Optimized Machine Learning Model for Driver's Drowsiness Detection Using KNN Classifier

Meghana, Pooja H S, Sanjana E, Vinutha Shetty E, Sheela B P

Meghana Information Science & Engineering& RYMEC Ballari
Pooja H S Information Science & Engineering& RYMEC Ballari
Sanjana E Information Science & Engineering& RYMEC Ballari
Vinutha Shetty E Information Science & Engineering& RYMEC Ballari
Sheela B P Assistant Professor Dept of ISE RYMEC Ballari

Abstract - This study explores using machine learning, specifically the K-Nearest Neighbors (KNN) algorithm, to detect driver drowsiness, a major cause of traffic accidents. The research focuses on optimizing the KNN model for accurate and efficient real-time detection. This involves selecting the most informative features from a dataset of driver behavior and physiological signals (like eye movements and facial expressions), tuning the KNN algorithm's parameters for best performance, and possibly reducing the data's complexity. The optimized model's performance is tested, aiming to create a reliable system that can be integrated into vehicles to prevent accidents by alerting drowsy drivers.

***Key Words*:** KNN, Feature Selection, Parameter Tuning, Machine Learning.

1.INTRODUCTION

This project tackles the critical issue of driver drowsiness, a major contributor to traffic accidents, by developing an automated detection system. The system employs the K-Nearest Neighbors (KNN) algorithm, a machine learning technique ideal for pattern recognition. A camera continuously monitors the driver's face, capturing video data which is then processed to extract key features indicative of drowsiness. These features include the eye aspect ratio (EAR), reflecting eye openness; the mouth aspect ratio (MAR), indicating yawning; yawning frequency; and blink frequency. Changes in these metrics, such as drooping eyelids (low EAR), frequent yawning (high MAR and yawning frequency), and slowed blinking, are strong indicators of fatigue.

The extracted features are then fed into the KNN classifier. This algorithm compares the current feature set to a pre-existing, labeled dataset containing examples of alert and drowsy driver states. By identifying the "nearest neighbors" in the dataset, the KNN algorithm classifies the driver's current state in real-time. If the system detects drowsiness, it triggers alerts, which can be auditory or visual. These alerts aim to rouse the driver and encourage them to take a break. This real-time feedback loop is crucial for preventing accidents caused by driver fatigue, offering a promising approach to improving road safety.

Problem Statement:

Road accidents caused by human errors are responsible for numerous fatalities and injuries worldwide. The primary reason behind such accidents is the driver's drowsiness, which could result from sleep deprivation or prolonged driving hours. To address this issue, it is imperative to develop a system that leverages the latest available technologies to minimize the likelihood of accidents. The main objective of this system is to create a model that can issue an alert in case the driver shows signs of drowsiness. This alert will help the driver become aware of their condition and take the necessary measures to prevent an accident.

Objective 1: Develop a Comprehensive Data Collection Framework

- To create a robust data collection system that utilizes a cameras to gather real-time information about the driver's physiological state.

Objective 2: Implement Advanced Drowsiness Detection Algorithm

- To design and deploy machine learning model that accurately identify drowsiness in drivers based on collected data.

Objective 3: Create an Effective Real-time Alert System

- To establish a responsive alert system that notifies drivers of drowsiness in real-time, prompting them to take necessary actions to prevent accidents.

2. LITERATURE REVIEW

2.1 Existing Solutions

One of the existing system is Camera-based driver drowsiness detection systems utilize real-time video monitoring to assess driver alertness by analyzing facial features and eye movements. These systems employ advanced image processing algorithms and machine learning techniques to detect critical indicators of fatigue, such as blink rate, yawning, and head position, and can even identify microsleeps—brief episodes of sleep that can occur without the driver's awareness. By integrating with other vehicle safety features, such as lane departure warnings, these systems provide a comprehensive safety net while ensuring a non-intrusive user experience. Complementing this technology, steering pattern analysis monitors the driver's steering inputs to identify unusual patterns that may indicate drowsiness. By collecting data on steering angles and the frequency of corrections, the system can detect erratic movements that suggest a loss of focus. This real-time feedback mechanism alerts drivers when their behavior indicates potential fatigue, enhancing overall road safety. Together, camera-based systems and steering pattern analysis create a holistic approach to monitoring driver alertness, significantly reducing the risk of accidents caused by drowsiness.

The disadvantages of existing solutions

1. Environmental Limitations: Camera-Based Systems: These systems can struggle in low-light conditions or adverse weather (e.g., rain, fog), which can affect the camera's ability to capture clear images of the driver's face.
2. False Positives/Negatives: Both camera-based and steering pattern analysis systems can produce false positives (alerting the driver when they are not drowsy) or false negatives (failing to alert when the driver is drowsy). This can lead to driver frustration or, worse, a lack of trust in the system
3. Intrusiveness: Some existing systems can be intrusive, providing frequent alerts that may distract the driver rather than assist them. This can lead to cognitive overload, especially if the alerts are not well-timed or relevant..
4. Integration Challenges: Integrating existing drowsiness detection systems with other vehicle safety features can be complex. If not properly synchronized, it may lead to inconsistent performance or user experience.
5. Cost: High-quality camera systems and advanced sensors can be expensive to implement, which may limit their availability in lower-end vehicles. This can create disparities in safety features across different vehicle models.
6. Data Privacy Concerns: Continuous monitoring of the driver's face and behavior raises significant privacy concerns.

