# ML-BASED EYE-BLINKING TRACKING FOR

# FATIGUE DETECTION USING VISUAL DATA

A PROJECT REPORT

***Submitted by***

**Arpita Singh (221B089)**

**Shruti Bhargava (221B374)**

**Snehil Sharma (221B387)**

**Under the guidance of: Prof. Sanjay Garg**

****

Nov - 2024

***Submitted in partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**IN**

COMPUTER SCIENCE & ENGINEERING  
**Department of Computer Science & Engineering  
JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY  
AB ROAD, RAGHOGARH, DT. GUNA-473226 MP, INDIA**

# Declaration by the Students

We hereby declare that the work reported in the B. Tech. project entitled as “**EYE BLINKING TRACKING FOR FATIGUE DETECTION**”, in partial fulfillment for the award of degree of Bachelor of Technology submitted at **Jaypee University of Engineering and Technology, Guna**, as per best of our knowledge and belief there is no infringement of intellectual property right and copyright. In case of any violation, we will solely be responsible.

Arpita Singh (221B089)

Shruti Bhargava (221B374)

Snehil Sharma (221B387)

Department of Computer Science and Engineering

Jaypee University of Engineering and Technology  
Guna, M.P., India

Date:20/11/2024

 **JAYPEE UNIVERSITY OF ENGINEERING &TECHNOLOGY**

**Grade ‘A+’** Accredited with by NAAC & Approved U/S 2(f) of the UGC Act, 1956

A.B. Road, Raghogarh, Dist.: Guna (M.P.) India, Pin-473226

Phone: 07544 267051, 267310-14, Fax: 07544 267011

Website: www.juet.ac.in

# CERTIFICATE

This is to certify that the work titled “**EYE BLINKING TRACKING FOR FATIGUE DETECTION**” submitted by “**Arpita Singh (221B089), Shruti Bhargava (221B374), Snehil Sharma (221B387)**” in partial fulfillment for the award of degree of B.Tech of **Jaypee University of Engineering & Technology, Guna** has been carried out under my supervision. As per best of my knowledge and belief there is no infringement of intellectual property right and copyright. Also, this work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma. In case of any violation concern student will solely be responsible.

Signature of Supervisor

**Prof. Sanjay Garg**

Head of Department (CSE)

Date:20/11/2024

# ACKNOWLEDGEMENT

First and foremost, we would like to express our gratitude to Jaypee University of Engineering and Technology, our college, for giving us this chance to share our collective talents. We would also like to extend our sincere gratitude to Prof. Sanjay Garg, our project mentor and supervisor, who has supported us during all the challenges and issues we encountered enroute to finishing our project. We also want to express our gratitude to our mentor, who spent valuable time proofreading and fixing our mistakes in addition to offering us solutions and pointing us in the right direction. We also acknowledge the invaluable assistance provided by our parents and friends in helping us to complete this project on time. To cap it off, we extend our heartfelt thanks to the team members of “EYE BLINKING TRACKING FOR FATIGUE DETECTION.”

**Arpita Singh (221B089)**

**Shruti Bhargava (221B374)**

**Snehil Sharma (221B387)**

Date: 20/11/2024

**EXECUTIVE SUMMARY**

As long-haul driving becomes increasingly common, the risk of accidents caused by driver fatigue has also risen. Drowsy driving is a significant contributor to road accidents, leading to countless injuries and fatalities each year. Understanding the urgency of this issue, our project focuses on developing a system that tracks eye blinking to detect fatigue in drivers. By harnessing the power of Convolutional Neural Networks (CNN) for model training and Haar Cascade algorithms for real-time monitoring, we aim to create a practical tool that enhances road safety for everyone.

The motivation behind our project is deeply rooted in the troubling statistics surrounding drowsy driving. Factors such as long shifts, irregular sleep, and monotonous driving conditions can exacerbate fatigue, making it essential to monitor driver alertness continuously. Our system addresses this challenge by focusing on eye-blinking patterns, which are reliable indicators of a driver’s level of fatigue. By training our CNN model on a diverse dataset of eye images, we enable the system to accurately assess various states of alertness.

For real-time implementation, we employ the Haar Cascade algorithm, allowing our system to efficiently detect faces and eyes. To further enhance our system's effectiveness, we have integrated an alarm mechanism that activates after detecting a specified number of consecutive frames indicating drowsiness. By setting a threshold for these frames, the alarm will beep only after confirming persistent signs of fatigue, providing timely alerts that encourage drivers to take necessary breaks.

The social impact of our project is profound. By helping to reduce the frequency of drowsy driving incidents, we can contribute to lowering accident rates, ultimately saving lives and preventing injuries. Our system can be especially beneficial for professional drivers, such as truck and bus operators, who often face long hours on the road. Implementing this technology can lead to better working conditions and a heightened sense of responsibility for road safety.

In summary, our eye-blinking tracking system for fatigue detection represents a significant step in tackling the serious issue of drowsy driving. By combining CNN and Haar Cascade technologies with an integrated alarm system, we have created a solution that not only prioritizes driver safety but also contributes to the broader goal of making our roads safer for everyone.

# List of Figures

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Figure No.** | **Figure Name** | **Page No.** |
| 1. | 4.1.1.1 | Haar Cascade approach | 12 |
| 2. | 4.2.3.1 | CNN Architecture | 16 |
| 3. | 4.3.2.1 | Implementation Flowchart | 19 |
| 4. | 5.2.1.1 | ROC Curve | 22 |
| 5. | 5.2.2.1 | Confusion matrix | 22 |

# Table of Contents

[EYE BLINKING TRACKING FOR FATIGUE DETECTION i](#_Toc182960850)

[Declaration by the Students ii](#_Toc182960851)

[CERTIFICATE iii](#_Toc182960852)

[ACKNOWLEDGEMENT iv](#_Toc182960853)

[Table of Contents vi](#_Toc182960854)

[Chapter 1: Introduction 1](#_Toc182960855)

[1.1 Problem Statement 1](#_Toc182960856)

[1.2 Motivation for Work 1](#_Toc182960857)

[1.3 Goal and Objectives 3](#_Toc182960858)

