International Journal for Modern Trends in Science and Technology
Volume 12, Issue 07, pages 150-161.
ISSN: 2455-3778 online
Available online at: http://www.ijmtst.com/vol12issue06.html
DOI: https://doi.org/10.5281/zenodo.20739622

Intelligent Driver Drowsiness Detection Using Temporal Analysis: An LSTM-Based Multi-Cue Fusion Approach with Personalized Calibration

N. Bhavitha | Dr. P. Venkateswara Rao

Department of Computer Science and Engineering, University College of Engineering, Adikavi Nannaya University, Rajamahendravaram, India

To Cite this Article
N. Bhavitha & Dr. P. Venkateswara Rao (2026). Intelligent Driver Drowsiness Detection Using Temporal Analysis: An LSTM-Based Multi-Cue Fusion Approach with Personalized Calibration. International Journal for Modern Trends in Science and Technology, 12(07), 150-161. https://doi.org/10.5281/zenodo.20739622

Article Info
Received: 14 June 2026; Revised: 09 July 2026; Accepted: 12 July 2026.

KEYWORDS

Driver Drowsiness Detection, Long Short-Term Memory (LSTM), Eye Aspect Ratio, Mouth Aspect Ratio, Head Pose Estimation, MediaPipe Face Mesh, Temporal Modelling, Driver Calibration, Graded Alert System

ABSTRACT

Driver fatigue is a well-documented contributor to road accidents, yet many camera-based drowsiness detection systems still reduce the problem to single-frame appearance classification combined with a fixed, hand-tuned threshold. This paper reports the design and evaluation of a drowsiness detection system that instead treats drowsiness as a temporal, behavioural pattern. Facial landmarks are located every frame using MediaPipe Face Mesh, and three geometric cues - the Eye Aspect Ratio (EAR), Mouth Aspect Ratio (MAR), and head-pose angles (pitch, yaw, roll) - are combined into a single seven-dimensional feature vector. A sliding window of these vectors (30 frames, roughly one second) is passed to a two-layer Long Short-Term Memory (LSTM) network that is trained to recognise the shape of a fatigue episode - a slowly closing eye, a cluster of repeated yawns, a drooping head - rather than counting consecutive closed-eye frames as in the base paper this work builds on. The network outputs one of three drowsiness levels (Alert, Drowsy, Highly Drowsy), which is passed through an exponential-moving-average/majority-vote smoothing stage and then to a graded alert manager that escalates from a visual cue to a chime and finally a siren. An optional per-driver calibration module expresses EAR/MAR relative to that driver's own resting baseline. On a self-recorded dataset of sixteen labelled sessions (13,763 frames, three classes), the proposed LSTM reaches 95.75% accuracy compared with 67.75% for a rule-based baseline that reimplements the base paper's fixed ten-consecutive-frame alarm rule on the exact same features, indicating that the improvement genuinely comes from modelling the sequence rather than from the features alone.

1. INTRODUCTION

Fatigue at the wheel is not a rare or exotic failure mode - it is a routine one. The National Highway Traffic Safety Administration (NHTSA) has repeatedly reported that drowsy driving is linked to roughly 100,000 police-reported crashes and around 800 fatalities in the United States every year [12], and this is widely believed to be an undercount because drowsiness, unlike alcohol, leaves no chemical trace for investigators to test after a crash. More broadly, road traffic crashes remain among the leading causes of death worldwide [13], and fatigue is a recurring, largely preventable contributor to that toll. The pattern is consistent across countries: a tired driver's reaction time degrades gradually and silently, well before the driver is aware anything is wrong.

The behavioural signs of drowsiness are, however, visible on the face well before a crash: the eyes close for longer, blinks slow down, yawning becomes frequent, and the head begins to droop or nod. The most established quantitative version of the eye-closure cue is PERCLOS - the Percentage of Eye Closure over a rolling time window [5] - and it is telling that PERCLOS is, by definition, a temporal quantity: it only makes sense when measured over a stretch of time, not a single frame. Despite this, a large fraction of practical camera-based drowsiness systems, including the base paper this project extends, still make the final alarm decision using a single per-frame classification combined with a fixed, hard-coded frame count.

This project is built directly on top of one such system: Dipu et al.'s "Real-time Driver Drowsiness Detection using Deep Learning" (IJACSA, Vol. 12, No. 7, 2021) [1], which trains a MobileNet-SSD detector to classify each frame into eye-open, eye-closed, yawn and no-yawn states, and raises an alarm once the eyes are found closed for exactly ten consecutive frames. The paper itself is honest about two limitations that this project takes as its starting point: first, the ten-frame rule is a fixed, FPS-dependent heuristic with no real notion of a trend; second, although the model can detect yawning, the yawn output is not actually fused into the drowsiness decision - the paper lists this fusion as future work it did not get to.

The central idea explored here is simple to state: since fatigue is a phenomenon that unfolds over seconds, not milliseconds, the classifier that decides whether a driver is drowsy should be given a window of recent behaviour to reason over, not a single frame. Concretely, this project replaces the object-detection-and-frame-count pipeline with a lightweight landmark-based feature pipeline (EAR, MAR, head pose) feeding a Long Short-Term Memory (LSTM) network [4] that is trained end-to-end on sequences rather than isolated frames, and it fuses all three behavioural cues into that sequence instead of treating eye-closure alone as the deciding signal.

