IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151073

# AI-Powered Smart Glasses for Digital Eye Strain and Drowsiness Detection

**Sanmaya I K[1], Sharanya S[1], Vinusha D Souza[1], Maneesha[1], Bharathi K[2]**

Students, Department of Computer Science & Engineering, Vivekananda College of Engineering and Technology, Puttur, India[1]

Assistant Professor, Department of Computer Science & Engineering, Vivekananda College of Engineering and Technology, Puttur, India[2]

**Abstract:** Prolonged digital screen usage can contribute to visual discomfort, eye fatigue, and drowsiness, particularly during extended periods of computer use. This paper presents an AI-powered smart glasses system designed to monitor eye activity and identify patterns associated with digital eye strain and drowsiness. The proposed architecture incorporates a Convolutional Neural Network (CNN) for eye-state classification and a Long Short-Term Memory (LSTM) network for temporal analysis of eye-related features. Eye activity is monitored using facial and eye landmarks, from which eye aspect ratio and related features are extracted. Sequential analysis is used to distinguish normal eye activity from patterns associated with eye strain and drowsiness. The system architecture also includes an alert mechanism intended to notify the user when potentially concerning eye-activity patterns are detected. The implementation work includes CNN-based eye-image classification, LSTM-based state classification, and a real-time eye-monitoring prototype. The approach provides a foundation for wearable monitoring of eye activity and supports further integration into a compact smart glasses platform.

**Keywords:** Smart Glasses, Digital Eye Strain, Drowsiness Detection, Convolutional Neural Network, Long Short-Term Memory, Eye Aspect Ratio, Real-Time Monitoring.

## I. INTRODUCTION

Computers, laptops, tablets and smartphones are central to education, work and entertainment, but long hours in front of screens can cause digital eye strain, with symptoms such as dry or irritated eyes, blurred vision, headaches and difficulty focusing. Continuous work also leads to tiredness and drowsiness, which is especially dangerous for drivers, factory workers and others who must stay alert. Fatigue reduces concentration and reaction time, so early identification of its signs is valuable.

Most existing drowsiness detection systems monitor the face and eyes with an external camera. Such systems can be effective but may be expensive and inconvenient to carry, and a camera that continuously records the user raises privacy concerns. Other approaches use specialized sensing hardware, which adds cost and complexity.

This paper proposes AI-Powered Smart Glasses, a compact wearable that monitors blink rate, blink duration and eye-closure time and uses machine learning to classify the user's state as normal, strained or drowsy. A Raspberry Pi 4 runs a CNN for eye-state recognition and an LSTM for temporal analysis, and alerts are given through a vibration motor and an LED. IoT connectivity is optional, so the core function does not depend on the internet and data stays on the device unless the user enables connected features. The main contributions are:

- A single wearable system that addresses both digital eye strain and drowsiness, whereas most prior work targets only one.
- A modular architecture combining CNN-based eye-image classification and LSTM-based temporal analysis for eye strain and drowsiness monitoring.
- A local-first design with optional IoT connectivity that improves reliability and privacy.
- A software-based real-time eye-monitoring prototype, providing a foundation for integration into a wearable smart glasses platform.

 This work is licensed under a Creative Commons Attribution 4.0 International License 

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151073

## II. LITERATURE REVIEW

Recent studies emphasize that monitoring eye activity, blink patterns, and fatigue markers is essential for addressing computer vision syndrome and drowsiness. Lapa et al. [1] investigated real-time blink detection as an indicator of computer vision syndrome in real-life settings, demonstrating the feasibility of monitoring eye behavior during screen usage. Similarly, Yin et al. [2] evaluated VDT-induced visual fatigue through the automatic detection of blink features, highlighting how computational metrics can quantify visual discomfort.

To address deeper behavioral and fatigue states, Park et al. [3] applied feature representation learning using deep neural networks for driver drowsiness detection. For efficient real-time spatial feature extraction, Soukupová and Čech [4] established a foundational approach using facial landmarks to calculate the Eye Aspect Ratio (EAR) for rapid eye-blink detection. Earlier foundational work by Divjak and Bischof [5] explored eye blink-based fatigue detection specifically for the prevention of computer vision syndrome.

**Research Gap:** While existing studies validate that blink rate, eye-closure duration, and facial landmarks are reliable indicators of visual strain and fatigue, most prior systems focus exclusively on a single condition—such as driver drowsiness or separate blink counters—often relying on heavy external infrastructure or high computational loads. The proposed system bridges this gap by combining a lightweight MediaPipe-based landmark extraction pipeline with a compact CNN-LSTM architecture deployed locally on a Raspberry Pi 4 to monitor both digital eye strain and drowsiness in a single wearable platform.

