Implementation of Driver Fatigue Detection System for Loco Pilot, Preventing Train Accidents Using Python and ESP32 CAM

Mr. Vivekanand P. Thakare[1], Miss. Rupali Bhalavi[2], Miss. Sana Shaikh[3], Miss. Simran Khan[4]
[1]Assistant Professor, Department Of Computer Science Engineering, Govindrao Wanjari College Of Engineering & Technology, Nagpur, India.
[2,3,4]Student, Department Of Computer Science Engineering, Govindrao Wanjari College Of Engineering & Technology, Nagpur, India

Abstract— **Fatigue among train drivers, especially loco pilots, has been identified as a critical factor contributing to train accidents worldwide. Prolonged working hours, monotonous routes, and high stress can lead to drowsiness and reduced alertness, severely impairing a driver's ability to respond promptly to signals and track conditions. In response to this safety challenge, the proposed system leverages the capabilities of embedded systems and artificial intelligence to develop a real-time Driver Fatigue Detection System using the ESP32-CAM module and Python-based image processing techniques. The primary aim of this project is to proactively detect signs of drowsiness or inattention in the loco pilot and initiate preventive measures to avoid accidents.**

This system uses an ESP32-CAM, a compact microcontroller with an onboard camera and Wi-Fi capability, to continuously monitor the facial features of the loco pilot. The camera captures real-time video frames of the driver's face, which are processed using machine learning algorithms either on-device or remotely using a server running Python. The Python-based backend employs computer vision techniques, particularly OpenCV and pre-trained models such as Haar cascades or Dlib's facial landmark detectors, to detect key indicators of fatigue such as eye closure duration (PERCLOS), blink rate, yawning frequency, and head tilt. These features are essential in estimating the alertness level of the driver.

Index Terms:- Driver Fatigue Detection, Loco Pilot Monitoring, Train Accident Prevention, ESP32-CAM, Python, OpenCV, Real-time Image Processing, Eye Blink Detection, Yawning Detection, Drowsiness Detection, Face Monitoring System, Internet of Things (IoT), Machine Learning, Computer Vision, Railway Safety System.

I. INTRODUCTION

In recent years, the safety of railway transportation has come under increasing scrutiny due to the rising number of accidents caused by human error, particularly driver fatigue. Loco pilots, who are responsible for operating trains over long and often monotonous routes, are especially susceptible to drowsiness, reduced alertness, and mental exhaustion. Unlike road vehicles where driver fatigue can sometimes be compensated by co-passengers or traffic signals, in rail transport, a fatigued loco pilot poses a direct and severe risk to hundreds of passengers and cargo. Traditional safety mechanisms such as dead man switches or periodic manual checks have proven insufficient to detect real-time fatigue or ensure continuous alertness. This has created an urgent need for intelligent systems that can monitor loco pilots in real time and initiate safety responses automatically when signs of fatigue are detected.

The Driver Fatigue Detection System using Python and ESP32CAM is an innovative solution designed to address this critical problem. The system aims to monitor the loco pilot's facial features and behavior continuously using a compact, low-cost hardware module—the ESP32CAM—paired with Python based image processing algorithms. The ESP32CAM is a WiFi enabled microcontroller with an onboard camera capable of capturing video in real time. When installed in the locomotive cabin, it captures video frames of the loco pilot's face, which are then analyzed to detect signs of fatigue such as prolonged eye closure (PERCLOS), frequent blinking, yawning, and head tilting.

The backend system, developed in Python, utilizes computer vision libraries like OpenCV and pertained models such as Hear cascades or Dlib's facial landmark detection to identify and track facial expressions and eye movements. These visual indicators are processed to determine the alertness level of the loco pilot. If the system detects fatigue, it immediately activates warning mechanisms such as buzzers, lights, or vibration motors, alerting the driver to regain focus. Additionally, the system can send real-time alerts to a centralized railway monitoring station using WiFi or IoT protocols like HTTP or MQTT, ensuring rapid intervention if necessary.

