ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

Comparative Analysis of Machine Learning and Deep Learning Models for Real-Time Driver Drowsiness Detection.

Charles Roland Haruna [1], Maame Gyamfua Asante-Mensah [2], Kwadwo Sarbeng-Baafi [3], Sandro Kwame Amofa [4]

[1,2,3,4]Department of Computer Science and Information Technology, University of Cape Coast, Cape Coast, Ghana

DOI: https://doi.org/10.51583/IJLTEMAS.2026.150600239

Received: 21 July 2026; Accepted: 26 July 2026; Published: 01 August 2026

ABSTRACT

Drowsy driving is among the major factors leading to accidents on roads, especially in scenarios where long hours of traveling or working are involved. The current paper performs a comparative analysis between traditional machine learning (ML) techniques and lightweight deep learning (DL) methods concerning driver drowsiness detection using the YawDD Dataset (a yawning detection dataset). This Dataset consists of 322 videos from the mirror (rear-view mirror camera position) subset and 29 videos from its Dash (dashboard) subset, recorded from 107 drivers of diverse ages, gender and ethnicities. The videos are presented in various levels of illumination and facial occlusions (glasses and sunglasses) as well as poses. Frames were extracted from the videos at regular intervals resulting in thousands of labelled images.

This study evaluates two types of algorithms based on their performance through several parameters such as accuracy, precision, recall, and F1 score while simultaneously considering the efficiency of computation measured by inference time, CPU usage, memory, and frames per second (FPS).

According to the findings, while DL models like the convolutional neural network based on EfficientNet and TinyCNN offer better classification performance with accuracy levels surpassing 93%, these models exhibit inferior inference rates. Consequently, these DL models cannot be employed in real-time situations since they require hardware accelerators. On the other hand, traditional ML models, mainly the combination of Random Forest and Support Vector Machine, offer a good balance between accuracy and efficiency. Logistic regression provides the best processing rate but with limited accuracy.

Conclusively, the results imply that traditional ML models are still applicable for real-time tasks in resource-constrained environments compared to DL models. However, DL models are recommended for implementations where hardware accelerators are available.

Keywords: Driver Drowsiness Detection, Machine Learning, Deep Learning, Real-Time Systems, Low-Power Devices, Computer Vision.

INTRODUCTION

The Critical Nature of Driver Drowsiness Detection.

Driver drowsiness constitutes one of the most insidious and lethal hazards present on contemporary roadways. Fatigue compromises an individual's alertness, reaction times, and decision-making capabilities, thereby contributing to a considerable percentage of accidents globally, some estimates indicate that as much as 10–20% of serious collisions may be associated with fatigued drivers (Moradi et al., 2019; SSATP, 2025). In nations such as Ghana, where social and familial travel is prevalent on weekends, the associated risk may be exacerbated by extended working hours and travel times that further elevate fatigue levels (Blom et al., 2025; Ibrahim, 2022; Moradi et al., 2019; SSATP, 2025). Nonetheless, in spite of this situation, initiatives aimed at the active detection and alleviation of drowsiness appear to fall short compared to public awareness efforts. An example of this is

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

the "Don't drive tired" initiative that a number of groups and companies start every year; unfortunately, there aren't many practical systems in our vehicles or on our roads to actually stop people from driving drowsy. The recent spike in road accidents and deaths in Ghana shows just how desperately we need better safety solutions (Ibrahim, 2022; National Road Safety Authority, 2025)

The Technical Challenge and Research Gap.

Detecting driver drowsiness in real time is a challenging problem, especially on low-power devices such as those embedded in vehicles, low-end laptops or mobile platforms. The core problem is striking a balance between detection accuracy and the limited computational resources at one's disposal. Deep learning (DL) models, whose performance is often more accurate, are normally computationally expensive; meanwhile, classical machine learning (ML) algorithms are efficient, though at the cost of reliability for some predictions (Deepika et al., 2025). Characteristically, the literature thus far overwhelmingly employs narrow concentration, either exploring DL strategies or scrutinising ML-dependent strategies in isolation, without drawing overall comparisons of their respective compromises within the context of Figures-of-Merit (FOM)-constrained surroundings (Deepika et al., 2025).

Aims and Objectives.

This study aims to address the existing gap by evaluating traditional machine learning (ML) models in comparison to efficient deep learning (DL) frameworks utilising the YawDD dataset, which serves as a conventional benchmark for drowsiness detection through video frames of drivers identified as either alert or drowsy. (Abtahi et al., 2014). The objective is to ascertain which methodologies could feasibly facilitate real-time drowsiness detection on devices with limited power, scrutinising not only accuracy but also the speed of inference and the consumption of resources. Through this investigation, we aspire to offer more explicit recommendations for the deployment of functional fatigue detection systems that require minimal computational resources while still providing prompt and dependable alerts.

Related Works

Driver Drowsiness Systems can be classified according to the data they utilise: physiological, vehicular, and behavioural (Arakawa, 2021). Physiological systems monitor a driver's biological signals. These systems, while accurate, are highly intrusive and not practical for daily use (Jasimet al., 2022). Vehicular systems diagnose the driver's behaviour on the road while driving. It studies steering patterns or lane departures and provides a warning when symptoms are already evident (Arakawa, 2021). The third classification, behavioural, consists of using vision-based behavioural systems to study the driver's facial features and eye movements using a camera. This makes it a non-intrusive, proactive and relatively low-cost alternative for real-time monitoring of drivers (Abtahi et al., 2014; Essahraui et al., 2025). This literature focuses exclusively on this area and proceeds to analyse traditional ML techniques, followed by modern DL techniques, and then a final assessment of comparative studies on low-power embedded platforms.

Traditional Machine Learning Approaches for Drowsiness Detection.

Handcrafted Features: The Foundation of Early Systems.

Initial drowsiness-detection systems utilizing visual inputs followed a two-phased approach. First, a facial detector and a predictor of facial landmarks were used to determine key points of the face (Abtahi et al., 2014). Then, the landmark predictor was utilized in order to extract custom features, such as the Eye Aspect Ratio (EAR), used to estimate the extent of eye openness; the Mouth Aspect Ratio (MAR), used to estimate mouth sizes in order to detect yawning (Srinivas et al., n.d.); and the Percentage of Eye Closure Over Time (PERCLOS), referring to the percentage of time elapsed by the eyes being more than 80% shut (Abe, 2023; Dinges et al., 1998; Wierwille et al., 1994). Even though these features have been time-verified and interpretable, their usefulness is strongly tied to the first feature extraction stage and the use of fixed thresholds a priory. This

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

"one-size-fits-all" strategy limits their accuracy under a wide range of scenarios, thus making these features prone to real-life issues, like head motion and inter-subject variation in drowsiness expression (Abe, 2023).

The Role of Classifiers: Support Vector Machines (SVM) and Random Forest (RF).

An ML classifier, most often SVM or RF uses handcrafted feature to identify the state of the driver: alert or drowsy (Dharshini R et al., 2024; Schwarz et al., 2023; Srinivas et al., n.d.). The accuracy of these ML classifiers varies significantly, ranging from 84% to 99% for the same RF algorithm (Elidrissi et al., 2023; Khanehshenas et al., 2024). This variance is a major inconsistency that can be attributed to the lack of a uniform benchmark. Each study uses a different methodology-various features sets and thresholds, dataset, even with the same dataset, they may extract different images and label it differently. Due to this an “apples to apples” comparison is impossible, making it difficult to establish the true state-of-the-art for traditional machine learning approaches.

Deep Learning Paradigms: From Feature Engineering to End-to-End Learning.

Convolutional Neural Networks and Automated Feature Extraction.

DL has transformed driver drowsiness detection by automating the feature extraction process. Convolutional Neural Networks (CNNs), unlike traditional ML methods learn intricate patterns directly from the raw images (Kaundal et al., 2025). This enables it to bypass the limitations of handcrafted features like noise and occlusion (Jasim et al., 2022). Various architectures like VGG16, VGG19, and ResNet50 have been used, with hybrid models combining CNNs with recurrent components like Long Short-Term Memory (LSTM). These hybrid models are very effective at analyzing temporal patterns, helping to distinguish between a brief blink and a prolonged, fatigue-induced eye closure(Fonseca & Ferreira, 2025).

Lightweight Architectures for Resource-Constrained Environments.

Low power devices can not handle the constraints of large deep learning models due to their high computational requirements. This led to the development of models like MobileNet and EfficientNet with lightweight architecture fit for low power devices. They utilize techniques to reduce parameters and computational overhead, often achieving high accuracy over 95%, while outperforming larger models like VGG16. MobileNet for instance reached an accuracy of 93.06% while EffRes-DrowsyNet achieved up to 97.71%. Lighter models outperforming larger ones indicate that high accuracy in a controlled environment is no longer the challenge, rather the focus is now on deploying in the real world where resources are constrained.

Benchmark Datasets and Model Performance.

