 ISSN XXXX-XXXX (online)

TENAR
Technology, Engineering, and Artificial Intelligence
Journal Page is available to www.sainsnusa.com/index.php/tea

Research article

Driver Drowsiness Detection Using LSTM and MediaPipe FaceMesh-Based Facial Landmark Analysis

Afif Fajar Hijriyanto [a,1,*]; *Ike Kurniati*[b,2]

[a] *Department of Informatic, University of Krisnadwipayana, Kota Bekasi, 17411 West Java, Indonesia*
[b] *Faculty of Technology, Swadharma Institute of Technology and Business, Jalan Malaka, 11230 West Jakarta, Indonesia*
email: [1,] afif.rpl2@gmail.com, [2] ikekurniati@swadharma.ac.id.*

** Correspondence*
http://doi.org/xxxx.xxxx

ARTICLE INFO

Article history:
Received : *-----*
Accepted : *-----*
Available online : -----

Keywords:
Drowsiness detection
Driver
Long Short-Term Memory (LSTM)
MediaPipe FaceMesh
Face classification
Deep learning
Video analysis

ABSTRACT

This research is motivated by the high number of traffic accidents caused by human factors, particularly driver drowsiness, which is often not detected early. This condition has the potential to reduce concentration and increase the risk of accidents. Therefore, this study aims to develop a facial classification model to automatically detect driver drowsiness based on deep learning technology. The method used is Long Short-Term Memory (LSTM) to process sequential video data, and MediaPipe FaceMesh to extract facial features consisting of 478 landmark points. The dataset consists of 220 videos with two classes: drowsy and non-drowsy, divided into training, validation, and testing data. The results show that the proposed model achieved an accuracy of 95.5%, with a precision of 95.9%, a recall of 95.5%, and an F1-score of 95.5%. This indicates that the model performs well in distinguishing between drowsy and non-drowsy states. In conclusion, the combination of LSTM and MediaPipe is effective for video-based sleepy facial classification. The contribution of this research is the development of an accurate drowsiness detection model that has the potential to be developed into a real-time driver monitoring system to improve driving safety.

1. INTRODUCTION

Traffic accidents remain a serious problem in various countries, including Indonesia, with human factors being the dominant cause. One of the main factors is driver fatigue and drowsiness, which can reduce concentration and slow response to road situations[1],[2]. Early detection of driver drowsiness is crucial to minimize the risk of accidents. Advances in Artificial Intelligence (AI) technology, particularly in the fields of computer vision and deep learning, open up opportunities to develop automated facial analysis-based systems capable of detecting drowsiness in real time[3],[4].

Several previous studies have examined driver drowsiness detection using various approaches. Eye-tracking-based methods have shown quite good results, but are highly dependent on lighting conditions and camera quality[5],[6]. Other approaches using classical algorithms such as Viola-Jones for eye detection also have limitations in accuracy and stability under dynamic conditions[7], [8]. Recent research based on deep learning, specifically using Long Short-Term Memory (LSTM) and facial keypoint analysis, has shown improved performance, achieving over 90% accuracy in detecting driver fatigue[2],[9]. Furthermore, the use of frameworks such as MediaPipe allows for more detailed and efficient facial feature extraction.

However, several research gaps remain. First, most research still focuses on single features, such as the eyes, thus underrepresenting the full range of facial expressions. Second, the utilization of sequential video data has not been optimally optimized, even though temporal information is crucial for detecting changes in sleepiness. Third, the integration of detailed facial landmark extraction with sequential-based deep learning models such as LSTM is still relatively limited.

Based on these issues, this study aims to develop a facial analysis-based driver drowsiness classification model using a combination of MediaPipe FaceMesh and the LSTM algorithm. The main contributions of this study are: (1) utilizing 478 facial landmark points as a comprehensive representation of facial features, (2) applying

ISSN XXXX-XXXX (online)
1 (1) April 2026 XX-XX

LSTM to capture temporal patterns from video data, and (3) developing a high-performance model with the potential for implementation in a real-time drowsiness detection system to improve driving safety.

2. METHODS

In this study, several main stages were carried out to automatically detect driver drowsiness using MediaPipe FaceMesh and Long Short-Term Memory (LSTM)[10]. These stages include data collection, video recording of facial gestures, and extraction of key facial landmarks using MediaPipe FaceMesh. Next, an LSTM-based artificial neural network model was built, which was then trained and evaluated using a confusion matrix. The final stage of the research was the implementation of the trained model in the form of a system prototype capable of automatically detecting and recognizing driver drowsiness. The overall flow of the research methodology can be seen in Figure 1.

