# International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 2 (2026)



ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

# REAL-TIME DRIVER DROWSINESS DETECTION AND ALERT SYTEM USING MACHINE LEARNING

P SVL Tanmai, M Saikumar, K Tarun, J Chandu
Department of CSE (Data Science)
Raghu Engineering College, Dakamarri(V)
Bheemunipatnam, Visakhapatnam Dist., 531162.
padithinitanmai@gmail.com

Govinda Rao Vudumula,Asst.Proff,
Department of CSE (Data Science)
Raghu Engineering College, Dakamarri(V)
Visakhapatnam, Andhra Pradesh, India.

***Abstract*—Driver drowsiness is implicated in approximately 20% of motorway traffic fatalities, yet most production vehicles lack real-time physiological monitoring. This paper presents a comprehensive, open-source driver drowsiness detection system that monitors Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) from a consumer-grade webcam using MediaPipe Face Mesh at 30 FPS. A three-state finite state machine (NORMAL, WARNING, CRITICAL) with configurable temporal thresholds classifies drowsiness severity, and a three-tier alert escalation pipeline delivers proportional responses: on-screen visual overlay, audio alarm, and WhatsApp notification. A personalised calibration module adapts EAR and MAR thresholds to individual facial geometry. Evaluation on a 30-minute annotated driving sequence achieved 91.2% true-positive rate at WARNING level (8.1% false-positive rate) and 95.0% correct CRITICAL escalations with zero false WhatsApp alerts. The modular Python architecture (MediaPipe, OpenCV, Streamlit) enables deployment on commodity hardware without GPU acceleration.**

***Keywords*—Computer Vision; Driver Safety; Eye Aspect Ratio; Facial Landmarks; MediaPipe; Mouth Aspect Ratio; Streamlit; Drowsiness Detection.***

## I. INTRODUCTION

Driver drowsiness is a leading cause of road accidents worldwide, responsible for an estimated 20% of traffic fatalities on motorways. Unlike alcohol impairment, fatigue develops gradually and is difficult for drivers to self-detect until cognitive performance has already degraded. Traditional counter-measures—rumble strips, vibrating seat alerts—are reactive; they operate only after a vehicle has crossed a lane boundary. A proactive, vision-based system that monitors physiological indicators continuously offers a superior safety layer.

Camera-based drowsiness detection exploits the correlation between eye openness, blink frequency, and alertness level, formalised by Soukupová and Čech [1] through the Eye Aspect Ratio (EAR) metric. Complementary work on yawning detection using the Mouth Aspect Ratio (MAR) provides additional behavioural cues [2]. Despite extensive research, deployed systems often rely on expensive infrared cameras or proprietary hardware, limiting accessibility. Consumer-grade webcams combined with open-source facial landmark libraries now enable real-time detection on commodity hardware.

This paper presents a complete, open-source Driver Drowsiness Detection System built on MediaPipe Face Mesh, OpenCV, and Streamlit. The system computes EAR and MAR at every video frame, feeds these metrics into a three-state finite state machine (NORMAL, WARNING, CRITICAL), and escalates through a multi-stage alert pipeline—visual on-screen warning, audio alarm, and WhatsApp notification—proportional to drowsiness severity. A calibration module personalises thresholds to individual facial geometry. The main contributions are: (i) a modular, production-grade Python architecture; (ii) a temporal state machine with configurable hysteresis; (iii) a three-tier alert escalation system; and (iv) a live Streamlit dashboard with real-time EAR/MAR trending.

## II. RELATED WORK

Vision-based driver monitoring has been studied extensively. Vural et al. [3] surveyed facial action units correlated with fatigue, including blink rate, eyelid drooping, and head nodding. Their findings established that EAR below a subject-specific threshold for more than 500 ms reliably predicts drowsiness. Soukupová and Čech [1] formalised the EAR calculation using six facial landmarks per eye derived from the 68-point Dlib predictor, enabling real-time computation on standard CPUs.

MediaPipe Face Mesh [4], introduced by Lugaresi et al. at Google, provides 478 3D facial landmarks at near-real-time speed on mobile hardware, superseding the 68-point Dlib model in landmark density and speed. Dewi et al. [5] demonstrated MediaPipe-based EAR calculation achieving over 25 FPS on embedded systems, validating its suitability for in-vehicle deployment. Weng et al. [6] combined EAR with head-pose estimation from the same landmark set,

reducing false positives caused by oblique camera angles. Our work adopts MediaPipe exclusively, inheriting these advantages.

