Politecnico di Torino

Master's Degree in Automotive Engineering

A Real-Time Driver Behavior Monitoring and Feedback Framework to deliver safe, comfortable, and energy-efficient driving

Candidate: WEI LI

Supervisors: Prof. Angelo Bonfitto
Prof. Shailesh Hegde

Academic Year 2024/25

Abstract

Understanding and responding to driver behavior is significant for improving road traffic safety, optimizing energy efficiency, and enhancing ride comfort, especially in electric vehicles. This thesis proposes a Human-Machine Interface (HMI) system framework that recognizes multiple types of abnormal driving behaviors in real time and provides personalized and immediate feedback to drivers through an intuitive interface.

First, typical roads and traffic scenes are selected using the SCANeR™ Studio virtual driving simulation platform, allowing realistic driving simulation and data acquisition. Dynamic parameters such as vehicle speed, longitudinal and lateral acceleration, steering angle, pedal position, and motor efficiency are collected. The K-means algorithm is applied for preliminary data segmentation, and then the Iterative Density-Based Spatial Clustering of Applications with Noise (I-DBSCAN) algorithm is used for unsupervised clustering to identify potential aggressive driving behaviors. These clustering results are further subdivided based on the thresholds of comfort and energy efficiency indicators (e.g., jerk, motor efficiency), resulting in a multi-category label set. A Long Short-Term Memory Network (LSTM) is used to train time series driving data using this label. Bayesian optimization is used to tune hyperparameters such as learning rate and the number of hidden layers. The final model classifies four behavior types in real time: *aggressive*, *uncomfortable*, *low energy efficiency*, and *normal*, achieving F1 scores of 0.95, 0.99, and 0.99, respectively.

Based on this classification model, an HMI is designed and implemented. In addition to presenting real-time dynamic vehicle data, it communicates driving behavior classification results to the driver through visual and auditory alerts. A unified "sensitivity threshold" is introduced, allowing drivers to adjust this threshold through the interface. During operation, the LSTM model continuously outputs probabilities for each behavior type. The system compares each probability with the user-defined threshold in order of priority (*aggressive* > *low energy efficiency* > *uncomfortable* > *normal*). Once any behavior's probability exceeds the threshold, it is regarded as the current driving state, and the corresponding prompt is triggered. This single-threshold, priority-based scheme enhances interaction efficiency and reduces distraction caused by overlapping alerts.

The framework is deployed on a real-time, hardware-in-the-loop platform composed of a driving simulator, the Speedgoat, and the Raspberry Pi. The User

Datagram Protocol (UDP) is used to enable low-latency communication between modules, ensuring real-time, closed-loop control of the data flow from acquisition, identification, and feedback.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ADAS Advanced Driver Assistance System.

C2X Car-to-Everything.

CAN Controller Area Network.

CNN Convolution Neural Network.

DAS Driver Assistance Systems.

DBSCAN Density-Based Spatial Clustering of Applications with Noise.

DHCP Dynamic Host Configuration Protocol.

DiL Driver-in-Loop.

DTW Dynamic Time Warping.

EDBs Elementary Driving Behaviors.

FCN Fully Convolutional Network.

GUI Graphical User Interface.

HiL Hardware-in-Loop.

HMI Human-Machine Interface.

I-DBSCAN Iterative Density-Based Spatial Clustering of Applications with Noise.

IP Internet Protocol.

LAN Local Area Network.

LHT Left Hand Traffic.

LSTM Long Short-Term Memory Network.

MP Monitoring Period.

NHTSA National Highway Traffic Safety Administration.

PCA Principal Component Analysis.

PLSR Partial Least Squares Regression.

RF Random Forest.

RHT Right Hand Traffic.

S3VM Semi-supervised Support Vector Machine.

SHM Shared Memory.

SiL Software-in-Loop.

SVM Support Vector Machine.

TCP Transmission Control Protocol.

UDP User Datagram Protocol.

Chapter 1

Introduction

1.1 Background

Although there has been some progress in global road traffic safety in recent years, the number of road traffic deaths remains unacceptably high, and the overall improvement trend has shown signs of stagnation. According to the 2023 Global Status Report on Road Safety, road traffic accidents caused approximately 1.19 million deaths worldwide in 2021. Furthermore, the report revealed that approximately 181,453 children aged 0–19 die in road traffic accidents annually. This figure highlights the serious threat that road traffic poses to the lives and health of young people.[1]

One of the reasons for these injuries is the improper driving behavior of drivers. Aggressive driving behavior is an important factor that easily leads to dangerous driving. According to statistics from the National Highway Traffic Safety Administration (NHTSA) in 2020, 66% of traffic fatalities in the United States were related to aggressive driving. More than 78% of American drivers said they had engaged in aggressive driving at least once in the past year [2]. NHTSA defines aggressive driving as ”operating a motor vehicle in a manner that could endanger other people or property”, typically involving multiple traffic violations during a single drive or segment [3]. It is important to note that aggressive driving and road rage are not the same thing. The former is a traffic violation, while the latter refers to behavior intended to cause harm and may constitute a criminal offense. Aggressive driving includes failure to yield to other drivers and pedestrians, failure to use turn signals; making extra, unnecessary lane changes; tailgating other drivers; making sudden stops, accelerations, or turns; and ignoring traffic controls [4]. Therefore, it is of great significance to improve road traffic safety by identifying and intervening in aggressive driving behavior.

In addition to driving safety, the impact of driving behavior on ride comfort and vehicle energy consumption is also receiving increasing attention. Some seemingly non-dangerous and non-illegal actions, such as frequent acceleration and deceleration, can make passengers feel uncomfortable, especially those who are elderly, children, or have sensitive bodies. While this type of driving is not radical, it still negatively affects the riding experience and can even cause physiological reactions such as motion sickness. However, as electric vehicles become more popular, these unstable driving styles will consume more electricity. [5] found that the frequency and intensity of acceleration, deceleration, and stopping directly affect the energy consumption of electric vehicles through real tests and modeling. The R^2 model achieved a goodness of fit of over 0.9, indicating that dynamic behavior significantly affects energy consumption. In addition, [6] also shows that frequent starting and stopping under traffic flow conditions will significantly increase energy consumption, especially in urban congestion environments.

Improper driving behavior poses significant challenges in terms of safety, comfort, and energy, so HMI has become an essential component of modern driver assistance systems. HMIs are no longer just passive information displays; they are increasingly designed to provide real-time feedback to help drivers recognize and correct undesirable or unsafe behaviors before they become more serious risks. For example, Innovative HMI designs (such as LED ambient lighting) can effectively enhance the driver's visual behavior and situational awareness, thereby improving driving safety [7]. In addition, HMI is also a key link in building driver trust in the autonomous driving system. Its information transparency and interactive design will significantly affect the user's understanding and acceptance of vehicle behavior [8]. Some studies also apply HMI to fuel efficiency guidance, and through the combination with vehicle performance data, realize the learning and optimization of driving behavior [9]. These studies suggest that further research is needed into the design and effectiveness of HMIs as behavioral intervention tools, particularly in addressing driving patterns related to safety, comfort, or efficiency.

1.2 Related work

1.2.1 Aggressive Driving Behavior Classification Techniques

Monitoring and effectively identifying aggressive driving behavior is the cornerstone of improving driving safety.

Early methods for judging drivers' driving behavior are generally based on rules, thresholds, or feature engineering. [10] derived an accident risk index based on a large data set of accelerometer readings that could only be collected by the drivers. [11] proposed a threshold-based decision tree system based on CAN-Bus signals (such as speed and steering wheel angle), which can automatically identify driving actions (such as steering and acceleration) and achieve a classification accuracy of 54–73% on real vehicle data. The vehicle acceleration change rate (jerk) has been shown to be an effective feature for identifying aggressive behavior. Through natural driving data, it was found that the frequency of large negative jerk (acceleration mutation at the moment of emergency braking) is highly correlated with aggressive behavior, with a detection rate of 77% and an AUC of up to 0.77 [12]. [13] proposed a method to describe the relationship between lateral and longitudinal acceleration and velocity, based on which the driver's behavior can be classified as safe or unsafe. These methods are based on basic driving signals and complete classification by setting thresholds or decision rules. They have the advantages of simple implementation and low computational complexity.

In recent years, supervised learning algorithms have been widely used in driving behavior classification tasks to predict the driver's operating behavior by training models. Such methods usually rely on labeled data for training. [14] proposed a novel system that uses a Dynamic Time Warping (DTW) algorithm and smartphone-based sensor fusion (accelerometer, gyroscope, magnetometer, GPS, video) to actively detect, identify, and record aggressive behavior without external processing, thereby increasing awareness of potential aggressive behavior and further ensuring driver safety. [15] used a method that fused visual and sensor features, modeled driving sessions using road images and vehicle dynamics (such as vehicle speed, speed), extracted feature vectors through Gaussian distribution, and classified them using a Support Vector Machine (SVM) classifier to determine aggressive driving behavior, with a detection rate of 93.1% on real traffic data. Based on the basic safety information data of connected vehicles, [16] proposes a real-world scenario lane crossing time (TLC) analysis method based on the Random Forest (RF) model, which successfully identifies aggressive driving behaviors on horizontal curves with an accuracy of up to 95.34%. In addition, more advanced deep learning structures have also been introduced into aggressive driving behavior recognition. In [17], the authors determined whether the driving style is safe or aggressive by detecting traffic direction signs, speed, and maneuver estimation. They used the Convolution Neural Network (CNN) algorithm for detection with an accuracy of 88.02%. The paper [18] proposed a system for classifying normal and aggressive driving behaviors based on a combination of a Fully Convolutional Network (FCN) and a LSTM algorithm. The system formulates the problem of whether aggressive driving behavior is involved as a time series classification. Using the UAH-DriveSet dataset, the system achieved an F-1 score of 95.88% at a window length of 5 minutes. In [19], a hyperparameter optimization method based on the LSTM model was proposed, and a Bayesian optimization model was established to optimize the hyperparameters. The optimal hyperparameters (including window size, learning rate, number of hidden layers, and number of hidden units) were determined to accurately predict driver behavior. The model accuracy reached 97.02%. This study [20] collected real data from vehicle accelerometers and gyroscopes, used statistical regression, time series analysis, and the following machine learning algorithms to identify aggressive driving behavior: GMM, Partial Least Squares Regression (PLSR), wavelet transform, and SVR. The results showed that when using a multi-source dataset, PLSR performed best with an F1 score of 0.77.

Manually labeling a large amount of driving data is very time-consuming due to the uncertainty of driver behavior and the differences between data analysts. To address this problem, the researchers proposed a semi-/unsupervised method for scenarios with few labels and high labeling costs. [21] used a Semi-supervised Support Vector Machine (S3VM) to distinguish between aggressive and normal driving styles based on a small amount of labeled data. Experiments show that S3VM only requires a small amount of labeled data, combined with a large amount of unlabeled data, to significantly improve the classification accuracy by about 10%.

1.2.2 Human Machine Interface

With the development of driving behavior recognition technology, its practical application in driving safety and driving behavior intervention has gradually become more important. Among them, the HMI, as a bridge connecting the driver and the vehicle, is widely used to deliver behavior recognition results, provide timely feedback, and guide driving behavior improvement. A well-designed HMI can not only improve driving safety and comfort but also enhance the driver's trust and acceptance of the assistance system. Therefore, research around HMI mainly focuses on the presentation of its feedback content (such as vision, hearing, touch, etc.), the information triggering mechanism, and the impact of the system on driving behavior, forming a number of representative research directions.

[22] combined real car and simulator experiments to explore the integration of future HMI and Driver Assistance Systems (DAS). They introduced sidestick control and virtual instruments in the prototype car to reduce the physical burden of

driving, and collected driver status data through the EEG interface of the driving simulator to improve the system's ability to understand the driving context, providing a feasible path and verification platform for the integrated design of HMI and DAS. [23] designed the concept car Carai to quickly develop and test the integration of Advanced Driver Assistance System (ADAS) and HMI. By integrating multiple environmental perception sensors and a modular computer framework, an LED display interface covering the entire front field of view was designed to provide real-time prompts for key driving information such as lane changes and pedestrians in blind spots, achieving a closed-loop design from perception to interaction, allowing HMI to not only display information, but also become an interactive outlet for ADAS. This shows the key role of HMI in improving the understandability and operability of ADAS. [24] evaluated an in-vehicle HMI system for guiding energy-saving and safe driving through a driving simulation experiment, and compared the effects of three information prompts on driving behavior: suggestions, feedback, and a combination of the two. The results showed that the interface that combined suggestions and feedback was the most effective in reducing fuel consumption, speeding, and improving driving stability, and was also accepted by most drivers. Studies have shown that if HMI can work closely with driving assistance functions, it can not only provide information but also effectively guide changes in driving behavior, and is an important means to achieve safety and energy-saving goals. [25] constructed a mathematical model to analyze how HMI design affects the driver's perception and reaction time to vehicle information in autonomous driving and connected environments. The study used HMI as a key link connecting drivers, sensors, and communication systems to quantify its impact on driving usability and safety. The results show that the shorter the reaction time and the more efficient the information transmission, the higher the system usability, emphasizing the importance of HMI design in reducing latency and improving the reliability of driving behavior. [26] proposed a HMI framework for autonomous driving, systematically sorting out various HMI types inside and outside the vehicle (such as autonomous driving status HMI, vehicle information HMI, entertainment HMI, dynamic HMI, and external HMI), and explored the relationship and influencing factors between these interfaces. The framework emphasizes that HMI design should be coordinated according to dynamic factors such as automation level, passenger status, and traffic environment to form a consistent and synchronized multi-channel interaction strategy to support efficient communication between people and systems, and between people and the traffic environment in autonomous driving.

[27] proposed a method that combines driving behavior recognition with an HMI feedback mechanism to improve the interactive safety of semi-automatic driving

systems. The system monitors the driver's status (such as distraction or abnormal behavior) and adjusts the presentation of warning and prompt information in the HMI in real time to guide safer driving reactions. The study emphasized that the integration of HMI and a behavior recognition system can effectively support human-vehicle collaboration and reduce accident risks. [28] proposed an HMI design that supports driver-supervised learning to help ADAS systems identify and learn individualized driving preferences. The HMI enables drivers to monitor system behavior in real time, intervene in decision conflicts, and provide feedback on the learning process, thereby improving the adaptability and transparency of the system. The experiment verified the required parameters and interaction requirements of the HMI through semantic analysis and behavioral observation, laying the foundation for the realization of personalized assistance systems.

Different types of HMI show significant differences in delivering driving-related information, guiding driving behavior, and influencing driving decisions. Their design methods, whether the content, form, or interaction of information presentation, will directly affect the driver's understanding efficiency, response behavior, and trust and acceptance of the system. Therefore, existing studies have widely explored the application effects of various HMI designs in different driving scenarios to evaluate their potential value in optimizing driving behavior.

[29] compared three HMI designs through simulation experiments to explore their effectiveness in supporting cooperative driving. The results showed that the interface that highlights cooperative vehicles and gaps can best enhance drivers' understanding and trust, and suggestive information is particularly useful, providing empirical reference for cooperative driving HMI design. Similarly, [30] designed and compared three HMI information display methods (distance information, suggestion information, and guidance information) to support safe overtaking behavior in a connected vehicle environment. The results showed that suggestion information was the most effective, which could improve driving performance and reduce visual burden; while guidance information increased the success rate, but had lower compliance. [31] compared three HMI: Baseline, which only displays basic information, Sensor HMI based on onboard sensors, and Car-to-Everything (C2X) HMI that integrates C2X communication to support cooperative driving when merging and turning left. The results show that C2X HMI performs best in improving user trust, acceptance, and interactive experience without increasing driving burden, indicating that clear and perceptible cooperative information is the key to efficient HMI design. [32] designed and integrated an HMI system based on shared control and role switching, combining a tactile steering wheel controller with a visual interface to achieve dynamic collaboration between the driver and the autonomous driving

system. The HMI clearly distinguishes between manual, shared, and autonomous driving modes through colors, icons, and animations, dynamically prompts the current control rights and reasons for intervention, improves the driver's understanding and trust in the system status, and supports a smoother takeover and interactive experience. [33] designed and tested three augmented reality-based HMI interfaces to guide drivers to achieve more efficient driving behavior in hybrid vehicles. Different interfaces provide optimal vehicle speed and operation suggestions in the form of numerical values, images, or text to help drivers adjust their behavior to reduce fuel consumption. The experimental results show that these HMIs can significantly improve fuel efficiency, among which the "text-based interface" performs best in terms of cognitive burden and acceptance. [34] designed and field-tested an in-vehicle omnidirectional collision warning HMI, comparing the effects of visual cues and combined audio-visual warnings on driving behavior under different collision types (front, rear, and side). The results showed that combined audio-visual warnings can shorten reaction time, reduce vehicle deviation, and improve safety, and that users with different driving experience have significant differences in their responses to warning forms.

1.3 Purpose & Contribution

This study aims to design and implement a HMI system based on driving behavior recognition, which can identify the driver's abnormal driving behavior in real time during vehicle operation and provide timely feedback through a friendly interface, thereby improving road safety, riding comfort, and energy efficiency of electric vehicles.

The main contributions of this thesis include:

1. **A High-Accuracy, Real-Time Driver Behavior Recognition Model**:
   Different dynamic driving data were collected through the SCANeR™ Studio virtual simulation platform using typical roads and traffic scenarios. K-means and improved DBSCAN clustering methods were then used to identify aggressive behavior. Then, a multi-category label set was obtained, and an LSTM network was used to train for time series modeling. Bayesian optimization was then used to adjust the hyperparameters, enabling the model to accurately identify four types of behaviors: *aggressive*, *uncomfortable*, *low energy efficiency,normal.* The highest F1 score was 0.98, which met the requirements for high real-time accuracy.

2. **An Efficient, Prioritized Feedback Mechanism**:

   The system is designed with a unified sensitivity threshold that allows the driver to adjust the model's response. The LSTM outputs four types of behavior probabilities. The system compares the threshold to check if it has been exceeded in order of priority (*aggressive* > *low energy efficiency* > *uncomfortable* > *normal*) and triggers the corresponding feedback prompt. This mechanism improves interaction efficiency, avoids redundant alarms and information interference, and implements a behavior intervention method that aligns more closely with human-machine cognitive characteristics.

3. **An Intuitive and Interactive HMI**:

   It provides a quick interface that combines visual and auditory. It displays real-time behavior classification results and vehicle dynamic data. It allows users to adjust behavior judgment thresholds in real time. It establishes a closed-loop control path between the driver and the system. It enhances the initiative and personalization of driving behavior intervention.

4. **A real-Time Hardware-in-the-Loop Implementation**:

   The system is based on a hardware-in-the-loop architecture using the Speedgoat real-time control platform and the Raspberry Pi terminal. It uses the UDP protocol to enable high-speed communication between modules. In the simulated driving scenario, key indicators such as reaction time, acceleration fluctuation, and changes in energy consumption are evaluated to determine the effect of different prompt modes on behavior guidance.

1.4 Outline

This thesis is composed of six chapters, each of which gradually builds upon the implementation and evaluation of a real-time driver behavior monitoring and feedback framework. The rest of the thesis is organized as follows:

Chapter 2 presents the overall structure of the proposed framework, describes the physical setup of the driving simulator and its integration with the simulation software, and explains in detail the interactions between the different hardware components.

Chapter 3 describes the development process of the behavior recognition model, covering the construction of typical driving scenarios, data collection and preprocessing, and labeling behavior patterns using clustering techniques. This chapter also discusses the implementation and training of a LSTM model for real-time behavior classification and evaluates its performance.

Chapter 4 focuses on the design and implementation of an HMI system that provides real-time feedback to the driver. This chapter covers interface layout, feedback logic, and visualization of different driving modes. A key feature, user-defined sensitivity settings, is explained, which allows users to adjust how the system interprets and responds to driving behavior. This chapter also introduces different types of feedback mechanisms, including visual and auditory warnings.

Chapter 5 focuses on the overall integration and verification of the system. First, the integrated structure and communication process of each module after the deployment of software and hardware are introduced, focusing on the overall operating status of the system in terms of real time performance, stability and feedback mechanism. On this basis, through multiple rounds of free driving tests, the end-to-end response speed of the system, the stability of model output, the reliability of UDP communication, and the integrity and consistency of HMI feedback are evaluated. Finally, a control experiment with/without HMI feedback is set up to further verify whether the HMI mechanism has a positive guidance effect under different driving behaviors.

Chapter 6 presents some conclusions of this thesis. Potential improvements and directions for the work are summarized, especially in terms of applying the system to real vehicles.

Chapter 2

System Architecture

2.1 Software Introduction

2.1.1 SCANeR™ Studio

SCANeR™ Studio, developed by AVSimulation, is an integrated virtual simulation and real vehicle fusion platform, designed to provide highly customizable modular solutions for all stages of automotive R&D. It integrates high-presicion vehicle dynamics models (such as multi-body dynamics, tyre modeling and suspension modeling), a comprehensive multi-sensor simulation library (covering cameras, radars, lidars and ultrasonic waves) and a powerful scenario editor. This allows users to build complex traffic environments, like multi-line roads, intersections, pedestrians, vehicle flows, traffic lights, dynamic weather, and day/night conditions. The primary function of the software is to offer users five distinct modes [35]:

1. **Vehicle mode**: Used to develop mathematical models of vehicles such as cars and trucks. These models are based on various vehicle components, including suspension systems, braking mechanisms, lighting systems, tires, and wheels.
2. **Terrain mode**: Used to build realistic road networks with logical features such as traffic signs, signal lights, and speed limit indicators. This mode also contains a 3D visualization environment to enhance the realism of the simulation.
3. **Scenario mode**: This mode creates training scenarios by integrating vehicles and terrain to improve driver skills, evaluate infrastructure, and test cockpit interfaces. It can also customize scenarios, monitor nearby autonomous vehicles, enforce specific rules, and collect accurate simulation data.

