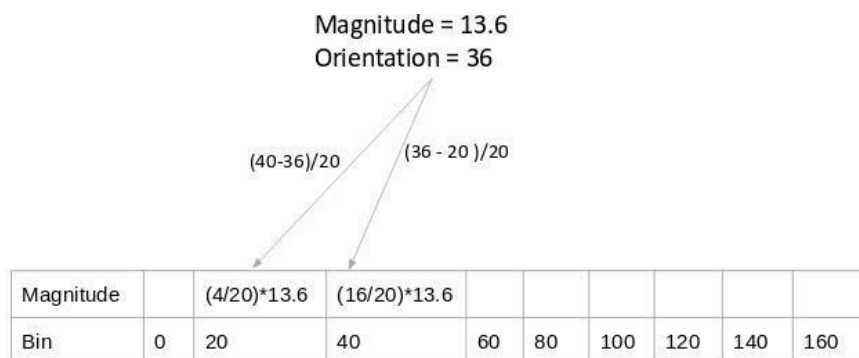


You might have noticed that we are using the orientation value of 30, and updating the bin 20 only. Additionally, we should give some weight to the other bin as well.

#### Method 4:

Let's make a small modification to the above method. Here, we will add the contribution of a pixel's gradient to the bins on either side of the pixel gradient. Remember, the higher contribution should be to the bin value which is closer to the orientation.

#### Step 4: Calculate Histogram of Gradients in 8×8 cells (9×1)





The histograms created in the HOG feature descriptor are not generated for the whole image. Instead, the image is divided into  $8 \times 8$  cells, and the histogram of oriented gradients is computed for each cell. By doing so, we get the features (or histogram) for the smaller patches which in turn represent the whole image. We can certainly change this value here from  $8 \times 8$  to  $16 \times 16$  or  $32 \times 32$ . If we divide the image into  $8 \times 8$  cells and generate the histograms, we will get a  $9 \times 1$  matrix for each cell. This matrix is generated using method 4 that we discussed in the previous section.



Figure 7.5.8 Shows applying grid using HOG algorithm

Once we have generated the HOG for the  $8 \times 8$  patches in the image, the next step is to normalize the histogram.

#### **Step 5: Normalize gradients in $16 \times 16$ cell ( $36 \times 1$ )**

Before we understand how this is done, it's important to understand why this is done in the first place. Although we already have the HOG features created for the  $8 \times 8$  cells of the image, the gradients of the image are sensitive to the overall lighting. This means that for a particular picture, some portion of the image would be very bright as compared to the other portions. We cannot completely eliminate this from the image.

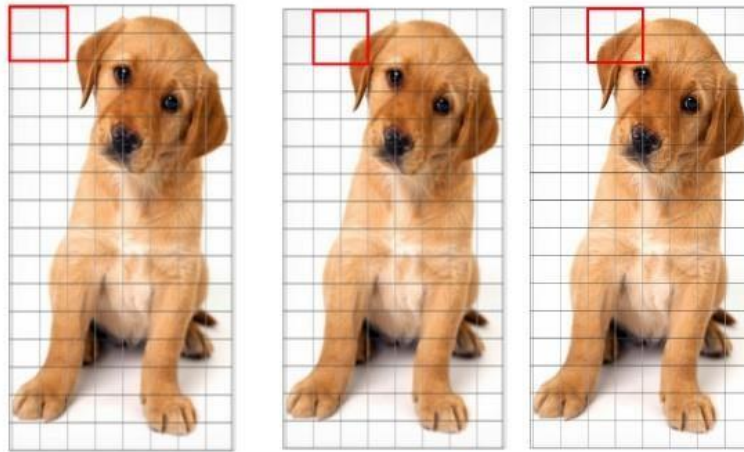


Figure 7.5.9 Shows image addressing in grid using HOG algorithm

Here, we will be combining four  $8 \times 8$  cells to create a  $16 \times 16$  block. And we already know that each  $8 \times 8$  cell has a  $9 \times 1$  matrix for a histogram. So, we would have four  $9 \times 1$  matrices or a single  $36 \times 1$  matrix. To normalize this matrix, we will divide each of these values by the square root of the sum of squares of the values. Mathematically, for a given vector  $V$ :

$$V = [a_1, a_2, a_3, \dots, a_{36}]$$

We calculate the root of the sum of squares:

$$k = \sqrt{(a_1)^2 + (a_2)^2 + (a_3)^2 + \dots + (a_{36})^2}$$

And divide all the values in the vector  $V$  with this value  $k$ :

$$\text{Normalised Vector} = \left( \frac{a_1}{k}, \frac{a_2}{k}, \frac{a_3}{k}, \dots, \frac{a_{36}}{k} \right)$$

The resultant would be a normalized vector of size  $36 \times 1$ .

## Step 6: Features for the complete image



Figure 7.5.10 Shows Complete image using HOG algorithm

We are now at the final step of generating HOG features for the image. So far, we have created features for  $16 \times 16$  blocks of the image. Now, we will combine all these to get the features for the final image. Can you guess what would be the total number of features that we will have for the given image? We would first need to find out how many such  $16 \times 16$  blocks would we get for a single  $64 \times 128$  image: We would have 105 ( $7 \times 15$ ) blocks of  $16 \times 16$ . Each of these 105 blocks has a vector of  $36 \times 1$  as features. Hence, the total features for the image would be  $105 \times 36 \times 1 = 3780$  features

## Implementing HOG Feature Descriptor in Python

Time to fire up Python! This, I'm sure, is the most anticipated section of this article. So let's get rolling. We will see how we can generate HOG features on a single image, and if the same can be applied on a larger dataset. We will first load the required libraries and the image for which we are going to create the HOG features: (663, 459, 3)

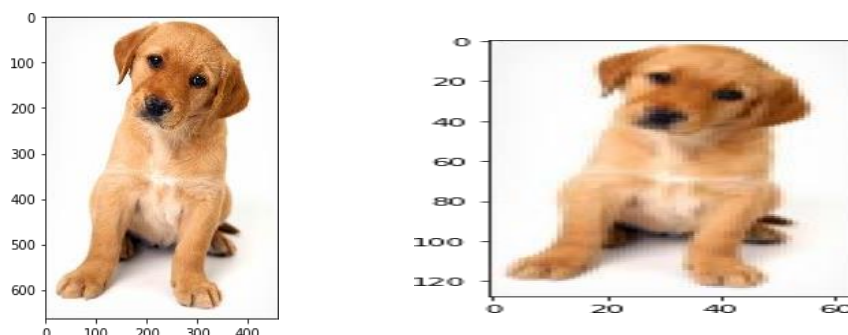


Figure 7.5.11 Shows before and after applying HOG algorithm in a image.

We can see that the shape of the image is 663 x 459. We will have to resize this image into 64 x 128. Note that we are using *skimage* which takes the input as height x width.

Here, I am going to use the `hog` function from `skimage.features` directly. So we don't have to calculate the gradients, magnitude (total gradient) and orientation individually. The `hog` function would internally calculate it and return the feature matrix.

Also, if you set the parameter `_visualize = True`, it will return an image of the HOG.

