
UNIVERSITY OF PISA

MASTER'S DEGREE IN
ARTIFICIAL INTELLIGENCE AND DATA ENGINEERING

Industrial Applications

# cLimaSenseTM:

an LSTM network to detect drowsiness

Professors:

**Pierfrancesco Foglia**

**Cosimo Antonio Prete**

Students:

**Francesco Pio Crispino**

**Massimiliano Romani**

Github Repository

---

Academic Year 2024/2025

# Abstract

Driver drowsiness is a critical factor in road accidents, demanding robust detection solutions. This project introduces cLimaSenseTM, an advanced real-time system, engineered to identify driver fatigue by leveraging temporal analysis of ocular cues. cLimaSenseTM employs a Long Short-Term Memory (LSTM) neural network, implemented in PyTorch, to process sequences of Eye Aspect Ratio (EAR) values extracted from live video feeds. Trained and rigorously evaluated on the public DMD (Driver Monitoring Dataset), the system is specifically designed to discern dynamic patterns in eye behavior that indicate the onset of drowsiness, offering a deeper insight than methods reliant on static or aggregated metrics. The primary objective is to provide a highly accurate and responsive early-warning system, capable of alerting drivers before cognitive impairment compromises safety, thereby making a substantial contribution to accident prevention and enhancing overall road safety.

# Index

**1 Introduction** **3**

**2 Related work** **3**
2.1 The Vicomtech Driver Monitoring Dataset . . . . . 4

**3 System Description** **5**

**4 Algorithm Description** **5**
4.1 Eye Aspect Ratio . . . . . 6
4.2 The Idea . . . . . 6

**5 Prototype and Demo Setup and Description** **8**
5.1 Raspberry Pi 3 Model B+ . . . . . 8
5.2 Camera Module . . . . . 8
5.3 Operating System Setup . . . . . 9

**6 Performance Evaluation and Optimization** **9**
6.1 LSTM Model Performances . . . . . 9
6.2 Comparative Inference Performance: PC vs. Raspberry Pi . . . . . 12

**7 Structure of the Demo** **12**

**8 Conclusions** **14**

# 1 Introduction

Drowsiness while driving is a major cause of road accidents worldwide, often leading to fatal or severely disabling consequences. According to recent estimates [4], Sleepiness while driving contributes to 3% to $>$ 30% of all road traffic accidents globally. When a driver is drowsy, their cognitive abilities and reaction times drastically deteriorate, exponentially increasing the risk of making critical errors that can lead to accidents.

In this context, developing intelligent systems capable of monitoring the driver's state of alertness and promptly alerting them in case of drowsiness is crucial for road safety. Such systems, known as Driver Drowsiness Detection Systems (DDDS), aim to identify the early signs of fatigue before they can compromise driving ability.

Among the various emerging technologies for driver monitoring, systems based on video analysis via a camera have shown particular promise. A camera, typically positioned on the dashboard or integrated into the rearview mirror, can non-invasively capture images of the driver's face and upper body. These images are then processed by computer vision and machine learning algorithms to extract relevant features associated with drowsiness. Common indicators include the frequency and duration of blinks (often measured by PERCLOS - Percentage of Eyelid Closure), yawning frequency, head tilt, gaze direction, and other behavioral cues. The advantage of a camera-based approach lies in its ability to detect direct physiological and behavioral manifestations of drowsiness, offering continuous and potentially more accurate monitoring compared to methods based on vehicle sensors or wearable devices.

This project fits into this line of research by proposing the development and implementation of a driver drowsiness detection system based on a Long Short-Term Memory (LSTM) neural network. LSTM networks, a specific type of recurrent neural network architecture, are particularly well-suited for analyzing sequential data like video streams, as they can learn long-term temporal dependencies. Our system was developed using the PyTorch framework and trained on the DMD (Driver Monitoring Dataset), a public resource specifically created for driver drowsiness research, containing videos of subjects in various states of alertness.

The objective of this work is to develop an effective LSTM-based system for recognizing driver drowsiness. We aim to demonstrate its capability to process video input from a camera, analyze relevant facial features over time, and accurately classify the driver's state of alertness. This report will detail the system architecture, the LSTM model, the dataset used, the experimental setup, and the performance evaluation of our proposed solution. Ultimately, this research contributes to the ongoing efforts to enhance road safety through advanced driver assistance technologies.