[1.4 Project Overview 3](#_Toc182960859)

[Chapter 2: Literature Survey 4](#_Toc182960860)

[2.1 Overview of Existing Research 4](#_Toc182960861)

[2.2 Summary 5](#_Toc182960862)

[Chapter 3: System Requirements 6](#_Toc182960863)

[3.1 Hardware Requirements 6](#_Toc182960864)

[3.2 Software Requirements 7](#_Toc182960865)

[3.2.1 Operating Systems 7](#_Toc182960866)

[3.2.2 Programming Language 7](#_Toc182960867)

[3.2.3 Python Libraries 7](#_Toc182960868)

[3.2.4 Model Requirements 9](#_Toc182960869)

[3.2.5 Real Time Detection Requirements 9](#_Toc182960870)

[3.2.6 Performance and Optimization Requirements 10](#_Toc182960871)

[3.2.7 Jupyter Notebook 10](#_Toc182960872)

[3.2.8 PyCharm 11](#_Toc182960873)

[Chapter 4: Design and Implementation 12](#_Toc182960874)

[4.1 Proposed System and Advantages 12](#_Toc182960875)

[4.1.1 Proposed System 12](#_Toc182960876)

[4.1.2 Advantages 13](#_Toc182960877)

[4.2 Methodology 14](#_Toc182960878)

[4.2.1 Data Collection 14](#_Toc182960879)

[4.2.2 Data Preprocessing 14](#_Toc182960880)

[4.2.3 Model Selection 15](#_Toc182960881)

[4.3 System Design 17](#_Toc182960882)

[4.2.3 System Architecture 17](#_Toc182960883)

[4.3.2 Component Interaction 19](#_Toc182960884)

[Chapter 5: Result and Discussion 20](#_Toc182960885)

[5.1 Performance Metrics 20](#_Toc182960886)

[5.1.1. Accuracy 20](#_Toc182960887)

[5.1.2. Precision 20](#_Toc182960888)

[5.1.3. Recall 20](#_Toc182960889)

[5.1.4.F1-Score 21](#_Toc182960890)

[5.1.5. Confusion Matrix 21](#_Toc182960891)

[5.1.6. Real-Time Inference Time 21](#_Toc182960892)

[5.1.7. False Positive Rate 21](#_Toc182960893)

[5.2 Result Analysis 22](#_Toc182960894)

[5.2.1 ROC Curve 22](#_Toc182960895)

[5.2.2 Confusion Matrix 22](#_Toc182960896)

[5.3 Screenshots 23](#_Toc182960897)

[23](#_Toc182960898)

[Conclusion 24](#_Toc182960899)

[References 25](#_Toc182960900)

[Contributors 26](#_Toc182960901)

# Chapter 1

**INTRODUCTION**

## 1.1 Problem Statement

Drowsy driving is a serious safety concern that puts everyone on the road at risk. Each year, countless accidents, injuries, and fatalities occur due to drivers who are too tired to stay alert. Research shows that fatigue can impair a driver’s ability to think clearly and react quickly, often to a level similar to being under the influence of alcohol. This is particularly alarming for long-haul drivers, who often face long hours behind the wheel, irregular sleep schedules, and monotonous driving conditions.

Unfortunately, many drivers don’t realize how fatigued they are until it’s too late, which can lead to devastating consequences. Existing methods for detecting fatigue, such as relying on drivers to self-assess their alertness or taking periodic breaks, are often inadequate and can be easily overlooked. There’s a clear need for a more effective solution that can monitor driver alertness in real time and provide immediate feedback to help prevent accidents.

## 1.2 Motivation for Work

The motivation for developing a Driver Drowsiness Detection System arises from the growing need for better road safety, fewer accidents, and utilizing modern technology to the fullest potential to ensure improved driving conditions. This work is backed by the following major reasons:

* **Enhancing Road Safety**: Every day, countless people get behind the wheel, often unaware of the dangers posed by drowsy driving. With studies showing that fatigue contributes to about 20% of all road accidents, our project is driven by a simple yet powerful goal: to make our roads safer for everyone. We want to help prevent those moments when a driver’s eyelids grow heavy and their focus drifts, potentially leading to devastating consequences.
* **Saving Lives**: Behind every statistic is a story—a family member, a friend, a loved one. Thousands of lives are lost each year due to drowsy driving, and each loss is a tragedy that ripples through communities. Our motivation is deeply personal: we want to create a system that can alert drivers when they’re at risk of falling asleep, giving them a chance to pull over and rest. It’s about protecting lives and ensuring that everyone can return home safely.
* **Economic Implications**: The impact of drowsy driving extends beyond the immediate tragedy of accidents; it also carries a heavy financial burden. Medical bills, property damage, and lost productivity add up quickly, affecting not just individuals but entire communities. By reducing the number of accidents caused by fatigue, we aim to alleviate some of these economic pressures, making roads safer and more cost-effective for everyone.
* **Advancement of Technology**: We live in an age of incredible technological advancements. With tools like artificial intelligence and machine learning at our fingertips, we have the opportunity to create innovative solutions that can genuinely make a difference. Our project harnesses these technologies to develop a drowsiness detection system that is not only effective but also easy to use, ensuring that it can be integrated into everyday driving experiences.
* **Usability Focus**: We understand that for any safety feature to be effective, it must fit seamlessly into a driver’s routine. Our system is designed to work quietly in the background, allowing drivers to focus on the road ahead. We want to create a solution that feels natural and unobtrusive, providing alerts only when necessary, so drivers can feel secure without being distracted.
* **Fostering a Culture of Safety**: Beyond technology, we believe in the power of education and awareness. Our project aims to spark conversations about the dangers of drowsy driving and encourage responsible behavior behind the wheel. By informing drivers about the risks associated with fatigue, we hope to empower them to make safer choices, fostering a culture where safety is a shared priority.
* **Scalability and Impact**: The beauty of our technology lies in its versatility. Whether it’s a family car or a commercial fleet, our drowsiness detection system can be adapted to various vehicles, reaching a wide audience. We envision a future where this technology is commonplace, helping drivers everywhere stay alert and safe.
* **Shared Responsibility**: Road safety is a collective effort. We want to inspire a sense of shared responsibility among drivers, encouraging everyone to prioritize alertness and safety. By working together, we can create a safer driving environment for all.
* **Long-term Vision**: Our vision extends beyond immediate solutions; we aim for lasting change. By significantly reducing drowsy driving incidents, we hope to contribute to a future where road safety is a given, not a goal.

