UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

VIETNAM FRANCE UNIVERSITY

Research and Development

BACHELOR THESIS

By

Tăng Văn Anh (BA12-007)

Information and communication technology

Title:

"Driver Fatigue Detection Using Emotional Facial Recognition"

Supervisor: Dr. Tran Hoang Tung - *ICT Lab*

Hanoi, 2025

To whom it may concern,

I, Trần Hoàng Tùng (supervisor's name), certify that the thesis/ internship report of Tăng Văn Anh (student's name) is qualified to be presented in the Internship Jury 2024-2025.

Hanoi, 3rd July 2025

Supervisor's signature

Table of Contents

ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who gave me the possibility to complete this thesis, in particular, as well as the academic support in general that I received during my time of studying at the University of Science and Technology of Hanoi.

To begin with, I want to express my deep gratitude to my supervisor, Dr. Tran Hoang Tung. During my internship, he always took the time to support and guide me whenever I needed help. His comments and suggestions made a big difference in how I approached and completed my project. I also learned a lot from his way of working and problem-solving. I truly appreciate his patience, dedication, and the useful knowledge he shared with me. Without his help, finishing this internship would have been much more difficult.

LIST OF ABBREVIATIONS

BP	Biotechnology- Pharmacology
CMOS	Complementary Metal-Oxide-Semiconductor
CMRR	Common-Mode Rejection Ratio
ICT	Information and Communication Technology
NMC	Nested Miller Compensation
GB	Gain Bandwidth
USTH	University of Science and Technology of Hanoi

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Driver fatigue is a leading cause of traffic accidents, especially in scenarios that demand constant attention such as long-distance driving. Traditional approaches to detecting drowsiness often rely on hand-crafted features like the Eye Aspect Ratio (EAR) or Mouth Aspect Ratio (MAR), which can be unreliable under varying head poses and lighting conditions. This thesis proposes a real-time driver drowsiness detection system that combines facial landmark extraction with deep learning techniques. MediaPipe is used for precise and efficient localization of eye and mouth regions from webcam input, while ResNet18 is employed to classify their states (open/closed and yawning). To capture the temporal dynamics of driver behavior, a Long Short-Term Memory (LSTM) network is integrated, enabling detection based on patterns over time rather than isolated frames. Unlike traditional geometry-based methods, the proposed system works on standard webcams without requiring any specialized equipment. It operates in real time and includes a visual alert interface to warn users upon detecting signs of fatigue. Experiments on the MRL Eye Dataset and Yawn Dataset demonstrate that the proposed approach achieves classification accuracy above 98% for eye and mouth states and over 95% for sequence-based fatigue detection using LSTM, confirming its effectiveness and robustness in practical scenarios.

I/ INTRODUCTION

1. Global context

In the context of increasingly developing traffic and increasing traffic volume, driving safety becomes extremely important. Among all the causes of traffic accidents, fatigue and drowsiness while driving are among the leading causes of death but are very difficult to detect in time with the naked eye. According to the World Health Organization (WHO), road traffic accidents cause about 1.19 million deaths worldwide each year and cause 20 to 50 million non-fatal injuries, and are the leading cause of death in people aged 5 to 29 [1].

Fatigue significantly reduces the ability of traffic participants to concentrate, react, and handle situations, making the driver no longer have good control of the vehicle. While speeding or alcohol use violations can be handled with specific measurement tools, fatigue recognition is a major challenge due to its subjectivity and changes in human psychological state over time.

Faced with this practical challenge, the development of real-time driver fatigue monitoring systems is becoming an inevitable trend in intelligent driving assistance systems (ADAS—Advanced Driver Assistance Systems). Among them, facial expression analysis, especially of the eye and mouth areas, is considered a potential solution thanks to its non-invasive nature and ease of deployment on common camera devices.

2. Literature review

There are three main strategies commonly used to detect driver fatigue: physiological signal-based methods, driving behavior analysis, and facial behavior monitoring. Each of these strategies employs different techniques and technologies, with specific advantages and limitations depending on the application context.

The first strategy, physiological signal-based detection, utilizes internal biological indicators such as EEG (brain waves), ECG (heart activity), and GSR (skin conductance) to assess the driver's level of alertness. For instance, Jap et al. (2009) applied a fuzzy-SVM model to classify fatigue using EEG spectral features, including alpha and theta wave patterns [2]. While this method offers high accuracy and enables early detection of drowsiness, it requires drivers to wear specialized sensors or headgear while driving. This makes it inconvenient and impractical for large-scale, real-world deployment.

The second strategy involves analyzing driving behavior, such as steering patterns, lane deviation, and pedal usage. Although this approach can detect changes in a driver's interaction with the vehicle, it typically requires additional in-vehicle sensors and is less frequently used. Furthermore, these behaviors can be heavily influenced by external conditions like traffic or road quality, which limits the reliability of this approach for fatigue detection.

The third and most widely adopted strategy is facial behavior-based analysis. This approach is non-intrusive, utilizes cameras already present in many modern vehicles, and focuses on interpreting facial cues such as eye closures, blink rates, yawning, and facial expressions. Traditional methods in this category often rely on geometric measurements. For example, Soukupová and Čech (2016) proposed the Eye Aspect Ratio (EAR) to detect eye closures by calculating distances between eye landmarks, while the Mouth Aspect Ratio (MAR) is used to identify yawning behavior [3]. However, these techniques are sensitive to changes in lighting, camera angle, and occlusions such as eyeglasses. Additionally, they depend on manually set threshold values, which reduces their adaptability across different environments and subjects.

To overcome these limitations, recent research has shifted towards deep learning-based methods. These models, especially Convolutional Neural Networks (CNNs), can automatically extract relevant features from facial images without relying on

handcrafted rules. For example, Florez et al. (2023) built a CNN-based model that achieved 99.7% accuracy in classifying eye states using the NITYMED dataset [4]. Furthermore, Gomaa et al. (2023) combined CNN and Long Short-Term Memory (LSTM) networks to analyze sequences of eye state changes over time, achieving over 97% validation accuracy on the NTHU dataset [5]. Another notable approach by Abtahi et al. (2014) integrated multiple features—eyes, mouth, and head posture—to assess fatigue comprehensively. However, the system required high-resolution cameras and powerful processing units, making real-time deployment in standard vehicles difficult [6].

From the review of existing methods, it is evident that geometry-based techniques like EAR and MAR, while simple and efficient, lack robustness under varying environmental conditions. In contrast, deep learning models such as CNNs offer strong adaptability, better generalization, and require no manual threshold setting.

In this study, we adopt the deep learning approach by employing a CNN-based model to directly classify eye and mouth states from video frames. By eliminating the need for handcrafted geometric features like EAR or MAR, our system is designed to function reliably in diverse lighting conditions and facial variations. It is lightweight enough to be deployed using standard webcams, making it suitable for practical use in real driving scenarios.

3. Main questions

From the above facts and overview, the scientific question raised in this study is: How can a driver fatigue detection system operate in real time using only a conventional camera and deep learning methods—specifically analyzing eye and mouth states through convolutional neural networks—without relying on traditional geometric indicators like EAR or MAR, while maintaining high accuracy under varied lighting conditions and driver postures? To address this question, the study aims to explore the integration of deep learning models, real-time image processing, and visual alert mechanisms to deliver an effective and practical driver fatigue monitoring solution.

4. Objectives

In this project, a CNN model will be trained to classify whether the driver's eyes are open or closed and whether the mouth is normal or yawning by using cropped camera images. A real-time video processing system will also be built, applying MediaPipe to locate the eye and mouth regions, crop the images, and send them to the CNN model for continuous detection. The system will continuously save the driver's status over time and examine these sequences to identify signs of chronic fatigue, either by applying logical rules or using RNN models. In the end, a basic web interface will be designed to show the driver's condition and provide alerts whenever fatigue is detected.

II/ OBJECTIVES

The main objective of this project is to develop a driver fatigue detection system that operates by directly analyzing eye and mouth conditions from live facial images using a CNN-based model. The system aims to classify the state of the eyes (open or closed) and the mouth (normal or yawning), then process these results over time to spot possible signs of ongoing or chronic tiredness. The whole system is designed to work efficiently with a standard webcam and present the driver's current condition, along with real-time alerts and notifications, through a straightforward and easy-to-use web interface.

III/ MATERIALS AND METHODS

1. Proposed system, method

As mentioned in the previous section, this project integrates both existing techniques and newly proposed components to build an eye and mouth feature recognition system for assessing driver fatigue. The system consists of three main modules: (i) Real-time classification of eye and mouth states using pretrained deep learning models; (ii) Sequential behavior analysis over time using a custom-tuned LSTM architecture; (iii) A real-time web-based interface to display drowsiness status and issue warnings.

Specifically, the system inherits the use of the MediaPipe Face Mesh framework for accurate facial landmark extraction, and leverages a pretrained ResNet-18 CNN model for classifying eye and mouth states.

In contrast, the proposed contributions include a custom pipeline that processes real-time webcam input, performs Region of Interest (ROI) extraction, and applies CNN + LSTM-based fatigue prediction without relying on traditional geometric indicators like EAR (Eye Aspect Ratio) or MAR (Mouth Aspect Ratio). Additionally, a dedicated user interface was designed to visualize driver status in real time and to provide fatigue alerts in a simple and intuitive way.

By combining existing tools with novel integration strategies and removing the need for manually defined thresholds, the system is designed to operate reliably in diverse real-world driving scenarios.

