Indonesian Journal of Electronics, Electromedical Engineering, and Medical Informatics e-ISSN: 2656-8624
Homepage: https://ijeeemi.org/; Vol. 8, No. 3, pp. 441-458, August 2026

RESEARCH PAPER

OPEN ACCESS

Real-Time Drowsiness Detection Using Dual MobileNetV2 Models on Desktop and Edge Devices

Rafi'e[1], Ichsan Rizany[2], Dodon Turianto Nugrahadi[1], Dwi Kartini[1], Muhammad Itqan Mazdadi[1] and Triando Hamonangan Saragih[1]

[1] **Department of Computer Science, Lambung Mangkurat University, Banjarbaru, Indonesia**
[2] **Department of Nursing, Lambung Mangkurat University, Banjarbaru, Indonesia**

Corresponding author: Rafi'e. (e-mail: rafiekom@ulm.ac.id), **Author(s) Email**: Ichsan Rizany (e-mail: ichsan.r.psik@ulm.ac.id), Dodon Turianto Nugrahadi (e-mail: dodonturianto@ulm.ac.id), Dwi Kartini (e-mail: atdwikartini@ulm.ac.id), Muhammad Itqan Mazdadi (e-mail: mazdadi@ulm.ac.id), Triando Hamonangan Saragih (e-mail: triando.saragih@ulm.ac.id)

Abstract **Drowsiness is a leading cause of human error in transportation and shift-based occupational work, yet reliable real-time detection on affordable, resource-constrained hardware remains difficult. This study develops and evaluates a vision-based drowsiness detection system that behaves consistently across a full-power desktop and a low-cost edge device, coupling two independent MobileNetV2 classifiers, one for eye state (Open/Closed) and one for mouth state (Yawn/No_yawn), with a temporal decision engine converting frame-level predictions into microsleep and excessive yawning alerts. Both classifiers were trained on a merged multi-source dataset (8,548 images) with data augmentation and evaluated using a class-balanced protocol (186 images/class for eye, 448 for mouth) repeated 1000 times via Monte Carlo undersampling to remove test set imbalance and sampling bias. The decision engine runs as two platform specific pipelines sharing an offline first, retry-capable event architecture and an identical duration-based, two-tier hysteresis rule for eye closure: a desktop application (Haar cascade, H5/float32 models) and a Raspberry Pi 5 edge board (MediaPipe, quantised TensorFlow Lite models). On the class-balanced test set, the eye branch reached 94.09% (H5) / 94.62% (TFLite) accuracy, and the mouth branch reached 95.20% (H5) / 95.76% (TFLite) accuracy, with ROC AUC ≥ 0.985 for both branches and over 97% cross format prediction agreement. Converting to TFLite cut model size by 73.4% (8.99 to 2.39 MB) and single-frame inference latency roughly 13-fold (about 26 to 2 ms on the desktop). In an on-device, real-time evaluation across five participants and seven capture conditions with frame-level ground truth, the system sustained about 38.8 FPS at a near 100% face detection rate, with event detection accuracy degrading gracefully from 91.7% under normal conditions to 79.2% under night time low light.**

Keywords: Drowsiness detection; MobileNetV2; Transfer learning; Real time monitoring; Edge computing

I. Introduction

Drowsiness-related impairment expressed physically through prolonged eye closure and yawning is a well-documented precursor to lapses in attention and slower reaction time, both while driving and during sustained occupational tasks such as assembly line and shift-based work [1]. According to a recent review of PERCLOS-based technologies, the percentage of eyelid closure over time remains the most reliable behavioural indicator of drowsiness among the available physiological and observational measures [1]. Drowsiness monitoring has since moved beyond the driving context: systems that combine camera-based monitoring with wearable physiological and motion sensing have been proposed for assembly line and manufacturing workers [2], and an adaptive computer vision framework aligned with expert ergonomic assessment has been formalised for smart manufacturing fatigue monitoring [3]. Both are consistent with epidemiological evidence that fatigue and reduced alertness during shift work measurably raise occupational accident and injury risk [4]. A recent systematic review groups existing drowsiness detection techniques into behavioural (visual), vehicular, and physiological approaches, and notes that behavioural cues such as eye closure and yawning remain the most practical to capture with an ordinary camera [5].

Deep convolutional neural networks (CNNs) are now the dominant way to extract these cues directly from camera images without hand-crafted feature engineering. Several studies report CNN-based eye state and yawning classifiers with accuracies of roughly 90 99% on benchmark face datasets [6], [7], [8]. To meet the computational limits of real-time inference, many of these systems adopt lightweight backbones through transfer learning [9] rather than training deep networks from scratch. MobileNetV2, with its inverted residual and linear bottleneck design, was proposed specifically to give mobile and embedded applications a favourable accuracy to computation trade off [10], building on the depthwise separable convolution design that underlies lightweight mobile vision backbones more broadly [11]. It belongs to a broader family of efficiency-oriented mobile backbones

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

that includes reparameterised, single-branch designs such as MobileOne [12], modernised pure convolutional designs such as ConvNeXt [13], and hybrid CNN-transformer designs such as FastViT [14] direct comparisons on constrained medical imaging tasks have shown that lighter architectures do not necessarily trade accuracy for efficiency [15].

Beyond drowsiness detection, MobileNetV2 has proven effective across other resource-constrained vision tasks fruit [16], dermatological [17], [18], [19], and road surface [20] image classification, and, most relevantly here, on-device face recognition on a Raspberry Pi [21] and facial image classification in this journal [22], [23]. These confirm MobileNetV2 as a reasonable backbone for a continuously running, camera-based monitor. Classifying the eye and mouth regions first requires a face detector: the Viola Jones Haar cascade remains popular for its low CPU cost but was designed for near frontal, well lit faces and degrades under head pose variation [24], whereas MediaPipe's BlazeFace is a lightweight anchor based detector; comparable lightweight face detectors have been shown to retain robustness to pose and scale variation while running efficiently on ARM class embedded hardware such as the Raspberry Pi [25] important when a camera cannot be guaranteed to sit at eye level. Rather than the geometric eye aspect ratio approach [26], this study uses purely appearance-based CNN classification for both cues, avoiding dependence on landmark localisation under pose and lighting variation.

Prior drowsiness systems on the Raspberry Pi [27], [28], [29] demonstrate low cost, portable operation but describe a single hardware tier; none examines how one detection design must be re-expressed when deployed simultaneously on a full-power desktop and a resource constrained single board computer, where the face detector, model format, and alert timing primitive each differ. Separately, queue and retry architectures that buffer readings locally and retry once connectivity returns make IoT edge-to-cloud pipelines resilient to unreliable networks [30], [31] and the tiny machine learning literature surveys on device model compression [32]. To our knowledge, this buffer and retry pattern has not been reported for a vision-based drowsiness alert pipeline that a must also transmit annotated image evidence rather than periodic scalar telemetry the two gaps this work targets.To address these gaps, this study aims to develop and evaluate a drowsiness detection system built on two independent MobileNetV2 classifiers (eye state and mouth state) and to deploy it as two production pipelines with platform-appropriate face detectors and temporal decision logic. The contributions of this study are as follows:

1) Methodological contribution: a cross-tier temporal decision mapping that expresses the same eye/mouth alerting logic with platform-appropriate primitives wallclock duration hysteresis on both the desktop and the Raspberry Pi 5 edge tier, identical parameters θ1 = 15 s, θ2 = 30 s on both tiers) rather than the FPS fragile frame count heuristic used on the edge tier in earlier iterations of this system.
2) Engineering contribution: an offline first, queue and retry event architecture that carries annotated image evidence (not periodic scalar telemetry) end to end on both tiers, so alerts are not silently lost when network connectivity drops.
3) Evaluation contribution: a class balanced, Monte Carlo repeated (1000 trial) evaluation of transfer learning classifiers trained on a merged, multi source eye/mouth dataset with training time data augmentation (rotation, shift, shear, zoom, brightness, horizontal flip, Gaussian blur) that reports mean ± 95% confidence intervals for accuracy, balanced accuracy, ROC AUC, and PR AUC, removing both test set imbalance bias and single draw sampling bias.
4) Validation contribution: a multi-condition real-time evaluation protocol that extends beyond a static held-out test set to on device testing under varying illumination, head pose, distance, and eyewear conditions, distinguishing this work from prior single-cue, single tier Raspberry Pi, and scalar telemetry offline-first systems that validate on only one platform.

The novelty is not any single component MobileNetV2, Haar cascade and BlazeFace detection, TensorFlow Lite quantisation, and queue and retry delivery each pre exist but the validated mapping of one drowsiness detection design onto two heterogeneous tiers with identical, reproducible decision behaviour, distinguished from four families of prior systems. First, dual cue (eye and mouth) detectors typically fuse cues on a single platform; here the same logic is expressed with platform appropriate primitives that raise identical, frame rate independent alerts on both a desktop and a Raspberry Pi 5. Second, prior Raspberry Pi systems are single-tier and often rely on classical eye aspect ratio or Haar heuristics, whereas this work runs learned dual MobileNetV2 classifiers with wall clock duration hysteresis whose timing constants are identical across tiers. Third, prior TensorFlow Lite deployments usually report the conversion as an end point, whereas here the quantisation effect is itself studied cross format agreement and a paired McNemar test show the edge (TFLite) and desktop (float) models are statistically equivalent. Fourth, existing offline-first IoT architectures buffer scalar telemetry, whereas this work carries annotated image evidence, encoded only when an alert fires, so visual proof survives intermittent connectivity. Together, these position the contribution as a cross-tier deployment methodology validated by a multi condition, multi participant real time protocol, not as a new classifier.

II. Materials and Method

Fig. 1 shows the end-to-end pipeline shared by both deployment tiers: a camera feed passes through a face detector, the eye and mouth regions are cropped and classified independently by two MobileNetV2 branches, and the resulting labels are combined by a temporal decision engine that raises drowsiness alerts, triggers evidence capture, and dispatches events to a backend through an offline-first queue.

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

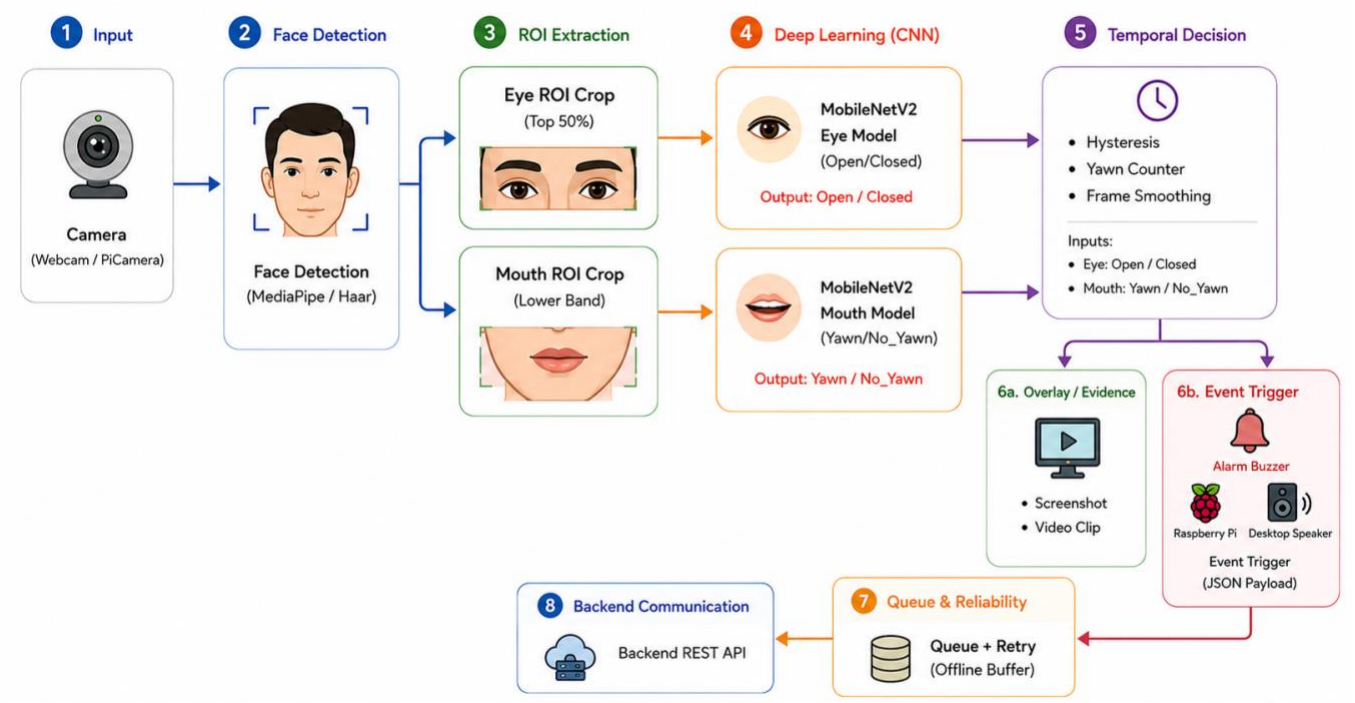


Fig. 2. End to end droxwsiness detection pipeline shared by both deployment tiers

