VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANA SANGAMA, BELAGAVI -590 018

DEEP LEARNING AND REINFORCEMENT LEARNING

Driver Drowsiness Detection using CNN

Submitted by

HARSHITHA M V **1AY22AI038**

Under the Guidance of
Dr. Vijayshekhar S S
Associate Professor
Department of AI & ML

2025-2026

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Acharya Institute of Technology

Acharya Dr. Sarvepalli Radhakrishnan Road, Soladevanahalli, Bengaluru-560107

ABSTRACT

Road accidents caused by driver fatigue and drowsiness are one of the leading contributors to loss of life and property across the world. Drowsiness reduces alertness, slows reaction time, and severely affects decision-making ability, thereby increasing the risk of accidents. Continuous monitoring of a driver's mental state using traditional methods is difficult and often unreliable, as it depends on self-awareness. To overcome this limitation, this project presents the design and development of a Universal Driver Drowsiness Detection System based on real-time computer vision techniques that can detect drowsiness for any person in front of the camera and immediately generate an alert.

The proposed system uses a webcam to continuously capture live video of the person. The captured frames are processed using OpenCV, where facial features are analyzed in real time. Haar Cascade classifiers are employed for efficient and accurate detection of the face and eyes. Once a face is detected, the eye region is extracted and continuously monitored to determine whether the eyes are open or closed. A fatigue scoring mechanism is implemented, where the score increases when the eyes remain closed continuously and decreases when the eyes are detected as open. If the fatigue score exceeds a predefined threshold, the system concludes that the person is drowsy and immediately triggers an audio alarm using the Pygame library to alert the user.

To make the system interactive and accessible, the entire real-time detection process is displayed through a Flask-based web interface running on a local server. The live video feed, fatigue score, and drowsiness alert message are streamed directly to a web browser using the local host. This ensures that the system remains platform-independent and easy to deploy on any standard computer without the need for specialized hardware.

The major advantage of this system is that it does not require physical contact with the driver or any wearable sensors. It operates entirely using a camera, making it a non-intrusive, low-cost, and efficient solution for real-world applications.

Experimental results show that the proposed system successfully detects face and eye activity in real time with high accuracy under normal lighting conditions.

1. INTRODUCTION

Driver drowsiness detection is one of the most important and challenging problems in computer vision and intelligent transportation systems. The task involves continuously monitoring a person's facial features, especially the eyes, to determine whether the person is alert or drowsy. Drowsiness leads to reduced concentration, slower reaction time, and impaired decision-making, which significantly increases the risk of accidents. Traditionally, driver fatigue has been detected using physical sensors such as EEG, heart rate monitors, and wearable devices. However, these methods are often uncomfortable, expensive, and intrusive. With the advancement of computer vision, real-time drowsiness detection using a camera has become a practical and non-intrusive solution.

1.1 Motivation

- Increasing number of road accidents due to driver fatigue and sleepiness
- Need for a real-time, automatic, and non-intrusive safety monitoring system
- Low-cost solution using only a webcam without wearable sensors
- Practical application of artificial intelligence and computer vision in road safety
- Growing demand for smart vehicle safety systems

1.2 Dataset

This project does not use a traditional static image dataset. Instead, it works on real-time live video captured using a webcam. The system uses:

- Live camera frames as dynamic input
- Grayscale face and eye images extracted in real time
- Pre-trained Haar Cascade XML files for face and eye detection
- Continuous frame-by-frame analysis instead of offline dataset training

This real-time data approach allows the system to work for any person in front of the camera without requiring a pre-collected dataset.

1.3 Project Objectives

1. Capture real-time video using a webcam
2. Detect human faces using Haar Cascade classifier
3. Detect eyes within the detected face region
4. Monitor eye closure continuously using fatigue scoring
5. Trigger an audio alarm when drowsiness is detected
6. Display real-time detection results using a Flask web interface
7. Ensure the system works for any person in front of the camera

2. METHODOLOGY

The methodology of the proposed Universal Driver Drowsiness Detection System is designed to ensure accurate, real-time monitoring of a person's alertness using computer vision techniques. The system follows a structured pipeline consisting of video acquisition, preprocessing, face detection, eye detection, fatigue evaluation, and alert generation. Each stage plays a crucial role in ensuring the reliability and efficiency of the system. The entire process operates continuously on a live video stream obtained from a webcam, allowing the system to detect drowsiness for any person in front of the camera.