1.1. Proposed system

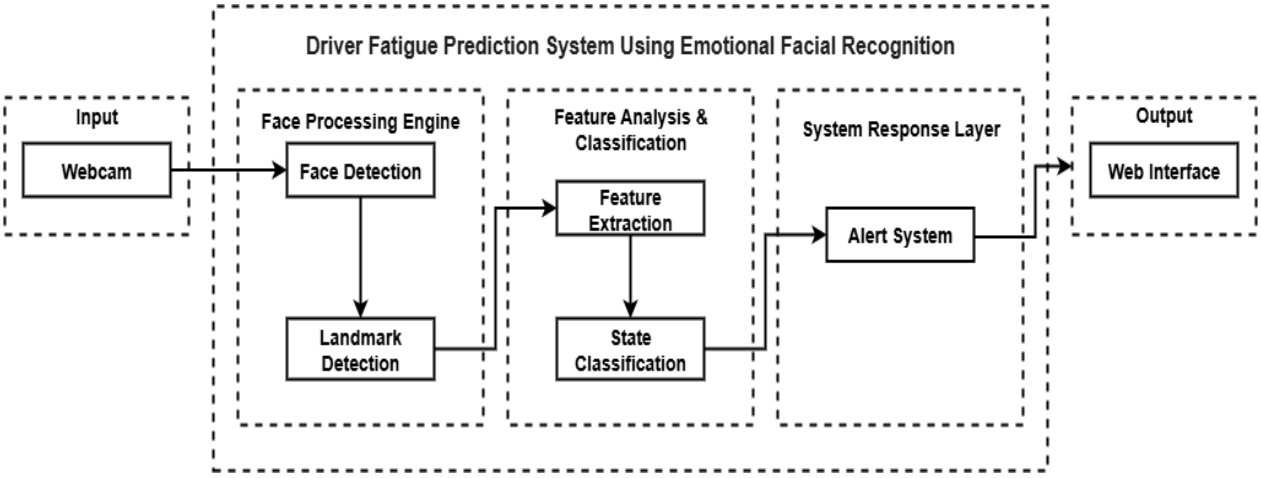

Figure 1: The block diagram of the proposed system architecture.

In this project, the system is designed with a clear layout, following a structured model as seen in the architecture diagram. It takes a modular approach to handle real-time driver fatigue detection through facial emotion analysis. The whole system consists of four main parts: Input, face processing, feature analysis with classification, and the feedback layer. The final results are shown through a web interface.

First, the webcam captures real-time facial images of the driver and sends them to the Face Processing Engine. This part of the system is responsible for detecting the face and then finding specific facial landmarks. The face detection step looks for the presence and position of the face in each frame, and the landmark detection finds key points such as around the eyes, mouth, and nose. These features are later used to analyze whether the driver is alert or tired.

After detecting the facial landmarks, the system moves on to the Feature Analysis & Classification stage. In this part, important facial features are taken and passed into a classification model to figure out the driver's current state. By looking at things like eye closure and how often the driver yawns, the system decides whether the driver is alert or showing signs of fatigue.

Once the system figures out the driver's condition, the result goes to the Response Layer, where a warning is sent if tiredness is detected. This feature is meant to help avoid

possible accidents by letting the driver know in time that they are too drowsy to continue safely.

At the last step, the prediction output is transferred to the web interface, where the driver's condition is displayed in a clean and easy-to-understand format. This feature allows people such as fleet supervisors to check the driver's status in real time and respond quickly if needed.

1.2. Proposed method

MediaPipe is a machine learning framework built by Google Research to help create real-time computer vision applications across different platforms [7]. It is made of small components called "calculators" that are connected like a flowchart. Each calculator has its job, such as detecting features, running models, or changing data formats. This makes MediaPipe easy to use and flexible, with good performance, especially because it can use a GPU on many devices. It works on Android, iOS, the web (with TensorFlow.js), and desktop systems.

ResNet-18 is a kind of deep learning model that people often use for image classification. It was introduced back in 2015 by He et al. [8], and it's known to work well, even though it's not too large. What makes it different from older CNN models is that it has something called "skip connections"—basically, some information is passed across layers without going through all of them. This way of connecting layers helps stop the problem where the gradients get too weak when training deep models. Thanks to that, the model can still be trained properly, even when it has more layers. ResNet-18, like the name says, has 18 layers. It starts with a normal convolution layer, and then it goes through four stages. Each of those has two blocks inside. The setup is pretty efficient because it lets the model understand both basic and more detailed features, but without making it too large or slow.

LSTM is a special kind of RNN that was introduced by Hochreiter and Schmidhuber back in 1997 [9]. It was made to deal with long-term information in data that comes in sequences. Normal RNNs usually have trouble remembering things for a

long time—they either forget the info or the gradients get messed up during training. To solve this issue, LSTM includes memory units and several gates that decide what data to keep and what to forget. These gates include input, forget, and output gates. Because of this design, LSTM can remember important things for a longer time, which is helpful when working with time-related data like sequences or time series.

Compared to traditional fatigue detection systems using geometric indicators such as EAR and MAR, the proposed method avoids the need to define threshold values and shows better robustness in real-world conditions. ResNet-18 is selected for its balance between performance and computational efficiency, making it suitable for deployment on standard devices. Meanwhile, LSTM is chosen over GRU due to its better capability of retaining long-term dependencies, which is essential for capturing patterns like prolonged eye closure or frequent yawning. This combination allows the system to detect fatigue more accurately and continuously in real-time, without relying on handcrafted features or high-end sensors.

2. Methodology and Technology Stack

2.1. Methodology

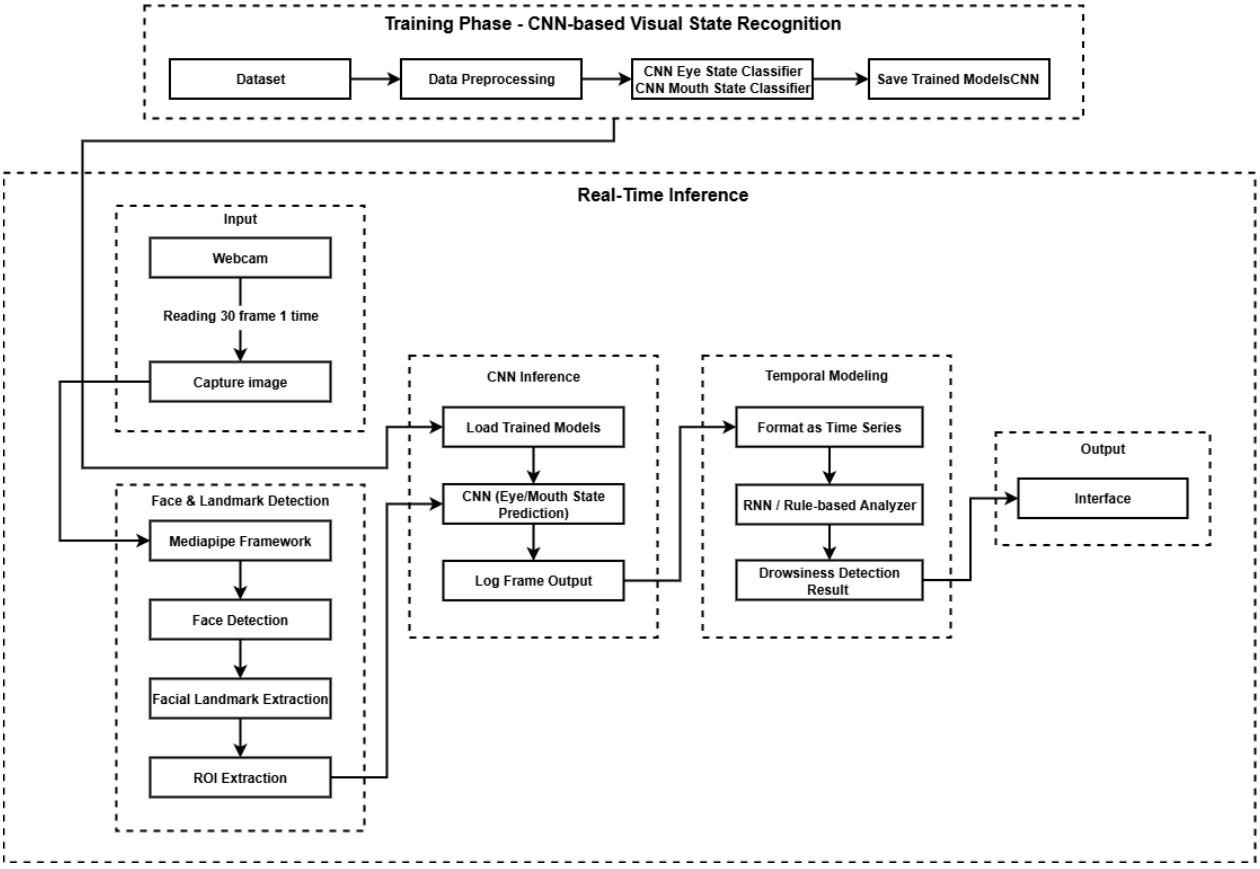

Figure 2: The overall pipeline of the architecture of the system.

The whole process is handled in two main phases. The first one is the training phase, where the model for classifying visual states is trained. It begins by collecting a large number of images showing drivers' faces, each one already labeled. These labels show two important things: whether the driver's eyes are open or closed, and whether the mouth is yawning or not. The data is then put through the data preprocessing step. The images are normalized in size (e.g., 112 × 112 pixels), adjusted for brightness/contrast, and data augmentation techniques are applied to minimize overfitting. At the same time, the dataset is divided into two parts: the training set and the validation set. Next, two separate CNN models are trained for the eyes and the mouth. After training, the CNN models are saved as "Trained Model Files" to serve the actual deployment phase.

Phase two (Real-Time Inference) is the process of applying the trained models to

the real environment to detect driver fatigue in real time. Input data is collected from the webcam mounted on the vehicle. Continuously read 30 frames at a time to ensure the smoothness and continuity of the recognition process. Each frame will be saved for processing in the next steps. Using the MediaPipe Framework to perform two important tasks: face detection from the frame and extracting facial landmarks (facial landmark extraction) to determine the location of the eyes and mouth. Then, ROI extraction is performed to cut the eye and mouth regions from the original frame. The extracted ROIs are fed into two previously trained CNN models (loaded from the archive files in the training phase). CNN performs eye and mouth state classification for each frame. The system stores the eye and mouth state from every video frame, along with the time it happened. All of this is saved into the LogFrame Output so that it can be analyzed as a time series. Then, the system uses either an RNN model or a rule-based method to go through the sequences and find signs that the driver might be sleepy or tired. After this step, it produces the Drowsiness Detection Result, which shows the current state of the driver's alertness. This result is finally displayed through a web interface made with Flask.