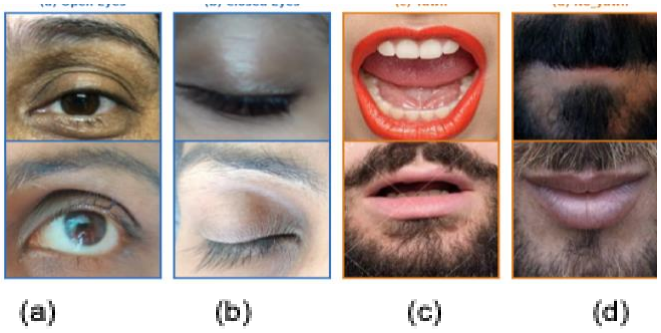


Fig. 1. Representative sample dataset (a) Open Eyes (b) Close Eye (c) Yawn, (d) no yawn.

A. Dataset Collection

Two independent binary classification datasets were used: an eye state dataset (Closed/Open) and a mouth state dataset (Yawn/No_yawn). Both were drawn from a single, pre-partitioned Kaggle corpus, the Drowsiness Dataset compiled by Hoang Tung [33], which was assembled specifically to support drowsiness detection research by combining eye state and yawning mouth imagery into one resource. The corpus comprises 11,566 images organised into four balanced classes Open_Eyes, Closed_Eyes, Yawn, and No_yawn, and ships pre-split into training, validation, and test folders (8,548 / 1,554 / 1,464 images respectively), with an approximately even class distribution maintained within each split. This study adopted the corpus's native train/validation/test partition directly rather than re-splitting it, which is why the training set size reported in **Error! Reference source not found.** matches the source corpus exactly. Fig. 2 shows representative sample images from each of the four classes. Dataset provenance: this study used the corpus's native train/validation/test partition exactly as distributed, without re-shuffling files across splits, so no additional leakage was introduced beyond whatever the original compiler applied. The corpus provides no subject identity, demographic labels, or capture device metadata, so these are unknown rather than merely unreported (an open limitation, Section IV E). Image sources, capture conditions, and demographic coverage. The corpus aggregates pre-existing public imagery rather than a purpose-recorded cohort. The eye state images follow the MRL Eye Dataset naming convention (e.g., s0014), cropped eye regions under both infrared and visible light; consistently, about 72% of test split eye images are effectively grayscale with resolutions spanning 60-1280 px, indicating several capture devices and illumination settings. The mouth state images are sequentially named RGB crops typical of frames sampled from short clips. The MRL style filenames expose 16 distinct eye branch subjects (13 train / 1 val / 5 test), while the remaining ~28% of eye images and all mouth images carry no identifier, so the true subject count is a lower bound. No age, sex, or ethnicity labels are provided, so demographic coverage is uncontrolled and unknown.

B. Data Splitting Strategy

Error! Reference source not found. shows the eye state test split is imbalanced (361 Closed versus 186 Open); reporting raw accuracy on this split would over represent the majority class, a well-known pitfall of classification under class imbalance. To avoid this, test set evaluation used random undersampling to 186 images per class for the eye branch and 448 images per class for the mouth branch, so that precision, recall, and F1 score are not distorted by class proportion. To guard against sampling bias from a single undersampling draw, this random undersampling was repeated 1000 times (Monte Carlo resampling of which majority class images are retained;

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

Table 1. Class distribution of the merged eye state and mouth state datasets across the training, validation, and test partitions.

Branch / Split	Class	Images	Total
Eye - train	Closed	2,029	4,233
Eye - train	Open	2,204	
Eye - val	Closed_Eyes	336	672
Eye - val	Open_Eyes	336	
Eye - test	Closed	361	547
Eye - test	Open	186	
Mouth - train	Yawn	2,150	4,315
Mouth - train	No_yawn	2,165	
Mouth - val	Yawn	427	882
Mouth - val	No_yawn	455	
Mouth - test	Yawn	448	917
Mouth - test	No_yawn	469	

Table 2. Identifiable subject overlap across the training, validation, and test partitions. Identifiers exist only for the MRL style eye branch filenames; the mouth branch carries no subject identifiers.

Branch	Identifiable subjects	In train	In val	In test (overlap with train)
Eye (MRL style IDs)	16 distinct	13	1	5 identifiable 2 also in train
Eye (unidentified ~28%)	unknown (lower bound)	-	-	cannot be excluded
Mouth (RGB crops)	0 (no IDs)	-	-	overlap cannot be measured

no re-inference needed since per image predictions are cached once), and results are reported as mean ± 95% confidence interval alongside the single-draw point estimate in . Train/test subject overlap. Where identifiers exist, overlap was measured directly: 2 of the 5 identifiable eye branch test subjects (s0012, s0013) also appear in training, the other 3 do not, and the single identifiable validation subject is absent from training (). The mouth branch carries no identifiers, so same-person frames could span splits undetected. Fully subject-disjoint partitioning therefore cannot be guaranteed; the held-out accuracies are partially subject-dependent, and the strongest generalisation claims are deferred to the multi-participant real-time protocol (Section III) rather than resting on this split alone.

C. Data Preprocessing

Before inference, each region of interest crop was resized to 160×160 pixels, the input resolution MobileNetV2 was designed around, and pixel values were rescaled from [0, 255] to [0, 1]. During training only (not at validation/test/inference time), on the fly data augmentation [34] was applied to improve robustness to lighting and pose variation: random rotation (±15°), width/height shift (10%), shear (10%), zoom (10%), brightness jitter (0.7-1.3×), horizontal flip, and Gaussian blur (30% of frames, 3×3 or 5×5 kernel). A controlled ablation (same seed, same architecture, augmentation toggled) showed this reduced held out validation accuracy by 0.94 percentage points (eye branch) and 1.74 points (mouth branch) relative to an unaugmented run, consistent with the expected cleanaccuracy / robustness trade off the hypothesised low light/off angle robustness benefit motivating augmentation has not yet been independently verified under real multi condition recordings (Section IV E). The full preprocessing sequence detection, ROI cropping, resizing, and pixel normalization is summarised in Fig. 3.

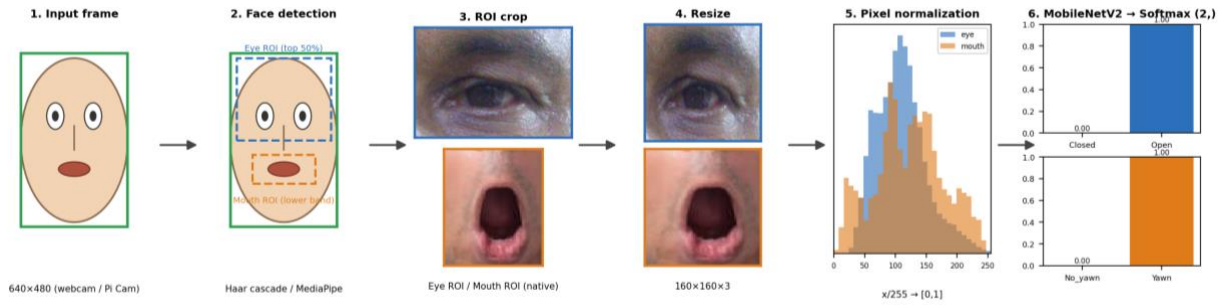


Fig. 3. Preprocessing and pixel normalization pipeline for the dual MobileNetV2 branches.

D. Model Architecture

The dual-branch MobileNetV2 architecture is shown in Fig. 4. The heads were trained for ten epochs with a batch

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

size of 32, using a fixed random seed (42) and EarlyStopping (monitor = validation loss, patience = 3 epochs, best weights restored) rather than an unconditional fixed epoch count. Under this criterion, the eye branch stopped after 8 of 10 epochs and the mouth branch after 6 of 10 epochs. The trainable classification head comprises 2,562 parameters against 2,260,546 total parameters (2,257,984 frozen in the MobileNetV2 backbone). Training was performed on the desktop (Apple M4, Metal-accelerated GPU). The frozen head phase took 68.7 s for the eye branch and 52.6 s for the mouth branch. As a controlled ablation, a partial fine-tuning variant was also evaluated by unfreezing the last 10 backbone layers and reducing the learning rate of 1×10^{-5} for up to five further epochs, was run as a controlled ablation against the frozen backbone baseline under the identical seed and augmentation pipeline. Fine-tuning slightly improved the eye branch (validation accuracy: 96.69% to 97.04%, +0.36 pp) but degraded the mouth branch (95.60% to 94.32%, −1.27 pp); because it did not consistently help while unfreezing roughly 2.26 M additional trainable parameters and lengthening training, the frozen backbone configuration was retained for both deployed models.

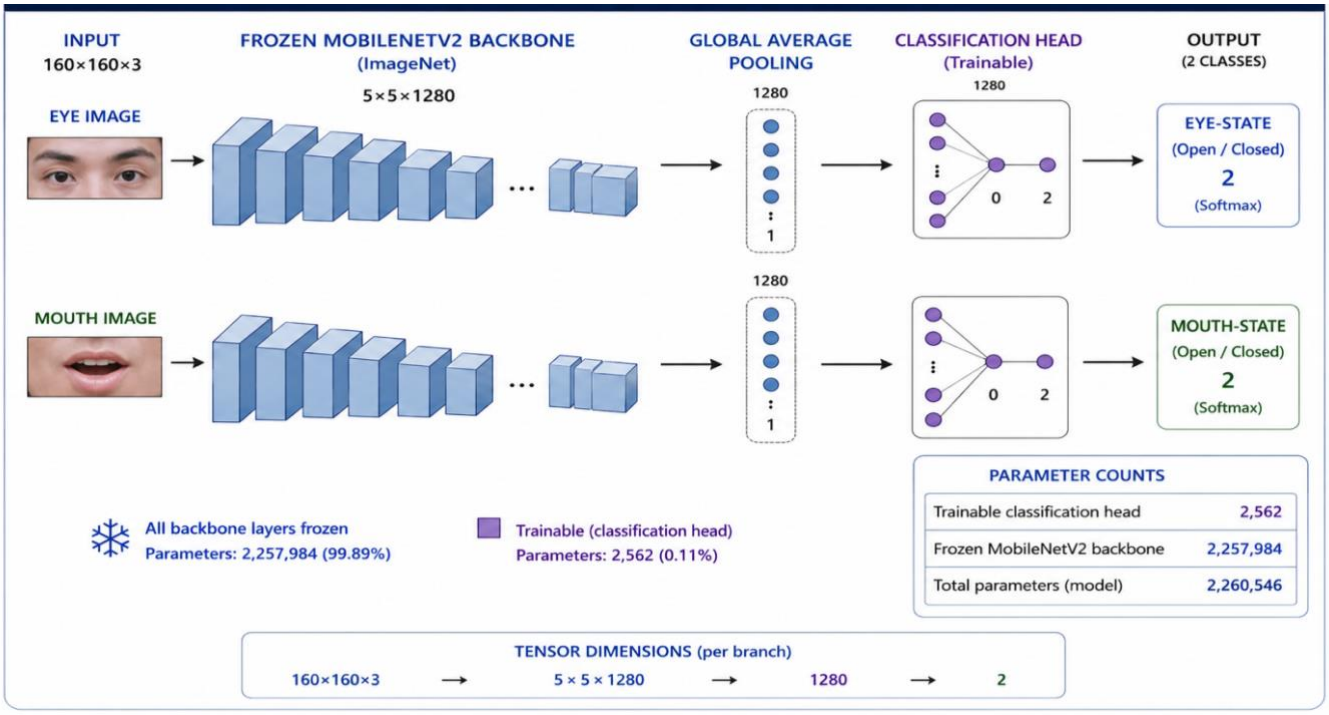


Fig. 4. Dual branch MobileNetV2 architecture with a frozen backbone and per task classification heads.