Figure 1. Research Methodology

2.1. Data Collection

The dataset used in this study consisted of facial videos of 22 different drivers, Figure 2. Data was obtained from two sources: primary data from live recordings and secondary data from public datasets. The data collection process involved recording driver activity, then cutting the raw video data into 4-second chunks.

Figure 2. Driver Face Dataset

The total dataset used was 220 videos, which were then classified into two states: drowsy and non-drowsy, with 110 videos each. Each video had a frame rate of 30 FPS, resulting in approximately 120 frames per video. This data is sequential because it represents changes in facial conditions over time. The data is then organized into a folder structure based on class and video, with each video consisting of a collection of frames. Furthermore, each frame will be extracted into features in the form of facial keypoints, which are used as input in the model training process.

2.2. Keypoint Extraction, Normalization, and Video Capture

The data preparation stage was carried out to process the raw data into data ready for use in the model training process. In this study, the main features used were facial keypoints obtained from MediaPipe FaceMesh. Each face is represented by 478 landmark points, where each point has coordinates (x, y, z) that describe the facial structure in detail, Figure 3.

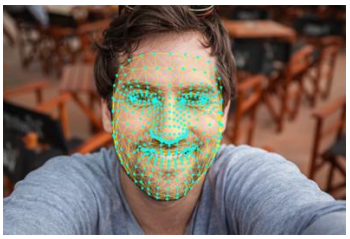


Figure 3. Landmark Model for Faces

ISSN XXXX-XXXX (online)
1 (1) April 2026 XX-XX

The x- and y-coordinates are normalized within the [0,1] range based on the frame size, while the z-coordinate is relative to depth. To ensure data consistency, normalization is performed using the Min-Max method so that the position of the landmark is relative to the entire face in one video, which can be calculated using formula 1.

$$x' = \frac{x - \min x}{\max x - \min x} \tag{1}$$

The keypoint extraction process was performed using video frames captured using OpenCV, which were then processed by MediaPipe FaceMesh, Figure 4. The extraction results were stored to avoid data reprocessing. Next, the processed data is labeled according to its class, namely sleepy and not sleepy. These labels are represented using one-hot encoding so they can be used in the model's classification process.

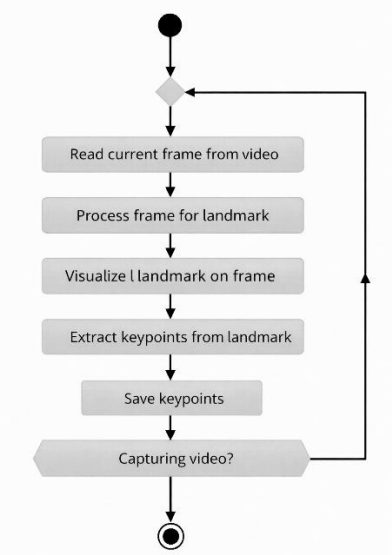


Figure 4. Flowchart for Video Capture and Keypoint Extraction

2.3. Long Short-Term Memory Modeling

LSTM is a development of RNN designed to address the vanishing gradient problem and improve its ability to handle long-term dependencies. The LSTM architecture consists of a memory cell, an input gate, an output gate, and a forget gate, which regulates the flow of information. The forget gate determines which information is retained or discarded, while the input gate and the tanh function are used to update the values in the memory cell. The main computation in LSTM involves the component $f_{t,}, i_t \; and \; \widetilde{C_t}$, which is used to control the information updates in the model[11], [12], as shown in formulas 2-4.

$$f_t = \sigma(W_f . [hidden_{t-1}, dm\_x_t] + b_f) \tag{2}$$

$$i_t = \sigma(W_t . [hidden_{t-1}, dm\_x_t] + b_t) \tag{3}$$

$$Ct\_hat = \tanh(W_c . [hidden_{t-1}, dm\_x_t] + b_c) \tag{4}$$

In the modeling phase, a Long Short-Term Memory (LSTM)-based classification model was built to process sequential data resulting from facial keypoint extraction. The model received 30 × 1434 input, representing 30 timesteps and 1434 features from 478 facial landmarks with coordinates (x, y, z). The model architecture consisted of an input layer, one to two LSTM layers as hidden layers, and a Dense output layer with a softmax activation function for two-class classification, Figure 5. To improve performance, hyperparameter optimization was performed using the Hyperband algorithm, including the number of LSTM layers, the number of units (128–1024), dropout (0.0–0.5), the learning rate (0.0001–0.01), and the batch size. The model was trained using the Adam optimizer due to its ability to accelerate convergence. The loss function used was categorical cross-entropy, while the evaluation metric used was categorical accuracy. The training process was carried out for 200 epochs to achieve optimal model performance.