2.2. Technology Stack

2.2.1. MediaPipe Framework

The feature extraction of the eye and mouth regions is handled overall through two main steps: face detection and face mesh.

MediaPipe Face Detection is developed based on the combination of modern deep learning models and traditional image processing techniques in the field of computer vision [10]. Specifically, MediaPipe Face Detection using Convolutional Neural Networks (CNNs) models, which are one of the most widely used deep neural network architectures in image classification and object detection problems. In particular, MediaPipe Face Detection applies models that have been optimized in the direction of Lightweight and Mobile-friendly, for example, Single Shot Multibox Detector (SSD) is a famous network architecture used for Object Detection problems, allowing detection of multiple objects in

a single inference (single forward pass) at high speed. SSD played an important role in the first versions of MediaPipe's face detection system. But since SSD is a bit heavy for mobile use, Google later developed lighter networks like MobileNet and BlazeFace. These were made to be fast and efficient for detecting faces on mobile phones and other small devices. BlazeFace is an integral part of MediaPipe Face Detection today.

MediaPipe Face Mesh is developed based on advanced technology and research platforms [11]. BlazeFace is a lightweight CNN model dedicated to real-time face detection, introduced by Google in 2019. BlazeFace provides fast and accurate input for the next landmark detection steps in MediaPipe Face Mesh. MediaPipe Face Mesh doesn't use the old way of segmenting and classifying each facial landmark. Instead, it uses regression to predict all points at the same time, which is faster and gives better results. During training, Google used 3D face scans and let the model learn patterns on its own with self-supervised learning. That way, they didn't have to label every single point. The system is also built using MediaPipe's graph style, where different steps are connected in order to make everything run smoothly and quickly.

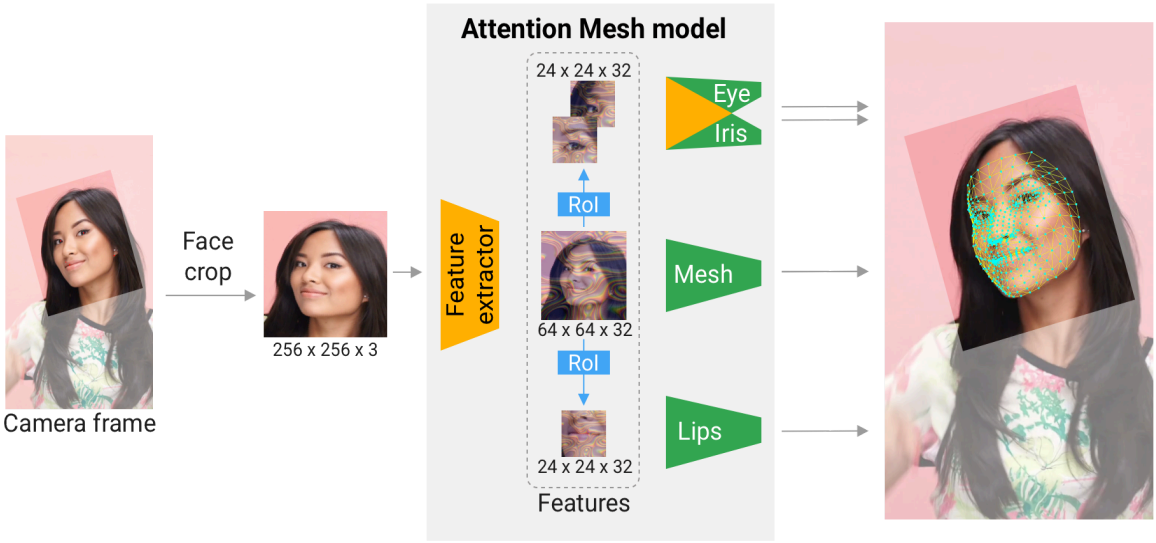

Figure 3: MediaPipe Face Mesh model with Attention Mesh Model.

The MediaPipe Face Mesh (Attention Mesh Model) architecture starts with the system input as a camera frame, containing the entire image of the user's face. Before entering the main model, the system performs Face Crop, cutting the face area based on the results from the face detection module (Face Detection). The normalized size of the face area after cropping is 256×256×3 pixels (RGB color image). The cropped image will be fed into a Feature Extractor, which is responsible for extracting features from the image. The result is a set of feature maps of intermediate sizes (e.g., 64×64×32 and 24×24×32), representing the image information needed for the next steps. The Attention Mesh Model is the core of the architecture, consisting of 2 feature layers with different processing steps. At each layer, the system uses RoI pooling or RoI selection to focus on particularly important areas such as the eyes and lips. Parallel processing branches, such as Mesh Branch, will create Face Mesh—that is, a set of 468 landmarks covering the entire face surface. The Eye & Iris Branch focuses on the eye and iris areas, used for tasks that require high precision in the eye area, such as gaze tracking or blink analysis. Lips Branch analyzes lip features, serving tasks such as lip movement analysis and mouth opening/closing state (related to yawning, talking, etc.). Finally, the model outputs a 3D mesh map (face mesh) that is displayed directly on the user's face. Landmarks are connected by wireframes, fully representing the 3D spatial structure of the face.

2.2.2. Classification: Eye and Mouth

In the field of computer vision, convolutional neural networks (CNNs) have made many breakthroughs in image recognition. However, as the network becomes deeper, the training performance degrades—a phenomenon called the “degradation problem”. To overcome this problem, He et al. (2015) proposed the Residual Network (ResNet) architecture—a model that uses “skip connections” to help the model learn residual functions instead of directly learning the output mapping. ResNet-18 is the simplest version of the ResNet family, consisting of 18 trainable layers. Despite its “lightweight” nature, ResNet-18 still outperforms traditional deep networks such as VGG-16 thanks to its smart design and simplicity in optimization.

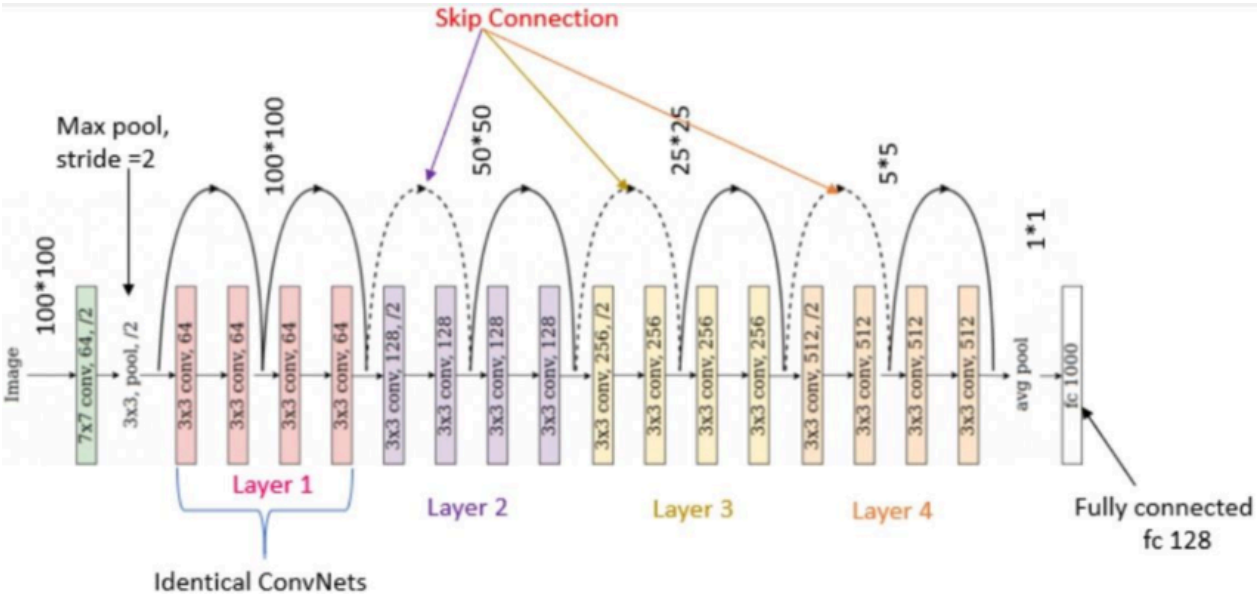

Figure 4: Original ResNet-18 Architecture.

Stage	Layer Name	Details	Output Size
Input	-	Input image 100×100 or 224×224	100×100 (example)
Conv1	7×7 conv, 64, stride=2	Large kernel filter to start feature extraction	~50×50
Max Pooling	3×3, stride=2	Reduces resolution to lower computational load	~25×25
Layer 1	2 residual blocks	3×3 conv, 64 filters × 2 times	25×25
Layer 2	2 residual blocks	3×3 conv, 128 filters, downsampling in 1st block	13×13

Layer 3	2 residual blocks	3×3 conv, 256 filters	7×7
Layer 4	2 residual blocks	3×3 conv, 512 filters	4×4 or 5×5
Global Avg Pool	-	Converts the entire feature map to a 1×1 vector	1×1×512
Fully Connected	Linear → 128 (or 1)	Classification or final feature representation	128-d

Table 1: Overview of ResNet-18 Architecture.

One of the most important contributions of ResNet is the residual block design, which effectively overcomes the performance degradation when the network becomes too deep. Instead of directly learning a function that maps input to output, as in traditional convolutional networks, the residual block allows the network to learn a residual function between the input and output. Specifically, instead of learning H(x), the network will learn the function $F(x)=H(x) - x$, from which the final output will be

$$H(x)=F(x)+x$$

where x is the input to the block, F(x) is the result after the convolutional layers inside the block, and F(x) + x is the element-wise addition between the input and the learned residual. Each residual block in ResNet-18 includes two 3×3 convolution layers, two batch normalization layers (BatchNorm), a ReLU activation function after the first layer, and a skip connection that passes the entire input from the beginning to the end of the block.

