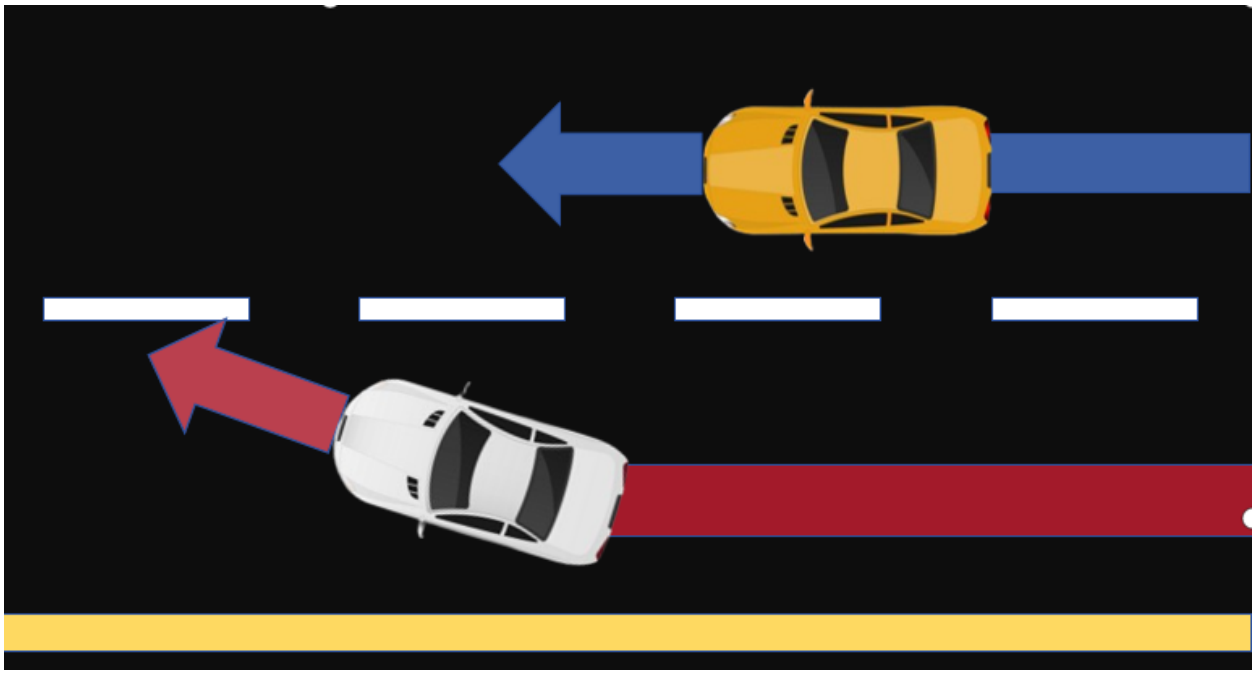
CHALMERS
UNIVERSITY OF TECHNOLOGY



A Machine-Learning Approach to Reduce the Risk of Collision When Changing Lanes

Master's thesis in Systems, Control and Mechatronics

Junyu Sun & Yaochen Song

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

A Machine-Learning Approach to Reduce the Risk of Collision When Changing Lanes

Junyu Sun & Yaochen Song

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024

A Machine-Learning Approach to Reduce the Risk of Collision When Changing Lanes
Junyu Sun & Yaochen Song

Supervisor: John Dahl, Zenseact
Examiner: Jonas Fredriksson, Department of Electrical Engineering in Chalmers

Master's Thesis 2024
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The image illustrates a top-down view of two cars on a highway, with one car making a dangerous lane change indicated by a red arrow.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Camera-Based Driver Monitoring for Intention-Aware Active Safety
Junyu Sun & Yaochen Song
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Advanced driver assistance systems support drivers to handle different complex traffic conditions. The support systems get traffic information from sensors and use algorithms to avoid risks. However, dealing with complex time series data from various sensors is challenging. In this thesis, a machine learning approach is proposed for threat assessment for lane changing. The emphasis is on vehicle state prediction and maneuvers for autonomous emergency steering. The work includes feature selection, model selection, and model validation. Feature selection is performed using the NSGA-II algorithm and correlation analysis to identify the most influencing features. This helps reduce the data dimension while maintaining prediction accuracy. An artificial neural network model structure inspired by ResNet is developed. This network structure is built from blocks, each with a shortcut. Various model configurations, including the number of input features and the network depth, are tested to find a reasonable tradeoff. In addition, driver-state information is also analyzed, and the "most probable gaze zone" data features enhance the model's performance. The proposed model is validated on real-world data and has good performance.

Keywords: advanced driver assistance system, time series data, feature selection, machine learning, neural network

Acknowledgements

Firstly, we want to thank Zenseact for providing a thesis opportunity and our industrial supervisors for initiating such an engaging thesis topic. In particular, we appreciate John for his guidance and help throughout the entire project. Lastly, we would like to thank our academic supervisor and examiner, Jonas, for initially navigating us in the right direction and continuously providing feedback for further improvements. Without any one of you, this thesis project would not have been successful.

Junyu Sun & Yaochen Song, Gothenburg, June 2024

Contents

1
Introduction

This chapter is organized as follows: First, it introduces a brief background of the Automatic Emergency Steering and Blind Spot Information Systems (AES-BLIS) problem. Then, it reviews the literature on existing solutions, provides a detailed introduction to the research aims and objectives covered in this thesis, and briefly describes the proposed method. Finally, it gives an outline of the following chapters.

1.1 Background

1.1.1 Advanced Driver Assistance System

Traffic accidents result in numerous fatalities and property damage, posing a significant concern worldwide. According to estimations from the World Health Organization (WHO), there are over 1.35 million fatalities caused by traffic accidents around the world every year [1].

To help reduce traffic-related injuries and fatalities, Advanced Driver Assistance Systems (ADAS) have been continuously developed and deployed over the last few decades [2]. ADAS uses automotive electronics, Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication, radar, and lidar to perform accurate detection and tracking for surrounding moving objects [3]. There are different types of ADAS based on levels of autonomy ranging from L1 to L5 [3]. While fully autonomous driving may be a long way off at the moment, L2 (partial automation) and L3 (conditional automation) levels of ADAS should be capable of performing tasks such as lane keep assist, auto-steering control, traffic jam assist, blind-spot assist, and intersection pilot. Studies have shown that such systems can significantly help reduce the total number of severe traffic accidents and casualties [4], enhancing road safety and saving lives.

There are mainly four modules in an ADAS system [6]. The first module is *Perception*, using sensors to turn the traffic conditions into different types of data. The second module is *Threat Assessment*, using the data to predict the future driving situation. The third module is *Tactical Decision Making* [7], making decisions about upcoming threats. The last module is *Actuation*, which performs the intervention when necessary.

1.1.2 AES-BLIS

Autonomous Emergency Steering (AES) systems steer the vehicle away from potential crash scenarios. This technology is often integrated with Autonomous Emergency Braking

(AEB) to provide a comprehensive safety solution when braking alone may not be sufficient to avert a collision.

Over the past few decades, many studies on AES have been conducted. AES utilizes advanced algorithms and a combination of sensors such as cameras, radar, and lidar to monitor the vehicle's surroundings and internal state. The system fuses the data to provide critical information and continuously analyzes the environment and the vehicle's status. Once it detects danger, it makes instructions for obstacle avoidance and lane-keeping. For example, [8] calculates the optimal steering maneuver for obstacle avoidance considering the vehicle dynamics. [9] proposes an algorithm that not only steers to avoid obstacles but also minimizes the deviation of the vehicle from the original path.

In manual driving, "blind spots" refer to the regions surrounding the vehicle that cannot be seen directly through the rear and side mirrors [10]. It is frequent for drivers to miss noticing vehicles in their blind spots when changing lanes on the road [11]. As a result, it is essential to provide blind spot detection in an AES system to alert the driver about objects in their blind spot. Numerous solutions have been developed to address the issue of blind spots [12], including sound-based sensors, radar-based systems, and camera-based systems. Sound-based sensors can provide short-range detection, radar systems offer longer-range capabilities, and camera-based systems deliver detailed visual information.

1.1.3 Driver Monitoring

In addition to the vehicle state and road situation, new kinds of ADAS systems can monitor driver parameters that may play an essential role in causing accidents, such as drowsiness, eating, drinking, smoking, unawareness, and distractions while driving. There are two kinds of driver monitoring systems, contact methods and contact-less methods. Contact methods include electroencephalogram (EEG), electrocardiogram (ECG), electromyography, or galvanic skin response. In contrast, contactless methods may use eye tracking, head movement, and facial expressions through camera detection and computer vision [15].

Driver behavior monitoring systems try to reflect whether the driver's current state of mind is suitable for driving and what actions they will take. This makes it easier to predict the future movement of the ego vehicle. For instance, intentional and unintentional driving behavior must be distinguished when predicting the vehicle trajectory. In [32, 16], online model-based prediction algorithms that use personalized driver information to predict unintended lane-changing behaviors are presented.