The widespread use of deep learning for drowsiness detection is largely due to the availability of public benchmark datasets like the Yawning Detection Dataset (YawDD) and the National Tsing Hua University Driver Drowsiness Detection (NTHU-DDD). While these datasets provide a common ground for evaluation, inconsistencies exist even among them. For example, some sources cite a subset of the YawDD while others mention the full dataset. The NTHU-DDD dataset is particularly valuable for its detailed annotations under various conditions, but the variety of these and other datasets, such as the MRL Eye and UTA-RLDD, still prevents truly fair, cross-study comparisons.

Evaluation of Comparative Studies on Low-Power Devices.

Whilst both DL and ML methods have been individually leveraged, detailed comparative evaluations with focus on their application within low-power, resource-constrained environments are scarce. Power consumption trade-offs are correctly quantified and inference speed (frames per second), along with cross-model portability on systems like Raspberry Pi or NVIDIA Jetson series, are rarely addressed. Common standardized benchmarking on embedded systems for drowsiness detection is also lacking to a significant degree, thus presenting a challenge for direct performance comparisons. The gap above is inherent, since practical automotive applications in the real world demand detection performance alongside low latency, low energy consumption, and hardware portability. Literature primarily addresses accuracy on desktop-class GPUs or sufficiently powered hardware at

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

the cost of neglecting low-power deployment issues in real scenarios. The solutions above are crucial for bridging laboratory study and massive on-road deployment.

METHODOLOGY

This work compares classical machine learning (ML) models and light-weight deep learning (DL) architectures for real-time drowsiness identification, with an emphasis on applicability on low-power instruments like smartphones or embedded devices. Methodology borrows from experimental workflows involving dataset pre-processing, feature extraction, model training, and performance analysis. All experiments were carried out under a Python 3.12 virtual environment in order to use TensorFLow. Main used libraries were OpenCV (for image processing), MediaPipe (for face landmark recognition), scikit-learn (for training of ML models), TensorFlow/Keras (for DL models) since they contain pre-built DL architectures, Pandas (for data manipulation and analysis), and psutil (for resource observation). Methodology places a special emphasis on computational cost, accuracy, and real-time inference, and treats low-power constraints through simulation by disabling GPU acceleration through the use of the CUDA_VISIBLE_DEVICES="" environment variables and execution under CPU only environments.

Dataset.

The Yaw Drowsiness Detection (YawDD) dataset was used. It consists of various videos taken of various drivers of various ethnicities. It has two subsets of images: Mirror (camera is installed under the front mirror of the car), Dash (camera is installed on the driver's dash). It is composed of 322 mirror videos and 29 dash videos from 107 drivers. For this research the "Mirror" subset was used, providing close-up facial views from male and female subjects under varied conditions (e.g., with/without glasses or sunglasses, varying illumination, head movements). The models were trained on the Mirror subset and then to simulate real time detection were tested on the Dash subset.

Frame Extraction and Annotation.

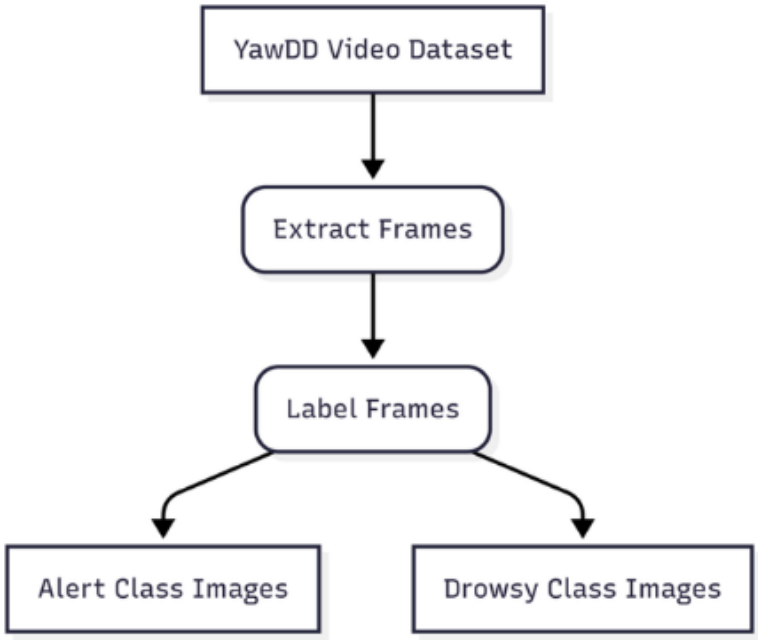

Figure 1: Frames Extraction and Classification Architecture

- Videos were processed using OpenCV's VideoCapture to extract frames at a sampling rate of every 10th frame to prevent having images that were too similar.
- Labels were derived from the filenames: videos with "Yawning" in their names were labeled as "drowsy" (class 1), "Normal" or "Talking" videos were labeled as "alert" (class 0). This approach leverages on the

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

existing structural naming conventions of the original dataset to ensure consistency and reduce manual effort.

- Extracted frames were saved in labelled directories: alert, drowsy.
- An image was resized to 128x128, 96x96, and 64x64 pixels using Pillow (PIL) to identify at which resolution that I as a human could still identify features in the image and to use that as a benchmark for model training. Images of size 128x128 were identified to be ideal.

Preprocessing.

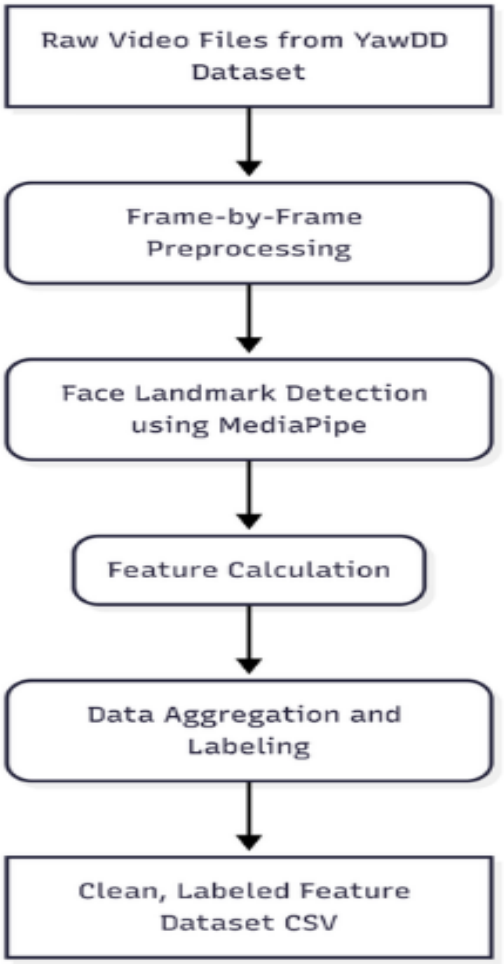

Figure 2:ML Preprocessing Architecture

For ML:

The ML pipeline did not rely on the raw image data as ML models cannot process images, they read data in 0 and 1. As a result, a feature-based approach where specific handcrafted features were extracted from the images, was used. The core processing steps were:

- Facial Landmark Detection: For each image frame, facial landmarks were detected using MediaPipe Face Mesh. This provided a set of 468 3D coordinates representing key points on the face.
- Handcrafted Feature Extraction: Using the detected landmarks, two key features were calculated: the Eye Aspect Ratio (EAR) and the Mouth Aspect Ratio (MAR). The EAR quantifies the degree of eye closure, while the MAR quantifies the degree of mouth opening. These features served as the direct input to the classifier.
- Feature Set Creation: The calculated EAR and MAR values, along with their corresponding labels, were compiled into a structured dataset. This data was then saved as a CSV file for training the machine learning models.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

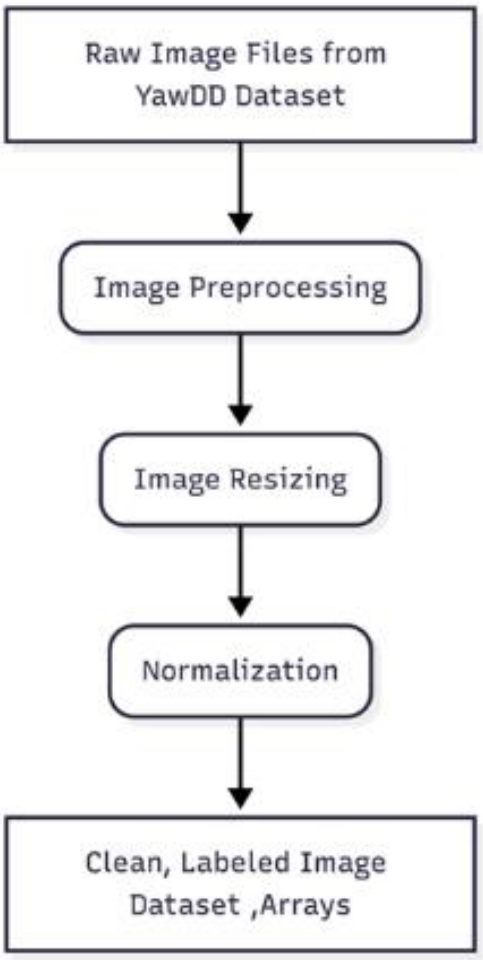

Figure 3: DL Preprocessing Architecture

For DL:

The DL pipeline utilizes a two-stage procedure: training, testing. The images were trained against by resizing them to 128x128 pixels and normalizing. As a means of preventing overfitting and enhancing generalisation, Keras's ImageDataGenerator was utilized to apply a range of real-time augmentation techniques, such as random flips, rotations and shifts.