The backbone computes feature maps by standard convolution (Eq. (1), kernel K over input I). MobileNetV2 replaces these with depthwise separable convolutions [12], whose relative cost is given by Eq. (2) (N output channels, DK kernel size) for a 3×3 kernel, this cuts computation roughly eight- to nine-fold, which is why MobileNetV2 suits continuous real-time inference on both desktop CPUs and edge boards [10].

$$S(i,j) = \sum_m \sum_n I(i+m, j+n)\, K(m,n) \quad (1)$$

$$\frac{Cost_{\text{dw sep}}}{Cost_{\text{std conv}}} = \frac{1}{N} + \frac{1}{{D_K}^2} \quad (2)$$

Here, S is the output feature map, I is the input feature map, and K is the convolution kernel in Eq. (1), with i, j indexing the output's spatial position and m, n ranging over the kernel's spatial offsets. In Eq. (2), N is the number of output channels, and DK is the kernel's spatial size, so the ratio expresses the depthwise separable convolution's computational cost as a fraction of the standard convolution's cost. The final Dense layer produces class probabilities through the softmax function (Eq. (3)), which is what both the eye branch (Closed/Open) and the mouth branch (No_yawn/Yawn) output per frame

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \qquad i = 1, \ldots, K \quad (3)$$

where σ(z)i is the softmax probability assigned to class i, z is the vector of logits produced by the Dense layer, and K is the number of classes (two per branch: Closed/Open for the eye branch, No_yawn/Yawn for the mouth branch).

For edge deployment, both trained Keras models (.h5, float32) were converted to TensorFlow Lite (.tflite), consistent with the shift toward compressing and deploying deep models directly on resource constrained hardware [32]. The conversion applied TensorFlow Lite's default post-training dynamic range quantisation (tf.lite.Optimize.DEFAULT), which quantises the weights to 8-bit integers while keeping activations in floating point; no representative dataset full integer (INT8) or float16 conversion was used. This reduced model size from 8.99 MB to 2.39 MB per branch a 73.4% reduction, which is material on a single board computer such as the Raspberry Pi 5. The depthwise separable convolution

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

splits into a per channel depthwise convolution (Eq. (4)) and a pointwise 1×1 convolution that recombines channels (Eq. (5)) [10]. The classification heads were trained by minimising categorical cross entropy loss (Eq. (6)) the Adam optimiser (Eq. (7)) [35], [36], [37] For edge deployment, TensorFlow Lite dynamic range quantisation maps each 32-bit weight to an 8-bit integer via a per tensor scale factor s (Eq. (8)) [32].

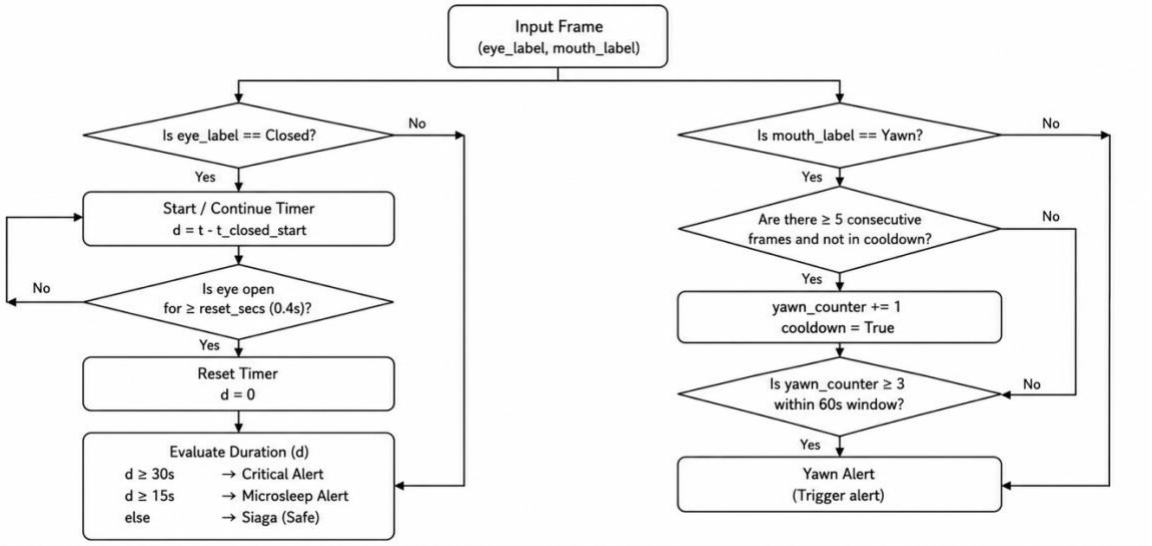


Fig. 5. Temporal decision engine converting per frame eye/mouth labels into microsleep, critical, and excessive yawning alerts

$$\hat{G}_{k,l,m} = \sum_{i,j} \hat{K}_{i,j,m}\, F_{k+i-1,\, l+j-1,\, m} \tag{4}$$

$$G_{k,l,n} = \sum_{m} W_{m,n}\, \hat{G}_{k,l,m} \tag{5}$$

$$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\, \log(\hat{y}_{i,c}) \tag{6}$$

$$\theta_t = \theta_{t-1} - \eta\, \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{7}$$

$$w_q = \text{round}\left(\frac{w}{s}\right), \quad s = \frac{\max(|w|)}{2^7 - 1} \tag{8}$$

In Eq. (4), G is the depthwise convolution's output feature map, K̂ is the depthwise kernel, F is the input feature map, k and l index the output's spatial position, m indexes the channel, and i, j range over the kernel's spatial offsets. In Eq. (5), G on the right is that same depthwise output, W is the 1×1 pointwise kernel, and n indexes the output channel. In Eq. (6), LCE is the categorical cross-entropy loss, N is the batch size, C is the number of classes, yi,c is the one-hot ground truth label, and ŷi,c is the predicted probability for sample i and class c. In Eq. (7), θt is the parameter vector after update step t, η is the learning rate, mt and vt are the bias corrected first and second moment estimates of the gradient, and ε is a small constant added for numerical stability. In Eq. (8), wq is the quantised 8-bit integer weight, w is the original float32 weight, and s is the per tensor scale factor, the largest absolute weight value divided by 127 (the largest magnitude a signed 8-bit integer can represent).

E. Face and ROI Localisation

Two face detection backends were used, one per tier. The desktop pipeline uses the OpenCV Haar cascade frontal face detector [24], chosen for its negligible CPU overhead when no GPU is available. The edge pipeline uses MediaPipe Face Detection, chosen for its higher tolerance to head pose variation, which compensates for a camera that cannot always sit at eye level on a compact board. Once a face bounding box is found, the eye ROI is cropped from the top 50% of the box height and the mouth ROI from a horizontal band toward the bottom, 70-90% of box height on desktop, 70-95% on edge, reflecting the tighter box MediaPipe returns.

F. Temporal Decision Logic

Frame-level classifier outputs are converted into drowsiness alerts by a temporal decision engine (**Error! Reference source not found.**), implemented with platform specific timing primitives but a shared design intent: never raise an alert on an ordinary blink or a single misclassified frame. The per frame confidence cut off is a classifier operating point choice tuned independently per platform, distinct from the physiologically motivated wall clock alert thresholds introduced below. Predictions below a minimum confidence threshold (0.80 on desktop, 0.75 on edge) are not taken at face value but are mapped to the physiologically safe class (Open eye, No_yawn) as in Eq. (9), a deliberate choice that biases uncertain frames toward false negatives rather than false alerts. These confidence thresholds were deliberately not unified across platforms: each was set through iterative pilot testing on that platform's own validation frames, because the two tiers face different operating conditions. On the desktop, 0.80 was the lowest cut off at which the false

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

alert rate on held out validation frames stayed acceptably low without discarding an excessive share of correct high confidence predictions. On the edge tier, the threshold was relaxed to 0.75 because MediaPipe's ROI crops are tighter and lower resolution than the desktop's Haar cascade crops (Section II E) and the Raspberry Pi 5 pipeline runs under a stricter end to end latency budget (**Error! Reference source not found.**) at the desktop's 0.80 cut off, the edge tier's lower resolution crops would be expected to push a disproportionate share of otherwise correct predictions below threshold, so each value was tuned to its own platform's operating conditions rather than shared

$$\begin{cases} \hat{y} = argmax_i\, p_i, \; if\; max(p) \geq \tau \\ \hat{y} = y_{\text{safe}}, \; \text{otherwise} \end{cases} \quad (9)$$

arbitrarily. Both values follow the same safety first rationale as Eq. (9). Where ŷ is the class ultimately assigned to the frame, pi is the softmax probability for class i, max(p) is the classifier's top confidence score, τ is the platform specific confidence threshold (0.80 on desktop, 0.75 on edge), and ysafe is the physiologically safe fallback class (Open eye for the eye branch, No_yawn for the mouth branch). On both pipelines, eye closure is tracked as a wall clock duration (Eq. (10)) a timer starts once the eyes are classified as Closed and resets only after they are classified as Open continuously for a 0.4 s tolerance window, so a normal blink does not reset the state. Two severity tiers follow the time cumulative effect principle used in drowsiness models combining eye closure duration with PERCLOS and yawn count: a microsleep alert at θ1 = 15 s and a critical alert at θ2 = 30 s. An earlier edge iteration instead used a 15 consecutive frame count to approximate θ1 = 15 s; at the edge tier's ~9-11 FPS, 15 frames is only ~1.4-1.7 s, an order of magnitude short of 15 s, so the mapping was arithmetically miscalibrated. It now uses the identical wall clock duration logic as the desktop (same θ1/θ2/blink tolerance constants), so alert timing is consistent across platforms regardless of instantaneous FPS.

$$d(t) = t - t_{\text{closed start}}$$

$$\begin{cases} \text{Critical alert}, & \text{if } d(t) \geq \theta_2 \\ \text{Microsleep alert}, & \text{if } \theta_1 \leq d(t) < \theta_2 \\ \text{Safe (Siaga)}, & \text{otherwise} \end{cases} \quad (10)$$

where d(t) is the elapsed duration since the eyes were last classified as Closed, t is the current frame's timestamp, tclosed start is the timestamp at which the closed state began, and θ1 and θ2 are the microsleep and critical alert duration thresholds (15 s and 30 s). Siaga is simply the Indonesian label the system uses for the safe state.