ISSN XXXX-XXXX (online)
1 (1) April 2026 XX-XX

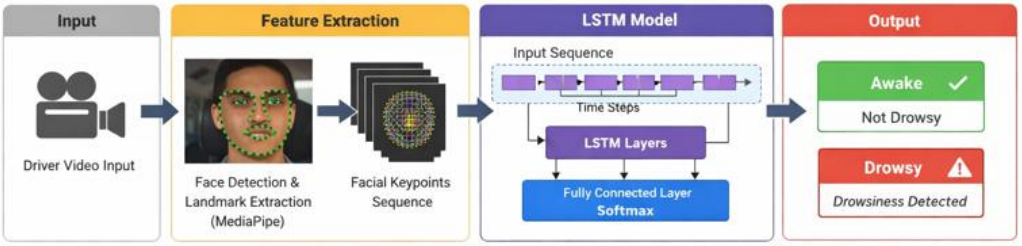


Figure 5. LSTM Model Architecture

2.4. Model Evaluation

The evaluation phase aims to measure the model's performance in classifying driver conditions into two classes: drowsy and non-drowsy. The evaluation is conducted using testing data not involved in the training process. One evaluation method used is a confusion matrix, which illustrates the comparison between the model's predicted results and the actual conditions[13], [14],[15]. The confusion matrix consists of four main components, Table 1.

Table 1. *Confusion Matrix*

Actual / Predicted	Drowsy	Non-Drowsy
Drowsy	True Positive (TP)	False Negative (FN)
Non-Drowsy	False Positive (FP)	True Negative (TN)

Based on these values, model performance is measured using several evaluation metrics, which can be calculated using Formula 5-7.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

$$Precision = \frac{TP}{TP + FP} \quad (6)$$

$$Recall = \frac{TP}{TP + FN} \quad (7)$$

$$F1 - Score = 2X \frac{Precision\ X\ Recall}{Precision + Recall} \quad (7)$$

2.5. Prototype Creation

The prototype creation stage is the process of implementing the trained model into a simple system capable of automatically detecting driver drowsiness. At this stage, the optimally performing LSTM model is integrated with a MediaPipe FaceMesh-based image processing system to perform face detection and facial keypoint extraction in real time. The prototype system receives live video or camera input, then processes each frame to extract facial features and classify the driver's condition as drowsy or non-drowsy. The prediction results are displayed directly as system output[16]. Thus, this prototype demonstrates that the developed model can be implemented in real-world applications as a driving safety support system.

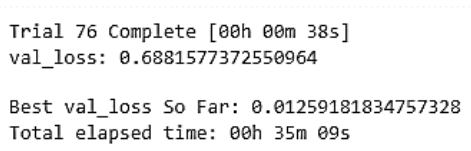
3. RESULT

The results of this study include an evaluation of the performance of a Long Short-Term Memory (LSTM)-based model in classifying driver conditions, which is analyzed using various evaluation metrics to measure the level of accuracy and reliability of the system. The system implementation is carried out using the Python programming language with the support of several libraries, namely OpenCV for image and video processing, NumPy for numerical computation, Matplotlib for data visualization, MediaPipe for facial feature extraction, and scikit-learn and TensorFlow for modeling and evaluation processes..

ISSN XXXX-XXXX (online)
1 (1) April 2026 XX-XX

3.1. Model Hyperparameters

Hyperparameter optimization using the Hyperband algorithm yielded the best configuration with a validation loss of 0.0126, indicating optimal model performance, Figure 6a. The model uses two LSTM layers with 128 and 256 units, with dropout rates of 0.2 and 0.3, respectively, effectively capturing temporal patterns while reducing overfitting. A low learning rate (0.0001789) helps maintain training stability, while a batch size of 256 speeds up computation, Figure 6b. The data split (10% testing and 5% validation) is considered proportional, and 200 epochs of training indicate that the model begins to converge from the 34th epoch. Overall, this parameter combination produces a model with high and stable performance, characterized by low validation loss and consistent evaluation results.