## 1.3 Goal and Objectives

The primary goal of this project is to make our roads safer by creating a real-time eye-blinking tracking system that can detect when drivers are drowsy. We understand that fatigue can sneak up on anyone, especially during long drives, and we want to help prevent accidents caused by drowsy driving. By using advanced technologies like Convolutional Neural Networks (CNN) and Haar Cascade algorithms, our system will monitor driver alertness and send timely alerts when it detects signs of fatigue. Our hope is that by encouraging drivers to take breaks when needed, we can significantly reduce the number of accidents related to drowsiness and promote a culture of safety on the road.

## 1.4 Project Overview

Driver drowsiness detection is one of the car safety technologies aimed at preventing car accidents attributed to the driver's drowsiness.

A deep learning Architecture detects the face and eyes based on the status of the eyes. In case the eyes are closed more than usual time, it generates an alarm intimating the driver.

Because of neglecting our duties toward safer traveling, hundreds of thousands of tragedies have been associated with this great invention every year. To monitor and prevent such negligence from causing a destructive outcome, many researchers have written papers on driver drowsiness detection systems. However, at certain times, some of the points and observations the system presents are not accurate enough. Hence, to bring data and another perspective concerning the problem being dealt with, and improving their implementations and further optimizing the solution, this project was done.

# Chapter 2

**LITERATURE SURVEY**

## 2.1 Overview of Existing Research

This section introduces the background and research context for driver drowsiness detection systems. It discusses the significance of drowsiness detection in reducing accidents and improving road safety, focusing on the integration of machine learning and computer vision. There are some research on Driver’s Drowsiness Detection, for the proper outcome of the subject and usage of it. The researches use different approaches for the application and the requirement processes.

**Drowsiness Detection Based On Driver Temporal Behaviour** (31-March 2021, F. Faraji, F. Lotfi, J. Khorramdel, A. Najafi, A. Ghaffari) -

In this research YOLOv3 CNN is applied as a pretrained network, which is proved to be utilized as a powerful means for object detection.LSTM (Long-Short Term Memory) neural network is employed to learn driver temporal behaviors including yawning and blinking time period as well as sequence classification. One of the main factors of the temporal behavior is that the driver becomes gradually diverted from the road and road traffic. Hence detection is not always accurate.

**A Survey on State of The Art Driver Drowsiness Detection Techniques**(1st December 2020, FHikmat Ullah Khan) –

The detection system includes the processes of face image extraction, yawning tendency, blink of eyes detection, eye area extraction etc. The percentage of the eyelid closure of the algorithms over the pupil over time is relatively very low.

**Driver Drowsiness Detection**(21-09-2020, V B Navya Kiran, Raksha R, Anisoor Rahman, Varsha K N, Dr. Nagamani N P) –

The detection system includes the processes of face image extraction, yawning tendency, blink of eyes detection, eye area extraction etc. This paper provides a comparative study on papers related to driver drowsiness detection and alert system. It is designed in such a way where system does not continuously record or retain any data.

**Driver Drowsiness Detection System**(12 December 2019, Pratyush Agarwal) –

This paper analyses the method used to detect driver’s drowsiness and proposes the results & solutions on the limited implementation of the various techniques that are used in such embedded systems.

**Driver Drowsiness Detection System**(May 2019,Muhammad Faique Shakeel and Nabita Bajwa). –

In this article, they propose a novel deep learning methodology based on Convolutional Neural Networks (CNN) to tackle the Project. In the trained model, we only use 250 low-light images.

## 2.2 Summary

In this Python project, we developed a drowsy driver alert system aimed at enhancing road safety. Using OpenCV, we implemented a Haar Cascade Classifier to detect faces and eyes in real-time. Once detected, we employed a Convolutional Neural Network (CNN) to predict the driver's alertness status, classifying whether the driver is awake, drowsy, or at risk of falling asleep. This system can be implemented in various ways, either as part of a vehicle's onboard system or as a standalone application, ultimately contributing to safer driving practices.

# Chapter 3

**SYSTEM REQUIREMENTS**

## 3.1 Hardware Requirements

**Minimum Specifications**

1.Processor: Intel Core i3 (2.4 GHz or faster)

Reason: The model inference and OpenCV processing require moderate computational

power for real-time operations.

2.RAM: 8 GB

Reason: To be able to handle image processing, model inference, and running multiple software simultaneously.

3.Storage: At least 10 GB free space

Reason: To store the dataset, trained model, and necessary dependencies.

4.Camera: Standard USB or built-in webcam with 720p resolution

Reason: To record the live video feed of the driver for analysis.

5.Display: Monitor of at least a 1024x768 resolution

Reason: To show the live feed and analysis during testing.

**Suggested Specifications**

1.CPU: Intel Core i5 or AMD Ryzen 5 (2.8 GHz above)

Reason: It will offer faster processing for the high-accuracy real-time detection.

2.RAM: 16 GB

Reason: This is optimum and shall ensure high performance during training, inference, and multi-tasking.

3.GPU: At least NVIDIA GTX 1050 or higher, CUDA-enabled

Reason: It accelerates model training and speeds up image processing tasks.

4.Storage:50 GB SSD or HDD (SSD preferable because the read/write speeds are faster)

Reason: To have the space for dataset storage, preprocessed file storage, and extra software

5.Camera: Full HD (1080p) webcam

Reason: Higher accuracy face and eye detection.

## 3.2 Software Requirements

### 3.2.1 Operating Systems

* **Windows:** Windows 10 or 11 (64-bit)
* **Linux:** Ubuntu 20.04 or higher
* **macOS:** Monterey or Ventura

### 3.2.2 Programming Language

(Python 3.6 or above)

Python is an interpreted, high-level, general-purpose programming language. Python is simple and easy to read syntax emphasizes readability and therefore reduces system maintenance costs. Python supports modules and packages, which promote system layout and code reuse. It saves space but it takes slightly higher time when its code is compiled. Indentation needs to be taken care of while coding. Python does the following:

1. Python can be used on a server to create web applications.
2. It can connect to database systems. It can also read and modify files.
3. It can be used to handle big data and perform complex mathematics.
4. It can be used for production-ready software development.