What makes this system particularly suitable for railway environments is its scalability, portability, and cost-effectiveness. The ESP32CAM is a compact and inexpensive board that can be easily installed in any locomotive without major modifications. Its wireless capability allows integration into existing train communication infrastructure, supporting remote monitoring and data logging for future analysis. Over time, fatigue pattern data collected from multiple drivers can also be used to optimize work schedules, develop better safety policies, and improve the overall wellbeing of railway personnel.

II. RELETED WORK

Various studies have explored effective methods for detecting driver fatigue using advanced technologies. Odhner et al. utilized facial expression recognition and eye blink detection to identify fatigue through yawning and eye movements. Miyashita et al. investigated heart rate variability (HRV) as a physiological marker, highlighting its reliability in detecting cognitive decline in drivers. Scherer et al. examined multiple detection techniques, including facial recognition, motion tracking, and biometric sensors, emphasizing their importance in high-risk environments like railways. Cheng et al. proposed machine learning models specifically SVMs and random forests—to predict fatigue using data from wearable sensors. Lastly, Karimi et al. reviewed eye-tracking methods, confirming that blink rate and eye closure duration are strong indicators of fatigue, especially useful for monitoring loco pilots. Collectively, these studies support the development of real-time, multi-modal fatigue detection systems for improving railway safety.

III. METHODOLOGY

The implementation of a Driver Fatigue Detection System for loco pilots involves a multidisciplinary approach combining embedded systems, computer vision, machine learning, and IoT technologies. The goal is to develop a reliable, real-time system that continuously monitors the loco pilot's facial expressions to detect signs of fatigue and issue timely alerts to prevent train accidents. The methodology for designing and implementing this system consists of the following key stages:

Hardware Setup using ESP32-CAM The core hardware component used in this system is the ESP32-CAM module, a low-cost microcontroller with an integrated camera and Wi-Fi functionality. The ESP32-CAM is mounted inside the locomotive cabin in such a way that it has a clear view of the loco pilot's face. The module is powered using a suitable DC power source and programmed to capture live video frames at regular intervals.

The ESP32-CAM captures real-time facial images of the loco pilot and either processes them onboard (for simple analysis) or streams them over a network to a Python-based server for more advanced processing. If onboard resources are limited, frames are transmitted via Wi-Fi using HTTP or MQTT protocols to a nearby edge device or remote server.

Once the image is received, preprocessing is performed using OpenCV in Python. This includes:

Converting images to grayscale.

Normalizing brightness/contrast to handle different lighting conditions.

Applying noise reduction techniques to enhance image quality.

Facial Feature Detection Using Haar cascade classifiers, Dlib, or MediaPipe, the system detects key facial landmarks:

Eyes

Mouth

Nose

Eyebrows

Head position

These landmarks are used to analyze facial behavior that indicates fatigue.

Fatigue Detection Algorithms The system applies logic to detect the following indicators:

PERCLOS (Percentage of Eye Closure): Calculates how long the eyes remain closed within a specific time frame.
Blink Rate: Monitors the number of blinks per minute; excessive or reduced blinking may indicate fatigue.
Yawning Detection: OpenCV is used to detect the frequency and duration of mouth opening.
Head Tilt Detection: Tracks nodding or downward head movements using facial landmarks.
Threshold values for these indicators are defined based on standard fatigue research data. If any threshold is exceeded, the driver is considered drowsy or inattentive.

Alert Mechanism Once fatigue is detected, the system activates real-time alerts:
Audio Alert: A buzzer or speaker plays a loud sound to wake the driver.
Vibration Alert: Optional vibrating module placed on the seat or console.
Visual Alert: A flashing LED can serve as an additional warning.

Remote Monitoring and Data Logging All detection events and status updates are sent over Wi-Fi to a central monitoring server. The server logs the data, displays driver alertness status on a dashboard, and can notify authorities via SMS, email, or a control panel interface.

Testing and Validation The system is tested in simulated and real environments. Accuracy of fatigue detection is validated against known datasets and manually observed driver behavior. Feedback from railway staff is used to optimize sensitivity and reduce false positives.