1.2 Problem statement

The problem investigated in this thesis is to develop a system for recognizing and predicting when a vehicle attempts to change lanes when there is a risk of collision. The system should then warn the driver or take control of the vehicle to avoid collisions by either steering or braking. This is denoted as the AES-BLIS problem.

The AES-BLIS problem is illustrated in Figure 1.1. The red car is the vehicle with the

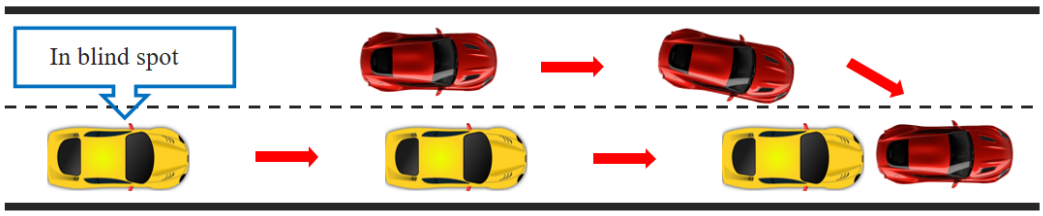


Figure 1.1: Illustration of AES-BLIS problem

ADAS system, and it is performing a lane change. However, the yellow car is in the driver's blind spot and moves at high speed. As a result, there is a high risk that the yellow car will crash into the red car. The challenge is to estimate the truly useful information and make accurate predictions of both vehicles' future motion. Hence, this thesis focuses on the threat assessment problem for lane changes, emphasizing the driver's attention or inattention to objects coming from the rear. The objective is to develop an algorithm to detect when a vehicle is beginning to change lanes unintentionally.

The problem can be formulated as a combination of trajectory prediction and collision risk assessment. With trajectory prediction, the ADAS system can predict whether the current vehicle will enter the lane next to it or not. With collision risk assessment, the ADAS system can analyze whether there is a risk of collision with other objects when a vehicle enters another lane.
The trajectory prediction problem can be formed as a regression problem because it involves estimating continuously the predicted distance to the next lanes. The collision risk assessment problem can be viewed as a classification problem because it involves determining whether there is a potential risk of collision in the future based on the current state of the vehicle and the surrounding object. The prediction horizon is an important parameter in both the regression problem and the classification problem. In the regression problem, the prediction horizon determines the target value of the dataset, and in the classification problem, the prediction horizon determines the size of the dataset. Also, a short prediction horizon does not give the system sufficient time to react, whereas an excessively long prediction horizon increases the likelihood of the system making erroneous actions.

1.3 Research Questions

This thesis attempts to address the following questions:

- How long should the prediction horizon be?
- Which data features are required to be input into the system?
- What selection criteria should be used when making feature selections?
- What model should be used to solve the problem?
- Does the introduction of driver-state information improve the system performance?
- Is it possible to combine the regression problem and the classification problem into a single problem to be solved?

1.4 Related Work

This section is about the related work in the threat assessment domain. Previous studies can be categorized into five classes: *threat metrics*, *optimization methods*, *formal methods*, *probabilistic methods* and *machine-learning methods*.

1.4.1 Threat Metrics

Numerous variations of threat metrics are commonly utilized as benchmarks for issuing warnings or initiating interventions [17, 18]. Time to Collision (TTC), Inverse Time to Collision (IVT), and Minimum Safe Distance (MSD) are the three most commonly used threat metrics to evaluate collision risks in driving situations within on-road environments [6].

TTC is defined as the time required for two vehicles to collide if they continue at their current speeds and trajectories. The mathematical formulation for TTC is given by:

$$TTC = \frac{D_{\text{nearest}}}{V_{\text{ego}} - V_{\text{nearest}}}, \tag{1.1}$$

where D_{nearest} represents the relative distance to the nearest vehicle, V_{nearest} denotes the velocity of the nearest vehicle, and V_{ego} signifies the velocity of the ego vehicle.

Inter-Vehicular Time (IVT) calculates the time until collision based on the assumption that only the ego vehicle continues on its path at its current speed. This is expressed as:

$$IVT = \frac{D_{\text{nearest}}}{V_{\text{ego}}}, \tag{1.2}$$

where the parameters are similarly defined as in TTC.

Minimal Safe Distance (MSD) indicates the distance needed between two vehicles to maintain safety, i.e., not colliding. This metric addresses scenarios where vehicles move at high speeds with minimal distance between them or are stopped due to traffic, ensuring a minimum safe interval.

1.4.2 Optimization Methods

There is also a wide range of ADAS systems using dynamic optimization for threat assessment and decision-making processes [6], where Model Predictive Control (MPC) is the most popular method.

An optimization problem involves finding the best solution from all feasible solutions, typically maximizing or minimizing an objective function while satisfying certain constraints. In practical driving assistance problems, the algorithm should be able to provide an optimal solution and, a controlling law over a finite time horizon. Recent developments in the MPC area, including designing multi-input, multi-output control systems and improving the control performance of single-input, single-output systems, have opened up opportunities for ADAS systems [20].

For instance, an MPC-based approach is introduced to avoid the ego vehicle running into another vehicle ahead [21]. Moreover, Nguyen [22] explores the application of a linear time-variant Model Predictive Control (MPC) within AES (as shown in Figure 1.2). In his research, multiple trajectories for emergency steering control are generated based on the distances and linear speeds between the ego vehicle and neighboring vehicles. This approach allows for dynamic adjustment in critical situations.

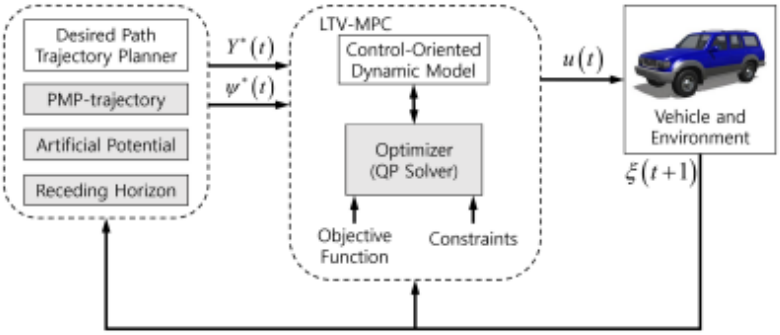


Figure 1.2: Autonomous emergency steering system using linear time-varying MPC Source: [22].

Figure 1.2 shows the layout of the Linear Time-Varying Model Predictive Control (LTV-MPC) based method for emergency steering control in autonomous vehicles. LTV-MPC is an advanced control strategy that adjusts the control inputs in real time, considering the varying dynamics of the vehicle.

Using signals such as $\dot{y}, \dot{x}, \psi, \dot{\psi}, Y, X$, the method first creates a planned path for the vehicle based on different driving situations through Pontryagin's Minimum Principle (PMP), a mathematical optimization technique used to determine the control inputs that minimize a cost function, and then checks it with the Artificial Potential Field (APF) technique, which uses virtual forces to avoid obstacles and ensure safe navigation [21].

After this, LTV-MPC develops a steering command (δ_f) by utilizing quadratic programming, a method to solve quadratic optimization problems. This command aims to follow the reference path while keeping the vehicle stable laterally. Although solving this problem in real-time is computationally demanding, the receding horizon approach makes the solution robust against changes in the environment. To manage the vehicle's navigation, Wang [23] establishes a quadratic optimization problem represented as the cost function:

$$\begin{aligned} J = \sum_{k=1}^{N} & \left[\mathbf{x}^T(k) - x_{\text{ref}}^T(k)\right] \mathbf{Q} \left[\mathbf{x}(k) - x_{\text{ref}}(k)\right] \\ & + \left[\mathbf{u}^T(k) - u_{\text{ref}}^T(k)\right] \mathbf{R} \left[\mathbf{u}(k) - u_{\text{ref}}^T(k)\right] + \Delta\mathbf{u}^T(k)\mathbf{P}\Delta\mathbf{u}(k) \\ & + \rho_{\mathbf{s}}\epsilon_s^2 + \rho_{\mathbf{e}}\epsilon_e^2 + \rho_{\mathbf{x}}\epsilon_x^2 \end{aligned} \tag{1.3}$$

where $\mathbf{Q}, \mathbf{R}, \mathbf{P}$ are positive semi-definite weight matrices applied to the discrete-time model of the vehicle (as specified in (1.3)), over the range $(k = 1, \ldots, N)$.The state variables $\mathbf{x}(k)$

represent the vehicle's state at time step k, and the control variables $\mathbf{u}(k)$ represent the control inputs at time step k. The slack variables ϵ_s, ϵ_e, and ϵ_x are introduced to ensure the feasibility of the constraints under different scenarios. The accompanying constraints are delineated as follows:

$$\begin{aligned} |\mathbf{u}(k)| &\leq \mathbf{u}_{\max}^k \\ |\Delta\mathbf{u}(k)| &\leq \Delta\mathbf{u}_{\max}^k \\ \mathbf{H_s}\mathbf{x}(k) &\leq \mathbf{G}_s^k + \epsilon_s \\ \mathbf{H_e}\mathbf{x}(k) &\leq \mathbf{G}_e^k + \epsilon_e \\ \Omega H_{x_{\text{mat}}}\mathbf{x}(k)^T &\leq \mathbf{G}_x + [\epsilon_x, \epsilon_x, \cdots, \epsilon_x]^T \\ \epsilon_s &\geq 0 \\ \epsilon_e &\geq 0 \\ \epsilon_x &\geq 0 \end{aligned} \tag{1.4}$$

Equation (1.4) crafts a multi-faceted objective function for vehicle control. It starts by accounting for state discrepancies to measure and correct the vehicle's path deviation. It then applies penalties for deviations in control inputs. The third element penalizes sudden changes in the control inputs, promoting smooth vehicle behavior and steady steering. Lastly, slack variables are introduced to maintain the problem's solvability across different scenarios, ensuring the vehicle's consistent trajectory following, lateral stability, and responsiveness to driver commands.