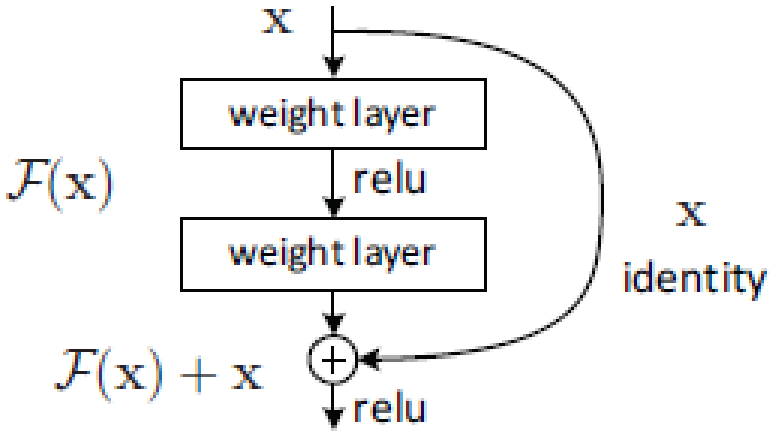

Figure 5: Residual Block Structure in ResNet.

When the input and output of a residual block have the same shape—like the same width, height, and number of channels—then the shortcut connection can just be added directly. But when they're different, for example, if downsampling happens or the number of channels changes, the shortcut can't be added as-is. In that case, a 1×1 convolution is used on the shortcut path so the shapes match. This setup helps the network pass information better and keeps the gradients from disappearing when training. And if the residual part F(x) is very small or close to zero, the model just learns to copy the input, so even deep networks can be trained more easily and don't get unstable. In ResNet-18, there are a total of 8 residual blocks, organized into 4 groups (Layers 1–4), with the number of filters increasing from 64 to 512 and the feature map size decreasing from 56×56 to 7×7. Each block maintains a simple yet effective structure, which is the main foundation for deep training of ResNet while still ensuring high accuracy.

In my setup, I used the ResNet-18 model, but instead of training it from scratch, I loaded weights that were already trained on ImageNet. After that, I made some changes so the model could handle a simple yes-or-no type of classification. To make the model fit the binary task I was working on, I didn't keep the original final layer. I changed it by adding a Dropout layer first—I used 0.3 as the rate—to help with generalization since my

dataset wasn't very big. Then I added a Linear layer with just one output, and after that, I put a sigmoid so that the final number would always stay between 0 and 1. This made it easier to figure out whether the input was showing something like closed eyes or maybe a yawn, depending on what label I was training for. Since I only needed a yes-or-no kind of answer, this setup worked pretty well.

2.2.3. LSTM

LSTM (Long Short-Term Memory) is a prominent variant of a recurrent neural network (RNN), designed to solve sequence problems with long-term memory capability. The LSTM network structure allows the system to efficiently process large data sequences thanks to the mechanism of controlling information entering and leaving the memory through control gates.

Stage	**Layer Name**	**Details**	**Output Size**
1	Input Layer	Takes input as a sequence of eye and mouth states over time (e.g., 60 frames)	(batch_size, sequence_length, input_dim)
2	LSTM Layer(s)	One or two LSTM layers with 64–128 hidden units, capable of remembering temporal information	(batch_size, hidden_dim) (Can use the last output or the whole sequence)
3	Dropout Layer	Dropout rate ~0.3 to reduce overfitting	No change in output size
4	Fully Connected	Linear layer with 1 output node	(batch_size, 1)

	Layer (Dense)		
5	Activation Layer	Sigmoid function to predict fatigue probability	(batch_size, 1)

Table 2: LSTM model overview structure.

The LSTM model architecture in the system is designed in a simple but effective multi-layer form, including main components such as the Input Layer, with the model receiving input as a time series representing the eye and mouth state in each frame. Each element in the series is a feature vector containing information such as eye state (open/closed) and mouth state (normal/yawn). With a series length of 60 frames, the input size is in the form of (batch_size, 60, input_dim). In which input_dim is usually 2, corresponding to two features: eyes and mouth. Next comes the LSTM layer, which is the core component of the model, responsible for learning and storing temporal relationships between consecutive frames. The model uses 1 to 2 LSTM layers, with the number of hidden units ranging from 64 to 128, helping the model to capture long-term behavioral trends in the state sequence. Next, a dropout layer to limit overfitting: a dropout layer with a ratio of about 0.3 is inserted after the LSTM layers, helping to increase the generalization ability of the model. The Fully Connected Layer (Dense Layer): The output from the LSTM layer is passed through a fully connected layer (Dense) with a single node, which is responsible for mapping the extracted features from the sequence into the binary classification space. Finally, the output activation layer (sigmoid activation layer): the model uses the sigmoid activation function to give the fatigue probability for the entire input sequence. The classification threshold (e.g., 0.5 or 0.7) is used to decide whether the output label is “tired” or “not tired”.

3. Experiment

3.1. Dataset

3.1.1. Data collection

The dataset here is taken from 2 datasets: MRL Eye Dataset and Yawn Dataset.

The **MRL Eye Dataset** is a forked version of the original MRL Eye Dataset, which contains infrared eye images classified into awake and sleep states. It is divided into training, validation, and test sets, consisting of over 85,000 images captured in various lighting conditions using multiple sensors. This dataset is specifically designed for tasks such as eye detection, gaze estimation, blink detection, and drowsiness analysis in computer vision. Here, I only use 34,361 images with 2 classes, divided into open and closed for training and validation with the model Resnet18.

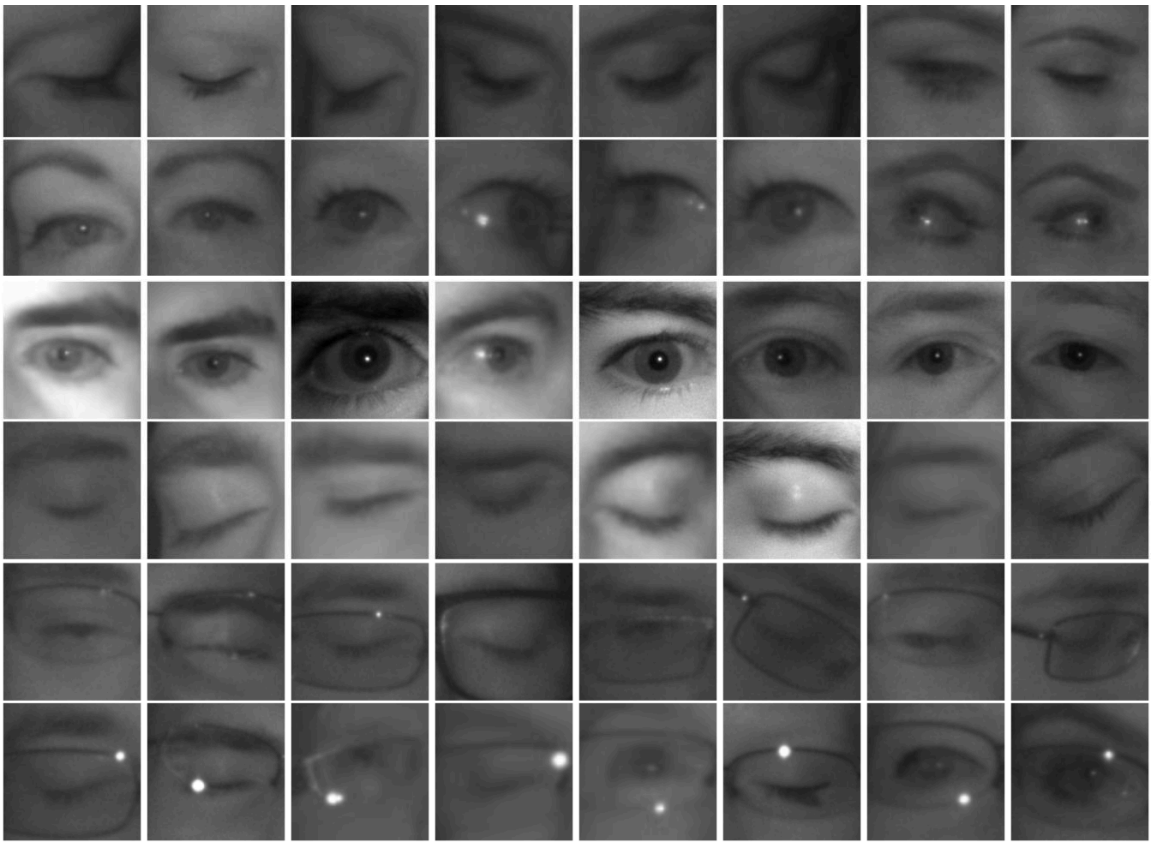

Figure 6: The MRL Eye Dataset.

The Yawn Dataset was created to classify the state of the mouth into one of two classes: open or closed. It can be used to detect whether people are yawning, talking, or singing. In this dataset, I used 5519 color and black and white images of different resolutions and divided them into 2 classes, yawn and no yawn, for training and validation with the model Resnet18.

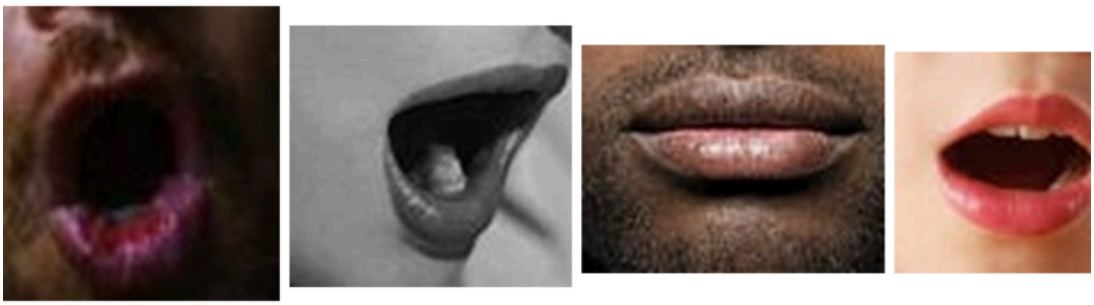

Figure 7: The Yawn Dataset.