IV. SYSTEM ARCHITECTURE

The Driver Fatigue Detection System is designed as a real-time, intelligent monitoring solution that integrates embedded hardware, computer vision software, and wireless communication technologies to detect signs of drowsiness in loco pilots and trigger timely alerts. The system architecture is modular, comprising hardware, software, processing logic, and communication interfaces, ensuring scalability, portability, and reliability in various locomotive environments.

1. Hardware Layer

a. ESP32-CAM Module - At the core of the hardware is the ESP32-CAM, a compact microcontroller with a built-in camera and Wi-Fi module. It captures live video or still images of the loco pilot’s face from the locomotive cabin. The ESP32-CAM is chosen for its affordability, low power consumption, and wireless capabilities, making it ideal for rail-based embedded applications.

b. Power Supply Unit -The ESP32-CAM is powered by a stable DC power source, typically derived from the locomotive’s auxiliary power system, or via a regulated battery pack in portable deployments.

c. Alerting Mechanisms -Includes: Buzzer/Speaker: For audio alerts when drowsiness is detected. Vibration Motor (optional): Installed on the seat or dashboard to provide tactile feedback. LED Indicator: Visual warning to attract attention.

2. Image Capture & Preprocessing Layer

The ESP32-CAM continuously captures image frames of the driver's face. The images are either: Processed locally (for lightweight tasks), or Transmitted via Wi-Fi to a Python-based processing unit (Raspberry Pi, laptop, or cloud server).

Captured frames are preprocessed using OpenCV libraries to convert to grayscale, apply histogram equalization, and remove noise. This improves the quality and accuracy of subsequent facial detection.

3. Facial Feature Detection Layer

This layer is powered by Python with OpenCV and machine learning libraries. It detects and tracks facial features such as:

Eyes (open/closed)

Eyelids (blink rate and duration)

Mouth (for yawning detection)

Head tilt/position

Techniques used:

Haar Cascade Classifiers for eye and face detection.

Dlib or MediaPipe for facial landmark extraction. Algorithms to compute PERCLOS (Percentage of Eye Closure), blink frequency, and yawning detection.

4. Decision-Making and Alert Layer

If fatigue indicators such as:

 Eyes closed for >3 seconds

 Excessive yawning

 Frequent head nodding

are detected, the decision logic triggers local alerts using buzzers or vibrations.

Severity levels can be categorized as:

 Level 1: Mild drowsiness – Audio alert only.

 Level 2: Moderate – Audio + vibration.

 Level 3: Severe – Alert + Remote message to control center.

 5. Communication and IoT Layer

All fatigue events and alert status are sent via Wi-Fi using HTTP or MQTT protocol to a central monitoring dashboard. This enables: Real-time tracking of loco pilot status. Emergency communication with control stations. Data storage for further analysis and training.

 6. Monitoring and Logging Layer

A Python-based web server (e.g., Flask or Django) receives the data and displays it on a live dashboard for railway supervisors. Logs are stored in a database (e.g., SQLite or MySQL) for future review and pattern analysis.

V. CONCLUSION

Railway safety is a matter of utmost importance in any nation's transportation infrastructure. Among the various factors contributing to train accidents, driver fatigue is one of the most prevalent yet under-addressed issues. Loco pilots often work under physically and mentally challenging conditions, including long hours, repetitive tasks, and limited rest, which lead to drowsiness and reduced cognitive functioning. Traditional safety systems in trains do not provide real-time monitoring of a driver's alertness level, leaving a critical gap in accident prevention mechanisms.

The proposed Driver Fatigue Detection System using Python and ESP32-CAM provides a novel and effective solution to bridge this gap. This system combines embedded hardware, computer vision algorithms, and wireless communication technologies to offer continuous, real-time monitoring of loco pilot behavior. By analyzing key facial features such as eye closure, blinking rate, yawning, and head movements, the system accurately identifies signs of fatigue or drowsiness. Once fatigue is detected, it promptly triggers alerts in the form of audio warnings, vibrations, and visual signals, thereby giving the loco pilot immediate feedback and encouraging alertness.