1.4.3 Formal Methods

Formal methods are divided into two categories: logic-based approaches and set-based approaches [6]. Logic-based formal methods leverage mathematical logic as their foundational framework for specifying system properties, verifying correctness, and reasoning about system behavior. Set-based formal methods primarily utilize set theory and related mathematical structures as their foundational framework for specifying system properties and reasoning about system behavior. For example, Higher Order Logic (HOL) can be used to formalize traffic rules [24]. The problem can also be formulated as a constraints satisfaction problem and solved by referring to reachability analysis and invariant set theory [25]. The advantage of formal methods is that they can help improve the interpretability, extensibility, and robustness of such algorithms.

In Ali's discussion [25], the focus is on understanding and controlling the behavior of discrete-time systems. He uses the notation and basic definitions of invariant and reachable sets for constrained systems as outlined. A thorough review of papers on set invariance theory is accessible in [26]. The system under consideration has its state, input, and disturbance vectors represented as $x(t)$, $u(t)$, and $w(t)$, respectively, and operates under constraints $x(t) \in \mathcal{X}$, $u(t) \in \mathcal{U}$, and $w(t) \in \mathcal{W}$. The state update function of the discrete-time system is defined by the equation $x(t+1) = f(x(t), u(t), w(t))$. The set of states from which the target set $\mathcal{S}$ can be reached in one time step is defined as $\text{Pre}_f(\mathcal{S}) \triangleq \{x \in \mathcal{X} \mid \exists u \in \mathcal{U} \text{ such that } f(x,u) \in \mathcal{S}\}$. This definition highlights the continuous feasibility of maintaining system states within $\mathcal{C}$ through appropriate control

actions across all time steps, ensuring the robustness and stability of the system under operational constraints.

1.4.4 Probabilistic Methods

Probabilistic approaches aim to derive insights from analyzing threats and making decisions in a probabilistic framework. Under specific assumptions, these methods assign probabilities to various possible events, such as vehicle collisions [27].

The Kalman filter, as a type of probabilistic approach, is widely used in autonomous emergency steering systems. In this paper [28], the localization algorithm utilized an extended Kalman filter to integrate GPS observations to control the vehicle's steering. The extended Kalman filter algorithm adheres to well-known steps. It starts with the prediction equations:

$$\begin{aligned} \hat{x}_k^- &= f\left(\hat{x}_{k-1}^+, u_{k-1}, 0\right) \\ P_k^- &= F_k P_{k-1}^+ F_k^T + Q \end{aligned} \tag{1.5}$$

These equations predict the next state and covariance based on the previous state, control inputs, and the process noise.

The calculation of the Kalman gain is given by:

$$K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + R\right)^{-1} \tag{1.6}$$

This step determines the weight given to the measurements versus the model's predictions. The correction equations adjust the predicted state based on actual measurements:

$$\begin{aligned} \hat{x}_k^+ &= \hat{x}_k^- + K_k\left(y_k - h_k\left(\hat{x}_k^-, u_k, 0\right)\right) \\ P_k^+ &= \left(I - K_k H_k\right) P_k^- \end{aligned} \tag{1.7}$$

These corrections are based on the difference between actual observations and predicted measurements. The initial state of the system is set using the first few GPS observations while the object is in motion.

Other probabilistic methods include using the Unscented Kalman Filter (UKF) [29] and Dynamical Bayesian Network (DBN) [30].

1.4.5 Machine-learning Methods

In ADAS, machine-learning plays an important role in helping the system recognize and understand the environment from sensor data by training on large amounts of data so that the system can make appropriate decisions and actions. For instance, a support vector machine can classify two scenarios: a vehicle leaving the lane and a vehicle remaining in the lane [31]. There are also different machine-learning models to predict lane departures [32], the movement of the ego car [33].

As for deep learning, approaches have been implemented within AES systems by numerous studies to estimate the distances between vehicles accurately. Halimi [34] demonstrates

how leveraging the computational prowess and predictive capabilities of deep learning algorithms can significantly boost AES's effectiveness. This is achieved by providing more accurate and reliable measurements of vehicle distances, enhancing road safety.

Arabi poses a comprehensive approach that uses deep learning and a monocular camera for estimating the distance of a following vehicle [35]. This method employs various models such as linear regression, a pinhole model, and artificial neural networks to compare their effectiveness in accurately estimating distances based on video data captured from a leading vehicle. Moreover, the M4Depth study proposes a novel system for monocular depth estimation in autonomous vehicles operating in unseen environments [36]. This method uses a multi-level architecture that calculates depth by creating visual disparity maps derived from motion information, offering a new way to enhance vehicle safety and operational efficiency in dynamic conditions.

With the rapid development of hardware computing power and the success of network structure development, there is a growing trend to use machine-learning for comprehensive ADAS applications. These comprehensive approaches not only enhance real-time threat detection but also foster adaptability to diverse driving scenarios. Taking ReasonNet as an example [37], the end-to-end driving model first receives the LiDAR input and multi-view RADAR input to extract features in the early stage of the framework. Then, it processes the temporal information and maintains a memory bank to store historical data. Last but not least, it uses a global reasoning module to analyze the relationship between the ego vehicle and the environment to detect adverse events.

1.5 Objectives and Method

The scope of the thesis is restricted to developing a system to help recognize and predict when a vehicle is attempting to change lanes when there is a risk of collision so that the ADAS system can take action to avoid collisions.

This thesis employs machine-learning techniques and neural network models to address the problem, as these methods have demonstrated strong performance across various driving scenarios and can efficiently handle large datasets, including time series data [37].

1.5.1 Objectives

This section shows the steps needed to solve the above AES-BLIS problem using machine-learning techniques.

1.5.1.1 Feature selection

Feature selection is essential for constructing an effective predictive model, particularly with a large feature set. Industry practices often prioritize features based on domain knowledge, as seen in prior work where velocity, acceleration of the ego vehicle, and the heading and distance to lane markers were selected for threat assessment and decision-making in Lane Keeping Aid (LKA) systems, demonstrating significant success [19]. However, manual feature selection may introduce human bias. In this thesis, we use a machine-learning

feature selection method to automatically rank data features based on their importance, which may improve the performance of our model.

1.5.1.2 Model development

This module centers on developing a driver intention prediction system using machine-learning. Since the input is time series data, the proposed model needs to process the relationship between data from the sensors in the time domain. Also, various methods and model parameters need to be tested to improve the model performance.

1.5.1.3 Evaluation and analysis

F1 score and Mean Squared Error (MSE) are often used to evaluate the classification and regression tasks, respectively. In addition to these two important criteria, our evaluation system also includes the number of trainable parameters. Reducing the number of trainable parameters while maintaining model performance reduces the computational cost and enhances the model's robustness.

1.5.2 Method

Dahl et al. proposed an intention-aware lane-keeping assist system based on machine-learning [16]. It shows the effectiveness of machine-learning and the usefulness of the gaze-tracking system. In this thesis, we extend that work to apply machine-learning and gaze-tracking system to handle the AES-BLIS problem.

Figure 1.3 from [16] shows the time series data of a vehicle navigating a turn. The current moment is represented by t, while the past moments are indicated by $t-1$, $t-2$, and $t-3$. The future moment is represented by $t+h$, where h denotes the prediction horizon. In this thesis, we use the same strategy of using the past information of the vehicle to predict the future state.