The overall objective of the methodology is to extract meaningful facial features from real-time video frames, analyze eye activity, and determine fatigue using a dynamic scoring approach. The system is implemented using Python with OpenCV for image processing, Haar Cascade classifiers for object detection, Flask for web deployment, and Pygame for audio alert generation.

2.1 System Workflow

The workflow of the system begins with the capture of live video frames through a webcam. These frames are passed through multiple processing stages that include image enhancement, object detection, and fatigue analysis. Once the fatigue score exceeds a predefined threshold, an alarm is generated to alert the user. This process is repeated continuously to ensure uninterrupted monitoring.

The major steps involved in the workflow are:

1. Video acquisition using a webcam
2. Frame preprocessing for better detection accuracy
3. Face detection using Haar Cascade classifier
4. Eye detection within the detected face region
5. Fatigue scoring based on eye closure duration
6. Drowsiness alert generation
7. Web-based real-time display using Flask

This systematic workflow ensures reliable and efficient detection of drowsiness under real-time conditions.

2.2 Video Acquisition

The first step in the methodology is video acquisition. A standard webcam is used to capture real-time video continuously. The camera captures frames at a regular interval, which are then processed by the system. The use of a webcam makes the system cost-effective and easily deployable on any standard computer without specialized hardware.

Each captured frame is immediately passed to the image preprocessing stage for further analysis. The system supports dynamic detection for any person who appears in front of the camera without requiring pre-registration.

2.3 Image Preprocessing

The captured video frames undergo preprocessing to improve detection accuracy. Preprocessing is an essential step because raw images obtained from the camera may contain noise, variations in lighting, and unwanted background elements.

The preprocessing steps include:

- Frame flipping to obtain mirror-like output for natural viewing
- Color conversion from RGB to grayscale to reduce computational complexity
- Histogram equalization to enhance contrast and improve visibility of facial features

These operations help in improving face and eye detection accuracy and ensure consistent performance even under varying lighting conditions.

2.4 Face Detection

Face detection is performed using the Haar Cascade classifier, which is a machine learning-based approach for object detection. The Haar Cascade classifier is trained using a large number of positive and negative images and is capable of detecting frontal human faces with high accuracy.

The grayscale image is scanned using multiple window sizes to detect potential face regions. Once a face is detected, a rectangular bounding box is drawn around it for visualization. Only the region inside the detected face is further used for eye detection. This reduces false detections and increases processing efficiency.

2.5 Eye Detection

After detecting the face, eye detection is performed within the face region of interest (ROI). The same Haar Cascade technique is used for eye detection. The eye detector searches for two eye-like structures inside the face region.

If the eyes are detected successfully, it is assumed that the person is alert. If the eyes are not detected continuously for a certain duration, it is considered as a sign of drowsiness. Rectangular boxes are drawn around the detected eyes for visualization.

2.6 Fatigue Scoring Mechanism

The fatigue scoring mechanism is the core logic of the drowsiness detection system. Instead of making a decision based on a single frame, the system monitors eye status continuously and assigns a fatigue score.

- If eyes are not detected continuously, the fatigue score is incremented.

- If eyes are detected, the fatigue score is decremented.
- The fatigue score is never allowed to become negative.
- A threshold value is defined to determine drowsiness.

When the fatigue score crosses the threshold value, the system confirms that the person is drowsy. This scoring mechanism avoids false alerts caused by normal blinking and ensures stability in decision making.