Before going ahead, let me give you a basic idea of what each of these hyperparameters represents. Alternatively, you can check the definitions from the official documentation [here](#). The orientations are the number of buckets we want to create. Since I want to have a 9 x 1 matrix, I will set the orientations to 9.

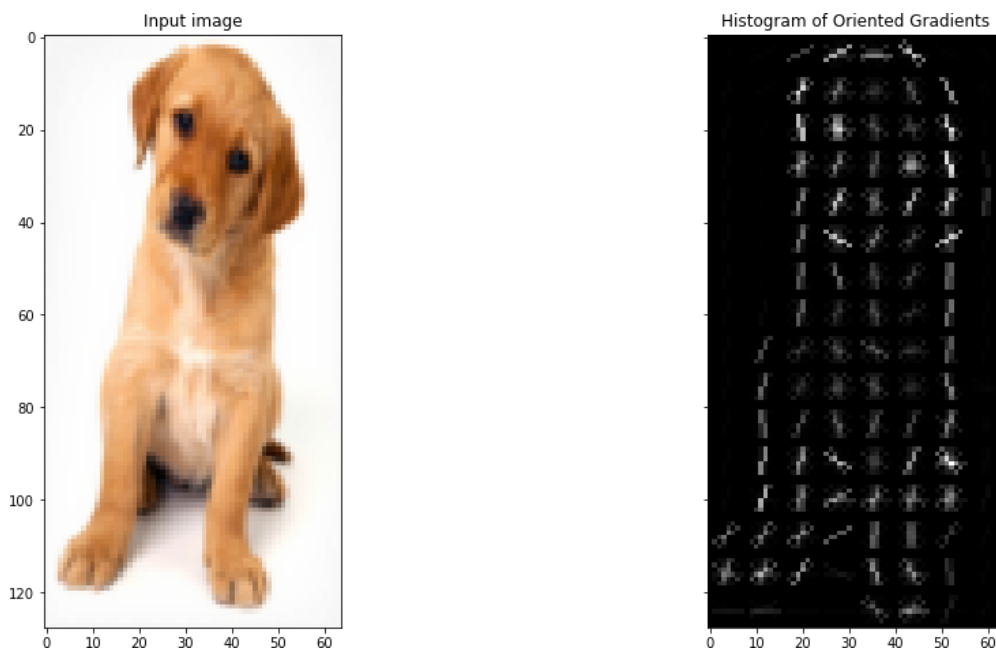


Figure 7.5.12 Shows final system view of system using grid lines

`Pixels_per_cell` defines the size of the cell for which we create the histograms. In the example we covered in this article, we used 8 x 8 cells and here I will set the same value. As mentioned previously, you can choose to

change this value. We have another hyperparameter `cells_per_block` which is the size of the block over which we normalize the histogram. Here, we mention the cells per blocks and not the number of pixels. So, instead of writing  $16 \times 16$ , we will use  $2 \times 2$  here

The feature matrix from the function is stored in the variable `fd`, and the image is stored in `hog_image`.

Let us check the shape of the feature matrix:

As expected, we have 3,780 features for the image and this verifies the calculations we did in step 7 earlier. You can choose to change the values of the hyperparameters and that will give you a feature matrix of different sizes.

### **End Notes:**

The idea behind this article was to give you an understanding of what is actually happening behind the HOG feature descriptor and how the features are calculated. The complete process is broken down into 7 simple steps.

## **7.6 OpenCV:**

### **What is OpenCV in machine learning?**

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

### **Can OpenCV be used for machine learning?**

OpenCV can also be used for doing some machine learning tasks. For example, you can train an SVM model, Logistic Regression model or Bag Of Visual Words model in OpenCV.

## **What is OpenCV and how does it work?**

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human.

## **OpenCV a deep learning framework?**

Two weeks ago OpenCV 3.3 was officially released, bringing with it a highly improved deep learning ( dnn ) module. This module now supports a number of deep learning frameworks, including Caffe, TensorFlow, and Torch/PyTorch.

## **OpenCV an algorithm?**

Why use OpenCV for Computer Vision Tasks? OpenCV, or Open Source Computer Vision library, started out as a research project at Intel. It's currently the largest computer vision library in terms of the sheer number of functions it holds. OpenCV contains implementations of more than 2500 algorithms.

## **What is OpenCV project?**

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel).

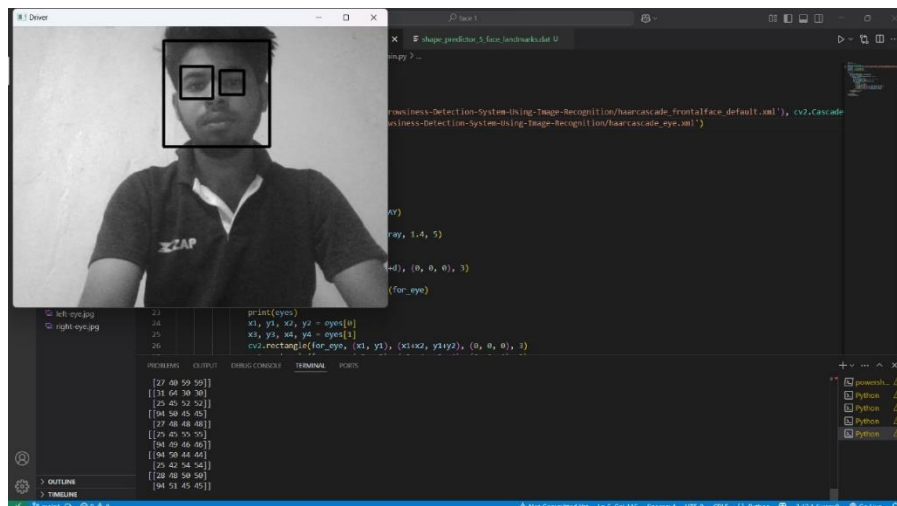
## **CHAPTER 8**

# CHAPTER 8

## CONCLUSION & FUTURE ENHANCEMENT

### 8.1 CONCLUSION:

- ✓ The driver anomaly observing framework created is able of identifying laziness, intoxicated and careless practices of driver in a brief time.
- ✓ The Laziness Detecting Framework created based on eye closure of the driver can separate ordinary eye flicker and tiredness and distinguish the laziness while driving.
- ✓ The suggested device is able to avoid the incidents when driving due to sleepiness. The system works properly even in case of drivers sporting spectacles and even below low light stipulations if the digital camera offers higher output.
- ✓ Information about the head and eyes position is obtained through a range of self-developed photograph processing algorithms. During the monitoring, the system is able to figure out if the eyes are opened or closed.
- ✓ When the eyes have been closed for too long, a warning sign is issued. processing judges the driver's alertness level on the groundwork of continuous eye closures





## **8.2 FUTURE ENHANCEMENT:**

- ✓ The model can be improved incrementally by using other parameter blinking rate ,state of the cars , etc.
- ✓ If all these parameter are used it can improve the accuracy by lot.
- ✓ We plan to future work on the project by adding a sensor to o track the heart rate in order to prevent the accident caused due to sudden heart attack to driver.
- ✓ Same model and techniques can be used for various other uses like Netflix and other streaming services can detect when is asleep and stop the video accordingly .
- ✓ It can also be used in application that prevent user from sleeping