4. **Simulation mode**: This mode is responsible for launching and managing simulation sessions, coordinating all related hardware and software systems (such as audio, video, and motion components) to ensure a consistent simulation experience.

5. **Analysis mode**: Designed for reviewing and evaluating training outcomes. It presents results in various formats, including charts, 3D animations, and data tables, to facilitate effective performance analysis.

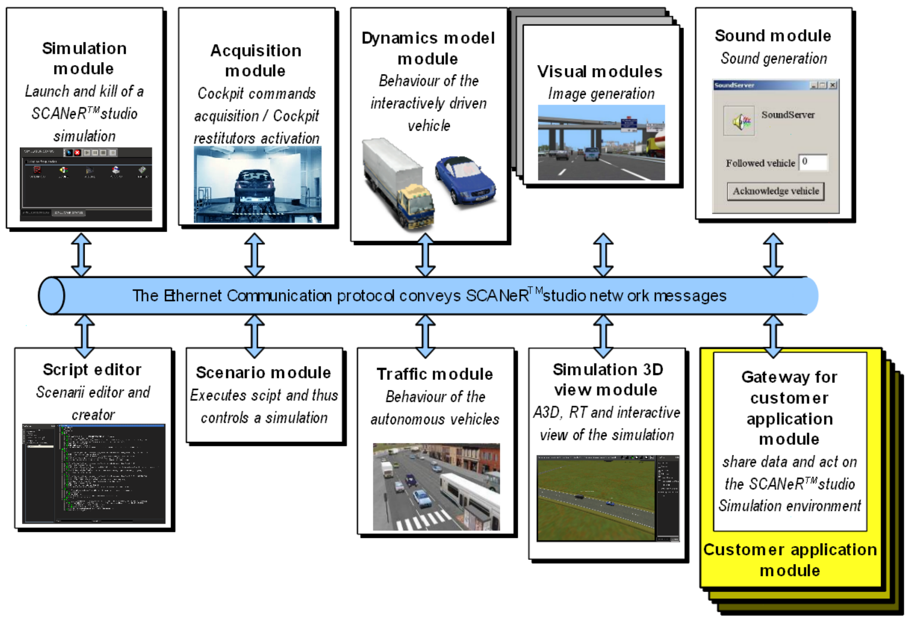

Simulation module
Launch and kill of a SCANeR™studio simulation
Acquisition module
Cockpit commands acquisition / Cockpit restitutors activation
Dynamics model module
Behaviour of the interactively driven vehicle
Visual modules
Image generation
Sound module
Sound generation
SoundServer
Followed vehicle
Acknowledge vehicle
The Ethernet Communication protocol conveys SCANeR™studio network messages
Script editor
Scenarii editor and creator
Scenario module
Executes scipt and thus controls a simulation
Traffic module
Behaviour of the autonomous vehicles
Simulation 3D view module
A3D, RT and interactive view of the simulation
Gateway for customer application module
share data and act on the SCANeR™studio Simulation environment
Customer application module

Figure 2.1: Ethernet modules communication.

SCANeR™ Studio has a distributed architecture, where the communication between the different modules is realized through Ethernet(shown as Figure 2.1). This architecture allows the modules to be implemented on multiple machines and to be scaled.

For communication between some specific modules, such as ACQUISITION (DriverHandle), MODELHANDER (dynamic model), MOTION, and the SCANeR API, uses SHM for fast and efficient communication with low latency (shown as Figure 2.2).

In terms of system interaction, SCANeR™ Studio provides a variety of open interfaces and supports the seamless integration of simulation and control platforms, such as MATLAB/Simulink and Python. It also enables real-time communication with vehicle controllers, sensor simulators, and actuators via standard protocols, including Controller Area Network (CAN), EtherCAT, UDP, and Transmission Control Protocol (TCP)/Internet Protocol (IP). This highly flexible interface de-

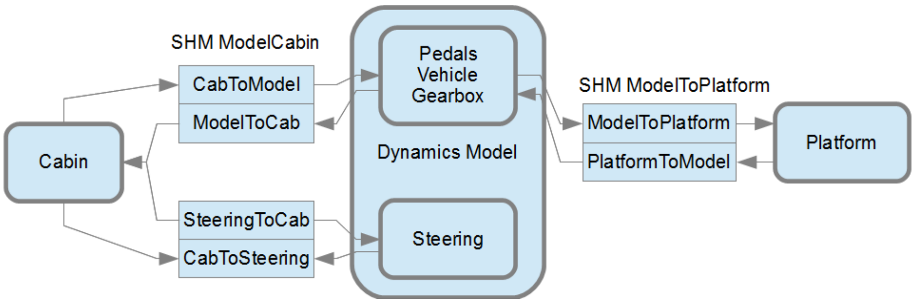
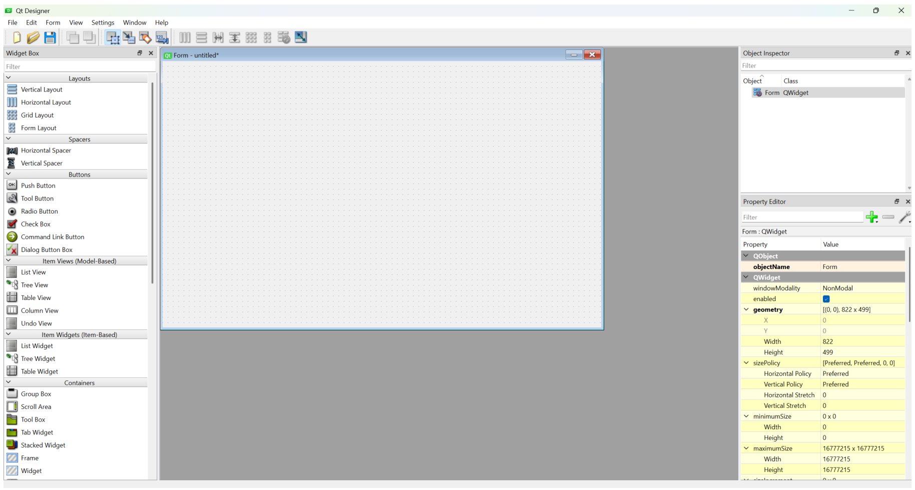


Figure 2.2: SHM Diagram.

sign makes it widely used in Software-in-Loop (SiL), Hardware-in-Loop (HiL), and Driver-in-Loop (DiL) testing processes.

SCANeR™ Studio can not only drive the rapid prototyping and verification of ADAS and autonomous driving algorithms, but it can also support research into driving behavior modeling, driver reaction analysis, HMI design, traffic flow modeling, and policy formulation. The platform also offers automated test management, data recording and playback, and the ability to compare virtual and real vehicle test results. These features greatly improve the repeatability of experiments, development efficiency, and rigor of system verification.

2.1.2 PyQt



Object	Class
Form	QWidget

Form : QWidget

Property	Value
QObject	
objectName	Form
QWidget	
windowModality	NonModal
enabled	✓
geometry	[(0, 0), 822 x 499]
X	0
Y	0
Width	822
Height	499
sizePolicy	[Preferred, Preferred, 0, 0]
Horizontal Policy	Preferred
Vertical Policy	Preferred
Horizontal Stretch	0
Vertical Stretch	0
minimumSize	0 x 0
Width	0
Height	0
maximumSize	16777215 x 16777215
Width	16777215
Height	16777215

Figure 2.3: User interface and main windows of Qt Designer.

PyQt5 is a development toolkit that encapsulates the functions of the Qt5 framework in Python, developed by Riverbank Computing. Qt5 is a cross-platform Graphical User Interface (GUI) development framework written in C++. PyQt5 provides a Python interface for Qt5, allowing developers to use all of Qt's functions and create GUI applications by Python.

PyQt5 is easier to learn and use than the traditional C++ development method, while still having the powerful functions of the Qt framework. PyQt5 supports various commonly used interface components, such as buttons, labels, text boxes, and tables. It also realizes interactive responses between controls through the ”signal and slot” mechanism. This mechanism makes the interface logic clearer and the structure more flexible.

In this project, PyQt5 is used for implementing the functional logic behind the interface, including button click responses, input data processing, and result display of driving behavior. Because the Python language has concise syntax and convenient debugging, using PyQt5 greatly improves development efficiency.

Qt Designer, a GUI tool provided by Qt, supports the visual creation and editing of user interfaces (shown in Figure 2.3). Unlike traditional code-based interface development, Qt Designer uses a ”drag-and-drop” method to quickly build a user interface that meets the requirements by directly adding controls and setting layouts and properties on the canvas.

After finishing the design, Qt Designer saves it as a `.ui` file in XML format. It can then be converted into a Python source code file by using the `pyuic5` tool provided by PyQt, or dynamically loaded at runtime through the `uic` module. This design method effectively separates the interface layer from the logic layer, improving the readability and maintainability of the code.

In this project, Qt Designer is widely used for designing the layout of the main window interface, including the window structure, button arrangement, and text display area. Most parameters, such as layout, size, font, and margin, of interface elements can be directly set in the Qt Designer, reducing the workload of manual code adjustments and ensuring interface consistency and aesthetics.

2.2 Hardware introduction

The experimental system has a layered architecture and is divided into four logical layers: data acquisition, data processing, interactive control, and the user interface. This modular structure improves the system’s scalability, functional separation, and maintainability. Standardized interfaces guarantee real-time communication and data flow between layers, facilitating the flexible expansion and reliable integration of software and hardware components.

2.2.1 Data Acquisition Layer

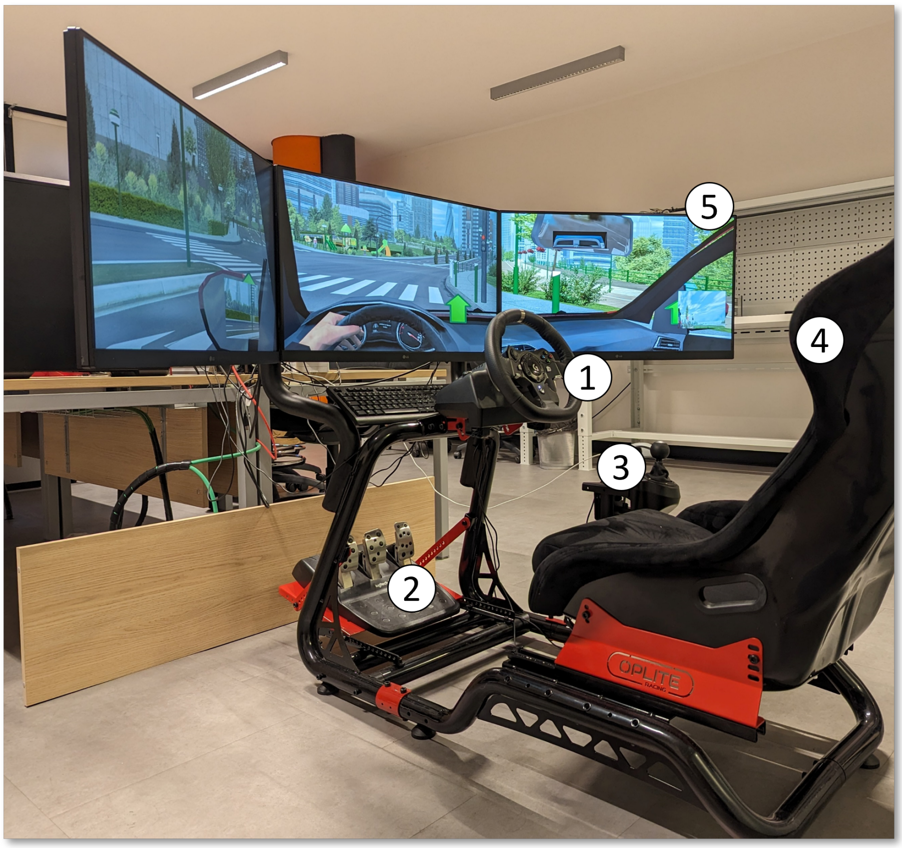


Figure 2.4: Driving simulator paired with SCANeR™ Studio. 1: force feedback steering wheel, 2: pedals, 3: manual gearbox, 4: bucket seat, 5: high resolution screens.

The data acquisition layer captures real-time driving input signals and reproduces the real driving environment (shown as Figure 2.4). The system uses a Logitech G290 force feedback steering wheel to simulate the tactile feel of real vehicle dynamics by providing physical feedback through steering resistance and vibration. This device includes a three-pedal system: clutch, accelerator, and brake, as well as a manual gear lever, which simulates a manual transmission and enhances driver engagement.

The cockpit is equipped with a racing bucket seat designed to provide ergonomic support and immersive feedback, further enhancing the driving experience. All driver operations, steering, acceleration, braking, and gear changes are continuously captured and input into the SCANeR™ Studio simulation environment, which is the central source of real-time driving data, road conditions, and scene interactions. This combination ensures that drivers experience a high-fidelity simulation that closely resembles real-world conditions.

2.2.2 Data Processing Layer

The data processing layer focuses on the real-time interpretation and calculation of vehicle dynamics, behavior classification, and control signal analysis. This functionality is provided by the Speedgoat real-time target machine (shown as Figure 2.5): a high-performance embedded computing system designed for HiL testing and rapid control prototyping.

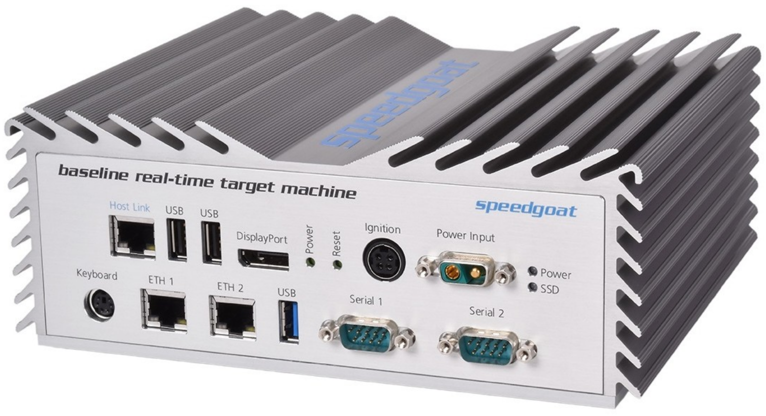


Figure 2.5: Speedgoat Baseline Education system (Baseline S).

The system is built around an Intel Celeron quad-core processor operating at 2.0 GHz with 4 GB DDR3 memory and a 256 GB SSD for storage. This configuration provides sufficient computational power for real-time simulations in MATLAB and Simulink Real-Time environments. The hardware architecture supports up to four I/O modules through mPCIe slots, enabling integration of various signal types, including analog, digital, and FPGA-based modules. Communication capabilities include an IO691 dual-channel CAN module with M12 connectors for automotive and industrial protocols, four Intel I210 Gigabit Ethernet controllers (three external RJ45 ports plus one host link), and two RS232 serial ports supporting transfer rates up to 115.2 kBaud. Additional interfaces comprise USB 3.0/2.0 ports and DisplayPort video output supporting resolutions up to 2560×1600 pixels. The system operates from an 8-36 VDC power input with a maximum consumption of 70W, making it suitable for both laboratory and mobile applications. Environmental specifications include an operating temperature range of 0°C to 60°C and humidity tolerance of 10-90% non-condensing. The fanless design with passive cooling through an integrated aluminum heatsink ensures silent operation while maintaining thermal stability. Physical dimensions of 210×190×80 mm and a weight of 2.6 kg contribute to its portability for educational use.

The Speedgoat is equipped with a built-in real-time operating system (RTOS) that can execute control algorithms with microsecond-level precision. Control logic developed in Simulink is automatically converted into deployable C code using Simulink Coder, which is then executed deterministically on the Speedgoat platform without the need for external compilers or other development environments. This allows the system to carry out low-latency, closed-loop calculations, such as the real-time evaluation of vehicle dynamics, processing of driver inputs, and classification of driving behavior. The generated data is then seamlessly transferred to the interactive control layer.

2.2.3 Interactive control Layer

This interactive control layer acts as a bridge between system intelligence and human-machine feedback. Built on the Raspberry Pi 4 Model B (shown as Figure 2.6): a versatile, credit-card-sized single-board computer featuring a powerful quad-core ARM Cortex-A72 processor operating at 1.5 GHz, coupled with up to 8 GB of high-speed LPDDR4-3200 SDRAM for enhanced multitasking capabilities.

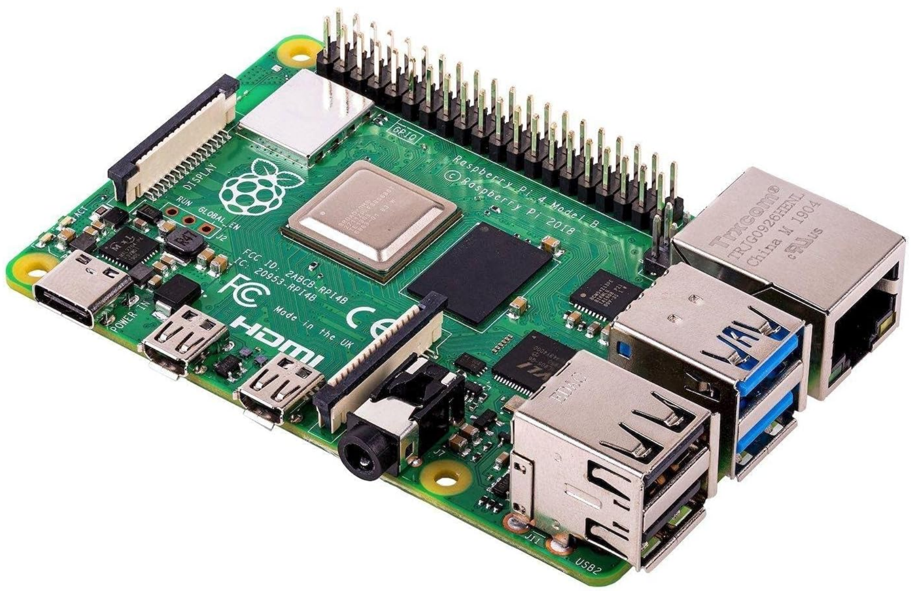


Figure 2.6: Raspberry Pi 4 Model B.

The platform incorporates comprehensive connectivity options, including dual-band 802.11ac Wi-Fi, Bluetooth 5.0, Gigabit Ethernet, and multiple USB ports (2× USB 3.0, 2× USB 2.0) for seamless peripheral integration. Additionally, it features dual micro-HDMI outputs supporting 4K resolution at 60Hz, enabling high-definition multimedia applications. The Raspberry Pi runs a customized Linux-based operating system (typically Raspberry Pi OS) that provides a stable, lightweight environment for executing real-time Python scripts with minimal latency.

The system leverages extensive open-source libraries and frameworks for diverse functionalities, including TCP/IP and serial data communication protocols, sophisticated HMI logic implementation, GPIO-based sensor interfacing, and multimedia processing for audio/video output. The platform's 40-pin GPIO header enables direct hardware interfacing for custom sensor integration and control applications, while the CSI camera interface and DSI display interface provide additional expansion capabilities for computer vision and custom display solutions.

2.2.4 User Interface Layer

The user interface layer provides visual and auditory channels to provide real-time feedback to the driver. It contains display components that provide an immersive, information-rich experience, replicating real-world driving feedback and communicating system warnings.

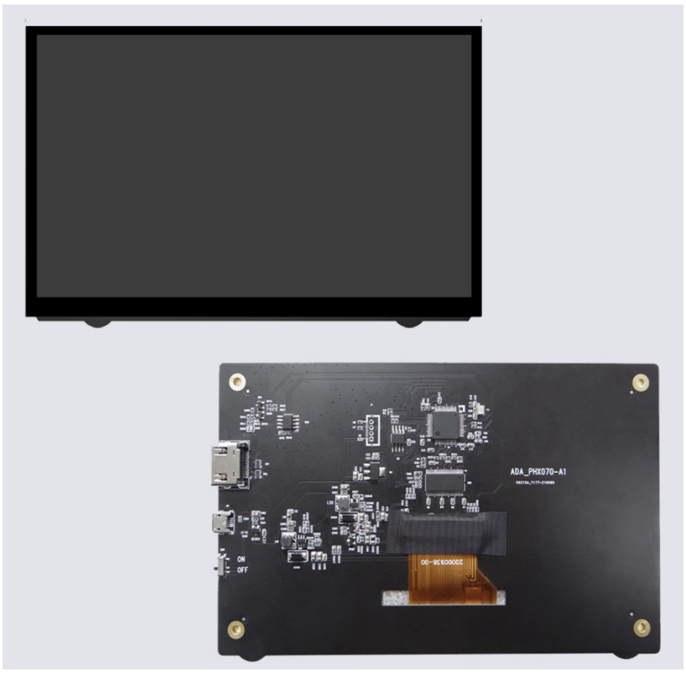


Figure 2.7: LCD Display of HMI.

Primary visual immersion is achieved through three LG high-definition displays, which are configured to provide a panoramic view of the simulated environment.

Additionally, a 6.49-inch auxiliary LCD screen (shown as Figure 2.7) mounted within the driver's field of vision displays different real-time visual feedback related to different driving behaviors.

Meanwhile, audio output devices (speakers) play audible warning prompts when behavior thresholds are exceeded, providing multi-modal feedback to ensure driver awareness and responsiveness.

2.3 Communication Protocol & Physical Connection

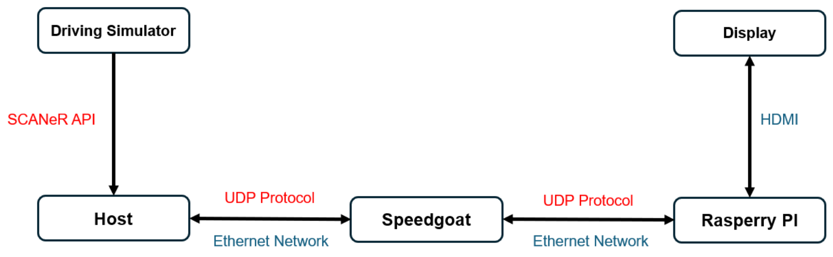


Figure 2.8: Communication protocol (red) and physical connection (Teal) between hardware.