Python has many inbuilt library functions that can be used easily for working with machine learning algorithms. All the necessary python libraries must be pre- installed using “pip” command.

### 3.2.3 Python Libraries

* TensorFlow 2.x

TensorFlow 2.x is an open-source deep learning framework for building, training, and deploying machine learning models. It supports the acceleration of training and inference using graphical processing units (GPUs). Development and training of the CNN model needed in order to classify whether the driver is currently in a state of sleep or wakefulness are integral to our project and can give real-time inferences immediately.

Key Features:

* Provides GPU acceleration for training and inference.
* Built-in support for CNNs, making it ideal for image classification tasks like this one.
* Keras

Keras is a high-level API for TensorFlow that simplifies model building, training, and evaluation. Its modular and user-friendly interface allows for quick experimentation with deep learning models. In our project, Keras is used to construct the CNN, define layers, and compile the model for efficient training and deployment.

* OpenCV

OpenCV is the free computer vision library that is applied for real-time video capture, face detection, and image processing. It has Haar Cascade Classifiers to detect faces and eyes in video frames. In the whole sleepiness detection system, OpenCV will capture the webcam video and process the frames to capture faces and eyes for the system.

Key Features:

* Provides Haar Cascade Classifiers for face and eye detection.
* Captures video frames from a webcam for real-time processing.
* NumPy

NumPy is a library that supports the numerical computation in Python. It includes efficient support for multi-dimensional arrays. It's very important to pre-process images by resizing and normalizing image data before feeding the data to the CNN for prediction. However, it ensures efficient handling of pixel data and image manipulation.

* Matplotlib

Matplotlib is a plotting library to show data, for example, model metrics when training. In this project, it's used to plot graphs that track the performance of a CNN model, for instance, loss and accuracy during its training. It helps in determining the suitability of the model and modifying the hyperparameters accordingly.

* Winsound

Winsound is the Python library used for the generation of simple sound alerts in Windows. At this stage, once it has been picked up as drowsy by the CNN model, it produces the beep sound to the driver. The alert sound informs the driver to wake up, thus securing safety.

* Pygame

Pygame is a multimedia library used for handling graphics and sounds. It can be used to generate more advanced audio alerts, offering flexibility in customizing sound notifications. It is especially useful for non-Windows platforms, where Winsound may not be available.

### 3.2.4 Model Requirements

Trained CNN Model:

Type: CNN; used for classification.

Model File: Loaded model that makes inferences on the real-time detection. It is given in the form of a pre-trained model file: fatigue\_model\_cnn.h5.

Model Input Size: The size of the input image during inference is 224x224. For most pre-trained models, 224 is the commonly used number.

Model Output: The model classifies the driver state into two classes: Active (0) and Sleepy (1).

Training Data: The processed data set of images belonging to two classes that is, active and sleepy was prepared and augmented as per the requirement of the project.

### 3.2.5 Real Time Detection Requirements

1. Video Capture and Face Detection:

* The video feed from the webcam is captured using OpenCV. The webcam resolution must be at least 720p for proper facial feature detection.
* Haar Cascade Classifiers are used in order to detect faces and eyes from the video frames.

1. Frame Processing:

* Each frame is preprocessed by converting it to grayscale, detecting the face and eyes, and then resizing the eye region to 224x224 pixels before feeding it into the CNN model.
* Image normalization (scaling pixel values between 0 and 1) is applied to match the model's training conditions.

1. Decision Layer and Alarm System:

* This system keeps track of a sleepy count, which counts the number of successive frames in which the driver is classified as sleepy.
* An alarm is raised whenever the sleepy count goes beyond the threshold, set to 5 in the code, with a beep sound produced by winsound library. The alarm is reset when the driver is classified as "Active."

1. User Interface:

* A Real-time display window for the webcam feed is presented along with status text indicating if the driver is "Active" or "Sleepy."
* Finally, the frame is displayed by using OpenCV's imshow function, where a rectangle is drawn around the detected face for visual purposes:.

### 3.2.6 Performance and Optimization Requirements

**Latency**: The design should have low latency to make real-time predictions while recording a video. It is critical to alert the driver in time.

**Accuracy:** The model will need to result in high accuracy rate for determination of the driver state to be either active or sleeping, classed through the trained dataset. The false alarms must be minimized.

**Scalability**: The system should be able to adapt to different lighting conditions and varied drivers with minimal additional configuration or training data.

### 3.2.7 Jupyter Notebook

Jupyter Notebook is an interactive web application that allows you to create and share documents that contain live code, visualizations, and narrative text. It’s widely used in data science, machine learning, and academic research because it provides a flexible environment for experimentation and documentation. With Jupyter, you can write code in a variety of programming languages, visualize data, and include explanatory text, all in one place.

### 3.2.8 PyCharm

For Python programming, Jupyter Notebook is a popular choice, but if you prefer a more traditional development environment, PyCharm is an excellent option. PyCharm is a well-liked integrated development environment (IDE) that offers sophisticated code analysis, debugging, and project navigation tools. It supports databases, web development frameworks, and popular version control systems. With its intuitive interface and intelligent coding assistance, PyCharm enhances the efficiency of Python development. Whether you choose Jupyter Notebook for its interactivity or PyCharm for its robust features, both tools can effectively support your programming needs.