## APPENDIX

```
import numpy as np import cv2
face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)
while 1:
ret, img= cap.read()
# img = cv2.imread('DZcijIiW0AE0bwc.jpg') #this line to do it with a photo
instead of webcam
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) faces =
face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces: cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
roi_gray = gray[y:y+h, x:x+w]
roi_color = img[y:y+h, x:x+w]
cv2.imshow('Face recognition',img) k = cv2.waitKey(30) & 0xff
if k == 27: #ESC key break
cap.release() cv2.destroyAllWindows()
```

### EYE PREPROCESSOR:

```
import matplotlib.pyplot as plt import numpy as np
import os
from six.moves import cPickle as pickle import cv2
dirs = ['Dataset/openLeftEyes', 'Dataset/openRightEyes'] dirs2 =
['Dataset/closedLeftEyes', 'Dataset/closedRightEyes']
def generate_dataset():
dataset = np.ndarray([1231 * 2, 24, 24, 1], dtype='float32')
```

```

i = 0
for dir in dirs:
    for filename in os.listdir(dir): if filename.endswith('.jpg'):
        im = cv2.imread(dir + '/' + filename)
        im = np.dot(np.array(im, dtype='float32'), [[0.2989], [0.5870],
        [0.1140]]) / 255
        dataset[i, :, :, :] = im[:, :, :] i += 1
    labels = np.ones([len(dataset), 1], dtype=int) return dataset, labels
def generate_dataset_closed():
    dataset = np.ndarray([1192 * 2, 24, 24, 1], dtype='float32')
    i = 0
    for dir in dirs2:
        for filename in os.listdir(dir): if filename.endswith('.jpg'):
            im = cv2.imread(dir + '/' + filename)
            im = np.dot(np.array(im, dtype='float32'), [[0.2989], [0.5870],
            [0.1140]]) / 255
            dataset[i, :, :, :] = im[:, :, :] i += 1
    labels = np.zeros([len(dataset), 1], dtype=int) return dataset, labels
dataset_open, labels_open = generate_dataset() dataset_closed,
labels_closed = generate_dataset_closed() print("done")
split = int(len(dataset_closed) * 0.8) train_dataset_closed =
dataset_closed[:split] train_labels_closed = labels_closed[:split]
test_dataset_closed = dataset_closed[split:] test_labels_closed =
labels_closed[split:]
pickle_file = 'closed_eyes.pickle'
try:
    f = open(pickle_file, 'wb') save = {
        'train_dataset': train_dataset_closed, 'train_labels': train_labels_closed,
        'test_dataset': test_dataset_closed, 'test_labels': test_labels_closed,
    }
    pickle.dump(save, f, pickle.HIGHEST_PROTOCOL) f.close()

```

```

except Exception as e:
    print ('Unable to save data to', pickle_file, ':', e) raise
    statinfo = os.stat(pickle_file)
    print ('Compressed pickle size:', statinfo.st_size)
    split = int(len(dataset_open) * 0.8) train_dataset_open = dataset_open[: split]
    train_labels_open = labels_open[: split] test_dataset_open =
    dataset_open[split:] test_labels_open = labels_open[split:
    pickle_file = 'open_eyes.pickle'
    try:
        f = open (pickle_file, 'wb') save = {
        'train_dataset': train_dataset_open, 'train_labels': train_labels_open,
        'test_dataset': test_dataset_open, 'test_labels': test_labels_open,
        }
        pickle.dump(save, f, pickle.HIGHEST_PROTOCOL) f.close()
    except Exception as e:
        print ('Unable to save data to', pickle_file, ':', e) raise
        statinfo = os.stat(pickle_file)
        print ('Compressed pickle size:', statinfo.st_size)

```

## **EYE.CNN:**

```

from __future__ import absolute_import from __future__ import print_function
import numpy as np
import os
from keras.utils import plot_model
os.environ["PATH"] += os.pathsep + 'C:/Users/gts/Downloads/bin/'
pickle_files = ['open_eyes.pickle', 'closed_eyes.pickle'] i = 0
for pickle_file in pickle_files:
    with open (pickle_file, 'rb') as f: save = pickle.load(f)
    if i == 0:

```

```

train_dataset = save['train_dataset'] train_labels = save['train_labels']
test_dataset = save['test_dataset'] test_labels = save['test_labels']
else:
print("here")
train_dataset = np.concatenate((train_dataset, save['train_dataset']))
train_labels = np.concatenate((train_labels, save['train_labels'])) test_dataset
= np.concatenate((test_dataset, save['test_dataset'])) test_labels =
np.concatenate((test_labels, save['test_labels']))
del save # hint to help gc free up memory
i += 1
print ('Training set', train_dataset.shape, train_labels.shape) print ('Test set',
test_dataset.shape, test_labels.shape)
batch_size = 3
nb_classes =
epochs = 12
X_train = train_dataset
X_train = X_train.reshape((X_train.shape[0], X_train.shape[3]) +
X_train.shape[1:3])
Y_train = train_labels
X_test = test_dataset
X_test = X_test.reshape((X_test.shape[0], X_test.shape[3]) +
X_test.shape[1:3])
Y_test = test_labels
# Print shape of data while model is building
Print ("{1} train samples, {4} channel {0}, {2}x{3}".
format (" " if X_train.shape[1] == 1 else "s", *X_train.shape))
Print ("{1} test samples, {4} channel {0}, {2}x{3}".
format (" " if X_test.shape[1] == 1 else "s", *X_test.shape))
# Input image dimensions
_, img_channels, img_rows, img_cols = X_train.shape
# Convert class vectors to binary class matrices

```

```

# Y_train = np_utils.to_categorical(y_train, nb_classes) # Y_test =
np_utils.to_categorical(y_test, nb_classes)
model = Sequential ()
model.add(Convolution2D(32, (3, 3), padding='same',
input_shape=(img_channels, img_rows,
img_cols),data_format='channels_first'))
model.add(Activation('relu'))
model.add(Convolution2D(24, (3, 3), data_format='channels_first'))
model.add(Activation('relu')) model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))model.add(Convolution2D(64, (3, 3),
padding='same', data_format='channels_first'))
model.add(Activation('relu')) model.add(Convolution2D(64, (3, 3)))
model.add(Activation('relu')) model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten()) model.add(Dense(512)) model.add(Activation('relu'))
model.add(Dropout(0.5)) model.add(Dense(nb_classes))
model.add(Activation('sigmoid'))
# let's train the model using SGD + momentum (how original). sgd = SGD
(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='binary_crossentropy', optimizer=sgd,
metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=batch_size, epochs=epochs,
verbose=2, validation_data=(X_test, Y_test))
score = model.evaluate(X_test, Y_test, verbose=1)
print ('Test score:', score [0]) print ('Test accuracy:', score [1])

```

```

import cv2

np.random.seed(1337) # for reproducibility

from keras.models import Sequential

from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.utils import np_utils

from keras.optimizers import SGD, Adadelta, Adagrad

from six.moves import cPickle as pickle

import tensorflow as tf

import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))

#Os.      environ["PATH"] += os.pathsep + 'C:/Program Files
(x86)/Graphviz2.38/bin/'

os.environ["PATH"] += os.pathsep + 'C:/Users/gts/Downloads/bin/'

pickle_files = ['open_eyes.pickle', 'closed_eyes.pickle'] i = 0

for pickle_file in pickle_files:

    with open (pickle_file, 'rb') as f: save = pickle.load(f)

    if i == 0:

        train_dataset = save['train_dataset'] train_labels = save['train_labels']
        test_dataset = save['test_dataset'] test_labels = save['test_labels']

    else:

        print("here")