## III. PROPOSED SYSTEM

The proposed system is a wearable that continuously observes the wearer's eye region and classifies the state as normal, strained or drowsy. It emphasizes four design goals: real-time detection, portability, low power consumption and user comfort. These goals trade off against each other. A more powerful processor would speed up inference but increase power use, heat and size, making the glasses heavier, so hardware and software were chosen to balance performance and practicality. The system comprises the following units:

- Input unit: Camera module capturing the eye region.
- Processing unit: Raspberry Pi 4 for preprocessing and model execution.
- AI detection unit: CNN for eye state, LSTM for temporal behavior.
- Decision unit: Evaluates blink rate, blink interval and eye-closure duration to classify the condition.
- Alert unit: Vibration motor and LED indicator.
- Power unit: Rechargeable power bank.
- Optional IoT/user interface: Mobile application or website for monitoring, history and settings.

The processing, AI detection and decision units are software stages running on the same Raspberry Pi, while the input, alert and power units are physical hardware. The overall architecture is shown in Fig. 1.



Fig. 1: System Architecture

 This work is licensed under a Creative Commons Attribution 4.0 International License 

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151073

## IV. METHODOLOGY

*A. Hardware Implementation*

The proposed smart glasses system is designed around a camera module, a processing unit, and an alert mechanism. The camera is positioned near the front of the glasses to capture the user's eye region during screen-related activities. The processing unit handles image acquisition, eye-feature extraction, and AI-based classification. A Raspberry Pi 4 is considered for embedded execution of the models, while a power bank provides portable power. A transistor-based switching circuit can be used to control the vibration motor.

*B. Preprocessing and CNN Eye-State Detection*

The convolutional neural network processes normalized eye region images to evaluate the user's immediate eye state. The preprocessing pipeline resizes input frames to a fixed spatial resolution and scales pixel intensities to standard numerical ranges. MobileNetV2 is implemented as the base feature extractor using transfer learning, paired with a custom dense classification head trained to categorize eye states into awake and sleepy conditions. The resulting model artifact is integrated into the system pipeline for automated image-level classification.

*C. Feature Extraction*

Eye-related features are extracted to represent changes in eye openness over time. MediaPipe Face Mesh identifies facial landmarks around the left and right eyes, allowing the Eye Aspect Ratio (EAR) to be calculated. The current feature-extraction pipeline uses left-eye EAR, right-eye EAR, average EAR, change in average EAR, eye-closure state, baseline ratio, and baseline difference. A personal eye-opening baseline is estimated during calibration to support relative analysis of eye activity. These features are organized into sequential observations for temporal classification.

*D. LSTM Temporal Analysis and Classification*

A Long Short-Term Memory network is used to analyze temporal patterns in eye-related features. The model processes sequences containing 60 consecutive frames, with seven numerical features representing eye activity in each frame. Feature standardization is performed using a scaler fitted to the training data. The LSTM classifies the observed sequence into Normal, Eye Strain, or Drowsiness categories. Temporal analysis enables the system to consider changes across multiple frames rather than relying only on an individual observation.

*E. Alerts and Optional IoT*

The system architecture includes an alert mechanism intended to notify the user when eye activity indicates a potentially concerning state. A vibration motor provides a physical warning. These components operate through the processing unit and associated control circuitry. Optional connectivity through a mobile application or web interface can provide access to status information and activity records.

*F. Algorithm*

1. Initialize the camera and the required software components.
2. Capture frames containing the user's face and eye region.
3. Detect facial landmarks and locate the eye regions.
4. Calculate eye-related features, including left-eye EAR, right-eye EAR, average EAR, and changes in eye openness.
5. Apply baseline-related feature calculations and organize observations into sequences.
6. Standardize the feature sequences and pass them to the trained LSTM model.
7. Determine the estimated state as Normal, Eye Strain, or Drowsiness using model predictions and defined decision rules.
8. Display the detected state and continue monitoring eye activity.
9. Activate the physical alert or transmit status information when the respective hardware or connectivity component is integrated and operational.

## V. IMPLEMENTATION AND RESULTS

*A. Software Environment*

The software components were developed using Python and TensorFlow/Keras. The CNN model uses MobileNetV2 for eye-image classification, while the LSTM model analyzes sequential eye-related features. NumPy and Pandas support numerical processing and dataset preparation, and Scikit-learn is used for feature standardization. MediaPipe Face Mesh is used in the real-time prototype to detect facial landmarks and calculate eye-related features. Model training and real-time testing are performed using the available development environment.

 This work is licensed under a Creative Commons Attribution 4.0 International License 

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151073

*B. CNN and LSTM Model Implementation*

The CNN model was trained using eye images organized into two categories, awake and sleepy. Transfer learning was applied using pretrained MobileNetV2 layers, followed by a custom classification head. The trained model was saved as a separate model file for image-level prediction.

The LSTM model was trained using sequential feature data representing Normal, Strain, and Drowsy states. Each sequence contains 60 frames, with seven eye-related features per frame. Training and testing sequences were separated at the video level, and a StandardScaler was fitted on the training data before being applied to the testing data. The trained model and scaler were saved for use in real-time inference.