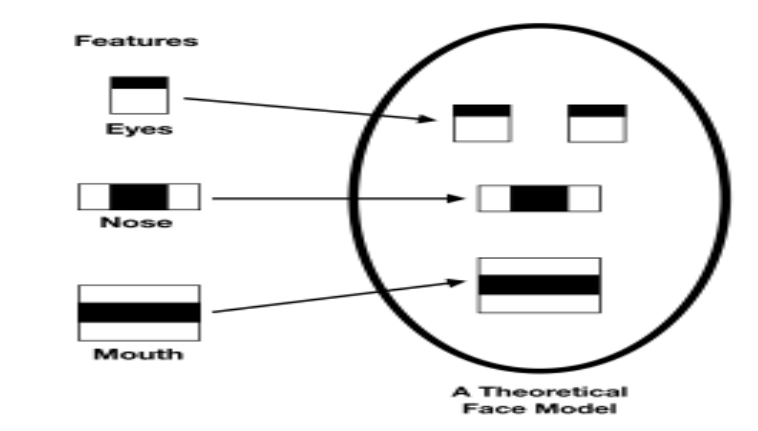
# Chapter 4

**DESIGN AND IMPLEMENTATION**

## 4.1 Proposed System and Advantages

### 4.1.1 Proposed System

The proposed system integrates a live detection tool for driver drowsiness based on deep learning, utilizing a Convolutional Neural Network (CNN) model in combination with computer vision techniques such as OpenCV and Haar Cascade Classifiers. The system is designed to classify the driver's eye state into "Active" or "Sleepy" and alert them in case of prolonged drowsiness. The key components of the system are:

1. Web Cam Integration to Monitor Live: This step involves integrating a webcam to capture live video frames of the driver's face. Each frame is analyzed in real-time to detect facial features.
2. Face Detection and Eye Detection using Haar Cascade Classifiers: OpenCV's Haar Cascade Classifiers are used to detect faces and eyes in each frame of video feed. Once we identify a face, we zoom in on the eye regions to ensure we capture the most critical details.

**Fig.1** Haar Cascade approach

1. CNN Model for Eye State Classification: The CNN model is used to classify eye regions into "Active" or "Sleepy." The reason behind choosing the CNN is for the ability to extract proper spatial features, particularly suited for real-time applications with robust and accurate prediction.
2. Alarm System Based on Sleepy Frame Threshold: If the system or the algorithm detects more than a given number of consecutive "Closed" eye frames, it activates the alarm. The mechanism, therefore, helps reduce false alarms resulting from minimal blinking and ensures that alerts are produced only when sleepiness is detected. In case the system detects drowsiness, it triggers an alert sound to wake the driver. When the driver's eyes recover into an "Open" state, the alarm automatically stops and the system returns to an "Active" status.
3. User-Friendly Implementation: We’ve built the system using Python, which makes it not only modular but also easy to tweak and expand as needed. This means we can easily add new features, like detecting yawning or tracking fatigue levels, without compromising the smooth operation of the system. Our goal is to create a seamless and non-intrusive driver drowsiness monitoring system that provides timely alerts, helping to keep drivers safe and prevent accidents on the road.

### 4.1.2 Advantages

The proposed system has a number of advantages over other existing techniques as follows:

1. Non-Intrusive and Non-Discomfort: Unlike methods that require physical sensors, our system uses just a camera to keep an eye on the driver’s attention. This means no uncomfortable devices are needed, making the driving experience much more pleasant.
2. Real-Time Detection and Response: Our CNN model works quickly, capturing and processing images in real time. This allows for immediate feedback, which can be crucial in preventing accidents caused by drowsiness.
3. Efficient and Optimized Performance: It uses the CNN architecture, which ensures robust feature extraction and accurate predictions. This model has good computational efficiency and can be easily deployed on standard hardware without much latency.
4. Enhanced Eye State Monitoring: The system focuses on monitoring whether the driver’s eyes are open or closed, providing a direct way to identify signs of drowsiness.
5. Adaptability to Real-World Conditions: Our system is designed to handle a variety of real-world situations utilizing OpenCV’s image processing capabilities.
6. Scalability for Future Development: The modular design of the system means it can easily be upgraded. This thoughtful approach, combined with the advanced capabilities of CNN, ensures that the system remains effective, user-friendly, and ready for real-world use.

## 4.2 Methodology

### 4.2.1 Data Collection

In our project, we started by gathering raw data from Kaggle, which offered a collection of images featuring open and closed eyes. This dataset included 726 images for each category, giving us a total of 1,452 images. While this was a good foundation, we quickly realized it wasn’t enough to effectively detect driver drowsiness. To address this limitation, we decided to enhance the dataset through a process called data augmentation.

To expand our collection, we employed several techniques to modify the original images. We rotated, shifted, and flipped them, creating five new versions of each image. This approach allowed us to significantly increase our dataset size to 7,260 images. By introducing this variety, we aimed to help our model learn to recognize eye states in different conditions and angles, making it more adaptable to real-world driving scenarios.

Once we had augmented the images, we resized them to ensure they were suitable for training our convolutional neural network (CNN). Resizing is important because it helps maintain image quality while fitting the model's input requirements. Additionally, we created an annotation file to label the images accurately, distinguishing between open and closed eyes. This labeling is crucial for supervised learning, as it provides the model with the information it needs to learn and make accurate predictions about driver drowsiness. Through these efforts, we aimed to build a comprehensive dataset that would enhance the effectiveness of our drowsiness detection system.

### 4.2.2 Data Preprocessing

Since the dataset obtained from Kaggle was not so big, preprocessing and augmentation were inevitable in order to rise the training performance and extend generalization for the model. The preprocessing process includes:

1. Image Resizing: The dimension of each image of the dataset has been resized into 224x224 pixels so that the model could work properly.
2. Data Augmentation: In order to expand the dataset and introduce variability, transformations have been applied:
   * Rotation: Images were randomly rotated between -30 and +30 degrees to simulate head movements.
   * Translation (Shifting): Horizontal and vertical shifts were applied to reflect slight positional changes.
   * Flipping: Horizontal flips were presented, introducing mirrored variations of each image-to help the model recognize features symmetrically.
   * Scaling and Cropping: We scaled zoom levels within small ranges so that minor scaling differences are induced in images.
3. Normalization: The pixel values for the image had to be normalized within the range 0-1, which again used the model learning features on features without variable lighting conditions or intensity of the image.

One preprocessing strategy was important in expanding the size and diversity of our data set, which would ensure that our model could efficiently train on limited original data while improving robustness in real-world scenarios.