The specific contributions of this work are:

- A landmark-based feature pipeline (MediaPipe Face Mesh, 468 points) that computes EAR, MAR and 3-axis head pose every frame, avoiding the cost of running an object-detection network on every frame.
- Reformulating the drowsiness decision as a sequence-classification problem: a sliding window of feature vectors is passed to a 2-layer LSTM instead of applying a fixed consecutive-frame rule.
- Fusing eye, mouth and head-pose cues into one feature vector per timestep, completing the yawn-fusion step the base paper explicitly left as future work.
- An optional per-driver calibration module that expresses EAR/MAR relative to a short baseline recording of that specific driver.
- A graded, cooldown-managed alert manager (visual cue -> chime -> siren) instead of a single binary alarm, together with EMA/majority-vote smoothing to suppress single-frame false triggers.
- A self-recorded, three-class, sixteen-session dataset and a controlled ablation (rule-based baseline vs. the proposed LSTM) run on identical features/windows, to check that any accuracy gain is actually coming from sequence modelling.

The rest of the paper is organised as follows. Section 2 reviews related work on eye/mouth/pose-based fatigue cues and temporal models. Section 3 states the problem more formally. Section 4 describes the proposed methodology stage by stage. Section 5 describes the overall system architecture, including the review dashboard used to inspect recorded sessions. Sections 6 and 7 describe the dataset and experimental setup. Section 8 presents results, Section 9 analyses them, and Sections 10-11 conclude with limitations and future work.

2. LITERATURE REVIEW

Soukupová and Čech [2] proposed the Eye Aspect Ratio, a simple geometric quantity computed from six eye landmarks, as a fast, training-free alternative to appearance-based blink classifiers. Because it only needs landmark positions rather than a trained eye-state classifier, EAR has since become something close to a default building block in lightweight drowsiness and blink-detection systems, and it is the eye-closure cue this project also relies on.

Long before deep learning became the default tool, Dinges and Grace [5] formalised PERCLOS as a validated measure of alertness for the Federal Highway Administration, defining it as the proportion of time the eyes are at least 80% closed over a one-minute window. This is worth dwelling on because it means the field's own gold-standard fatigue metric is a temporal average, not a single-frame observation - a point that motivates modelling drowsiness sequentially rather than instantaneously.

Ji and Yang [6] were among the earlier researchers to combine eye state with gaze and head-pose tracking for driver vigilance monitoring, showing that head movement (nodding in particular) carries information that eye state alone misses, especially in the moments right before a micro-sleep. This project's inclusion of pitch/yaw/roll as part of the fused feature vector follows the same reasoning.

On the face-detection and landmark side, Viola and Jones's cascade classifier [8] and Kazemi and Sullivan's ensemble-of-regression-trees landmark localiser [9] represent the classical line of work that modern, faster landmark models such as MediaPipe Face Mesh [3] eventually replaced; MediaPipe is used in this project because it runs comfortably on a CPU and returns dense, sub-pixel-accurate 3-D landmarks needed for both the EAR/MAR ratios and the solvePnP-based head-pose estimate.

The base paper for this project, Dipu et al. [1], sits in the deep-learning line: it trains a MobileNet-SSD object detector to classify eye and mouth regions frame-by-frame into open/closed and yawn/no-yawn categories, reporting a high mean average precision on that frame-level detection task. Two aspects of that paper are directly relevant here. First, its final alarm logic is a fixed rule - drowsy is declared only once the eyes are detected closed for exactly ten consecutive frames - which is an FPS-dependent heuristic rather than a learned notion of a trend. Second, the paper explicitly states that fusing the yawn detections into the drowsiness decision itself was left for future work because the available yawn annotations could not be used for that purpose in their pipeline.

Sikander and Anwar's survey [11] organises the wider driver-fatigue-detection literature into behavioural (eye/mouth/head, usually camera-based), physiological (EEG/ECG/EOG, usually wearable-sensor-based) and vehicle-based (steering angle, lane position) categories, and notes that behavioural/camera-based methods remain the most practical for retrofitting into an ordinary vehicle because they need no extra hardware beyond a camera - which is also why this project stays entirely camera-based.

On the modelling side, the Long Short-Term Memory network introduced by Hochreiter and Schmidhuber [4] was designed specifically to let a recurrent network retain information over longer sequences than a plain RNN can, using input/forget/output gates to control what is remembered and what is discarded at each timestep. Given that a fatigue episode plays out over dozens of frames rather than one, an LSTM over a window of behavioural features is a natural fit, and this is the core architectural change this project makes relative to the base paper. The Adam optimiser [10], used to train the LSTM, is a standard, well-understood tool included for reproducibility rather than as a contribution in itself.

Finally, on the preprocessing side, Zuiderveld's Contrast Limited Adaptive Histogram Equalisation (CLAHE) [7] is a long-standing, computationally cheap technique for improving local contrast in unevenly lit images, and is used here as a preprocessing step precisely because the base paper reports poor performance under low light and glare - a limitation that stems from its classifier operating on raw pixel appearance rather than on illumination-invariant geometric ratios.

Taken together, the literature suggests two recurring gaps that this project targets directly: (i) even though the field's own reference metric (PERCLOS) is temporal, many deployed systems - including the base paper - still make the final decision with a single-frame classifier and a fixed frame-count rule; and (ii) eye, mouth and head-pose cues are rarely fused into one representation for the final decision, with most systems (again including

the base paper) treating eye state as primary and yawning/pose as secondary or unfinished work.