Hyperparameter	Score
num_of_lstm_layers	2
lstm_0_units	128
lstm_0_dropout_rate	0.2
learning_rate	0.00017893444548566697
lstm_1_units	256
lstm_1_dropout_rate	0.30000000000000004
batch_size	256
test_size	0.1
validation_size	0.05
tuner/epochs	200
tuner/initial_epoch	34
tuner/bracket	1
tuner/round	1
tuner/trial_id	'0060'

(a) (b)

Figure 6. (a) Results of Searching for The Best Hyperparameter Configuration, and (b) Best Hyperparameters

3.2. Model Building and Training

The model used is a Sequential architecture with two LSTM layers and one Dense output layer. The first LSTM layer produces sequential output (120 timesteps, 128 units), which serves to capture the initial temporal patterns of the data. The second LSTM layer reduces the output to a 256-dimensional vector, enabling more complex feature extraction. The Dense output layer, with two neurons, uses a softmax activation function to classify two classes: sleepy and non-sleepy. The total model parameters are 1,195,010, all of which are trainable, indicating that the model has sufficient capacity to learn data patterns, Figure 7. However, the large number of parameters also needs to be balanced with regularization techniques such as dropout to avoid overfitting. Overall, this architecture is well-suited for handling sequential data and is able to capture temporal patterns effectively.

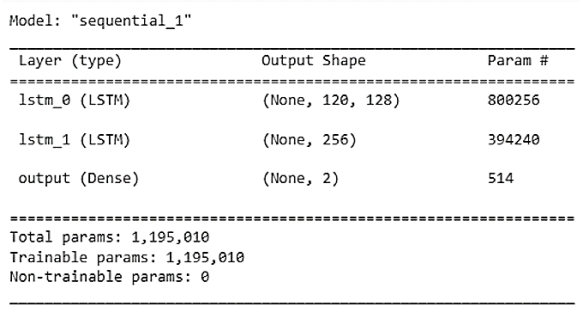


Figure 7. Results of The Model with The Best Hyperparameters

The data splitting process was performed using the train_test_split function, with the test size proportion derived from the hyperparameter optimization results, Figure 8. Using random_state=0 ensures consistent and reproducible data splitting. During the model training phase, the model.fit() function was used with 200 epochs, allowing the model to learn optimally from the data. The batch size and validation split parameters were also derived from the Hyperband tuning results, so the training configuration was adjusted to achieve optimal performance. Furthermore, using the same validation batch size as the batch size helped maintain consistency in the evaluation process throughout training. Overall, this approach demonstrates that the training process was designed adaptively and based on optimization results, increasing the model's chances of achieving stable and accurate performance.

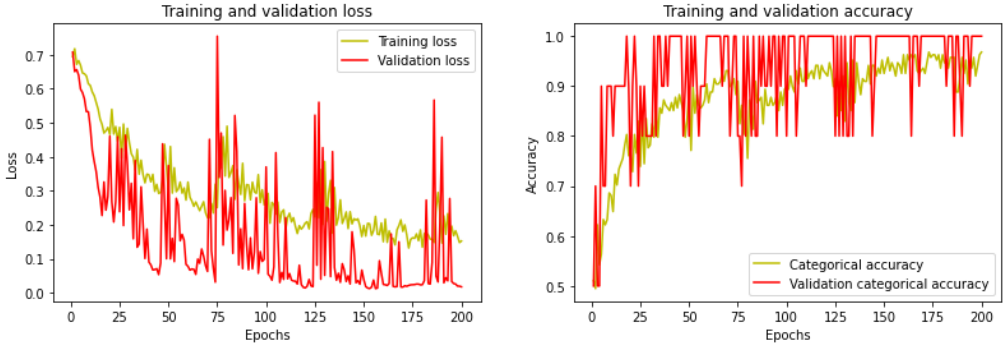
ISSN XXXX-XXXX (online)
1 (1) April 2026 XX-XX

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=best_hp.get('test_size'),
    random_state=0)
```