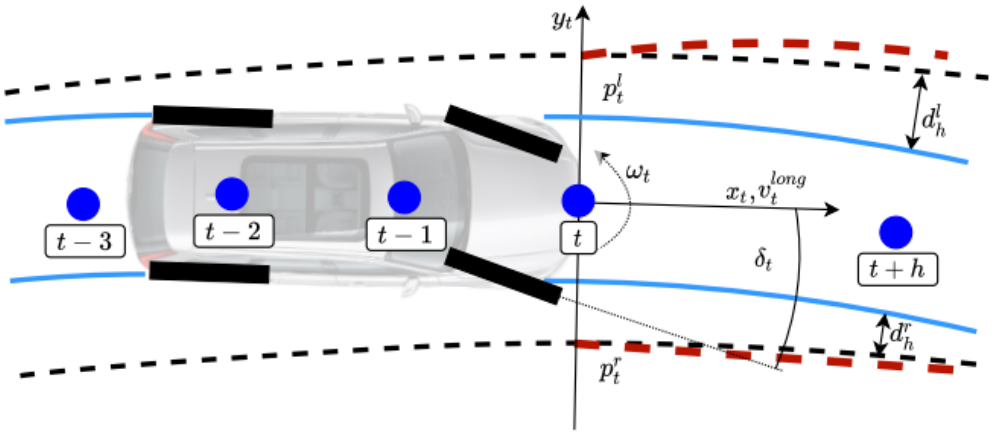


Figure 1.3: The time series data plot.

Our proposed method includes four steps, as illustrated in Figure 1.4. First, the raw data are pre-processed, filtered, and labeled to generate training and test data. For the classification problem, the data within the prediction horizon are labeled positive (danger), and the rest are labeled negative (safe). For regression problems, the vehicle motion trajectory and distance to the lane markers after the prediction horizon are viewed as the targets to predict. The regression trajectory task focuses more on the moving tendency of the vehicle, while the regression distance task shows the distance of the vehicle.

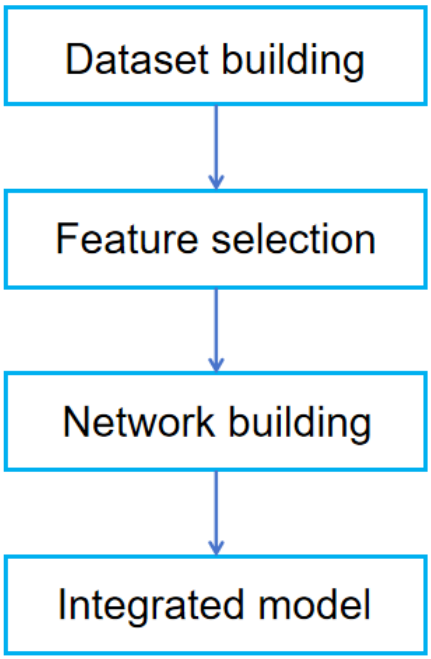


Figure 1.4: 4 steps of our proposed method

Second, the feature selection process combines correlation analysis with non-dominated sorting from the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [39]. Initially, the ranking of data features is determined through non-dominated sorting. Subsequently, the correlation method is employed to eliminate data features with redundant information.

Third, a neural network model structure inspired by ResNet [40] is used. The third step also includes a comprehensive analysis of the results obtained from the experiment, comparing the different approaches under different parameter settings and highlighting the best solution, especially the effect of the driver-state information.

Finally, an integrated model producing both the regression and classification outputs is investigated. By utilizing the hidden features of data, the model is intended to produce as good results as isolated models with fewer learnable parameters.

Figure 1.5 shows that we first build the time series labeled data based on the training targets and feature selection. The time series data features are categorized into three classes: vehicle information, object information, and driver-state information.

Vehicle information, also known as ego information, encompasses various dynamic parame-

ters of the vehicle, such as speed, acceleration, and steering angle. Object information involves identifying and tracking external entities, such as other vehicles, pedestrians, and obstacles. Driver-state information includes data on driver behavior and actions, such as braking patterns and steering maneuvers.

Vehicle and driver information are inputted into the regression model to predict the vehicle's future trajectory. We connect the regression model to the classification model so that the classification model accepts the information extracted from the vehicle information and target information combined with the object information to give the final classification prediction.

In the integrated model, the regression module's hidden features contribute to improving the classification module's performance. The integrated model makes precise predictions regarding the vehicle's future trajectory and safety assessment classification.

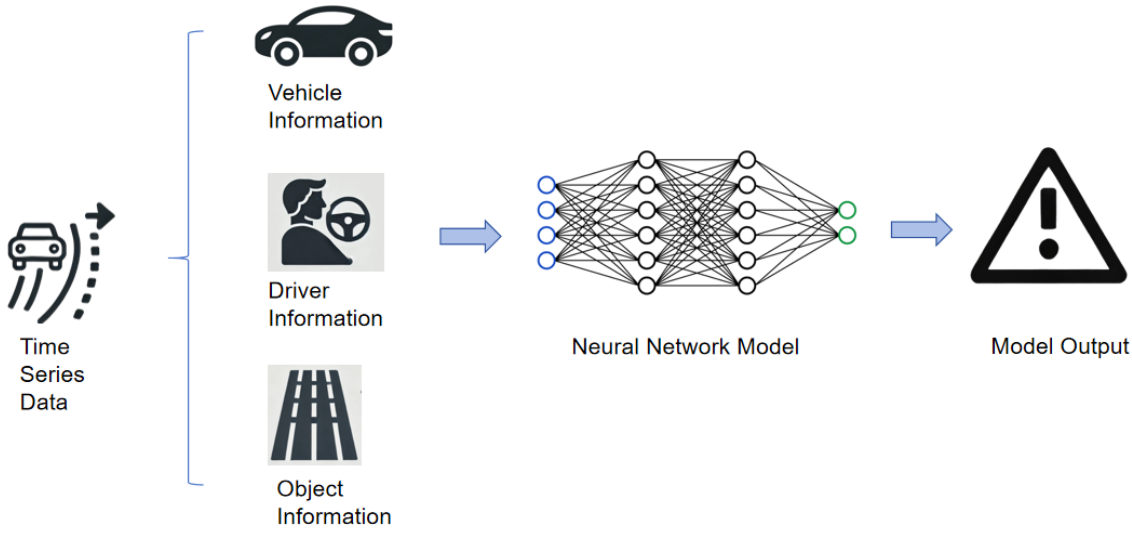


Figure 1.5: An illustration of the proposed method

1.6 Thesis Outline

The outline of the thesis is given as follows. Chapter 2 provides a comprehensive introduction to machine-learning. It starts with a basic overview of machine-learning theory, followed by a detailed description of all the methods and techniques used in our experiments. Each method is discussed in relation to its relevance and application to our study. Chapter 3 explores methods proposed for the AES-BLIS problem. It initially discusses the feature selection process, which uses two criteria to determine model input. This is followed by examining neural networks for classification and regression tasks. Chapter 4 presents the results obtained from our experiments and discusses them thoroughly. It examines the impact of various factors, such as the number of input features, network structure, and parameters. Additionally, the role of driver information input is analyzed. Finally, we present an integrated model with main blocks and side blocks. The main block

is used to get the regression output, while the side block is used to get the classification output. Chapter 5 summarizes the thesis and the overall project. It synthesizes all the findings, discusses any limitations encountered, and offers several directions for future research. This chapter aims to provide a comprehensive conclusion and set the stage for continued exploration in this field.

2

Machine-learning Introduction

This section covers the machine-learning techniques that are used in our proposed algorithm. First, it introduces the multi-objective optimization and NSGA-II algorithm [39], which is applied in the feature selection part of our model. Then it gives detailed information about neural networks. The key point of the experiment is to compare the impact of different model structures and settings, including the activation function and loss function. Furthermore, some common problems during the training of neural networks are shown, like over-fitting and vanishing gradients. Finally, the popular neural network structure, ResNet, is introduced, which is beneficial to solving certain training problems by utilizing skipped connections.

2.1 Multi-objective optimization

Multi-objective optimization (MOO) is a mathematical problem that involves optimizing several objective functions simultaneously, where optimal decisions need to be made in the presence of trade-offs between two or more conflicting objectives. This often indicates that improving one objective by traditional optimization methods, including gradient descent, Newton's method, and interior-point methods, may result in worsening another. The mathematical description of the multi-objective optimization problem is presented as follows [41]:

$$\begin{aligned} \min_{x} \quad & F = \{f_1(x), f_2(x), \ldots, f_k(x)\} \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, 2, \ldots, n \\ & x = (x_1, x_2, \ldots, x_m) \in \mathbb{R}^m, \end{aligned} \tag{2.1}$$

where, $f_i(x)$ denotes the i-th objective function, x is the vector of control variables, n signifies the total count of constraints, and k represents the overall quantity of objectives to be optimized. In real-world problems, multiple criteria are often focused on. For example, in the machine-learning domain, larger and deeper networks generally perform better. However, hardware limitations and requirements for training speed require fewer network layers. In this case, "better accuracy performance" and "smaller network size" form two different criteria. By balancing different criteria, MOO offers robustness and flexibility in the process of decision-making by evaluating different solutions that satisfy the constraints.