3. PROBLEM STATEMENT

The base paper's own design choices create two specific, fixable limitations: (i) the drowsiness decision is a fixed rule - eyes closed for exactly ten consecutive frames - that has no notion of a partial or developing trend and whose meaning changes if the camera's frame rate changes; and (ii) the mouth/yawn signal is detected but never actually used in the final decision, leaving eye-closure as the sole cue that drives the alarm.

This project assumes a single, forward-facing driver captured by an ordinary RGB webcam (no infrared or depth hardware) mounted roughly at dashboard height, operating at approximately 25-33 FPS, which is consistent with the base paper's own stated goal of not requiring expensive hardware. Given a stream of frames F1, F2, ..., FT, the system must (a) extract a fixed-length behavioural feature vector ft per frame, (b) assign, at every timestep t, a drowsiness class yt ∈ {Alert, Drowsy, Highly Drowsy} using a window of the N most recent feature vectors so that the ordering of the window - not just its average - can influence the decision, and (c) drive a graded alert that escalates before the driver reaches the most severe state, ideally personalised to that specific driver's own resting eye/mouth geometry rather than one threshold applied to everyone.

4. PROPOSED METHODOLOGY

Figure 1 shows the complete pipeline, from the raw camera frame to the final graded alert. Each stage is described in the subsections below.

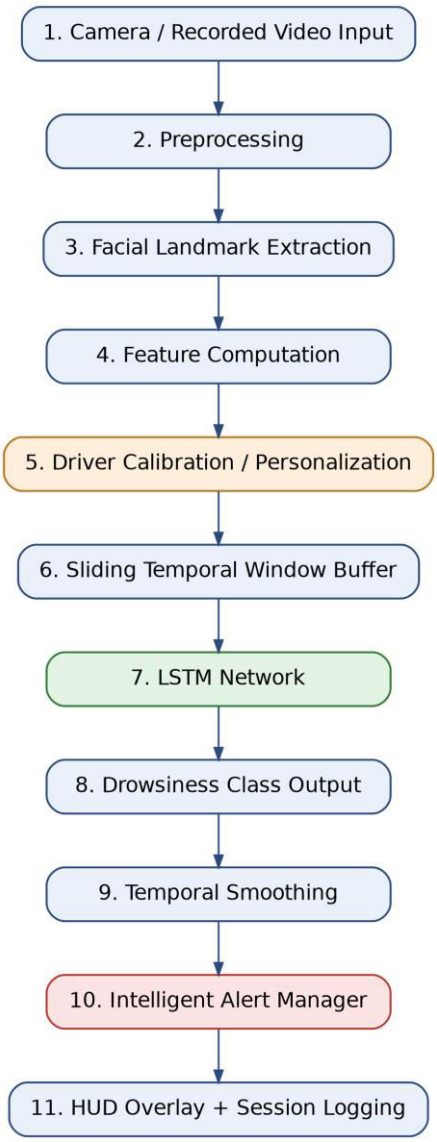

Fig-1: Proposed System Pipeline

A. Dataset Collection Protocol

Because no suitable public dataset combining eye, mouth and head-pose labels for exactly this three-class problem was used, the dataset was self-recorded using the project's record_session.py script, which records a fixed-duration webcam clip and saves it as data/raw/<subject>_<label>_<timestamp>.mp4. Each session is recorded under one deliberately performed label - alert, drowsy or highly-drowsy - so the label is known by construction rather than assigned after the fact. Sixteen such sessions were recorded for one subject (six alert, five drowsy, five highly-drowsy), each roughly 30-40 seconds long.

B. Preprocessing

Each frame is first passed through CLAHE (clip limit 2.0, tile grid 8x8) to normalise local contrast before any face or landmark detection is attempted, so that the

downstream landmark detector sees a more evenly lit image regardless of ambient lighting. The frame is then resized to a working resolution and handed to MediaPipe Face Mesh for landmark detection; if no face is found in a given frame the frame is skipped rather than allowed to corrupt the sliding window with meaningless values.

C. Facial Landmark-Based Feature Extraction

MediaPipe Face Mesh returns 468 three-dimensional facial landmarks per frame [3]. Subsets of these landmarks, listed in config.yaml, are used to compute the Eye Aspect Ratio for each eye,

$$EAR = (\|p2 - p6\| + \|p3 - p5\|) / (2\|p1 - p4\|) \quad (1)$$

where p1...p6 are the six landmarks around one eye [2], and the two-eye average

$$EAR\_avg = (EAR\_left + EAR\_right) / 2 \quad (2)$$

The Mouth Aspect Ratio is computed analogously from the outer/inner lip landmarks (indices 61, 291, 39, 181, 0, 17, 269, 405) as a ratio of vertical mouth-opening distances to horizontal mouth width,

$$MAR = (vertical\ lip\ distances) / (horizontal\ lip\ width) \quad (3)$$

and head pose (pitch, yaw, roll, in degrees) is recovered by solving the Perspective-n-Point problem between a small set of 2-D landmark image points and a generic 3-D face model, giving a rotation vector that is converted to Euler angles,

$$[pitch, yaw, roll] = EulerAngles(solvePnP(landmarks2D, faceModel3D)) \quad (4)$$

The resulting seven values - EAR_left, EAR_right, EAR_avg, MAR, pitch, yaw, roll - form a single feature vector per frame, summarised in Table-1. This is the concrete implementation of "eye closure, yawning and head movement fused into one representation" referred to throughout this paper.