```
history = model.fit(
    X_train, y_train,
    epochs=200,
    batch_size=best_hp.get('batch_size'),
    validation_split=best_hp.get('validation_size'),
    validation_batch_size=best_hp.get('batch_size'))
```

Figure 8. Separating Training and Testing Data

The training loss graph shows a steady downward trend as the number of epochs increases, indicating that the model has successfully learned the data patterns, Figure 9a. Meanwhile, the validation loss also tends to decrease, although there are significant fluctuations. These fluctuations are likely caused by variations in the validation data or the relatively limited dataset size. The accuracy graph shows that the training accuracy consistently increases to a high value, indicating the model's ability to recognize patterns in the training data, Figure 9b. Conversely, the validation accuracy also shows an increase, but with quite sharp fluctuations. This indicates that the model's performance on new data remains slightly unstable. Overall, the model has demonstrated good performance with a tendency to converge, but there are still mild indications of overfitting or sensitivity to the validation data. However, the difference between training and validation is not significant, indicating that the model still has quite good generalization ability.



(a) (b)

Figure 9. (a) Training and Validation Loss, and (b) Accuracy by Training and Validation Categories

3.3. Model Evaluation

Model evaluation is performed to assess model performance using various evaluation metrics. This stage aims to identify the model's strengths and limitations and ensure its performance. Furthermore, evaluation is used to monitor model performance after the training process. In the evaluation process, the model is used to predict pre-separated test data using the command pred = model.predict(X_test, batch_size=best_hp.get('batch_size')). The predicted results are then compared with the actual labels to measure the model's accuracy and overall performance. Based on Figure 10a, the output displayed represents the model's prediction results in the form of probabilities for two classes: sleepy and not sleepy. Each row shows the model's confidence level for each class, with higher values indicating a more dominant prediction. Most probability values are close to 0 or 1, indicating that the model has a high level of confidence in its classification. For example, values such as [0.005, 0.994] indicate that the model is very confident in the second class, while [0.985, 0.014] indicates high confidence in the first class.

However, there are some values closer to the middle, such as [0.26, 0.73] or [0.32, 0.67], indicating that the model is less confident in determining the class. This typically occurs in data with ambiguous or unclear

ISSN XXXX-XXXX (online)
1 (1) April 2026 XX-XX

characteristics. Overall, these results indicate that the model is capable of classifying with a high level of confidence on most data, although there are still some cases with a degree of uncertainty.

```
1/1 [==============================] - 1s 665ms/step
array([[0.0095209 , 0.99047905],
       [0.9850187 , 0.01498133],
       [0.985559  , 0.01444099],
       [0.5197255 , 0.4802745 ],
       [0.00893096, 0.9910691 ],
       [0.01833282, 0.98166716],
       [0.9804466 , 0.01955337],
       [0.00500804, 0.99499196],
       [0.00520086, 0.99479914],
       [0.03038928, 0.9696107 ],
       [0.26116192, 0.7388381 ],
       [0.9876171 , 0.01238296],
       [0.98329127, 0.01670881],
       [0.02527615, 0.9747239 ],
       [0.02125087, 0.9787491 ],
       [0.01361615, 0.98638386],
       [0.329512  , 0.670488  ],
       [0.9814755 , 0.01852449],
       [0.97139794, 0.02860205],
       [0.98270434, 0.01729569],
       [0.9855978 , 0.01440217],
       [0.9838743 , 0.01612572]], dtype=float32)
```

(a)

```
array([1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1])