2.7 Drowsiness Alert Generation

Once drowsiness is detected, the system generates an immediate alert. The alert mechanism consists of:

- A visual alert message displayed on the screen
- An audio alarm sound generated using the Pygame library

The alarm continues to ring at regular intervals until the fatigue score drops below the threshold, indicating that the person has become alert again. This dual alert system ensures maximum effectiveness in preventing accidents.

2.8 Web-Based Real-Time Display using Flask

To make the system interactive and user-friendly, the real-time video feed is displayed on a web browser using the Flask framework. The processed frames are streamed continuously to the browser using a local server.

The Flask-based interface displays:

- Live video feed
- Fatigue score
- Drowsiness alert message

This makes the system platform-independent and allows monitoring on any device connected to the local network.

2.9 Hardware and Software Requirements

Hardware Requirements

- Webcam or laptop camera
- Computer/Laptop with minimum 4 GB RAM
- Speakers or headphones for alarm output

Software Requirements

- Python 3.10

- OpenCV
- Flask
- Pygame
- Visual Studio Code
- Windows Operating System

2.10 Advantages of the Proposed Methodology

- Works in real time without noticeable delay
- Does not require physical sensors or wearable devices
- Low-cost and non-intrusive solution
- Can detect drowsiness for any person
- Easy to deploy and scalable

3. SYSTEM ARCHITECTURE

The system architecture of the Universal Driver Drowsiness Detection System defines the structural design and interaction between various hardware and software components used to implement the real-time fatigue monitoring solution. The architecture is designed to be modular, scalable, and efficient so that it can operate smoothly under real-time constraints. It integrates computer vision techniques, real-time video processing, audio alert mechanisms, and web-based visualization to provide a complete safety monitoring solution.

The architecture follows a layered and pipeline-based approach, where each stage performs a specific function in the drowsiness detection process. These stages include video capture, preprocessing, face detection, eye detection, fatigue analysis, alert generation, and web streaming. The complete system runs locally on a computer using a webcam and does not require any external sensors or cloud services.

3.1 Overview of the Architecture

The proposed system consists of the following major modules:

1. Input Module (Webcam)
2. Preprocessing Module
3. Face Detection Module
4. Eye Detection Module
5. Fatigue Analysis Module
6. Alert Generation Module
7. Web Interface Module (Flask Server)

Each module is connected in a sequential manner and operates in real time. The output of one module becomes the input for the next module, thereby forming a continuous processing pipeline.

3.2 Input Module (Webcam Interface)

The input module is responsible for capturing real-time video from a webcam. A standard built-in laptop camera or an external USB webcam is used as the primary video source. The webcam continuously captures frames and sends them to the processing pipeline.

The advantages of using a webcam as the input device include:

- Low cost and easy availability
- No need for special sensors
- Real-time data acquisition
- Compatibility with most computing systems

Each captured frame is immediately forwarded to the preprocessing module for further enhancement and feature extraction.

3.3 Preprocessing Module

The preprocessing module improves the quality of the captured frames before applying detection algorithms. Since raw frames may contain noise, poor lighting, or unwanted background information, preprocessing plays a vital role in improving detection accuracy.

The preprocessing operations include:

- Frame flipping for natural mirror-view display
- Color space conversion from RGB to grayscale to reduce computational complexity
- Histogram equalization to enhance image contrast and visibility of facial features

These operations help the detection algorithms work efficiently and reduce false positives.

3.4 Face Detection Module

The face detection module uses a Haar Cascade Classifier to detect frontal human faces in each preprocessed frame. Haar Cascade is a machine learning-based object detection method that scans the image at multiple scales and locations to identify potential face regions.

Once a face is detected:

- A bounding box is drawn around the face
- Only the detected face region is passed to the eye detection module
- Background noise is minimized for accurate eye analysis

This module ensures that eye detection is applied only to the relevant facial area, improving both accuracy and speed.

3.5 Eye Detection Module

The eye detection module operates within the detected face region. Another Haar Cascade classifier trained specifically for eye detection is used to locate the eyes.