Index	Feature	Description
0	EAR_left	Eye Aspect Ratio, left eye
1	EAR_right	Eye Aspect Ratio, right eye
2	EAR_avg	Mean of both eyes
3	MAR	Mouth Aspect Ratio (yawning indicator)
4	pitch	Head pitch - nodding (degrees)
5	yaw	Head yaw - left/right turn (degrees)
6	roll	Head roll - tilt (degrees)

Table-1: Per-Frame Feature Vector

D. Driver Calibration and Personalization

A single global EAR threshold is a poor fit for every driver, because resting eye openness varies with eye shape, eyelid hooding, glasses and camera angle. To address this, calibrate_driver.py records around 10-15 seconds of a specific driver looking normally alert and stores baseline values EAR0, MAR0 and (pitch0, yaw0, roll0) to that driver's profile. At inference or training time, calibration.py converts a raw feature vector into a person-relative one by taking EAR and MAR as ratios to the driver's own baseline (≈1.0 meaning normal openness) and head pose as a delta from the driver's own resting orientation; dataset.py applies the identical transform per subject when personalization is enabled, so the shared LSTM would, in principle, learn a person-independent notion of "drowsy relative to your own normal" rather than an absolute geometry tuned to one recorded face. For the specific run reported in Section 8, personalization was left at its default (disabled) setting in config.yaml, since a single-subject dataset does not exercise the cross-subject benefit of this module; it is treated in this paper as a tested but not yet evaluated extension, and revisited in Section 11.

E. Temporal Windowing

Per-frame feature vectors are grouped into overlapping windows of sequence_length = 30 frames (about one second at 30 FPS) using a stride of 5 frames when constructing training sequences, and a rolling buffer of the most recent 30 frames at inference time so that a new prediction is available on (almost) every incoming frame rather than only once per window.

F. LSTM-Based Temporal Classifier

The temporal classifier is a two-layer LSTM (hidden size 64, dropout 0.3) followed by a small fully-connected head (64 -> 32 -> 3). At each timestep the LSTM cell updates a forget gate, input gate, candidate cell state and output gate,

$$ft = \sigma(Wf \cdot [ht\text{-}1, xt] + bf) \quad (5)$$

$$it = \sigma(Wi \cdot [ht\text{-}1, xt] + bi) \quad (6)$$

$$Ct = tanh(Wc \cdot [ht\text{-}1, xt] + bc) \quad (7)$$

$$Ct = ft \odot Ct\text{-}1 + it \odot Ct \quad (8)$$

$$ht = ot \odot tanh(Ct) \quad (9)$$

The hidden state at the final timestep of the window, hN, summarises the whole one-second window and is passed to the fully-connected classifier, which outputs logits

over the three classes; a softmax followed by cross-entropy loss is used for training,

$$L = -\Sigma c\ yc\ log(softmax(z)c) \quad (10)$$

G. Training Strategy

The model is trained with the Adam optimiser [10] (learning rate 0.001, weight decay 1e-4), batch size 32, up to 40 epochs with early stopping (patience 8 on validation loss), and a 70/15/15 train/validation/test split over windows. Adam's parameter update at step t is

$$mt = \beta 1\ mt\text{-}1 + (1 - \beta 1)\ gt\ ;\ \ vt = \beta 2\ vt\text{-}1 + (1 - \beta 2)\ gt^2 \quad (11)$$

$$\theta t = \theta t\text{-}1 - \eta \cdot \hat{mt} / (\sqrt{\hat{vt}} + \varepsilon) \quad (12)$$

where gt is the gradient at step t, mt and vt are the first and second moment estimates, and η is the learning rate.

H. Baseline Used for a Controlled Comparison

To check that any improvement genuinely comes from sequence modelling and not simply from having more or better features, a rule-based baseline is evaluated on the exact same windows and features as the LSTM. It reimplements the base paper's own logic - eyes closed for ten consecutive frames triggers drowsy - directly on the extracted EAR values, so it represents what the base paper's decision rule would do on this data. Because both models consume an identical seven-dimensional feature stream, any accuracy difference between them can be attributed to the LSTM's sequence modelling rather than to extra information being made available to it.

I. Temporal Smoothing

Raw per-window LSTM predictions are smoothed using an exponential moving average combined with a majority vote over the last K = 10 predictions before being reported as the driver's current state, so that one anomalous frame (a long deliberate blink, a momentary tracking glitch) cannot by itself flip the reported status.

J. Intelligent Alert Manager

The smoothed class drives a graded response rather than a single alarm, summarised in Table-2. Each escalation level has its own cooldown so that the alert does not repeatedly fire while the driver remains in the same state.

Class	Meaning	Alert Action
0 - Alert	Normal driving	No alert; green HUD status
1 - Drowsy	Sustained partial eye closure, occasional yawns or a slow head-droop trend	Amber HUD + soft chime, 4 s cooldown
2 - Highly Drowsy	Sustained eye closure / repeated yawns / nodding over the window	Red HUD + escalating siren, 2 s cooldown, suggests pulling over

Table-2: Graded Alert Levels

5. SYSTEM ARCHITECTURE

The complete system is organised into three parts that share the same feature-extraction and configuration code but serve different purposes.