# 2 Related work

Drowsiness Detection is a hot topic in the automotive field, so there are numbers of approaches more or less effective, but instead of comparing these approaches, in this section we would like to focus on the dataset we trained our network on: **Driver Monitoring Dataset** [2]. The paramount work needed to create an accurate and extensive dataset is what primarily hinders the creation of new approaches, specially for **neural networks**, models that require tons of data.

## 2.1 The Vicomtech Driver Monitoring Dataset

The **Vicomtech Driver Monitoring Dataset (DMD)** [2] stands out as a comprehensive multi-modal resource specifically designed to advance research and development in driver monitoring systems. Developed by Vicomtech, this dataset emphasizes naturalistic driving conditions and aims to facilitate the study of various driver states crucial for road safety, including distraction, stress, and notably, **drowsiness**.

**General Dataset Content:** The Vicomtech DMD comprises data collected from **100 participants**, accumulating approximately **30 hours of driving sessions**. Its strength lies in its multi-modality, providing a rich array of synchronized data streams:

- **Video Streams:**
  - *Driver-facing RGB Camera:* High-resolution video focused on the driver's face, capturing expressions, head movements, and eye behavior.
  - *Driver-facing Infrared (IR) Camera:* Essential for robust performance in varying lighting conditions, including low light and nighttime scenarios.
  - *Front Road-facing RGB Camera:* Captures the external driving environment and traffic conditions.
  - *Rearview Mirror RGB Camera:* Provides an additional perspective of the cabin and driver.
- **Audio Recordings:** From a cabin microphone, useful for detecting audible cues like yawns or speech patterns.
- **Vehicle Data (CAN Bus):** Comprehensive information from the vehicle's controller area network, including speed, acceleration, braking, steering wheel angle, and other vehicle dynamics.

The dataset also incorporates diverse participants (age, gender, driving experience) and varied driving scenarios (different times of day, road types), enhancing its generalizability for training robust models.

Among all the possibilities, videos with labeled frames were used in the training and testing of our approach. Each frame is either labeled as:

- open: eyes are wide open, stating an awake situation;
- opening: eyes are opening from being closed, useful for blinking detection;
- closing: eyes are closing from being open, useful for blinking detection;
- close: eyes are close shut, this states a micro-sleep situation;
- undefined: for errors in labeling.

As it is possible to see, *open* and *close* labels do not represents simply a frame with eyes opened or closed as it may seem, they represent a wider concept of being awake or micro-sleeping. This is actually not explicitly described in the dataset, but rather inferred during the dataset exploration, based on these key factors:

- micro-sleep feature: in the website is possible to find **micro-sleep** as feature of the dataset, however, other than the one described, there is no further labeling.

- close state repetition: close labeled frames are never unique, they always appear in at least 6 consecutive frames.

These factors lead to the assumption that the first *close* in a sequence indicates the start of a micro-sleep session and the last *close* indicates the end of it, framing a window of severe drowsiness. This particular labeling is perfect to train a model that analyze consecutive frames to output the drowsiness state.

# 3 System Description

The Drowsiness Detection module of the ClimaSense system is composed of four components:

**Camera:** this device is mounted above the dashboard, facing the driver, so that it can capture the face of the driver with a rate of 30 frames per second.

**Car Controller:** this device is crucial for the behaviour of the system. Among all the activities it does on a regular basis, in this system it also:

- Computes the EAR values for each frame using **FaceMesh**;
- Utilizes **cLimaSenseTM** to obtain a label for the given frame;
- Controls the airflow to redirect cold air in the area of neck and lips of the driver;

**Display:** this device is used to show the driver an alert about theirs drowsiness state and the temperature of the air flowing.