### 4.2.3 Model Selection

Choosing the right model for detecting driver drowsiness is one of the most important decisions we made for our project. After exploring various options, we decided to go with a Convolutional Neural Network (CNN). This choice was influenced by the unique strengths of CNNs, especially when it comes to recognizing and classifying images—an essential aspect of identifying drowsiness.

Why CNN for Drowsiness Detection?

Particular Suitability to Vision Data:

CNNs are particularly well-suited for working with images. They excel at learning the spatial relationships in visual data, which makes them great at detecting whether a driver’s eyes are open or closed. This ability to automatically learn from the images means that our system can effectively pick up on subtle signs of drowsiness, helping to keep drivers safe on the road.

Automatic Feature Extraction

One of the standout features of CNNs is their capability to extract important characteristics from images without needing manual intervention. Unlike traditional methods that require us to specify what features to look for, CNNs can learn to recognize edges, shapes, and textures on their own. This is particularly useful for our project, as it allows the model to adapt to different eye patterns associated with drowsiness without extensive preprocessing.

High Accuracy and Performance:

CNNs have a proven track record of delivering high accuracy in image classification tasks. They are designed to handle variations in lighting and facial orientation, which are common in real-world situations. This robustness is crucial for our drowsiness detection system, as it needs to perform reliably no matter the conditions or the individual driver.

Scalability and Adaptability:

Another reason we chose CNNs is their scalability and adaptability. These models can easily be adjusted or expanded for specific applications, allowing us to improve performance as new challenges arise. Their modular design means we can integrate new techniques or layers into the model, ensuring that our system can evolve alongside advancements in technology.

|  |  |  |
| --- | --- | --- |
| *Model: “sequential\_3”* |  |  |
| ***Layer (type)*** | ***Output shape*** | ***Param#*** |
| *Conv2D* | (None,222,222,32) | 896 |
| *BatchNormalization* | (None,222,222,32) | 128 |
| *MaxPooling2D* | (None,111,111,32) | 0 |
| *Dropout* | (None,111,111,32) | 0 |
| *Conv2D* | (None,109,109,64) | 18496 |
| *BatchNormalization* | (None,109,109,64) | 256 |
| *MaxPooling2D* | (None,54,54,64) | 0 |
| *Dropout* | (None,54,54,64) | 0 |
| *Conv2D* | (None,52,52,128) | 73856 |
| *BatchNormalization* | (None,52,52,128) | 512 |
| *MaxPooling2D* | (None,26,26,128) | 0 |
| *Dropout* | (None,26,26,128) | 0 |
| *Conv2D* | (None,24,24,256) | 295168 |
| *BatchNormalization* | (None,24,24,256) | 1024 |
| *MaxPooling2D* | (None,12,12,256) | 0 |
| *Dropout* | (None,12,12,256) | 0 |
| *Flatten* | (None,36864) | 0 |
| *Dense* | (None,128) | 4718720 |
| *Dropout* | (None,128) | 0 |
| *Dense*  **Total params**: 5,109,187  **Trainable params**: 5,108,225  **Non-trainable params**: 960  **Optimizer params**: 2 | (None,1) | 129 |

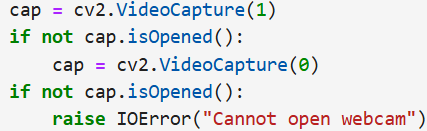
**Fig.2** CNN Architecture

## 4.3 System Design

### 4.2.3 System Architecture

The design of the architecture for the system of driver drowsiness detection is always real-time; therefore, it moves from the capture of video frames to the generation of alerts. The system consists of the following interconnected layers:

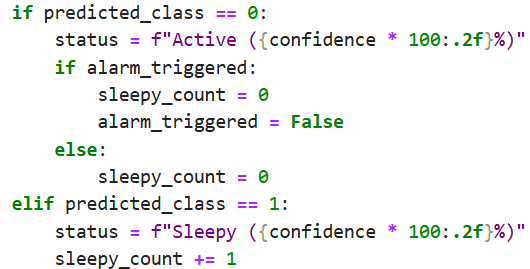
1. Data Input (Camera Feed)

* Purpose: Capture live video of the driver's face in real time.
* Implementation: Initialize a webcam using OpenCV and attempt to open an external or built-in camera. Video Frames are captured continuously then are processed individually. If the webcam cannot open, error is raised.

1. Preprocessing Layer

* Motive: To prepare each video frame for further feature extraction and classification.
* Implementation: Face Detection **-** Haar Cascade Classifier is used for detecting the face of the driver within a frame. It makes sure that all the processing around the area of interest would be done. Eye Detection **-** Within the identified face, Haar Cascade Classifier identifies the areas for eyes. The regions are cropped and processed for classification.
* Image Resizing and Normalization: Crop the eye region to a size that will fit the CNN model's input size, 224 x 224. Normalize pixel values to be in the range of 0-1 through dividing by 255.0, thus having the same input go into the model.

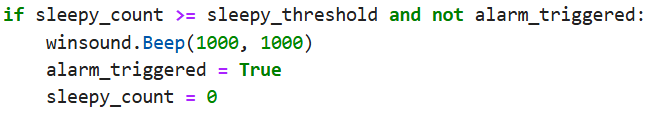
1. Feature Extraction and Model Inference Layer

* Objective: Predict the state of the driver as Active or Sleepy through a CNN-based deep learning model.
* Loads: A pre-trained CNN model named fatigue\_model\_cnn.h5 is to be loaded using TensorFlow. The processed eye region is passed to the model for inference .The CNN model has outputs in the form of probabilities of two classes: Class 0 ‘Active’ and Class 1 ‘Sleepy’.
* The state with the maximum-probability prediction is selected and its respective confidence score is logged.

1. Decision Layer

* Goal: Predict across frames successively in order to avoid false alarms and determine at which frames to raise alerts.
* Sleepy Counter Mechanism: It keeps track of the count of successive frames classified as Sleepy. The threshold to ring an alarm is set to 5 consecutive Sleepy frames.
* Reset Mechanism: A frame identified as Active, the counter and all alarms triggered are reset to zero.
* Threshold-Based Alarm: Counter exceeding the threshold value, an audio alarm is invoked with winsound.