The ESP32-CAM plays a crucial role in this system due to its compact size, low power consumption, and built-in camera and Wi-Fi capabilities. Its affordability makes the system highly scalable, enabling widespread implementation across locomotives with minimal infrastructure changes. Meanwhile, the Python-based processing unit performs sophisticated image analysis using OpenCV and machine learning models to detect fatigue indicators with accuracy and speed. The system's ability to communicate with remote monitoring stations via IoT protocols further enhances its value by enabling real-time alerts to railway authorities for necessary intervention.

REFERENCES

[1] T. R. Ismail, M. K. Omar, & S. A. Ghani. (2020). "**Real-Time Driver Drowsiness Detection for Embedded System Using Eye Aspect Ratio**." *International Journal of Advanced Computer Science and Applications (IJACSA)*, Vol. 11, No. 4.

[2] Kazemi, V., & Sullivan, J. (2014). "**One Millisecond Face Alignment with an Ensemble of Regression Trees**." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[3] M. Garai & A. Dasgupta. (2019). "**Smart Real-Time Driver Drowsiness Detection System Using Machine Learning and Computer Vision**." *Procedia Computer Science*, Vol. 152, pp. 105–112.

[4] Espressif Systems. (2023). **ESP32-CAM Development Board** – Technical Reference Manual. Retrieved from: https://www.espressif.com/en/products/devkits/esp32-cam

[5] Dlib Library. (2023). **Machine Learning and Facial Landmark Detection Toolkit**. Retrieved from: http://dlib.net

[6] National Transportation Safety Board (NTSB). (2020). **Safety Report: Reducing Fatigue-Related Accidents in Transportation**. Retrieved from: https://www.ntsb.gov

[7] Indian Railways. (2022). **Railway Safety Annual Report**. Ministry of Railways, Government of India. Retrieved from: https://indianrailways.gov.in

[8] MQTT.org. (2024). **MQTT v3.1.1 Specification** – Lightweight Messaging Protocol for IoT. Retrieved from: https://mqtt.org

[9] Flask Web Framework. (2024). **Official Flask Documentation** – Python-Based Micro Web Framework. Retrieved from: https://flask.palletsprojects.com

[10] Raspberry Pi Foundation. (2023). "**IoT Projects Using ESP32 and Python**." Retrieved from: https://www.raspberrypi.org

[11] Smith, R., & Zhang, Y. (2018). "IoT-enabled Greenhouse Monitoring System with Wireless Sensor Networks." IEEE Sensors Journal, 18(5), 2342-2350.

[12] Yadav, S., & Singh, M. (2021). "A Low-Cost IoT Solution for Small-Scale Farmers." Agricultural Informatics Journal, 10(2), 87-95.

[13] Gutiérrez, J., Villa-Medina, J. F., Nieto-Garibay, A., & Porta-Gándara, M. Á. (2014). Automated irrigation system using a wireless sensor network and GPRS module. IEEE Transactions on Instrumentation and Measurement, 63(1), 166-176. https://doi.org/10.1109/TIM.2013.2276487

[14] Li, S., Xu, L. D., & Zhao, S. (2015). The internet of things: a survey. Information Systems Frontiers, 17(2), 243-259. https://doi.org/10.1007/s10796-014-9492-7

[15] Castellanos, M., & Calderón, J. M. (2019). IoT-enabled systems for monitoring and controlling agricultural environments. International Journal of Smart and Connected Agriculture, 3(2), 45-58.

[16] Mahesh, T., & Krishna, M. (2020). Implementation of IoT in smart greenhouses: A review. International Journal of Emerging Technology and Advanced Engineering, 10(8), 123-130.

[17] Rajasekaran, A., & Ramachandran, R. (2021). Cloud-based greenhouse monitoring system using IoT. Proceedings of the International Conference on IoT in Smart Agriculture, 98-103. DOI: 10.xxxx/IoT.2021.56891

[18] Jahnavi, R., & Sharma, A. (2021). IoT solutions for precision agriculture and greenhouse management. Springer Advances in IoT Technology, 12, 223-240.

[19] Hasan, M. K., Alam, M. S., & Rahman, M. M. (2022). Energy-efficient IoT-based greenhouse automation system. Journal of Green Energy Technology, 5(1), 34-47.

[20] Arduino Blog. (n.d.). Smart farming and IoT applications in greenhouses. Retrieved from https://blog.arduino.cc