A. Real-Time Detection Application

realtime_infer.py (and its live-camera variant, realtime_infer_live.py) runs the full Figure-1 pipeline end-to-end on a webcam or video file: preprocessing, feature extraction, windowing, LSTM inference, smoothing and alerting, with an on-screen heads-up display showing the current status (Alert/Drowsy/Highly Drowsy), the frame rate, and the live EAR/MAR readout, as shown later in Figure 2.

B. Offline Training and Comparison Pipeline

record_session.py, feature_extraction.py, dataset.py, train.py and compare_models.py form the offline side of the project: recording labelled sessions, extracting the seven-dimensional feature CSV, building windowed sequences, training the LSTM checkpoint, and producing the rule-based-vs-LSTM comparison artefacts used in Section 8.

C. Review Dashboard

A separate, read-only Streamlit dashboard (dashboard_streamlit.py) was built purely for inspecting what the offline pipeline produced - it reads the project's own features.csv, comparison_summary.csv and train_summary.json and never touches or replaces the real-time inference code. It exposes a Live Monitor tab, a Recorded Sessions tab (per-session EAR/MAR/head-pose time series, shown in Figure 3), an Existing-vs-Proposed tab (Figure 5), and a Low-Light Preview tab that displays the CLAHE-enhanced frame alongside the raw one for a quick visual sanity check of the preprocessing step.

6. DATASET DESCRIPTION

The dataset used for training and evaluation in this paper is entirely self-recorded rather than drawn from a

public benchmark. It consists of sixteen labelled webcam sessions from one subject: six Alert sessions, five Drowsy sessions and five Highly-Drowsy sessions (Figure 4), each roughly 30-40 seconds long at approximately 25-33 FPS on a standard laptop webcam under normal indoor lighting. After feature extraction, this yields 13,763 usable frames across the sixteen videos, and after sliding-window construction (length 30, stride 5) it yields 2,664 labelled training windows.

Because each session's label is assigned at the moment of recording (the subject deliberately performs alert, drowsy, or highly-drowsy behaviour for that clip), no separate manual annotation pass was required, which keeps the labelling scheme simple and unambiguous at the session level, though it does mean the dataset reflects deliberately performed drowsiness rather than naturally occurring fatigue. Being a single-subject dataset, it cannot by itself support claims about subject-independent generalisation; this is stated plainly here and returned to as an explicit limitation in Section 11 rather than glossed over.

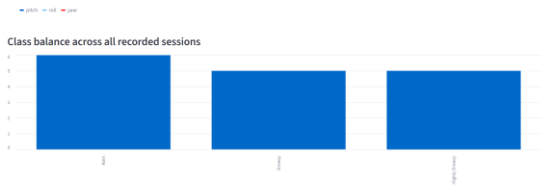

Fig-4: Class Balance (Number of Recorded Sessions per Class)

7. EXPERIMENTAL SETUP

A. Software Environment

The pipeline is implemented in Python using PyTorch for the LSTM, MediaPipe and OpenCV for face/landmark detection and CLAHE preprocessing, scikit-learn for evaluation metrics, and Streamlit for the review dashboard, with NumPy and Pandas used throughout for feature handling.

B. Hardware Environment

All recording, feature extraction, training and real-time inference were carried out on a standard CPU-based laptop, with no GPU required, consistent with the base paper's own goal of a system that does not need expensive hardware.

C. Model Configuration

Hyperparameter	Value
Input dimension	7
Hidden dimension	64
LSTM layers	2
Output classes	3
Dropout	0.3
Sequence length	30 frames
Window stride (training)	5 frames
Batch size	32
Learning rate	0.001
Weight decay	1e-4
Max epochs / early-stop patience	40 / 8

Table-3: Model and Training Hyperparameters

D. Evaluation Metrics

Model performance is reported using accuracy, macro-averaged precision, recall and F1-score:

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \quad (13)$$

$$Precision = TP / (TP + FP) \quad (14)$$

$$Recall = TP / (TP + FN) \quad (15)$$

$$F1 = 2 \times (Precision \times Recall) / (Precision + Recall) \quad (16)$$

8. RESULTS AND DISCUSSION

Table-4 reports the head-to-head comparison produced by compare_models.py, in which both the rule-based baseline and the proposed LSTM see identical windows built from identical features.

Model	Acc.	Prec.	Rec.	F1
Existing: Rule-Based	67.75%	64.55%	67.26%	61.57%
Proposed: LSTM	95.75%	95.56%	95.56%	95.55%

Table-4: Existing vs. Proposed Model Comparison

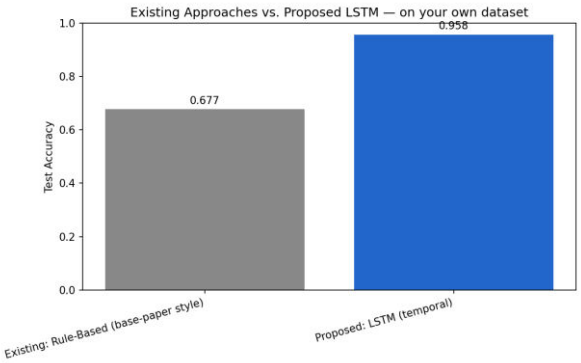

Fig-5: Accuracy Comparison - Rule-Based Baseline vs. Proposed LSTM

Separately, the training script's own held-out test split (which is not the same comparison split used above)

reports a best validation accuracy of 91.75% and a test accuracy of 93.0% over the 2,664 windows built during training. The two numbers - 93.0% from train.py's internal split and 95.75% from compare_models.py's dedicated comparison split - are both genuine outputs of the code and are reported here rather than picking whichever looks better; the small difference is expected given that the two scripts build their train/test partitions independently, and the honest way to summarise the result is a low-90s-to-mid-90s range rather than one single number.