2.2 Comparison with the Proposed Solution

Existing System(camera Based System)	KNN-Based Drowsiness Detection System
Image processing	K-nearest neighbour algorithm
High complexity	Medium complexity
Low accurcy	High accuracy
High computational cost	Low computational cost
Non intrusive	Potentially intrusive
Automatic Feature selection	Manual requires careful selection of relevant feature
Large data sets are required	Moderate datasets often sufficient
More complex, requires specialized hardware/software	Simpler to implement

3. PROPOSED SOLUTION

3.1 Project Scope

The scope of driver drowsiness detection using KNN is extensive, impacting automotive safety and beyond. KNN can enhance ADAS features like lane-keeping and collision avoidance by providing real-time driver alertness monitoring. Personalized driver profiles can be created, adapting alert sensitivity to individual driving patterns and drowsiness indicators, especially useful in shared vehicles. Integration with wearables allows for comprehensive fatigue assessment through physiological signal monitoring, benefiting long-haul drivers and offering insights into overall health.

In fleet management, KNN improves driver alertness, reducing accidents and enhancing logistics safety. Aggregated data analysis informs fatigue management training programs. This research contributes to machine learning and AI advancements, leading to sophisticated algorithms. KNN-based systems aid manufacturers in complying with safety standards, potentially influencing vehicle safety ratings and insurance incentives.

These systems also play a role in public awareness campaigns, promoting safe driving and providing driver feedback. Ultimately, KNN-driven drowsiness detection contributes to reducing accidents caused by fatigue, improving road safety for everyone. Its adaptability and potential for integration make it a valuable tool in combating driver drowsiness. Future research may focus on refining its accuracy and expanding its applications.

3.2 High-level Architecture

The high-level architecture for a driver drowsiness detection system using the K-Nearest Neighbors (KNN) classifier consists of several key components. The Data Acquisition Layer includes high-resolution cameras to capture the driver's facial expressions. This data is processed in the Data Processing Layer, where feature extraction occur. The selected features are then analyzed in the KNN Classifier Module, which classifies the driver's state based on a trained dataset of alertness levels.

The Decision-Making Layer generates alerts if drowsiness is detected, using visual, auditory, or haptic signals to notify the driver. The User Interface Layer presents real-time alertness information through a dashboard, potentially supplemented by a mobile app for additional insights. Finally, the Data Storage and Analytics Layer logs data for future analysis, helping to identify patterns in driver behavior and improve system performance. This architecture enables effective real-time monitoring of driver alertness, enhancing road safety through timely alerts and feedback.

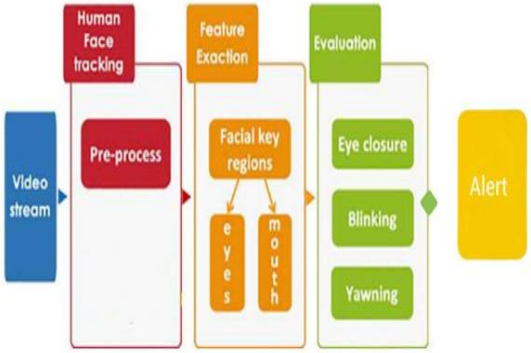

Fig 5.2.1 Architecture of proposed system

3.3 Objective of the Study

Driver drowsiness is a major cause of road accidents. This system uses a KNN classifier to detect drowsiness in real-time, enhancing road safety. High-resolution cameras capture facial expressions, identifying signs like prolonged blinking, yawning. The KNN classifier analyzes this data, categorizing the driver's state as alert, drowsy, or fatigued. Upon detecting drowsiness, the system issues timely visual, auditory alerts prompting the driver to take a break. The system also provides feedback on alertness levels promoting safer practices. The goal is to reduce drowsiness-related accidents and improve road safety. This involves integrating multiple data sources for accurate drowsiness detection. The KNN model is trained on a diverse dataset to recognize fatigue indicators. Alerts are designed to be noticeable yet non-alarming. Long-term feedback helps drivers understand their fatigue patterns. The system aims to create a safer driving environment for all. It leverages advanced technology and machine learning for effective drowsiness detection. This system promotes responsible driving and reduces the risk of fatigue-related accidents. It contributes to a culture of road safety and driver well-being. Future enhancements may include integration with other ADAS features. The system's focus is on real-time intervention and long-term behavioral change. It aspires to minimize the dangers of drowsy driving. The ultimate aim is to save lives and prevent injuries caused by driver fatigue.

4. ANALYSIS

4.1 System analysis

System analysis for a driver drowsiness detection system using the K-Nearest Neighbors (KNN) classifier involves evaluating the requirements, functionalities, and performance metrics of the system. This analysis helps in understanding how the system will operate, the challenges it may face, and the overall impact on driver safety.