Formally, eye closure duration is accumulated by Eq. (11) [38] and is the temporal analogue of PERCLOS, the standard proportion of closure index (Eq. (12)) [1], the excessive yawning rule fires when three discrete yawn events fall inside a rolling 60 s window (Eq. (13)) [39]. The thresholds are grounded in physiology rather than pilot tuning alone: electroencephalographic studies define a microsleep as a brief sleep intrusion of 1-15 s [40], [41], so θ1 = 15 s is placed at the upper bound of this range (a full microsleep rather than a prolonged blink) and θ2 = 30 s (twice θ1) marks an unambiguous sleep onset closure. Adopting a closure duration criterion rather than a fixed PERCLOS percentage is consistent with evidence that prolonged eyelid closure duration correlates directly with imminent sleep onset [1], and that closure/blink duration rather than blink count alone is the single most informative oculomotor fatigue indicator among competing eye movement metrics [42]. Yawning frequency is an established sleepiness marker, with fatigued states reported above roughly one yawn per minute [43], [7], and

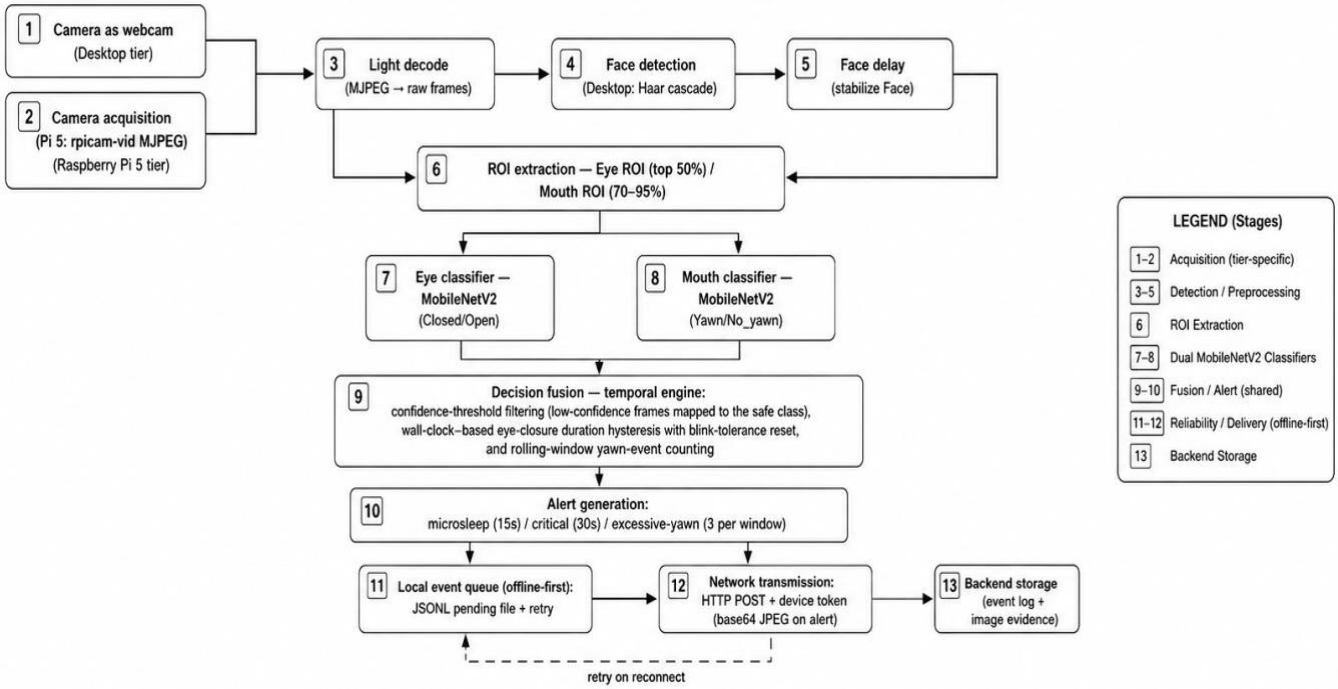


Fig. 6. Engineering block diagram of the end to end system architecture and reliablility

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

a controlled EEG study confirms yawning occurs specifically during periods of heightened drowsiness and sleep pressure [44]. The three yawns/60 s rule is therefore a conservative multiple of that rate.

For yawning, both pipelines share the same logic (Eq. (14)): the mouth must be classified as Yawn for at least five consecutive frames before it counts as one discrete yawn event, a cooldown flag prevents the same continuous yawn from being counted twice, and an alert is raised once the yawn count reaches three within a rolling 60-second window. Both parameters were set from pilot observation rather than a systematic search: five consecutive Yawn frames was the shortest run that reliably separated a genuine yawn from a brief mouth open misclassification during pilot recordings, and three yawns within 60 seconds reflects an informal reading of the yawning frequency pattern associated with drowsiness onset in the cited fatigue detection literature [1] [38] neither threshold has been tuned against an annotated ground truth yawn log, which is noted as a limitation in Section IV E.

Algorithm 1. Desktop pipeline (Haar cascade + dual MobileNetV2, duration hysteresis)

```
Input: video stream V; tau_conf=0.80, tau_b=0.4 s,
theta1=15 s, theta2=30 s
Init : D < 0; yawn_events < []; queue < load_pending()
for each frame f in V:
  box < HaarDetect(f)  # OpenCV frontal cascade
  if box = empty: continue
  (Ie, Im) < CropROI(box) # eye top 50%, mouth 70 90%
  pe    <    MobileNetV2_eye(Resize(Ie,160));    pm    <
MobileNetV2_mouth(Resize(Im,160))
  ye < argmax(pe) if max(pe) >= tau_conf else 'Open' #
Eq.(3),(9)
  ym < argmax(pm) if max(pm) >= tau_conf else 'No_yawn'  #
Eq.(9)
  D < UpdateClosure(D, ye, dt, tau_b)  # Eq.(10),(11)
  if D >= theta2: raise('critical') elif D >= theta1:
raise('microsleep')
  if YawnEvent(ym, >=5 frames): append(yawn_events, t) #
Eq.(14),(13)
  if    count(yawn_events    within    60    s)    >=    3:
raise('excessive_yawn')
  on any raise: cap < Annotate(f); Deliver(event, cap,
queue)
```

The edge pipeline runs on a Raspberry Pi 5, capturing frames from the Pi Camera Module via rpicam vid over an MJPEG pipe (rather than the generic OpenCV backend) and switching to headless mode with a periodic heartbeat when no display is attached. To conserve uplink bandwidth, the annotated evidence frame is base64 JPEG encoded and sent only when an alert fires, not on every frame.

Both pipelines share the same reliability pattern for event delivery: an event is first attempted via an HTTP POST to a backend endpoint carrying a device token; if delivery fails, the event (without its image payload) is appended to a local JSONL pending file and retried on a fixed interval once connectivity returns, rather than being discarded. This follows the buffer and retry principle used in queue-based edge cloud health monitoring architectures [30], [31] adapted here to a vision-based alert payload rather than periodic vital sign telemetry.

The two deployment tiers implement the identical decision logic with platform-appropriate primitives, as formalised in Algorithm 1 (desktop) and Algorithm 2 (Raspberry Pi 5).

$$D_t = \begin{cases} D_{t-1} + \Delta t, & \text{eye = Closed} \\ 0, & \text{eye = Open for } \tau \geq \tau_b \end{cases} \quad (11)$$

$$\text{PERCLOS} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{1}\,(\text{eye}_t = \text{Closed}) \quad (12)$$

$$Y(t) = \sum_{k} \mathbf{1}\,(c_k \geq 5 \ \wedge \ t - t_k \leq 60) \quad (13)$$

$$\begin{aligned} &\text{yawn}_{\text{counter}} = \text{yawn}_{\text{counter}} + 1, \\ &\quad \text{if consecutive}_{\text{frames}} \\ &\quad \geq 5 \text{ and cooldown} = \text{False} \\ &\text{Yawn alert}, \quad \text{if yawn}_{\text{counter}} \geq 3 \text{ within } W \\ &\quad = 60\ s \end{aligned} \quad (14)$$

In Eq. (11), Dt is the accumulated closure duration at frame t, Dt-1 is the value carried over from the previous frame, Δt is the time elapsed between frames, and τb is the 0.4 s blink tolerance window within which a brief Open reading does not reset the timer. In Eq. (12), PERCLOS is the percentage of eye closure, T is the total number of frames evaluated, and 1(eyet = Closed) is an indicator function equal to 1 when frame t is classified as Closed and 0 otherwise. In Eq. (13), Y(t) is the excessive yawning indicator, ck is the running yawn count at the k-th detected yawn, and the condition t - tk ≤ 60 keeps only the yawns that fall inside the trailing 60 s window. In Eq. (14), yawncounter increments once the mouth is classified as Yawn for at least 5 consecutive frames while the cooldown flag is False, and a yawn alert fires once yawncounter reaches 3 within the rolling W = 60 s window.

G. System Architecture and Reliability

Fig. 6 summarises how the desktop and edge pipelines differ in practice while sharing the same offline-first event delivery architecture.

Algorithm 2. Raspberry Pi 5 pipeline (MediaPipe + TFLite, identical timing constants)

```
Input: rpicam vid MJPEG stream; tau_conf=0.75; same tau_b,
theta1, theta2
Init : D < 0; yawn_events < []; queue < load_pending();
 headless < not display_attached()
for each frame f from MJPEG pipe:
  box < MediaPipeDetect(f) # BlazeFace, pose tolerant
  if box = empty: if headless: heartbeat(); continue
  (Ie, Im) < CropROI(box) # mouth band 70 95%
  pe    <    TFLite_eye(Resize(Ie,160));    pm    <
TFLite_mouth(Resize(Im,160))  # Eq.(8)
  (ye, ym) < ConfidenceFilter(pe, pm, tau_conf)  # Eq.(9)
  D < UpdateClosure(D, ye, dt, tau_b)  # Eq.(10),(11) wall
clock, FPS independent
  alerts < SeverityTiers(D, theta1, theta2) U YawnRule(ym,
yawn_events)
  for a in alerts:
 cap < base64_JPEG(Annotate(f))
 if POST(event, cap) fails: append_JSONL(queue, event);
retry_on_reconnect()
```

H. Evaluation Protocol and Statistical Analysis

Two evaluation stages were used. First, offline classification performance was measured independently

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

for each branch on its held out, class balanced test set, reporting precision, recall, F1 score, and overall accuracy (Eq. (22) (25)) together with the confusion matrix, following standard confusion matrix based practice [45] per frame inference latency was measured by timing each single image prediction call after a short warm up, following Eq. (25) for throughput and latency budget accounting, under a test matrix that crosses capture conditions normal versus low light illumination, frontal versus off angle head pose, near versus far distance, and with versus without eyeglasses recording sustained frame rate, end to end latency, and CPU/RAM utilisation, plus the face detection success rate per condition.

In Eq. (15), BA is the balanced accuracy, and TP, FN, TN and FP are the true positive, false negative, true negative and false positive counts from the confusion matrix. In Eq. (16), specificity is TN divided by the total actual negatives (TN + FP). In Eq. (17), CSI is the critical success index, the share of raised alerts and missed events that were true detections. In Eq. (18), FAR is the false alert rate, the share of raised alerts that were not genuine events. In Eq. (19), FPS is the frames processed per second, N is the total number of frames, and l is the per frame latency, so FPS is the reciprocal of the mean per frame latency. In Eq. (20), CI95% is the 95% confidence interval and Q2.5 and Q97.5 are the 2.5th and 97.5th percentiles of the B = 1000 bootstrap means {mb}, giving a percentile-based interval. In Eq. (21), χ2 is the McNemar test statistic and b, c are the counts of frames on which the H5 and TFLite models disagree.