```

```

train_dataset = np.concatenate((train_dataset, save['train_dataset']))
train_labels = np.concatenate((train_labels, save['train_labels'])) test_dataset
= np.concatenate((test_dataset, save['test_dataset'])) test_labels =
np.concatenate((test_labels, save['test_labels']))

del save # hint to help gc free up memory

i += 1

print ('Training set', train_dataset.shape, train_labels.shape) print ('Test set',
test_dataset.shape, test_labels.shape)

batch_size = 3

nb_classes =

epochs = 12

X_train = train_dataset

X_train = X_train.reshape((X_train.shape[0], X_train.shape[3]) +
X_train.shape[1:3])

Y_train = train_labels

X_test = test_dataset

X_test = X_test.reshape((X_test.shape[0], X_test.shape[3]) +
X_test.shape[1:3])

Y_test = test_labels

# Print shape of data while model is building

Print ("{1} train samples, {4} channel {0}, {2}x{3}".
format (" " if X_train.shape[1] == 1 else "s", *X_train.shape))

Print ("{1} test samples, {4} channel {0}, {2}x{3}".
format (" " if X_test.shape[1] == 1 else "s", *X_test.shape))

# Input image dimensions

```



```

_, img_channels, img_rows, img_cols = X_train.shape

# Convert class vectors to binary class matrices

# Y_train = np_utils.to_categorical(y_train, nb_classes) # Y_test =
np_utils.to_categorical(y_test, nb_classes)

model = Sequential ()

model.add(Convolution2D(32, (3, 3), padding='same',

input_shape=(img_channels, img_rows,
img_cols),data_format='channels_first'))

model.add(Activation('relu'))

model.add(Convolution2D(24, (3, 3), data_format='channels_first'))
model.add(Activation('relu')) model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, (3, 3), padding='same',
data_format='channels_first'))

model.add(Activation('relu')) model.add(Convolution2D(64, (3, 3)))
model.add(Activation('relu')) model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten()) model.add(Dense(512)) model.add(Activation('relu'))
model.add(Dropout(0.5)) model.add(Dense(nb_classes))
model.add(Activation('sigmoid'))

# let's train the model using SGD + momentum (how original). sgd = SGD
(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

model.fit(X_train, Y_train, batch_size=batch_size, epochs=epochs,
verbose=2, validation_data=(X_test, Y_test))

score = model.evaluate(X_test, Y_test, verbose=1)

print ('Test score:', score [0]) print ('Test accuracy:', score [1])

```

# CHAPTER 9

# CHAPTER 9

## REFERENCES

### REFERENCE PAPERS:

[1] Alshaqaqi, B., Baquhaizel, A. S., Amine Ouis, M. E., Boumehed, M., Ouamri, A., & Keche,

M. 2013. Driver drowsiness detection system. 2013 28th International Workshop on Systems, Signal Processing and Their Applications (WoSSPA).

[2] Baek, J. W., Han, B.-G., Kim, K.-J., Chung, Y.-S., & Lee, S.-I. 2018. Real-Time Drowsiness Detection Algorithm for Driver State Monitoring Systems. 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN).

[3] You, F., Li, X., Gong, Y., Wang, H., & Li, H. 2019. A Real-time Driving Drowsiness Detection Algorithm with Individual Differences Consideration. IEEE Access, vol. 7, pp. 179396-179408.

[4] Dasgupta, A., Rahman, D., & Routray, A. 2018. A Smartphone-Based Drowsiness Detection and Warning System for Automotive Drivers. IEEE Transactions on Intelligent Transportation Systems, 1–10.

[5] Eraldo, B., Quispe, G., Chavez-Arias, H., Raymundo-Ibanez, C., & Dominguez, F. 2019.

Design of a control and monitoring system to reduce traffic accidents due to drowsiness through image processing. 2019 IEEE 39th Central America and Panama Convention (CONCAPAN XXXIX).

- [6] Lashkov, I., Kashevnik, A., Shilov, N., Parfenov, V., & Shabaev, A. 2019. Driver Dangerous State Detection Based on OpenCV & Dlib Libraries Using Mobile Video Processing. 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC).
- [7] Zhang, S., & Wang, X. 2013. Human detection and object tracking based on Histograms of Oriented Gradients. 2013 Ninth International Conference on Natural Computation (ICNC).
- [8] Kamran, M., Mannan, M., & Jeong, Y. 2019. Drowsiness, Fatigue and Poor Sleep's Causes and Detection: A Comprehensive Study. IEEE Access, vol. 7, pp. 167172-167186.
- [9] Zhang, W., Cheng, B., & Lin, Y. 2012. Driver drowsiness recognition based on computer vision technology. Tsinghua Science and Technology, vol. 17, no. 3, pp. 354-362.
- [10] Manu, B. N. 2016. Facial features monitoring for real time drowsiness detection. 2016 12th International Conference on Innovations in Information Technology (IIT).

## **CNN AND SVM-BASED DROWSINESS MONITORING & ALERTNESS SYSTEM**

**Prabhu S 1<sup>a</sup>, Dr.M.Nithya 2<sup>b</sup>, Prakash S 3<sup>c</sup>, Rathiswaran K 4<sup>d</sup>,  
Harikrishana K 5<sup>e</sup>**

*<sup>1,</sup> AssociateProfessor, Paavai College of Engineering, Namakkal, Tamilnadu.*

*<sup>2,</sup> Professor, Vinayaka Mission Kirupananda Variyar Engineering College,  
Vinayaka Mission Research Foundation(DU),,Salem, Tamilnadu.*

*<sup>3,4,5</sup> Paavai College of Engineering, Namakkal, Tamilnadu.*

### **Abstract**

Drivers fatigued driving is a significant and latent danger in traffic accidents are a serious issue globally resulting in thousands of deaths annually, research has shown that drowsy driving reduces reaction time lowers consciousness and heightens the risk of accidents. Inorder to counter this serious problem a detection system based on machine learning is implemented to track drivers by continuous monitoring and warn them before they doze off while driving. This system takes advantage of opencv and deep learning algorithms to monitor facial expressions such as eye closure, blinking rate, yawning and head orientation other sensor-based data like lane distance behavior can be fused to achieve more precise predictions by using sophisticated algorithms. Such as ‘Convolutional Neural Networks (CNNs)’ and ‘Support Vector Machines (SVMs)’. The system is capable of distinguishing alert from drowsy conditions with great precision the suggested solute.ion works by obtaining a live video feed of the face of the driver computing major landmarks which are in our face and measuring head pose levels in the event that the system picks up signs of tiredness it initiates alerts in the form of audio alarms steering wheel vibrations or visually alarms driver. This project is intended to enhance road safety through minimizing accidents resulting from drowsy driving the system is conceived to be efficient cost-saving and able to perform in continuos monitoring thus fitting into contemporary cars and fleet management systems.