1. Alarm System

* Goal: Sound an alarm as soon as and as long as the system establishes that the driver is persistently sleepy.
* Implementation: Audio beep is played using winsound. Beep whenever drowsiness is detected for the threshold number of consecutive frames. The system prevents unnecessary alerts by resetting the counter and deactivating the alarm after it has been triggered.

1. Real-Time Feedback (UI Layer)

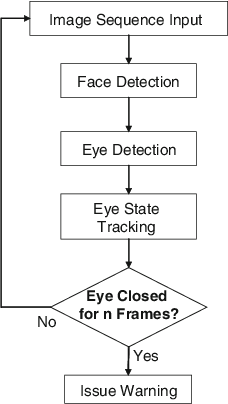
* Function: Update in real time, the state of the driver across the video feed
* Implementation: The state of the driver as detected (Active or Sleepy) along with a confidence score overlay on the video feed using OpenCV. It also superimposes bounding boxes around detected faces to visually highlight the region of interest.

The processed frames are displayed in a window titled "Sleepiness Detection Window."

This layered architecture provides a sound and efficient drowsiness detection system that balances computational efficiency with real-time responsiveness to safeguard drivers.

### 4.3.2 Component Interaction

The interaction between components in the driver drowsiness detection system follows a structured flow to ensure efficient, real-time detection and alerting:



**Fig.3** Flowchart Implementation

# Chapter 5

**RESULT AND DISCUSSION**

## 5.1 Performance Metrics

For your system, the performance metrics focus on evaluating how well the model identifies the driver’s state (active or sleepy). Here are the key metrics for your project:

### 5.1.1. Accuracy

* **Description**: This metric measures the overall correctness of the model in classifying the driver’s state (active, sleepy, or yawning).
* **Formula**: Accuracy=
* **Relevance**: It shows how often the model makes the correct prediction, which is critical for ensuring the system performs reliably during real-time detection.

### **5.1.2.** **Precision**

* **Description**: Precision measures the accuracy of positive predictions (how many instances predicted as "Sleepy" are actually "Sleepy").
* **Formula**: Precision=

* **Relevance**: For drowsiness detection, high precision is important to avoid false positives (e.g., mistakenly detecting a driver as "Sleepy" when they are not).

### **5.1.3. Recall**

* **Description**: Recall measures the ability of the model to correctly identify all positive instances (how many "Sleepy" drivers are actually classified as "Sleepy").
* **Formula**: Recall=

* **Relevance**: High recall ensures that the model does not miss out on detecting a "Sleepy" driver, which is crucial for safety.

### 5.1.4.F1-Score

* **Description**: The F1-score is the harmonic mean of precision and recall, providing a balance between the two.
* **Formula**: F1-Score=

* **Relevance**: The F1-score is particularly important in this project as it balances the need to avoid false positives and false negatives, ensuring that both accurate detection and safety are maintained.

### 5.1.5. Confusion Matrix

* **Description**: A confusion matrix provides a breakdown of the model’s predictions against the actual labels, showing the true positives, true negatives, false positives, and false negatives.
* **Relevance**: The confusion matrix helps to visually identify where the model is making errors and provides insight into which states (active, sleepy, yawn) are being misclassified.

### 5.1.6. Real-Time Inference Time

* **Description**: Measures how long the model takes to process each frame of the camera feed and make predictions.
* **Relevance**: Since the system needs to detect drowsiness in real time, a fast inference time is crucial to ensure timely alerts.

### 5.1.7. False Positive Rate

* **Description**: The rate at which the model incorrectly classifies a non-drowsy driver as "Sleepy".
* **Formula**: FPR =

* **Relevance**: A low false positive rate ensures that the model doesn't unnecessarily alert the driver, which could lead to distractions or inconvenience.

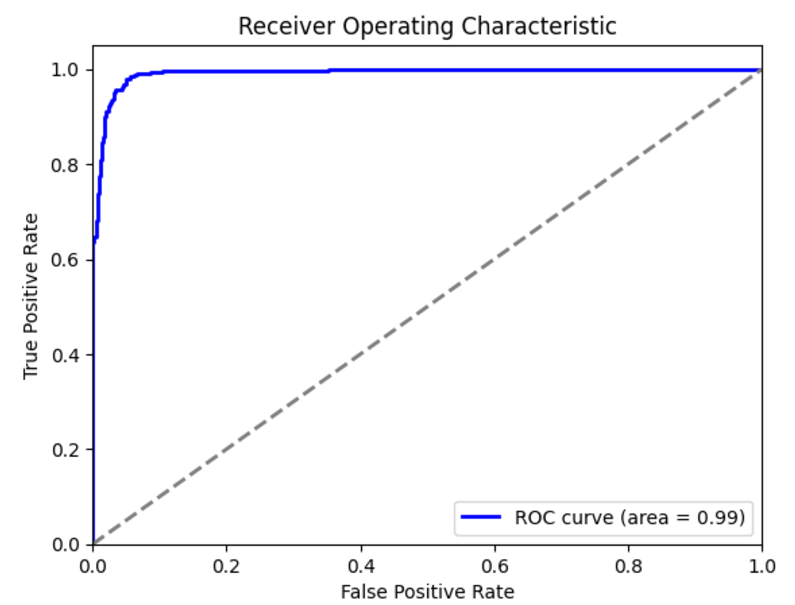
## 5.2 Result Analysis

### 5.2.1 ROC Curve

The training ROC curve displays the model’s ability to

distinguish between the two classes based on its predictions on the test data. From the Figure it is observed that, an AUC of 0.95 indicates that the model performs exceptionally well on the test set, as it achieves a high true positive rate while maintaining a low false positive rate across various classification thresholds. It is also observed that, a high AUC on the test data suggests that the model has effectively generalized and can reliably separate positive and negative examples in unseen data.