# 4 Algorithm Description

This section is dedicated to the description of the algorithm and its main components; it starts with an explanation of the general idea, and follows with the details about the components

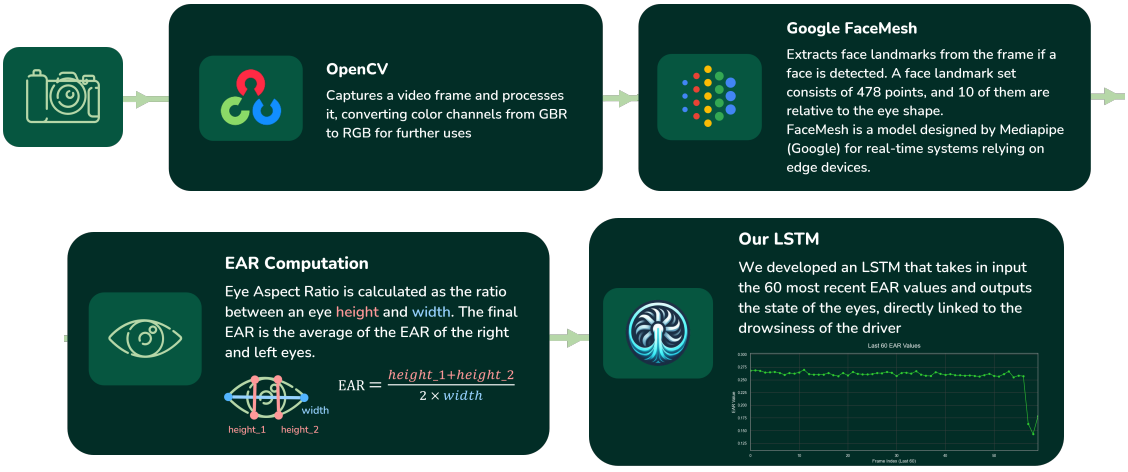


Figure 1: Algorithm's logic

## 4.1 Eye Aspect Ratio

A small camera mounted on the dashboard is one of the least invasive devices that creates data for drowsiness detection, but how can an image be used to achieve it? A standard camera takes around 30 images per second and each image is called a **frame**. One unprocessed frame offers little to no information to a computer, because the device itself is not capable of understanding images or videos like humans do. To be informative, a frame needs to be processed by a program, but writing a program capable of extracting informations from an image would be a paramount job for a human being, therefore neural networks are exploited to learn to recognize patterns to extract data. In this project we used Google's **FaceMesh**, a Convolutional Neural Network, to extract the face landmarks of a person (the driver in this case). Among all the landmarks, we are interested in the eye's ones, because they offer the possibility to compute the area of the eye, drawn around the eyelid (large area when the eyes are open, small area when the eyes are closed).

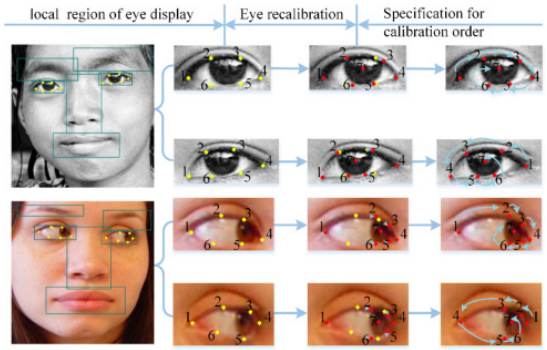


Figure 2: example of eye landmarks [1]

The Eye Aspect Ratio (EAR) is computed as follows:

$$\text{EAR} = \frac{\|P_2 - P_6\| + \|P_3 - P_5\|}{2\|P_1 - P_4\|}$$

Where:

- $P_i = (x_i, y_i)$ represents the coordinates of $i$-th eye landmark.
- $\|P_a - P_b\|$ indicates the Euclidean distance between $P_a$ e $P_b$.

## 4.2 The Idea

The system wants to detect driver's drowsiness by using a camera. There are multiple factors highlighting the drowsiness state of a person, (e.g. body movements slowing down, higher response time, yawning), but the probably most representative one is the increase in eyes closure time. One can tell if another person is drowsy or not by looking at their eyes and how much this other person keeps them close. This particular concept is the foundation of the PERCLOS approach, which was developed in 1994. It calculates the percentage of frames in which the driver has closed their eyes over the total number of examined frames; however, this approach has three main problems:

- **Classified frames**: this approach is the last step of a larger view algorithm, because it must work over already classified labels, which makes it at the same time an highly adaptive algorithm but also not easily comparable.
- **Window size fine tuning**: a small window may result in a lot of false positives, while a large window reduces the responsiveness of the system (which is a key aspect in this task).
- **Independence of frames**: There is no way to account for the sequence of values in the window, because every frame is considered as independent from the others, and this can degrade both responsiveness and accuracy.