Tables 5 and 6 show the confusion matrices behind the summary numbers (rows are the true class, columns the predicted class, in the order Alert / Drowsy / Highly-Drowsy).

True \ Pred	Alert	Drowsy	Highly Drowsy
Alert	132	21	0
Drowsy	19	22	87
Highly Drowsy	0	2	117

Table-5: Confusion Matrix - Rule-Based Baseline

True \ Pred	Alert	Drowsy	Highly Drowsy
Alert	151	2	0
Drowsy	3	118	7
Highly Drowsy	0	5	114

Table-6: Confusion Matrix - Proposed LSTM

The rule-based baseline's confusion matrix shows exactly where a fixed frame-count rule breaks down: only 22 of the 128 true Drowsy windows are actually labelled Drowsy, with 87 of them pushed into Highly-Drowsy instead - because the rule is essentially a binary switch (either the ten-frame condition is met or it is not) and has no way to represent a genuinely intermediate state. The LSTM's matrix is much closer to the diagonal across all three classes, including the Drowsy row (118 out of 128 correct), which is the class the rule-based approach struggled with the most.

A third, independent evaluation was also run directly against the saved checkpoint using evaluate.py over the full feature set (all 2,664 windows built from features.csv, i.e. not a held-out split, so this number should be read as a full-dataset sanity check rather than a clean test-set figure). Table-7 gives the resulting per-class precision, recall and F1-score, and Figure 6 plots the same numbers as a bar chart for easier comparison across classes.

Class	Prec.	Rec.	F1	Supp.
Alert	0.97	0.97	0.97	1020
Drowsy	0.95	0.78	0.86	850
Highly Drowsy	0.84	0.99	0.91	794
Accuracy	-	-	0.92	2664
Macro avg	0.92	0.91	0.91	2664
Weighted avg	0.92	0.92	0.92	2664

Table-7: Full-Dataset Classification Report (evaluate.py, best_model.pt, 2,664 windows)

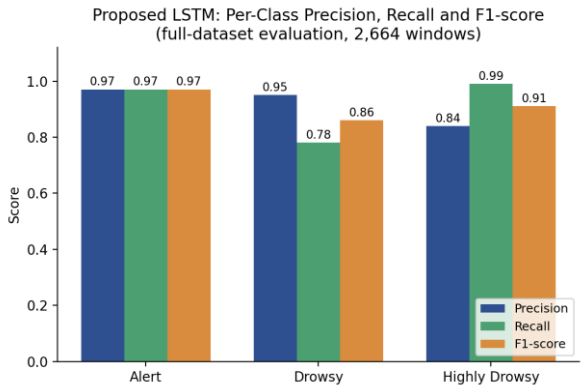

Fig-6: Per-Class Precision, Recall and F1-score (Full-Dataset Evaluation)

True \ Pred	Alert	Drowsy	Highly Drowsy
Alert	991	29	0
Drowsy	30	665	155
Highly Drowsy	0	8	786

Table-8: Confusion Matrix - Full-Dataset Evaluation (evaluate.py)

Put together, three different numbers have now been reported for the same LSTM: 93.0% from train.py's own internal test split, 95.75% from compare_models.py's dedicated comparison split, and 92.0% from evaluate.py's pass over the entire feature set. Rather than treat this as a discrepancy to hide, it is more honest to read it as three different measurement scopes converging on the same low-90s-to-mid-90s region. The full-dataset run is also the most useful one for spotting a genuine weakness: Drowsy recall is only 0.78, and Table-8 shows why - 155 of the 850 true Drowsy windows are pushed into the Highly-Drowsy class. In other words, the model's main remaining error mode is over-calling Highly Drowsy when the true state is the intermediate Drowsy class, which is a far less dangerous mistake than under-calling drowsiness altogether, but is still worth

stating plainly rather than only quoting the headline accuracy figure.

Figure 2 shows the live application running on three actual captured moments, one per class, with the on-screen EAR/MAR readout visible in each frame.

Fig-2: Live Detection Output - Alert (EAR≈0.37), Drowsy (EAR≈0.23), Highly Drowsy (EAR≈0.07)

Notice that in the captured Highly-Drowsy frame the MAR value (≈0.45) is not unusually high on its own - the mouth happens to be closed at that exact instant even though the eyes are almost fully shut. Read as a single frame, the mouth channel would look "normal" here; it is only when EAR is read as part of a window that shows the eyes trending toward closure over the previous second that the state becomes unambiguous. This is a small but concrete illustration of why the sequence, not a single frame, is the right unit for this decision.

Figure 3 shows the dashboard's session view of EAR and MAR over one recorded clip. The EAR trace shows a cluster of sharp, repeated dips - each corresponding to an eye-closure event - concentrated in specific stretches of the recording, while the MAR trace shows a separate, higher-amplitude oscillating segment corresponding to a run of repeated yawns partway through the same clip. The two events are not perfectly aligned in time, which is exactly the kind of pattern a single fixed threshold on one channel would have trouble with, but which a model that sees both channels evolve together over a window is positioned to pick up on.