4.1.1 The Existing System

Driver drowsiness detection systems aim to enhance road safety by monitoring driver behavior and physiological signs for signs of fatigue. These systems typically utilize cameras to track eye movements, analyze steering patterns, or even monitor facial expressions for signs of drowsiness. While these systems offer the potential to prevent accidents caused by fatigued drivers, they also have limitations.

One significant disadvantage is the potential for false alarms. Factors like bright sunlight, sudden movements, or even the driver's appearance can trigger false alerts, distracting and potentially irritating the driver. Additionally, the accuracy of these systems can vary depending on individual driving styles, environmental conditions, and the specific technology employed. Over-reliance on technology can also lead to a false sense of security, potentially discouraging drivers from taking necessary breaks when they feel fatigued. Furthermore, privacy concerns arise with camera-based systems that constantly monitor the driver's facial expressions.

4.1.2 The Proposed System

The proposed system works by first collecting real-time data from various sources, including video feeds capturing the driver's facial expressions and eye movements. This data is then pre-processed to extract relevant features such as eye aspect ratios, head pose, and steering wheel movements. These extracted features are then used to create a feature vector representing the driver's current state. A K-Nearest Neighbors (KNN) classifier is trained on a labeled dataset of driver behavior, where each data point is classified as "drowsy" or "alert." During real-time operation, the system uses the trained KNN model to classify the driver's current state based on the extracted features. If the system classifies the driver as drowsy, an appropriate alert, such as a visual warning or an audible alarm, is triggered to warn the driver.

This approach aims to improve upon existing methods by leveraging KNN's ability to learn individual driving styles, enabling personalized drowsiness models. By analyzing these individual patterns, the system can potentially reduce false alarms and increase the accuracy of detecting fatigue, ultimately leading to more reliable drowsiness alerts and potentially preventing accidents caused by fatigued drivers.

4.1.3 Hardware and Software Selection

Hardware Selection

- A camera with at least 2.0 MP resolution.
- An audio speaker for alerts.
- A Raspberry Pi or a computer system to connect and process the frames using Python.

Software Selection

Operating system: Windows

Frameworks and Libraries:

- **dlib**: A machine learning library for face recognition and landmark prediction.
- **imutils**: A collection of convenience functions to work with images and video frames.
- **scipy**: Used to calculate yawning and eye focus percentages.
- **OpenCV**: A comprehensive library for image and video processing.
- **argparse**: Provides a command-line interface for running the project with specific parameters.

Development Tools: VS code

4.2 Functional Requirements

Drowsiness Detection:

- The system shall accurately detect driver drowsiness in real-time.
- The system shall utilize multiple cues, such as eye closure, head nods, and steering wheel movements, to identify drowsiness.

Alerting Mechanisms:

- The system shall provide clear and timely alerts to the driver when drowsiness is detected.
- Alerting mechanisms may include visual warnings (e.g., flashing lights, on-screen messages), audible alarms, or haptic feedback.

Performance:

- The system shall have a low false alarm rate to minimize driver distraction.
- The system shall have a high detection rate to effectively identify drowsy drivers.
- The system shall operate reliably in various lighting conditions and environmental conditions.

User Interface:

- The system shall provide a user-friendly interface for system configuration and calibration.
- The system shall provide clear feedback to the driver regarding the system's status and alerts.

Data Privacy:

- The system shall comply with relevant data privacy regulations.
- The system shall minimize the collection and storage of personal data.
- The system shall provide users with control over data collection and usage

4.3 External Interface Requirements

These external interface requirements ensure that the driver drowsiness detection system can effectively interact with users, vehicles, and other systems, providing a seamless and user-friendly experience while maintaining compatibility and interoperability.

4.3.1 User Interface

The user interface design is intentionally minimalistic, prioritizing usability and driver safety. Instead of a graphical user interface (GUI), the system relies on command-line interactions, making it lightweight and efficient. Users interact with the system primarily through terminal commands, entering text-based instructions to control and configure the application. Clear and concise instructions are provided for running the application, covering setup, basic usage, and advanced features. The system provides immediate audio feedback based on the driver's detected state, ensuring the user is constantly aware of their drowsiness level. This real-time feedback is crucial for preventing accidents caused by driver fatigue. The use of voice alerts makes the system accessible to users who may not be able to, or should not, read text while driving. This hands-free approach minimizes distractions and keeps the driver's focus on the road. While lacking a GUI, the system allows users to modify key settings, such as thresholds for yawning frequency and eye focus, directly within the code. This approach provides a high degree of customization, allowing users to tailor the system to their individual needs and driving habits. The overall design philosophy prioritizes safety and functionality, ensuring that drivers receive timely and non-distracting alerts to combat drowsiness and promote safer driving practices. The command-line interface, combined with audio feedback, offers a practical and effective way to alert drivers without adding visual distractions.