As already said, there are a number of algorithms that can be employed to classify frames as *eyes closed* or *eyes open*, but the sequence classification is always left to PERCLOS. On the other hand, we attempted to implement a convolutional neural network (CNN) by feeding it individual frames and their corresponding labels from the DMD dataset. Although the model achieved a very high F1-score during validation ($>0.9$), it failed to detect any meaningful eye state transitions or signs of drowsiness during testing. This outcome likely stems, at least in part, from a critical limitation shared with the PERCLOS-based approach: CNNs inherently lack the ability to capture temporal dynamics, which is essential for reliable detection of drowsiness. Furthermore, the opaque nature of CNN feature extraction makes it difficult to interpret what visual patterns the network has learned, further undermining confidence in its generalization capabilities across different subjects, lighting conditions, and environmental contexts. These limitations are not specific to our implementation, but rather common to CNN-based architectures that operate on isolated frames, treating each image independently, with no temporal continuity, and whose internal attention may focus on irrelevant or suboptimal visual regions, rather than those truly indicative of drowsiness. These observations reinforce the idea that, without an explicit understanding of temporal patterns, such architectures are inadequate for robust and person-independent detection of drowsiness.

Our idea then shifted towards observing the EAR value evolution over the last 60 frames and classify the current value of this sequence as drowsy or not, so that the falling asleep of a person is captured.
**cLimaSenseTM** is our **Long Short-Term Memory Recurrent Neural Network**, designed to classify the current eye status based on the EAR value of the last 60 frames. Each cell composing the LSTM takes as input an hidden state and an input value (in this case The EAR value of the frame) and outputs an hidden state, which can be either fed to the next cell of the network or used for tasks such as classification. This architecture is designed to capture the behavior of sequences, therefore it is perfect for capturing the onset of drowsiness of a driver, but its parameters need to be fine-tuned to balance accuracy and inference time. **cLimaSenseTM** was trained on Vicomtech's DMD Drowsiness dataset[2] to output one of the five classes. When the output label is close, the system performs all the operations related to the drowsiness state, which in this case is showing on-screen an alert box.


Figure 3: Drowsiness detected

# 5 Prototype and Demo Setup and Description

The ClimaSense system prototype for detecting drowsiness can be broken down in three main components:

## 5.1 Raspberry Pi 3 Model B+

The Raspberry Pi 3 Model B+ is a third-generation single-board computer, featuring:

- **Processor:** 1.4GHz 64-bit quad-core Broadcom BCM2837B0, Cortex-A53 (ARMv8) SoC.
- **Memory:** 1GB LPDDR2 SDRAM.
- **Wireless Connectivity:** Dual-band 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2/BLE.
- **Ethernet:** Gigabit Ethernet over USB 2.0, with a maximum throughput of 300 Mbps.
- **GPIO:** Extended 40-pin GPIO header.
- **Camera Support:** CSI camera port for connecting a Raspberry Pi camera.
- **Power Input:** 5V/2.5A DC power input.

## 5.2 Camera Module

For real-time detection, the Raspberry Pi Camera Module 2 (v2.1) was used. The module includes:

- **Sensor:** Sony IMX219, 8-megapixel sensor.
- **Capabilities:**
  - High-definition video recording: 1080p30, 720p60, VGA90 video modes.
  - Still image capture.

- **Connection:** Attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi.
- **Software Support:** Numerous third-party libraries are available, including the Picamera Python library.

### 5.3 Operating System Setup

The operating system was installed on a 64GB KIOXIA EXCERIA microSDXC U1 I card with speeds up to 10MB/s. Balena Etcher software was used to install the following version:

- **OS:** DietPi OS with Desktop.
- **OS Version:** 9.11.2.
- **Release Date:** February 23th, 2025
- **System:** 64-bit.
- **Image Size:** 1024 MB.

DietPi is a highly optimised and minimal Debian-based Linux distribution. It is extremely lightweight at its core, and also extremely easy to install and use. These qualities made fundamental performance improvements with respect to using the default Raspbian OS.

## 6 Performance Evaluation and Optimization

This section is dedicated to a comprehensive evaluation of LSTM model performance. The primary objectives were to select the most suitable LSTM model architecture by comparing various configurations and to assess the final inference times of the chosen model on both PC and Raspberry Pi platforms.