array([1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1])
```

(b)

Figure 10. (a) Results of testing data prediction, and (b) Results of taking the position of the highest predicted value index

To generate a confusion matrix, the model first generates predictions in the form of probabilities for each class. These probability values are then converted into class labels by selecting the highest index using argmax, Figure 10b. The same process is applied to the actual data. The predicted results and actual labels are then compared to evaluate model performance using the confusion matrix and other evaluation metrics.

3.3.1. Confusion Matrix

Model performance evaluation was conducted using a confusion matrix to provide a detailed overview of the model's ability to classify sleepy and non-sleepy conditions in the test data. Based on the results obtained, out of a total of 22 test data, 21 data were successfully classified correctly, consisting of 10 sleepy data that were correctly detected (true positive) and 11 non-sleepy data that were also correctly classified (true negative), Figure 11. Meanwhile, there was 1 non-sleepy data that was incorrectly classified as sleepy (false positive), and no classification errors were found in the sleepy data (false negative). These results indicate that the model has a high level of accuracy and is able to distinguish between the two classes well, although there are still small classification errors that are likely caused by ambiguous data characteristics.

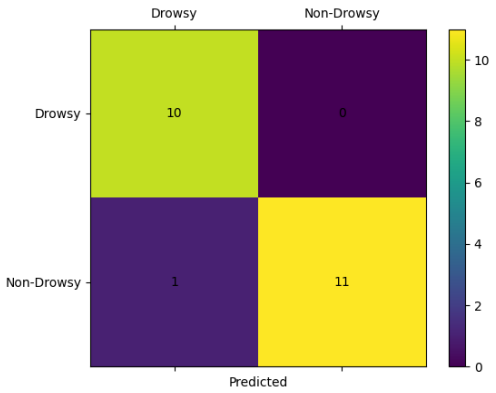


Figure 11. Confusion Matrix

3.3.2. Accuracy, Precision, Recall, F1-score, and Support

The classification report function in scikit-learn is used to present model evaluation metrics, including accuracy, precision, recall, F1-score, and support through the classification_report(y_true, y_pred) command, the results can be seen in Table 2. Accuracy describes the overall level of model accuracy, precision shows how accurate the predictions are for each class, recall measures the model's ability to recognize correct data, while F1-score is the balance value between precision and recall. Model performance was evaluated using accuracy,

ISSN XXXX-XXXX (online)
1 (1) April 2026 XX-XX

precision, recall, F1-score, and support metrics. The model achieved an accuracy of 95.45%, indicating that most of the data was correctly classified. A precision of 90.91% indicates that most of the predictions for the drowsiness class were correct, although there were still a few false positives. Meanwhile, a recall of 100% indicates that all true drowsiness data were detected by the model without any false negatives.

Table 2. *Classification Report from The Prediction and Testing Data*

Metric	Drowsy	Non-Drowsy	Overall
Accuracy	–	–	**95.45%**
Precision	90.91%	100%	–
Recall	100%	91.67%	–
F1-Score	95.24%	95.65%	–
Support	10	12	22

An F1-score of 95.24% demonstrates a good balance between precision and recall, so the model can be considered to have stable performance in detecting drowsiness. The support value indicates the number of data points for each class: 10 for the drowsiness class and 12 for the non-drowsy class. Overall, these results indicate that the model performs very well in classifying driver conditions, with a very low error rate and high detection capability for drowsiness.

3.4. Prototype Creation

At this stage, the trained model was successfully implemented in a prototype system to automatically detect driver drowsiness. The system is capable of receiving real-time video or camera input, extracting facial keypoints using MediaPipe FaceMesh, and classifying them using an LSTM model. Test results showed that the prototype can effectively recognize drowsiness and non-drowsiness and display prediction results directly. This demonstrates that the developed model can be practically implemented in a driver drowsiness detection system.

4. DISCUSSIONS

This discussion examines the experimental results obtained in depth, including the model's advantages and limitations, and comparisons with previous research in the field of driver drowsiness detection. Based on related studies, driver drowsiness detection generally utilizes two main approaches: facial feature-based approaches using MediaPipe FaceMesh and deep learning-based approaches such as LSTM or CNN-LSTM, Table 3. FaceMesh-based approaches have proven effective in extracting facial features in real time with relatively low computational costs, as demonstrated by studies by Albadawi et al. (2023).