2.1.1 Pareto Optimality

Unlike convex optimization, where both the objective function and the constraints are convex functions, multi-objective optimization deals with problems involving multiple conflicting objectives. This leads to the introduction of the Pareto optimal solutions concept. Originating from the work of Vilfredo Pareto [42], the Pareto front represents a set of non-dominated solutions, termed Pareto Optimal Solutions, which collectively form the Pareto Optimal Front. This front defines the boundary of achievable trade-offs between objectives, and all the solutions on the front are optimal. The equation of the Pareto optimal solution set is shown as follows:

$$P = \{x \in \Omega \mid \neg \exists x_k \in \Omega : x_k \prec x\}, \tag{2.2}$$

where the x^* represents the Pareto optimal solutions, and $x_k \prec x^*$ indicates every criterion of x_k is better or equal to x^* .

Each solution on the Pareto front outperforms all other solutions on at least one objective without worsening any other objective. The Pareto-optimal solutions provide a comprehensive view of the trade-offs inherent in the problem space.

2.1.2 NSGA-II and Non-dominated Sorting

Traditional optimization methods find it difficult to directly optimize objects in MMOs, so different approximative numerical methods have been proposed in the literature, one of them being evolutionary algorithms. An evolutionary algorithm is a kind of heuristic optimization algorithm that searches the solution space of a problem through random mutation and crossover operations on generations of individuals.

The basic steps of an evolutionary algorithm include: (1) creating an initial population representing potential solutions. (2) selecting individuals based on the fitness function,i.e., the value of the objective function. (3) generating individuals by crossover operations. Here, the crossover introduces diversity and explores the solution space by exchanging some of the genes of the parent individuals to generate new offspring individuals. (4) random mutation of newly generated individuals is used to increase diversity. (5) updating the population according to a selection strategy. (6) if the stopping condition is satisfied or the algorithm already reaches the maximum iteration, the algorithm outputs the best individual based on the fitness function. Otherwise, it continues with steps 3 to 5.

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) [39], proposed by Deb et al., introduces fast non-dominated sorting to better identify Pareto optimal solutions. This method efficiently separates the solutions into layers of dominance. It starts with calculating the dominance relations among them. Solutions that are not dominated by any others ($n_i = 0$) form the first layer, representing the current Pareto front. For each individual j belonging to the dominating set s_i of individual i, its dominance count n_j is decreased by 1. When n_j becomes 0, individual j is placed into the next non-dominated level. In this way, all the solutions are categorized into different layers. In NSGA-II, lower-layer individuals, being closer to the true Pareto front, are favored for selection in the next generation.

2.2 Neural Network

The early conception of neural networks can be traced back to the work of McCulloch et al. [44], who introduced a simplified computational model of a neuron. This work lays the groundwork for later developments in the field. Multilayer perceptron has been widely studied and applied as an important architecture since the application of backpropagation algorithms.

In recent years, the AlexNet architecture [45], which won the ImageNet competition in 2012, significantly increased the interest and research in deep learning. From then on, the development of neural network architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) has been commonly used in all kinds of fields, including computer vision and natural language processing. CNNs are particularly effective for image and spatial data processing due to their hierarchical structure and ability to capture local patterns. RNNs are designed for sequence prediction tasks, incorporating cycles to handle temporal dependencies.

2.2.1 Multilayer Perceptron

Multilayer Perceptron (MLP) is a class of feedforward artificial neural network [48]. It consists of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer, as illustrated in figure 2.1. Each layer, except the input layer, contains multiple neurons that are fully connected to the neurons in the adjacent layers. Neurons in each layer apply an activation function to the weighted sum of inputs from the previous layer.

For each layer of the MLP's computational process, $\mathbf{a}$ denotes the input vector, $\mathbf{W}^{(l)}$ represents the weight matrix for layer l, $\mathbf{b}^{(l)}$ is the bias vector for layer l, g is the activation function. The process of the calculation in figure 2.1 can be written as follows:

$$\mathbf{y} = g\left(\mathbf{W}^{(l)} \cdot \mathbf{a}^{(l)} + \mathbf{b}^{(l)}\right) \tag{2.3}$$

Depending on the activation function in the output layer, the output y can be used for tasks like classification or regression. The goal of classification is to predict a discrete label or category for a given input, while regression is to predict continuous values.
MLPs are widely used for solving complex problems, e.g., pattern recognition or classification. One of the significant advancements in MLP was made when Rumelhart et al. [49] introduced the backpropagation training algorithm, allowing MLPs to adjust their internal parameters and more effectively learn from data.

2.2.2 Activation Functions

The output of each layer in MLP is a linear transformation of the inputs. As a result, nonlinear activation functions are introduced so that the network can fit any function. Common activation functions include ReLU (Rectified Linear Activation), sigmoid, tanh (Hyperbolic Tangent), and softmax.

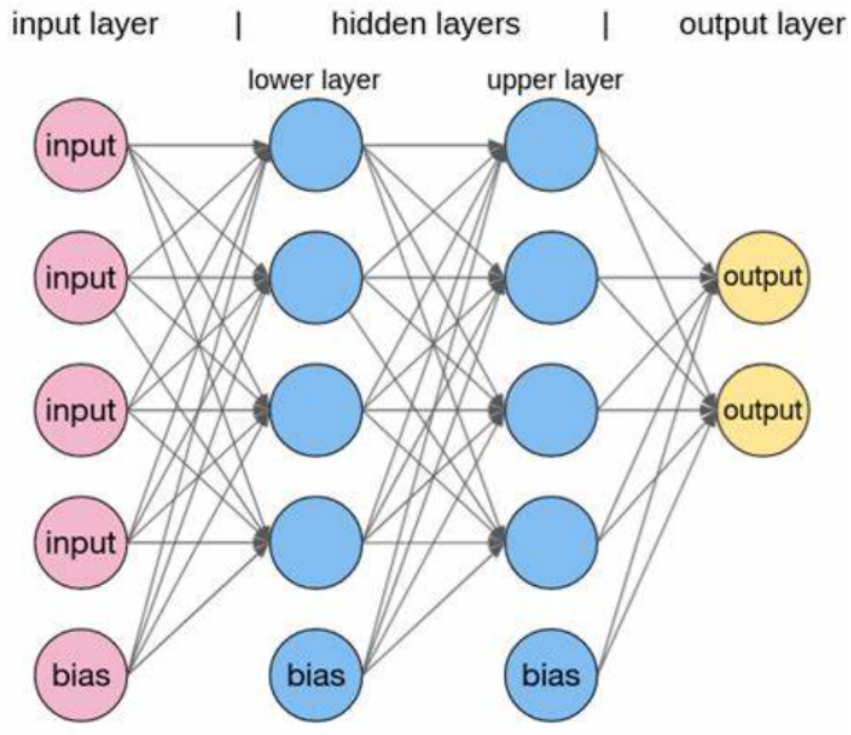


Figure 2.1: Multilayer Perceptron. Source: [47].

As shown in Figure 2.1, the ReLU activation function is widely used in neural networks due to its simplicity and effectiveness [43] in addressing the vanishing gradient problem [50]. ReLU is defined mathematically as $f(x) = \max(0, x)$, which means it outputs the input directly if it's positive. Otherwise, it outputs zero, as Figure 2.2 shows. This function has been widely appreciated for its ability to provide a nonlinear transformation with less computational complexity.

Furthermore, PReLU extends the ReLU function by introducing a learnable parameter α that scales the negative inputs instead of setting them to zero, as seen in the formula $f(x) = \max(\alpha x, x)$ [46]. α allows PReLU to adapt during training and possibly overcome the limitations of ReLU, such as the problem of dying neurons where neurons can become inactive and only output zero.

2.2.3 Optimizer and Loss functions

2.2.3.1 Optimizer

During the training of a neural network, the optimizer is responsible for updating the parameters based on the gradients of the loss function. In figure 2.1, the optimizer calculates the learnable parameter between different layers. Common optimizers include Stochastic Gradient Descent (SGD) [52], Adam [53], RMSprop [54], and Adagrad [55], each with its own update rule and hyperparameters.

The Adam optimizer estimates the momentum v_t and quadratic moments s_t of the gradient

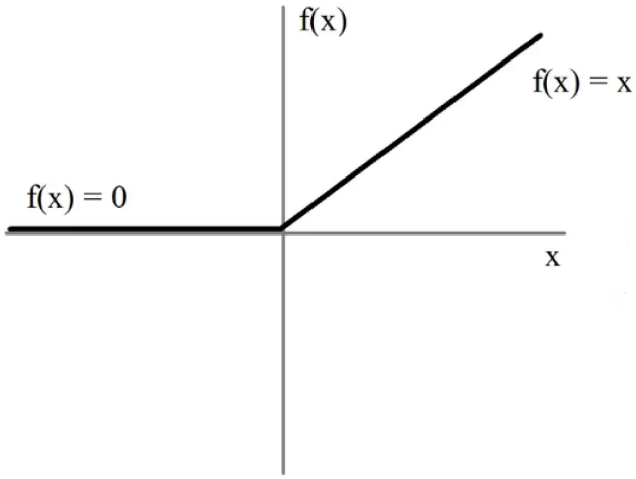


Figure 2.2: ReLU activation function. Source: [51]

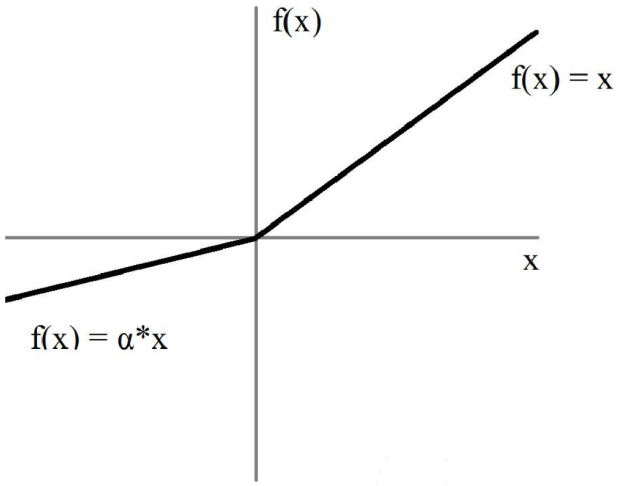


Figure 2.3: PRelu activation function. Source: [51]

to correct the gradient $g_t^{'}$, speeding up the optimization process and increasing the convergence rate. The Adam algorithm has become one of the most common optimization algorithms in deep learning due to its simplicity, high efficiency, and easy implementation.