### 6.1 LSTM Model Performances

The architecture of the LSTM models was explored by varying two key hyperparameters: the number of neurons (hidden units) per layer and the total number of layers. The results were computed on the **PC** because of its greater computational power.

- **Neurons (Hidden Units per Layer)**: This hyperparameter determines the dimensionality of the hidden state and cell state within each LSTM layer. A larger number of neurons generally increases the model's capacity to learn complex patterns and relationships from the input sequences. However, it also increases the number of parameters and computational cost. The tested values for this hyperparameter were:
  - 32
  - 64
  - 96
  - 128

  - 150
  - 182

- **Layers**: This hyperparameter specifies the number of LSTM layers stacked sequentially in the model. Deeper models (i.e., models with more layers) can potentially learn more abstract and hierarchical features from the data. However, increasing the number of layers also leads to a more complex model, which can increase training time, inference time, and the risk of overfitting if not managed properly. The tested values for this hyperparameter were:
  - 1
  - 2
  - 3

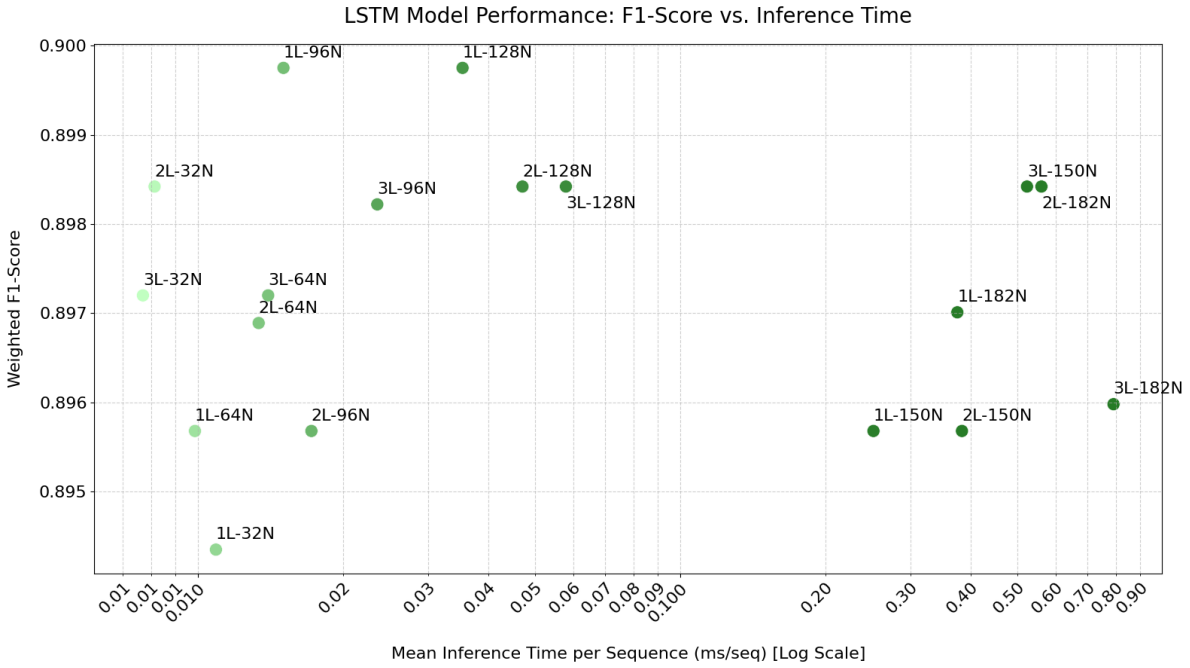


Figure 4: Comparison of LSTM model configurations, obtained on the PC, based on Weighted F1-Score versus Mean Inference Time per Sequence (Log Scale). Each point represents a specific combination of layers (L) and neurons (N), e.g., 1L-96N denotes 1 layer with 96 neurons.