**Keywords**

Driver Alertness Monitoring System, Machine Learning, Convolutional Neural Networks, Support Vector Machines, Real-Time Monitoring, Road Safety.

**1] Introduction**

Drowsy driving is one of the main causes of accidents on roads across the globe. the decreased alertness of a driver can cause delayed responses and poor decision-making, and therefore it is imperative to create systems that can detect and react to drowsiness in real-time. Current measures like lane detection or physical alert sensors are constrained by their precision and real-time ability. This paper suggests a system that employs sophisticated machine learning methods to overcome these limitations.

**2] Literature Review**

Existing works have employed multiple machine learning strategies for detecting drowsiness among drivers. Rule-based methods depend on thresholded features like closure of the eye, but are not adaptive in nature. Deep learning models, particularly CNNs, have achieved improved performance when extracting features from facial images. Environmental variations and occlusions remain problematic, nonetheless. Hybrid schemes that integrate CNNs and SVMs have presented potential for achieving higher accuracy.

**Title:** An Integrated System for Drivers' Drowsiness Detection Using Deep Learning Frameworks.

**Authors:** Biswarup Ganguly, Debangshu Dey, Sugata Munshi.

**Abstract:** This paper presents a real-time driver drowsiness detection system using deep learning. It integrates Faster R-CNN for eye region detection and CNN for eye state classification to determine drowsiness levels. An Atmega328p microcontroller is used to implement an alert system, ensuring timely warnings to prevent accidents caused by drowsy driving.

**Date Published:** 26-27 February 2022

**Title:** A Systematic Review on Driver Drowsiness Detection Using Eye Activity Measures.

**Authors:** AHMET KOLUS

**Abstract:** This study systematically reviews driver drowsiness detection (DDD) systems based on eye activity measures, which are effective in early drowsiness detection. By analyzing 41 empirical studies, it classifies eye activity indicators, measurement technologies, and decision-making algorithms used for drowsiness prediction. The findings provide insights for future research and development of more effective DDD systems.

**Date Published:** 8 July 2024

**Title:** Driver Drowsiness Detection and Monitoring System using Machine Learning

**Authors:** Rohith Chinthalachervu, Immaneni Teja, M. Ajay Kumar, N. Sai Harshith, T. Santosh Kumar

**Abstract:** This research presents a real-time driver drowsiness detection system using facial expression analysis and machine learning. The system captures facial movements via a webcam and calculates Eye Aspect Ratio, Mouth Opening Ratio, and Nose Length Ratio for drowsiness detection. A Support Vector Machine (SVM) model achieves 95.58% sensitivity and 100% specificity, ensuring high accuracy. The system is cost-effective, does not require expensive sensors, and is compatible with all vehicle types.

**Date Published:** 01 September 2022

**Title:** A Deep Learning Approach To Detect Driver Drowsiness

**Authors:** Madhav Tibrewal, Aayush Srivastava, Dr. R. Kayalvizhi

**Abstract:** This paper presents a driver drowsiness detection system that analyzes the driver's eye state to detect drowsiness and issue timely alerts, helping prevent accidents. With 40% of road accidents caused by drowsy driving (as per CRRI), this system aims to enhance road safety by identifying fatigue early and notifying drivers before any serious risks occur.

**Date Published:** 05 May-2021

**Title:** Driver Drowsiness Detection Using Machine Learning

**Authors:** Manoj Rode, Anirudh Bethi, Santosh Baddam, Bala Laxmi Prasanna Kondaveti, Tejaswini Gadipally, Naresh Katroth

**Abstract:** This project develops a real-time driver drowsiness detection system that continuously monitors the driver using a webcam and analyzes eye blinking patterns with OpenCV. If the driver's eyes remain closed for more than two frames, the system detects drowsiness and triggers an alarm alert, helping prevent fatigue-related accidents and enhance road safety.

**Date Published:** 22 November 2022

**Title:** Early Identification and Detection of Driver Drowsiness by Hybrid Machine Learning

**Authors:** Ayman Altameem, Ankit Kumar, Ramesh Chandra Poonia, Sandeep Kumar, Abdul Khade Rjilani Saudagar

**Abstract:** This paper presents a real-time driver monitoring system that detects drowsiness and emotional changes using machine learning. The system employs facial expression analysis with Support Vector Machines (SVM) to identify signs of fatigue, anger, or inattention. Upon detection, it alerts the driver and can slow down the vehicle for safety. Tested under variable lighting conditions, the model achieved 83.25% accuracy, improving existing detection methods.

**Date Published:** 30 November 2021

**Title:** Driver Drowsiness Detection System – An Approach By Machine Learning Application

**Authors:** Jagbeer Singh , Ritika Kanojia , Rishika Singh , Rishita Bansal , Sakshi Bansal

**Abstract:** This research focuses on developing a driver drowsiness detection system using face and eye tracking to prevent road accidents. The system extracts eye images, compares them with a dataset, and triggers an alarm if the eyes remain closed beyond a threshold. If the driver responds, tracking continues; otherwise, the alert persists. The model achieves 80% accuracy, helping reduce fatigue-related accidents and improve road safety.

**Date Published:** 31 DEC 2022

### 3] Methodology



## Visual Tracking Object For Input

The visual camera employed for a driver drowsiness warning system with night mode observation needs to be an infrared (IR) or near-infrared (NIR) high-resolution camera with a minimum resolution of 720p (1280x720 pixels) for sharp facial feature recognition. It must provide a minimum frame rate of 30 fps for real-time detection of eye closure, yawning and head movements, the camera should be equipped with a wide dynamic range WDR and low-light sensitivity to perform well in both daylight and night environments for nighttime monitoring an IR-cut filter removal functionality or NOIR (No Infrared Filter) camera should be implemented enabling enhanced visibility in the dark. Furthermore the camera should possess IR led illuminators 850nm or 940nm wavelength in order to deliver improved facial recognition in pure darkness without inducing driver discomfort. A global shutter sensor is used over a rolling shutter to minimize the motion blur in order to ensure proper tracking of rapid eye blinking and head nods the field of view (fov) must be set between 60 and 90 so the entire face of the driver is being captured even on slight movements for embedded systems it is advisable to use a USB or MIPI interface for simple integration with raspberry pi, jetson nano or other edge computing hardware.

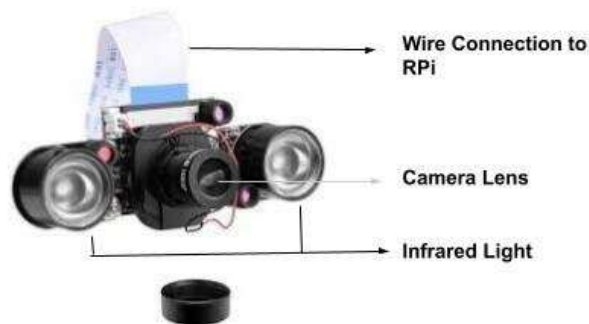


Fig1 camera module used in the system

**Datasets:** images and videos of drivers were obtained from public datasets such as yawdd that contain labeled samples of sleepy and wakeful states

**Preprocessing:** methods like grayscale conversion cropping and normalization were used to provide uniform input data quality.