To achieve real-time simulation, deterministic control, and feedback, we design an integrated and low-latency communication network among the host computer, Speedgoat real-time target, and Raspberry Pi 4 Model B.

As shown in the Figure 2.8, the host (which contains simulation platforms like SCANeR™ Studio, MATLAB/Simulink, etc.) connects to the Speedgoat target through a dedicated Gigabit Ethernet link. High-frequency data, like simulation commands, sensor outputs, and control signals, are sent from the host through Speedgoat in the form of IP and UDP with almost no overhead and latency.

Speedgoat real-time target and Raspberry Pi 4 connect to each other through Ethernet, and operate in the same Local Area Network (LAN) environment through fixed IP addressing or the Dynamic Host Configuration Protocol (DHCP) assignment method. Both Speedgoat and Raspberry Pi use UDP in the communication process. Speedgoat sends real-time driving behavior classification results and vehicle states data to the Raspberry Pi through UDP. Raspberry Pi executes user-defined HMI Logic (including driver sensitivity threshold) and returns control instructions or acknowledgement through UDP based on logic analysis and condition judgment. This control structure not only improves computational efficiency but also makes the system more modular and extensible. The HMI system design in this scenario can evolve in parallel with the core simulation platform without dependency.

Raspberry Pi is also connected to a 6.49-inch high-resolution LCD panel through an HDMI connection as the main feedback interface for the driver. It displays different types of context-related warnings, driving behavior summary, and real-time simulation indication etc. Raspberry Pi uses its onboard GPU and Linux-based graphics driver to achieve smooth rendering performance with low latency and temporal conformity to vehicle simulation and control loop.

UDP protocol chosen as the communication protocol for all above mentioned links intentionally. As a basic and fast communication protocol, UDP does not have any connection setup through a three-way handshake process, which significantly reduces communication latency. The protocol design also has a very basic 8-byte header and does not have any reliability mechanism; it is a perfect choice for a real-time system where timeliness is more important than reliable delivery. The protocol is also natively supported in different environments like Simulink, Python, or C/C++, etc., and can achieve seamless integration through different platforms without any protocol wrapper or middleware layer. Even if packet loss happens in any link, it will not cause a significant impact on system performance, as it is running on a high refresh rate and redundant transmission of high-definition image data. From end-to-end, this multi-layer integrated architecture (From host command to hardware control and final display to driver) forms a complete control pipeline. Every link in this pipeline is connected and communicates with each other in an efficient way to achieve a reliable and extendable real-time simulation and control system.

2.4 Data Flow Paths

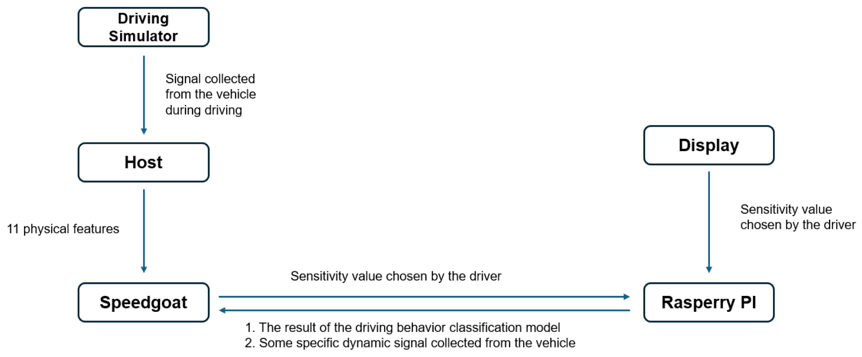

Driving Simulator
Signal collected from the vehicle during driving
Host
11 physical features
Speedgoat
Display
Sensitivity value chosen by the driver
Sensitivity value chosen by the driver
Rasperry PI
1. The result of the driving behavior classification model
2. Some specific dynamic signal collected from the vehicle

Figure 2.9: Data flow path between hardware.

During the simulation process, MATLAB connects to the SCANER™ Studio via the SCANeR API in order to access the real-time vehicle signals that have been collected by the driving simulator. These raw signals include key driving data, such as steering wheel angle, vehicle speed, and acceleration (will be discussed in Section 3.2.1). Simulink then processes these inputs and transforms them into eleven standardized physical features that consistently describe driving behavior.

These features are then transmitted via a high-speed UDP link to the Speedgoat real-time target, which acts as the system's core computing platform. Operating under a deterministic real-time kernel, the Speedgoat target processes the incoming data using a fixed monitoring window size and statistical feature extraction logic (e.g., moving averages, standard deviations). The processed time-series data is then fed into a pre-trained LSTM model, which has been optimized for real-time inference of driving behavior states. Based on learned temporal patterns, the LSTM model provides a predicted classification of the driver's behavior.

In parallel, Speedgoat transmits the resulting behavior signal, alongside selected real-time vehicle dynamics data, to the Raspberry Pi 4 over a UDP-based synchronous communication channel. This low-latency connection ensures that both control signals and behavioral insights are delivered without delay. Upon reception, the Raspberry Pi parses the incoming data and triggers the relevant HMI actions. These include providing visual feedback on the 6.49-inch LCD screen and generating auditory alerts to help the driver be aware of and respond to their driving behavior.

At the same time, the Raspberry Pi also serves as a node for user interaction. Through the HMI, the driver can select their preferred sensitivity value, which determines how tolerant the system is to deviations in driving behavior. This user-defined setting is sent back to Speedgoat via the same UDP channel. Speedgoat then uses this sensitivity input to dynamically adjust the threshold values or weight coefficients in the LSTM output interpretation. This two-way data exchange creates a closed-loop control system that delivers intelligent behavior feedback and tailors the response based on the driver's real-time preferences.

The entire data flow (shown as Figure 2.9), ranging from raw signal acquisition to feature extraction, behavior prediction, HMI generation, and user feedback integration, ensures synchronized, modular, and adaptable system operation.

Chapter 3

Driving Behavior Classification Model

3.1 Scenario Construction

Scenario construction defines the simulation environment in which driving data is generated and collected. This includes designing road networks, placing dynamic and static elements, configuring initial and boundary conditions, and organizing event sequences through SCANER™ Studio's scenario editor. Well-constructed scenarios help ensure the quality, diversity, and relevance of the driving behavior data collected for analysis and model training.

3.1.1 Terrain Construction

The foundation of the simulation environment is the terrain. SCANER™ Studio provides both predefined and user-defined terrains with integrated logic and 3D representations. These terrains simulate real-world driving environments with adjustable complexity and realism. Key components include [35]:

- **Environment types**: built-in layouts include urban roads, highways, rural paths, and complex intersections.
- **Infrastructure**: traffic signals, speed limit signs, road signs, crosswalks, and roundabouts can be added to define vehicle interaction rules.
- **3D assets**: trees, buildings, road barriers, and lighting objects are included to enhance immersion and spatial perception.

In this study, the default terrain: **Riviera** was used (shown as Figure 3.1). Riviera is a nice terrain (Right Hand Traffic (RHT) and Left Hand Traffic (LHT)) with sea, seaside, tunnel, country, village (pedestrian crossing), mountains, and forest. The characteristics of the Riviera Environment are shown in the Table 3.1. This scenario was chosen because it is representative and can cover a variety of driving environments. For example, seaside and tunnel roads are usually narrow, with limited vision and a large turning radius. Drivers are prone to aggressive behaviors such as sharp turns and sudden acceleration when driving at high speeds. Rural roads have complex road conditions and are prone to frequent acceleration and deceleration, causing discomfort to passengers. In addition, inefficient behaviors may be enhanced.

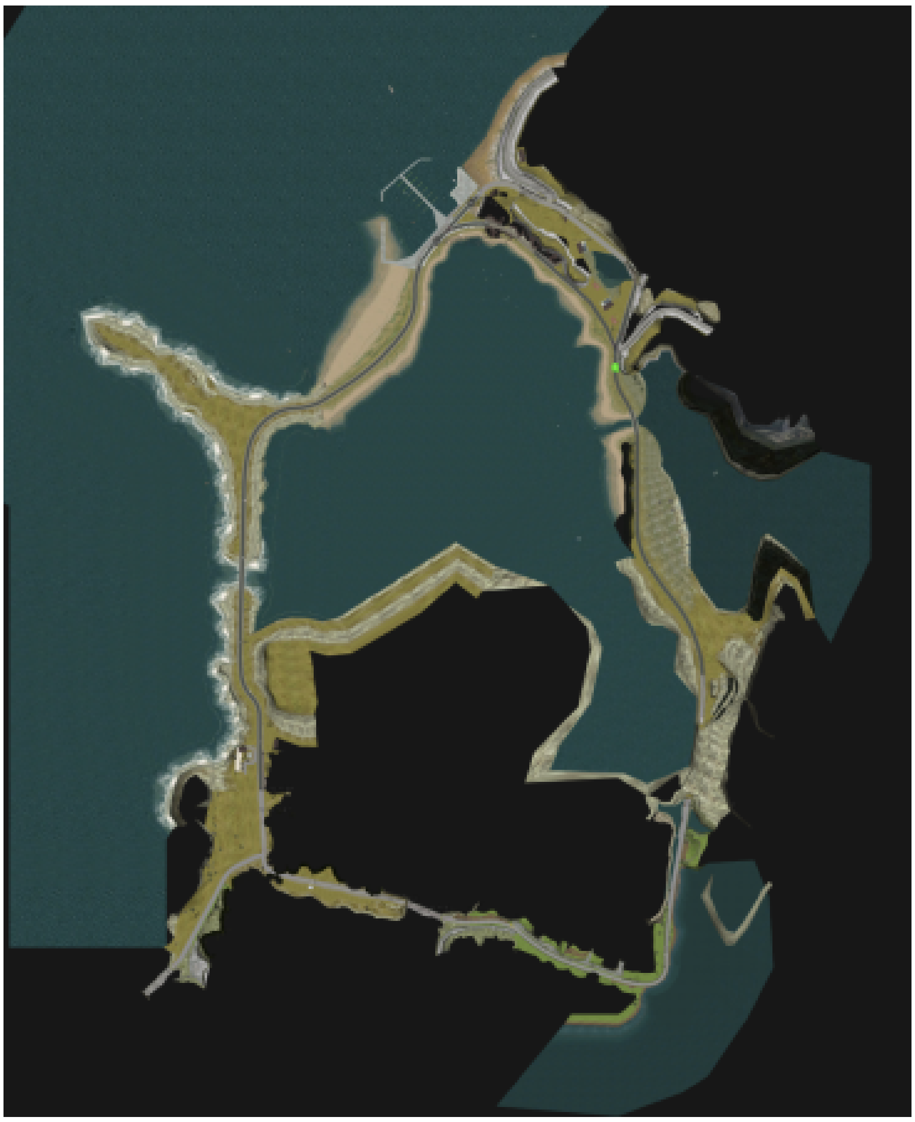


Figure 3.1: Top view of Riviera.

Characteristics	**Quantity**
Road length [km]	5.6
Driving side	RHT / RHT
Traffic lights	0
Barriers	0
Intersections	12
PHYSICS compliant	Yes

Table 3.1: Characteristics of the Riviera Environment.

3.1.2 Resources Selection

As mentioned at the beginning of this chapter, to ensure the simulation environment is realistic and complex, the scene should include terrain as well as 3D objects such as vehicles and pedestrians. In this section, we will provide a brief description of the driving and other vehicles in the scene.

In SCANeR^{TM} Studio, vehicle models are divided into several categories, including simple models, Callas models, and CarSim models.

The simple vehicle's tire, suspension, and steering models are not sufficiently detailed to have an appropriate response to subtle things such as the rumble strips. This simple dynamics model implements basic functions: it uses a dual-axle (two wheels per axle) structure, calculates the vehicle position and achieves terrain following by collecting road information at a single point (usually at the center of the rear axle), and includes basic simulation of the engine, transmission, brakes and steering. However, this model has significant limitations: it does not have a detailed physical component model (such as engine, suspension, etc.), ignores roll motion, the initial state of the vehicle is ”engine start” ready, and only relies on single-point road surface collection, which may cause problems in positioning on slopes, super elevations curves or obstacles. Most importantly, it assumes ”infinite” grip, which means that the vehicle will not slide and its lateral velocity is basically always zero [35].

Autonomous vehicles primarily rely on the traffic module to follow pre-planned routes. In such cases, simple models are typically sufficient. In this study, all autonomous vehicles in the environment are composed of simple models from the default library, including cars, buses, bicycles, motorbikes, trucks, and trailers. Their distribution is shown in the table 3.2. The driving behavior of autonomous vehicles is categorized into three types (normal, cautious, and aggressive), as shown in Table 3.3.

Vehicle type	Vehicle distribution (%)
Cars	65
Buses	5
Bicycles	10
Motorbikes	10
Trucks	5
Trailer assemblies	5

Table 3.2: Distribution of simple vehicles.

Driving behavior	Vehicle distribution (%)
Normal	90
Cautious	5
Aggressive	5

Table 3.3: Driving behavior distribution of simple vehicles.

On the other hand, vehicles that are used for real-time human interaction (via driving simulators, keyboards, or other interfaces) need more accurate dynamics. Simple models don't show how people actually drive in the real world in these situations. For these types of situations, the Callas model is used to provide more realistic vehicle responses that are suitable for experimental needs.

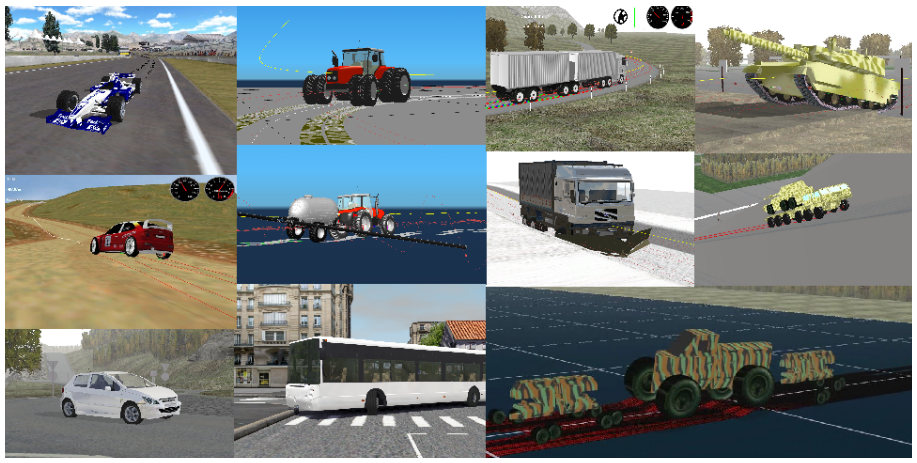


Figure 3.2: Callas Model.

Callas (shown in Figure 3.2) is the name of dynamic model vehicle, such as trucks, buses, cars and midgets, motorsport, machineries, tractors, and military vehicles (such as tracked vehicles) [35]. Unlike the simple model, it includes suspension and powertrain components. Suspension types can adopt all existing geometries, such as stiff axles, independent wheels, tracks (caterpillars), and hybrid drive trains. And the powertrain can be electric or combustion, with a full range of transmission schemes. The specific vehicle model chosen was `SmallFamilyCarElectric` (shown in Figure 3.3), a compact electric car suitable for urban driving simulations.

The specific information is shown in Table 3.4

Figure 3.3: SmallFamilyCarElectric.

Engine	
Aspiration	Electric
Max Power (kW)	80
Electric Motor RPM (rpm)	10390
Max Torque (daN*m)	28
Transmission	
Transmission Type	Front Wheel Drive
Gearbox Technology Type	Auto
Front Gear Ratio Number	1
Rear Gear Ratio Number	/
Dimensions	
Length (mm)	4440
Width (mm)	1770
Height (mm)	1545
Weight (kg)	1523
Front Overhang (mm)	952
Rear Overhang (mm)	788
Wheelbase (mm)	2700
CoG Height from ground (mm)	520
Front Track / Rear Track (mm)	1540 / 1535
Ground Clearance	155

Driver Side	Left
Frame	
Steering wheel turn lock-to-lock	3
Steering diameter between sidewalks (m)	11
Steering diameter between walls (m)	11.8
Tires dimensions	195/65 15
Anti-Block Brake system	yes
Active yaw control	yes
Traction control	yes
Front suspension	Independent McPherson
Rear suspension	Twist Beam
Performances	
Max speed (km/h)	145
0-100 km/h (s)	25.6
Standing 400 m (s)	22.2
Standing 1000 m (s)	42.5
Specific suspension roll (°/G)	3.7
Specific suspension pitch (°/G)	3.97
Max Slope (%)	22
Max Banking (%)	36

Table 3.4: SmallFamilyCarElectric's Technical Document

Additionally, the vehicle is equipped with a Long Range Radar Sensor (see Figure 3.4) in order to detect the distance to collision and is capable of detecting mobile obstacles (for example, cars, pedestrians, bicycles, motorbikes). The sensor is positioned in the car front at a distance from the ground of 0.5 m (see Figure 3.5). Its technical specifications include a maximum detection range of 250 meters, a horizontal field of view (FOV) of $-30°$ to $30°$, and a vertical FOV of $-30°$ to $30°$. In this study, the sampling frequency of all sensor data is set to 100 Hz.

The entire scenario setup, including road logic, dynamic agents, and data logging configurations, ensures that each test run is consistent and repeatable, while also allowing flexibility to modify conditions for specific experimental needs.

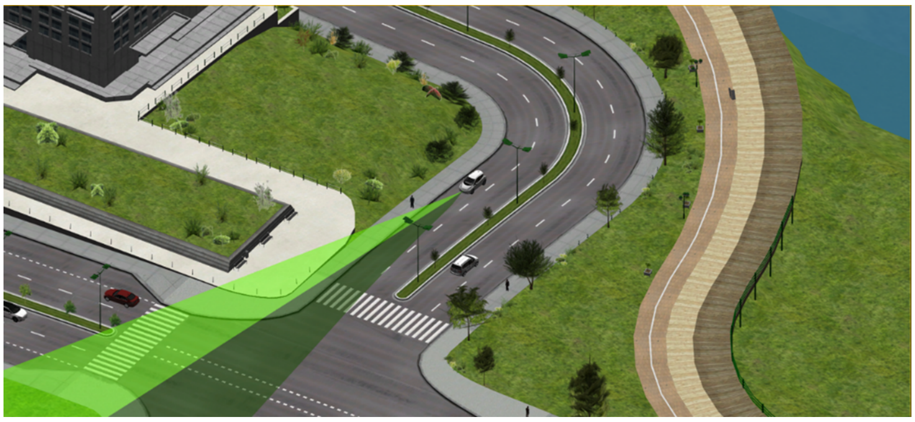


Figure 3.4: Long range radar sensor with maximum beam range of 250m.

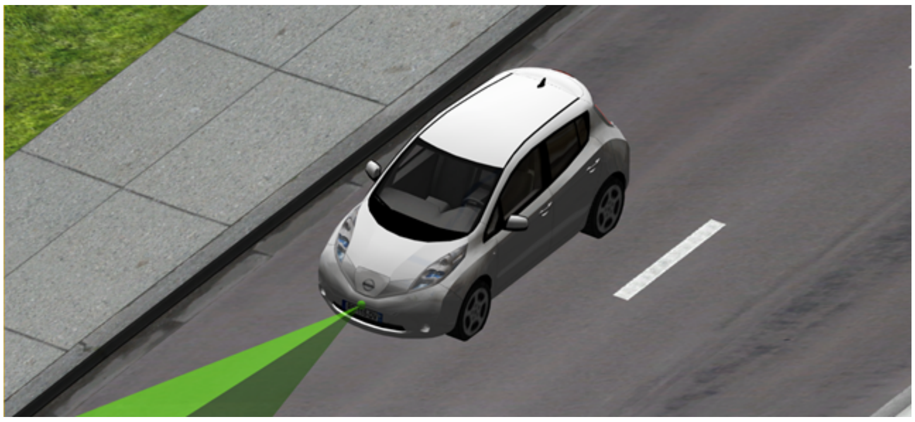


Figure 3.5: Zoomed view of radar sensor placement.

3.2 Methodology

The present study proposes a four-stage driving behavior analysis framework. This framework is based on the SCANER™ studio simulation platform, MATLAB, and Simulink. The technical process of the framework is as follows:

1. **Data Acquisition & Preprocessing:** Raw simulation data is first transformed into a set of interpretable physical features. These features are then processed using a sliding window strategy to compute statistical summaries representing behavior over time. These statistical outputs are then used as both labels and sequence inputs to train a sequential model.

2. **Data Segmentation:** The data is segmented into basic driving behavior units (Elementary Driving Behaviors (EDBs)) based on dynamic metrics such as vehicle speed and steering wheel angle.

3. **Behavioral Characteristics Labeling:** The I-DBSCAN algorithm is employed in the EDBs cluster analysis to identify normal and abnormal driving

behaviors. Then, continue to classify according to the predefined threshold.

4. **Model Training:** A LSTM neural network is used to train the dataset with labels to finally obtain an intelligent recognition system for driving behavior.

   The technical details of each stage are described in the following section.

3.2.1 Data Acquisition & Preprocessing

In the simulation, the traffic driver (autonomous) is used to control an interactive vehicle, designated as a "small family electric car", which has already been described in the last section, to simulate realistic driving behavior in the scenario.

Signals	**Unit**
Longitudinal Speed	m/s
Longitudinal Acceleration	m/s^2
Lateral Acceleration	m/s^2
Steering wheel angle	rad
Steering wheel speed	rad/s
Accelerator pedal	[-]
Brake force	N
Motor Efficiency	%
Distance to collision	m
Relative speed	m/s

Table 3.5: Raw signals collected from the vehicle.