As illustrated in Figure 4, various LSTM configurations were benchmarked. After careful consideration of the trade-off between predictive performance (Weighted F1-Score) and computational efficiency (Inference Time), the model configured with **One layer (1L)** and **96 Neurons (96N)** was identified as the optimal choice. This particular configuration demonstrated a strong balance, achieving a high F1-score while maintaining a relatively low inference time. For example, while a model with **2 layers and 32 neurons (2L-32N)** exhibited a faster inference time, its corresponding F1-Score was notably lower, making the 1L-96N model more suitable for the target application where both accuracy and speed are important.

| Model | Layers | Neurons | F1-Score per Class | | | | | Support per Classe | | | | | Inference Time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Close | Closing | Open | Opening | Undef. | Close | Closing | Open | Opening | Undef. | (s/seq) |
| LSTM 1L - 32N | 1 | 32 | 0.90 | 0.71 | 0.95 | 0.81 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.010 91 |
| LSTM 1L - 64N | 1 | 64 | 0.90 | 0.71 | 0.95 | 0.82 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.009 87 |
| LSTM 1L - 96N | 1 | 96 | 0.91 | 0.72 | 0.95 | 0.83 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.015 06 |
| LSTM 1L - 128N | 1 | 128 | 0.91 | 0.72 | 0.95 | 0.83 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.035 41 |
| LSTM 1L - 150N | 1 | 150 | 0.90 | 0.71 | 0.95 | 0.82 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.251 73 |
| LSTM 1L - 182N | 1 | 182 | 0.90 | 0.71 | 0.95 | 0.83 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.375 68 |
| LSTM 2L - 32N | 2 | 32 | 0.91 | 0.72 | 0.95 | 0.82 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.008 14 |
| LSTM 2L - 64N | 2 | 64 | 0.90 | 0.72 | 0.95 | 0.82 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.013 38 |
| LSTM 2L - 96N | 2 | 96 | 0.90 | 0.71 | 0.95 | 0.82 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.017 21 |
| LSTM 2L - 128N | 2 | 128 | 0.91 | 0.72 | 0.95 | 0.82 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.047 13 |
| LSTM 2L - 150N | 2 | 150 | 0.90 | 0.71 | 0.95 | 0.82 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.384 07 |
| LSTM 2L - 182N | 2 | 182 | 0.91 | 0.72 | 0.95 | 0.82 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.561 24 |
| LSTM 3L - 32N | 3 | 32 | 0.91 | 0.71 | 0.95 | 0.82 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.007 70 |
| LSTM 3L - 64N | 3 | 64 | 0.91 | 0.71 | 0.95 | 0.82 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.014 00 |
| LSTM 3L - 96N | 3 | 96 | 0.90 | 0.72 | 0.95 | 0.83 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.023 57 |
| LSTM 3L - 128N | 3 | 128 | 0.91 | 0.72 | 0.95 | 0.82 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.057 99 |
| LSTM 3L - 150N | 3 | 150 | 0.91 | 0.72 | 0.95 | 0.82 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.523 91 |
| LSTM 3L - 182N | 3 | 182 | 0.91 | 0.70 | 0.95 | 0.82 | 0.00 | 1695 | 1358 | 6604 | 1482 | 2 | 0.791 79 |

Table 1: LSTM model results, obtained varying layers and neurons.

### 6.2 Comparative Inference Performance: PC vs. Raspberry Pi

Following the selection of the optimal LSTM model (1L-96N), its real-world inference performance, along with other critical stages of the processing pipeline, was benchmarked on two distinct hardware platforms. The objective was to quantify the performance differences between a high-specification PC and a resource-constrained Raspberry Pi.

The test platforms were:

- **PC**: Equipped with an Intel Core i7-1280P processor, an NVIDIA GeForce RTX 3050 GPU (leveraging CUDA 12.8), and 16 GB of RAM.
- **Raspberry Pi**: A Model 3B+.

The mean processing times (in milliseconds, ms) and standard deviations (std.) for key pipeline components, along with the estimated average Frames Per Second (FPS), are presented in Table 6.2.

| Processing Stage | PC (i7-1280P, RTX 3050) | Raspberry Pi 3B+ |
|---|---|---|
| Frame Capturing (ms) | 9.55 ± 13.93 | 16.90 ± 14.79 |
| FaceMesh + EAR Computation (ms) | 2.93 ± 0.37 | 55.78 ± 35.12 |
| LSTM Inference (ms) | 1.74 ± 2.32 | 18.40 ± 4.11 |
| Mean Frame Processing Time (ms) | 23.38 ± 14.91 | 86.51 ± 37.37 |
| Estimated Average FPS | 39.96 | 11.40 |

Table 2: Performance comparison of key processing stages between PC and Raspberry Pi 3B+. Values are mean ± standard deviation in milliseconds (ms), unless otherwise noted.