The images were downscaled to a size of 128x128 pixels, changed to RGB color format from BGR, and normalized during testing. The testing process also involved invalid image handling for which they were substituted with zero-filled arrays to ensure batch stability and correct processable handling.

Models

ML Models:

This study employs three machine learning models for classification: Support Vector Machine (SVM), Random Forest, and Logistic Regression.

Support Vector Machine (SVM): It applies a radial basis function (RBF) kernel in finding the optimal separation between divergent classes. It is intended for handling imbalanced data by assigning weights to the classes (0: 0.7011, 1: 1.85) and provides probability estimates in the projection. A fixed random state is applied for ensuring reproducible results.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

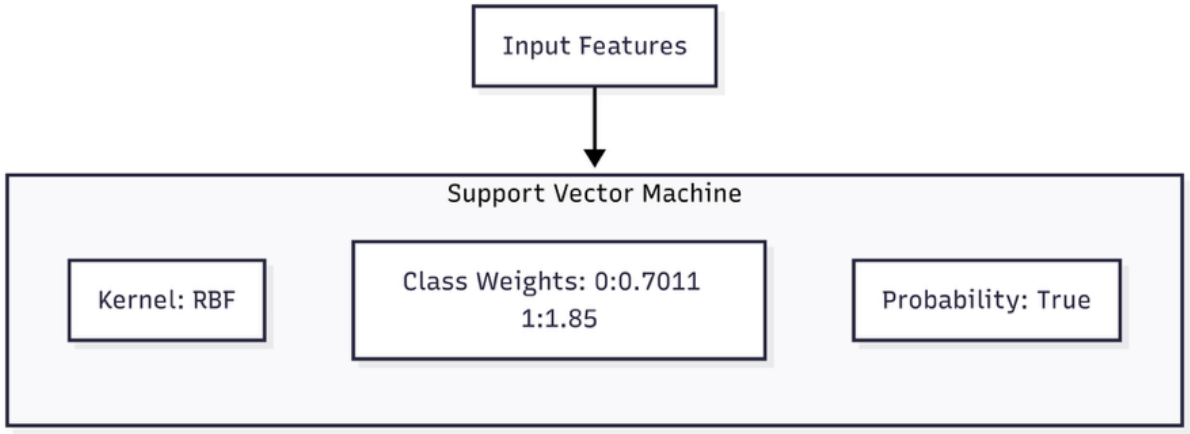

Figure 4:SVM Model Architecture.

Random Forest It predicts by voting through 100 decision trees, limiting the trees' depth up to 5 and requiring at least 2 samples at the split or a leaf node so as not to overfit. It also considers the imbalance of classes through weights (0: 0.7011, 1: 1.93) and a specified random state for reproducibility purposes.

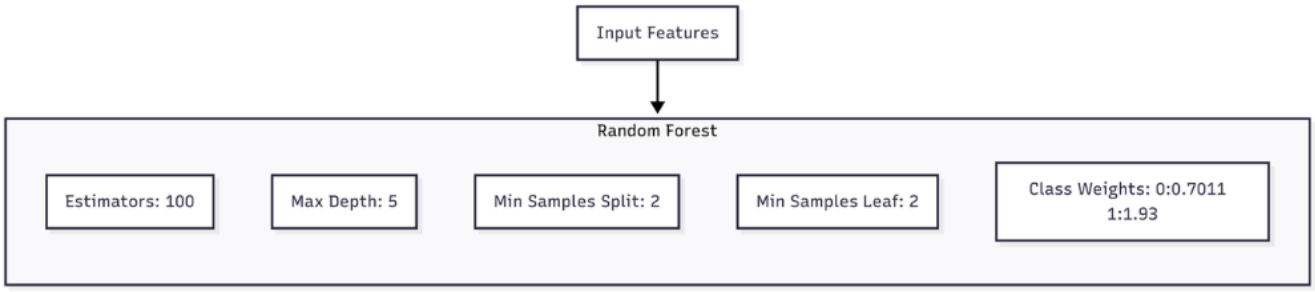

Figure 5: RF Model Architecture.

Logistic Regression: It predicts probabilities of membership in a class by a logistic function. It may iterate at most 100 times, and a regularization strength of 10 and class weights of (0: 0.7011, 1: 1.92) have been chosen in order to handle imbalance in data. It is ensured to obtain deterministic results by a fixed random state.

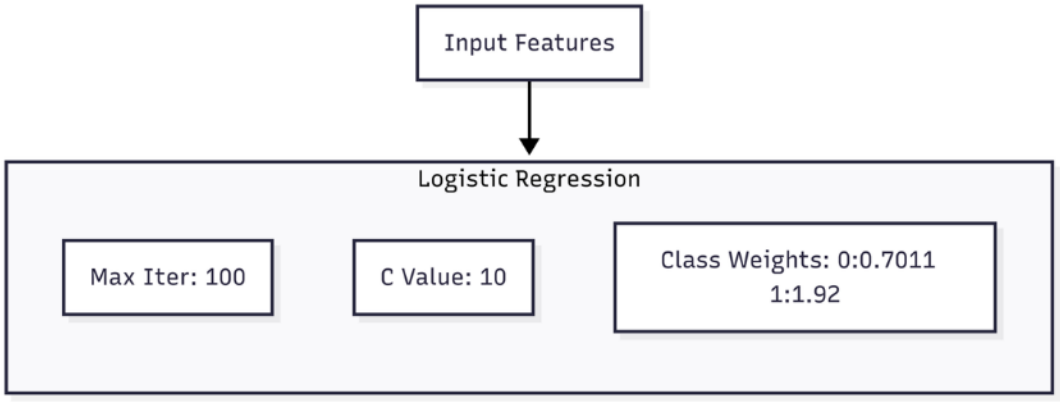

Figure 6: Logistic Regression Model Architecture.

These models are designed to classify data effectively, with adjustments for imbalanced classes and settings to ensure reliable, reproducible result.

Ensemble RF + SVM: The ensemble model proposed here applies a soft-voting method, whereby the ensemble of a Random Forest (RF) and a Support Vector Machine (SVM) classifier is used for improving drowsiness detection. Under this approach, each base learner produces probabilities of classes, which are aggregated through weighted averaging, and a larger weight is given to RF (with value 2) over SVM (with value 1) in order to put

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

more priority on recall. It ensures, specifically, that the system is optimized more toward recognizing drowsy conditions and, thus, minimizing false negatives, which is critical in safety-critical situations. RF provides strong recall performance through the ability to capture non-linear feature interactions, and SVM provides well-calibrated decision boundaries, resulting in better probability estimation overall. Final prediction is then obtained through weighted probabilities of the ensemble, resulting in a hybrid decision-making approach superior in accuracy, precision, recall, and F1-score for individually performing classifiers, and thus, it is suited for efficient real-time applications.

DL Models

The data was divided into training (70%) and temporary (30%) sets. The temporary set was subdivided equally into validation (15%) and test (15%) sets by stratified sampling for maintaining class distribution. A deterministic random seed was utilized for reproducibility

purposes. Owing to the fact that there was a class imbalance (15,728 alert and 6,709 drowsy samples), inverse of class frequencies weights, also called class weights, were calculated and utilized while training for diminishing bias towards the majority class. These weights have been hand-tuned for optimum performance on some of the models such as the TinyCNN {0:0.7011,1:1.71}.

TinyCNN: We introduced and trained, in this paper, a Convolutional Neural Network (CNN) and will, from here onwards, refer to it as the "TinyCNN." Network structure was established as a sequential stack of layers, whose initialization was done through three convolutional blocks. These contained a two-dimensional convolutional layer of 3×3 kernel and rectified linear unit (ReLU) activation, followed by a two-dimensional max pooling layer of 2×2 pool size. Filter sizes in these blocks were increased methodically (16→32→64), allowing the model to learn progressively more abstract and complex features from the input images.

Following the feature extraction blocks, the resulting output was converted into a single-dimensional vector. It was then sent into a fully connected layer, which started off with a dense layer of 64 neurons and used a ReLU activation function. A dropout layer with 0.3 rate was used in order to reduce the chance of overfitting. The structure finished off with a last dense layer using a sigmoid activation function, thus producing a single output reserved for binary classification.