This module performs the following functions:

- Detects the presence of eyes
- Determines whether the eyes are open or closed
- Tracks eye visibility continuously across frames

If eyes are detected in the current frame, the person is considered alert. If eyes are missing continuously, the system interprets it as a sign of drowsiness.

3.6 Fatigue Analysis Module

The fatigue analysis module is the decision-making core of the system. Instead of relying on a single frame, it uses a fatigue scoring mechanism to evaluate alertness over time.

This module works as follows:

- If eyes are not detected, the fatigue score is incremented
- If eyes are detected, the fatigue score is decremented
- The score is never allowed to go below zero
- A predefined threshold determines the drowsiness condition

This time-based scoring approach prevents false alarms caused by normal eye blinking and ensures stable detection of actual fatigue.

3.7 Alert Generation Module

The alert generation module is activated when the fatigue score crosses the predefined threshold. This module is responsible for warning the user immediately to prevent accidents.

It consists of:

- Visual alert: A warning message displayed on the video frame
- Audio alert: An alarm sound generated using the Pygame library

The alarm continues to ring until the fatigue score drops below the threshold, indicating that the user has regained alertness.

3.8 Web Interface Module (Flask Server)

The web interface module uses the Flask framework to display the real-time video output in a browser. Instead of showing the output in a local OpenCV window, the processed frames are streamed as an MJPEG stream to a web page hosted on the local server.

The Flask interface displays:

- Live video feed
- Fatigue score
- Drowsiness alert message

The server runs on:

http://127.0.0.1:5000

This makes the system:

- Platform-independent
- Easy to monitor remotely
- Suitable for demonstration and real-time monitoring

3.9 Interaction Between Modules

The interaction between different modules follows a sequential flow:

1. Webcam captures video
2. Preprocessing enhances the frame
3. Face detection locates the face
4. Eye detection identifies eye status
5. Fatigue analysis calculates alertness level
6. Alert generation triggers alarm
7. Flask module streams the output

This continuous interaction ensures smooth real-time operation.

3.10 Hardware Architecture

The hardware architecture consists of:

- Webcam (input device)
- Computer/Laptop (processing unit)
- Speaker (audio alert output)

All processing is done locally on the system without requiring cloud services.

3.11 Software Architecture

The software architecture is based on:

- Python for overall control logic
- OpenCV for image and video processing
- Haar Cascade classifiers for face and eye detection
- Flask for web deployment
- Pygame for alarm sound generation

Each software component operates as a separate module but communicates through shared memory and function calls.

3.12 Security and Reliability Considerations

- The system runs completely offline, ensuring data privacy
- No video data is stored permanently
- Real-time processing avoids data leakage

- Alert system is independent of internet connectivity

3.13 Scalability of the Architecture

The proposed architecture can be easily extended to:

- Support deep learning-based eye tracking
- Integrate infrared cameras for night detection
- Add cloud storage and analytics
- Connect with vehicle control systems

4. IMPLEMENTATION DETAILS

The implementation of the Universal Driver Drowsiness Detection System is carried out using the Python programming language along with computer vision, web framework, and audio processing libraries. The complete system is implemented on a local machine and operates in real time using a standard webcam. The objective of implementation is to integrate all modules such as video capture, image processing, fatigue analysis, alert generation, and web-based visualization into a single working application.

The system begins by initializing the webcam using OpenCV's VideoCapture function. The camera continuously captures real-time video frames, which are then processed frame by frame. Each frame is first flipped horizontally to provide a mirror-like view for better user interaction. The frame is then converted into grayscale format to reduce computational complexity and improve detection speed. Histogram equalization is applied to enhance the contrast and visibility of facial features.

Face detection is implemented using the pre-trained Haar Cascade classifier for frontal face detection. Once a face is detected, a bounding box is drawn around it and only that region is passed to the eye detection module. The eye detection module also uses a Haar Cascade classifier trained to detect human eyes. If eyes are detected in the face region, the person is considered alert. If eyes are continuously not detected across multiple frames, the system interprets it as a sign of drowsiness.