The results clearly highlight the substantial performance advantage of the PC across all computationally intensive tasks. The GPU acceleration on the PC significantly reduced the time required for FaceMesh + EAR computation and LSTM inference compared to the Raspberry Pi. While frame capturing times were somewhat comparable, the cumulative effect of faster processing for subsequent stages resulted in the PC achieving an estimated average FPS approximately 3.5 times higher than the Raspberry Pi 3B+ (39.96 FPS vs. 11.40 FPS). The "Mean Frame Processing Time" also reflects this disparity, indicating lower latency and less jitter (as suggested by the standard deviation relative to the mean, though both platforms show considerable variability) on the PC. This comparison underscores the hardware-dependent nature of real-time application performance, however, as stated in [3] 11.40 FPS are enough to detect drowsiness in drivers, especially if the accuracy of the system is high like in this case.

## 7 Structure of the Demo

The demonstration component of the ClimaSense system, encapsulated in the Python script `realtime_nn_drowsiness_detection.py`, serves as the practical, real-time deployment of the drowsiness detection model developed and trained previously. This script is engineered to operate continuously, processing live video input from a standard webcam to assess the driver's alertness level.

The operational flow of the demo can be segmented into several key stages:

1. **Initialization and Configuration:** Upon execution, the script begins by establishing essential configurations. Critical pre-trained components, artifacts of the model training phase, are loaded. These include:

   - The **StandardScaler** (`ear_scaler.pkl`): Essential for normalizing the raw EAR values to the same scale used during model training, ensuring consistent input distribution for the neural network.
   - The **LabelEncoder** (`label_encoder.pkl`): Used to convert the model's numerical output (class indices) back into human-readable state labels (e.g., 'open', 'close').
   - The **Trained LSTM Model** (`lstm_1_layer_relu_96.pth`): The core intelligence of the system. The specific model, `EyeStateLSTM_96` (a single-layer LSTM with 96 hidden units), is instantiated, and its learned weights are loaded. The script also intelligently selects the appropriate computation device (CPU or GPU).

   Interface components are also initialized: `cv2.VideoCapture` for webcam access and Google/MediaPipe's `FaceMesh` solution for robust facial landmark detection. The choice of Google/MediaPipe provides an efficient, pre-built mechanism for locating key facial points, including those around the eyes, which are fundamental for EAR calculation.

2. **Real-Time Processing Loop:** The system then enters its main operational loop, processing video frames sequentially:

   - **Frame Acquisition and Preprocessing:** Each frame is captured from the webcam, flipped horizontally (for a more natural mirror-like view), and converted from BGR to RGB format, as required by FaaceMesh.
   - **Facial Landmark Detection:** The RGB frame is processed by the initialized `FaceMesh` model. If a face is detected, the 3D coordinates of its landmarks are returned.
   - **Eye Aspect Ratio (EAR) Calculation:** From the detected facial landmarks, specific points corresponding to the left and right eyes (defined by `LEFT_EYE_IDXS` and `RIGHT_EYE_IDXS`) are extracted. The `eye_aspect_ratio` function then computes the EAR for each eye, and an average EAR is derived. This ratio serves as a quantitative measure of eye openness.
   - **Temporal Sequence Aggregation:** The calculated `current_ear` value is appended to a `deque` (double-ended queue) named `ear_sequence`. This data structure efficiently maintains a sliding window of the most recent `SEQ_LENGTH=60` EAR values.
   - **LSTM Model Inference:** Once the `ear_sequence` is populated with enough values, it is prepared for the LSTM model. The sequence is converted to a NumPy array, reshaped, and scaled using the pre-loaded `StandardScaler`. This scaled sequence is then transformed into a PyTorch tensor and fed into the LSTM model. The model outputs logits for each class, and the class with the highest logit is selected as the predicted state index. This index is then decoded into a string label (e.g., "close") using the `LabelEncoder`. If insufficient EAR values have been collected, a status indicating data collection

is displayed. If no face is detected, the EAR sequence is cleared, and "No face" is indicated.