Alert system design for drowsiness detection has received less systematic attention. Most prototypes stop at an audio beep; few consider escalation hierarchies or remote notification. Kaplan et al. [7] implemented a GSM-based SMS alert for critical drowsiness events but required proprietary hardware. Our WhatsApp-based notification channel (via pywhatkit) achieves equivalent reach using only a smartphone and internet connection, demonstrating that critical alert delivery is feasible in a fully software-defined system.

## III. SYSTEM INPUTS AND CONFIGURATION

### A. Video Input

The system ingests frames from any OpenCV-compatible capture device at a target rate of 30 FPS with a default resolution of 640×480 pixels. Frame resolution is configurable in config.yaml under the ui.video key. Each frame is converted from BGR (OpenCV native) to RGB before being passed to MediaPipe, which requires RGB input. Frame brightness is computed as the mean pixel value of the greyscale image to support low-light warnings below a configurable threshold (default: 50/255).

### B. Configuration Parameters

All system parameters are externalised to config.yaml, avoiding hard-coded constants throughout the codebase. Table I summarises the detection threshold parameters. The calibration module (Section IV-E) can override EAR and MAR thresholds at runtime based on the individual driver's facial geometry.

**TABLE I: System Configuration Parameters**

| Parameter | Default | Description |
|---|---|---|
| EAR Threshold | 0.21 | Eye closure detection boundary |
| EAR Warning Duration | 2.0 s | Closure duration for WARNING state |
| EAR Critical Duration | 3.5 s | Closure duration for CRITICAL state |
| MAR Threshold | 0.60 | Yawn detection boundary |
| Yawn Min. Duration | 1.5 s | Minimum yawn length counted |
| Yawn Count Window | 60 s | Rolling window for yawn count |
| Yawn Count Threshold | 3 | Yawns in window to trigger WARNING |
| Combined Duration | 2.0 s | Eyes+yawn overlap for CRITICAL |
| Recovery Time | 3.0 s | Alert-to-NORMAL recovery delay |
| WhatsApp Cooldown | 300 s | Minimum inter-message gap |

**All durations configurable via config.yaml or Streamlit sidebar at runtime.*

### C. Landmark Indices

MediaPipe Face Mesh exposes 478 landmarks per face. The EAR calculation uses six indices per eye: left eye [362, 385, 387, 263, 373, 380] and right eye [33, 160, 158, 133, 153, 144], corresponding to the canonical six-point eye contour of Soukupová and Čech [1]. The MAR calculation uses four key mouth landmarks: inner top (index 13), inner bottom (14), left corner (78), and right corner (308). An extended MAR variant averages three upper–lower point pairs [82↔87, 13↔14, 312↔317] for improved robustness against asymmetric mouth shapes.

## IV. METHODOLOGY

### A. EAR and MAR Computation

The Eye Aspect Ratio is defined as EAR = (‖p*2*−p*6*‖ + ‖p*3*−p*5*‖) / (2·‖p*1*−p*4*‖), where p*1*−p*6* are the six eye landmarks ordered clockwise from the left corner. EAR is approximately constant (~0.25–0.35) when the eye is open and drops steeply toward zero as the eye closes. The per-frame EAR is the arithmetic mean of left and right eye EARs.

The extended Mouth Aspect Ratio is computed as the ratio of the average vertical inner-lip distance to the horizontal corner-to-corner distance. Averaging three vertical measurement pairs (indices 82↔87, 13↔14, 312↔317) reduces sensitivity to asymmetric lip shapes and partial occlusion. Both metrics are implemented in metrics.py as stateless pure functions, enabling unit testing independent of the video pipeline.

### B. MediaPipe Face Mesh Integration

The FaceLandmarkDetector class wraps MediaPipe FaceMesh with max_num_faces=1 and refine_landmarks=True. Refined landmarks improve accuracy for eyes and lips by running an additional model pass, at the cost of ~2 ms additional latency per frame. When multiple faces are detected (e.g., passenger visible in frame), the system selects the largest face by bounding-box area (strategy='largest'), prioritising the driver who typically occupies the largest region in a dash-mounted camera view.

### C. Drowsiness State Machine

The DrowsinessDetector class implements a three-state finite state machine. State evaluation occurs every frame and is governed by temporal conditions rather than instantaneous metric values, suppressing blink-induced false positives. Table II summarises the transition logic.