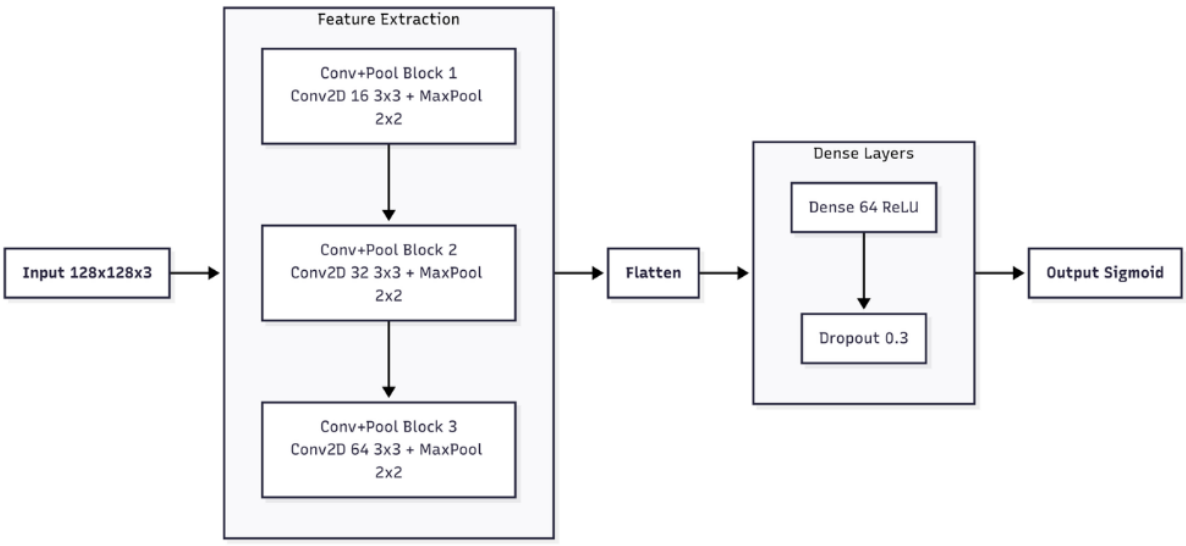

Figure 7:TinyCNN Model Architecture

MobileNetV2: We built the model on top of the strong pre-trained MobileNetV2 base, serving as the key feature extractor. Initially, the weights of the base model were frozen, meaning they were kept unchanged over the first training stage. Using this approach, called transfer learning, we were able to leverage the comprehensive, hierarchical features that the pre-trained MobileNetV2 had learned through the large ImageNet database. Above the frozen base, we added an adapted classification head. It contained a GlobalAveragePooling2D layer for

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

compacting the feature maps, a Dense layer utilizing a ReLU activation for nonlinear transformation, and a Dropout layer used for discouraging overfitting by randomly disabling a fraction of the neurons. The final layer contained a Dense layer of a single neuron with sigmoid activation, tailored specifically for the task of binary classification at hand.

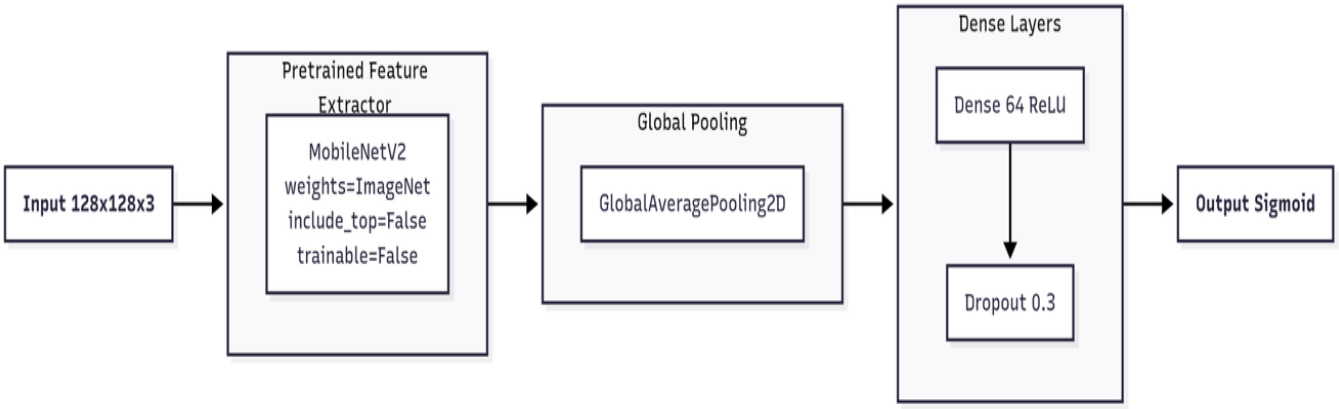

Figure 8:MobileNetV2 Model Architecture

EfficientNet Style CNN: Our model's baseline architecture for feature extraction included a sequence of convolutional layers with ReLU activation and pooling done by max pooling to efficiently downsample the feature maps. We included Batch Normalization layers after each pooling operation in order to increase stability while training and increase general performance. A key design choice was the use of depthwise separable convolutions, an approach central to EfficientNet, which significantly reduces both the number of parameters and the computational cost compared to standard convolution. Using this approach allowed us to create a deep and resilient network without paying a huge cost in terms of computations. Finally, the last feature maps were condensed using a Global Average Pooling layer.

The model's classification head was a straightforward but efficient dense stack of layers with ReLU activation. Dropout layers were used to generalize better and avoid overfitting to new, unseen data. We concluded the model structure on top of a last dense layer activated by sigmoid, giving a unit probability value pertinent to the binary task of the moment.

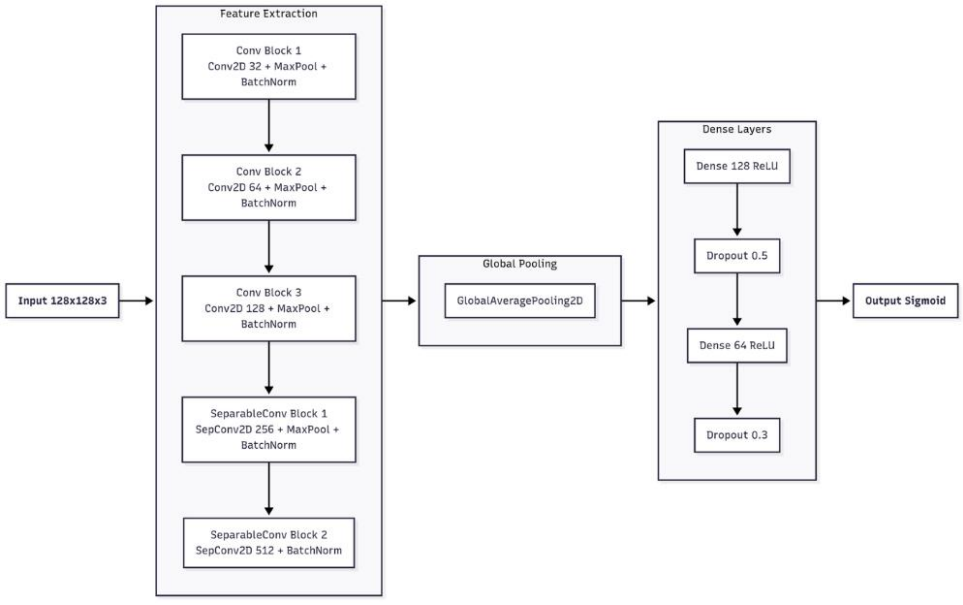

Figure 9:EfficientNet Style CNN Model Architecture

Input: Resized RGB images (128x128).

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

Training the Models

All the deep learning models were trained using the Adam optimizer, a learning rate of 0.001, batch size of 32 and up to 20 epochs with an early stopping callback (patience = 5, monitored on validation accuracy). During training, the data was augmented using random horizontal flips and its brightness was adjusted to ± 0.2 and a contrast adjustment between 0.8 and 1.2.

A stratified 70/15/15 train/validation/test split with a fixed random state of 42 was used for all models (ML and DL) to ensure reproducibility. The traditional ML models (SVM, Random Forest, Logistic Regression, RF + SVM ensemble) were all trained using scikit-learn with class weights where applicable. All experiments were performed on CPU-only (CUDA disabled) hardware to simulate real-world resource-constrained environments.

Evaluation Metrics and Testing

The performance of all models was assessed using standard classification metrics: Accuracy, Precision, Recall, and F1-score. These metrics were calculated to provide a comprehensive view of how well each model performed in classifying driver drowsiness.

In addition, we measured the computational efficiency of each model in a simulated real-time environment. This included monitoring CPU usage, RAM usage, and inference time per frame to evaluate the models' suitability for resource-constrained, low-power devices.

Experimental Setup

Hardware: A 1.4 Ghz Quad-core intel Core i5 MacBook Pro 2019, 8 GB 2133 MHz LPDDR3 memory, and Intel Iris Plus Graphics 645 (1536 MB). A machine under a 15.6.1 version of the MacOS was utilized. Low power device constraints were emulated by disabling the GPU.

All experiments were carried out under a Python 3.12.11 virtual environment under Jupyter Notebook running in Visual Studio Code. Major libraries used were TensorFlow-Keras for deep learning models, scikit-learn for machine learning classifiers, OpenCV for image processing, MediaPipe for facial landmark detection, Pandas and NumPy for data manipulation, and psutil for resource tracking. GPU acceleration was switched off to emulate low-power CPU-only environments through environment variables. Other utility libraries like joblib, os, time, Matplotlib, and Seaborn aided model saving, GPU checks, time, and visualization tasks.

RESULTS AND DISCUSSIONS

Summary of DL Model Performance during training

Table 1:Comparative Summary of DL Training Performance.