2.2.3.2 Loss Function

The loss function measures the difference between the predicted and actual output, guiding the optimization process. Neural networks use the loss function to quantify the difference, or degree of loss, between the model's predicted results and the true labels.

Mean Squared Error (MSE) loss function and Binary Cross-Entropy (BCE) loss function are two common loss functions:

$$MSE = \frac{1}{D}\sum_{i=1}^{D}(y-p)^2 \tag{2.4}$$

$$BCE = -\frac{1}{D}\sum_{i=1}^{D}\left(y\log(p) + (1-y)\log(1-p)\right) \tag{2.5}$$

where y indicates the true value and p indicates the predicted value. The former loss function is used in regression tasks to calculate the mean of the squared difference between the predicted value and the true value. The latter loss function is used in classification tasks. If the model makes an incorrect prediction, then the loss function penalizes it.

2.3 Common problems and solutions

This section is about the common problems in machine-learning and their solutions. Problems like over-fitting, degradation, and class imbalance are included. Techniques like early stopping, Residual Network, and data augmentation are also introduced.

2.3.1 Over-fitting

Over-fitting occurs when a machine-learning model performs well on training data but poorly on unseen test data. This phenomenon usually occurs when the model is too complex (too many model parameters) or the amount of training data is too small.

Despite that increasing the amount of training data is one of the most effective ways to prevent over-fitting, there are also several other common solutions for over-fitting. For example, a regularization technique is to add an extra term to the loss function to penalize the model parameters, including L1 regularization and L2 regularization [56].

$$Loss_{L1} = Loss + \lambda\sum_{i}|w_i| \tag{2.6}$$

$$Loss_{L2} = Loss + \lambda\sum_{i}w_i^2 \tag{2.7}$$

Another way is to use cross-validation to assess the generalization ability of the model by dividing the dataset into multiple subsets and training and validating on different subsets.

Early stopping is a simple but effective method to prevent over-fitting. During the training epochs, the model's performance on the validation set is monitored, and when it is optimal, the training process stops.

2.3.2 Degradation

As the depth of the neural network increases, the performance of the model stops improving and even begins to decline, such a situation is called degradation. As seen in figure 2.4, the 20-layer neural network is actually performing better than the 56-layer neural network on both the training dataset and the validation dataset. This phenomenon is abnormal, as deeper networks should be able to learn more complex features and representations, resulting in better performance.

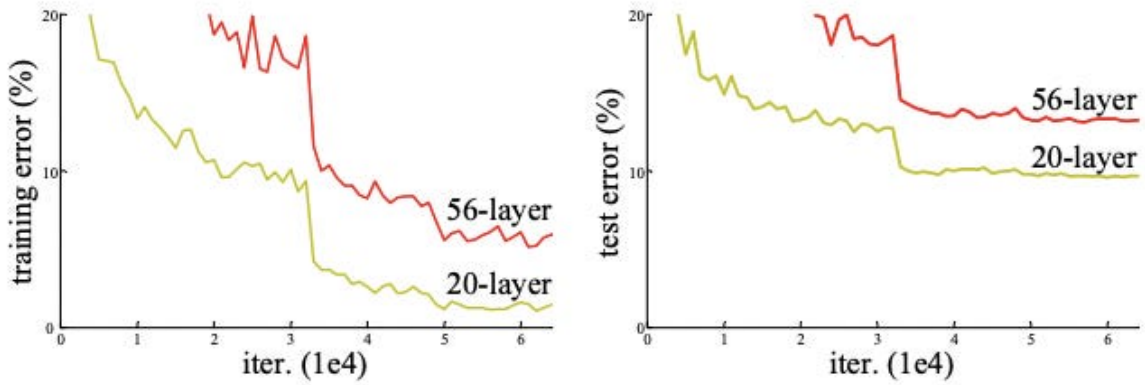


Figure 2.4: Training and test errors for 20-layer and 56-layer deep residual networks (ResNets) over iterations. Source: [40].

Degradation occurs due to the increased difficulty of optimizing the parameters of deep neural networks, which involves tuning parameters such as learning rate, weight initialization, and network architecture [40]. As the depth of the network increases, gradient propagation becomes more difficult, leading to the problem of gradient vanishing or gradient explosion, making the model difficult to train.

Residual Network (ResNet) is a neural network design created by He et al [40]. Its main feature, the unique residual blocks with skip connections, enables it to bypass some layers, effectively handling information flow in deep networks. As figure 2.5 shows, the residual block in ResNet consists of a skip connection to perform identity mapping, directly adding inputs to outputs. Residual connection allows gradients to propagate to previous layers via a direct path, not just via chaining from layer to layer. This results in a shorter path for the gradient to pass, mitigating gradient vanishing and explosion problems.

In this thesis, we will compare the normal MLP network with the MLP network that has residual blocks to prevent potential degradation. Furthermore, Batch Normalization (BN) is also used for accelerating neural network training and improving performance [57]. The

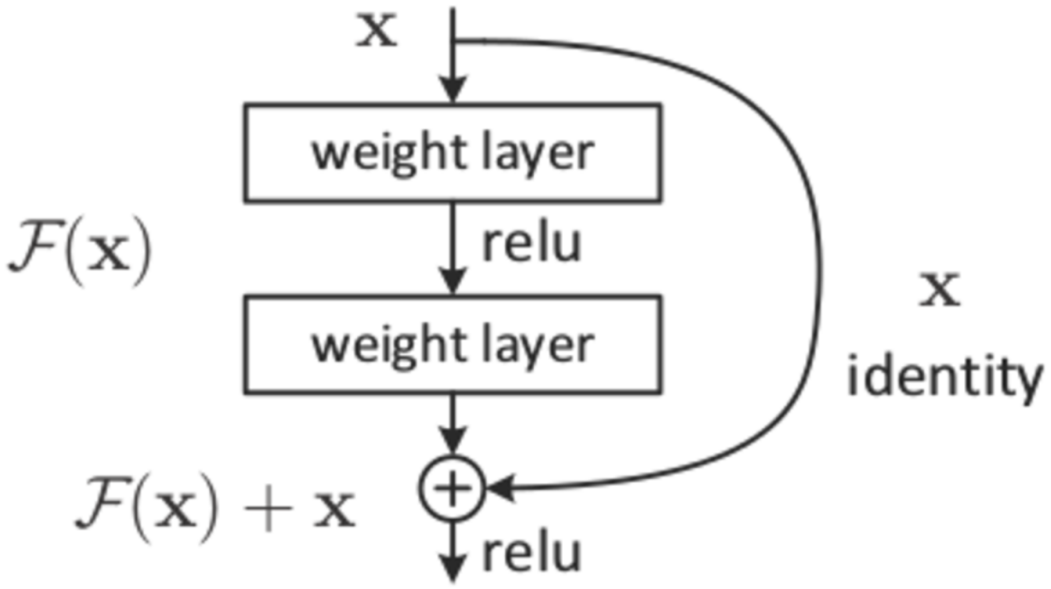


Figure 2.5: ResNet. Source: [40].

Batch Normalization normalizes the data during training to ensure that the mean of the inputted data is close to 0 and the standard deviation is close to 1.

3
Methodology

This section presents a comprehensive solution for the AES-BLIS problem following the method presented in Chapter 1. It includes the following parts: (1) extracting a subset of data from raw data and annotating it, (2) a feature selection module based on both non-dominated sorting from the NSGA-II algorithm and correlation analysis methods, (3) effective neural network model structure inspired by ResNet to predict the vehicle trajectory and current level of danger, and (4) an integrated model combining regression model and classification model to improve performance and save computation resource.