On the other hand, deep learning-based approaches, particularly LSTM, excel at capturing temporal patterns from sequential data. Studies by Yu et al. (2024) and Gautam et al. (2025) show that the combination of LSTM with facial features or CNN can achieve accuracy above 96%. Furthermore, the integration of MediaPipe FaceMesh and LSTM, as in the study by Akrout & Fakhfakh (2023), demonstrates excellent performance due to its ability to combine the advantages of feature extraction and temporal modeling. However, some studies still have limitations, such as reliance on a single feature (e.g., EAR alone), the use of limited datasets, or high model complexity. Furthermore, the CNN-LSTM approach tends to require more computational resources than landmark-based methods.

Based on these studies, this research occupies a relevant position by combining MediaPipe FaceMesh for detailed facial feature extraction (478 landmarks) and LSTM for capturing temporal patterns. This approach is expected to produce a model with high accuracy, efficiency, and suitability for real-time implementation, thus making a significant contribution to the development of driver drowsiness detection systems.

Table 3. *Comparison with Previous Research*

Reference & Year	Main Features	Core Technology	Dataset / Setting	Main Performance	Relevance Notes
Albadawi et al., 2023[17]	EAR, MAR, head pose	FaceMesh + RF, NN, SVM	NTHU DDD	Accuracy up to 99%	FaceMesh effective for real-time visual features

Jiao et al., 2025[18]	EAR, MOR, eye closure duration	MediaPipe + rule-based, compared with CNN-LSTM	YawDD	High accuracy, low FP/FN	Comparison between FaceMesh and CNN-LSTM
Badri et al., 2025[19]	Blinking, facial histogram	MediaPipe + regression tree	Bus driver video	88.75%	Comparison between CNN and MediaPipe
Yu et al., 2024[2]	Facial landmarks + HRV (PPG)	LSTM	Real-road data	97.36%	Combines facial and physiological data
Akrout & Fakhfakh, 2023[20]	Gaze, head pose	FaceMesh + MobileNetV3 + LSTM	Multi-level awareness	98.4%	Highly relevant (FaceMesh + LSTM)
Gautam et al., 2025[21]	Eye closure, yawning	CNN + LSTM	Real-time camera	96.3%	Hybrid CNN-LSTM approach
Das et al., 2024[22]	Facial/body movement	U-Net + CNN-LSTM + IoT	Multi-frame video	High performance	Combines segmentation and LSTM
Dalve et al., 2023[23]	Facial landmarks	FaceMesh + NN + fuzzy logic	Driving video	92%	Minimizes false alerts
Rathod et al., 2023[24]	EAR, MAR, object detection	FaceMesh + YOLO + PERCLOS	Real-time	94%	Combines landmarks and object detection

5. CONCLUSION

This research successfully developed a driver drowsiness detection model based on Long Short-Term Memory (LSTM) and MediaPipe FaceMesh, capable of classifying drowsy and non-drowsy states with high accuracy. The use of 478 facial landmarks and sequential data modeling proved effective in capturing facial change patterns, resulting in stable and reliable performance. For further development, it is recommended to expand the dataset to improve the model's generalizability and develop a more detailed classification of drowsiness levels. Furthermore, integration with other methods and real-time implementation of the system on real devices are also necessary to improve the system's performance and usability in driving safety applications.

AUTHOR CONTRIBUTIONS

Afif Fajar Hijriyanto: Conceptualization, Data Curation, Methodology, Validation, Writing-Original Draft Preparation;
Ike kurniati: Project Administration, Writing – Review & Editing;

CONFLICT OF INTEREST

The authors declares that there is no conflict of interest between the authors or with research object in this paper.

ACKNOWLEDGEMENT

Acknowledgement is only addressed to funders or donors and object of research. Acknowledgement can also be expressed to those who helped carry out the research.

REFERENCES

[1] M.-Z. Liu, X. Xu, J. Hu, and Q.-N. Jiang, “Real time detection of driver fatigue based on CNN-LSTM,” *IET Image Process.*, vol. 16, no. 2, pp. 576–595, 2022.
[2] L. Yu, X. Yang, H. Wei, J. Liu, and B. Li, “Driver fatigue detection using PPG signal, facial features, head postures with an LSTM model,” *Heliyon*, vol. 10, no. 21, 2024.
[3] S. A. El-Nabi, W. El-Shafai, E.-S. M. El-Rabaie, K. F. Ramadan, F. E. Abd El-Samie, and S. Mohsen, “Machine learning and deep learning techniques for driver fatigue and drowsiness detection: a review,” *Multimed. Tools Appl.*, vol. 83, no. 3, pp. 9441–9477, 2024, doi: https://doi.org/10.1007/s11042-023-15054-0 .
[4] T. Fonseca and S. Ferreira, “Drowsiness Detection in Drivers: A Systematic Review of Deep Learning-

ISSN XXXX-XXXX (online)
1 (1) April 2026 XX-XX