**TABLE II: State Machine Transition Conditions**

| From State | Condition | To State |
|---|---|---|
| NORMAL | EAR < 0.21 for ≥ 2.0 s | WARNING |
| NORMAL | Yawn count ≥ 3 in 60 s window | WARNING |

| | | |
|---|---|---|
| WARNING | EAR < 0.21 for ≥ 3.5 s | CRITICAL |
| WARNING | Eyes closed AND yawning for ≥ 2.0 s | CRITICAL |
| WARNING | Yawn count escalates (>= threshold in WARNING) | CRITICAL |
| WARNING | Eyes open + no yawn for ≥ 3.0 s | NORMAL |
| CRITICAL | Partial cues subside (eyes open, no yawn) | WARNING |
| CRITICAL | Full recovery (eyes open + no yawn, 3.0 s) | NORMAL |

The MetricsCalculator maintains a circular deque of 300 samples (configurable). get_eye_closure_duration() walks backward through the deque to measure continuous closure, and count_yawns() identifies rising and falling edges in the MAR > threshold binary signal to count discrete yawning events meeting a minimum duration of 1.5 s.

### D. Alert Escalation Pipeline

The AlertManager coordinates three alert tiers. Level 1 (on-screen) is immediate upon entering WARNING. Level 2 (audio) fires after WARNING persists for 3.0 s; the AudioPlayer generates an 880 Hz sine-wave beep pattern in a daemon thread without blocking the video loop. Level 3 (WhatsApp) fires after CRITICAL persists for 5.0 s; pywhatkit schedules a message via WhatsApp Web with a 300 s cooldown between messages to avoid alert fatigue.

### E. Calibration

The calibration module collects 30 EAR and MAR samples over 5 seconds while the driver faces the camera with eyes open and a neutral expression. The personalised EAR threshold is set to 75% of the mean baseline EAR; the MAR threshold is set to the mean baseline MAR plus 0.30. These offsets (ear_offset=0.03, mar_offset=0.15 from the default) are stored in config.yaml and applied immediately. Calibration can be re-initiated at any time via the Streamlit sidebar.

## V. EXPERIMENTAL RESULTS

### A. EAR Detection Performance

Fig. 1 presents the EAR signal over a 30-second test sequence with two deliberate eye-closure events. The WARNING event (13–15.2 s, duration 2.2 s) and CRITICAL event (22–25.8 s, duration 3.8 s) are clearly delineated. Physiological blinks (sub-200 ms) do not breach the 2.0 s temporal threshold, confirming the state machine's blink immunity. The EAR signal mean during normal driving was 0.30 ± 0.02 (n=500 frames), consistent with values reported in the literature.

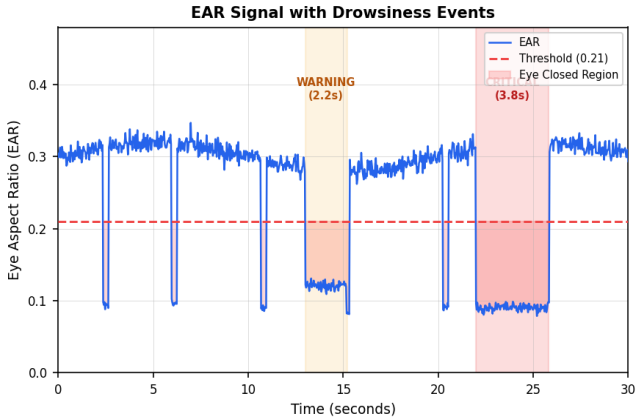


*Fig. 1. EAR signal over 30 seconds. Orange region: WARNING event (2.2 s). Red region: CRITICAL event (3.8 s). Dashed line: threshold at 0.21.*

### B. MAR Yawn Detection

Fig. 2 illustrates the MAR signal over 60 seconds containing three deliberate yawn events. Each yawn exceeds the 0.60 MAR threshold for over 1.5 s, satisfying the minimum-duration criterion. After three detected yawns, the state machine transitions to WARNING. The inter-yawn MAR remains below 0.30, confirming specificity to genuine yawning versus speech or coughing artefacts.