3.1 Dataset Building

The data provided contains three types of information: vehicle information, object information, and driver information. Vehicle information includes thousands of time series data from different sensors, including vehicle distance, speed, acceleration towards the lane markers, and so on. Object information gives the state of other moving objects, and driver information mainly focuses on the most probable gaze zone and driver unawareness. The most probable gaze zone represents where the driver is most likely looking, ranging from 0 to 9. 0 represents the driver-side mirror, 1 represents the front area, 2 represents looking down, 3 represents the driver information module, 4 represents the rear view mirror, 5 represents center stack display, 6 represents the place where a passenger in the front seat looks straight, 7 represents passenger side mirror, 8 represents driver side window, and 9 represents passenger side window.

In addition to the most probable gaze, the influence of driver forward unawareness is used. Driver forward unawareness refers to the driver's awareness of the road information in the front. It is determined by the time elapsed since the driver last focused on that area. When the driver is actively looking at the area, the value of driver forward unawareness is zero. Otherwise, it increases by 40 every second.

To capture historical driving information, the data features are sampled multiple times. When the data are inputted into the network, we not only input the data at the current time instant but also sample the data in the second before the current time as historical information. This historical information helps the network to understand the trajectory of the vehicle movement better, as stated in [16]. We call this sampled time range, the history depth. Specifically, for a certain data feature d, instead of just sampling the data at time t d_t, given a sampling interval I, a set $D_t = \{d_t, d_{t-I}, \ldots, d_{t-nI}\}$ is fed to the neural network model. In the experiment, $n = 5$ and $I = 0.2s$ are used. These settings are proved to be effective in [16].

When constructing the dataset, the data before changing the lane within the prediction horizon is labeled as a positive category, and the other data are labeled as a negative category. In this way, the numbers of positive cases and negative cases are equal, which is essential to improve the system's performance.

3.2 Feature Selection

Feature selection is crucial for improving the model's prediction performance, making the model faster and more cost-effective. By selecting only the most relevant features, the model generalizes better to unseen data, improving its predictive performance. Additionally, good feature selection can help to understand the problem better [61]. Another important factor is controlling the neural network model's total number of parameters and computation cost, which leads to high demands on the vehicle's processing hardware.

In this thesis, thousands of time series data provided are likely to contain a lot of redundant information. Developing a good feature selection module is essential. There are several successful cases of feature selection. For example, researchers have identified important features such as ego vehicle speed, acceleration, and steering angle through expert experience [16]. However, in addition to the features chosen by the expert, there may be other features that are helpful for model performance. When there are too many features, the model may learn the noise and details in the training data rather than the true pattern of the data, leading to over-fitting the training data and degradation on unseen data. A machine-learning based feature selection method is applied, ranking the time series' features in a specific order. Machine-learning based feature selection methods are automated and rely on objective criteria derived from the data rather than subjective judgment. Moreover, machine-learning based methods can capture hidden patterns in the data that may not be apparent to domain experts.

To perform feature selection, we need to assign importance to different features so that we can rank them. The importance is defined by the change in the data patterns. For example, if the vehicle remains at high speed when moving forward and slows down when performing lane change, the model may need to focus on decreasing longitudinal speed. Two criteria are proposed to find these data patterns: relative rate of change and absolute rate of change.

Relative rate of change is defined as follows: consider a sampled time series data feature as d_t, ordered by time instant t, where $t = 0$ indicates the start of the driving record, and $t = N$ indicates when the vehicle encounters a critical event. Define two different time horizons H_1 and H_2, and let $\overline{d_t^{H_i}}$ represent the mean value of the data feature from time $t - H_i$ to time t. Similar evaluation criteria Cr_1 and Cr_2 focusing on the relative rate of change:

$$Cr_i = \left| \frac{\overline{d_N^{H_i}} - \overline{d_t^{H_i}}}{\overline{d_N^{H_i}}} - \frac{\overline{d_t^{H_i}} - \overline{d_{t-(N-t)}^{H_i}}}{\overline{d_t^{H_i}}} \right| \tag{3.1}$$

$(\overline{d_N^{H_i}} - \overline{d_t^{H_i}})/(\overline{d_N^{H_i}})$ shows how the average value of the data feature changes before the lane

change happens, while $(\overline{d_t^{H_i}} - \overline{d_{t-(N-t)}^{H_i}})/(\overline{d_t^{H_i}})$ shows how this data feature normally changes over time. The latter term is necessary because data, maps, and GPS information are continuously changing in driving record data. Without it, such information may introduce noise into the model, reducing the accuracy of the model prediction.

Similarly, the absolute rate of change can be defined as follows:

$$Ca_i = \left|(\overline{d_N^{H_i}} - \overline{d_t^{H_i}}) - (\overline{d_t^{H_i}} - \overline{d_{t-(N-t)}^{H_i}})\right| \tag{3.2}$$

In equation 3.1, by adjusting the different time horizons H_i, we obtain data features that either have relatively large value changes in a short period before the lane change or over a much longer period. Non-dominated sorting is then used to sort all the data features according to these criteria. Figure 3.1 illustrates how to determine the ranking based on non-dominated sorting and different criteria. Assume, there are six points indicating different data features that correspond to three Pareto fronts. For example, the blue Pareto front includes features that are only dominated by those on the red front but not by the other. We first pick the data features on the red front, then those data features on the blue front, and finally the data feature on the brown front.

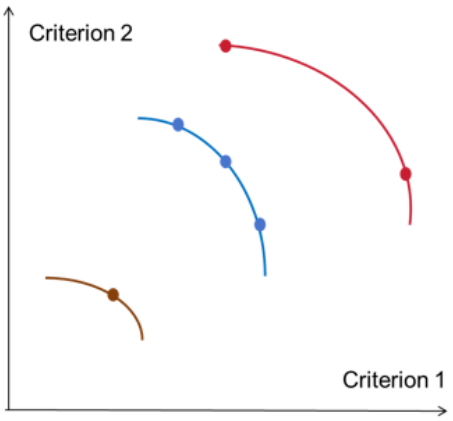


Figure 3.1: Illustration of non-dominated sorting on different criteria

In this thesis, only a subset of 175 data features are selected out of thousands of features, as they show an apparent relative change rate. Non-dominated sorting ranks these 175 data features and the ranking criteria are the relative rate of change over 3 seconds and 6 seconds.

Correlation analysis is also used for feature selection. Correlation analysis assesses the correlation between each feature and other features. When two features have a very high correlation coefficient, the lower-ranked feature is removed to reduce redundancy in feature information. Specifically, when the correlation coefficient is close to 1 or -1, it indicates a high positive or high negative correlation between the variables. When the value of one variable increases, the value of the other variable also increases or decreases. When the correlation coefficient is close to 0, it means there is no linear relationship between the variables, i.e., no correlation.

Figure 3.2 shows an example of perform the correlation analysis after using the non-dominated sorting to rank the features in Figure 3.1. In this example, there is a high correlation between the first red data feature and the first blue data feature, so we delete the first blue data feature to reduce the redundant information.

Figure 3.2: Illustration of correlation analysis

3.3 Network Building

Historical driving information is crucial in the ADAS system. Many models are developed to maintain this information, including RNN, [59], and LSTM, [60]. However, these models have drawbacks such as gradient vanishing and explosion. This is due to long-term dependencies in the backpropagation process, making the update process difficult. Moreover, their computation efficiency is not high. For longer sequences, RNN training and inference may become very slow. In this thesis, the basic model is an MLP, which is a simple neural network structure that can be computed efficiently. As discussed in Section 3.1, previous states are fed to the network to introduce historical information to the MLP model.

The MLP model may experience degradation because the model layers can be very deep. A ResNet-like network structure is designed as Figure 3.3 shows. The model is built up by blocks. The block-based structure allows complex neural networks to be broken down into smaller, more manageable modules. In each block, there is a fully connected layer, a BatchNorm layer, a PReLU layer, followed by another fully connected layer, a BatchNorm layer, and a PReLU layer.The fully connected layer plays the role of feature extraction and feature combination in neural networks. Batch normalization stabilizes and accelerates training by normalizing input features. The PReLU activation function increases non-linearity of the model and address the problem dead zones of the ReLU function.

Additionally there is a shortcut, as shown in Figure 2.5, which facilitates gradient flow during backpropagation, effectively mitigating the vanishing gradient problem. Throughout the thesis, the models without and with shortcuts are referred to as SimpleMLP and ResNetMLP, respectively. This comparative analysis underscores the advantages of incorporating residual connections in enhancing model robustness and effectiveness.