Model	Accuracy (%)	Precision	Recall	F1-Score
MobileNetV2 (Transfer Learning)	87.80	0.79	0.80	0.80
TinyCnn	93.02	0.89	0.87	0.88
EfficientNet Style CNN	93.55	0.92	0.86	0.89

As shown in the table, the EfficientNet Style CNN consistently achieved the highest metrics across almost all metrics. With the top accuracy, precision, and F1-score, it proved to be the most effective model for our classification task. While both the TinyCNN and MobileNetV2 models performed well, the Simplified EfficientNet model demonstrated a clear and measurable superiority in its ability to correctly classify both positive and negative cases, making it the preferred choice for this application.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

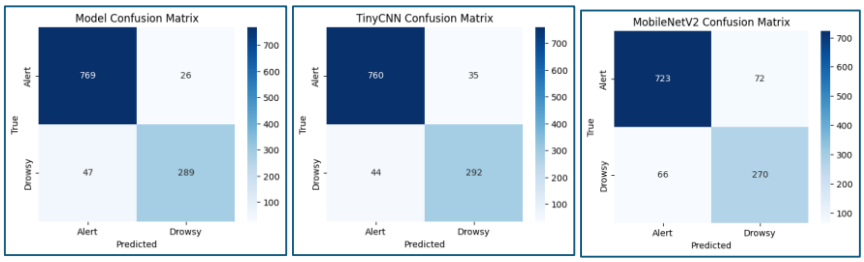

Figure 10: EfficientNet vs TinyCNN vs MobileNetV2 Models Training Confusion Matrices

The confusion matrices provide a clear visual breakdown of each model's performance on the validation set.

The MobileNetV2 model accurately identified most of the cases but had a notable number of false positives (72) and false negatives (66), indicating a tendency to misclassify both "Alert" and "Drowsy" states at a modest rate.

The TinyCNN model showed an improved ability to identify "Alert" states, with a very low number of false positives (35). It also performed well on the "Drowsy" class but had a slightly higher number of false negatives (44) than its false positives.

The Simplified EfficientNet was the top performing of the three. It had the lowest error rate of false positives (26) and false negatives (47), and is, therefore, more efficient at correctly identifying both "Alert" and "Drowsy" conditions. It is, then, the most reliable model.

The observation from the confusion matrices reveals that the Simplified EfficientNet model produced the highest accuracy with the fewest classification errors.

Summary of ML Model Performance during training

To check how well different machine learning classifiers work for detecting drowsiness, we trained three basic models, Support Vector Machine (SVM), Random Forest (RF), and Logistic Regression, and compared them to a combined model (RF + SVM). We looked at performance measures such as accuracy, precision, recall, and F1-score.

Table 2: Comparative Summary of ML Models during training.

Model	Accuracy (%)	Precision	Recall	F1-Score
SVM	61.81	0.70	0.62	0.63
Random Forest (RF)	63.54	0.70	0.64	0.65
Logistic Regression	53.56	0.61	0.54	0.56
Ensemble RF + SVM	71.06	0.71	0.71	0.71

Logistic Regression exhibited the poorest performance (53.56% accuracy, F1 = 0.56). It would suggest that the linear decision boundary it outputs cannot capture the non-linear relationships of the data.

Random Forest and SVM both did alright, and Random Forest slightly better than SVM across all metrics. It demonstrates the superiority of tree procedures utilizing ensemble methods over classifiers utilizing only margins.

The Ensemble model (RF + SVM) achieved the highest performance in all metrics, exhibiting a marked gain in accuracy (71.06%) and a compatible, balanced precision, recall, and F1-score of 0.71. It demonstrates that

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

combining the strength of Random Forest in recall and SVM's strength in probability calibration produces a superior classifier.

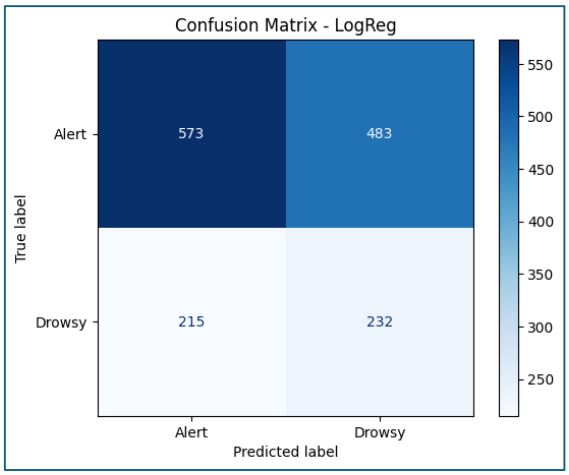

Figure 11:LogReg Training Confusion Matrix

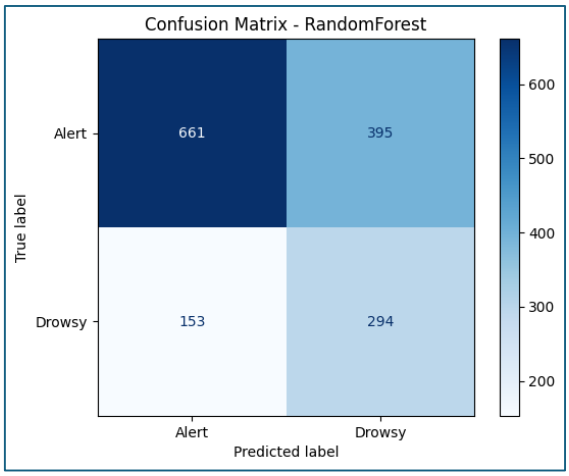

Figure 12: RF Training Confusion Matrix

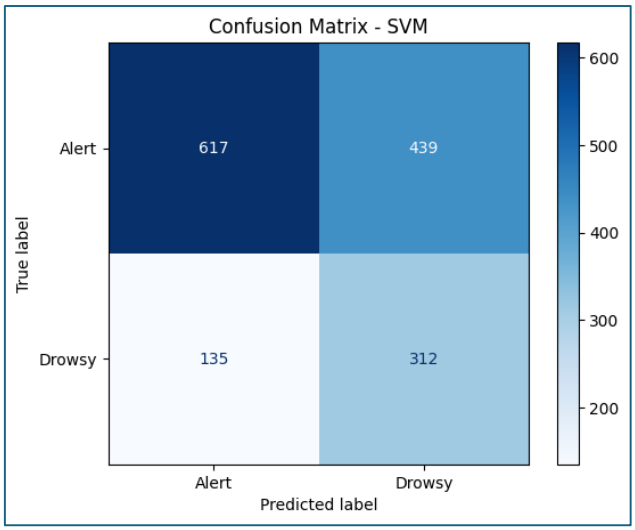

Figure 13: SVM Training Confusion Matrix

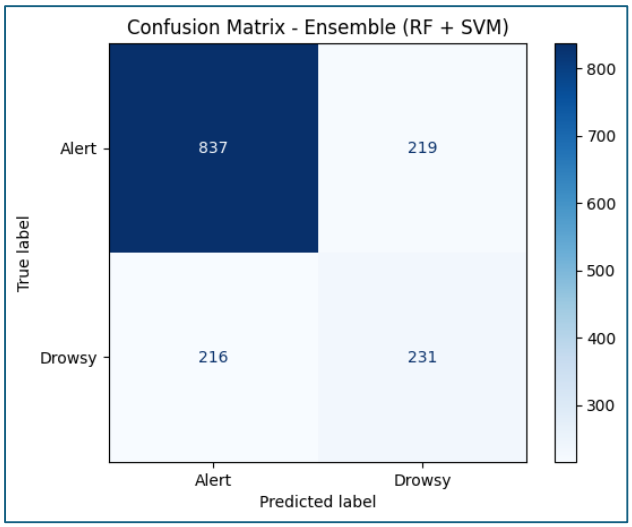

Figure 14: RF+SVM Training Confusion Matrix

The confusion matrices indicate that the Ensemble (RF + SVM) model significantly outperformed the separate models. All of the Logistic Regression, SVM, and Random Forest models committed many false positives by inappropriately flagging "Alert" states. By significantly reducing these errors, the ensemble approach demonstrated it was the most reliable and efficient classifier for our task and committed fewer total misclassifications.

The findings show that using group methods can help in detecting drowsiness because combined models can do better than single classifiers by using their different strengths. This is especially helpful for monitoring drivers in real time, where both accuracy (to lower false alarms) and sensitivity (to reduce missed detections) are very important.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

1.1.1 Comparison of ML & DL Models on Simulated Real time Inference

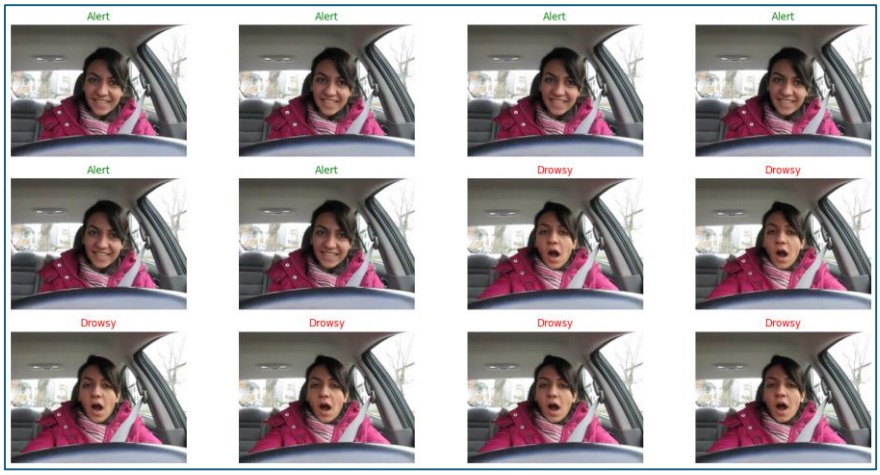

Figure 15: Sample of Images Extracted from Video.