- **Visualization and Alerting:** The processed frame is annotated with several pieces of information for the user:
  - The current EAR value.
  - The predicted drowsiness state.
  - An adaptive text coloring scheme (`get_adaptive_text_color`) is employed to ensure text visibility against varying background brightness levels within the video frame, enhancing the user experience.
  - Crucially, if the predicted state corresponds to the label *close*, a prominent visual alert – "!!! DROWSINESS ALERT !!!" – is overlaid on the screen with a distinct background. In the real case, this action would be replaced by the flowing of cold air in the neck zone of the driver, along with an acoustic alert, to gently increase the alertness of the driver.

3. **System Termination:** The loop continues until the user presses the 'q' key, at which point the script performs necessary cleanup, releasing the webcam and closing all OpenCV windows.

The demo effectively showcases the integration of computer vision techniques for feature extraction (FaceMesh and EAR) with a temporal deep learning model (LSTM) for state classification. Its structure prioritizes real-time performance and clear user feedback, particularly the immediate and unambiguous drowsiness alert, which is paramount for the application's safety objective. The modular loading of pre-trained components underscores a robust development pipeline where model training and deployment are distinct, well-managed phases.

# 8 Conclusions

The ClimaSense project has successfully demonstrated the significant advantages of employing Long Short-Term Memory (LSTM) networks for real-time driver drowsiness detection. Our work validates that by analyzing the temporal evolution of Eye Aspect Ratio (EAR) data, rather than isolated or simply aggregated ocular measurements, a more nuanced and reliable assessment of driver alertness can be achieved.

A key outcome of this research is the clear demonstration that understanding the sequence of eye behavior is paramount. The selected 1L-96N LSTM architecture, chosen through a careful design process balancing weighted F1-score, inference speed, and resource utilization (as detailed in Section 6.1 and visualized in Figure 3), excels at capturing these critical temporal dependencies. This ability to model patterns such as blink rate variations, prolonged eye closures characteristic of micro-sleeps, and the subtle transitions into fatigue differentiates our approach. Systems that do not inherently process the temporal dimension of such physiological signals often struggle to distinguish between benign, momentary eye closures and genuine indicators of drowsiness, potentially leading to either reduced sensitivity or increased false alarms. ClimaSense, by learning these complex temporal relationships from the extensive DMD dataset, offers a more robust and contextually aware detection mechanism.

The performance evaluations confirm ClimaSense's suitability for practical application. On capable PC hardware, the system achieves an average processing rate of approximately 39.96 FPS, underscoring its real-time responsiveness. The detailed breakdown of processing times further reveals that while facial landmark detection (FaceMesh+EAR computation) can be a bottleneck on less powerful hardware like the Raspberry Pi 3B+ (contributing 55.78ms per frame), the LSTM inference itself is relatively efficient ( 1.74ms on PC, 18.40ms on RPi). This insight is crucial for guiding future optimizations, particularly for embedded deployments where efficient feature extraction is as critical as efficient model inference. The 11.40 FPS achieved on the Raspberry Pi 3B+, along with its CPU and RAM usage metrics, provides a valuable real-world baseline for such considerations.

In essence, ClimaSense advances the field by shifting the paradigm from static observation to dynamic temporal understanding of driver fatigue indicators. This inherent ability to process and learn from sequential data provides a superior foundation for early and accurate drowsiness detection compared to non-sequential methods (e.g. PERCLOS). The project not only delivers a functional prototype but also offers a robust framework and critical performance insights for the continued development of intelligent systems aimed at significantly enhancing driver safety.

# References


[1] Bin Huang, Renwen Chen, Qinbang Zhou, and Wang Xu. Eye landmarks detection via weakly supervised learning.

[2] Juan Diego Ortega, Neslihan Kose, Paola Canas, Min-An Chao, Alexander Unnervik, Marcos Nieto, Oihana Otaegui, , and Luis Salgado. Dmd: A large-scale multi-modal driver monitoring dataset for attention and alertness analysis. *Proc. Eur. Conf. Comput. Vis.Cham, Switzerland: Springer*, 2020.

[3] Florez R., Palomino-Quispe F., Alvarez A.B., R.J. Coaquira-Castillo, and Herrera-Levano J.C. A real-time embedded system for driver drowsiness detection based on visual analysis of the eyes and mouth using convolutional neural network and mouth aspect ratio. *Sensors 2024, 24, 6261*, 2024.

[4] Shehzad Saleem. Risk assessment of road traffic accidents related to sleepiness during driving: a systematic review.