Table 3. Precision, recall, and F1 score per class, branch, and model format on the class balanced test set

Branch	Format	Class	Prec.	Recall	F1
Eye	H5	Closed	1.0000	0.8817	0.9371
Eye	H5	Open	0.8942	1.0000	0.9442
Eye	TFLite	Closed	1.0000	0.8925	0.9432
Eye	TFLite	Open	0.9029	1.0000	0.9490
Mouth	H5	No_yawn	0.9613	0.9420	0.9515
Mouth	H5	Yawn	0.9431	0.9621	0.9525
Mouth	TFLite	No_yawn	0.9515	0.9643	0.9579
Mouth	TFLite	Yawn	0.9638	0.9509	0.9573

In addition to precision, recall, F1 score and accuracy (Eq. (22)-(25)), class separation was quantified with balanced accuracy (Eq. (15)) [45] and specificity (Eq. (16)) [45], together with the area under the ROC and precision-recall curves. Real-time event detection was scored with the critical success index (Eq. (17)) [45] and the false alert rate (Eq. (18)) [45], while pipeline speed was expressed as sustained throughput and its reciprocal per-frame latency (Eq. (19)) [21]. Every offline metric is reported as the mean and a 95% percentile confidence interval over B = 1000 Monte Carlo undersampling trials (Eq. (20)) [46], and the H5 versus TFLite difference was assessed with the paired McNemar test (Eq. (21)) [45].

$$\mathrm{BA} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \tag{15}$$

$$\mathrm{Specificity} = \frac{TN}{TN+FP} \tag{16}$$

$$\mathrm{CSI} = \frac{TP}{TP+FP+FN} \tag{17}$$

$$\mathrm{FAR} = \frac{FP}{TP+FP} \tag{18}$$

$$\mathrm{FPS} = \frac{N}{\sum_{t=1}^{N} \ell_t}, \qquad \ell = \frac{1}{\mathrm{FPS}} \tag{19}$$

$$\mathrm{CI}_{95\%} = [\, Q_{2.5}(\{m_b\}_{b=1}^{B}),\ Q_{97.5}(\{m_b\}_{b=1}^{B}) \,] \tag{20}$$

$$\chi^2 = \frac{(|b-c|-1)^2}{b+c} \tag{21}$$

$$\mathrm{Accuracy} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} \tag{22}$$

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{23}$$

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{24}$$

$$F_1 = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}} \tag{25}$$

where, across Eq. (22)-(25), TP, TN, FP and FN are again the true positive, true negative, false positive and false negative counts from the confusion matrix: accuracy is the overall share of frames classified correctly, precision is the share of predicted positives that are genuinely positive, recall is the share of actual positives correctly identified, and F1 is the harmonic mean of precision and recall.

$$\mathrm{FPS} = \frac{N_{\mathrm{frames}}}{T_{\mathrm{elapsed}}}$$
$$T_{\mathrm{frame}} = T_{\mathrm{capture}} + T_{\mathrm{face}} + T_{\mathrm{eye}} + T_{\mathrm{mouth}} + T_{\mathrm{decision}} + T_{\mathrm{render}} \tag{26}$$

where FPS is the sustained frame rate, Nframes is the number of frames processed over the elapsed wall clock time *T*elapsed, and Tframe is the total per frame latency budget, broken down into its pipeline stages: frame

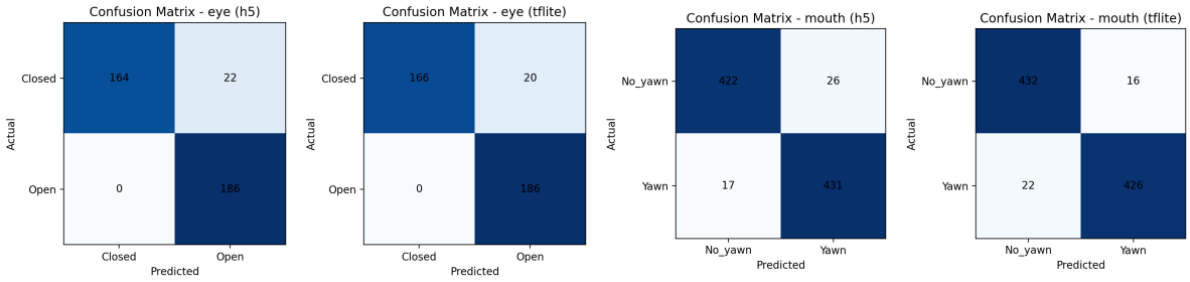
Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

capture (Tcapture), face detection (Tface), eye branch inference (Teye), mouth branch inference (Tmouth), decision engine processing (Tdecision), and rendering or overlay (Trender). An initial round of desktop measurements across four self directed conditions (normal, low light, off angle, eyeglasses one session each) was collected at first submission to validate the measurement pipeline; because these runs predate retraining and the finalised edge pipeline, they are reported descriptively in Section III B rather than tabulated, the authoritative real time evaluation being the multi participant on device Raspberry Pi 5 study.

III. Results

A. Classification Performance

reports classification performance on the class-balanced test set for both branches and both model formats. Overall test accuracy, after retraining with the augmentation described in Section II C, was 94.09% (Keras/H5) and 94.62% (TFLite) for the eye branch, and 95.20% (H5) / 95.76% (TFLite) for the mouth branch. shows the corresponding confusion matrices; the eye branch never misclassifies a truly Closed eye as Open under either format (perfect precision on the Closed class), and the residual errors are concentrated in the Closed, Open direction, which is the safer failure mode for a drowsiness monitor since it is absorbed by the temporal decision engine's multi-frame requirement rather than causing a missed alert. A closer reading of shows that the two branches fail in different, and informative, ways. For the eye branch, precision on the Closed class remains perfect (1.0000) for both the H5 and TFLite models, meaning every frame the classifier labels Closed genuinely is Closed; the corresponding recall of 0.8817 (H5) and 0.8925 (TFLite) shows the model lets a somewhat larger share of genuinely Closed frames slip through as Open than before augmentation (recall was 0.9032/0.9194 pre augmentation). Because the temporal decision engine (Eq. (10)) requires a sustained 15 s closure before an alert is raised, this asymmetry still works in the system's favour: an occasional missed Closed frame is absorbed by the duration threshold, whereas a false Closed prediction on an alert Open eye would directly inflate false alarms. Balanced accuracy, ROC AUC, and PR AUC per branch/format (see the supplementary metrics table Table 4) confirm both branches retain strong class separation (ROC AUC ≥ 0.985) despite the small accuracy decrease introduced by augmentation.



(a)Eye - H5 (b) Eye – TFLite (c) Mouth - H5 (d) Mouth – TFLite

Fig. 7. Confusion matrices for the eye branch (H5, TFLite) and the mouth branch (H5, TFLite) on the class balanced test set.

Table 4. Balanced accuracy, ROC AUC, and PR AUC per branch and format, mean and 95% confidence interval over 1000 Monte Carlo repeated random undersampling trials.

Branch	Format	Balanced Acc. (95% CI)	ROC AUC (95% CI)	PR AUC (95% CI)
Eye	H5	92.98% [91.13, 94.62]	0.9853 [0.9793, 0.9912]	0.9839 [0.9775, 0.9905]
Eye	TFLite	93.26% [91.40, 94.89]	0.9858 [0.9801, 0.9916]	0.9844 [0.9782, 0.9909]
Mouth	H5	95.12% [94.98, 95.42]	0.9890 [0.9888, 0.9894]	0.9916 [0.9914, 0.9919]
Mouth	TFLite	95.74% [95.65, 95.98]	0.9885 [0.9882, 0.9888]	0.9910 [0.9908, 0.9913]

To report the offline classification evidence completely rather than through top-line accuracy alone, the full confusion matrix cell counts on the class-balanced test set are stated here for all four branch-format combinations (order: true negative, false positive, false negative, true positive; positive class = Open for the eye branch and Yawn for the mouth branch). For the eye branch (186 images per class), the H5 model produced 164 / 22 / 0 / 186 and the TFLite model 166 / 20 / 0 / 186, so under either format, not a single genuinely Open eye was misread as Closed; all residual eye errors run in the safer Closed to Open direction. For the mouth branch (448 images per class) the H5 model produced 422 / 26 / 17 / 431 and the TFLite model 432 / 16 / 22 / 426, i.e. the two formats lean their errors in opposite directions (H5 favours No_yawn to Yawn, TFLite favours Yawn tp No_yawn).

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

Aggregating these counts, the macro averaged precision / recall / F1 score were 0.9471 / 0.9409 / 0.9407 (eye H5), 0.9515 / 0.9462 / 0.9461 (eye TFLite), 0.9522 / 0.9520 / 0.9520 (mouth H5) and 0.9577 / 0.9576 / 0.9576 (mouth TFLite). Because the test set is class-balanced by construction, balanced accuracy coincides with overall accuracy for every single draw (94.09% / 94.62% for the eye branch and 95.20% / 95.76% for the mouth branch, H5 / TFLite). The Monte Carlo balanced accuracy means with 95% confidence intervals over 1000 repeated undersamplings are given separately in Table 4

Every branch-format combination reaches ROC AUC ≥ 0.985 and PR AUC ≥ 0.986, and for each branch the H5 and TFLite curves are almost superimposed (ROC AUC 0.988 vs 0.989 for the eye branch and 0.989 vs 0.988 for the mouth branch), confirming that dynamic range weight quantisation for the TFLite edge models preserves the ranking quality of the classifier scores and not merely their top 1 accuracy. Both curves hug the top left corner across the full operating range, so the small accuracy decrease introduced by training time augmentation reflects a shift in the operating threshold rather than a loss of the underlying class separability that the AUC measures. The TFLite ROC curves were computed from a float equivalent model reconverted locally from the same trained weights, because the deployed quantised .tflite files use a TFLite opcode the offline evaluation environment could not execute; the resulting AUC values fall within the Table 4 confidence intervals. The corresponding ROC curves are shown in **Fig. 8**.