The objective of the driver drowsiness models is usage in real-time applications, so drowsiness must be recognized while happening. All of the models were trained and evaluated for real-time recognition of driver drowsiness in the YawDD dataset. Individual frames were pulled from a tiny driving sequence and then processed consecutively, one after the other.

ML Models.

There were 2,741 frames sequentially extracted from the driving video. These frames were analyzed by FaceMesh of the MediaPipe library so as to extract geometric features, which included the left and right eyes' Eye Aspect Ratio (EAR) and the Mouth Aspect Ratio (MAR) of the face. These attributes were provided as inputs to the trained ML classifiers.

DL Models.

Table 3: Performance Analysis Table for Simulated Real Time Inference of DL and ML models

	SVM	RF	Logistic Regression	Ensemble RF+ SVM	TInyCnn	MobileNetV2	Efficientnet Style Cnn
CPU Usage (%)	21.90	19.10	7.70	16.40	6.30	13.60	7.10
RAM Usage(%)	3.90	6.10	19.70	4.90	4.90	0.10	1.60
Inference Time (ms/frame)	69.82	58.12	48.83	59.42	92.53	119.31	92.38
FPS	14.32	17.21	20.48	16.83	10.81	8.38	10.82

The performance analysis (Table 3) compared machine learning (ML) and deep learning (DL) models in terms of CPU usage, RAM usage, inference time, and achievable frames per second (FPS).

Logistic Regression was the best performing in terms of performance for lightweight monitoring since it resulted in the highest value of FPS of 20.48. Its predictability was, however, less than ensemble and tree-based algorithms.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

Random Forest (17.21 FPS) and Ensemble of RF+SVM (16.83 FPS) presented an acceptable trade-off between accuracy and speed, and hence appropriate for real-time observation purposes, though these fell short of delivering the 30 FPS constraint for real-time video analysis processing.

SVM (14.32 FPS) was, however, slower although it utilized more CPU (21.9%), hence limiting its scalability on low-power or in-embedded devices.

Of the DL models, TinyCNN (10.81 FPS) and EfficientNet-style CNN (10.82 FPS) executed at similar rates, both of which were slow for real-time use, but potentially possible for semi-offline driver monitoring systems.

MobileNetV2 (8.38 FPS) exhibited the lowest throughput, though optimized for mobile use, underlining computation bottlenecks running on CPU-only configurations.

Practical Implications

The experiments demonstrated a clear trade-off among model complexity, memory consumption, and computational efficiency around real-time identification of driver drowsiness. Logistic Regression achieved the best throughput at around 20 frames per second (FPS) and consuming a low CPU resource of 7.7%, thus making it the most suitable solution for near real-time observation in CPU-only platforms having limited resources. Random Forest and the Ensemble approach (RF + SVM) presented good performance at around 17 FPS, but at higher CPU powers of between 16% and 19%. It thus means the models are better suited for cars having moderate levels of computation.

SVM (14 FPS) exhibited relatively moderate CPU usage (~21.9%) and was unable to exceed the 30 FPS real-time threshold, exhibiting limited scalability on embedded platforms. Deep learning models exhibited mixed performance:

TinyCNN and EfficientNet exhibited ~10–11 FPS at moderate CPU usage, while MobileNetV2 lagged behind at ~8 FPS even though it was originally intended for mobile platforms. Of particular note, MobileNetV2 exhibited extremely low memory usage (0.1%), demonstrating efficient memory utilization, but at the sacrifice of its slow inference speed on CPUs. By contrast, Logistic Regression was paired with the highest memory usage (~19.7%), which would be restrictive on memory-limited platforms even though it was fast.

In general, the results suggest the traditional machine learning models (Random Forest, Ensemble, Logistic Regression) continue to be the most realistic options for CPU-reliant, in-vehicle measurement schemes, given their compromise of inference speed and moderate memory allocation. By contrast, deep learning models stand better in accuracy-focused applications and GPU/TPU-aided hardware, in which their poor speed on CPU can be offset.

Table 4:Consolidated Summary Comparison Table

Model	Accuracy (%)	Precision	Recall	F1-Score	CPU Usage (%)	RAM Usage (%)	Inference Time (ms/frame)	FPS
Logistic Regression	53.56	0.61	0.54	0.56	7.7	19.7	48.83	20.48
Random Forest	63.54	0.7	0.64	0.65	19.1	6.1	58.12	17.21
SVM	61.81	0.7	0.62	0.63	21.9	3.9	69.82	14.32
Ensemble RF + SVM	71.06	0.71	0.71	0.71	16.4	4.9	59.42	16.83
TinyCNN	93.02	0.89	0.87	0.88	6.3	4.9	92.53	10.81

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

MobileNetV2	87.8	0.79	0.8	0.8	13.6	0.1	119.31	8.38
EfficientNet Style CNN	93.55	0.92	0.86	0.89	7.1	1.6	92.38	10.82

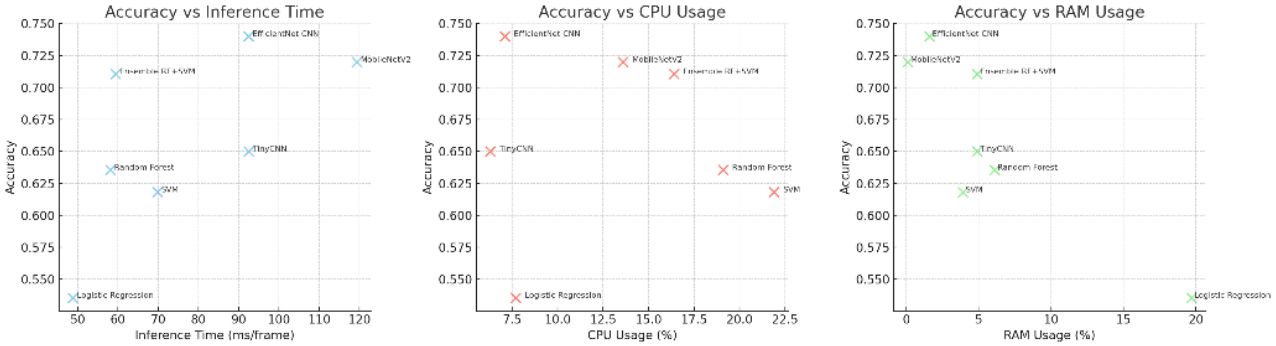

Figure 16: Three panel visualisation for Accuracy vs Inference Time, Cpu Usage and Ram Usage

Comparison of ML Models in Making Predictions on Unseen Data.

For this test we take a single video, extract 30 frames at intervals and use the various models to try and predict yawning and not yawning on those frames. The video was chosen from the Dash subset of Yawdd as a representation of Unseen data to verify the ability of the models to predict images that they were not trained on.