5.DESIGN

5.1 Data Design

Data design in the context of the Driver Drowsiness Detection System involves the organization, storage, and processing of data that the system uses to function effectively. This includes the input data (video frames), the processed data (facial landmarks, eye aspect ratios, and yawn distances), and the output data (alerts and visual feedback).

i. Input Data

Video Frames: The primary input data consists of video frames captured from the webcam. Each frame is a 2D array of pixel values representing the image. The frames are processed in real-time, typically at a rate of 30 frames per second (FPS) or higher, depending on the camera and processing capabilities.

Facial Landmark Data: The system uses a pre-trained model (dlib's shape predictor) to detect facial landmarks. This model outputs a set of coordinates for 68 key points on the face, which are used to calculate the Eye Aspect Ratio (EAR) and lip distance.

ii. Processed Data

Eye Aspect Ratio (EAR): The EAR is calculated using the coordinates of the eye landmarks. It is a single floating-point value that indicates whether the eyes are open or closed. The formula for EAR involves the

distances between specific eye landmarks, and it is computed as follows:

EAR= (A + B) / (2.0 * C)

where (A) and (B) are the vertical distances between the eye landmarks, and (C) is the horizontal distance.

Lip Distance: The distance between the upper and lower lips is calculated using the coordinates of the lip landmarks. This value is also a floating-point number and is used to determine if the driver is yawning.

Counters: Two counters are maintained:

COUNTER: Tracks consecutive frames where the EAR is below the threshold, indicating potential drowsiness.

YARN_FRAME: Tracks consecutive frames where the lip distance exceeds the yawning threshold.

iii. Output Data

Alerts: The system generates audio alerts based on the processed data.

There are two types of alerts:

1. Drowsiness Alert: Triggered when the EAR indicates closed eyes for a specified number of consecutive frames.
2. Yawn Alert: Triggered when the lip distance indicates yawning for a specified number of consecutive frames.

Visual Feedback: The system overlays text on the video frames to provide real-time feedback to the user. This includes:

- The current EAR value.
- The current lip distance value.
- Alert messages indicating drowsiness or yawning.

iv. Data Flow

The data flow in the system can be summarized as follows:

Capture: The webcam captures video frames continuously.

Preprocessing: Each frame is resized and converted to grayscale for processing.

Face Detection: The system detects faces in the frame and extracts facial landmarks.

Feature Extraction: The EAR and lip distance are calculated from the facial landmarks.

Condition Checking: The system checks the EAR and lip distance against predefined thresholds to determine if alerts should be triggered.

Alert Generation: If conditions are met, audio alerts are played, and visual feedback is displayed on the video stream.

Loop: The process repeats for each frame until the user terminates the program.

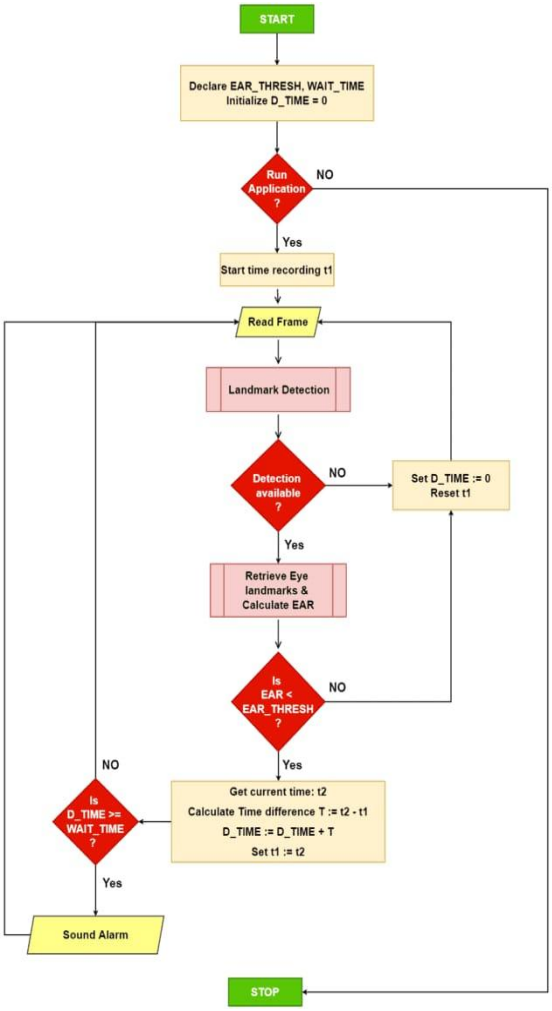

Fig 5.1.1 Data Flow Diagram

5.2 Architectural Design

The architectural design of the Driver Drowsiness Detection System outlines the overall structure and components of the system, detailing how they interact with each other to achieve the desired functionality.