Based Models," *Appl. Sci.*, vol. 15, p. 9018, Aug. 2025, doi: 10.3390/app15169018.

[5] Z. Gomolka, D. Kordos, E. Dudek-Dyduch, and B. Twarog, "New Perspectives on Eye-Tracking: Theory, Methods, and Applications," *Appl. Sci.*, vol. 15, no. 21, p. 11463, 2025, doi: https://doi.org/10.3390/app152111463.

[6] S. Bendjebar, Y. Lafifi, R. Boudjehem, and A. Laouissi, "Data-Driven Insights into E-Learning: A Comprehensive Review of Eye-Tracking Applications in Learning Systems," *J. Eye Mov. Res.*, vol. 19, no. 2, p. 41, 2026, doi: 10.3390/jemr19020041.

[7] F. Nemsia, Y. Zouhir, M. Zarka, and K. Ouni, "Improving Object Detection and Tracking: A Hybrid Method for Real-Time Applications," in *2025 IEEE International Conference on Advanced Systems and Emergent Technologies (IC_ASET)*, IEEE, 2025, pp. 1–6. doi: DOI: 10.1109/IC_ASET65966.2025.11232214.

[8] V. V Arlazarov, D. P. Matalov, D. P. Nikolaev, and S. A. Usilin, "Evolution of the Viola-Jones object detection method: A survey," *Bull. South Ural State Univ. Ser. Math. Model. Program. Comput. Softw.*, vol. 14, no. 4, pp. 5–23, 2021, doi: DOI: 10.14529/mmp210401.

[9] L. Chen, G. Xin, Y. Liu, and J. Huang, "Driver fatigue detection based on facial key points and LSTM," *Secur. Commun. Networks*, vol. 2021, no. 1, p. 5383573, 2021, doi: https://doi.org/10.1155/2021/5383573.

[10] W. Utomo, Y. Suhanda, H. Ar-Rasyid, and A. Dharmalau, "Indonesian Language Sign Detection using Mediapipe with Long Short-Term Memory (LSTM) Algorithm," *J. Informatics Web Eng.*, vol. 4, no. 3, pp. 245–258, 2025, doi: 10.33093/jiwe.2025.4.3.15.

[11] J. F. Torres, F. Martínez-Álvarez, and A. Troncoso, "A deep LSTM network for the Spanish electricity consumption forecasting," *Neural Comput. Appl.*, vol. 34, pp. 10533–10545, 2022, doi: 10.1007/s00521-021-06773-2.

[12] N. Akhtar, J. Fan, A. R. Buzdar, M. Ahmed, and A. Raza, "VLSI Design of LSTM-Based ECG Classification for Continuous Cardiac Monitoring on Wearable Devices," *Electron. Lett.*, vol. 61, p. e70269, 2025, doi: 10.1049/ell2.70269.

[13] M. I. B. Ahmed *et al.*, "A Deep-Learning Approach to Driver Drowsiness Detection," *Safety*, vol. 9, no. 3, p. 65, 2023, doi: 10.3390/safety9030065.

[14] G. Nguyen, L. Wang, Y. Jiang, and T. Gedeon, "Detecting Fake News Belief via Skin and Blood Flow Signals," *IEEE Access*, vol. 14, pp. 30265–30277, 2026, doi: 10.1109/ACCESS.2024.3423723.

[15] E. Essel, F. Lacy, F. Albalooshi, W. Elmedany, and Y. Ismail, "Drowsiness Detection in Drivers Using Facial Feature Analysis," *Appl. Sci.*, vol. 15, no. 1, p. 20, 2025, doi: 10.3390/app15010020.

[16] O. O. Ajayi, A. M. Kurien, K. Djouani, and L. Dieng, "A Multimodal Systematic Review of Drivers' Fatigue Detection Methodologies, Datasets, and Models," *IEEE Access*, vol. 13, pp. 158266–158284, 2025, doi: 10.1109/ACCESS.2025.3606900.

[17] Y. Albadawi, A. AlRedhaei, and M. Takruri, "Real-Time Machine Learning-Based Driver Drowsiness Detection Using Visual Features," *J. Imaging*, vol. 9, p. 91, 2023, doi: 10.3390/jimaging9050091.

[18] X. Jiao, "Multi-Fatigue Indicator Detection for Drivers Using FaceMesh," in *2025 6th International Conference on Electronic Communication and Artificial Intelligence (ICECAI)*, 2025, pp. 396–399. doi: 10.1109/icecai66283.2025.11171211.

[19] F. Badri, S. U. R. Sari, and S. A. Bin Hamzah, "Analysis of Driver Drowsiness Detection System Based on Landmarks and MediaPipe," *Inf. J. Ilm. Bid. Teknol. Inf. dan Komun.*, vol. 10, no. 1, 2025, doi: 10.25139/inform.v10i1.9325.

[20] B. Akrout and S. Fakhfakh, "How to Prevent Drivers before Their Sleepiness Using Deep Learning-Based Approach," *Electronics*, vol. 12, p. 965, 2023, doi: 10.3390/electronics12040965.

[21] T. Gautam, "Novel CNN Based Model Using LSTM for Driver Drowsiness Detection," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 13, no. 4, 2025, doi: 10.22214/ijraset.2025.69131.

[22] S. Das, "IoT-Assisted Automatic Driver Drowsiness Detection through Facial Movement Analysis Using Deep Learning and a U-Net-Based Architecture," *Information*, vol. 15, p. 30, 2024, doi: 10.3390/info15010030.

[23] S. Dalve, I. Ramdasi, G. Kothawade, Y. Khadke, and M. Wete, "Real Time Prevention of Driver Fatigue Using Deep Learning and MediaPipe," *Int. J. Innov. Res. Comput. Sci. Technol.*, vol. 11, no. 3, pp. 7–11, 2023, doi: 10.55524/ijircst.2023.11.3.2.

[24] S. Rathod, T. Mali, Y. Jogani, N. Faldu, V. Odedra, and P. Barik, "RealD3: A Real-time Driver Drowsiness Detection Scheme Using Machine Learning," in *2023 IEEE Wireless Antenna and Microwave Symposium (WAMS)*, 2023. doi: 10.1109/wams57261.2023.10242860.