3.1.2. Data Preprocessing

Preprocessing is performed to standardize all input images, ensuring consistency in format, size, and data structure. Each image in the dataset is converted to a standard size of 64×64 pixels, using grayscale format to reduce the number of data dimensions while still retaining basic geometric features. The processed images are saved to the output directory system in two groups: training set and validation set, corresponding to a split ratio of 80% for training and 20% for validation. Labeling of each image is done automatically based on the name of the folder containing the image. For example, images in the "open" folder will be labeled as "open eyes," while images in the "yawn" folder will be labeled as yawn. Thanks to the clear and consistent data structure, the labeling process becomes fast and accurate and does not require repetitive manual operations. After completing the data normalization and partitioning steps, the input dataset is loaded and organized in a batch format to serve the training of the deep learning model in the following steps.

3.1.3. Data Augmentation

Data augmentation techniques such as image flipping, slight rotation, and brightness and contrast adjustment are also applied to the training set to increase data diversity and improve the generalization ability of the model.

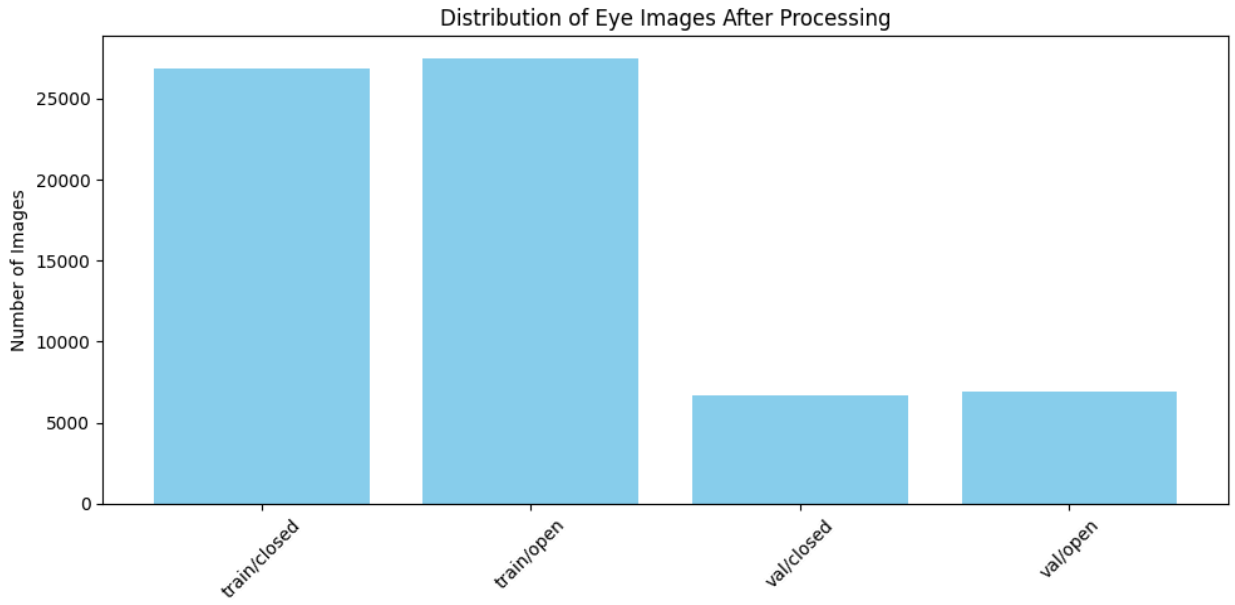

Figure 8: Distribution of Eye Images After Processing.

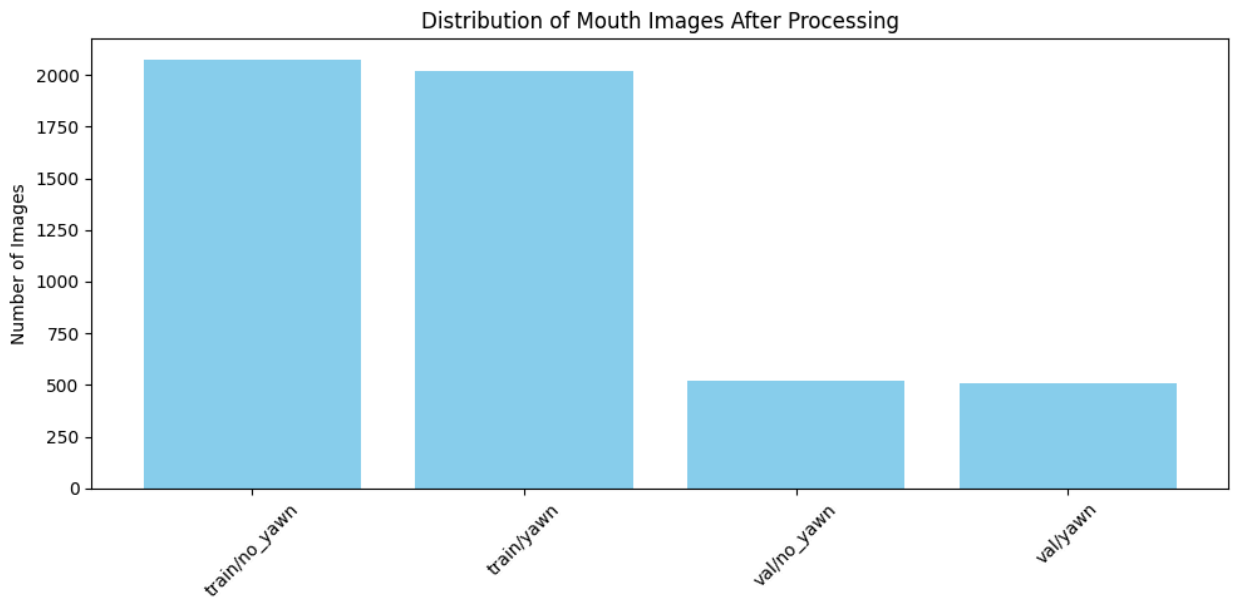

Figure 9: Distribution of Mouth Images After Processing.

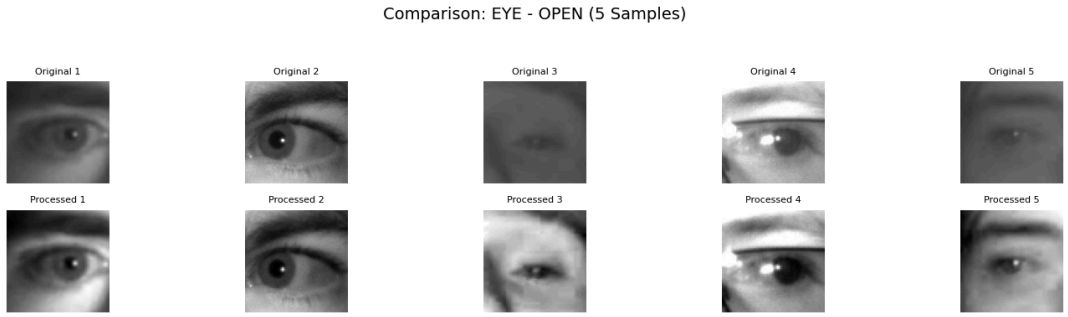

Figure 10: Eye Open After Processing.

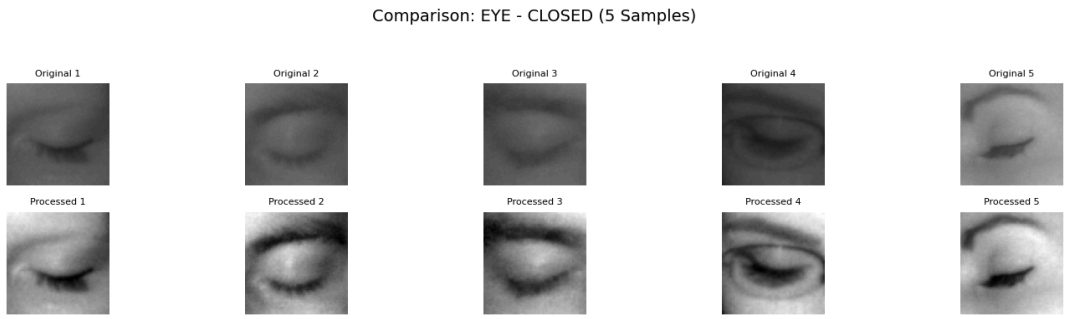

Figure 11: Eye Closed After Processing.

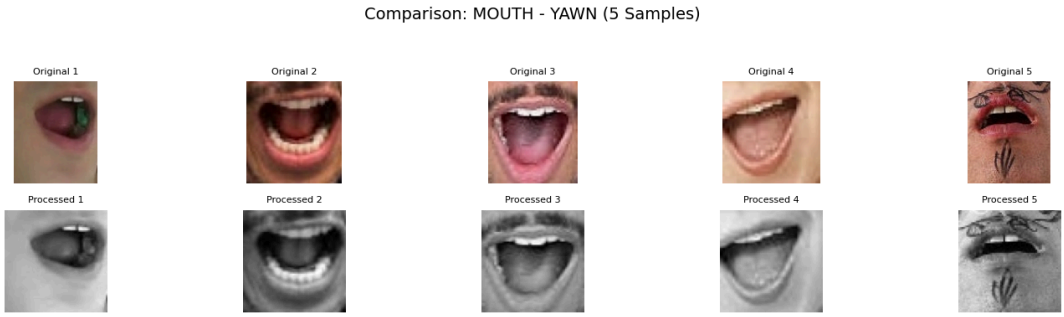

Figure 12: Mouth Yawn After Processing.

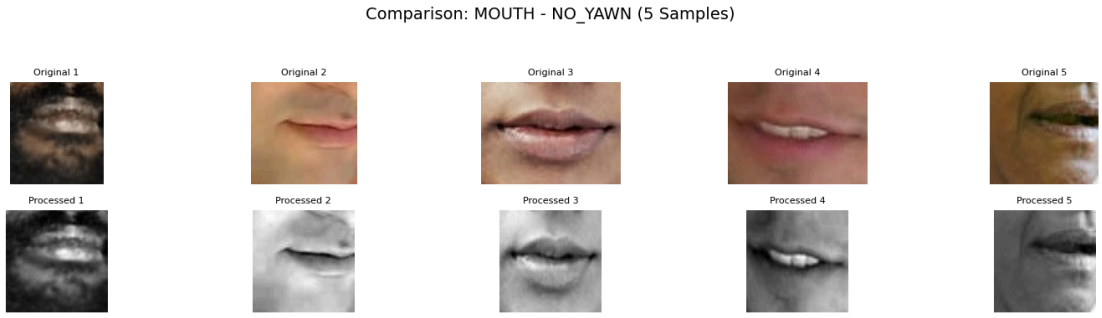

Figure 13: Mouth No Yawn After Processing.