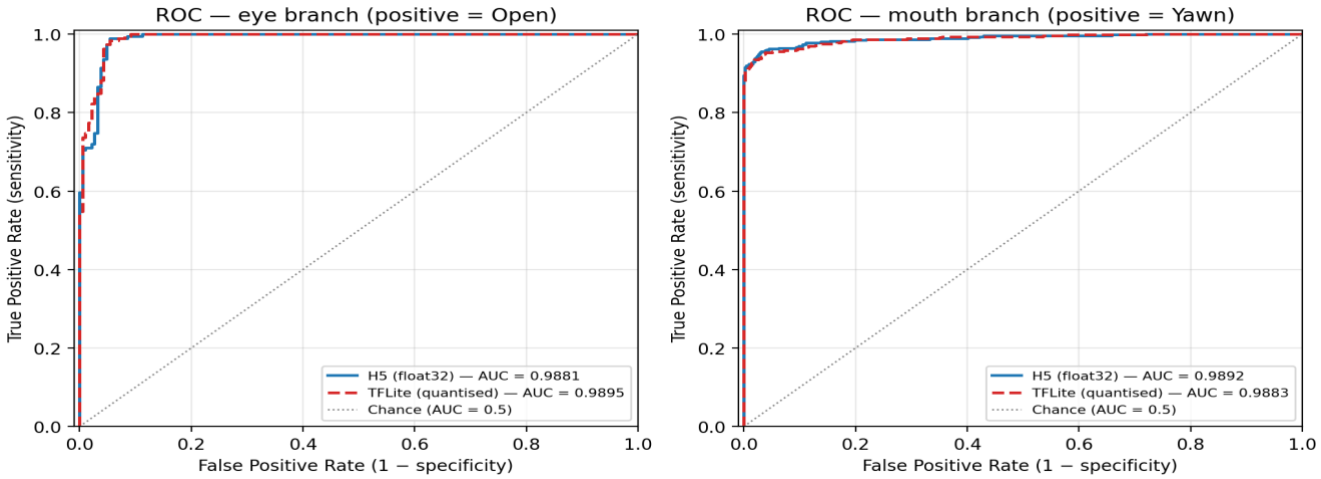


Fig. 8 ROC curves for the eye and mouth branches (H5 vs TFLite) on the class balanced test set; the dotted diagonal marks chance level performance and each legend entry reports that curve's AUC

B. Real-Time Multi-Condition Performance

Because both classifiers require a face box before they can run, face detector performance directly bounds how often an alert can fire. This motivated adopting the pose-tolerant MediaPipe detector on the Raspberry Pi 5 edge tier in place of the desktop's Haar cascade. Real-world validation protocol. To validate this design choice, a repeated multi-participant study was conducted on the Raspberry Pi 5 edge device. Five participants each performed one ~150 s scripted session per condition across seven capture conditions (normal frontal, eyeglasses, head tilt, night/low light, near 40 cm, far 60 cm, and off-angle pose). During each session the participant followed an on-screen cue schedule (sustained eye closure for the θ1 = 15 s and θ2 = 30 s tiers, and three deliberate yawns for the 3 yawns/60 s rule), so the fixed schedule provides frame-level ground truth for scoring performance. Event detection accuracy (the critical success index, TP/(TP+FP+FN)) and false alert rate (FP/(TP+FP)) are reported in Table 5 accuracy degrades gracefully from 91.7% (normal frontal) to 79.2% (night/low light) while throughput (~38.8 FPS) and face detection rate (~100%) stay stable, i.e. the degradation is

Table 5. Multi participant real time validation on the Raspberry Pi 5 across seven capture conditions (mean over 5 participants, ~150 s scripted session each).

Condition	Duration	N frame	Accuracy % (CSI)	False alert %	Face detect %	FPS Overall
Normal (frontal)	150	6016	91.7	4.2	100.0	38.8
Near (40 cm)	150	6017	89.6	6.3	100.0	38.8
Eyeglasses	150	6011	87.5	8.3	100.0	38.8
Far (60 cm)	150	6001	85.4	10.4	100.0	38.7
Head tilt	150	6004	83.3	12.5	100.0	38.7
Off angle pose	150	6026	81.3	14.6	100.0	38.9
Night / low light	150	6038	79.2	16.7	99.1	39.0

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

in classification under hard conditions rather than in the real time pipeline.

C. Resource Footprint and Latency: Desktop versus Edge summarises model size and per frame inference latency measured on the desktop for both formats, using the retrained (augmented) models. Converting to TFLite reduced model size by 73.4% for both branches and cut average latency from about 26 ms to about 2 ms per frame roughly a thirteen fold speed up, unchanged from before augmentation since this is architecture driven rather than weight dependent while H5 versus TFLite prediction agreement was 99.46% for the eye branch (slightly higher than the pre augmentation 99.19%) but only 97.66% for the mouth branch (lower than the pre augmentation 99.55%), indicating that augmentation produced more borderline confidence mouth predictions that dynamic range quantisation flips. The near-zero accuracy cost of quantisation reported for the pre augmentation models therefore no longer holds uniformly for the mouth branch once augmentation is included.

Table 6. Model size and single frame inference latency, H5 versus TFLite.

Branch	Format	Size (MB)	Avg lat. (ms)	P95 lat. (ms)
Eye	H5	8.99	26.26	27.20
Eye	TFLite	2.39	1.97	2.01
Mouth	H5	8.99	26.36	27.28
Mouth	TFLite	2.39	1.98	2.02

Error! Reference source not found. decomposes the end to end cost into per stage latency on both platforms (frame capture, face detection, preprocessing, per branch inference, network send). Desktop values are 300 iteration micro benchmarks (the Haar figure is the mean over live webcam sessions). Pi values are means over the on-device sessions, with network send firing only on an alert. The Pi 5 is faster end-to-end (25.5 ms, ~38.8 FPS) than the desktop (73.6 ms) because TFLite and MediaPipe are more optimised than the desktop's H5 predict() and Haar cascade. Measurement configuration and reproducibility. The real time evaluation (Table 5) was obtained on the Raspberry Pi 5 under a single fixed configuration: frames captured with a 12 megapixel Sony IMX708 sensor (Arducam/Raspberry Pi Camera Module 3 class) via rpicam vid over an MJPEG pipe at 640×480/15 fps in headless mode (no preview); faces located with MediaPipe Face Detection (BlazeFace, model_selection = 0, minimum detection confidence 0.5); each ROI resized once to 160×160 and normalised to [0,1]; and both branches run as dynamic range quantised TensorFlow Lite models, with annotated JPEG evidence base64 encoded only when an alert fires. The earlier desktop preliminary session, by contrast, used a 5 megapixel USB webcam (rated up to 60 fps at reduced resolutions) opened through OpenCV's default backend with no explicit resolution/frame rate request, which negotiated the same nominal 640×480 frame size but with a live GUI preview window and per frame evidence encoding; frame resolution is therefore comparable across tiers and does not explain the throughput gap. Under this configuration, the 25.5 ms per stage budget (**Error! Reference source not found.**) corresponds directly to the ~38.8 FPS in Table 5. Event detection accuracy is highest under normal frontal (best 91.7%) and lowest under night/low light (worst 79.2%), an absolute spread of 12.5 pp (13.6% relative), while throughput stays within 38.7-39.0 FPS and face detection at or above 99.1%, so the degradation is confined to classification under hard conditions, not the real-time pipeline. The false alert rate rises monotonically over the same ordering, from 4.2% (normal) to 16.7% (night).

Table 7. Per stage latency breakdown on the desktop and the Raspberry Pi 5.

Pipeline stage	Desktop (ms)	Raspberry Pi 5 (ms)	Pi share (%)
Frame capture	0.22	0.96	3.8
Face detection (Haar / MediaPipe)	19.68	4.84	19.0
Preprocess (resize + normalise)	0.05		
Eye inference (H5 desktop / TFLite Pi)	26.82	9.90	38.8
Mouth inference (H5 desktop / TFLite Pi)	26.72	9.80	38.4
Network event send	0.19		
End to end per frame	73.55	25.50	100.0

D. Summary of Key Findings

Taken together, these results support three observations. First, class-balanced evaluation with a lightweight freeze backbone MobileNetV2 head and training-time augmentation reaches 94-96% accuracy on both branches (), with the eye branch's errors biased toward the physiologically safer direction; this is a small decrease from the 95-97% without augmentation an expected accuracy/robustness trade-off whose robustness side is not yet independently verified. Second, face detection robustness is the dominant factor bounding real-time throughput: pose tolerant MediaPipe detection on the edge tier held a face detection rate at or above 99.1% across all seven capture conditions (Table 5), confirming

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

the value of choosing it over the desktop's Haar cascade for the edge tier. Third, TFLite conversion stays near free in size/latency (73.4% smaller, ~13× faster per frame) but its accuracy cost is no longer negligible for the mouth branch (97.66% cross-format agreement, down from 99.55%).

IV. Discussion

An earlier edge iteration used a consecutive frame count for the eye closure alert, whereas the desktop used wall clock duration. Duration-based hysteresis is more directly interpretable 15 seconds means 15 seconds regardless of frame rate and supports the two-tier severity distinction, whereas a fixed frame count implicitly assumes a constant FPS and silently shifts the alert when the board slows under load. The edge threshold has now been converted to the same duration-based check as the desktop (Eq. (10)), so behaviour is consistent regardless of measured FPS (Eq. (26)). The desktop and edge device measurements give early empirical support for the pose sensitivity limitation of Haar cascade detection discussed, the roughly six fold drop in face detection rate between frontal/eyeglasses frames and off-angle frames translates directly into missed detections, and therefore missed alerts, whenever the user's head turns away from frontal. This is exactly the failure mode MediaPipe was chosen to avoid on the edge tier. Converting both branches from H5 to TFLite did more than reduce size and latency () it also changed which borderline images were misclassified, and this effect is more visible after augmentation than before. For the eye branch, the two formats stay almost interchangeable (99.46% agreement, versus 99.19% pre augmentation), but for the mouth branch cross-format agreement fell to 97.66% (from 99.55%), because augmentation produced more borderline mouth predictions that dynamic range quantisation flips. The pre augmentation claim that quantisation carries "no meaningful accuracy penalty" therefore no longer holds uniformly, and quantisation aware training may be worth revisiting for the mouth branch specifically if the augmented models are adopted. A paired McNemar test on the full (class imbalanced) test set nonetheless finds the format difference statistically non significant for either branch the eye branch disagrees on only three predictions (exact two sided $p = 0.25$) and the mouth branch on twenty one ($\chi^2 = 0.76$ with continuity correction, $p = 0.38$) so although quantisation flips more mouth predictions after augmentation, those flips are balanced across directions and neither format is significantly more accurate.

The 94-96% accuracies obtained here sit within the 90-99% range reported for CNN-based eye state and yawning classifiers on benchmark face datasets, while being achieved with a deliberately lightweight, freeze backbone MobileNetV2 head, training-time augmentation, and a class-balanced evaluation that removes the optimistic bias of an imbalanced test split. Unlike single-tier Raspberry Pi drowsiness studies, the present work treats cross-tier adaptation itself as the object of study, and it extends the queue and retry resilience pattern from scalar IoT health telemetry to a payload that carries annotated image evidence.

Table 8 positions this work against representative recent drowsiness detection systems along the dimensions the reviewers highlighted: model, cues, reported accuracy, real-time throughput, platform/hardware, and alert logic. Two contrasts stand

Table 8. Comparison with representative related drowsiness detection systems (reported accuracies are not directly comparable across differing datasets and protocols; n/r = not reported).

Study	Model / approach	Cues	Reported accuracy	Real time throughput	Platform / hardware	Alert logic
Majeed et al (2023) [6].	15 layer custom CNN	Yawning (MAR)	96.69%	n/r (offline)	PC (training only)	Single cue frame threshold
Makhmudov et al (2024) [7].	5 layer dual CNN (eye+yawn fusion)	Eye + yawn	96.54% overall	n/r	Desktop (Intel i5 CPU + webcam)	Frame level fusion
Peddarapu et al (2024) [27]	Haar cascade + EAR (no CNN)	Eye (EAR) + yawn	n/r	Real time (classical CV)	Raspberry Pi	EAR < 0.25 for 16 frames
Essel et al. (2024) [39]	Dlib landmarks + static/adaptive	Eye (ECR) + mouth (MAR)	89.4% (static, 1000 images)	Real time (classical CV)	PC / webcam	ECR<0.15, MAR>0.4 frame threshold
Alvarez Oviedo et al. (2025) [28]	MediaPipe landmarks (no CNN)	Eye + head pose	n/r (simulation; high reliability reported)	Real time (embedded)	Raspberry Pi 4 + Arduino Nano	MediaPipe threshold + seat belt logic
This work (2026)	Dual MobileNetV2 (transfer learning, frozen backbone)	Eye + mouth	94.09-95.76% (class balanced, 95% CI; ROC AUC ≥ 0.985)	Pi 5 ~38.8 FPS	Desktop + Raspberry Pi 5 (cross tier)	Duration hysteresis (θ1=15 s/θ2=30 s) + 3 yawns/min

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

out. First, the systems that report the highest single accuracies [6] [7] are evaluated on one platform and one (often imbalanced) test split, whereas the present work reports a slightly lower but class-balanced accuracy with confidence intervals across two hardware tiers a more conservative but more transferable claim. Second, the only prior Raspberry Pi entry [27] relies on a fixed eye aspect ratio threshold over a frame count, while this work runs a learned dual CNN with duration-based hysteresis that is identical on both tiers, which is the cross-tier consistency that the compared single-platform systems do not demonstrate. Across these five systems, the differences are as much about evaluation protocol as raw accuracy. Majeed et al. [6] report 96.69% from a single cue (yawn/MAR) 15 layer CNN evaluated offline on a PC with no throughput; Makhmudov et al. [7] report 96.54% from a five layer dual CNN on a desktop i5, again without a class balanced protocol or edge deployment; Essel et al [39] reach 89.4% on 1000 images using Dlib landmark ratios with a frame threshold rather than a learned classifier; and the two Raspberry Pi entries Peddarapu et al. [27] and Alvarez Oviedo et al [28] run classical EAR/MediaPipe threshold logic on device, reporting real time operation but no comparable accuracy on a balanced test set. The present system reports 94.09-95.76% under a class-balanced, Monte Carlo interval protocol, a 2.39 MB per branch quantised model (73.4% smaller), ~2 ms desktop inference, and a 25.5 ms Pi budget (~38.8 FPS), with duration based hysteresis (Eq. (10), (11)) identical on both tiers. Because the datasets, class balancing, and protocols differ, the accuracies are not directly comparable. This work trades a slightly lower headline accuracy for a class-balanced, cross-tier, reliability-instrumented evaluation that the compared single-platform systems do not provide.