Signals	**Unit**
Longitudinal Speed	m/s
Longitudinal Acceleration	m/s^2
Lateral Acceleration	m/s^2
Steering wheel angle	rad
Steering wheel speed	rad/s
Accelerator pedal	[-]
Brake force	N
Jerk	m/s^3
Distance to collision	m
Efficiency	%
Time to collision	s

Table 3.6: Physical features.

The SCANeR API enables Simulink to receive real-time vehicle dynamics data from SCANER™ Studio at a sampling rate of 100 Hz. These raw signals include key variables such as vehicle speed, longitudinal acceleration, and steering angle (shown in Table 3.5). Some derived features, such as jerk (computed from longitudinal acceleration) and time-to-collision (TTC, calculated from relative speed and distance to collision), are also computed during this stage. A total of 11 physical driving features are extracted to form the basis for behavior analysis (see Table 3.6).

The time-series data is analyzed using a sliding time window strategy to understand instantaneous and short-term behavior. Specifically, the continuous data is segmented into Monitoring Period (MP)—each with a fixed window size of 0.3 seconds (i.e., 30 data points) and a sliding interval of 0.1 seconds (i.e., 20-point overlap) [36].

Statistical Function	**Description**
Mean	Mean of a signal
Min	Minimum value of a signal
Max	Maximum value of a signal
Variance	Square of the standard deviation of a signal
STD	Standard deviation of a signal
RMS	Root mean square
Q_1	25th percentile
Q_2	50th percentile
Q_3	75th percentile
Peak amplitude	Difference between the maximum and minimum value of the signal

Table 3.7: Statistical functions for feature engineering.

At each sampling point, two parallel datasets are generated (as shown in Figure 3.6):

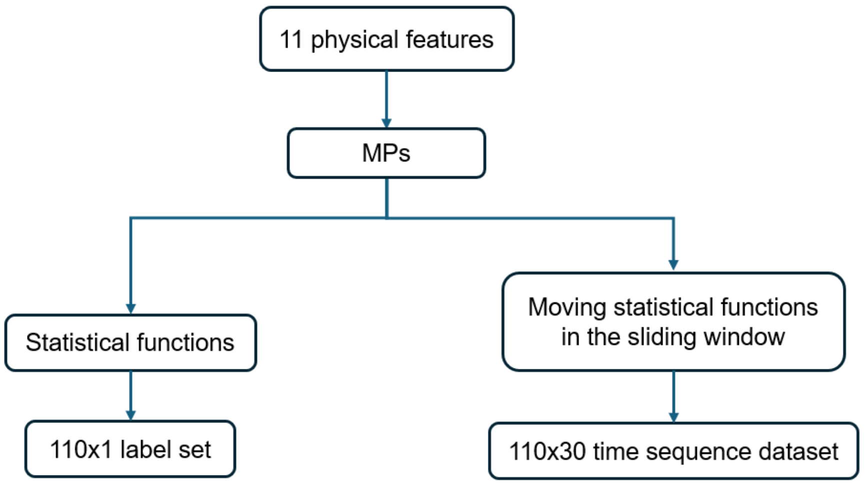

11 physical features
MPs
Statistical functions
Moving statistical functions in the sliding window
110x1 label set
110x30 time sequence dataset

Figure 3.6: Parallel datasets generated after data preprocessing.

- **Label dataset** (110 × 1):

For each 0.3-second window, ten time-domain statistical functions (e.g., mean, standard deviation, minimum and maximum... see Table 3.7) are applied to each of the 11 physical features. This results in a 110-dimensional vector, which is then labeled using I-DBSCAN to summarize the behavioral characteristics during that short time period. These vectors are used as a ground-truth reference in model training.

- **Sequential dataset** (110×30):

  Rather than compressing each monitoring window into a single statistical value, this dataset preserves the dynamic changes of each statistic in the full 0.3-second window and 30 consecutive time steps. In this way, the data in each window forms a 110 (feature dimension) $\times$ 30 (time step) matrix, maintaining both the diversity of features and the fine temporal structure. This matrix structure is sent as the input sample to the LSTM network, enabling the model to learn and identify the temporal dynamic features and their changing patterns contained in driving behavior.

3.2.2 Data Segmentation

In order to provide a more detailed description of driving behavior, we divide the label dataset into 15 subsets of behaviors (EDBs), including straight driving, turning when the degree is slight or sharp under low, medium, and high velocity conditions. Each EDB corresponds to a data profile. Even if the data profiles are similar, aggressive behaviors will stand out from the average behavior pattern [37] [38].

To achieve the subdivision of these behavior subsets, we combine the K-means clustering algorithm and threshold segmentation to process the data. The k-means clustering algorithm (shown as Alg. 1) is a widely used unsupervised learning method, whose purpose is to divide n samples into k clusters so that each sample belongs to the cluster center closest to it, thereby minimizing the variance within the cluster [39].

The specific content of this k-means algorithm is as follows:

$$E = \arg\min_{S} \sum_{i=1}^{k} \sum_{n \in S_i} \|x - \mu_i\|^2 \tag{1}$$

In this context k is the number of clusters, S_i represents the ith cluster, μ_i is the center of the cluster, and the calculation formula is:

$$\mu_i = \frac{1}{|S_i|} \sum_{n \in S_i} x \tag{2}$$

The overall process is to continuously adjust the division of clusters so that the points in each cluster are as close to the cluster centroids as possible, thereby achieving effective clustering.

Algorithm 1 K-means clustering pseudocode

```
1: Initialise Cluster Centers
2: for each iteration $l$ do
3:     Compute $r_{nk}$:
4:     for each data point $x_n$ do
5:         Assign each data point to a cluster:
6:         for each cluster $k$ do
7:             if $k == \arg\min \|x_n - \mu_k^{l-1}\|$ then
8:                 $r_{nk} = 1$
9:             else
10:                $r_{nk} = 0$
11:            end if
12:        end for
13:    end for
14:    for each cluster $k$ do
15:        Update cluster centers as the mean of each cluster:
                   $\mu_k^l = \frac{\sum r_{nk} x_n}{\sum r_{nk}}$
16:    end for
17: end for
```

3.2.3 Behavioral characteristics Labeling

After successfully identifying the subset of EDBs, this thesis uses the DBSCAN algorithm to identify and differentiate the aggressive driving behaviors within each EDB.

Before the specific analysis, the DBSCAN algorithm is introduced, and its pseudo-code structure is presented (Alg.2).

DBSCAN is a density-based clustering algorithm. Its basic idea is to identify clustering structures by evaluating the density around data points. The algorithm

Algorithm 2 DBSCAN Clustering Algorithm

```
Require: Dataset $D = \{p_1, p_2, \ldots, p_n\}$, neighborhood radius $\varepsilon$, minimum points $MinPts$
Ensure: Clusters $C = \{C_1, C_2, \ldots, C_k\}$ and noise points $N$
 1: Initialize all points in $D$ as UNVISITED
 2: $C \leftarrow \emptyset$, $N \leftarrow \emptyset$, $cluster\_id \leftarrow 0$
 3: for each point $p \in D$ do
 4:   if $p$ is UNVISITED then
 5:     Mark $p$ as VISITED
 6:     $NeighborPts \leftarrow$ all points within distance $\varepsilon$ of $p$
 7:     if $|NeighborPts| < MinPts$ then
 8:       Mark $p$ as NOISE and add to $N$
 9:     else
10:       $cluster\_id \leftarrow cluster\_id + 1$
11:       Create new cluster $C_{cluster\_id}$ and add $p$
12:       $Seeds \leftarrow NeighborPts \setminus \{p\}$
13:       while $Seeds \neq \emptyset$ do
14:         $q \leftarrow$ select point from $Seeds$ and remove from $Seeds$
15:         if $q$ is UNVISITED then
16:           Mark $q$ as VISITED
17:           $NeighborPts' \leftarrow$ all points within distance $\varepsilon$ of $q$
18:           if $|NeighborPts'| \geq MinPts$ then
19:             $Seeds \leftarrow Seeds \cup NeighborPts'$
20:           end if
21:         end if
22:         if $q$ does not belong to any cluster then
23:           Add $q$ to cluster $C_{cluster\_id}$
24:         end if
25:       end while
26:       $C \leftarrow C \cup \{C_{cluster\_id}\}$
27:     end if
28:   end if
29: end for
30: return $C, N$
```

mainly relies on two parameters: ϵ (neighborhood radius) and *minPts* (minimum number of points in the neighborhood). When the ϵ neighborhood of a sample point contains at least *minPts* samples, the point is considered a *core point.* DBSCAN divides different clusters according to density accessibility and identifies points in sparse areas as noise points.

In the dataset, DBSCAN classifies samples into three categories:

- **Core Point:** If the number of points contained in the ϵ neighborhood of a point p is not less than *minPts*, then p is a core point. Its mathematical expression is:

$$N_\epsilon(p) \geq \text{MinPts}, \quad N_\epsilon(p) = \{q \in D \mid \text{dist}(p, q) \leq \epsilon\}$$

- **Border Point:** A sample h that does not meet the core point condition is called a border point if it is located in the ϵ neighborhood of a core point p.

$$h \in N_\epsilon(p)$$

- **Noise Point:** If a sample is neither a core point nor in the ϵ neighborhood of any core point, the point is marked as a noise point.

$$n \notin N_\epsilon(p)$$

Generally, the value of *minPts* is recommended to be twice the number of features after Principal Component Analysis (PCA) dimensionality reduction [40]. High-dimensional data can lead to a problem known as the ”curse of dimensionality”. PCA helps to improve the performance of the clustering algorithm. The selection of ϵ adopts the **Elbow Method**: calculate the distance between each point and its K-th nearest neighbor (where K = *minPts*), sort these distances in descending order, and draw the sorted k-distance graph to find the best ϵ value [41].

In this study, to further identify abnormal driving behaviors, the I-DBSCAN algorithm (the pseudo-code is as follows 3) is introduced [36]. The specific steps are as follows [36][37]:

1. Determine the basic parameters required by DBSCAN: *minPts* and ϵ;
2. Set a threshold *normPercent* for the proportion of normal driving behavior, and check whether this requirement is met after executing DBSCAN;

3. Based on the clustering results, divide the samples into three categories: normal behavior, abnormal behavior, and noise;

4. If the proportion of normal driving samples reaches or exceeds *normPercent*, terminate the algorithm; otherwise, re-cluster only the samples currently judged as normal behavior, and continue to iterate until the condition is met.

Algorithm 3 I-DBSCAN Algorithm

```
1: **Input:** data, normPercent, stopThresh
2: **Output:** abnormal
3:
4: $abnormal \leftarrow \emptyset$
5: $stopCounter \leftarrow 0$
6: $minPts \leftarrow \text{dimensions}(data) \times 2$
7:
8: **while** $stopCounter < stopThresh$ **do**
9:     $\varepsilon \leftarrow \text{identifyEps}(data, minPts)$
10:    $results \leftarrow \text{DBSCAN}(data, minPts, \varepsilon)$
11:    **if** cluster exists that is $normPercent$ of data **then**
12:        $normal \leftarrow$ that cluster
13:    **else**
14:        **throw error** (change parameters)
15:    **end if**
16:    $abnormal \leftarrow abnormal \cup (data \setminus normal)$
17:    **if** only one cluster found **then**
18:        $stopCounter \leftarrow stopCounter + 1$
19:    **end if**
20:    $data \leftarrow normal$
21: **end while**
22:
23: **return** $abnormal$
```

To further complete the classification of driving behaviors, samples with motor efficiency below the 25th percentile are labeled as low-efficiency driving behaviors.

And behaviors with RMS values of jerk exceeding $1.67 m/s^3$ are identified as comfort-affecting driving behaviors [42]. These behaviors, typically characterized by frequent acceleration or deceleration, may lead to passenger discomfort or motion sickness.

The complete labeling steps are shown in the Figure 3.7.

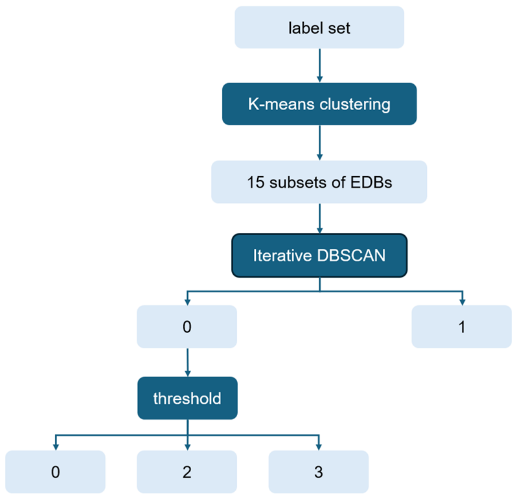


Figure 3.7: Generally data labeling steps.

3.2.4 Model Training

Once a labeled dataset is obtained, a LSTM neural network based on Bayesian optimization is utilized to learn to classify driving behaviors.

LSTM is an enhanced recurrent neural network architecture designed to model temporal dependencies through specialized gating mechanisms. Its core components include three adaptive gates: the forget gate (f_t), input gate (i_t), and output gate (o_t). These gates collaboratively regulate information flow to mitigate gradient-related challenges (e.g., vanishing/exploding gradients) in long-sequence training, thereby improving long-term dependency learning.

The network states are governed by the hidden state z_t and cell state c_t, updated as follows:

$$\begin{aligned} z_t &= o_t \odot \tanh(c_t), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tanh\left(W_c[z_{t-1}, x_t] + b_c\right), \end{aligned}$$

where $\odot$ denotes the Hadamard product. The gates employ sigmoid activation (σ) to dynamically modulate information:

$$\begin{aligned} f_t &= \sigma\left(W_f[z_{t-1}, x_t, y_t, s] + b_f\right), \\ i_t &= \sigma\left(W_i[z_{t-1}, x_t, y_t, s] + b_i\right), \\ o_t &= \sigma\left(W_o[z_{t-1}, x_t, y_t, s] + b_o\right). \end{aligned}$$

Here, $W_{(\cdot)}$ and $b_{(\cdot)}$ represent scalable weight matrices and biases, respectively. The notation z_{t-1} denotes the prior hidden state, x_t the current input, y_t the target variable, and s static metadata [43]. By selectively retaining, updating, and exposing state information, LSTM autonomously captures complex temporal patterns, achieving robust performance in sequential data tasks.

When building an LSTM neural network, it is important to choose the correct initial parameters. The optimized parameters provide a good starting point for the LSTM model, allowing the model to converge faster and achieve higher prediction accuracy. Traditional methods, such as network search and random search, require a large number of blind tests of many parameter combinations, which is computationally expensive. Therefore, we use the Bayesian optimization method to determine the key initial parameters of the LSTM model, including the dropout rate, initial learning rate, and L2 regularization coefficient. Bayesian optimization can sample smartly in the parameter region where the optimal solution is more likely to be found based on the existing evaluation results. This targeted method greatly reduces the amount of calculation and the time required to find the nearest parameters.

3.3 Performance evaluation

A total of 10 hours of data were collected, with a total of 354,768 samples. We used k-means clustering (k=3) to group driving data into three speed categories: low, medium, and high (the result is shown in Figure 3.8). Each speed group was then further divided based on steering angle: straight ($\leq 10°$), slight turn ($10 - 45°$), and sharp turn ($> 45°$). Positive and negative angles represent right and left turns, respectively. This resulted in a final dataset with 15 different speed-steering combinations. The specific process is shown in the Figure 3.9.

Before applying I-DBSCAN, we used PCA to reduce the dimensionality of our data. PCA identified 15 principal components explaining 90% of the data variance (Figure 3.10). For I-DBSCAN, we set minPts to 30 (twice the number of principal components) and determined the optimal epsilon for each of the 15 datasets using elbow diagrams (Figure 3.11). I-DBSCAN was run multiple times per dataset, stopping when clusters encompassing at least 80% of the data were found. This process identified 286,641 samples of normal driving (labeled 0) and 68,127 samples of aggressive driving (labeled 1).

We compared some key features (average, mean maximum, and mean minimum values) of normal and aggressive driving behaviors (Table 3.8). Significant differences were found in the mean maximum and mean minimum longitudinal acceleration, showing the effectiveness of the algorithm. Aggressive driving showed greater variability (higher standard deviation) than normal driving, which exhib-

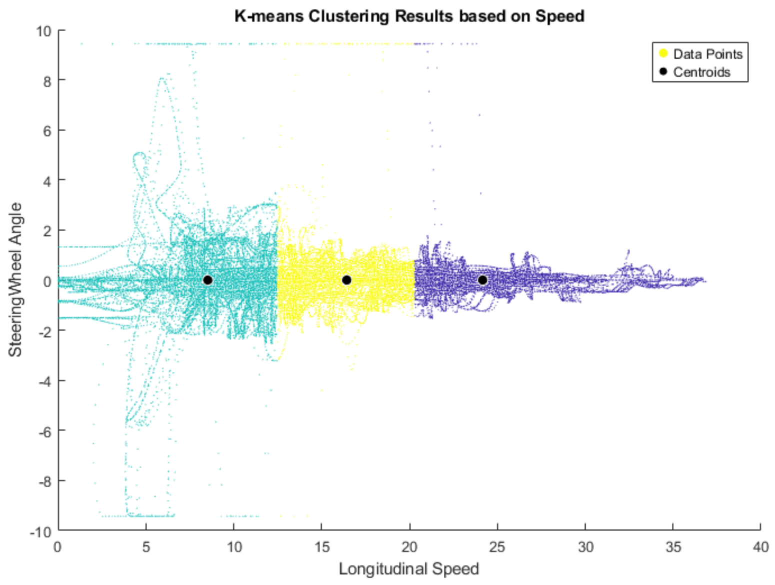


Figure 3.8: K-means clustering of speed. Black dot: centroid, cyan: low-speed data point, yellow: medium-speed data points, purple: high-speed data points.

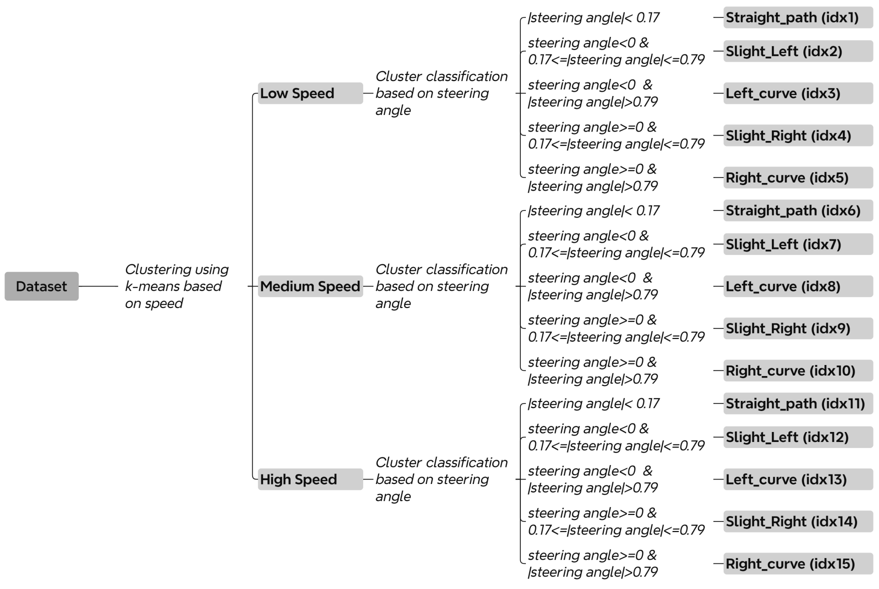


Figure 3.9: The clustering steps of EDB.

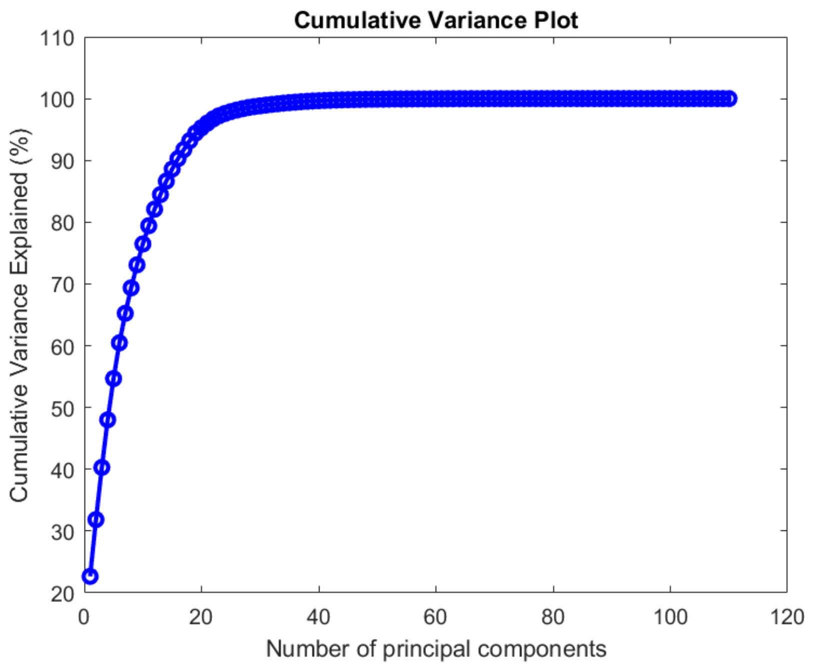


Figure 3.10: Principal component analysis.