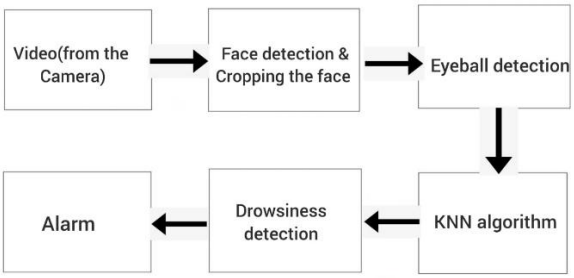

Fig 5.2.1 Architecture of proposed system

System Components:

- Human Face tracking: This component detects the presence of a human face in the video stream and tracks its position over time.
- Feature Extraction: This component extracts key facial features such as eyes, mouth, and their movements.
- Evaluation: This component analyzes the extracted features to determine if they indicate drowsiness. It might use thresholds and algorithms to identify patterns associated with drowsiness.
- Alert: This component generates an alert when drowsiness is detected. This involves visual alerts and audible alert.

5.3 Interface Design

The interface design of the Driver Drowsiness Detection Alert System focuses on functionality and user interaction.

- Camera Interface: Utilizes a webcam or camera module to capture video frames. The system processes these frames in real-time to detect drowsiness indicators.
- Alert System: Provides audio alerts through speakers when drowsiness is detected. Alerts include specific voice commands like "Open your eyes, sir" and "Take some fresh air, sir".
- Command-Line Interface (CLI): The application is run via a command-line interface using argparse. Users can specify parameters and configurations directly from the terminal.
- Real-Time Feedback: The system continuously monitors the driver's state and provides immediate feedback through audio alert

5.3.1 User Interface Design

- The user interface design is minimalistic and focuses on usability.
- The interface does not have a graphical user interface (GUI) but relies on command-line interactions.
- Users interact with the system primarily through terminal commands.
- Clear instructions are provided for running the application, including setup and usage.
- The system provides immediate audio feedback based on the driver's state, ensuring that the user is aware of their drowsiness status.
- The use of voice alerts makes the system accessible to users who may not be able to read text while driving.
- Users can modify settings such as thresholds for yawning and eye focus directly in the code, allowing for customization based on individual needs.

Overall, the user interface design prioritize safety and functionality, ensuring that drivers receive timely alerts without distractions.