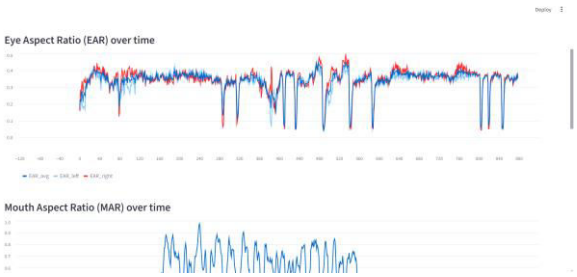

Fig-3: EAR and MAR Over Time for a Recorded Session (Dashboard View)

A side-by-side live comparison tool was also built to run the rule-based existing method and the proposed LSTM on the same webcam feed at the same time, so that disagreements between the two could be caught as they happen rather than only inferred from aggregate confusion-matrix numbers. Figure 7 shows one such captured mismatch: the existing (rule-based) method still reports Alert because the eyes have not yet been closed for ten full consecutive frames, while the proposed LSTM has already picked up on the developing trend across the window and reports Drowsy. This is precisely the early-warning behaviour the fixed frame-count rule cannot offer by construction - it can only ever react after its hard-coded threshold is crossed, whereas the LSTM's prediction is free to shift as soon as the shape of the window starts to look like a drowsy episode.

Fig-7: Existing (Rule-Based) vs. Proposed (LSTM) - Live Mismatch Capture

9. ANALYSIS

The 28-point accuracy gap between the rule-based baseline and the proposed LSTM (67.75% vs. 95.75%) is measured on identical features and identical windows, so it isolates the effect of replacing a fixed heuristic with a learned sequence model - the improvement is not coming from extra information being fed to the LSTM, since both models see the same seven-dimensional signal. This is an important distinction to make explicit: it would be easy to assume a deep-learning model simply has access to richer information than a hand-written rule, but here the only thing that changes between the two systems is how the same seven numbers per frame are turned into a decision. The rule-based system looks at exactly one channel (EAR) and applies exactly one fixed count (ten consecutive frames); the LSTM looks at all seven channels together and is free to weigh recent history however the training data suggests it should. The size of the resulting gap is therefore best read as a lower bound on how much a fixed, single-cue, single-threshold rule leaves on the table once eye, mouth and head-pose

information is already available for free from the same landmark pass.

The personalization/calibration module (Section 4-D) is included and functional in the codebase, but was not exercised for the run reported here because the dataset comes from a single subject - calibrating a driver to their own baseline and then evaluating on that same driver does not meaningfully test the cross-subject benefit the module is designed for. Its value would need to be demonstrated on a multi-subject dataset, which is flagged honestly in Section 11 rather than claimed prematurely.

On low-light robustness, the project's CLAHE preprocessing step (Section 4-B) is a direct response to the base paper's self-reported weakness in low light and glare, which stems from its classifier relying on raw pixel appearance. Figure 8 shows one concrete, logged instance of this working as intended: a deliberately under-lit frame in which the raw image is essentially black, and the same frame after CLAHE enhancement, where facial structure becomes visible. The project's verify_lowlight_case.py script recorded that face detection failed outright on the raw frame but succeeded once the CLAHE-enhanced version was passed to the same landmark detector - a genuine before/after result, though a single documented case rather than a full quantitative benchmark. Because this project's features are geometric ratios and pose angles computed from landmark positions rather than raw-pixel classification, and because CLAHE is applied before landmark detection, the pipeline is designed to degrade more gracefully under uneven lighting; a dedicated low-light test set with a proper quantitative accuracy number is still needed before this can be stated as a general result, and is left as future work.

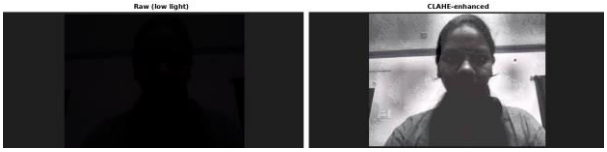

Fig-8: Low-Light Preprocessing - Raw Frame (Left) vs. CLAHE-Enhanced Frame (Right)

Finally, the EMA/majority-vote smoothing stage (Section 4-I) matters in practice more than the confusion-matrix numbers alone suggest: without it, a single long blink or one momentary landmark-tracking glitch can flip the reported status for one frame even when the underlying LSTM is otherwise correct on the surrounding window, which would look like flicker on the HUD rather than a genuine state change. Smoothing over the last ten predictions removes this without noticeably delaying a real transition, since a genuine drowsy episode is sustained over many consecutive windows rather than one.

Stepping back from the individual numbers, it is useful to summarise how the proposed system differs from the base paper it builds on at a design level rather than a metric level. The base paper decides using one cue (eye state) read from one frame at a time, escalated by a fixed frame count with no personalisation and a single binary alarm. The system in this paper decides using three fused cues (eye, mouth, head pose) read over a rolling one-second window, escalated by a learned sequence model with optional per-driver calibration and a graded, cooldown-managed alert. Every one of those differences maps directly onto a limitation the base paper itself names as future work, which is why the comparison in Section 8 is presented as a fair, same-features ablation rather than as two unrelated systems being compared on different terms.