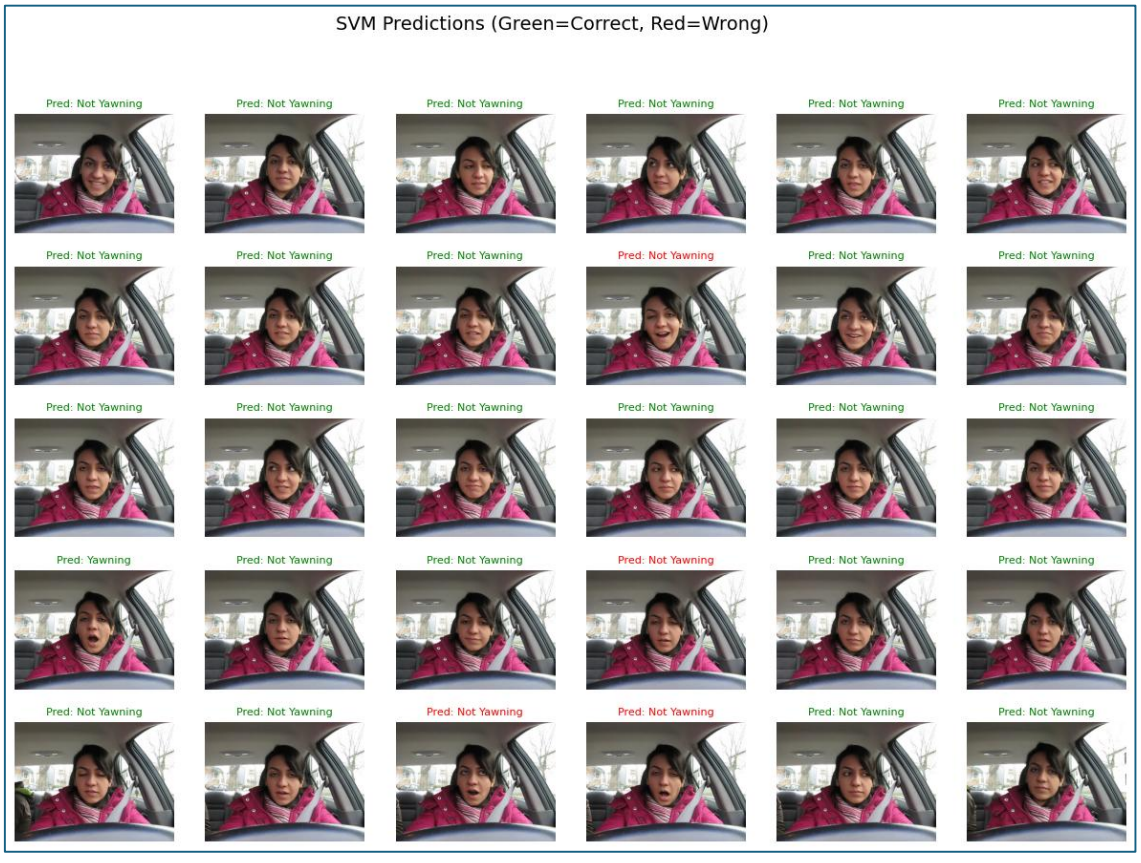

Figure 17: The 30 frames of Unseen Data

Table 5: Macro Averages of All Models tested on Unseen data

	SVM	RF	Logistic Regression	Ensemble RF + SVM	TinyCnn	MobileNetV2	Efficientnet Style Cnn
Precision	0.93	0.88	0.95	0.76	0.42	0.58	0.59

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

Recall	0.60	0.88	0.70	0.84	0.50	0.60	0.52
F1-Score	0.63	0.88	0.76	0.79	0.45	0.58	0.19

This study compared the performance of several machine learning and deep learning models for driver drowsiness detection. Using macro-averaged metrics, which provide a balanced evaluation by giving equal weight to both the "drowsy" and "not drowsy" classes, the results indicate that traditional machine learning models significantly outperformed their deep learning counterparts.

Specifically, the Ensemble RF + SVM and Logistic Regression models achieved superior scores across Precision, Recall, and F1-Score. In contrast, the deep learning models (TinyCnn, MobileNetV2, and Efficientnet Style CNN) showed notably lower performance, with their low F1-Scores suggesting a poor ability to generalize on the given dataset. This finding highlights that for a limited dataset like YawDD, simpler, handcrafted-feature-based models are more effective than complex deep learning architectures that typically require extensive data to achieve their full potential.

Table 6: Weighted Average Across All Models tested on Unseen data

	SVM	RF	Logistic Regression	Ensemble RF + SVM	TinyCnn	MobileNetV2	Efficientnet Style Cnn
Precision	0.89	0.93	0.91	0.89	0.69	0.77	0.86
Recall	0.87	0.93	0.90	0.87	0.83	0.73	0.20
F1-Score	0.83	0.93	0.88	0.87	0.76	0.75	0.11

Weighted averages on unseen data reflect significant differences between ML and DL approaches. Random Forest exhibited the highest overall performance, 0.93 in terms of precision, recall, and F1-score, confirming its stability between drowsy and alert classes. Logistic Regression (0.91, 0.90, 0.88) and Ensemble of RF+SVM (precision/recall/F1 all 0.87) offered balanced performance and, accordingly, may act as stable candidates for implementation.

SVM exhibited slightly lower overall performance (F1 = 0.83) but very good precision (0.89), again exhibiting its ability to keep false positives low at the cost of recall. Among deep learning architectures, TinyCNN and MobileNetV2 produced moderate performance (F1 = 0.76 and 0.75, respectively), though ML baselines surpassed these architectures. The EfficientNet-style CNN, though relatively good at precision (0.86), exhibited extremely poor recall (0.20) and F1 (0.11) and thus exhibited instability of predictions under unseen data.

Overall, these findings suggest conventional ML models generalize better than the tested DL architectures on this data set, and, first of all, the Random Forest, which strikes the best balance of precision, recall, and F1-score.

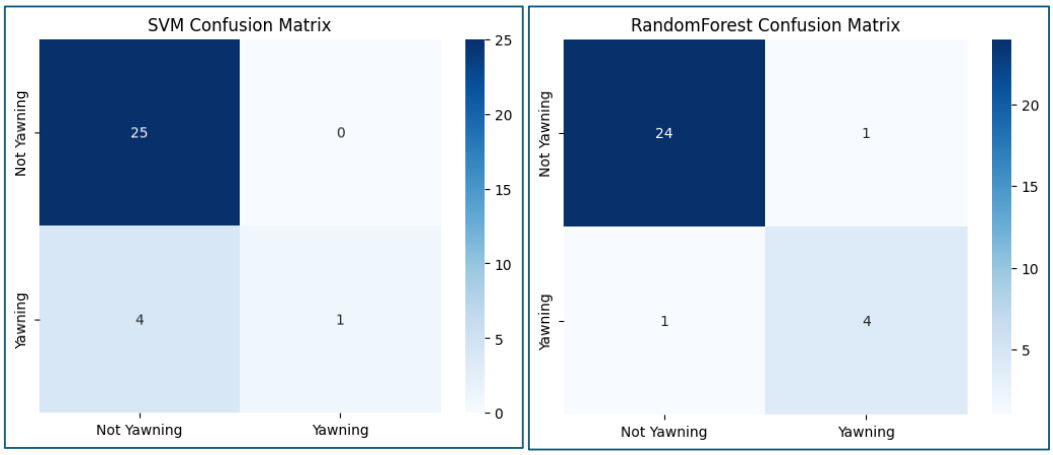

Figure 18: Confusion Matrices of SVM & RF during Unseen Data Testing

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

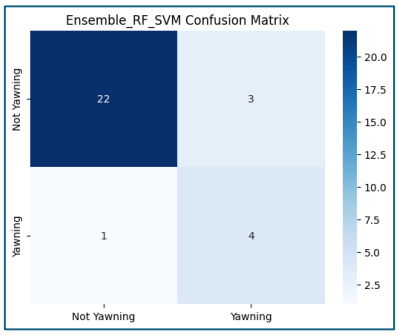

Figure 19: Confusion Matrices of LogReg & Ensemble RF+SVM during Unseen Data Testing

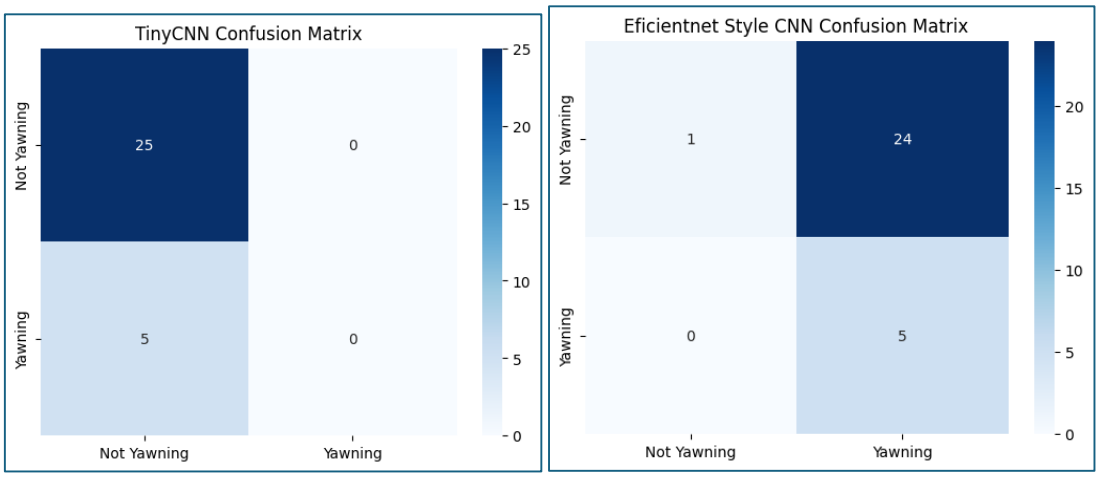

Figure 20: Confusion Matrices of TinyCNN & EfficientNet Style CNN during Unseen Data Testing

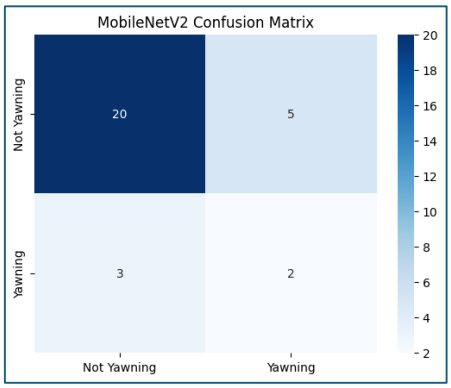

Figure 21: Confusion Matrices of MobileNetV2 during Unseen Data Testing

The confusion matrices highlight that while ML models generally performed better in balancing false positives and false negatives, DL models were prone to misclassification, particularly of drowsy cases. This finding reinforces the conclusion that ML remains more practical for CPU-only real-time deployment, whereas DL requires optimization and hardware acceleration to avoid dangerous false negatives in real-world driver monitoring systems.