**Feature Extraction:**

Important indicators are landmark points on face for drowsiness detection based on facial landmarks the most important points are around the eyes nose and mouth through which eye closure duration head pose and other facial features can be analyzed to decide whether a person is drowsy or not the human face consists of 68 key landmarks it is used to identifies the face and as well as used to monitor it by including various formulas.

### Diagram:

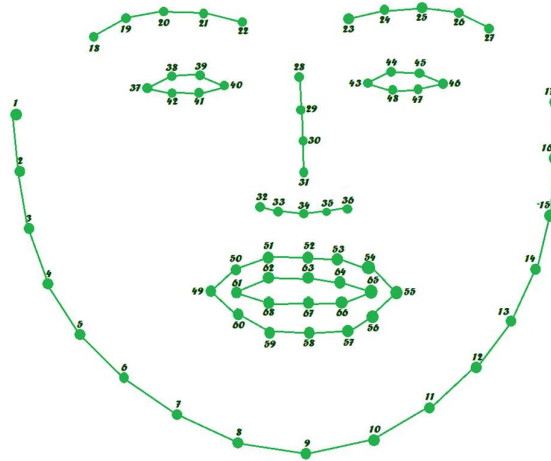


Fig 2 Shows the 68 Facial Landmarks

**Eye Aspect Ratio (EAR):** computation of distances between certain eye points to recognize abnormal eye closure.

### Formula

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

### Model

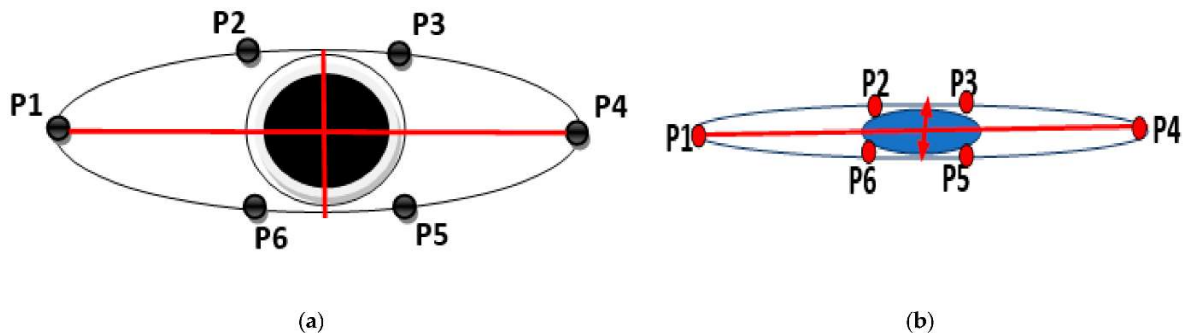


Fig3 shows EAR formula working process

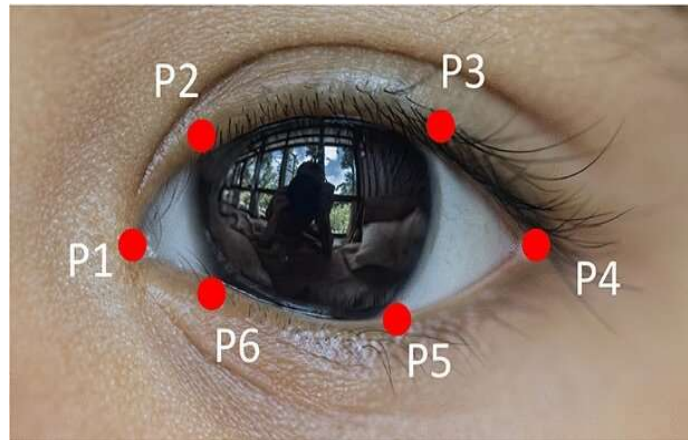


Fig4 Shows the Real Time monitoring of Eye by using EAR Formula

**Mouth Opening Ratio (MOR):** calculation of upper and lower lip distance to detect yawning.

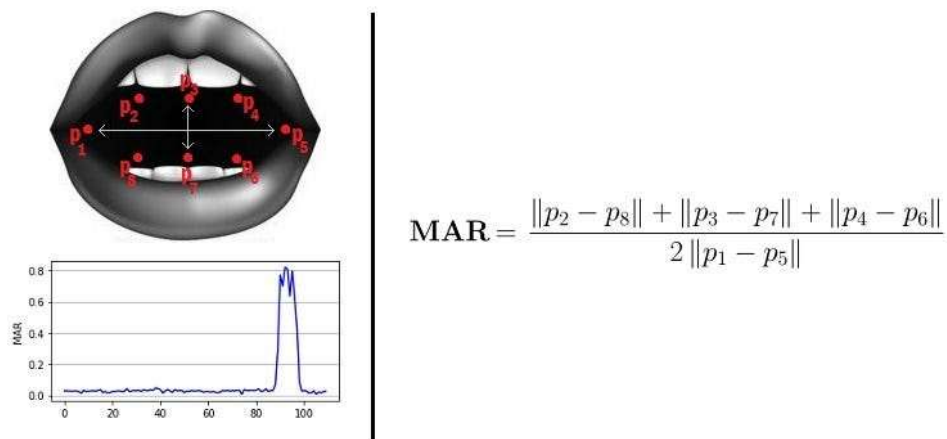


Fig 5 shows the mor formula with the pictorial presentation

**Head Pose Estimation:** key point detection for recognition of head tilting or head nodding.

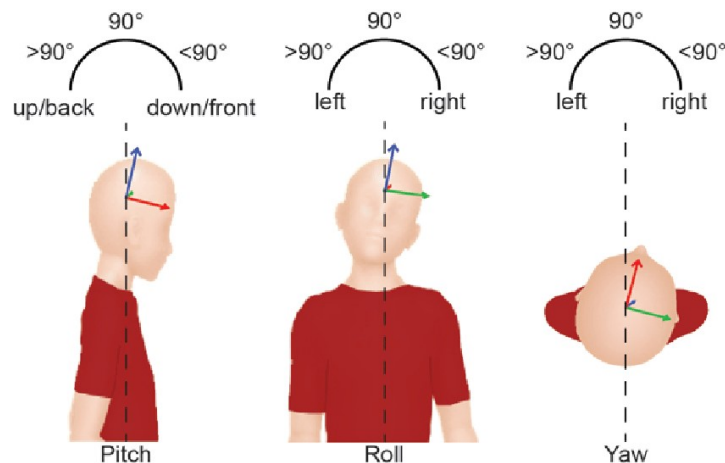


Fig 6 shows the head position monitoring with angular presentation

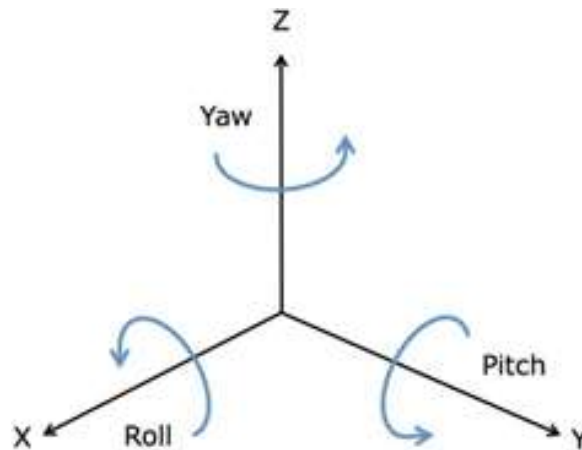


Fig7 shows the x,y,z co-ordinates for roll,pitch and yaw

### Formula

**Formulas for Yaw, Pitch, Roll from Rotation Matrix  $R$ :**

$$\text{Yaw} = \arctan\left(\frac{r_{32}}{r_{33}}\right)$$

$$\text{Pitch} = \arctan\left(-\frac{r_{31}}{\sqrt{r_{32}^2 + r_{33}^2}}\right)$$

$$\text{Roll} = \arctan\left(\frac{r_{21}}{r_{11}}\right)$$

## 3.2 Algorithms And Models Utilized

### Convolutional Neural Networks (CNNs):

CNNs were employed for feature extraction the structure had convolutional pooling and fully connected layers for recognizing visual patterns such as eye closure and yawning from images.