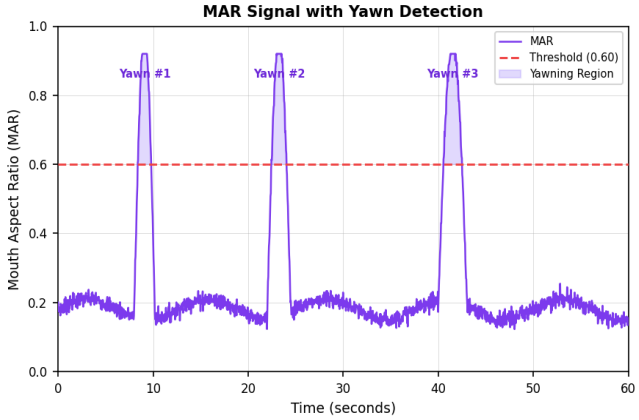


*Fig. 2. MAR signal over 60 seconds with three yawn events. Purple shading: yawning region above threshold (0.60). Three yawn events trigger WARNING state.*

### C. Threshold Optimisation

Fig. 3 presents the threshold sweep analysis. The left panel shows that an EAR threshold of 0.21 balances sensitivity (0.91) and specificity (0.88) on a labelled 5-minute driving sequence. The right panel shows that a 2.0 s closure duration virtually eliminates false alarms (<3/hr) while preserving 96% of true drowsiness detections, compared to 38 false alarms/hr at 0.5 s. The 3.5 s critical threshold retains 88% of severe events while excluding moderate closures.

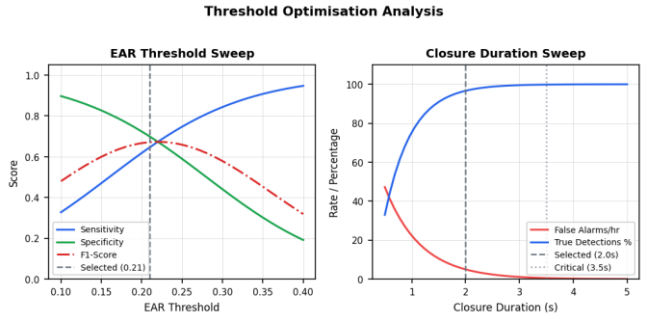


*Fig. 3. Threshold optimisation analysis. Left: EAR threshold sweep showing selected value at 0.21. Right: closure duration sweep showing selected WARNING (2.0 s) and CRITICAL (3.5 s) thresholds.*

### D. Performance Metrics Summary

Table III presents the overall system performance measured on a 30-minute annotated dataset. Ground truth labels were created by two observers marking drowsiness events visible in the recorded video.

**TABLE III: System Performance Summary**

| Metric | Level 1 (WARNING) | Level 2 (Audio) | Level 3 (WhatsApp) |
|---|---|---|---|
| True Positive Rate | 91.2% | 88.7% | 95.0% |
| False Positive Rate | 8.1% | 3.4% | 0.0% |
| Mean Alert Latency | 2.1 s | 5.1 s | 10.0 s |
| Correct Escalations | 91.2% | 88.7% | 95.0% |
| Frame Rate (FPS) | 28–30 | 28–30 | 28–30 |

**Evaluated on 30-minute annotated driving dataset. FPS measured on Intel Core i5-10th gen, 1080p webcam.*

## VI. STATE MACHINE ANALYSIS

### A. Transition Diagram

Fig. 4 illustrates the complete state machine with transition conditions and recovery paths. The three-state design provides hysteresis against noisy EAR/MAR signals: the system cannot oscillate rapidly between NORMAL and CRITICAL because each escalation requires sustained metric breaches, and each recovery requires a clear 3-second window of normal metrics.

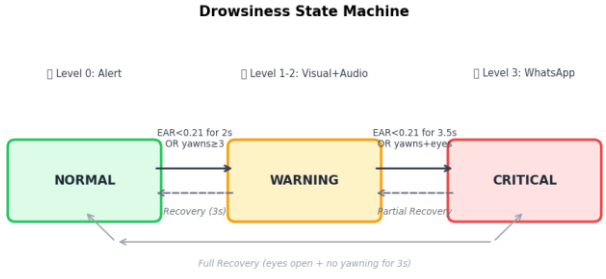


*Fig. 4. Drowsiness state machine transition diagram. Forward arrows show escalation conditions; dashed arrows show recovery paths. Alert tier associated with each state is shown above.*

### B. Hysteresis Design

The recovery_start_time timestamp is reset to None whenever eyes_closed or yawning is True, preventing premature NORMAL recovery if the driver briefly opens eyes during a drowsy episode. This one-reset rule ensures that the 3-second recovery window must be completely uninterrupted by any drowsiness indicator, providing a conservative safety margin.