3.2. Experiment setup

Prepare for the evaluation of the Resnet18 model for face detection and the LSTM model for recognition under standard lighting conditions of 300 lux. Our performance metrics are accuracy (The percentage of all predictions (positive and negative) that the model got correct), precision (Out of all the regions the model predicted as eyes/mouth, how many were correct), recall (Out of all the actual eyes/mouth regions, how many the model correctly detected.). F1-score (Combines precision and recall into a single balanced score, especially useful when dealing with class imbalance). These metrics are used to evaluate the model's ability to detect eyes and mouths as follows:

$$\text{Accuracy} = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$$

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

F1-score = $2 * \frac{precision*recall}{precision + recall}$

3.3. Training

After designing and fine-tuning the ResNet-18 architecture for the eye state classification problem (open/closed), the model was trained on a preprocessed image dataset. The training goal is to optimize the ability to distinguish these two states based on spatial features in the eye region image. The input images are 224×224 pixels, three color channels (RGB), resized from the original cropped images. The dataset is divided into 80% for training, 20% for testing, and loaded via PyTorch's DataLoader. During training, the model uses augmentation on the training images (flip, rotate, adjust light/dark) to increase diversity and reduce overfitting. The specific training configuration is as follows:

Parameter	**Configuration**
Model architecture	ResNet-18 (pretrained on ImageNet)
Input image size	224×224 RGB
Batch size	32
Number of epochs	30
Learning rate	0.001
Optimizer	Adam
Loss function	Binary Cross-Entropy (BCELoss)
Computing device	GPU

Table 3: Configuration of the CNN Model (ResNet-18)

Next, the RNN model is built as an LSTM (Long Short-Term Memory) network with a simple design, suitable for real-time processing environments but still ensuring the ability to learn long-term behavior. After completing the inference process using the CNN

model, the system proceeds to record the prediction results for each frame into a CSV log file. Each line in the log file corresponds to a frame, containing information including

Time (timestamp): The time the frame was processed,

Eye state (eye_state): The result of classifying the eyes as “Open” or “Closed”,

Mouth state (mouth_state): The result of classifying the mouth as “Yawning” or “Not Yawning”.

The sample data recorded after training the CNN is as follows:

Timestamp	**Eye State**	**Mouth State**
2025-06-06 14:57:51	Open	Yawning
2025-06-06 14:57:51	Open	Yawning
2025-06-06 14:57:51	Open	Not Yawning
...	...	...
2025-06-06 14:57:52	Open	Not Yawning

Table 4: Example of CNN output log data before feeding into the RNN model.

From this log, the system will group the frames into sequences of fixed length (sequence length = 30 frames). Each frame is encoded into a number before being fed into the RNN as follows:

State	**Value Label**
Eye State	
Open	0
Closed	1
Mouth State	

Not Yawning	0
Yawning	1

Table 5: Eye and Mouth State Encoding Convention for RNN.

Thus, a 30-frame log sequence is converted into a matrix of size (30×2) before being passed to the LSTM model.

Next, the RNN model is built as an LSTM (Long Short-Term Memory) network with a simple design, suitable for real-time processing environments but still ensuring the ability to learn long-term behavior.

Component	Description
Input Size	2 features per time step: eye state and mouth state
Sequence Length	30 time steps (corresponding to 30 consecutive frames)
Hidden Size	64 hidden units in the LSTM layer
LSTM Layer	1 LSTM layer, configured with batch_first=True for input shape of (batch, sequence_length, features)
Fully Connected Layer (Dense)	1 output node
Activation Function	Sigmoid—outputting a probability between 0 and 1
Output	A single probability value indicating the likelihood of driver drowsiness

Table 6: The specific architecture of the DrowsinessRNN model.

Each time of inference, the model receives a fixed-length sequence of 30 time steps; each step is a vector of 2 values (eyes, mouth). The data is normalized in the form

Eye state: 0 (open) and 1 (closed); mouth state: 0 (normal) and 1 (yawn). The input data sequence is passed through an LSTM layer with 64 hidden units, helping the model learn temporal relationship patterns between consecutive frames, such as the number of times the eyes are closed for a long time, the frequency of yawning, and the relationship between two states in a short period of time. After LSTM, the model takes the final hidden state output (hn[-1]), passes it through the fully connected layer, and then through the sigmoid function to transform it into the probability that the driver is tired. If the output probability > 0.5, the system predicts that the driver is tired. Conversely, if ≤ 0.5, the prediction is not tiring.

IV/ RESULTS AND DISCUSSION

1. Mediapipe framework

To ensure accurate extraction of facial features, the system uses a set of 468 3D landmarks provided by MediaPipe Face Mesh. These landmarks are evenly distributed across the entire face, covering important areas such as the eyes, nose, mouth, and facial contour. As a result, the system can easily determine the exact coordinates of ROI (Region of Interest) regions such as the left eye, right eye, and mouth. The figure below illustrates the network structure of the facial landmarks, clearly showing the spatial connection between the main functional regions. This structure helps ensure that the system can accurately identify the eyes and mouth, even when the driver makes small movements or changes in posture.

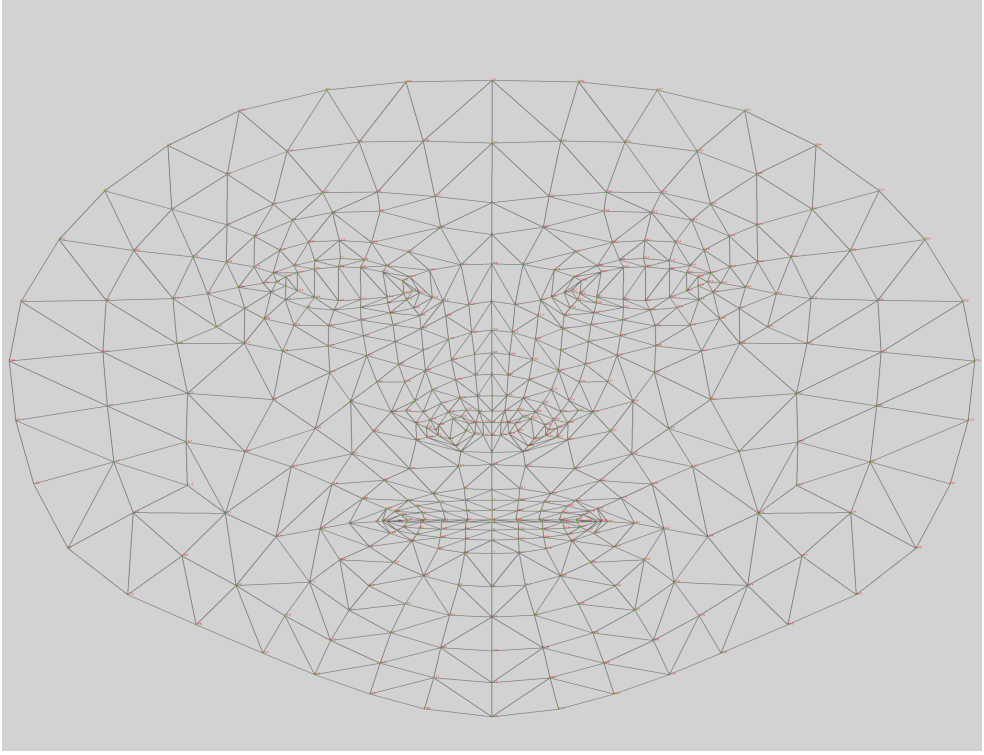

Figure 14: 3D facial landmark mesh structure detected by MediaPipe Face Mesh.

To evaluate the performance of MediaPipe Face Mesh, this study tested its ability to recognize facial landmarks under standard lighting conditions. Experimental

observations and evaluation results show that MediaPipe Face Mesh provides very stable and accurate results in the eye and mouth regions.

Figure 15: Landmark locations of the eye and mouth areas on the face image after processing with MediaPipe.

(a)

(b)

Figure 16: Tilt right **(a)** and left **(b)** when assigning landmark positions to eyes and mouth.

During continuous testing of multiple frames (frame sequences), the landmarks did not show any jumps or strong fluctuations between consecutive frames. When the driver tilted his head, looked to the sides, or made small changes in posture, MediaPipe maintained stable landmark locations around the eyes and mouth.