*C. Real-Time Eye Monitoring*

A real-time monitoring prototype was developed using camera input and MediaPipe Face Mesh. Eye landmarks are used to calculate the Eye Aspect Ratio, and a calibration stage estimates a baseline for the user's eye openness. The extracted features are organized into sequences and supplied to the trained LSTM model. The predicted state is combined with decision rules to display the estimated condition during monitoring. The current prototype demonstrates the software-based eye-monitoring and temporal-classification workflow.

*D. Testing and Observations*

The CNN and LSTM components were developed and evaluated separately. The CNN supports image-level classification into awake and sleepy categories, while the LSTM processes sequential eye-feature data to classify Normal, Eye Strain, and Drowsiness states. The real-time prototype was observed processing camera frames, extracting eye-related features, and displaying state changes during testing.

Table I: Test Case Verification

| ID | Test case | Expected outcome | Actual outcome | Result |
|---|---|---|---|---|
| 1 | Camera Initialization | Camera initializes successfully | Camera opened successfully | Pass |
| 2 | Real-Time Frame Capture | Live video frames are captured | Live video frames were displayed | Pass |
| 3 | Eye Landmark Detection | Eye landmarks are detected using MediaPipe | Eye landmarks were detected in the live feed | Pass |
| 4 | EAR Calculation | Eye Aspect Ratio values are calculated | Left, right and average EAR values were obtained | Pass |
| 5 | LSTM State Prediction | Model predicts Normal, Strain or Drowsy | State predictions were generated by the LSTM model | Pass |
| 6 | Real-Time State Display | Predicted state appears on screen | State labels were displayed during monitoring | Pass |

*E. Discussion and Limitations*

The developed components provide a foundation for wearable eye monitoring using image classification and temporal analysis. The use of facial landmarks enables numerical eye-feature extraction, while the LSTM considers changes in these features across consecutive frames. However, variations in lighting, camera placement, facial movement, and individual eye behavior may influence the extracted features and classification results. The integrated CNN–LSTM pipeline and the complete wearable alert mechanism require further verification under practical operating conditions. The system is intended as an assistive monitoring tool and not as a medical diagnostic device.

## VI. CONCLUSION

This work presents an AI-powered smart glasses approach for monitoring eye activity and identifying patterns associated with digital eye strain and drowsiness. The development includes a MobileNetV2-based CNN for eye-image classification, an LSTM model for sequential eye-feature analysis, and a real-time prototype that extracts eye landmarks and EAR-based features using MediaPipe Face Mesh. The system architecture combines image-level eye-state analysis

DOI: 10.17148/IJARCCE.2026.151073

with temporal pattern recognition and includes an alert mechanism for wearable use. The developed components provide a foundation for an integrated smart glasses system intended to support awareness of eye fatigue and drowsiness during prolonged screen use.

***Scope for Future Work***

Future work includes:

- A fully developed mobile application and cloud storage for long-term eye and fatigue records.
- Lighter and more efficient models to improve accuracy while reducing load on the Raspberry Pi.
- Additional sensors such as heart rate and head movement to strengthen fatigue detection.
- Broader real-world testing across users, lighting conditions and usage durations.

## ACKNOWLEDGMENT

We express our sincere gratitude to our project guide, Prof. Bharathi K, for her valuable guidance, encouragement, and support throughout this work. We thank Prof. Pradeep Kumar K G, Head of the Department of Computer Science and Engineering, and Dr. Mahesh Prasanna K, Principal, for their encouragement and for providing the necessary facilities. We also extend our thanks to the college management and all teaching and non-teaching staff members of the department for their support in completing this project.

## REFERENCES

[1]. I. Lapa, S. Ferreira, C. Mateus, N. Rocha, and M. A. Rodrigues, "Real-Time Blink Detection as an Indicator of Computer Vision Syndrome in Real-Life Settings: An Exploratory Study," *International Journal of Environmental Research and Public Health*, vol. 20, no. 5, Art. no. 4569, 2023. doi: 10.3390/ijerph20054569.

[2]. Z. Yin, B. Liu, D. Hao, L. Yang, and Y. Feng, "Evaluation of VDT-Induced Visual Fatigue by Automatic Detection of Blink Features," *Sensors*, vol. 22, no. 3, Art. no. 916, 2022. doi: 10.3390/s22030916.

[3]. S. Park, F. Pan, S. Kang, and C. D. Yoo, "Driver Drowsiness Detection System Based on Feature Representation Learning Using Various Deep Networks," in *Computer Vision – ACCV 2016 Workshops*, Lecture Notes in Computer Science, vol. 10118, pp. 154–164, Springer, 2017. doi: 10.1007/978-3-319-54526-4_12.

[4]. T. Soukupová and J. Čech, "Real-Time Eye Blink Detection Using Facial Landmarks," in *Proceedings of the 21st Computer Vision Winter Workshop*, 2016.

[5]. M. Divjak and H. Bischof, "Eye Blink Based Fatigue Detection for Prevention of Computer Vision Syndrome," in *Proceedings of the 11th IAPR Conference on Machine Vision Applications*, pp. 350–353, 2009.

 This work is licensed under a Creative Commons Attribution 4.0 International License 