### Support Vector Machines (SVMs):

SVMs labeled the features extracted as belonging to two classes alert and drowsy a radial basis function rbf kernel was employed for dealing with nonlinearity separability.

## Hybrid Model

By integrating cnn-based feature extraction with svm classification the system attained improved detection accuracy and fewer false positives.

### 3.3 workflow

The workflow consists of the following steps

1. **Image/Video Input:** real-time video feed from a camera mounted on a dashboard
2. **Preprocessing Frames:** are preprocessed to normalize lighting conditions and remove noise
3. **Feature Extraction:** facial features eg eye landmarks mouth movements are extracted using cnn
4. **Classification:** svm model classifies the driver as drowsy or alert
5. **Alert Mechanism:** when drowsiness is recognized the system provides audible and visual warnings

## 4] System Architecture And Flow Diagram:

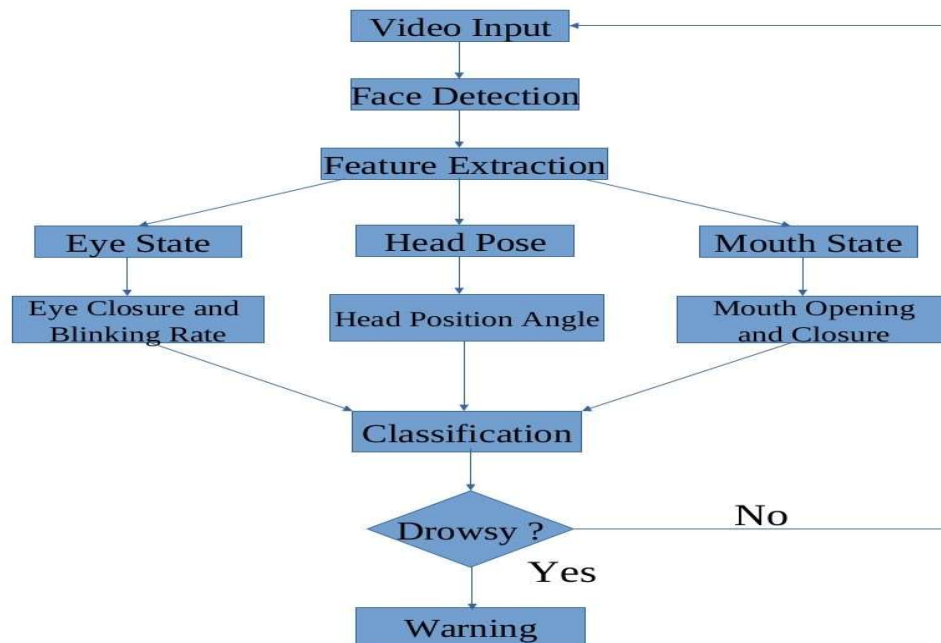


fig8 shows the architecture flow of the system working process  
flow diagram description

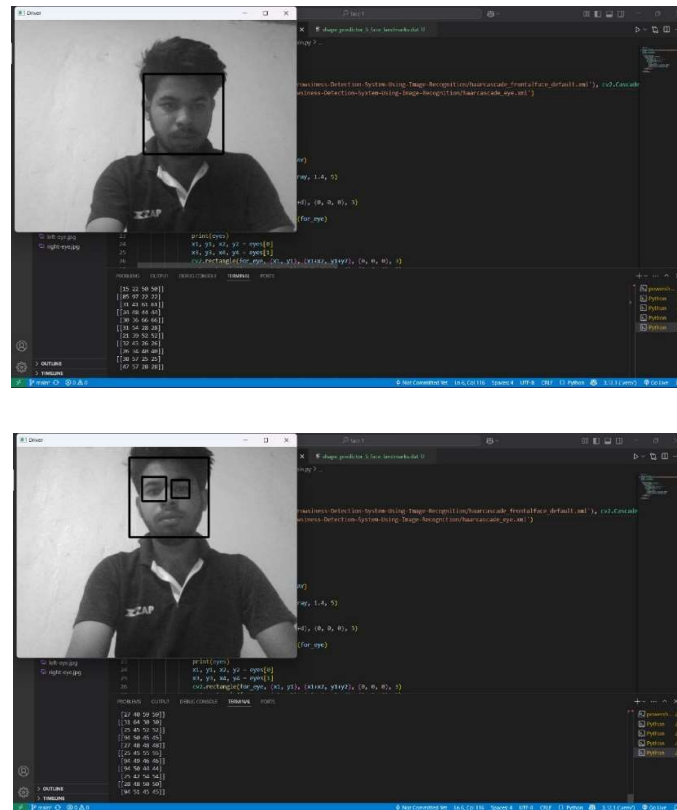
The flow diagram describes the system workflow from data acquisition to generating an alert the following are the breakdown of each component

1. **Camera Module:** grabs real-time images or video of the driver

2. **Preprocessing Unit:** preprocesses the data to improve quality eg resizing noise removal
  3. **Feature Extraction Module:** extracts the appropriate features such as eye and mouth landmarks using a cnn model
  4. **Classification Module:** uses an svm to classify the state of the driver alert/drowsy
  5. **Decision Module:** sends an alert if the result of classification shows drowsiness
- [Camera] → [Preprocessing] → [Feature Extraction (CNN)] → [Classification (SVM)] → [Alert System]

## 5] Results And Discussion

The system was evaluated using accuracy precision recall and f1-score metrics it achieved the following results



Accuracy-96 %  
 Precision-93%  
 Recall-96%  
 F1-Score-94%

The hybrid cnn-svm model performed better than isolated models demonstrating remarkable robustness under different lighting conditions however occlusion issues (e.g., sunglasses) are still areas of future work.