2. Training results

2.1. Eye Model

After 30 epochs of training, the ResNet-18 architecture-based CNN model achieved satisfactory training results on the eye image dataset. The figure below shows the loss and accuracy curves for each epoch during the training process:

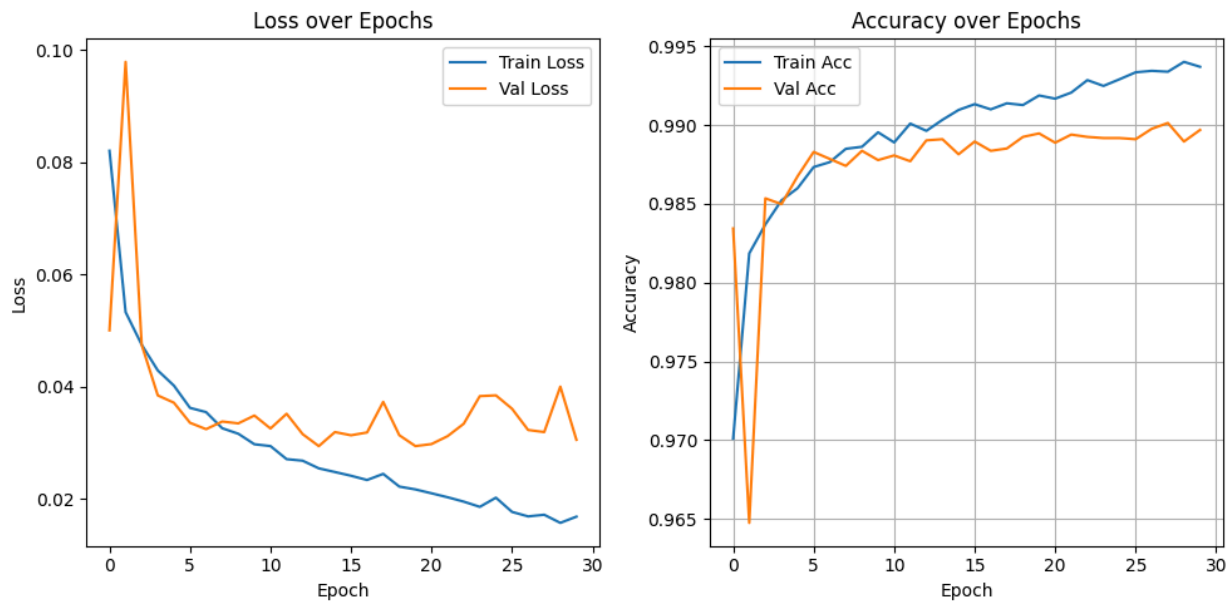

Figure 17: Loss and Accuracy progression graph by epochs during CNN model training (eye state classification).

From about epoch 5 onwards, the model achieved accuracy above 98% on both the training and validation sets. By the end of training (epoch 30), the accuracy was approximately 99.5% on the training set and approximately 99% on the validation set. The difference between train accuracy and val accuracy is very small, indicating that the model is not severely overfitting. The loss on the training set decreases steadily over the number of epochs, from about 0.08 initially to about 0.01. The loss on the validation set fluctuates initially but then also decreases and stays low (~0.03 to 0.04), indicating good convergence. The loss and accuracy curves between the training and validation sets tend to go hand in hand, without any strong fluctuations or large differences. This shows that the ResNet-18 CNN model has learned well, with good generalization ability on the testing set.

2.2. Mouth Model

The mouth state classification CNN model (normal/yawn) was trained for 15 epochs using the same ResNet-18 architecture as the eye model. The input data were

preprocessed mouth images, with two target labels. The figure below illustrates the process of loss reduction and accuracy increase during training:

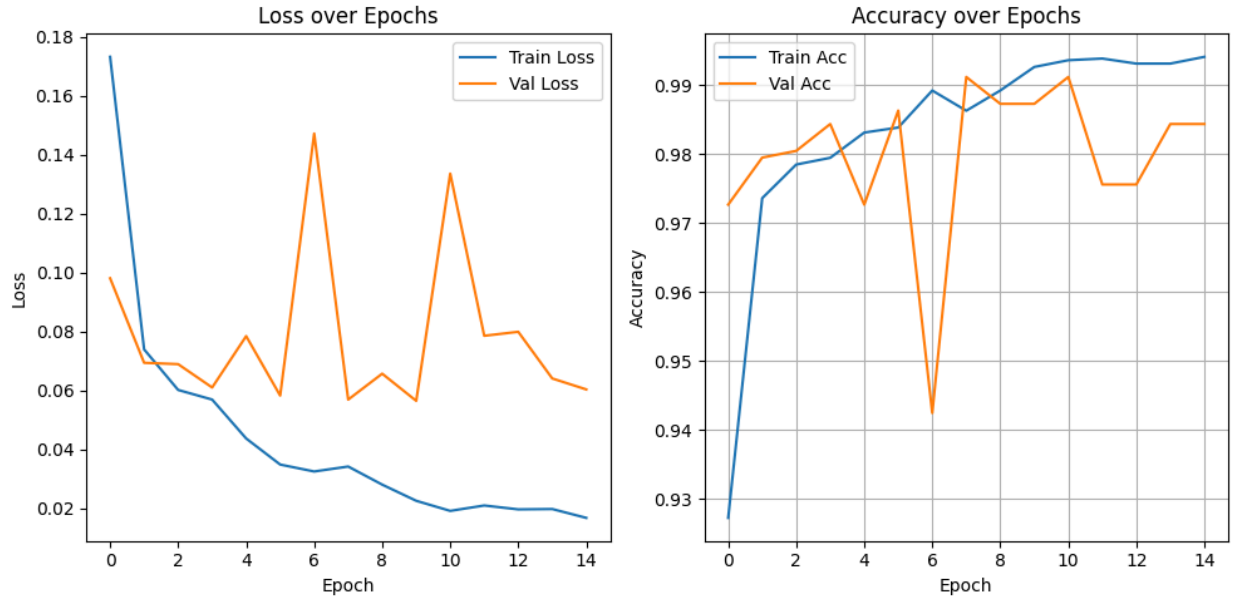

Figure 18: Loss and Accuracy graph by epochs during CNN model training (mouth state classification).

The accuracy on the training set increased rapidly and reached nearly 99% after only 10 epochs. On the validation set, the accuracy fluctuated around 98–99%, showing that the model had good generalization ability. The loss on the training set decreased steadily over each epoch, from ~0.20 initially to ~0.02 at epoch 14. The loss on the validation set fluctuated a bit but remained low (averaging around 0.05–0.10), indicating stable convergence. Although the loss curve on the test set fluctuates slightly between epochs, there is no obvious overfitting, thanks to the application of dropout and data augmentation during training.

3. CNN model performance evaluation

3.1. Eye classification model

The results of testing the CNN model to classify eye states are evaluated in detail through the confusion matrix on the test dataset. This matrix helps reflect the model's

ability to distinguish between two eye states (open for class 0 and closed for class 1). The results on the test set for the eye classification model show:

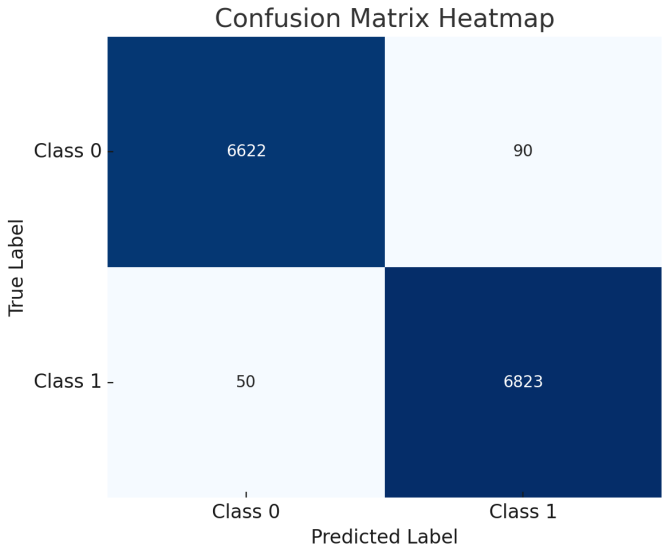

Figure 19: Confusion Matrix of the Eye State Classification Model.

CNN-based	Class Name	Precision	Recall	F1-Score	Accuracy
ResNet18	Binary (0/1)	0.9870	0.9927	0.9898	0.9897

Table 7: Metrics in Training of Eye (Validation).

The CNN model for classifying eye states achieved very high performance on the test set with an accuracy of nearly 99% and an F1-score of about 99%. The number of false positives (90 false positives and 50 false negatives out of nearly 13,500 samples) was very low, indicating that the model is capable of accurately distinguishing between open and closed eye states. The high sensitivity ensures that the system does not miss closed-eye cases, which is especially important for fatigue detection applications.

3.2. Mouth classification model

For the mouth state classification task (normal/yawn), the ResNet-18 CNN model is trained and evaluated on the test set with two classes: Class 0: Normal mouth, Class 1: Yawn mouth.

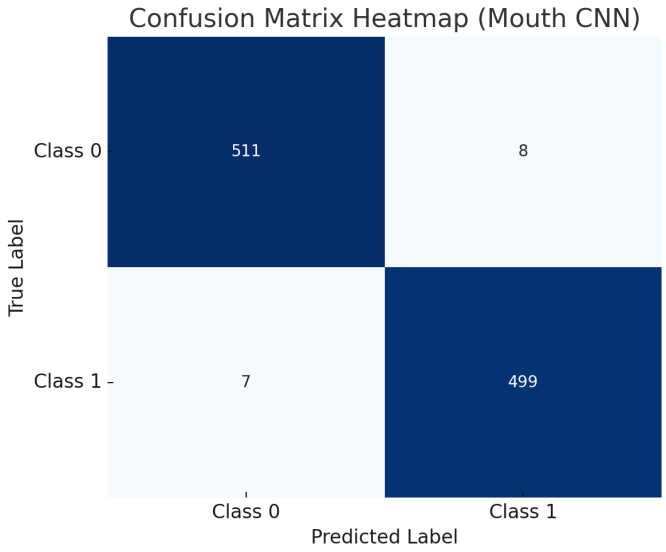

Figure 20: Confusion Matrix of the Mouth State Classification Model.

CNN-Based	Class Name	Precision	Recall	F1-Score	Accuracy
ResNet18	Binary (0/1)	0.9842	0.9862	0.9852	0.9854

Table 8: Metrics in Training of Mouth (Validation).

The mouth state classification model achieved very high accuracy (98.5%) on the test set. The number of false positives (only 8 false positives and 7 false negatives out of 1025 samples) was low, demonstrating that the model was able to distinguish between normal and yawning mouth states well. In particular, the high sensitivity ensured that the system did not miss yawns—an important sign in fatigue recognition. Compared to the eye classification model, the mouth model had fewer total samples but still achieved nearly equivalent performance, demonstrating that the ResNet-18 CNN architecture performed well on both tasks.

To provide a better understanding of the model performance, we compare the results of our system with several related studies that also focus on drowsiness detection using facial cues.