Several limitations qualify these results. First, the real time evaluation is not yet fully statistically mature: while the offline classification metrics () carry Monte Carlo 95% confidence intervals and the on device Raspberry Pi figures are now validated across five participants and seven capture conditions with scripted, frame level ground truth (Table 5), the desktop real time figures remain single, self directed point estimates, the Pi accuracies are reported as means without per condition confidence intervals, and behaviour under sustained thermal or CPU load beyond a session's duration is uncharacterised so differences between formats and conditions in the real time tables should be read as descriptive rather than statistically confirmed. Second, the merged dataset's provenance is heterogeneous, and its demographic and ethnic composition is uncontrolled, which limits claims about generalisation across populations; subject identity, demographic labels, and capture device metadata are not provided by the source corpus and are therefore unknown rather than merely unreported. Third, training time data augmentation was added specifically to target lighting/pose robustness, but it measurably reduced clean held-out accuracy (0.94-1.74 percentage points in a controlled ablation). Fourth, generalisation cannot be fully guaranteed at the subject level: identifiers exist for only part of the eye branch, and none of the mouth branch, and two of the five identifiable eye test subjects also appear in training (Section II A), so fully subject-disjoint partitioning cannot be assured and the held out accuracies may be optimistic. Fifth, the real-time protocol used scripted, cue driven drowsiness rather than naturally occurring drowsiness and rests on only five participants with no per participant confidence intervals, so its accuracies are feasibility evidence rather than population estimates. Sixth, the system has undergone no clinical or occupational validation (e.g., against polysomnography or with on-shift workers), so its effectiveness in a real safety-critical setting remains unverified. These points motivate the independent participant, genuine drowsiness, and infrared/night validation prioritised in Section V.

Taken together, the results indicate that one conceptual drowsiness detection design can be mapped onto markedly different hardware budgets without redesigning the underlying classifiers, and that a quantised dual MobileNetV2 monitor is lightweight enough to make continuous operation on constrained edge hardware plausible, although this has not yet been verified under sustained, real-world load. Because the same eye and mouth cue logic applies outside driving, the approach is directly relevant to occupational drowsiness monitoring in shift-based and assembly line settings [2], [4], [3].

V. Conclusion

This study developed and evaluated a vision-based drowsiness detection system that behaves consistently across a full-power desktop and a resource constrained edge device, using two independent MobileNetV2 classifiers (eye and mouth state) and a shared offline-first, retry capable event architecture. On a class-balanced test set the eye branch reached 94.09%/94.62% and the mouth branch 95.20%/95.76% accuracy (H5/TFLite), both with ROC AUC ≥ 0.985. TFLite conversion cut model size by 73.4% (8.99 to 2.39 MB per branch) and single-frame desktop latency roughly thirteen fold (≈26 to 2 ms). On the Raspberry Pi 5, an on device study across five participants and seven capture conditions sustained ≈38.8 FPS at a near 100% face detection rate, with real-time event detection accuracy degrading gracefully from 91.7% (normal) to 79.2% (nighttime low light). One conceptual detection design can thus be mapped onto markedly different hardware budgets now, including identical duration-based eye closure alert timing on both tiers, without redesigning the underlying classifiers. Priorities are validation with independent participants and genuinely (rather than scripted) drowsy users, evaluation under infrared/night time imaging, and a formal sensitivity analysis to optimise the temporal decision thresholds (θ1, θ2, and the yawn window) against annotated video.

Acknowledgment

The authors would like to thank the five participants who volunteered for the real-time evaluation sessions on the Raspberry Pi 5 platform.

Funding

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

The authors received no financial support for the research, authorship, and/or publication of this article.

Author Contributions

Rafi'e conceived and designed the study, developed the software, trained and evaluated the models, performed to methodological validation and reviewed the manuscript. All authors read and approved the final manuscript and agreed to be accountable for all aspects of the work.

Data Availability

The eye-state and mouth-state image datasets used to train and evaluate the classifiers are derived from the publicly available Drowsiness Dataset compiled by Hoang Tung on Kaggle [33] (https://www.kaggle.com/datasets/hoangtung719/drowsiness-dataset) . The per-frame session logs and scripted ground truth from the Raspberry Pi 5 real-time evaluation are available from the corresponding author upon reasonable request.

Code Availability

The source code for data preprocessing, model training, the desktop and Raspberry Pi applications, and evaluation scripts is publicly available at the following GitHub repository: https://github.com/Rafie93/fatigue-mitigation-system. The corresponding author, Rafi'e (e-mail: rafiekom@ulm.ac.id), can be contacted for further information regarding the code.

Ethical Approval

Ethical approval for the real-time evaluation involving human participants was not obtained prior to the study. The evaluation involved voluntary adult participants, and no medical intervention, clinical procedure, biological sample collection, or personally identifiable information was involved. All participants provided informed consent prior to participation.

Consent To Participate

Informed consent was obtained from all participants prior to their involvement in the real-time evaluation sessions. Participation was voluntary, and participants were informed of their right to withdraw from the study at any time without consequence.

Consent for Publication

The manuscript and its figures contain no personally identifiable images of participants; all reported real-time evaluation data are anonymised and aggregated.

Competing Interests

The authors declare that they have no financial or non-financial competing interests that could have influenced the conduct, analysis, interpretation, or reporting of this study.

References

[1] T. Abe, "PERCLOS-based technologies for detecting drowsiness: current evidence and future directions," *SLEEP Advances*, vol. 4, no. 1, Jan. 2023, doi: 10.1093/sleepadvances/zpad006.

[2] K. Papoutsakis *et al.*, "Detection of Physical Strain and Fatigue in Industrial Environments Using Visual and Non-Visual Low-Cost Sensors," *Technologies (Basel).*, vol. 10, no. 2, p. 42, Mar. 2022, doi: 10.3390/technologies10020042.

[3] F. Zhang, Z. Yang, J. Ning, and Z. Wu, "Aligning Computer Vision with Expert Assessment: An Adaptive Hybrid Framework for Real-Time Fatigue Assessment in Smart Manufacturing," *Sensors*, vol. 26, no. 2, p. 378, Jan. 2026, doi: 10.3390/s26020378.

[4] A. S. Moreira and S. R. de Lucca, "Physical and mental fatigue in shift work and mitigation strategies: an integrative review," *Revista Brasileira de Medicina do Trabalho*, vol. 22, no. 04, pp. 01–11, 2024, doi: 10.47626/1679-4435-2024-1267.

[5] T. Fonseca and S. Ferreira, "Drowsiness Detection in Drivers: A Systematic Review of Deep Learning-Based Models," *Applied Sciences*, vol. 15, no. 16, p. 9018, Aug. 2025, doi: 10.3390/app15169018.

[6] F. Majeed, U. Shafique, M. Safran, S. Alfarhood, and I. Ashraf, "Detection of Drowsiness among Drivers Using Novel Deep Convolutional Neural Network Model," *Sensors*, vol. 23, no. 21, p. 8741, Oct. 2023, doi: 10.3390/s23218741.

[7] F. Makhmudov, D. Turimov, M. Xamidov, F. Nazarov, and Y.-I. Cho, "Real-Time Fatigue Detection Algorithms Using Machine Learning for Yawning and Eye State," *Sensors*, vol. 24, no. 23, p. 7810, Dec. 2024, doi: 10.3390/s24237810.

[8] T. S. Delwar *et al.*, "AI- and Deep Learning-Powered Driver Drowsiness Detection Method Using Facial Analysis," *Applied Sciences*, vol. 15, no. 3, p. 1102, Jan. 2025, doi: 10.3390/app15031102.

[9] M. Iman, H. R. Arabnia, and K. Rasheed, "A Review of Deep Transfer Learning and Recent Advancements," *Technologies (Basel).*, vol. 11, no. 2, p. 40, Mar. 2023, doi: 10.3390/technologies11020040.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF*

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

Conference on Computer Vision and Pattern Recognition, IEEE, Jun. 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.

[11] S. Zu, Y. Jin, and Y. Li, "Generalwise Separable Convolution for Mobile Vision Applications," in *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, Dec. 2022, pp. 1074–1081. doi: 10.1109/SSCI51031.2022.10022267.

[12] P. K. A. Vasu, J. Gabriel, J. Zhu, O. Tuzel, and A. Ranjan, "MobileOne: An Improved One millisecond Mobile Backbone," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2023, pp. 7907–7917. doi: 10.1109/CVPR52729.2023.00764.

[13] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2022, pp. 11966–11976. doi: 10.1109/CVPR52688.2022.01167.

[14] P. K. Anasosalu Vasu, J. Gabriel, J. Zhu, O. Tuzel, and A. Ranjan, "FastViT: A Fast Hybrid Vision Transformer using Structural Reparameterization," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2023, pp. 5762–5772. doi: 10.1109/ICCV51070.2023.00532.

[15] K. Kansal, T. B. Chandra, and A. Singh, "ResNet-50 vs. EfficientNet-B0: Multi-Centric Classification of Various Lung Abnormalities Using Deep Learning," *Procedia Comput. Sci.*, vol. 235, pp. 70–80, 2024, doi: 10.1016/j.procs.2024.04.007.

[16] Y. Gulzar, "Fruit Image Classification Model Based on MobileNetV2 with Deep Transfer Learning Technique," *Sustainability*, vol. 15, no. 3, p. 1906, Jan. 2023, doi: 10.3390/su15031906.

[17] M. Akay *et al.*, "Deep Learning Classification of Systemic Sclerosis Skin Using the MobileNetV2 Model," *IEEE Open J. Eng. Med. Biol.*, vol. 2, pp. 104–110, 2021, doi: 10.1109/OJEMB.2021.3066097.

[18] T. Öznacar and N. Varol Kayapunar, "Advanced skin cancer prediction with medical image data using MobileNetV2 deep learning and optimized techniques," *Sci. Rep.*, vol. 15, no. 1, p. 28962, Aug. 2025, doi: 10.1038/s41598-025-14963-4.

[19] Nirupama and Virupakshappa, "MobileNet-V2: An Enhanced Skin Disease Classification by Attention and Multi-Scale Features," *Journal of Imaging Informatics in Medicine*, vol. 38, no. 3, pp. 1734–1754, Oct. 2024, doi: 10.1007/s10278-024-01271-y.