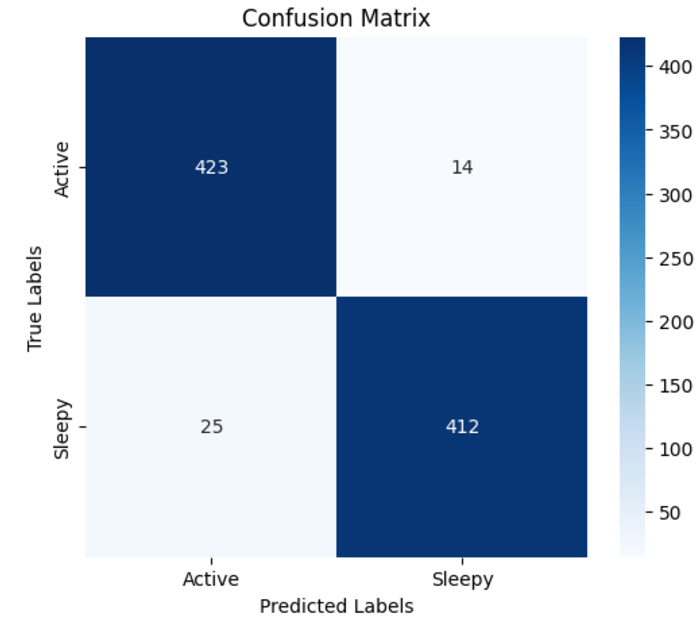
**Fig.4** ROC Curve



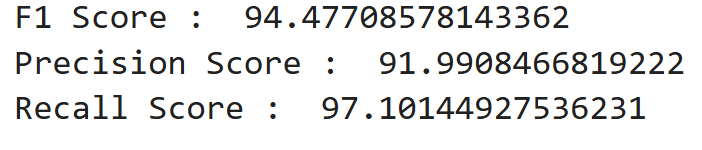
### **5.2.2 Confusion Matrix**

Confusion Matrix provides a clear overview of how well the system is performing by comparing its predictions against the actual states of the drivers. Analyzing this confusion matrix allows developers to pinpoint weaknesses in the detection system, such as a high rate of false negatives, which poses a significant safety risk by failing to identify drowsy drivers. Similarly, frequent false positives could lead to unnecessary alerts that frustrate drivers. By understanding these dynamics, developers can refine the detection algorithms, ultimately enhancing the system's accuracy and contributing to safer driving

**Fig.5** Confusion Matrix



conditions on the road.



## 5.3 Screenshots

# 

# Conclusion

As we conclude our project on detecting driver drowsiness, we’ve come to appreciate the significant role that technology plays in enhancing road safety. By utilizing advanced algorithms and machine learning techniques, we can create solutions that not only track driver alertness but also provide timely alerts to help prevent potential accidents.

This project has deepened our understanding of how innovative solutions can address real-world challenges, ultimately paving the way for safer driving experiences and reducing the risks associated with drowsy driving.

At the heart of this project is our commitment to addressing a real-world problem—drowsy driving. We hope that our work contributes to reducing the risks associated with this issue and ultimately helps save lives. As we look ahead, we’re inspired by the potential for further research and development in this area, and we believe that our project is just the beginning of what can be achieved in enhancing automotive safety. Together, we can create a safer driving environment for everyone on the road.

While this system effectively alerts drowsy drivers, it is important to note that technology alone cannot eliminate the risks associated with drowsy driving. Drivers must remain aware of their physical limits and avoid relying solely on alerts to stay awake. An alert triggered even one second too late might be insufficient to prevent a crash. Therefore, this system should complement, not replace, responsible driving practices.

# References

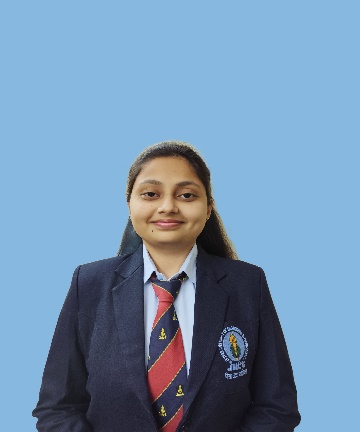
1. Wikipedia: [www.wikipedia.org/](http://www.wikipedia.org/)
2. ResearchGate: [www.researchgate.net/](http://www.researchgate.net/)
3. OpenAI: [chat.openai.com](https://chat.openai.com)
4. Kaggle Dataset: [www.kaggle.com/datasets/dheerajperumandla/drowsiness-dataset](http://www.kaggle.com/datasets/dheerajperumandla/drowsiness-dataset)
5. Research Article: [www.mdpi.com/1424-8220/23/14/6459](http://www.mdpi.com/1424-8220/23/14/6459)
6. Drowsy Driving Behaviour Survey: [rosap.ntl.bts.gov/view/dot/1725/dot\_1725\_DS 1.pdf](https://rosap.ntl.bts.gov/view/dot/1725/dot_1725_DS%201.pdf)

# Contributors

1. **ARPITA SINGH**  
   Enrolment Number: 221B089  
   Email: [221b089@juetguna.in](mailto:221b089@juetguna.in)

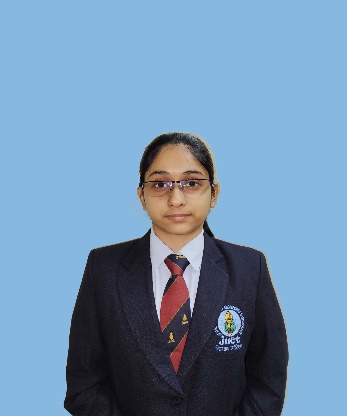
**Personal Email:** [arpitasinghrajput16@gmail.com](mailto:arpitasinghrajput16@gmail.com)

**Address:** Lucknow, U.P.

1. **SHRUTI BHARGAVA**  
   Enrolment Number: 221B374  
   Email: [221b374@juetguna.in](mailto:221b374@juetguna.in)

**Personal Email:** [shrutibhargava2004@gmail.com](mailto:shrutibhargava2004@gmail.com)

**Address:** Betul, M.P.

1. **SNEHIL SHARMA**Enrolment Number: 221B387   
   Email: [221b387@juetguna.in](mailto:221b387@juetguna.in)

**Personal Email:** [ssnehil06@gmail.com](mailto:ssnehil06@gmail.com)

**Address:** Bhilwara, Raj.