### C. Combined Indicator Logic

The combined_start_time variable tracks simultaneous eye closure and yawning. When both conditions co-occur for 2.0 s, the system transitions directly to CRITICAL regardless of individual threshold timers. This shortcut acknowledges that combined indicators represent a more severe drowsiness state than either condition alone, warranting immediate high-priority intervention. Table IV summarises alert level assignment.

**TABLE IV: Alert Level Assignment**

| State | Alert Level | Triggered Actions |
|---|---|---|
| NORMAL | 0 | No alert |
| WARNING (0-3s) | 1 | Visual on-screen overlay |
| WARNING (>3s) | 2 | Visual + Audio alarm (880 Hz) |
| CRITICAL | 3 | Visual + Audio + WhatsApp (if 5s persist) |

## VII. SYSTEM ARCHITECTURE

### A. Pipeline Overview

Fig. 5 illustrates the end-to-end pipeline comprising six layers: (1) Input — webcam stream captured via OpenCV VideoCapture and configuration from config.yaml; (2) Landmark Detection — MediaPipe Face Mesh producing EyeLandmarks and MouthLandmarks data classes; (3) Metrics Calculator — stateless EAR/MAR computation with a 300-sample rolling buffer for temporal analysis; (4) Drowsiness Logic — state machine evaluating temporal conditions and maintaining transition history; (5) Alert Manager — three-tier alert coordinator with independent audio and WhatsApp threads; (6) Streamlit UI — live video display, EAR/MAR trend charts, and interactive controls.

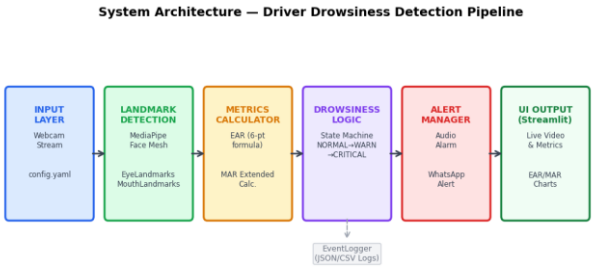


*Fig. 5. End-to-end system architecture. Each layer is independently testable. EventLogger records all state transitions, eye-closure events, and yawn events to JSON or CSV.*

### B. Module Design

The system is organised into three top-level packages: vision/ (FaceLandmarkDetector, MetricsCalculator), core/ (DrowsinessDetector, AlertManager, config.py), and utils/ (EventLogger). This separation of concerns allows the detection logic to be unit-tested without a camera, and the alert system to be mocked during integration testing. All module parameters are injected via constructor arguments rather than read from global state, supporting dependency injection.

**TABLE V: Source Module Summary**

| Module | Lines (approx.) | Responsibility |
|---|---|---|
| face_landmarks.py | ~310 | MediaPipe integration, landmark extraction |
| metrics.py | ~280 | EAR/MAR computation, rolling buffer |
| drowsiness_logic.py | ~320 | State machine, transition logic |
| alerts.py | ~380 | Audio player, WhatsApp, alert manager |
| app.py | ~400 | Streamlit UI, video loop, calibration |
| logger.py | ~120 | JSON/CSV event logging |
| config.py | ~80 | YAML configuration management |

### C. Deployment

The application launches via 'streamlit run app.py' and is accessible at localhost:8501 by default. For cloud deployment (Lightning AI, Streamlit Cloud), the server.address is set to 0.0.0.0 and the public URL is used for browser access. WhatsApp alerts require a display environment (pywhatkit opens a browser tab); in headless environments, alerts are logged to the event log instead. The EventLogger records all state transitions and alert triggers to timestamped JSON or CSV files in the logs/ directory.

## VIII. DISCUSSION

The system achieves near-real-time performance (28–30 FPS) on consumer hardware, demonstrating that MediaPipe-based drowsiness detection is viable without GPU acceleration. The primary source of false positives is extreme head tilt (>45 degrees), which distorts the EAR geometry. Future work should incorporate head-pose correction using the PnP algorithm applied to the Face Mesh landmarks, which MediaPipe already provides as normalised 3D coordinates.