## 6] Conclusion

this work proposed a driver drowsiness detection system based on machine learning with cnn and svm models the system accurately detects drowsiness in real time and sends an alert to the driver minimizing the occurrence of road accidents future research will consider integrating physiological signs such as heart rate and enhancing performance in harsh environmental conditions

## 7] Referance

1. Rohith Chinthalachervu, Immaneni Teja, M. Ajay Kumar, N. Sai Harshith and T. Santosh Kumar “Driver Drowsiness Detection And Monitoring System Using Machine Learning” Journal of Physics: Conference Series, Volume 2325, International Conference on Electronic Circuits and Signalling Technologies 02/06/2022 - 03/06/2022 Online
2. Biswarup Ganguly, Debangshu Dey, Sugata Munshi “An Integrated System for Drivers’ Drowsiness Detection Using Deep Learning Frameworks” IEEE VLSI Device, Circuit and System Conference 2022 (IEEE VLSI DCS 2022), 26-27 February 2022, IEEE ED MSIT SBC, Kolkata, India.
3. Manoj Rode, Anirudh Bethi, Santosh Baddam, Bala Laxmi Prasanna Kondaveti, Tejaswini Gadipally, Naresh Katroth “Driver Drowsiness Detection Using Machine Learning” International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 10 Issue XI Nov 2022.
4. Ayman Altameem, Ankit Kumar, Ramesh Chandra Poonia, Sandeep Kumar, Abdul Khade Rjilani Saudagar “Early Identification and Detection of Driver Drowsiness by Hybrid Machine Learning” Digital Object Identifier 10.1109/ACCESS.2021.3131601 November 30, 2021.
5. Jagbeer Singh, Ritika Kanojia, Rishika Singh, Rishita Bansal, Sakshi Bansal “Driver Drowsiness Detection System – An Approach By Machine Learning Application” DOI: 10.47750/pnr.2022.13.S10.361 Journal of Pharmaceutical Negative Results, Volume 13 Special Issue 10 2022.



6. Madhav Tibrewal, Aayush Srivastava, Dr. R. Kayalvizhi “A Deep Learning Approach To Detect Driver Drowsiness” International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 May-2021.
7. S. Sharan, R. Reddy and P. Reddy, "Multi-level Drowsiness Detection using Multi-Contrast Convolutional Neural Networks and Single Shot Detector," proc. International Conference on Intelligent Technologies (CONIT), 2021.
8. M. Ngxande, J.-R. Tapamo, and M. Burke, “Bias remediation in driver drowsiness detection systems using generative adversarial networks,” IEEE Access, vol. 8, pp. 55592–55601, 2020.
9. M. Ramzan, H. U. Khan, S. M. Awan, A. Ismail, M. Ilyas, and A. Mahmood, A survey on state-of-the-art drowsiness detection techniques, IEEE Access, vol. 7, pp. 6190461919, 2019, doi: 10.1109/ACCESS.2019.2914373.
10. S. Johnpaul, C. T. Selvan and P. J. Raguraman, "IoT based Smart Helmet System for Accident Prevention," 2022 International Conference on Edge Computing and Applications (ICECAA), Tamilnadu, India, 2022, pp. 599-602, doi: 10.1109/ICECAA55415.2022.9936313.
11. R. Sathya, P. J. Raguraman, S. Thavamaniyan, S. Ananthi, T. Kathirvel and T. S. Abdul Razack, "Real Time Implementation of Driver Drowsiness Detection for an Intelligent Transportation System," 2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), Bengaluru, India, 2024, pp. 204-208, doi: 10.1109/IDCIoT59759.2024.10467766.



# GRADIVA REVIEW JOURNAL

An UGC-CARE Approved Group-II Journal

ISSN NO : 0363-8057 / Website : <http://gradivareview.com/>

Email : [Submitgrjournal@gmail.Com](mailto:Submitgrjournal@gmail.Com)



## *Certificate of Publication*

Paper ID : GRJ/8070

This is to certify that the paper titled

Cnn And Svm-based Drowsiness Monitoring & Alertness System

Authored by

Prabhu S

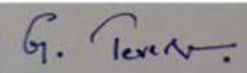
From

Paavai College of Engineering, Namakkal, Tamilnadu

Has been published in

**GRADIVA REVIEW JOURNAL Volume 11, Issue 4 , April 2025.**



  
**Teresa Gallart**  
**Editor-in-Chief**  
**Gradiva Review Journal**



# GRADIVA REVIEW JOURNAL

An UGC-CARE Approved Group-II Journal

ISSN NO : 0363-8057 / Website : <http://gradivareview.com/>

Email : [Submitgrjournal@gmail.com](mailto:Submitgrjournal@gmail.com)



## *Certificate of Publication*

Paper ID : GRJ/8070

This is to certify that the paper titled

Cnn And Svm-based Drowsiness Monitoring & Alertness System

Authored by

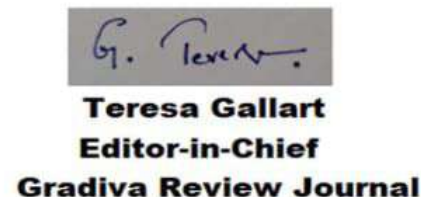
Dr.M.Nithya

From

Vinayaka Mission Kirupananda Variyar Engineering College

Has been published in

**GRADIVA REVIEW JOURNAL Volume 11, Issue 4 , April 2025.**





# GRADIVA REVIEW JOURNAL

An UGC-CARE Approved Group-II Journal

ISSN NO : 0363-8057 / Website : <http://gradivareview.com/>

Email : [Submitgrjournal@gmail.Com](mailto:Submitgrjournal@gmail.Com)



## *Certificate of Publication*

Paper ID : GRJ/8070

This is to certify that the paper titled

Cnn And Svm-based Drowsiness Monitoring & Alertness System

Authored by

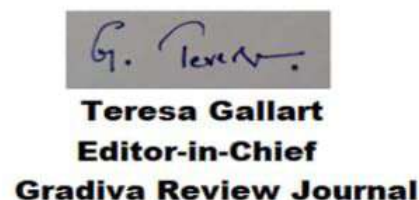
Rathiswaran K

From

Paavai College of Engineering, Namakkal, Tamilnadu

Has been published in

**GRADIVA REVIEW JOURNAL Volume 11, Issue 4 , April 2025.**



# GRADIVA REVIEW JOURNAL

An UGC-CARE Approved Group-II Journal

ISSN NO : 0363-8057 / Website : <http://gradivareview.com/>

Email : [Submitgrjournal@gmail.com](mailto:Submitgrjournal@gmail.com)



## *Certificate of Publication*

Paper ID : GRJ/8070

This is to certify that the paper titled

Cnn And Svm-based Drowsiness Monitoring & Alertness System

Authored by

Prakash S

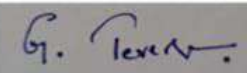
From

Paavai College of Engineering, Namakkal, Tamilnadu

Has been published in

**GRADIVA REVIEW JOURNAL Volume 11, Issue 4 , April 2025.**



  
**Teresa Gallart**  
**Editor-in-Chief**  
**Gradiva Review Journal**





# GRADIVA REVIEW JOURNAL

An UGC-CARE Approved Group-II Journal

ISSN NO : 0363-8057 / Website : <http://gradivareview.com/>

Email : [Submitgrjournal@gmail.Com](mailto:Submitgrjournal@gmail.Com)



## *Certificate of Publication*

Paper ID : GRJ/8070

This is to certify that the paper titled

Cnn And Svm-based Drowsiness Monitoring & Alertness System

Authored by

Harikrishana K

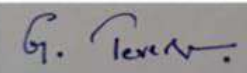
From

Paavai College of Engineering, Namakkal, Tamilnadu

Has been published in

**GRADIVA REVIEW JOURNAL Volume 11, Issue 4 , April 2025.**



  
**Teresa Gallart**  
**Editor-in-Chief**  
**Gradiva Review Journal**