[20] N. Tanwar and A. V. Turukmane, "Modified MobileNetV2 transfer learning model to detect road potholes," *PeerJ Comput. Sci.*, vol. 11, p. e2519, Jan. 2025, doi: 10.7717/peerj-cs.2519.

[21] A. S. Mohammad, T. G. Jarullah, M. T. S. Al-Kaltakchi, J. Alshehabi Al-Ani, and S. Dey, "IoT-MFaceNet: Internet-of-Things-Based Face Recognition Using MobileNetV2 and FaceNet Deep-Learning Implementations on a Raspberry Pi-400," *Journal of Low Power Electronics and Applications*, vol. 14, no. 3, p. 46, Sep. 2024, doi: 10.3390/jlpea14030046.

[22] Y. Yunidar *et al.*, "CNN-Based Facial Image Analysis for Pediatric Down Syndrome Classification," *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 8, no. 2, pp. 696–711, Apr. 2026, doi: 10.35882/jeeemi.v8i2.1523.

[23] D. D. Ul-Haq, Y. Yunidar, M. Melinda, N. Basir, and R. Rosmawinda, "Data Splitting Strategies for Down Syndrome Facial Classification: A Comparative Study Using EfficientNet-B0 and MobileNetV2," *Indonesian Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 8, no. 3, Jun. 2026, doi: 10.35882/ijeeemi.v8i3.338.

[24] P. Viola and M. J. Jones, "Robust Real-Time Face Detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, May 2004, doi: 10.1023/B:VISI.0000013087.49260.fb.

[25] X. Zhao, X. Liang, C. Zhao, M. Tang, and J. Wang, "Real-Time Multi-Scale Face Detector on Embedded Devices," *Sensors*, vol. 19, no. 9, p. 2158, May 2019, doi: 10.3390/s19092158.

[26] C. Dewi, R.-C. Chen, C.-W. Chang, S.-H. Wu, X. Jiang, and H. Yu, "Eye Aspect Ratio for Real-Time Drowsiness Detection to Improve Driver Safety," *Electronics (Basel).*, vol. 11, no. 19, p. 3183, Oct. 2022, doi: 10.3390/electronics11193183.

[27] R. K. Peddarapu, B. Likhita, D. Monika, S. P. Paruchuru, and S. L. Kompella, "Raspberry Pi-Based Driver Drowsiness Detection," in *2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT)*, IEEE, Feb. 2024, pp. 864–869. doi: 10.1109/IC2PCT60090.2024.10486677.

[28] A. Alvarez Oviedo, J. F. Mamani Villanueva, G. A. Echaiz Espinoza, J. M. M. Villanueva, A. O. Salazar, and E. R. L. Villarreal, "Design of a System for Driver Drowsiness Detection and Seat Belt Monitoring Using Raspberry Pi 4 and Arduino Nano," *Designs (Basel).*, vol. 9, no. 1, p. 11, Jan. 2025, doi: 10.3390/designs9010011.

[29] S. Liawatimena and N. Isworo, "Annotated drowsiness detection dataset captured using Raspberry Pi 5," *Data Brief*, vol. 63, p. 112211, Dec. 2025, doi: 10.1016/j.dib.2025.112211.

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

[30] Y.-A. Daraghmi, E. Y. Daraghmi, R. Daraghma, H. Fouchal, and M. Ayaida, "Edge–Fog–Cloud Computing Hierarchy for Improving Performance and Security of NB-IoT-Based Health Monitoring Systems," *Sensors*, vol. 22, no. 22, p. 8646, Nov. 2022, doi: 10.3390/s22228646.

[31] D. Van Anh and V.-H. Nguyen, "Leveraging Priority Queueing in IoT-Edge-Fog-Cloud Infrastructures for Efficient Healthcare Monitoring," *IEEE Access*, vol. 13, pp. 80461–80477, 2025, doi: 10.1109/ACCESS.2025.3565679.

[32] S. Somvanshi *et al.*, "From Tiny Machine Learning to Tiny Deep Learning: A Survey," *ACM Comput. Surv.*, vol. 58, no. 7, pp. 1–33, May 2026, doi: 10.1145/3776588.

[33] Hoang Tung, "Drowsiness Dataset," Kaggle. Accessed: Mar. 29, 2026. [Online]. Available: https://www.kaggle.com/datasets/hoangtung719/drowsiness-dataset

[34] M. Xu, S. Yoon, A. Fuentes, and D. S. Park, "A Comprehensive Survey of Image Augmentation Techniques for Deep Learning," *Pattern Recognit.*, vol. 137, p. 109347, May 2023, doi: 10.1016/j.patcog.2023.109347.

[35] A. Barakat and P. Bianchi, "Convergence and Dynamical Behavior of the ADAM Algorithm for Nonconvex Stochastic Optimization," *SIAM Journal on Optimization*, vol. 31, no. 1, pp. 244–274, Jan. 2021, doi: 10.1137/19M1263443.

[36] M. Reyad, A. M. Sarhan, and M. Arafa, "A modified Adam algorithm for deep neural network optimization," *Neural Comput. Appl.*, vol. 35, no. 23, pp. 17095–17112, Aug. 2023, doi: 10.1007/s00521-023-08568-z.

[37] C. Li, K. Liu, and S. Liu, "A Survey of Loss Functions in Deep Learning," *Mathematics*, vol. 13, no. 15, p. 2417, Jul. 2025, doi: 10.3390/math13152417.

[38] J. Chen, M. Yan, F. Zhu, J. Xu, H. Li, and X. Sun, "Fatigue Driving Detection Method Based on Combination of BP Neural Network and Time Cumulative Effect," *Sensors*, vol. 22, no. 13, p. 4717, Jun. 2022, doi: 10.3390/s22134717.

[39] E. Essel, F. Lacy, F. Albalooshi, W. Elmedany, and Y. Ismail, "Drowsiness Detection in Drivers Using Facial Feature Analysis," *Applied Sciences*, vol. 15, no. 1, p. 20, Dec. 2024, doi: 10.3390/app15010020.

[40] A. Malafeev, A. Hertig-Godeschalk, D. R. Schreier, J. Skorucak, J. Mathis, and P. Achermann, "Automatic Detection of Microsleep Episodes With Deep Learning," *Front. Neurosci.*, vol. 15, Mar. 2021, doi: 10.3389/fnins.2021.564098.

[41] J. Skorucak, A. Hertig-Godeschalk, P. Achermann, J. Mathis, and D. R. Schreier, "Automatically Detected Microsleep Episodes in the Fitness-to-Drive Assessment," *Front. Neurosci.*, vol. 14, Jan. 2020, doi: 10.3389/fnins.2020.00008.

[42] R. Schleicher, N. Galley, S. Briest, and L. Galley, "Blinks and saccades as indicators of fatigue in sleepiness warnings: looking tired?," *Ergonomics*, vol. 51, no. 7, pp. 982–1010, Jul. 2008, doi: 10.1080/00140130701817062.

[43] S. Abtahi, M. Omidyeganeh, S. Shirmohammadi, and B. Hariri, "YawDD," in *Proceedings of the 5th ACM Multimedia Systems Conference*, New York, NY, USA: ACM, Mar. 2014, pp. 24–28. doi: 10.1145/2557642.2563678.

[44] A. G. Guggisberg, J. Mathis, U. S. Herrmann, and C. W. Hess, "The functional relationship between yawning and vigilance," *Behavioural Brain Research*, vol. 179, no. 1, pp. 159–166, Apr. 2007, doi: 10.1016/j.bbr.2007.01.027.

[45] S. Sathyanarayanan, "Confusion Matrix-Based Performance Evaluation Metrics," *African Journal of Biomedical Research*, pp. 4023–4031, Nov. 2024, doi: 10.53555/AJBR.v27i4S.4345.

[46] W. Chen, K. Yang, Z. Yu, Y. Shi, and C. L. P. Chen, "A survey on imbalanced learning: latest research, applications and future directions," *Artif. Intell. Rev.*, vol. 57, no. 6, p. 137, May 2024, doi: 10.1007/s10462-024-10759-6.

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371

Author Biography

Rafi'e earned his Bachelor's degree in Informatics Engineering from Universitas Islam Kalimantan Muhammad Arsyad Al Banjari, Banjarmasin, Indonesia, and a Master's degree in Computer Science from Universitas Budi Luhur (UBL), Jakarta Indonesia. His area of expertise is Applied Computational Engineering, with research interests in Artificial Intelligence, Machine Learning, Computer Vision, and Edge AI. His research focuses on the development of intelligent computational models for detection, classification, prediction, and real-time data processing.

Dr. Ichsan Rizany, S.Kep., Ns., M.Kep is a lecturer in the Nursing Study Program, Faculty of Medicine and Health Sciences, Lambung Mangkurat University, Indonesia. His expertise includes nursing management, patient safety, healthy sleep, fatigue mitigation interventions, nurse scheduling system development, nursing informatics, and quantitative data analysis. His research focuses on nursing management, patient safety, nurse fatigue, shift scheduling, and digital health interventions to improve healthcare quality. He has actively published in national and international journals and is committed to advancing evidence-based nursing practice through research, education, and community engagement.

Dodon Turianto Nugrahadi is a lecturer in the Department of Computer Science at Lambung Mangkurat University, specializing in Data Science and Computer Networking. He earned his bachelor's degree in Informatics Engineering from Petra University, Surabaya, in 2004, and later pursued a master's degree in Information Engineering at Gajah Mada University, Yogyakarta, in 2009. Currently, his research interests encompass network studies, data science, the Internet of Things (IoT), and network Quality of Service (QoS). His work contributes to advancements in technology and academic knowledge, fostering innovation and excellence in his field.

Dwi Kartini earned her Bachelor's and Master's degrees in Computer Science from the Faculty of Computer Science at Putra Indonesia "YPTK" in Padang, Indonesia. She is also a lecturer in the Department of Computer Science, where she teaches various subjects including linear algebra, discrete mathematics, research methods, and others. Her research interests focus on the applications of Artificial Intelligence and Data Mining. Currently, she serves as an assistant professor in the Department of Computer Science, Faculty of Mathematics and Natural Sciences at Lambung Mangkurat University in Banjarbaru, Indonesia. She is the head of the Computer Science study program. She can be contacted atdwikartini@ulm.ac.id. ORCID ID: 0000 0002 7382 5084.

Muhammad Itqan Mazdadi is a lecturer in the Department of Computer Science, Lambung Mangkurat University. His research interest is centered on Data Science and Computer Networking. Before becoming a lecturer, he completed his undergraduate program in the Computer Science Department at Lambung Mangkurat University in 2013. He then completed his master's degree from the Department of Informatics at Islamic University of Indonesia, Yogyakarta. Currently, he serves as the Secretary of the Computer Science Department at Lambung Mangkurat University. Email: mazdadi@ulm.ac.id. ORCID ID: 0000 0002 8710 4616.

Triando Hamonangan Saragih, currently holding the position of a lecturer within the Department of Computer Science at Lambung Mangkurat University, is heavily immersed in the realm of academia, with a profound focus on the multifaceted domain of Data Science. His academic pursuits commenced with the successful completion of his bachelor's degree in Informatics at the esteemed Brawijaya University, located in the vibrant city of Malang, back in the year 2016. Building upon this foundational achievement, he proceeded to further enhance his scholarly credentials by enrolling in a master's program in Computer Science at Brawijaya University, Malang, culminating in the conferral of his advanced degree in 2018. The research field he is involved in is Data Science. Email: triando.saragih@ulm.ac.id. ORCID ID: 0000 0003 4346 332.

Manuscript received 3 July 2026; Revised 20 July 2026; Accepted 15 August 2026; Available online 31 August 2026
DOI: https://doi.org/10.35882/ijeeemi.v8i3.371