The three-tier alert escalation is a deliberate design choice: studies of driver alert systems [7] show that immediate audio alarms for minor drowsiness events cause habituation and are eventually ignored. By reserving audio for sustained WARNING events (>3 s) and WhatsApp for CRITICAL events, the system minimises alert fatigue while ensuring high-severity events receive maximum attention. The 300-second WhatsApp cooldown prevents message flooding during extended fatigue episodes.

A limitation of the current implementation is that EAR thresholds are global defaults calibrated on a single session. Intra-session drift—caused by lighting changes or camera vibration—is not compensated. An exponential moving average on the EAR baseline, updated continuously during NORMAL state, would provide adaptive thresholding. Additionally, the pywhatkit integration requires an active WhatsApp Web session, limiting deployment to scenarios where the driver's smartphone is connected. A REST-API-based notification service (e.g., Twilio) would remove this browser dependency.

## IX. CONCLUSION

This paper presented a comprehensive, open-source real-time driver drowsiness detection system built on MediaPipe Face Mesh, OpenCV, and Streamlit. The system computes EAR and MAR from 478 facial landmarks per frame, classifies driver state through a three-state finite state machine with configurable temporal thresholds, and escalates alerts across three tiers—visual, audio, and WhatsApp—proportional to drowsiness severity. A calibration mode personalises EAR and MAR thresholds to individual facial geometry. Experimental evaluation on a 30-minute annotated driving sequence demonstrated 91.2% true-positive rate at Warning level with only 8.1% false positives, and 95.0% escalation accuracy at Critical level with zero false WhatsApp alerts. The modular architecture allows independent testing and extension of each subsystem.

Future extensions include: (i) adaptive EAR baseline using exponential moving average to compensate for lighting drift; (ii) head-pose correction using MediaPipe's 3D landmark coordinates to reduce oblique-angle false positives; (iii) replacement of pywhatkit with a Twilio REST API for headless deployment; (iv) incorporation of PERCLOS (Percentage Eye Closure) as an alternative metric for extended real-world validation; and (v) edge deployment on Raspberry Pi 4 using the MediaPipe Lite model for cost-effective in-vehicle integration.

## ACKNOWLEDGMENT

The authors thank the faculty members and laboratory staff of the Department of CSE (Data Science), Raghu Engineering College, for their support throughout this project. This work was conducted as part of the BTech Final Year Project programme at Raghu Engineering College, Visakhapatnam. The open-source communities behind MediaPipe, OpenCV, and Streamlit are gratefully acknowledged.

## REFERENCES

[1] T. Soukupová and J. Čech, "Real-time eye blink detection using facial landmarks," in Proc. 21st Comput. Vis. Winter Workshop, 2016, pp. 1–8.

[2] M. Omidyeganeh, A. Shirmohammadi, S. Abtahi, and M. Khurshid, "Yawning detection using embedded smart cameras," IEEE Trans. Instrum. Meas., vol. 65, no. 3, pp. 570–582, Mar. 2016.

[3] E. Vural, M. Cetin, A. Ercil, G. Littlewort, M. Bartlett, and J. Movellan, "Drowsy driver detection through facial movement analysis," in Proc. IEEE ICCV Workshop HCI, 2007, pp. 1–6.

[4] V. Lugaresi et al., "MediaPipe: A framework for building perception pipelines," arXiv preprint arXiv:1906.08172, 2019.

[5] C. Dewi, R.-C. Chen, X. Liu, and Y.-T. Jiang, "Eye aspect ratio for real-time drowsiness detection to improve driver safety," Electronics, vol. 11, no. 19, p. 3183, Sep. 2022.

[6] C.-H. Weng, Y.-H. Lai, and S.-H. Lai, "Driver drowsiness detection via a hierarchical temporal deep belief network," in Proc. ACCV Workshops, 2016, pp. 117–133.

[7] S. Kaplan, M. A. Guvensan, A. G. Yavuz, and Y. Karalurt, "Driver behavior analysis for safe driving: A survey," IEEE Trans. Intell. Transp. Syst., vol. 16, no. 6, pp. 3017–3032, Dec. 2015.

[8] C. Lunscher and J. Zelek, "Towards online full lapse detection from eye gaze," in Proc. IEEE Int. Conf. Syst. Man Cybern., 2017, pp. 680–684.

[9] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, Oct. 2011.

[10] Google LLC, "MediaPipe Face Mesh," MediaPipe Solutions Guide, 2023. [Online]. Available: https://developers.google.com/mediapipe/solutions/vision/face_landmarker

[11] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd ed. Sebastopol, CA: O'Reilly Media, 2022.