Study	Method	Dataset	Accuracy	Notes
Florez et al. (2023) [4]	CNN	NITYMED	99.7%	Focused on eye state only
Gomaa et al. (2023) [5]	CNN + LSTM	NTHU	97.3% (val)	Used temporal eye sequences
Our model	CNN (ResNet-18) + LSTM	MRL Eye, Yawn	98.9% (Eye CNN), 98.5% (Mouth CNN), 95% (RNN)	Real-time capable; uses low-cost webcam; flexible system

Table 9: Summarizes the accuracy, datasets, and methods used.

As shown in the table, our model achieved comparable accuracy to other state-of-the-art methods. Although the dataset used in this study is not identical to those in other works, the results still show strong generalization and robustness. Due to differences in datasets (e.g., lighting, resolution, data volume), direct comparison may not be entirely fair. However, the consistent high performance (>95%) of our system on both static and sequence classification tasks suggests it is reliable for real-time driver fatigue detection.

4. Results of sequence analysis using RNN

The training of the RNN model (using LSTM architecture) was performed in 20 epochs to analyze eye and mouth state sequences to predict the risk of driver fatigue. The figure below shows the evolution of the loss function and accuracy on both the training and testing sets throughout the training process:

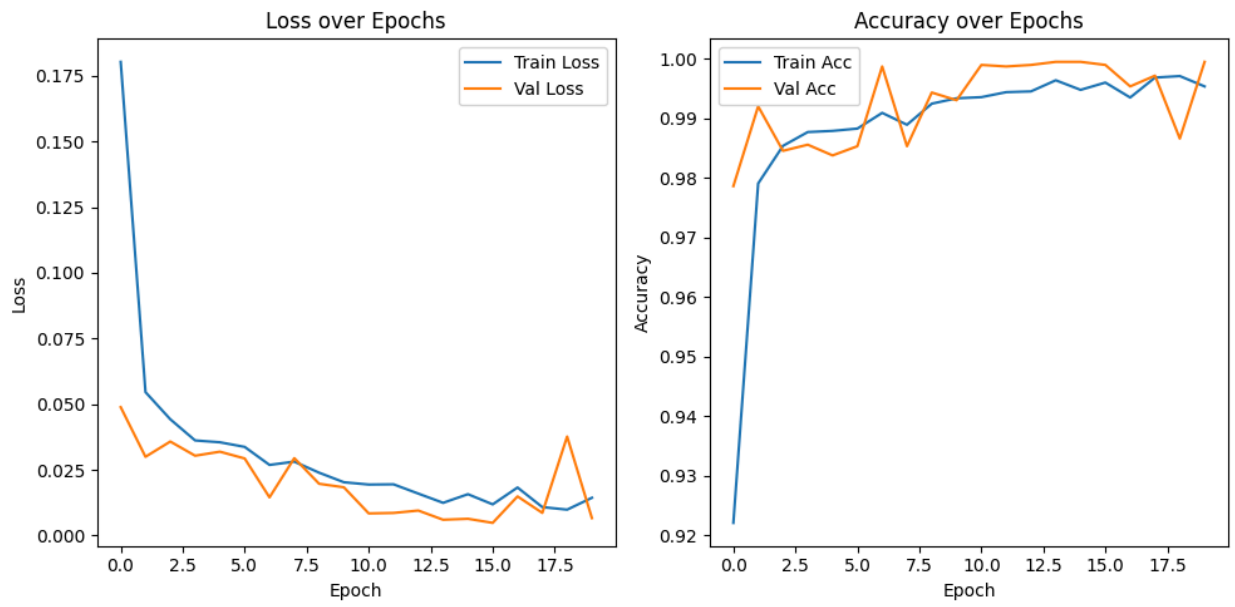

Figure 21: Loss and Accuracy graph by epochs during RNN model training.

After about the first 5 epochs, the model achieved an accuracy of over 98% on both the training and testing sets. From the 10th epoch onwards, the accuracy remained stable at around 98–99%, indicating good convergence. The loss on both the training and validation sets decreased steadily over the epochs, from the initial level of 0.18 to below 0.02 after 20 epochs. The loss on the testing set fluctuated slightly but remained at a very low level, with no signs of overfitting. The Loss and Accuracy curves on the two datasets were parallel and close to each other, indicating that the model did not experience overfitting or underfitting. In particular, the deviation between training and validation was almost negligible in the later stages.

V/ CONCLUSION & PERSPECTIVE

In this study, a complete driver fatigue detection system was successfully designed and implemented, integrating multiple modern computer vision and deep learning technologies. The system utilizes MediaPipe Face Mesh for accurate and real-time facial landmark detection, combined with Convolutional Neural Networks (CNNs) based on ResNet-18 to classify eye and mouth states from static images. Furthermore, an LSTM-based Recurrent Neural Network (RNN) was developed to analyze temporal behavioral patterns, allowing the system to evaluate driver drowsiness over time rather than relying solely on single-frame predictions.

Experimental results demonstrate that the CNN models achieved high classification accuracy (>98%) for both eye and mouth state detection on the test dataset. Similarly, the LSTM model showed a strong ability to recognize fatigue patterns across sequential frames, achieving a final classification accuracy of approximately 95% on the validation set.

The overall system is capable of real-time operation, thanks to the lightweight and optimized design of each component. Additionally, a web-based interface was implemented using Flask, providing users with real-time visualization and alerts.

This project successfully achieves the primary goal: providing an accurate, real-time, and user-friendly driver fatigue detection solution based on facial emotional recognition cues.

REFERENCES

[1] World Health Organization (WHO), Road traffic injuries. https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries/ , 13 December 2023.

[2] Budi Thomas Jap, Sara Lal, Peter Fischer, Evangelos Bekiaris, Using EEG spectral components to assess algorithms for detecting fatigue, Expert Systems with Applications, Elsevier, March 2009, Pages 2352-2359. https://doi.org/10.1016/j.eswa.2007.12.043

[3] T. Soukupová, J. Čech, Real-Time Eye Blink Detection using Facial Landmarks, in: Proceedings of the 21st Computer Vision Winter Workshop, Czech Technical University, Prague, 2016, pp. 1–8. https://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf

[4] R. Florez, F. Palomino-Quispe, R. J. Coaquira-Castillo, J. C. Herrera-Levano, T. Paixão, and A. B. Alvarez, "A CNN-Based Approach for Driver Drowsiness Detection by Real-Time Eye State Identification," *Sensors*, vol. 23, no. 2, Article 879, pp. 1–15, 2023. https://www.mdpi.com/2076-3417/13/13/7849

[5] M.W. Gomaa, R.O. Mahmoud, A.M. Sarhan, CNN-LSTM-based Deep Learning Approach for Driver Drowsiness Prediction, Journal of Engineering Research (ERJ), Vol. 6, No. 3, pp. 59–70, 2022. https://www.researchgate.net/publication/363199756_A_CNN-LSTM-based_Deep_Learning_Approach_for_Driver_Drowsiness_Prediction

[6] S. Abtahi, M. Omidyeganeh, S. Shirmohammadi, B. Hariri, YawDD: A Yawning Detection Dataset, in: Proceedings of the 5th ACM Multimedia Systems Conference (MMSys '14), ACM, Singapore, 2014, pp. 24–28.

[7] Google AI for Developers, MediaPipe Solutions guide. https://ai.google.dev/edge/mediapipe/solutions/guide

[8] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: arXiv preprint arXiv:1512.03385, 2015. https://doi.org/10.48550/arXiv.1512.03385

[9] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, *Neural Computation*, MIT Press, 1997, pp. 1735–1780. https://ieeexplore.ieee.org/abstract/document/6795963

[10] Google AI for Developers, MediaPipe Solutions Guide, MediaPipe Face Detection, *May 10, 2023.* https://mediapipe.readthedocs.io/en/latest/solutions/face_detection.html

[11] Google AI for Developers, MediaPipe Solutions Guide, MediaPipe Face Mesh, *May 10, 2023.* https://mediapipe.readthedocs.io/en/latest/solutions/face_mesh.html

[12] Akash Shingha, Media Research Lab, MRL Eye Dataset. https://www.kaggle.com/datasets/akashshingha850/mrl-eye-dataset/data

[13] David Vazquez, Yawn Dataset
https://www.kaggle.com/datasets/davidvazquezcic/yawn-dataset/data

APPENDICES

APPENDIX 1

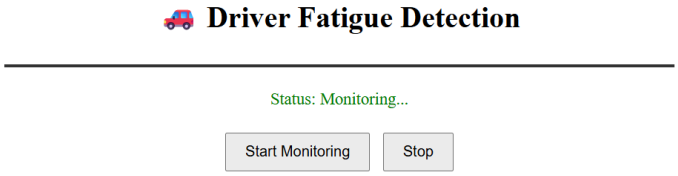

Figure 22: User interface beginning.

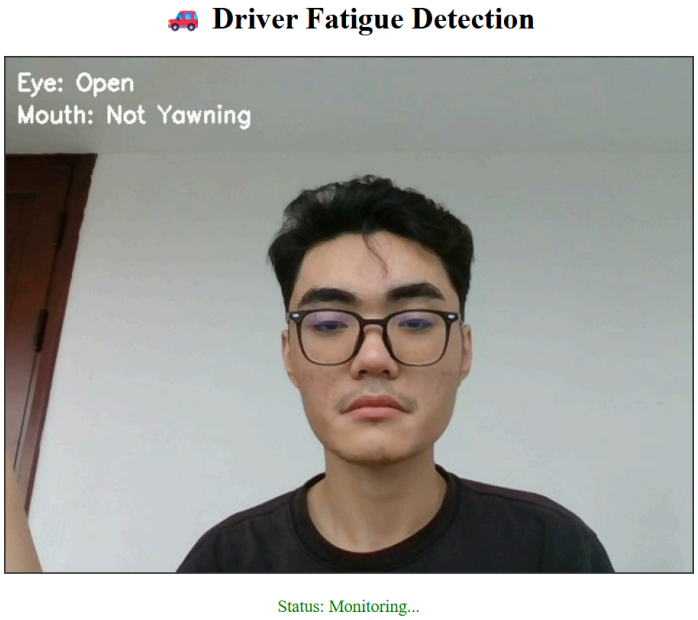

Figure 23: The user interface displays the result of the state.