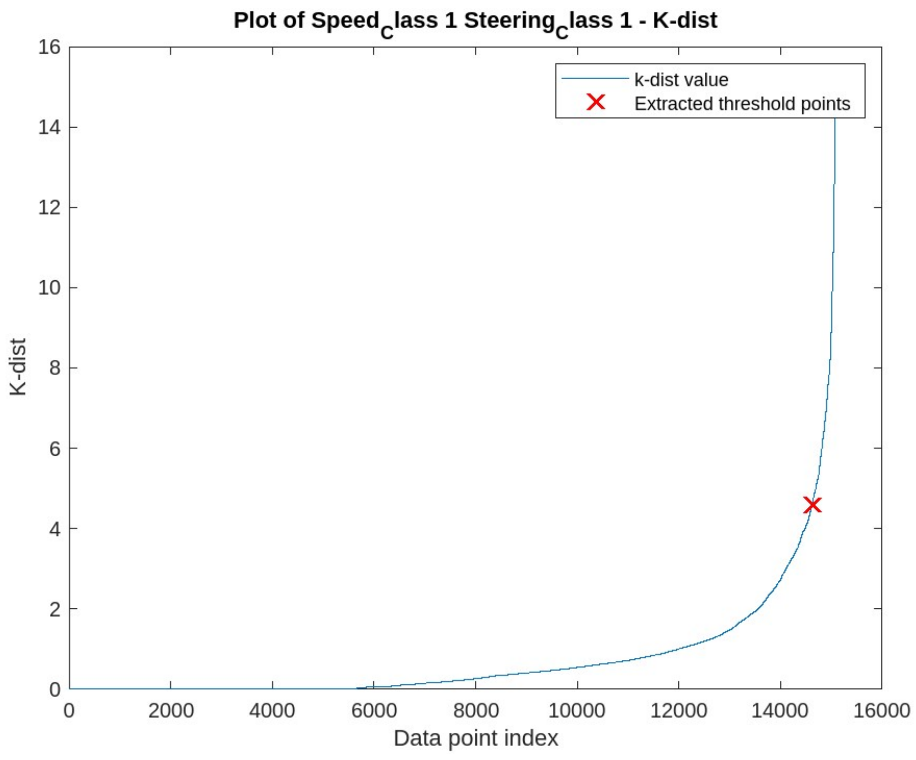


Figure 3.11: K-dist plot. Red cross: chosen ϵ.

ited more consistent behavior. Figure 3.12 illustrates some overlap between normal and aggressive driving clusters based on longitudinal acceleration; however, other kinematic features likely contribute to the separation of these clusters.

Features	**Units**	**Normal**	**Aggressive**
Average Speed	(m/s)	16.46 (6.17)	15.43 (8.56)
Average Longitudinal Acceleration	(m/s^2)	0.04 (1.04)	-0.2 (1.35)
Average Lateral Acceleration	(m/s^2)	-0.11 (1.55)	0.12 (1.47)
Average Jerk	(m/s^3)	-0.01 (1.99)	0.05 (3.24)
Max Speed	(m/s)	16.58 (6.16)	15.6 (8.56)
Max Longitudinal Acceleration	(m/s^2)	0.43 (0.77)	0.28 (1.16)
Max Lateral Acceleration	(m/s^2)	0.007 (1.54)	0.33 (1.56)
Max Jerk	(m/s^3)	10.95 (17.4)	12.86 (18.37)
Min Speed	(m/s)	16.33 (6.17)	15.26 (8.56)
Min Longitudinal Acceleration	(m/s^2)	-0.39 (1.47)	-0.74 (1.71)
Min Lateral Acceleration	(m/s^2)	-0.23 (1.57)	-0.09 (1.47)
Min Jerk	(m/s^3)	-10.85 (17.03)	-12.54 (17.68)

Table 3.8: Statistical feature of two labels. Values in round bracket are standard deviation.

After categorizing driving behaviors as aggressive and normal, we tried to find uncomfortable driving behaviors using a jerk RMS threshold of $1.67 m/s^3$ and low energy efficiency using the 25th percentile of average efficiency. Three models were trained:

- **Model 1:** Classifies driving behaviors as normal, aggressive, and uncomfortable.
- **Model 2:** Categorizes driving behaviors as normal, aggressive, and low energy efficient.
- **Model 3:** Integrates all four labels (normal, aggressive, uncomfortable, and inefficient)

3.3.1 Model 1 (normal, aggressive, uncomfortable)

The specific labeling flowchart is shown in Figure 3.13. Based on this classification method, we successfully divided the data into three categories: 43,761 samples representing normal driving behavior (labeled as 0); 1,062 samples representing general aggressive driving behavior (labeled as 1); 351 samples representing driving behaviors that affect comfort (labeled as 2).

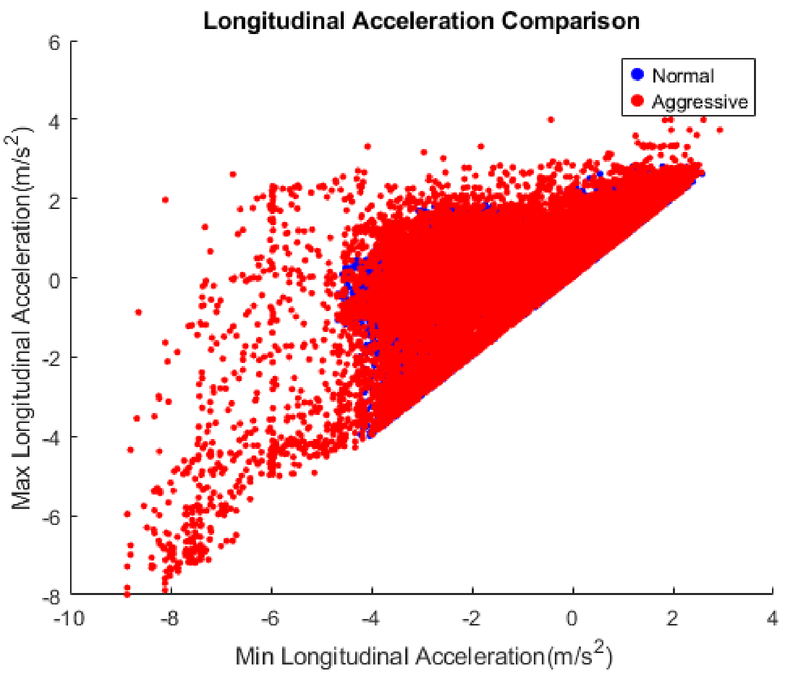


Figure 3.12: Max longitudinal acceleration vs Min longitudinal acceleration. Red dots: aggressive behavior, blue dots: normal behavior.

After completing the labeling of the three types of driving behaviors, we input the labeling results, the serialized data set with a time step of 30, and the final initial parameters determined by Bayesian optimization (see Table 3.9) into the LSTM neural network for training. This setting helps the model learn different behavior patterns more effectively and improves recognition accuracy and convergence speed.

Hyperparameters	**Value**
LSTM number of layers	2
Number of hidden units	128
Maximum epochs	75
Batch size	128
Dropout rate	0.3
Initial learn rate	4.3595e-4
L2 Regularization	1.0736e-5
Loss function	Cross-Entropy

Table 3.9: Hyperparameters of LSTM network for uncomfortable recognition.

The confusion matrix can be used to evaluate the performance of the classification model. It can intuitively show the correspondence between the model's prediction results and the true labels in each category. Especially in multi-category tasks, it can not only reflect the model's ability to identify each category accurately, but also show the confusion that may exist between different categories. Each row of the matrix represents the true label, and each column represents the model's pre-

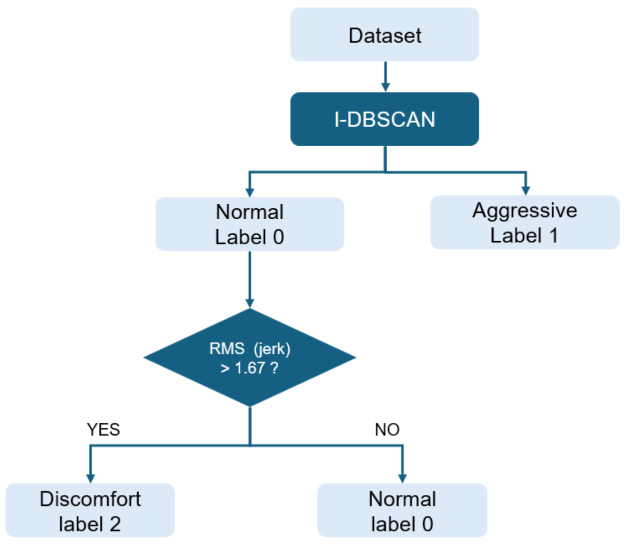


Figure 3.13: Specific labeling step of normal, aggressive, and uncomfortable.

diction result. Ideally, in an ideal condition most samples should be distributed in the diagonal position, that is, the prediction and the truth are consistent.

In this model 1, the confusion matrix is used to evaluate the performance of the LSTM model in three types of driving behaviors: aggressive driving, uncomfortable driving, and normal driving. As can be seen from Figure 3.14, the model performs best at identifying normal behavior. There are 34,050 samples in total, 33,570 of which are accurately identified, with an accuracy rate of 98.6%. The recognition effect of the discomfort category is also relatively stable, with 20,399 of 21,072 samples correctly classified, with an accuracy rate of 96.8%. The aggressive category is relatively difficult to identify, with 11,217 of 12,392 samples correctly classified, with an accuracy rate of 90.5%. It can be seen that aggressive driving and uncomfortable driving have high similarities in driving characteristics, so aggressive behavior is more likely to be confused with discomfort driving behavior.

The F1 score should also be analyzed. The F1 score is a balanced indicator that takes into account both precision and recall. In simple terms, it tells us how well the model performs in avoiding false positives and false negatives. An F1 score close to 1.0 indicates that the model is very reliable in identifying correct behaviors and has a low error rate. The F1 values of the three types of driving behaviors are shown in Table 3.10. The results show that the overall performance of the model is balanced in the three categories, especially in the normal and discomfort categories with larger sample sizes. Performance is more stable, and the F1 score for the aggressive category is slightly lower, which means that the model still has room for improvement in identifying this category.

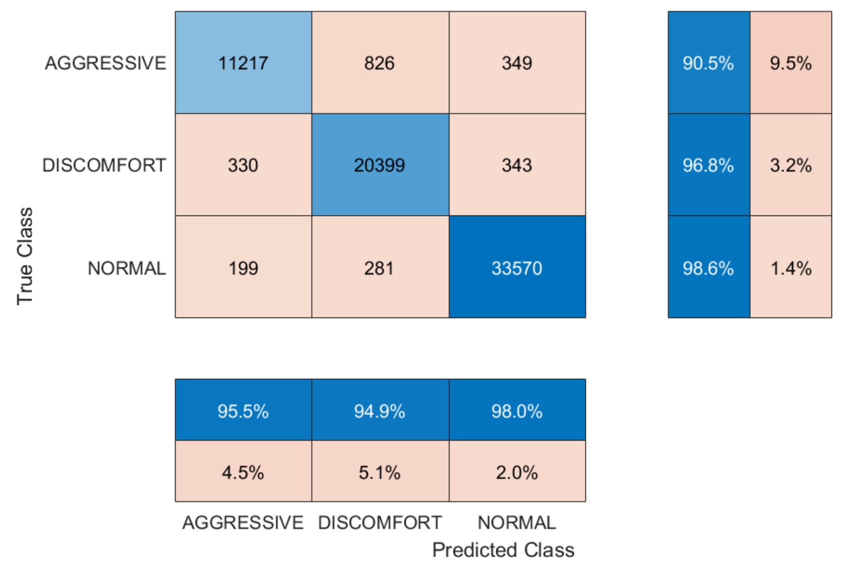


Figure 3.14: Confusion matrix for uncomfortable recognition. Below the matrix: precision for each label, right side of the matrix: recall for each label.

	F1 score
Aggressive	0.93
Discomfort	0.96
Normal	0.98

Table 3.10: Performance of LSTM model 1.

Figure 3.15 shows how the model identifies driving behavior based on feature input. The LSTM model outputs three values: 0 for normal driving, 1 for aggressive driving, and 2 for uncomfortable driving. By observing the jerk RMS value, it is obvious that in the time intervals of 1.5 seconds to 2.1 seconds and 2.5 to 3.2 seconds, the value is too large, which will lead to extremely poor ride comfort, and the classification value of the neural network has also become label 2. At the same time, by observing the longitudinal acceleration, we noticed a significant mutation around 3 s. From -2 m/s^2 to nearly -8 m/s^2, it is an intense braking behavior. Since aggressive driving has a higher priority than uncomfortable driving, the output value of the neural network changes from 2 to 1 in the time interval from 3 to 3.2 seconds.

3.3.2 Model 2 (normal, aggressive, low energy efficiency)

As depicted in the classification flow chart (Figure 3.16), the data were categorized into three groups: 43,761 samples of normal driving (labeled 0), 1,062 samples of aggressive driving (labeled 1), and 351 samples of low-energy driving (labeled 2).

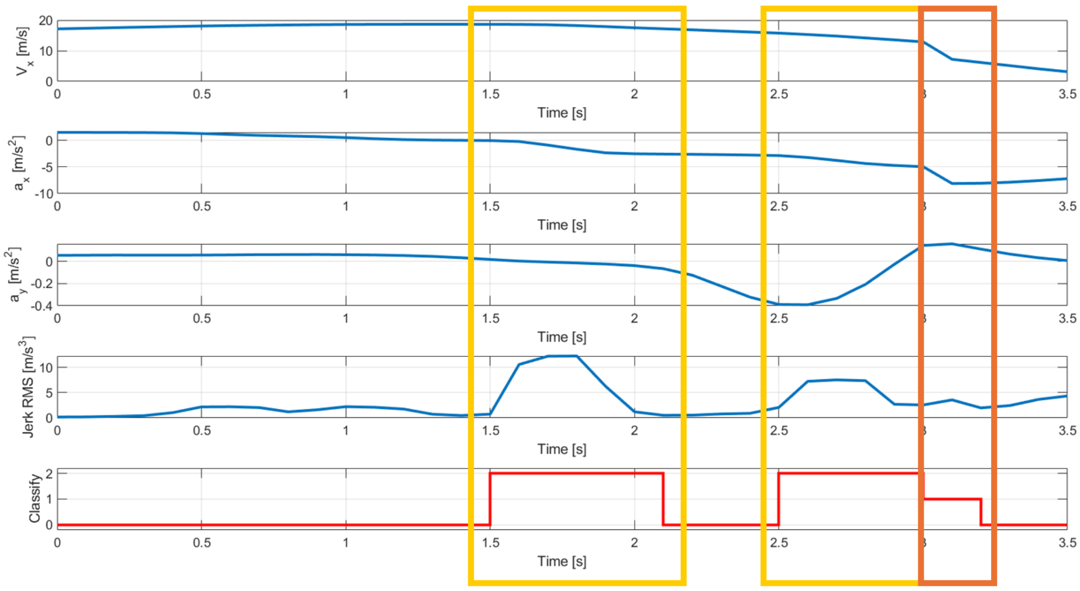


Figure 3.15: Neural network input and output. The first row: longitudinal velocity signal, the second row: longitudinal acceleration signal, the third row: lateral acceleration signal, the fourth row: jerk RMS value, the fifth row: driving behavior classification (0:normal; 1:aggressive; 2:discomfort). The orange box highlights the aggressive driving, and the yellow box highlights the discomfort driving.

Hyperparameters	**Value**
LSTM number of layers	2
Number of hidden units	128
Maximum epochs	75
Batch size	128
Dropout rate	0.4
Initial learn rate	3.6362e-4
L2 Regularization	1.0047e-6
Loss function	Cross-Entropy

Table 3.11: Hyperparameters of LSTM network for efficiency recognition.

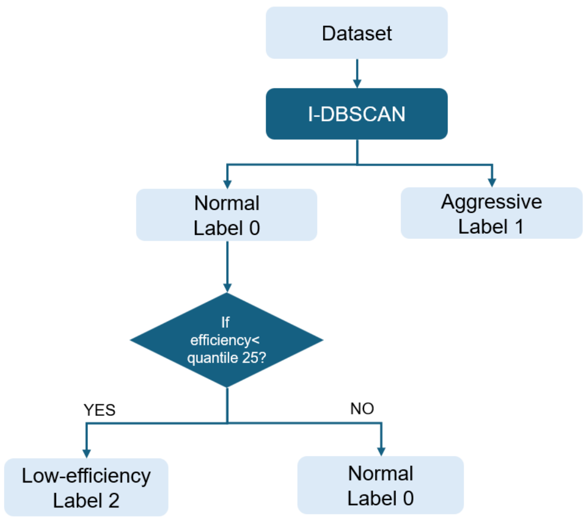


Figure 3.16: Specific labeling steps of normal, aggressive, and low efficiency.

This subsection will not repeat the definition of the confusion matrix and related indicators. For details, please refer to the previous subsection.

Model 2 showed excellent classification ability in the recognition task of three types of driving behaviors. As can be seen from the confusion matrix in Figure 3.17, the model is extremely accurate in identifying the normal and low efficiency categories. It correctly classified 99.2% of the 40,805 normal samples and 99.2% of the samples in the low efficiency category, and almost no cross-category misjudgments occurred; the corresponding F1 scores reached 0.99 (shown in Table 3.12), indicating that the precision and recall of the model in these two categories are maintained at a very high level.

The aggressive category is slightly difficult to identify. Although 94.1% of the samples are still correctly classified, about 5.9% are misclassified as other categories, mainly confused with the normal category. This result is also reflected in the F1 score of 0.95, which is slightly lower than the other two categories, but still within an acceptable range, indicating that the model has a strong recognition ability.

Compared to Model 1, Model 2 performs better in identifying abnormal driving behaviors and is more stable overall. This may be because the classification boundary between low efficiency and aggressive driving behaviors is more obvious than that of discomfort driving behaviors, so it has higher sensitivity and accuracy when facing aggressive driving behaviors.

Figure 3.18 shows that from 25 to 28 seconds, it can be seen from the longitudinal acceleration that the driver suddenly braked and then suddenly accelerated.

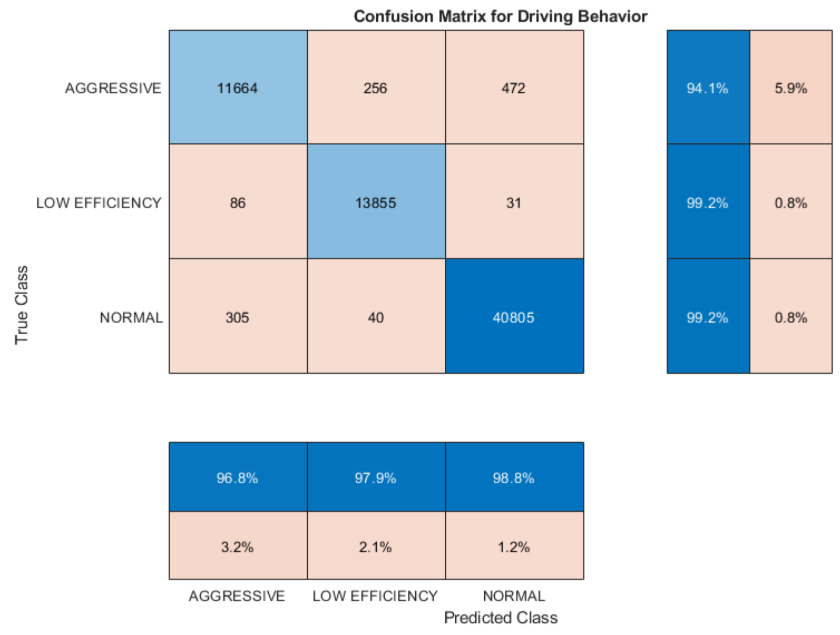


Figure 3.17: Confusion matrix for efficiency recognition. Below the matrix: precision for each label, right side of the matrix: recall for each label.

	F1 score
Aggressive	0.95
Discomfort	0.99
Normal	0.99

Table 3.12: Performance of LSTM model 2.

The longitudinal acceleration shows that the driver performed these two operations when turning, which is a typical aggressive acceleration behavior. The driving classification results also correctly classify these two behaviors as aggressive driving behaviors. From 32 to 38 seconds, the motor efficiency is low. Although driving is smooth, motor efficiency will gradually decrease due to increased load during the continuous acceleration phase until deceleration or gliding to return to the high-efficiency zone. The model also successfully identified the inefficient stage, and the output value became 2. By observing the driving data, there is no abnormality at other times, and the LSTM model classification result is also 0.

3.3.3 Model 3 (normal, aggressive, uncomfortable, low energy efficiency)

Following the classification scheme illustrated in Figure 3.19, the data set was partitioned into four categories: 43,761 samples of normal driving (labeled 0), 1,062

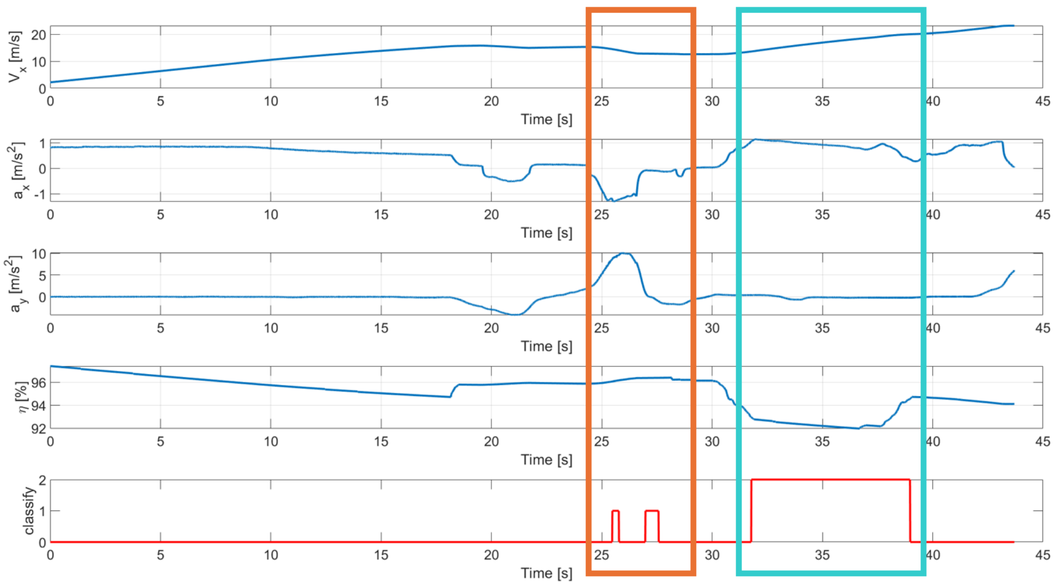


Figure 3.18: Neural network input and output. The first row: longitudinal velocity signal, the second row: longitudinal acceleration signal, the third row: lateral acceleration signal, the fourth row: average motor efficiency, the fifth row: driving behavior classification (0:normal; 1:aggressive; 2:inefficiency). The orange box highlights the aggressive driving, and the cyan box highlights the low-efficiency driving.

samples of aggressive driving (labeled 1), 351 samples that affect passenger comfort (labeled 2), and 351 samples characterized by low energy driving (labeled 3).