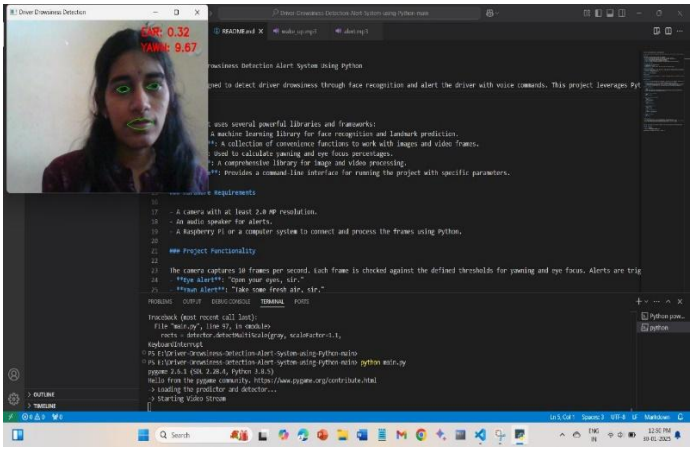

Fig 5.3.1 User Interfac

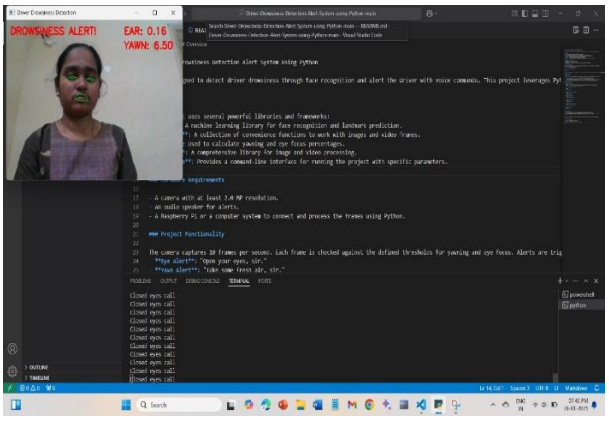

Fig 5.3.2 Drowsiness Alert for closed eyes call

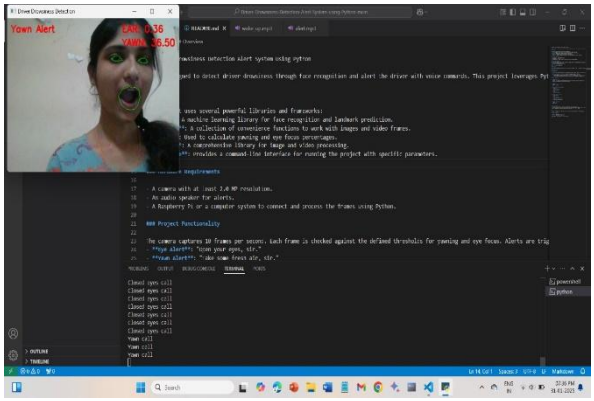

Fig 5.3.3 Drowsiness alert for Yawn call

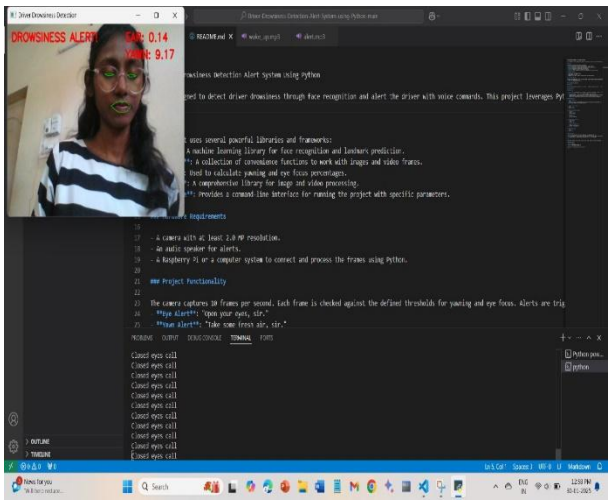

Fig 5.3.4 Drowsiness detection with glasses

5.4 Procedural Design

It describes the process of developing and deploying a driver drowsiness detection system. It outlines the steps from data collection to real-time monitoring and alerting the driver.

Start: The process begins with the start of the system.

Create Dataset: The first step is to create a dataset of images or videos to train the drowsiness detection model. If the dataset already exists, this step is skipped.

Train Model: After collecting the dataset, the system trains a model to detect drowsiness based on facial features or eye movements. This step is executed only if a pre-trained model doesn't exist.

Get Live Camera Feed: Once the model is trained, the system starts receiving live camera feed from the driver's vehicle.

Detect Face and Eye Region: The system processes the camera feed, detecting the driver's face and focusing on the eye region.

Detect Drowsiness: The system analyzes the detected eye region to detect signs of drowsiness, such as eye closure or blinking patterns.

Alert The Driver: If the system detects drowsiness, it triggers an alert to inform the driver about their state of alertness. The alert could be in the form of an audio or visual notification.

Continue Detecting: If drowsiness is not detected, the system continues to monitor the driver's face and eye region for drowsiness signs.

This cycle continues as long as the system is active, continuously monitoring the driver and alerting them in case of drowsiness detection.