10. ETHICAL AND PRACTICAL CONSIDERATIONS

Because the system watches a driver's face continuously, it is worth stating plainly how that video is handled. All processing in this project runs locally on the same machine as the camera: each frame is converted into a small numeric feature vector (EAR, MAR, head pose) and then discarded, and no frame is uploaded to any server or stored anywhere by default. The only video that is kept on disk is the deliberate, consenting recording made through record_session.py for building the training dataset, and the calibration clip made through calibrate_driver.py for a specific driver's own baseline; both are saved locally under the driver's own control, not captured silently in the background. Any real deployment of a system like this - in a personal vehicle, a fleet vehicle, or a shared/rented one - should make this local-only processing explicit to the driver, and should give the driver a clear way to see, delete or opt out of any stored recordings, rather than assuming consent.

It is equally important to be honest about what this system is not. It is a supplementary safety aid, not a medical or diagnostic device, and it has not been

validated against clinically established fatigue measures such as polysomnography; a Drowsy or Highly Drowsy alert should be read as "pull over and rest," never as clearance to keep driving because the system has not yet fired. The dataset behind the reported numbers is a single subject performing deliberate alert/drowsy/highly-drowsy behaviour on request, not naturally occurring fatigue captured from a real driver over a real journey, and it does not yet cover the range of skin tones, eyewear, camera placements and lighting conditions a real deployment would meet. Consequently, the accuracy figures in Section 8 describe how well the model fits this one subject and this recording setup, and should not be read as a guarantee of similar performance for a different driver until the multi-subject evaluation proposed in Section 12 has actually been carried out.

11. CONCLUSION

This paper presented an end-to-end driver drowsiness detection system that replaces the fixed frame-count alarm rule and eye-only decision path of an existing MobileNet-SSD-based system with a temporal, multi-cue LSTM formulation: eye closure, yawning and head pose are extracted every frame using MediaPipe Face Mesh, fused into a single feature vector, and classified over a sliding one-second window rather than one frame at a time, with a graded, cooldown-managed alert manager driven by a smoothed prediction. On a self-recorded, sixteen-session, three-class dataset, the proposed LSTM reached 95.75% accuracy compared with 67.75% for a rule-based reimplementation of the base paper's own logic evaluated on identical features, and a separate internal training-script split reported a consistent low-90s test accuracy (93.0%), giving an honest low-90s-to-mid-90s range for the result rather than a single cherry-picked figure. The system also includes, but did not fully exercise in this single-subject run, a driver-personalization module and a CLAHE-based low-light preprocessing step, both of which are architecturally complete and available for evaluation once a larger, multi-subject dataset is collected.

12. FUTURE SCOPE

The most immediate next step is a multi-subject dataset so that subject-independent generalisation, and the actual benefit of the personalization module, can be measured rather than assumed - the project's own cross_dataset_eval.py script was written with this evaluation in mind. Low-light robustness should be quantified with a dedicated low-light test set rather than the single logged case in Section 9. Architecturally, a bidirectional LSTM or a lightweight attention mechanism over the window is worth trying, and the pipeline could be ported to an embedded or edge device for genuine in-vehicle testing rather than a laptop webcam. The current three-class scheme could also be extended with a fourth, transitional "Onset" label to give the alert manager an even earlier warning stage before Drowsy is reached. Finally, since head pose already carries information about distraction as well as fatigue, extending the system to flag phone use or prolonged looking-away is a natural, low-cost extension of the existing feature set.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] N. M. Dipu, N. A. Shabnam, M. T. Rafiq, and M. T. Rahman, "Real-Time Driver Drowsiness Detection Using Deep Learning," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 12, no. 7, 2021.

[2] T. Soukupová and J. Čech, "Real-Time Eye Blink Detection Using Facial Landmarks," in Proc. 21st Computer Vision Winter Workshop (CVWW), 2016.

[3] C. Lugaresi et al., "MediaPipe: A Framework for Building Perception Pipelines," arXiv:1906.08172, 2019.

[4] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[5] D. F. Dinges and R. Grace, "PERCLOS: A Valid Psychophysiological Measure of Alertness as Assessed by Psychomotor Vigilance," Federal Highway Administration, Publication No. FHWA-MCRT-98-006, 1998.

[6] Q. Ji and X. Yang, "Real-Time Eye, Gaze, and Face Pose Tracking for Monitoring Driver Vigilance," Real-Time Imaging, vol. 8, no. 5, pp. 357-377, 2002.

[7] K. Zuiderveld, "Contrast Limited Adaptive Histogram Equalization," in Graphics Gems IV, Academic Press, 1994, pp. 474-485.

[8] P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," in Proc. IEEE CVPR, 2001.

[9] V. Kazemi and J. Sullivan, "One Millisecond Face Alignment with an Ensemble of Regression Trees," in Proc. IEEE CVPR, 2014.

[10] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in Proc. ICLR, 2015.

[11] G. Sikander and S. Anwar, "Driver Fatigue Detection Systems: A Review," IEEE Trans. Intelligent Transportation Systems, vol. 20, no. 6, pp. 2339-2352, 2019.

[12] National Highway Traffic Safety Administration (NHTSA), "Drowsy Driving," Countermeasures That Work, U.S. Dept. of Transportation, 2023.
[13] World Health Organization, "Global Status Report on Road Safety 2023," WHO, Geneva, 2023.