The hyperparameters chosen by the Bayesian optimization method are shown in Table 3.13.

Hyperparameters	**Value**
LSTM number of layers	2
Number of hidden units	128
Maximum epochs	75
Batch size	128
Dropout rate	0.3
Initial learn rate	4.6663e-4
L2 Regularization	1.0343e-4
Loss function	Cross-Entropy

Table 3.13: Hyperparameters of LSTM network for uncomfortable & efficiency recognition.

The overall accuracy of this four-class model exceeded 96%. The confusion matrix is shown in Figure 3.20. This model identified the inefficient driving category best in these four classes, with high accuracy (98.3%) and recall (98.8%). The model correctly identified 13,800 inefficient driving samples, and only 123 samples were misclassified as aggressive driving. This shows that the behavioral patterns or

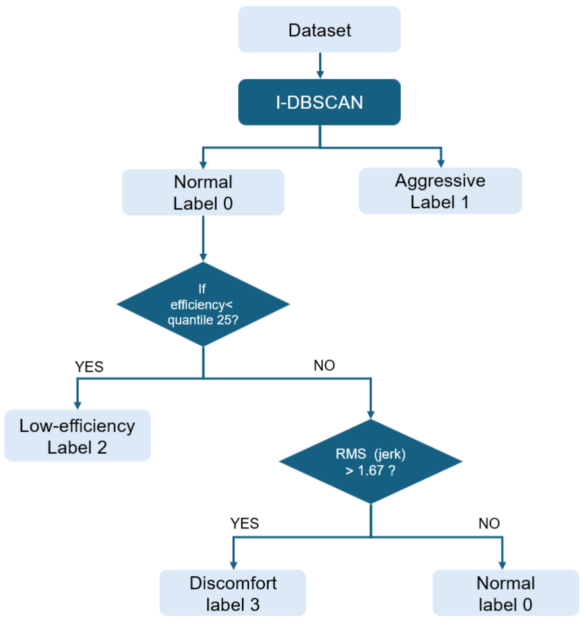
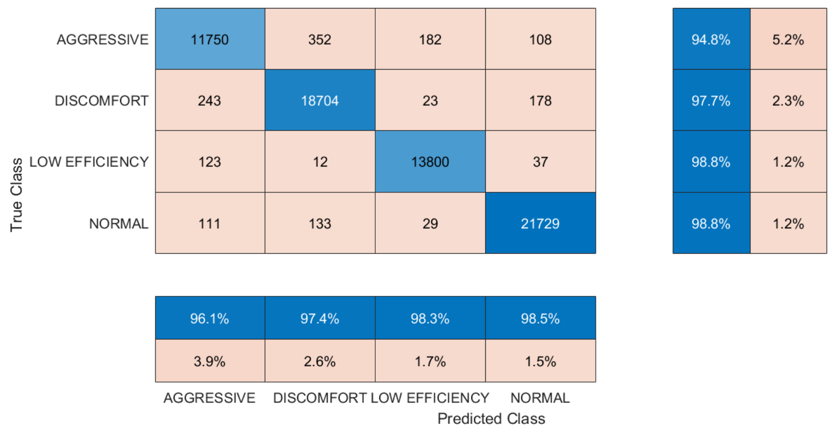


Figure 3.19: Specific labeling steps of normal, aggressive, uncomfortable, and low efficiency.

quantitative indicators of inefficient driving are highly specific, forming a very clear classification boundary in the multidimensional feature space on which the model is based.

The performance of the uncomfortable driving category is at a moderate level (accuracy 97.4%, recall 97.7%), and the model correctly classified 18,704 samples. However, its misclassification is bidirectional: 243 discomfort driving samples are misclassified as aggressive driving, while 178 samples are misclassified as normal driving. This result reflects the complexity of the characterization of uncomfortable driving, and its features overlap to a certain extent with both aggressive driving and normal driving, indicating that it may be located in the transition area of the feature space.

In contrast, the performance of the aggressive driving category is relatively poor (96.1% precision and 94.8% recall), with 11,750 correctly classified samples. There is a clear confusion between this category and the discomfort driving category, especially 352 aggressive driving samples were misclassified as discomfort driving. This asymmetric misclassification result shows that there may be an intrinsic connection or feature similarity between the aggressive driving behaviors and discomfort driving behaviors. The normal driving class performs well overall, but a small number of bidirectional misclassifications with all other classes (inefficient, uncomfortable, aggressive) still exist.



True Class	AGGRESSIVE	DISCOMFORT	LOW EFFICIENCY	NORMAL		
AGGRESSIVE	11750	352	182	108	94.8%	5.2%
DISCOMFORT	243	18704	23	178	97.7%	2.3%
LOW EFFICIENCY	123	12	13800	37	98.8%	1.2%
NORMAL	111	133	29	21729	98.8%	1.2%
	96.1%	97.4%	98.3%	98.5%		
	3.9%	2.6%	1.7%	1.5%		

Predicted Class

Figure 3.20: Confusion matrix for 4-label recognition. Below the matrix: precision for each label, right side of the matrix: recall for each label.

	F1 score
Aggressive	0.95
Discomfort	0.99
Low efficiency	0.99
Normal	0.99

Table 3.14: Performance of LSTM model 3.

The Figure 3.21 shows how the LSTM model that outputs four labels identifies different driving behaviors based on the input features. According to the jerk value, it can be found that the driver's start is very unstable, which affects the ride comfort. In addition, the driver frequently changes the longitudinal acceleration between 30 seconds and 40 seconds and between 75 seconds and 78 seconds. However, the longitudinal acceleration changes are within an acceptable range, so it only affects the ride comfort. The LSTM model also classifies it as uncomfortable driving behavior. From 51 seconds to 54 seconds, the motor efficiency is obviously low, and the model also classifies it correctly. From 91 seconds to 93 seconds, by observing the longitudinal acceleration, it can be seen that the longitudinal acceleration drops rapidly, which is an emergency braking behavior. The model classification results show that the first discomfort and then aggressive driving behavior is because the jerk value first changes suddenly, affecting the comfort.

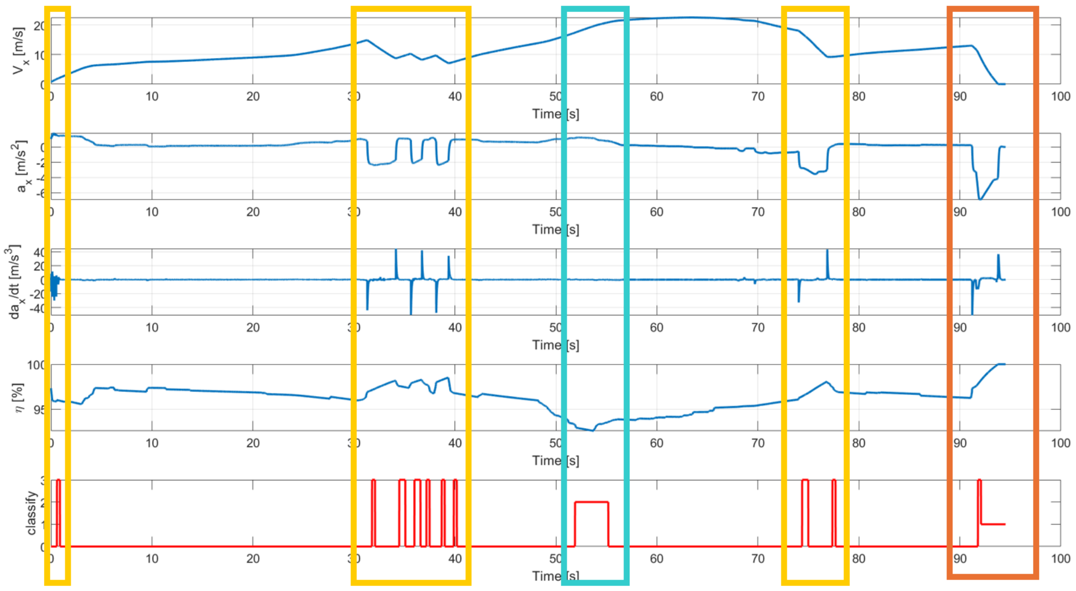


Figure 3.21: Neural network input and output. The first row: longitudinal velocity signal, the second row: longitudinal acceleration signal, the third row: jerk value, the fourth row: average motor efficiency, the fifth row: driving behavior classification (0:normal; 1:aggressive; 2:inefficiency; 3:discomfort). The orange box highlights the aggressive driving, the cyan box highlights the low-efficiency driving, and the yellow box highlights the discomfort driving.

Chapter 4

Human Machine Interface

4.1 HMI Design Objectives & Principles

In the development of electric vehicles, the HMI system is the core medium for information transmission and feedback between the driver and the vehicle. The scientificity and effectiveness of its interaction design directly affect driving safety, comfort, and user acceptance. In order to improve the practicality and user experience of the system, its core design principles need to balance safety, efficiency, comfort, and technical adaptability.

- **Safety**
  In electric vehicle HMI systems, safety refers to the system accurately and promptly conveying important driving safety information while keeping the driver focused. The system should be able to identify potential risks and communicate essential information to the driver without disrupting normal driving operations, such as abnormal vehicle status or deviations in driving behavior. At the same time, information should be communicated in a way that avoids misleading, distracting, or delayed responses. For instance, complex interfaces should not appear when driving at high speeds, and confusing prompts should be avoided. By managing information priorities reasonably and using concise, effective prompt methods, potential safety hazards caused by omissions or misjudgments can be reduced, and the active safety capabilities of the entire vehicle can be improved.

- **Intuitiveness**
  Apart from the requirement of timely information transmission, we also need to consider whether the content of the prompt is easy to understand and respond

to quickly. By using uniform and clear graphic symbols, color coding, and soft sound prompts, the driver can understand the intention of the system without deep thinking. Especially in the case of sudden or high-pressure situations, intuitive design can further reduce the cognitive load of the driver and let them focus on the driving task, and then improving the response efficiency and system effectiveness.

- **Graded Feedback**
  Different driving behaviors will bring different risk levels, so HMI should be able to adjust the method and intensity of the prompt dynamically according to risk level. For example, when the vehicle speed is slightly higher than the threshold, only a soft visual prompt is needed. If behavior of driver is in the case of a possible collision, HMI will output a stronger visual and auditory joint alarm. Graded prompt method not only can help to improve the awareness of drivers for risk, but also can reduce the phenomenon of “alarm fatigue” caused by many unnecessary or frequent alarms, and further improve the reliability and effectiveness of the system.

- **Customizability**
  Due to the differences in driving style, risk perception, and the tolerance of warning information, driver can adjust the parameters of prompt range in a certain interval according to their habit. For example, feedback sensitivity, alarm volume, obviousness of visual prompt, etc. If providing flexible configuration options, not only can it improve user experience, but also make the system more applicable in different driving conditions.

In summary, the interactive design of electric vehicle HMI should not only focus on the efficiency of information transmission, but also consider the individual differences of drivers during use, while always taking driving safety as the premise. The above design principles provide direction and basis for the functional realization and system development of subsequent HMI modules, and also lay the foundation for creating a smarter assisted driving environment that is closer to the driver’s usage habits. The following subsections will further introduce the specific design scheme of the HMI system in this study, including interface layout, feedback form, and system response logic.

4.2 HMI system Design

The HMI of this system aims to provide intuitive and real-time interaction between the driver and the vehicle, trying to ensure timely feedback on driving behavior and vehicle status.

In this part, the Python language is used, and the graphical interface is built with the PyQt5 framework. By establishing a communication connection with the vehicle simulation system, the HMI system receives vehicle dynamic data in real-time, such as speed, acceleration, and driving mode, and graphically presents this information in the interface. A multimodal feedback mechanism is used to enhance the driver's perception of potential abnormal behavior. When the system receives the driving behavior classification results (i.e., "driving mode") identified by the external model, it will combine the internal preset feedback logic to prompt the driver through visual pop-up windows, icon color changes, and prompt sound effects. At the same time, it also provides a visual parameter adjustment function to allow drivers to adjust the thresholds for judging different driving behaviors according to their habits and preferences. It is also equipped with a logging mechanism to track key events.

The HMI adopts a hierarchical and modular structure. It mainly includes four core functional modules: interface display module, communication processing module, driving behavior feedback module, and user interaction module. The modules work together through clear interfaces to form a complete closed-loop feedback system. The implementation details and design ideas of these modules will be expanded one by one.

4.2.1 Interface display module

The interface display module is the most user-facing component of the HMI system It is mainly responsible for presenting the received vehicle dynamic data on the screen in a graphical form, so that the driver can monitor driving-related information in real time. This module is not only responsible for the display and update of numerical data, but also enhances information transmission through graphic animation. It is the core of the system's interactive experience.

As shown in the Figure 4.1, the system adopts a dark background color scheme with high-contrast black large fonts to reduce visual fatigue when users observe the

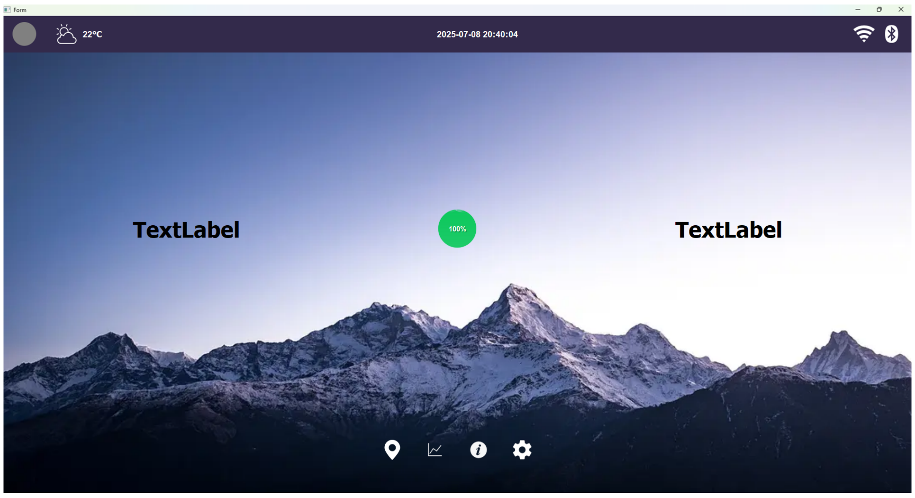


Figure 4.1: Main window interface of HMI system.

interface. In terms of interface layout, a horizontal distribution design approach is adopted, with three core pieces of information—vehicle speed, driving efficiency, and electric vehicle energy consumption—arranged side by side in the central area. The three form a visually symmetrical layout, helping drivers quickly obtain key information.

Among these, vehicle speed and energy consumption values are directly displayed in text format, clearly labeled with units, centered in position, emphasizing their importance in driving monitoring; while the motor efficiency metric uses a dynamic visualization component—the efficiency ring. The design of the efficiency ring is similar to the display of liquid level height, and the efficiency change is simulated through dynamic wave animation. Different efficiency levels correspond to different colors: high efficiency is green, low efficiency is orange, and both color and waveform height automatically adjust based on real-time data. This design provides a more intuitive visual representation of the current energy consumption status.

In addition to core data, the interface also integrates a time display and a small status indicator light (top-left corner). The status light's color changes based on driving mode, e.g., green indicates normal driving, orange indicates aggressive driving, and yellow indicates uncomfortable driving. The driver can perceive the current status without reading text prompts, which further improves the intuitiveness of information transmission.

The implementation of this module is mainly based on the layout manager and custom drawing functions in the PyQt5 framework. The interface uses automatic adaptation, with all components scaled proportionally during initialization based on the current screen resolution to ensure consistent display across different devices.

The text information area uses `QHBoxLayout` for alignment and elastic distribution to ensure that the main window interface remains orderly under different window sizes. The efficiency ring component is implemented using a custom class that inherits from `QWidget` and rewrites the `paintEvent` function to complete the drawing logic. Combined with a timer to update the animation, this achieves a continuous and smooth visual effect.

Overall, the interface display module meets the real-time display requirements for vehicle operational status while optimizing visual style and interaction design, ensuring the system has good readability and user experience. This provides a clear and intuitive foundation for the subsequent operation of the driving behavior feedback mechanism.

4.2.2 Communication processing module

The communication processing module serves as the core bridge for real-time data updates in the system, primarily responsible for data exchange with external models. This module enables bidirectional communication via the UDP protocol: on one hand, it receives driving state data sent by the simulation platform, and on the other hand, it supports feeding back parameters set by users in the interface to the driving behavior classification model.

In terms of data reception, the system runs the UDP receiver (UDPReceiver) in an independent thread to avoid network latency or data blocking affecting interface responsiveness. The receiver monitors the data stream on the specified port and parses the data upon arrival. The data packet received contains many driving parameters, like the current vehicle speed, acceleration, braking, clutch position, and the ”driving mode”, which is determined by the driving behavior classification model. The system verifies the data length and format before triggering the subsequent data update process.

The received data is unpacked according to the predefined structure. For example, each data packet contains 10 floating-point numbers corresponding to different driving status indicators. To maintain system stability, the communication module includes a receiving frequency recording function to monitor the data refresh rate and detect problems in time when the data is interrupted or fluctuates abnormally.

On the other hand, in terms of parameter transmission, the system has a built-in UDP sender (UDPSender) to package the user-set values (driving sensitivity set via sliders) into UDP data and send them to the IP and port specified by the simulation platform. The transmission operation is usually triggered by interface interaction and accompanied by log recording for easy retrospection.

The entire communication module is designed to be lightweight and stable. By using the PyQt threading mechanism to separate communication tasks from the main interface thread, user interface lag issues are effectively avoided. In addition, common problems such as network anomalies, port occupation, and data format errors are also handled within the module, enhancing the robustness of the system.

In summary, the communication processing module is the key to connecting the HMI system with the external simulation platform, ensuring the real-time nature of data input and the timeliness of parameter output. It provides the data foundation for the entire interaction process and creates the conditions for the operation of functional modules such as driving behavior feedback.

4.2.3 Driving behavior feedback module

The behavior feedback module is a functional unit in the HMI system that responds to changes in driving status and provides prompts. This module does not undertake the task of identifying driving behavior, but triggers the preset feedback logic in the system based on the driving mode number (driving mode) input from the external driving behavior classification model. It guides and alerts the driver about their current status to ensure the safety and stability of driving behavior.

Feedback Trigger Logic The system uses a floating-point number to represent the driving behavior classification results, with each number corresponding to a driving state. For instance, 0 indicates normal driving, 1 indicates aggressive driving, 2 indicates inefficient driving, and 3 indicates uncomfortable driving. Whenever the system receives a new driving mode number, it checks to see if it is different from the previous one. If it changes, the corresponding feedback logic will be triggered immediately.

Like the ”aggressive driving” mode (label 1), as shown in the Figure. When the system detects this mode number, a warning window pops up immediately on the interface. The window title is displayed as ”Aggressive Driving Detected,” and the subtitle will display the current acceleration value, such as ”The current acceleration is 2.5 m/s^2, please slow down.” Additionally, the color of the status light in the lower left corner of the interface will change from green to orange to provide a more intuitive visual warning.

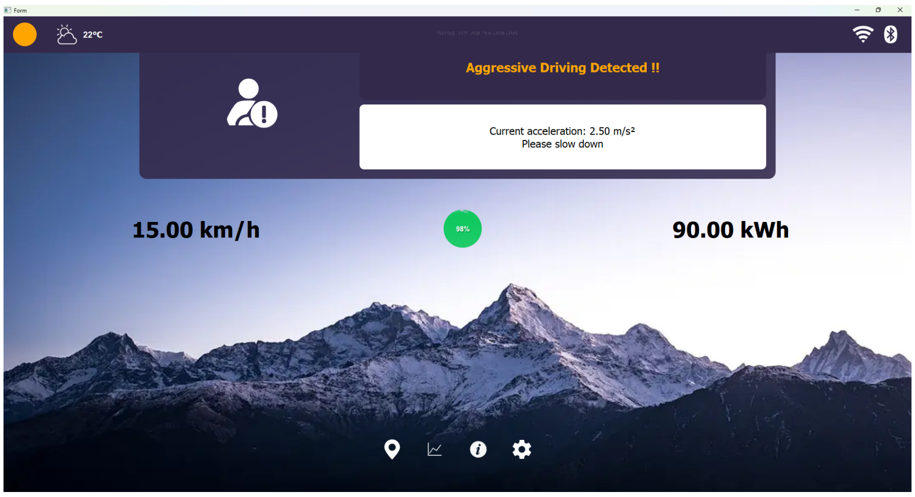
The system decides when to trigger the prompt based on the received driving mode number. Original parameters, such as acceleration value, are supplementary information that enriches the prompt content and makes it easier for drivers to understand why the prompt was triggered.

To avoid repeated prompts when driving status fluctuates frequently, the feedback module has a state locking mechanism: the system only updates the prompt content when the received mode number differs from the previous one. This effectively avoids frequent interface refreshes and improves user acceptance and operational stability.

Multimodal Feedback Design To improve the effectiveness and perceptibility of alerts, the system incorporates a multimodal feedback mechanism that combines both visual and auditory channels. This approach helps drivers notice warnings even when their attention is not focused on the screen.