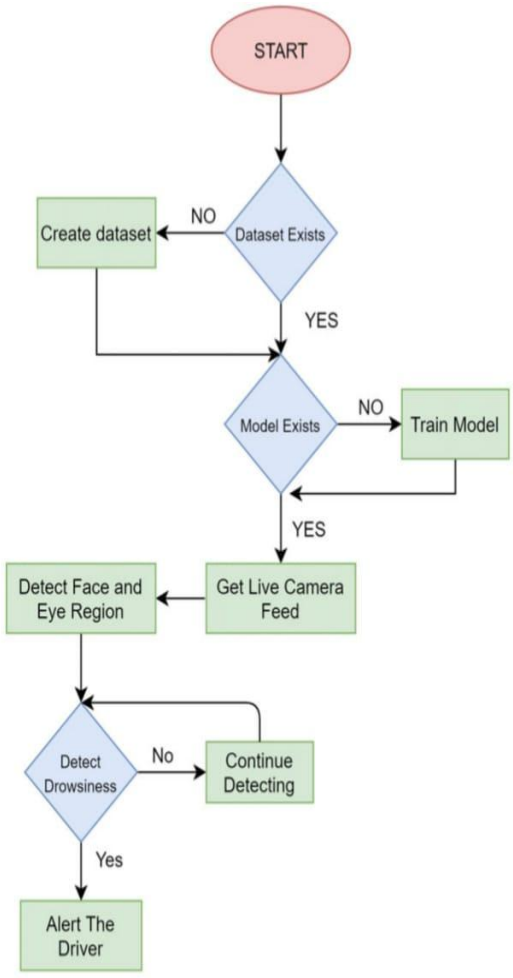

Fig 5.4.1 Procedural Design of Proposed System

6.PSEUDO CODE

#Main.py

```
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
from threading import Thread
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
import os
from pygame import mixer

mixer.init()
sound1 = mixer.Sound('wake_up.mp3')
sound2 = mixer.Sound('alert.mp3')

def alarm(msg):
    global alarm_status
    global alarm_status2
    global saying

    while alarm_status:
        print('Closed eyes call')
        saying = True
        sound1.play()
        saying = False
    if alarm_status2:
        print('Yawn call')

        saying = True
        sound2.play()
        saying = False


def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear

def final_ear(shape):
                (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
                (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

    leftEye = shape[lStart:lEnd]
    rightEye = shape[rStart:rEnd]

    leftEAR = eye_aspect_ratio(leftEye)
    rightEAR = eye_aspect_ratio(rightEye)

    ear = (leftEAR + rightEAR) / 2.0
    return (ear, leftEye, rightEye)

def lip_distance(shape):
    top_lip = shape[50:53]
    top_lip = np.concatenate((top_lip, shape[61:64]))

    low_lip = shape[56:59]
    low_lip = np.concatenate((low_lip, shape[65:68]))

    top_mean = np.mean(top_lip, axis=0)
    low_mean = np.mean(low_lip, axis=0)
```

```
    distance = abs(top_mean[1] - low_mean[1])
    return distance

ap = argparse.ArgumentParser()
ap.add_argument("-w", "--webcam", type=int, default=0,
            help="index of webcam on system")
args = vars(ap.parse_args())

EYE_AR_THRESH = 0.2
EYE_AR_CONSEC_FRAMES = 35
YAWN_THRESH = 30
YAWN_CONSEC_FRAMES = 30  # New threshold for yawn detection
alarm_status = False
alarm_status2 = False
saying = False
COUNTER = 0
YARN_FRAME = 0  # New counter for yawn frames

print("-> Loading the predictor and detector...")
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

print("-> Starting Video Stream")
vs = VideoStream(src=args["webcam"]).start()

while True:

    frame = vs.read()
    frame = imutils.resize(frame, width=450)
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

            rects = detector.detectMultiScale(gray, scaleFactor=1.1,
                                minNeighbors=5, minSize=(30, 30),
                                flags=cv2.CASCADE_SCALE_IMAGE)

    for (x, y, w, h) in rects:
        rect = dlib.rectangle(int(x), int(y), int(x + w), int(y + h))

        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)

        eye = final_ear(shape)
        ear = eye[0]
        leftEye = eye[1]
        rightEye = eye[2]

        distance = lip_distance(shape)

        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)

        lip = shape[48:60]
        cv2.drawContours(frame, [lip], -1, (0, 255, 0), 1)

        if ear < EYE_AR_THRESH:
            COUNTER += 1
```

```
                if COUNTER >= EYE_AR_CONSEC_FRAMES:
                    if not alarm_status:
                        alarm_status = True
                        t = Thread(target=alarm, args=('wake up sir',))
                        t.daemon = True
                        t.start()

                    cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            else:
                COUNTER = 0
                alarm_status = False

            if distance > YAWN_THRESH:
                YARN_FRAME += 1

                if YARN_FRAME >= YAWN_CONSEC_FRAMES:
                    if not alarm_status2:
                        alarm_status2 = True
                        t = Thread(target=alarm, args=('take some fresh air sir',))
                        t.daemon = True
                        t.start()

                    cv2.putText(frame, "Yawn Alert", (10, 30),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            else:
                YARN_FRAME = 0
                alarm_status2 = False

            cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            cv2.putText(frame, "YAWN: {:.2f}".format(distance), (300, 60),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

        cv2.imshow("Driver Drowsiness Detection", frame)
        key = cv2.waitKey(1) & 0xFF

        if key == ord("q"):
            break

    cv2.destroyAllWindows()
    vs.stop()
```

7.Testing

Testing a drowsiness detection system involves several steps to ensure that each component functions correctly and that the system as a whole performs as expected.

Test Type	Description	Test Cases
1.Unit Testing	Tests individual components such as functions and modules in isolation.	dataset_exists(), train_model(), detect_drowsiness() - verifying correct input/output, error handling.
2.Integration Testing	Tests how components work together.	Data flow between camera feed, face detection, and drowsiness assessment. Ensuring smooth

		transitions between modules.
3.Functional Testing	Tests system features against requirements .	Simulating various drowsiness levels, testing alert triggering, edge cases. (poor lighting, glasses).
4.Performance	Evaluates speed, resource usage, and accuracy.	Time from image capture to alert, system load under stress, false positive/negative rates.
5.User Acceptance Testing	Real users (drivers) test in a controlled environment.	Gathering feedback on alert effectiveness, ease of use, comfort levels.
6.Field Testing	Real-world driving tests.	Evaluating performance in diverse conditions (weather, traffic), long-term reliability.
7.Iterative Testing	Continuous testing and improvement based on feedback.	Addressing bugs, refining algorithms, improving user experience based on test results.

8.Conclusion

This project successfully developed a driver drowsiness detection system using the KNN algorithm and EAR, achieving promising accuracy by analyzing changes in eye openness. This simple, computationally efficient approach offers a valuable tool for improving driver safety. Trained and validated on EAR data, the system effectively distinguishes between alert and drowsy states. While currently limited by factors like lighting and head pose, future work will focus on enhancing robustness through image preprocessing and data augmentation. Integrating additional features like blink rate and exploring adaptive alerts are also planned to further improve performance and real-world applicability.

9. ANNEXURE

Machine Learning: Machine learning is a subfield of artificial intelligence (AI) focused on enabling computers to learn from data without being explicitly programmed. Machine learning algorithms identify patterns, make predictions, and improve their accuracy over time by being exposed to more data.

KNN Classifier: The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

Supervised learning: Supervised learning is a machine learning technique that uses labeled data to train algorithms to predict outcomes. It's a common way to build machine learning models.

Eye Aspect Ratio(EAR): Eye aspect ratio (EAR) is a metric that measures how open or closed a person's eyes are. It's calculated using the distance between the upper and lower eyelids.

Computer Vision: Computer vision is a type of artificial intelligence (AI) that allows computers to understand and interpret visual data.

Image Processing: Image processing is a broad term that refers to any kind of manipulation or analysis of digital images. It involves using algorithms and techniques to modify images, extract information from them, or enhance their quality.