Limitations

Several constraints temper these results. it was noted that the data heavily skews towards alertness and as such measures to balance it before training should be taken. It was also noted that the Dash subset of videos was not annotated or labelled in any way so to make use of it, some level of annotation is required.

In addition, Logistic Regression, Random Forest, and Ensemble (RF + SVM) are best suited for CPU-based, real-time applications. These models, however, depend on hand-engineered features like Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR). These kind of features usually perform okay and calculate rapidly, but

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

tend to be corrupted by noise, objects obstructing the view (e.g., sunglasses), and varying illumination, which commonly occur under real driving conditions. As a consequence of these, these models' accuracy and reliability may diminish under challenging real-world conditions.

Additionally, the hand-crafted elements applied for ML models (EAR, MAR) get impacted by sunglasses and varied illumination conditions that occur in real-world driving conditions, thus reducing their potency.

Furthermore, ML models have limited capability in representing high-level spatial and temporal behaviors of face. Not only does drowsiness involve punctual blinks or yawns, but it involves nuanced eye gaze, head motion, and long temporal dependencies as well. Architectures of deep learning, CNNs and LSTMs, can learn these advanced spatio-temporal features more effectively and end-to-end from raw images or video frames.

Thus, while ML models excel in low-resource scenarios and hold promise for real-time use, generalizability over a large population of drivers and diverse conditions is relatively limited. By contrast, deep learning models, while computationally expensive, provide a larger, more comprehensive of drowsiness signals and potentially surpass ML approaches under application on special-purpose hardware (GPU/TPU).

CONCLUSION

This paper contrasted the performance of typical machine learning (ML) methods and deep learning (DL) frameworks for driver drowsiness recognition utilizing the YawDD database. All of these ML methods, especially Logistic Regression, Random Forest, and the ensemble of RF+SVM, exhibited larger inference rates and reduced computation needs, and hence would be more practically usable in CPU-based in-vehicle implementations. Nevertheless, the need for these algorithms to be based on hand-crafted features, e.g., the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR), places a limit on their ability under harsh real-world conditions of poor lighting, occlusions, and inter-subject variation.

Conversely, the deep learning architectures (TinyCNN, MobileNetV2, and EfficientNet-like CNN) exhibited improved representational capability and future accuracy, but they incurred significantly higher inference times when running exclusively on CPU platforms. While these architectures might not be suitable for highly demanding real-time applications under these conditions, they show promise under GPU/TPU-aided environments or in application through model compression frameworks like pruning and quantization.

Overall, the work suggests ML methods remain most realistic for current low-resource, real-time driver monitoring platforms, and DL architectures become a scalable future prospect when available resources rise in newer vehicles.

Future Work

Future work should address dataset limitations by incorporating more diverse subjects, real-world driving conditions, and multimodal signals, as well as exploring optimized DL architectures capable of achieving both robustness and real-time performance on embedded hardware. Future work should also address optimal balancing techniques for driver drowsiness models. Comprehensive benchmarking of ML and lightweight DL models on embedded platforms, taking into account metrics like power consumption, inference speed, and memory footprints, should be the main focus of future research. The viability and resilience of real-time drowsiness detection systems for automotive safety applications may also be improved by investigating model compression, hardware-aware neural architecture search, and multimodal sensor fusion. To add, Future work should explore model compression techniques such as pruning and quantization to enable deployment of DL models on embedded hardware, and investigate hardware-aware neural architecture search to optimize models for resource-constrained real-world systems

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

REFERENCES

1. Abe, T. (2023). PERCLOS-based technologies for detecting drowsiness: current evidence and future directions. SLEEP Advances, 4(1), 1–13. https://doi.org/10.1093/SLEEPADVANCES/ZPAD006
2. Abtahi, S., Omidyeganeh, M., Shirmohammadi, S., & Hariri, B. (2014). YawDD: A yawning detection dataset. Proceedings of the 5th ACM Multimedia Systems Conference, MMSys 2014, 24–28. https://doi.org/10.1145/2557642.2563678
3. Arakawa, T. (2021). Trends and Future Prospects of the Drowsiness Detection and Estimation Technology. Italian National Conference on Sensors, 21(23). https://doi.org/10.3390/S21237921
4. Blom, J., Sam, E. F., Egner, L. E., Nævestad, T. O., & Fiangor, A. (2025). Fatigue among bus drivers in Ghana and Norway: Examining the influence of working conditions and national road safety culture. Transportation Research Procedia, 89, 394–403. https://doi.org/10.1016/j.trpro.2025.05.070
5. Deepika, C. N. J., Jithendra, K., Poojitha, B., Ranganath, R. M., Vaithilingam, C. A., & Sivaraman, K. (2025). Driver drowsiness detection using machine learning algorithms. AIP Conference Proceedings, 3253(1). https://doi.org/10.1063/5.0252639
6. Dharshini R, ; Janani V, & Bharanidharn, ; C. (2024). DRIVER DROWSINESS DETECTION USING MACHINE LEARNING. www.tijer.org
7. Dinges, D. F., Mallis, M. M., Maislin, G., Powell, J. W., & Administration, U. States. D. of Transportation. N. H. T. S. (1998). Evaluation of techniques for ocular measurement as an index of fatigue and as the basis for alertness management. https://doi.org/10.21949/1503647
8. Elidrissi, M. E., Essoukaki, E., Taleb, L. Ben, Mouhsen, A., & Harmouchi, M. (2023). Drivers' drowsiness detection based on an optimized random forest classification and single-channel electroencephalogram. International Journal of Electrical and Computer Engineering, 13(3), 3398–3406. https://doi.org/10.11591/ijece.v13i3.pp3398-3406
9. Essahraui, S., Lamaakal, I., El Hamly, I., Maleh, Y., Ouahbi, I., El Makkaoui, K., Filali Bouami, M., Pławiak, P., Alfarraj, O., & Abd El-Latif, A. A. (2025). Real-Time Driver Drowsiness Detection Using Facial Analysis and Machine Learning Techniques. Sensors 2025, Vol. 25, Page 812, 25(3), 812. https://doi.org/10.3390/S25030812
10. Fonseca, T., & Ferreira, S. (2025). Drowsiness Detection in Drivers: A Systematic Review of Deep Learning-Based Models. Applied Sciences 2025, Vol. 15, Page 9018, 15(16), 9018. https://doi.org/10.3390/APP15169018
11. Ibrahim, A. (2022, May 6). Driver fatigue primary cause of road crashes in Ghana - Road Safety Authority. CITI NEWSROOM. https://citinewsroom.com/2022/05/driver-fatigue-primary-cause-of-road-crashes-in-ghana-road-safety-authority/
12. Jasim, S. S., Karim, A., & Hassan, A. (2022). Modern Drowsiness Detection in Deep Learning: A review. Journal of Al-Qadisiyah for Computer Science and Mathematics, 14(3), 119–129. https://doi.org/10.29304/jqcm.2022.14.3.1023
13. Kaundal, S., Chauhan, K., & Rana, A. (2025). A Comprehensive Literature Review on Driver Drowsiness Detection and Alert System. International Journal of Electrical, Electronics ISSN No, 14(2), 32–35. www.researchtrend.net
14. Khanehshenas, F., Mazloumi, A., Nahvi, A., Nickabadi, A., Sadeghniiat, K., Rahimiforoushani, A., & Aghamalizadeh, A. (2024). A hybrid approach for driver drowsiness detection utilizing practical data to improve performance system and applicability. Work, 77(4), 1165–1177. https://doi.org/10.3233/WOR-230179
15. Moradi, A., Nazari, S. S. H., & Rahmani, K. (2019). Sleepiness and the risk of road traffic accidents: A systematic review and meta-analysis of previous studies. Transportation Research Part F: Traffic Psychology and Behaviour, 65, 620–629. https://doi.org/10.1016/J.TRF.2018.09.013
16. National Road Safety Authority. (2025). January–March 2025 Crash Report Summary. Government of Ghana. https://www.nrsa.gov.gh/
17. Schwarz, C., Gaspar, J., & Yousefian, R. (2023). Multi-sensor driver monitoring for drowsiness prediction. Traffic Injury Prevention, 24(S1), S100–S104. https://doi.org/10.1080/15389588.2023.2164839

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

18. Srinivas, K., V M U Vinay Karthik, G. V, Suhas, I., Surendra, A., & Sujith, B. (n.d.). Driver Drowsiness Detection using Machine Learning and Deep Learning. https://doi.org/10.48175/IJARSCT-17442
19. SSATP. (2025). Africa Status Report on Road Safety. SSATP. https://www.ssatp.org/sites/default/files/publication/SSATP_Africa%20Status%20Report%20on%20Road%20Safety%202025_single.pdf
20. Wierwille, W. W., Wreggit, S. S., Kirn, C. L., Ellsworth, L. A., & Fairbanks, R. J. (1994). Research on Vehicle-Based Driver Status/Performance Monitoring; Development, Validation, and Refinement of Algorithms For Detection of Driver Drowsiness. https://rosap.ntl.bts.gov/view/dot/2578