Visual Feedback: The visual component includes pop-up windows, icon color changes, and a dynamic efficiency ring. When an abnormal driving mode is received, a warning window appears with a mode-specific icon, title, and description:

- **Aggressive Driving (Mode 1):** Orange window with acceleration value and "Please slow down" message. Status lamp changes to orange. (See Figure 4.2)

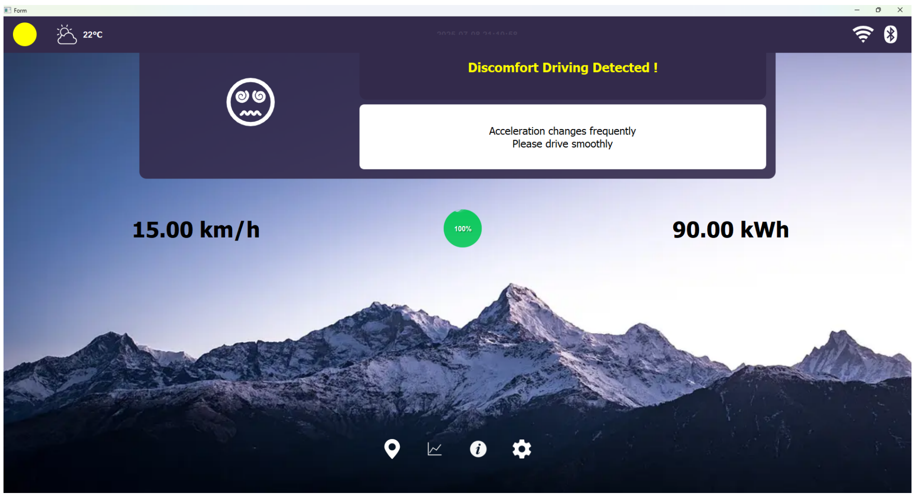

Aggressive Driving Detected !!

Current acceleration: 2.50 m/s²
Please slow down

Figure 4.2: Visual prompt of aggressive driving behavior.

- **Discomfort Driving (Mode 2):** Yellow window indicating frequent acceleration changes. The icon resembles dizziness. The status lamp turns yellow.

(See Figure 4.3)



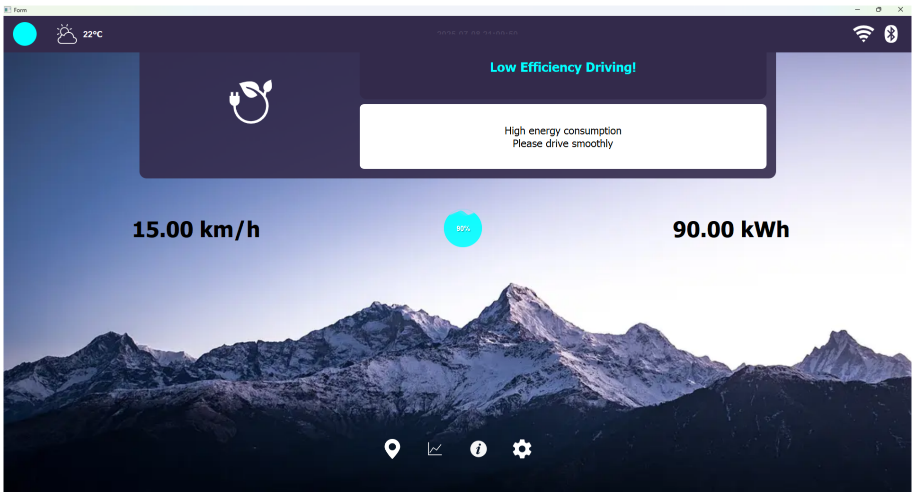
Acceleration changes frequently
Please drive smoothly

15.00 km/h

90.00 kWh

Figure 4.3: Visual prompt of discomfort driving behavior.

- **Low-Efficiency Driving (Mode 3):** Cyan-colored title with message "High energy consumption." The efficiency ring color changes from green to orange. (See Figure 4.4)



High energy consumption
Please drive smoothly

15.00 km/h

90.00 kWh

Figure 4.4: Visual prompt of low-efficiency driving behavior.

Auditory Feedback: Each driving mode also triggers a different sound pattern:

- Aggressive: Fast, high-frequency beeps (e.g., alternating 1000Hz and 1200Hz)
- Discomfort: Wavy tone pattern (e.g., 400Hz and 550Hz loop)
- Low-efficiency: Short single-beep alert (e.g., 600Hz)

These sounds are played through a dedicated audio thread to avoid blocking the main interface. Visual and auditory cues are triggered simultaneously to ensure a consistent user experience.

Coordination Strategy: The system ensures that multimodal feedback is synchronized and not overly repetitive. Warnings are shown only once per mode change. Sound alerts are not played again unless the driving mode changes, preventing unnecessary disturbance.

This combined feedback design helps drivers quickly recognize and respond to unsafe or inefficient behavior through multiple sensory channels, thus improving overall driving awareness and response time.

4.2.4 User interaction module

The user interaction module provides a direct interface between the driver and the system. While most of the HMI system functions are automated and data-driven, this module enables users to manually configure key parameters and view supplementary information, allowing for personalized control and enhanced system flexibility.

Sensitivity Value Configuration

The core function of this module is the setting of the **sensitivity value**, which can be adjusted by the user through a graphical slider interface. The sensitivity value directly affects the judgment result of the driving behavior classification model, and therefore plays a key role in the system's operation logic.

The driving behavior classification model is a structure based on the LSTM. Its output result is not a fixed label, but a probability distribution of a set of behavior categories, which respectively indicate the possibility that the current state belongs to ”aggressive driving”, ”inefficient driving”, or ”uncomfortable driving”. Since the model output is a continuous probability value, in order to convert it into a specific driving mode, a threshold needs to be set - this is the sensitivity value set by the user through the interactive interface. This value is sent via UDP from the HMI interface to the classification model module, where it serves as a decision boundary for interpreting the probability scores.

Decision Priority Logic: Once the model receives the user-defined threshold, it applies a priority-based decision logic:

1. If the probability of aggressive driving exceeds the sensitivity value, the system classifies the current mode as **aggressive**.
2. If not, but the probability of low-efficiency driving exceeds the threshold, the mode is set to **low-efficiency**.
3. If neither of the above applies, but discomfort driving exceeds the threshold, the mode becomes **discomfort**.
4. If none of the probabilities exceeds the sensitivity value, the system defaults to **normal driving**.
5. If the driver does not adjust this value and the driving classification model does not receive any updates about the sensitivity value, it will automatically set this value to 0.8.

This priority logic ensures that when the probability of multiple behaviors is close or exists at the same time, the system gives priority to the most serious driving behavior, improving the accuracy and pertinence of feedback.

By adjusting the sensitivity value, driver can control the ”strictness” of behavior recognition according to actual needs: the lower the sensitivity value, the easier it is for the system to determine that the driving behavior is abnormal; conversely, the higher the value, the more ”tolerant” the system is, and prompts will only be triggered in very obvious abnormal situations. This setting is particularly important in experimental environments. This can be particularly useful in experimental settings, where different test conditions may call for different classification sensitivities.

Interaction Interface Design

From a user interface perspective, the interaction module includes:

- A **slider dialog** for sensitivity adjustment, with real-time display of the current value;
- A **control button** to open the configuration dialog at any time during system operation;

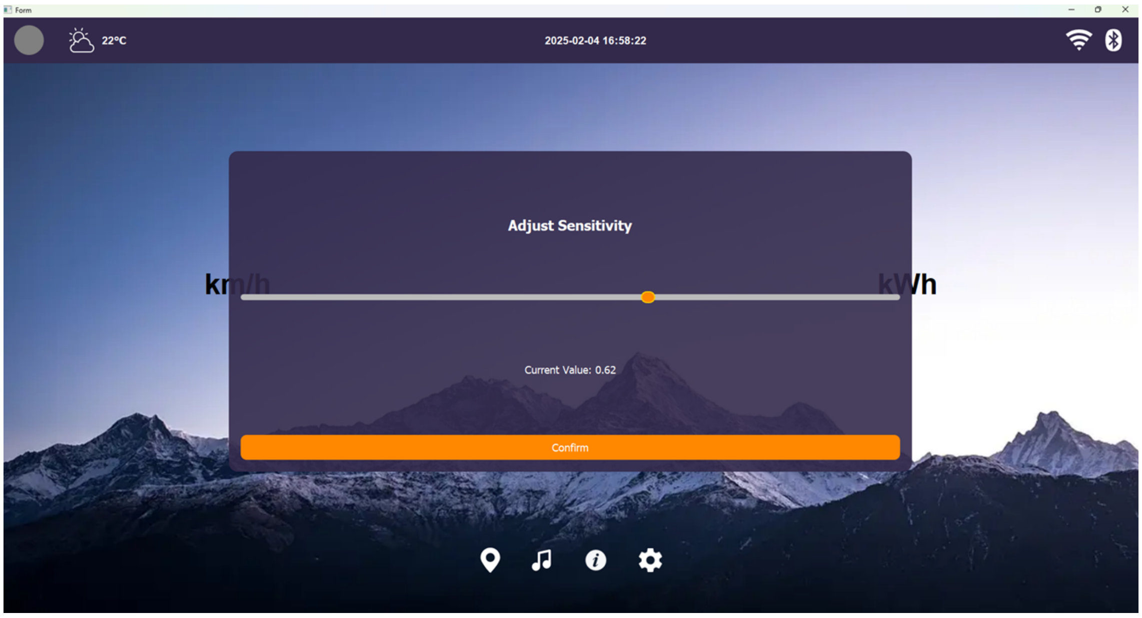


Figure 4.5: Slider dialog for sensitivity adjustment.

- A secondary window to display extended driving information, including detailed values for steering angle, brake pressure, clutch usage, throttle position, and handbrake status.

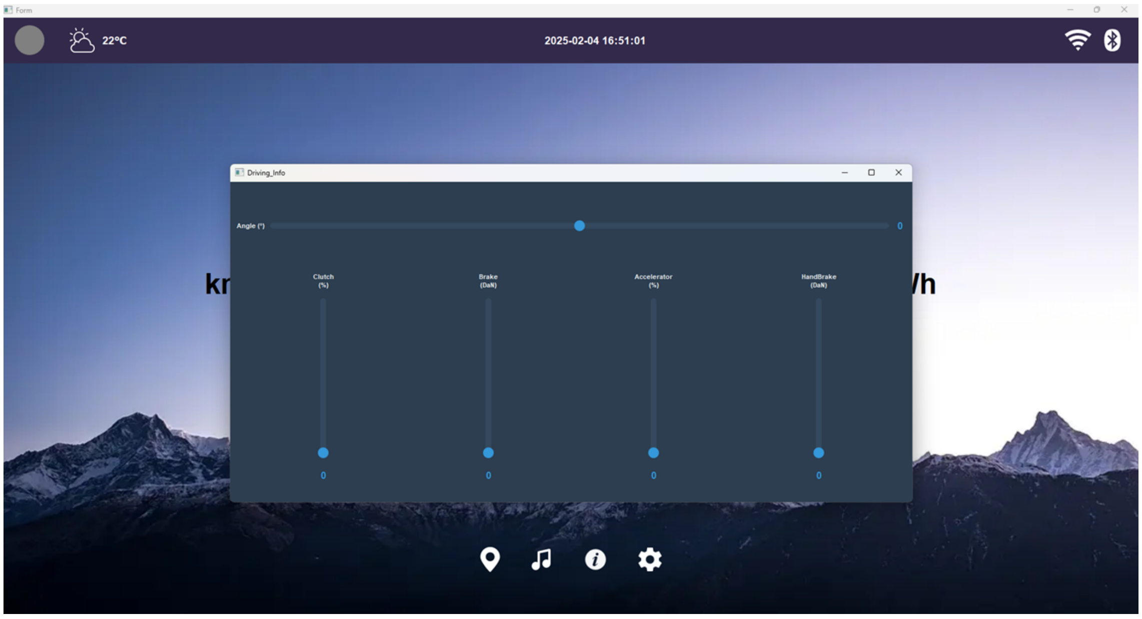


Figure 4.6: The secondary window about driving information.

All settings made by the user are logged and can be modified in real time without interrupting the main data stream or visual output. Once the sensitivity value is confirmed, it is immediately sent to the external classifier using the system's UDP sender thread.

Design Considerations

The sensitivity configuration feature is particularly important in experimental settings. It allows researchers to dynamically adjust the classification strictness without modifying the model itself. This enables controlled experiments to compare driver behavior under different feedback conditions, or to evaluate driver reactions to different levels of system tolerance.

Furthermore, by placing this control in the user's hands, the system gains an additional layer of flexibility. Instead of relying on hard-coded thresholds, the HMI provides a way to adjust classification behavior based on the driving scenario, driver type, or research goal. This design supports both autonomous deployment and experimental control use cases.

Chapter 5

System Integration and Performance Evaluation

5.1 System Integration Overview

After the system design was complete and each module had been tested individually, the software and hardware components of the system were fully deployed and integrated to create a real-time driving behavior recognition and feedback platform with closed-loop control capabilities. Unlike the modular description of the system architecture in Chapter 2, this section focuses on the coordination and stability of each component within the operating environment.

The system consists of a driving simulator, SCANeR Studio, Simulink, a Speedgoat real-time control platform, a Raspberry Pi, and a human-computer interaction interface. These devices are connected via Ethernet and distributed within the same local area network. They use fixed IP addresses to ensure interoperability. While the system is running, SCANeR™ Studio collects driving data continuously at a sampling frequency of 100 Hz and transmits the data to Simulink. Extracted features are transmitted to Speedgoat via UDP for real-time processing and LSTM behavior recognition model execution. The recognition results and driving data are transmitted to the Raspberry Pi in real time via UDP to complete the final interface display and sound feedback. At the same time, the Raspberry Pi transmits the model sensitivity values selected by the user to Speedgoat via UDP.

Note that although the system's overall sampling frequency is 100 Hz and Speedgoat's real-time step size is 1 ms, the classification result's actual output cycle is approximately once every 0.3 seconds. This is because the driving behavior recognition model uses a 0.3-second sliding window structure for feature modeling and reasoning.

The system's overall integration is visualized in Figure 5.1, which illustrates the physical and data connections between modules. Key data flows, including feature extraction, behavior classification, dynamic signal transfer, and feedback response, are clearly represented, showing the bidirectional communication between Raspberry Pi and Speedgoat, and the HDMI-based display output. For more details, see Section 2.3 and Section 2.4

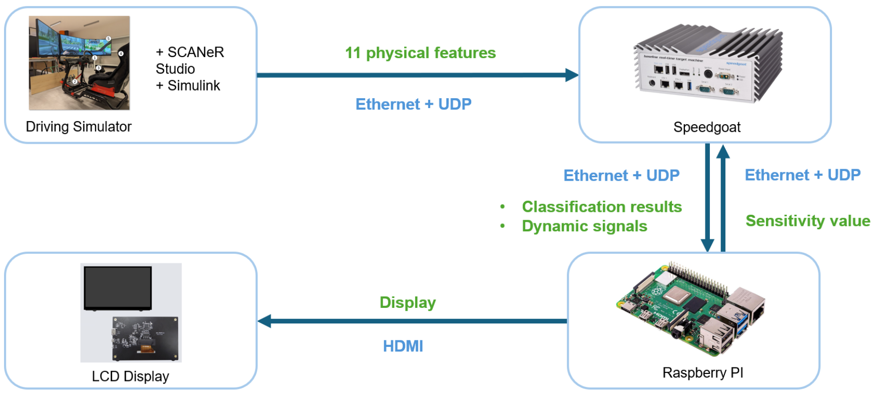


Figure 5.1: System overall architecture: green texts represent the types of data being transmitted between modules, while blue text indicates the physical connections and communication protocols.

During the integrated joint debugging process, we tested the stability of the system communication link and module response in combination with real-time data display, log recording, and multiple rounds of simulation. The test results show that:

- The SCANeR Studio end can stably output complete vehicle dynamic signals.
- Data transmission between Simulink and Speedgoat is correct, and the behavior mode can be output at the set frequency.
- The UDP communication between Speedgoat and Raspberry Pi is stable and reliable with minimal packet loss.
- The HMI can promptly receive the behavior status and trigger the corresponding feedback. The visual and auditory prompts are triggered synchronously without jamming or delay accumulation.

Overall, the system demonstrates good stability and coordination under various driving conditions, including normal, intense, uncomfortable, and inefficient.

Data transmission between modules is continuous, timing is synchronized, and feedback triggering is timely. These features meet the basic requirements of end-to-end closed-loop control. The next section will conduct a quantitative evaluation of the system's real-time performance from multiple aspects, including key indicators such as communication delay, inference delay, and feedback response, based on this integration result.

5.2 System Real-Time Performance Evaluation

5.2.1 Evaluation metrics and methodology

In order to verify whether the system has stable and efficient response capabilities in actual operation, this subsection evaluates the performance of Model 3 from the following aspects:

- **Real-time performance**: whether the system can complete recognition and trigger feedback in time after the driving behavior occurs.
- **Processing stability**: whether the recognition model outputs classification results at a stable frequency.
- **Communication reliability**: whether the data transmission between modules is continuous and stable, and whether there is packet loss or delay mutation.
- **Feedback integrity and synchronization**: whether the HMI can accurately and timely output visual or sound prompts after receiving the recognition results, and whether different feedback methods are synchronized.

Around the above goals, the system designs three types of evaluation indicators: response delay, communication and feedback stability, and behavior recognition coverage. Combined with system log records and video recording analysis, the verification method of each indicator is explained as follows.

(1) **Response Delay Metrics**
This part focuses on the time cost of each stage from the occurrence of behavior to the feedback output of the system, including the following three categories:

Model output frequency: In theory, the driving behavior recognition model outputs a classification result every 0.3 seconds. The model output frequency is indirectly calculated by recording the data reception time interval on the Raspberry Pi side, which is used to determine whether the system is running stably.

Interface processing time: After each behavior result is received by the HMI, the graphical interface needs to be updated, and the sound is played. The system records the total processing time from the receipt of data to the completion of rendering of all feedback elements, as well as the processing time of each submodule. This can reflect the response speed of the system at the human-machine feedback end.

Overall response time estimation: Due to the inconsistency of the system time of each device, it is impossible to directly measure the actual delay from recognition output to feedback presentation. Therefore, the test uses video recording to record driving operations and interface feedback, and estimates the time difference from the occurrence of driving behavior to visual/auditory feedback through frame-to-frame comparison, thereby indirectly evaluating the overall response capability of the system.

(2) **Communication and Feedback Reliability Metrics**

The following indicators are chosen to verify the stability and reliability of the system during operation:

UDP receiving frequency fluctuation: The data receiving frequency is continuously recorded to observe whether there are similar unstable situations, such as reception interruption, sudden increase in delay. If so, it may indicate packet loss or unstable communication.

Feedback consistency and integrity: By comparing classification results with actual feedback behavior, verify that each behavior recognition triggers the corresponding visual or auditory prompt. If recognition is successful, but no feedback is provided, the feedback is considered invalid. The log records the current recognition mode number, and the video helps observe if the feedback is timely and accurate.

(3) **Behavior Recognition Coverage Metrics**

This indicator mainly evaluates whether the system can identify all driving behavior types in the design target during actual driving. The model defines a variety of driving modes, including normal, intense, uncomfortable, inefficient, etc. The following should be observed during the test:

- Whether each type of behavior is triggered at least once.

- Whether each type of mode can be accurately fed back after being identified.
- Whether the system can continue to respond to frequent or repeated behaviors without omissions or freezes.

In general, each indicator is based on the log records in the system and cross-validated with external video observation. This method can quantitatively evaluate the processing time of each stage, and can also evaluate the overall response effect of the system through intuitive feedback, ensuring the integrity and persuasiveness of the verification results.

5.2.2 Experimental results and analysis

In order to verify the performance of the system in actual operation, free driving was selected as the test scenario. The behavior recognition frequency and interface processing time were recorded through the Raspberry Pi log. At the same time, the driving operation and human-machine interface response were recorded by video to assist in analyzing the overall response delay of the system from the occurrence of behavior to the presentation of feedback.

(1) **Model output and data receiving frequency**
The system identification model is designed to output the classification result once every 0.3 seconds, and the theoretical receiving frequency should be 3.3 Hz. The time interval of each data reception is recorded by the Raspberry Pi log, and the frequency is calculated to obtain the actual output of the system during free driving. As shown in the Figure 5.2, the data receiving frequency is generally stable, with an average of 3.333 Hz, a maximum of 3.410 Hz, and a minimum of 3.260 Hz. There is no obvious interruption or abnormal fluctuation, indicating that the model output is stable and the UDP communication link is reliable.

(2) **Interface feedback processing time**
Whenever the recognition result reaches the Raspberry Pi, the system needs to complete feedback operations such as interface information update and sound effect playback. The log records the total processing time from data reception to feedback completion. As shown in the Figure 5.3, the average UI processing time is 1.21 ms, the minimum is 0.58 ms, and the maximum is 6.65 ms, which is a one-time burst peak. Except for a few abnormal frames, the processing time

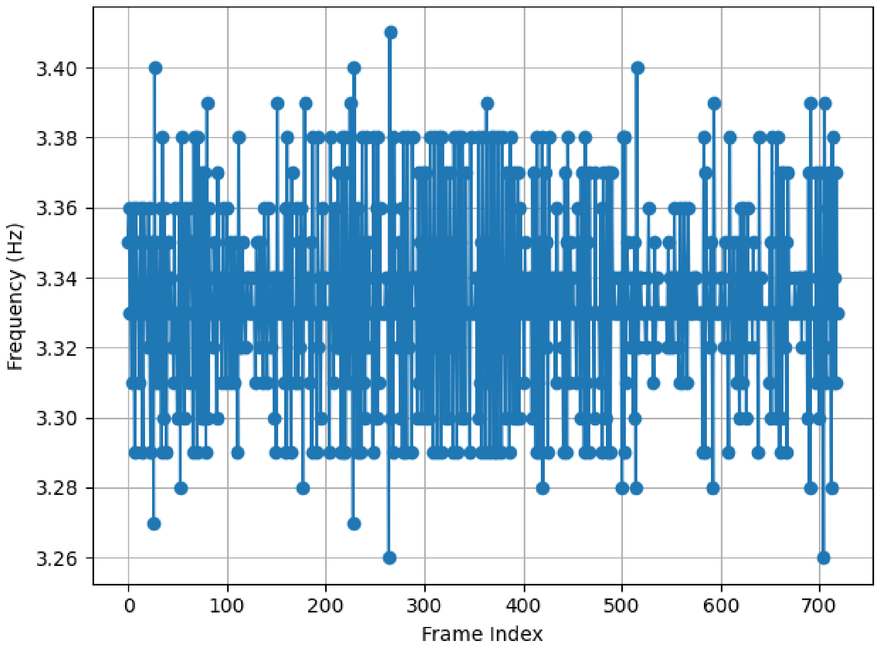


Figure 5.2: Receiving frequency during free driving.

of most frames remains stable with a small fluctuation range, indicating that the HMI module responds quickly from the behavior recognition result to the feedback trigger, without obvious delay accumulation or processing jamming.