10. REFERENCES

https://www.blackbox.ai/

https://gemini.google.com/app?hl=en-IN

https://scholar.google.com/

https://doi.org/10.1109/access.2019.2958667

Mehta, S., Dadhich, S., Gumber, S., Bhatt, A.J.: Real-time driver drowsiness detection system using eye aspect ratio and eye closure ratio. SSRN Electron. J. (2019). https://doi.org/10.2139/ssrn.3356401

Sathasivam, S., Mahamad, A.K., Saon, S., Sidek, A., Som, M.M., Ameen, H.A.: Drowsiness detection system using eye aspect ratio technique. In 2020 IEEE Student Conference on Research and Development (SCOReD) (2020). https://doi.org/10.1109/scored50371.2020.9251035

Abtahi, S., Omidyeganeh, M., Shirmohammadi, S., Hariri, B.: YawDD: yawning detection dataset. IEEE Dataport (2020). https://doi.org/10.21227/e1qm-hb90.

COMPUTATIONALLY EFFICIENT FACE DETECTION; B. SCHLKOPF-A.BLAKE, S. ROMDHANI, AND P. TORR.

USE OF THE HOUGH TRANSFORMATION TO DETECT LINES ANDCURVES IN PICTURE; R. DUDA AND P. E. HART.

JAIN, "FACE DETECTION IN COLOR IMAGES; R. L. HSU, M.ABDEL-MOTTALEB, AND A. K. JAIN.

OPEN/CLOSED EYE ANALYSIS FOR DROWSINESS DETECTION; P.R.TABRIZI AND R. A. ZOROOFI.

National Highway Traffic Safety Administration. "Traffic safety facts crash stats:Drowsy driving 2019," Oct. 2017. [Online]. Available: http://www.nhtsa.gov/risky-driving/drowsy-driving

European New Car Assessment Program. "Euro NCAP 2025 Roadmap," Sep.2019. [Online]. Available:https://cdn.euroncap.com/media/30700/euroncap-roadmap-2025-v4.pdf

A. Sahayadhas, K. Sundaraj, and M. Murugappan, "Detecting driver drowsiness based on sensors: A review," Sensors, vol. 12, no. 12, pp. 6937–16953, Dec. 2018.

Y. Dong, Z. Hu, K. Uchimura, and N. Murayama, "Driver inattention monitoringsystem for intelligent vehicles: A review," IEEE Trans. Transp. Syst., vol. 12, no. 2,pp. 596–614, Jun. 2020.

C. Bila, F. Sivrikaya, M. A. Khan, and S. Albayrak, "Vehicles of the future: Asurvey of research on safety issues," IEEE Trans. Intell. Transp. Syst., vol. 18, no.5, pp. 1046–1065, 2020.

D. Liu, P. Sun, Y. Xiao, and Y. Yin, "Drowsiness Detection Based on Eyelid Movement," in Education [1]Technology and Computer Science (ETCS), 2010 Second International Workshop on, 2010, pp. 49-52.

T. Danisman, I. M. Bilasco, C. Djeraba, and N. Ihaddadene, "Drowsy driver detection system using eye blink patterns," in Machine and Web Intelligence(ICMWI), 2010 International Conference on, 2010, pp. 230-233.

Qing, W., BingXi, S., Bin, X., & Junjie, Z. (2010, October). A perclos-based driverfatigue recognition application for smart vehicle space. In Information Processing(ISIP), 2010 Third International Symposium on (pp. 437-441). IEEE.

Nakano.T, Suzuki.M ,Yamamoto.N, Yamamoto.O and Yamamoto.S ,Measurement of driver's consciousness by image processing a method forpresuming driver's drowsiness by eye- blinks coping with individual differences.‖Systems, Man and Cybernetics , vol. 4, 2006.

Bradski.G, Kaehler.A, -Learning OpenCV, O'Reilly, 2008.

Snehal S.Bharambe and P.M.Mahajan, "Implementation of Real Time Driver Drowsiness Detection System" IJSR, vol 4 issue1, Jan 2015.

Pratyush Agarwala and Rizul Sharma, "Driver Drowsiness Detection Techniques:Review" EasyChair, Dec 21,2019.

Tejal Jadhav, Siddhi Gajare, Shubhman Salve and Prof. Hani Patil, "Advanced Driver Assistance System for drivers using Machine Learning and Artificial Intelligence Techniques" IJRASET, vol 10 issue V, May 2022.

BIOGRAPHIES

Meghana

Final Year Information Science & Engineering Student at RYMEC, Ballari.

Pooja H S

Final Year Information Science & Engineering Student at RYMEC, Ballari.

Sanjana E

Final Year Information Science & Engineering Student at RYMEC, Ballari.

Vinutha Shetty E

Final Year Information Science & Engineering Student at RYMEC, Ballari.

Prof SHEELA B.P

Faculty member in the Information Science and Engineering Department at RYMEC College, Ballari. With extensive expertise in machine learning, artificial intelligence, and data science.