A fatigue scoring mechanism is implemented to avoid false alarms due to normal eye blinking. The fatigue score increases when eyes are not detected and decreases when eyes are detected. The score is constrained so that it never becomes negative. A predefined threshold value is used to determine the drowsiness condition. When the fatigue score exceeds the threshold, the system confirms drowsiness.

Once drowsiness is detected, an audio alert is generated using the Pygame library. The alarm sound is played through the system speakers to alert the person immediately. A cooldown time is applied so that the alarm is not continuously retriggered within a short time interval, ensuring smooth operation.

To make the system interactive and user-friendly, the real-time video output is displayed through a Flask-based web interface. The processed frames are streamed as an MJPEG stream to a local server and can be accessed through a web browser using the URL

http://127.0.0.1:5000. This web interface displays the live camera feed, fatigue score, and drowsiness alert message.

The entire system is implemented and tested successfully on a Windows operating system using Python 3.10, OpenCV, Flask, and Pygame. The integration of all these modules ensures that the system operates smoothly in real time and accurately detects drowsiness for any person in front of the camera.

Face detection is implemented using the pre-trained Haar Cascade classifier for frontal face detection. Once a face is detected, a bounding box is drawn around it and only that region is passed to the eye detection module. The eye detection module also uses a Haar Cascade classifier trained to detect human eyes. If eyes are detected in the face region, the person is considered alert. If eyes are continuously not detected across multiple frames, the system interprets it as a sign of drowsiness.

5. RESULTS

The Universal Driver Drowsiness Detection System was tested under real-time conditions using a standard webcam. The system was evaluated for face detection accuracy, eye detection accuracy, fatigue scoring response, alarm triggering performance, and web interface display. The results demonstrate that the system performs effectively in real-world scenarios under normal lighting conditions.

During normal operation, when the person keeps the eyes open, the system successfully detects the face and eyes and displays the fatigue score as low. The system remains in a safe state and no alarm is triggered. This confirms that the system does not generate false alarms during normal alert conditions.

When the person closes the eyes continuously for a few seconds, the system fails to detect the eyes and the fatigue score increases gradually. Once the fatigue score crosses the predefined threshold, the system immediately displays the “DROWSINESS ALERT” message on the screen and triggers an audio alarm. This confirms that the fatigue detection logic works correctly and the system responds in real time.

The Flask-based web interface successfully displays the live video feed with bounding boxes around detected faces and eyes. The fatigue score is updated dynamically on the screen. The alarm response time is observed to be less than two seconds after continuous eye closure, which is sufficient for practical accident prevention.

PHOTOGRAPHIC ANALYSIS:

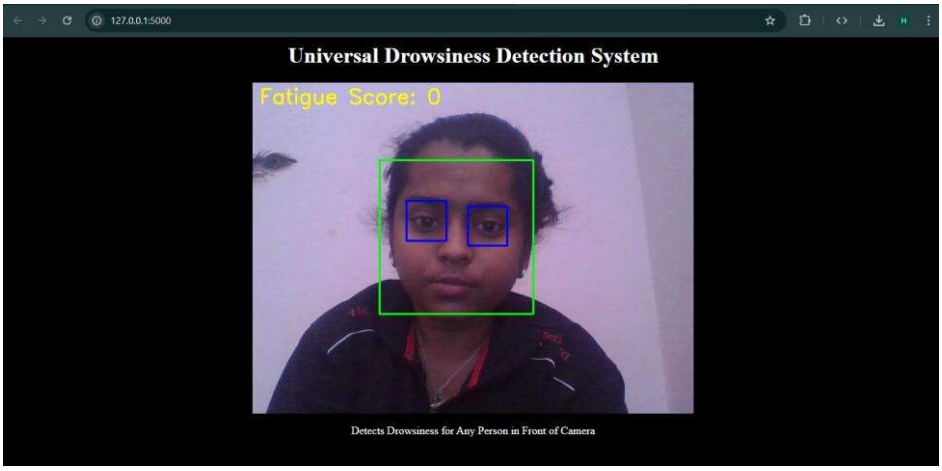

Figure 6.1: *Output when face and eyes are detected normally (No alarm state)*

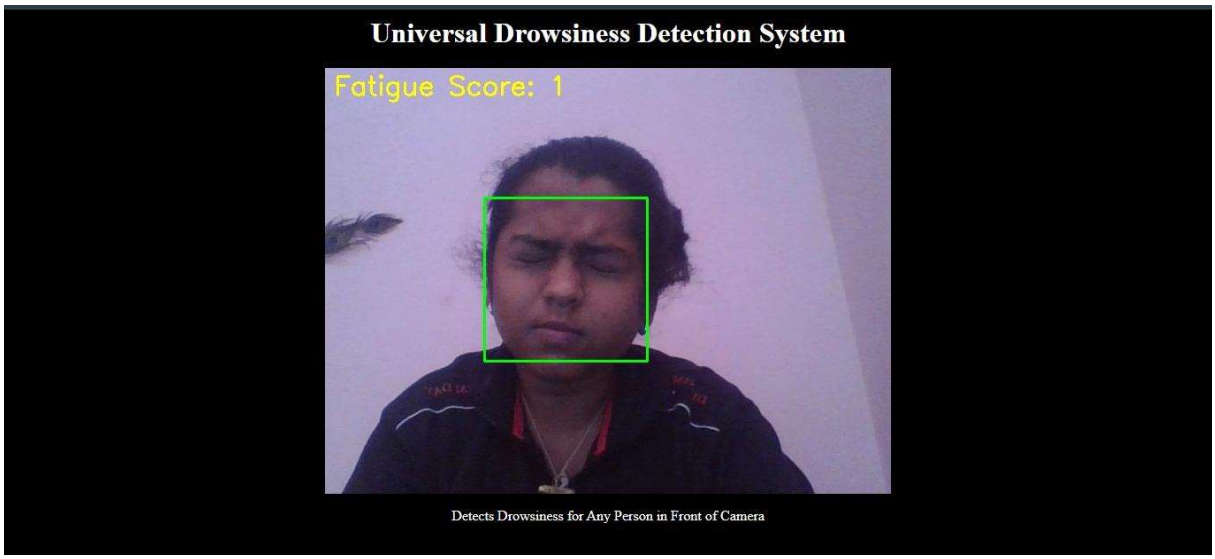

Figure 6.2: Output when eyes are closed and fatigue score is increasing

7. REFERENCES

[1] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1, 511–518.

[2] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.

[3] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788.

[4] OpenCV Team. (2024). *OpenCV Library Documentation*. https://opencv.org/

[5] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

[6] Flask Development Team. (2024). *Flask Web Framework Documentation*. https://flask.palletsprojects.com/

[7] Pygame Community. (2024). *Pygame Audio Library Documentation*. https://www.pygame.org/

[8] World Health Organization (WHO). (2023). *Global Status Report on Road Safety*. World Health Organization Publications.

[9] Abtahi, S., Omidyeganeh, M., Shirmohammadi, S., & Hariri, B. (2011). Yawning detection using embedded smart cameras. *Proceedings of the IEEE International Conference on Multimedia and Expo*, 1–6.

[10] Ji, Q., Yang, X., & Liao, J. (2004). Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. *Real-Time Imaging*, 8(5), 357–377.

[11] Kumar, S., & Nidhin, S. (2019). Real-time driver drowsiness detection system using Haar cascade classifier. *International Journal of Engineering Research & Technology (IJERT)*, 8(6), 1142–1146.

[12] TensorFlow Team. (2024). *TensorFlow and Keras Documentation*. https://www.tensorflow.org/

[13] Scikit-learn Developers. (2024). *Machine Learning in Python*. https://scikit-learn.org/

[14] IEEE Intelligent Transportation Systems Society. (2023). *Research Articles on Driver Safety and Fatigue Detection*. IEEE Publications.

[15] UC Irvine Machine Learning Repository. (2024). *Eye State & Drowsiness Datasets*. https://archive.ics.uci.edu/