(3) **Feedback consistency and completeness**

By comparing the driving mode change records in the log and the interface feedback in the video, it was found that the system accurately triggered the visual pop-up window and sound prompt each time it recognized the mode change, and there was no situation of no response or wrong feedback after recognition. In most cases, the visual and auditory feedback can be presented synchronously without obvious lag or delay accumulation, indicating that the system feedback link is complete and reliable.

(4) **Behavior recognition coverage**

The system defines four types of driving behavior patterns: normal, aggressive, uncomfortable, and inefficient. During free driving, the four types of behaviors were naturally triggered at least once and recorded in the system log. As shown in the Figure 5.4, the number of recognitions corresponding to each type of behavior is 54, 31, 18, 30, and all successfully triggered feedback, indicating that the model has good behavior coverage and the system can stably identify different types of driving behaviors.

(5) **Overall response capability estimation**

Since the system time of each device in the system is not synchronized, it is

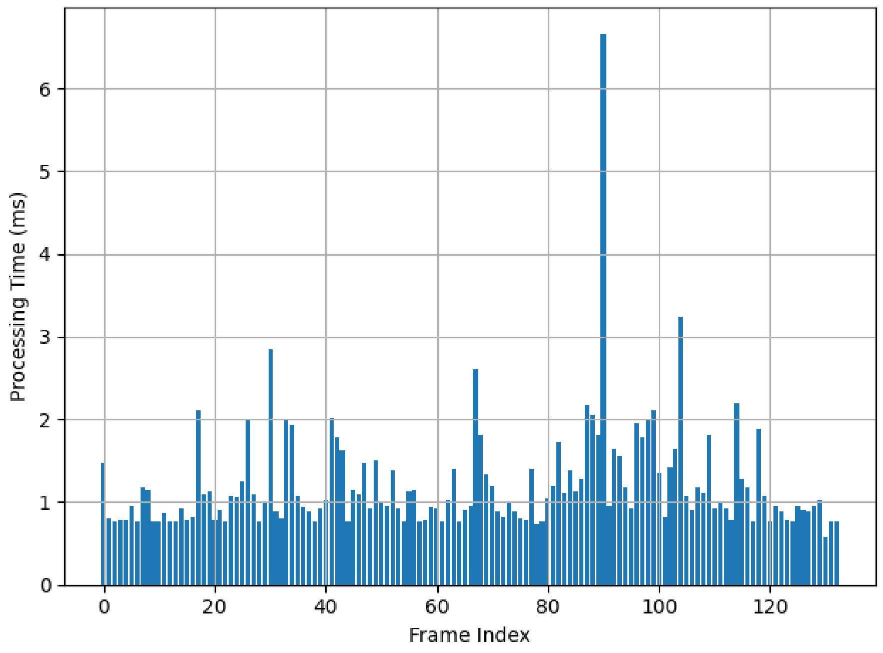


Figure 5.3: Total UI processing time per frame.

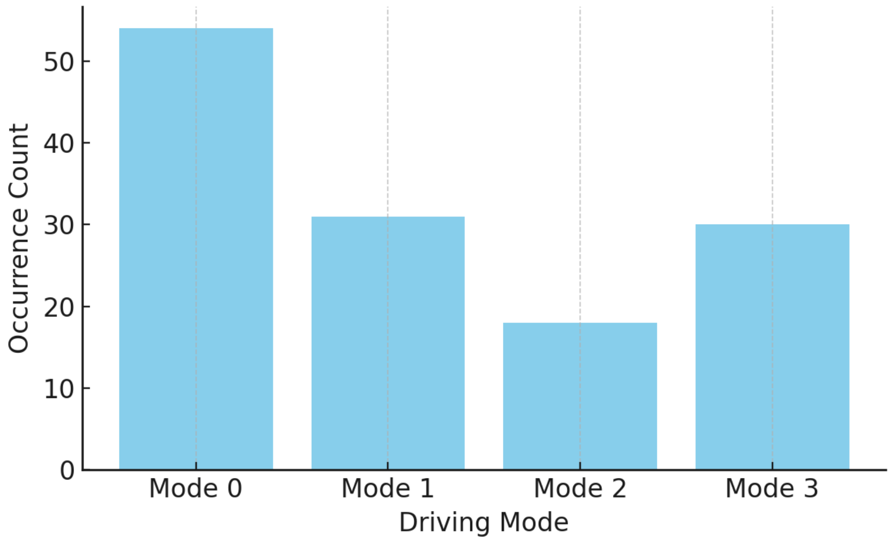


Figure 5.4: Behavior recognition coverage.

impossible to calculate the exact delay from recognition to feedback directly. Therefore, the end-to-end response time of the system is roughly estimated by comparing the time when the driving behavior occurs and the time when the feedback is triggered through video playback. The test sample indicates that the average response time from the occurrence of the behavior to the interface prompt is approximately 160 ms, which falls within the near real-time range and meets the feedback requirements in the human-machine interaction scenario.

In summary, the system exhibits good model stability, communication reliability, and feedback consistency in a free driving environment, can achieve continuous monitoring and timely feedback of driving behavior, and meets the basic performance requirements of an interactive closed-loop control system.

5.3 HMI Feedback Effectiveness Experiment

This subsection aims to verify whether the HMI system can actually guide the driver's behavior after the behavior recognition is completed and the prompt is triggered, so as to promote driving operation to be stable or energy-saving. Different from the system response and stability verification in the previous part, this experiment focuses more on observing whether the driver responds positively to the feedback and performing trend analysis through quantitative indicators.

5.3.1 Experimental design and triggering events

The experiment was designed with two control structures:

- **Experimental group (HMI ON)**: After the system identified abnormal behavior, it immediately gave prompts and feedback through the interface and sound.
- **Baseline (HMI OFF)**: The system also ran the behavior recognition model, but did not provide any feedback to the driver.

To ensure the fairness of the comparison and the repeatability of the experiment, the experimental scene, route, and traffic environment were kept consistent, and both groups performed driving tasks on the same road network. By setting trigger events at key locations, three types of driving behaviors were naturally triggered:

- **Aggressive driving (mode 1)**: Inducing high acceleration behavior by encouraging rapid overtaking and lane changes;
- **Inefficient driving (mode 2)**: Inducing deep accelerator pedaling on smooth roads, causing motor efficiency to decrease.
- **Uncomfortable driving (mode 3)**: Inducing frequent jerks by setting the front vehicle to accelerate and decelerate intermittently;

5.3.2 Data collection and analysis

The system records the driving mode number in each recognition result through logs and aligns the vehicle dynamic data with the timestamp. After each target behavior is recognized, a fixed-length analysis window is selected to count the key driving indicators after the behavior occurs.

The analysis window and Key Performance Indicator (KPI) settings for each behavior type are as follows (Table 5.1):

Driving behavior	**Window**	**KPI**
Aggressive (mode 1)	3 s	Average longitudinal acceleration
Low-efficiency (mode 2)	4 s	Average motor efficiency
Discomfort (mode 2)	2 s	Jerk exceedance count

Table 5.1: Analysis window and evaluation index for abnormal driving behavior.

5.3.3 Results and conclusion

As can be seen from Table 5.2, among the three types of driving behaviors, the HMI prompt group showed more positive changes in key driving indicators compared with the no prompt group.

In the scenario of intense driving (mode 1), after the driver received the HMI prompt, the average longitudinal acceleration dropped from $+0.62$ m/s^2 to -0.18 m/s^2, indicating that after identifying the intense driving, the driver tended to actively slow down and adjust the driving operation in time.

In the case of uncomfortable driving (mode 3), the number of jerk value exceeding the threshold dropped from 7 times in the HMI OFF group to 2 times in the baseline group, the driving process was smoother, the sudden changes of vehicle acceleration and deceleration were reduced, and the ride comfort was improved.

For inefficient driving (mode 2), HMI prompts also showed a guiding role. The average motor efficiency increased from 89.6% to 95.3%, indicating that the driver may actively reduce unnecessary throttle operations after the prompt, thereby improving energy efficiency.

Driving behaviors	**group**	**KPI**
Aggressive(mode 1)	HMI OFF	$+0.62$ m/s^2(Average longitudinal acceleration)
	HMI ON	-0.18 m/s^2(Average longitudinal acceleration)
Discomfort (mode 3)	HMI OFF	7 times (Jerk exceedance count)
	HMI ON	2 times (Jerk exceedance count)
Low-efficiency (mode 2)	HMI OFF	89.6%(Average motor efficiency)
	HMI ON	95.3%(Average motor efficiency)

Table 5.2: The KPI results under different feedback conditions (ON/OFF HMI).

Overall, this experiment preliminarily verified the positive guiding effect of the HMI prompt system based on behavior recognition on driving behavior. Without interfering with driving control, the system can guide the driver to adjust the operation in time when aggressive, uncomfortable, or inefficient driving occurs through simple and intuitive prompt information. Although the prompt itself is not mandatory, the experimental results show that drivers generally respond according to the prompt and show smoother or more energy-efficient driving behavior. This provides support for the subsequent promotion and application of the system in actual traffic environments.

Chapter 6

Conclusion

This thesis proposes a real-time driver behavior monitoring and feedback framework to improve driving safety, comfort, and energy efficiency, especially in the field of electric vehicles. The system combines data-driven behavior recognition with a responsive human-machine interface (HMI) to form a closed-loop control solution that can support drivers in real time.

The main contributions include the development of a high-performance LSTM-based classification model that can accurately identify four driving behaviors: normal, aggressive, uncomfortable, and inefficient, with an F1 score of more than 0.95. The iterative clustering and labeling method based on I-DBSCAN significantly reduces the reliance on manual annotations, enabling effective behavior recognition from unlabeled simulated data. The behavior classification results are effectively integrated into an intuitive HMI, which uses multimodal feedback (visual and auditory) to timely alert the driver, while introducing a unified ”sensitivity threshold” mechanism to support users to adjust the sensitivity of system feedback according to their personal habits.

The framework is deployed in a hardware-in-the-loop test architecture consisting of a driving simulator, Speedgoat real-time platform, and Raspberry Pi. Low-latency communication between modules is achieved through the UDP protocol, ensuring the real-time and stability of the system’s closed-loop response. In addition, free driving tests and simple control experiments with and without feedback were used to further verify the basic real-time closed-loop operation capability of the framework and the positive guidance role of the designed HMI mechanism under different driving behaviors.

Subsequent work may consider applying the system to actual road tests to further verify its performance and adaptability in real scenarios.

Acknowledgment

The graduation finally arrived. Over the past few years at Politecnico di Torino, from undergraduate to graduate studies, I've had the chance to learn from many professors whose courses and teaching have shaped the way I think and approach problems. I'm grateful for everything they've taught me; these lessons have been the foundation of my academic journey. I'd also like to thank my classmates at PoliTo. Whether it was preparing for exams, working in difficult teamwork, or just sharing notes and tips, they were always willing to help. Everyone was so warm and supportive, and I feel fortunate to have studied among such kind people.

I'm especially grateful to my thesis supervisors, Prof. Angelo Bonfitto and Prof. Shailesh Sudhakara Hegde, for their patience, encouragement, and insightful guidance throughout the thesis process. It was a great honor to have the opportunity to work under their supervision.

My final gratitude goes to those who know I'm not perfect but love me anyway. To my parents, Li Lin and Liu Yunhua, who have always supported me with love and trust. And to my friends, Deng Qiyang, Wan Qiushan, and Liu Yuxin, whose companionship made the journey not only more interesting but also more meaningful.

My academic journey has taught me more than just knowledge; it has helped me grow, both personally and academically. To everyone who has walked with me along the way, thank you.

References

[1] World Health Organization. *Global Status Report on Road Safety 2023: Summary.* World Health Organization, Geneva, 2023. Accessed: 2025-06-11.

[2] AAA Foundation for Traffic Safety. 2017 traffic safety culture index. Technical report, AAA Foundation for Traffic Safety, Washington, D.C., March 2018. Accessed: 2025-06-11.

[3] National Highway Traffic Safety Administration. Aggressive driving enforcement: Strategies for implementing best practices. Technical Report DOT HS 809 707, U.S. Department of Transportation, 2003. Accessed: 2025-06-11.

[4] Occupational Safety and Health Administration. Aggressive driving, n.d. Accessed: 2025-06-11.

[5] Adriana Skuza, Emilia M. Szumska, Rafał Jurecki, and Artur Pawelec. Modeling the impact of traffic parameters on electric vehicle energy consumption. *Energies*, 17(21):5423, 2024. Published: 30 October 2024.

[6] Adriana Skuza, Rafał Jurecki, and Emilia Szumska. Influence of traffic conditions on the energy consumption of an electric vehicle. *Communications – Scientific Letters of the University of Žilina*, 25(1):B22–B33, 2023. Open access under CC BY 4.0.

[7] Yifan Yang, Berna Karakaya, Gian Carlo Dominioni, Kyosuke Kawabe, and Klaus Bengler. An hmi concept to improve driver's visual behavior and situation awareness in automated vehicle. In *Proceedings of the 2018 IEEE 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 650–655. IEEE, 2018.

[8] Lia Morra, Fabrizio Lamberti, F. Gabriele Pratticó, Salvatore La Rosa, and Paolo Montuschi. Building trust in autonomous vehicles: Role of virtual reality driving simulators in hmi design. *IEEE Transactions on Vehicular Technology*, 68(10):9438–9450, 2019.

[9] Ondrej Linda and Milos Manic. Improving vehicle fleet fuel economy via learning fuel-efficient driving behaviors. In *2012 5th International Conference on Human System Interactions*, pages 137–143. IEEE, 2012.

[10] Tatsuaki Osafune, Toshimitsu Takahashi, Noboru Kiyama, Tsuneo Sobue, Hirozumi Yamaguchi, and Teruo Higashino. Analysis of accident risks from driv-

ing behaviors. *International journal of intelligent transportation systems research*, 15:192–202, 2017.

[11] Yang Zheng, Amardeep Sathyanarayana, and John HL Hansen. Threshold based decision-tree for automatic driving maneuver recognition using can-bus signal. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2834–2839. IEEE, 2014.

[12] Fred Feng, Shan Bao, James R Sayer, Carol Flannagan, Michael Manser, and Robert Wunderlich. Can vehicle longitudinal jerk be used to identify aggressive drivers? an examination using naturalistic driving data. *Accident Analysis & Prevention*, 104:125–136, 2017.

[13] Laura Eboli, Gabriella Mazzulla, and Giuseppe Pungillo. Combining speed and acceleration to define car users' safe or unsafe driving behaviour. *Transportation research part C: emerging technologies*, 68:113–125, 2016.

[14] Derick A Johnson and Mohan M Trivedi. Driving style recognition using a smartphone as a sensor platform. In *2011 14th International IEEE conference on intelligent transportation systems (ITSC)*, pages 1609–1615. Ieee, 2011.

[15] Omurcan Kumtepe, Gozde Bozdagi Akar, and Enes Yuncu. Driver aggressiveness detection via multisensory data fusion. *EURASIP Journal on Image and Video Processing*, 2016:1–16, 2016.

[16] Arash Jahangiri, Vincent J Berardi, and Sahar Ghanipoor Machiani. Application of real field connected vehicle data for aggressive driving identification on horizontal curves. *IEEE Transactions on Intelligent Transportation Systems*, 19(7):2316–2324, 2017.

[17] Mucahit Karaduman and Haluk Eren. Deep learning based traffic direction sign detection and determining driving style. In *2017 International Conference on Computer Science and Engineering (UBMK)*, pages 1046–1050. IEEE, 2017.

[18] Youness Moukafih, Hakim Hafidi, and Mounir Ghogho. Aggressive driving detection using deep learning-based time series classification. In *2019 IEEE international symposium on INnovations in intelligent SysTems and applications (INISTA)*, pages 1–5. IEEE, 2019.

[19] D Deva Hema and K Ashok Kumar. Hyperparameter optimization of lstm based driver's aggressive behavior prediction model. In *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, pages 752–757. IEEE, 2021.

[20] Yongfeng Ma, Ziyu Zhang, Shuyan Chen, Yanan Yu, and Kun Tang. A comparative study of aggressive driving behavior recognition algorithms based on vehicle motion data. *IEEE Access*, 7:8028–8038, 2018.

[21] Wenshuo Wang, Junqiang Xi, Alexandre Chong, and Lin Li. Driving style classification using a semisupervised support vector machine. *IEEE Transactions on Human-Machine Systems*, 47(5):650–660, 2017.

[22] Ljubo Mercep, Gernot Spiegelberg, and Alois Knoll. A case study on implementing future human-machine interfaces. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 1077–1082. IEEE, 2013.

[23] Robin Schubert, Eric Richter, Norman Mattern, Philipp Lindner, and Gerd Wanielik. A concept vehicle for rapid prototyping of advanced driver assistance systems. In *Advanced microsystems for automotive applications 2010: smart systems for green cars and safe mobility*, pages 211–219. Springer, 2011.

[24] Atiyeh Vaezipour, Andry Rakotonirainy, Narelle Haworth, and Patricia Delhomme. A simulator evaluation of in-vehicle human machine interfaces for eco-safe driving. *Transportation Research Part A: Policy and Practice*, 118:696–713, 2018.

[25] Mahmoud Zaki Iskandarani. Relating driver behaviour and response to messages through hmi in autonomous and connected vehicular environment. *Cogent Engineering*, 9(1):2002793, 2022.

[26] Klaus Bengler, Michael Rettenmaier, Nicole Fritz, and Alexander Feierle. From hmi to hmis: Towards an hmi framework for automated driving. *Information*, 11(2):61, 2020.

[27] Luca Davoli, Marco Martalò, Antonio Cilfone, Laura Belli, Gianluigi Ferrari, Roberta Presta, Roberto Montanari, Maura Mengoni, Luca Giraldi, Elvio G Amparore, et al. On driver behavior recognition for increased safety: a roadmap. *Safety*, 6(4):55, 2020.

[28] Yikai Wang, Serge Debernard, Jean-Christophe Popieul, Philippe Simon, and Jérôme Floris. Empowering adas with driver-supervised learning of preferences: Parameterization and human-machine interaction. In *IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6. IEEE, 2023.

[29] Ann-Kathrin Kraft, Christian Maag, and Martin Baumann. How to support cooperative driving by hmi design? *Transportation research interdisciplinary perspectives*, 3:100064, 2019.

[30] Rui Fu, Wenxiao Liu, Hailun Zhang, Xue Liu, and Wei Yuan. Adopting an hmi for overtaking assistance-impact of distance display, advice, and guidance information on driver gaze and performance. *Accident Analysis & Prevention*, 191:107204, 2023.

[31] Christian Maag, Ann-Kathrin Kraft, Alexandra Neukum, and Martin Baumann. Supporting cooperative driving behaviour by technology–hmi solution, acceptance by drivers and effects on workload and driving behaviour. *Transportation research part F: traffic psychology and behaviour*, 84:139–154, 2022.

[32] Mauricio Marcano, Andrea Castellano, Sergio Díaz, Joshué Pérez, Fabio Tango, Elisa Landini, and Paolo Burgio. Shared and traded control for human-automation interaction: A haptic steering controller and a visual interface. *Human-Intelligent Systems Integration*, 3:25–35, 2021.

[33] Jiao Wang and Dirk Söffker. Improving driving efficiency for hybrid electric vehicle with suitable interface. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 000928–000933. IEEE, 2016.

[34] Wenjing Zhao, Siyuan Gong, Dezong Zhao, Fenglin Liu, NN Sze, and Helai Huang. Effects of collision warning characteristics on driving behaviors and safety in connected vehicle environments. *Accident Analysis & Prevention*, 186:107053, 2023.

[35] AVSimulation. *SCANeR™studio's documentation.* AVSimulation, n.d. `https://www.avsimulation.com`.

[36] Charles Marks, Arash Jahangiri, and Sahar Ghanipoor Machiani. Iterative db-scan (i-dbscan) to identify aggressive driving behaviors within unlabeled real-world driving data. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2324–2329. IEEE, 2019.

[37] Alessandro Mauro Danusso. *Aggressive driving detection through Iterative DBSCAN labeling and supervised pattern recognition.* PhD thesis, Politecnico di Torino, 2022.

[38] Arash Jahangiri, Sahar Ghanipoor Machiani, and Vahid Balali. Big data exploration to examine aggressive driving behavior in the era of smart cities. In *Data Analytics For Smart Cities*, pages 163–182. Auerbach Publications, 2018.

[39] Kristina P Sinaga and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE access*, 8:80716–80727, 2020.

[40] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data mining and knowledge discovery*, 2:169–194, 1998.

[41] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.

[42] Quanan Huang and Huiyi Wang. Fundamental study of jerk: evaluation of shift quality and ride comfort. Technical report, SAE Technical Paper, 2004.

[43] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.