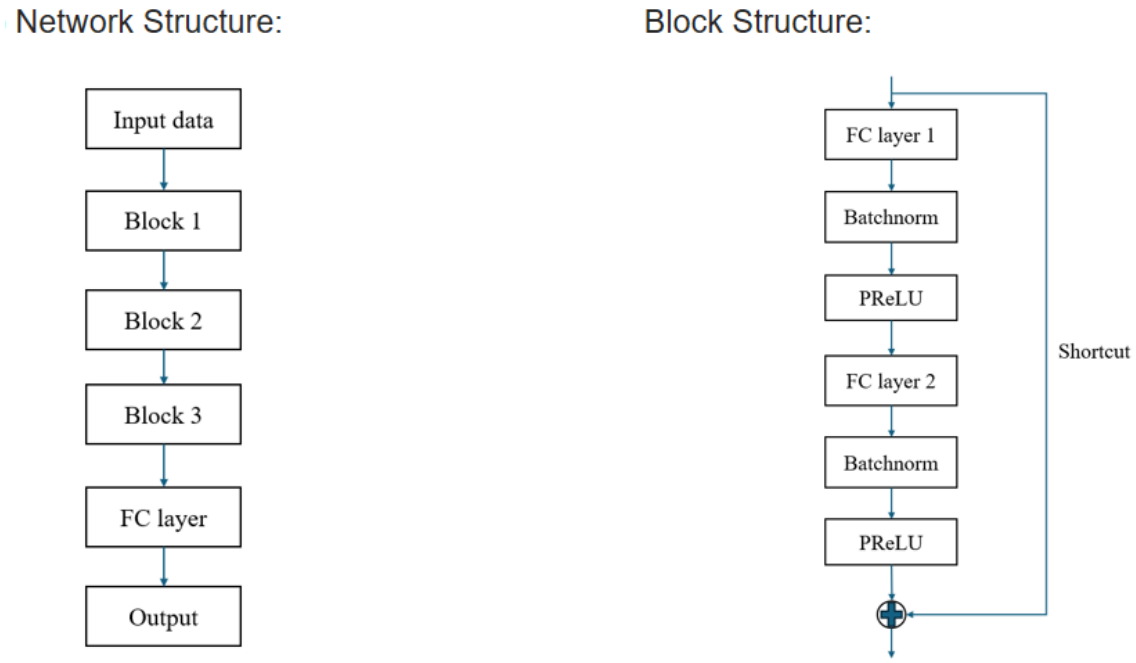


Figure 3.3: Illustration of the network structure and block structure.

3.4 Integrated Model

The underlying idea of the integrated model is that the regression task is more complex and needs more neural network layers to process. In contrast, the classification task is simpler and requires few computation resources. As a result, we want to use the regression model to improve the classification model and reduce computation resources. Moreover, the regression task's optimizing targets include distance, speed, and acceleration. This information helps the regression model have a clearer direction for optimizing space. As a result, the regression model's generalization ability outperforms the classification model.

The idea is illustrated in figure 3.4. The blue circle represents the "optimizing space", which defines the range of possible model parameter configurations. This space encompasses all potential solutions that the algorithm explores. The objective of the neural network is to navigate within this parameter space to identify the set of parameters that minimize the loss function and achieve the optimal model parameters. The green space represents a minimum. Minimas can be either local or global. A local minima may result in a model that performs well on the training set, but poorly on the test set. This phenomenon can be considered as a form of over-fitting. The black space represents the optimal parameter setting for the network. In this position, the model not only performs well on the training data, but also shows strong generalization ability on the test data. The red dot is the model parameters' starting point, and the yellow arrows represent the possible optimizing directions. In the classification tasks, where the outcome is typically binary (positive or negative), the model is more prone to losing the direction of the optimal solution and getting stuck in local minima. In contrast, regression tasks typically provide more detailed targets, which help the model navigate more effectively through the optimization space

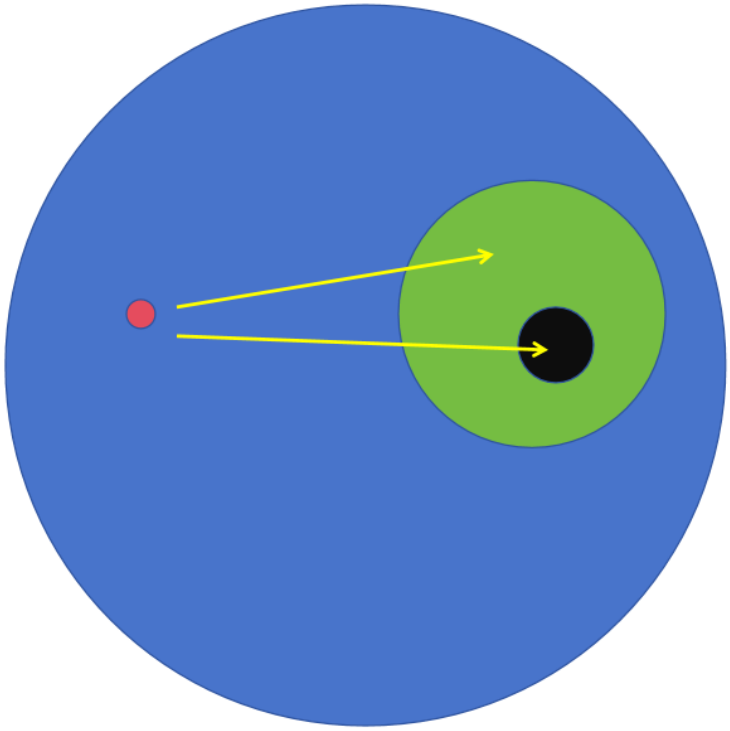


Figure 3.4: Comparison of optimization trajectories in the classification and regression models

towards the optimal solution.

The integrated model uses the regression model to help improve the classification model's performance by inputting the hidden features of the regression model into the classification model. Figure 3.5 illustrates the integrated model layout. Vehicle data features and driver-state information are inputted into the main block part to extract relevant features for the regression task. The object data features and the features produced by the main block are then input into the side block to generate the classification output.

3.5 Implementation

The prediction models are implemented in Python, using PyTorch, NVIDIA Quadro RTX 4000 Mobile Max-Q GPU, Driver version 535, and CUDA version 12.2. Unless otherwise specified, the model is trained for 35 epochs during the training process, using the Adam optimizer with a learning rate of 10^{-3}. The criterion to evaluate the model is MSE loss for regression tasks and F1 score for classification tasks. The model with the best performance on the validation set is chosen as the result of the training.

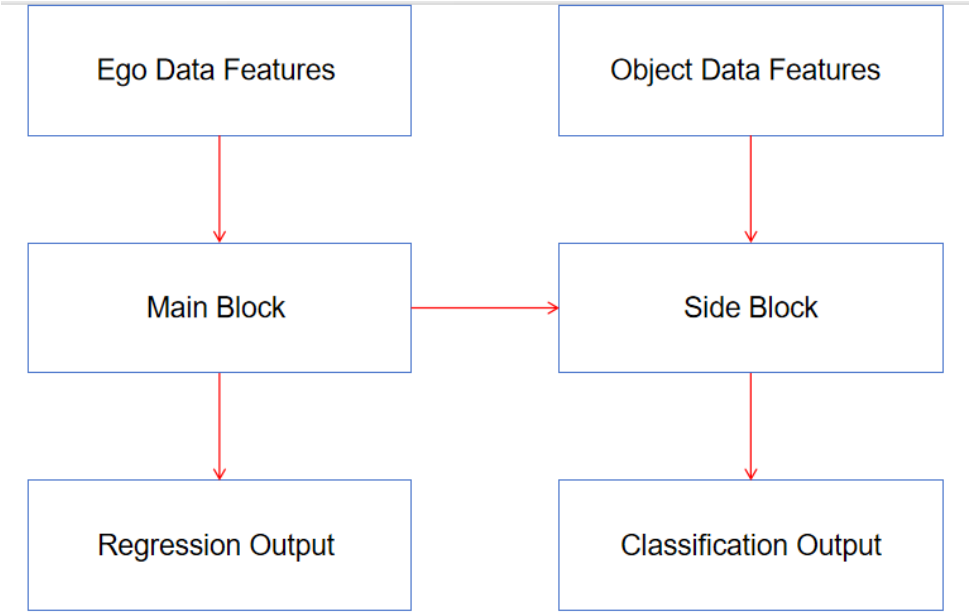


Figure 3.5: Integrated model structure

4

Results and Analysis

This chapter summarizes the results. First, it shows the significance of feature selection when creating a dataset. Next, it compares the performance of different numbers of input features, network parameters, and prediction horizons. It then analyzes the driver behavior pattern and discusses the impact of introducing the driver information to the model. Finally, it shows the results of the integrated model for solving classification and regression tasks simultaneously.

4.1 Feature selection

4.1.1 Ranking by non-dominated sorting

In this experiment, different feature sets are inputted into the same model architecture to show the necessity of feature selection. The ranking criteria are the relative rate of change over 6 seconds and 3 seconds. After the non-dominated sorting, correlation analysis is used to reduce redundancy. As shown in Table 4.1, "selected" indicates the best 30 ranked features. "sample" indicates a subset of randomly selected 30 features (which can be regarded as the basic performance of 30 input features), and "worst" indicates the worst 30 ranked features. Finally, "whole" indicates that all 175 selected features have been sent to the model.

Category	selected	sample	worst	whole
ResNetMLP Trajectory	**0.1770**	0.2052	3.153	failed to converge
ResNetMLP Distance	**0.5209**	0.5734	2.1092	failed to converge
ResNetMLP Classification	**0.7165**	0.7147	0.6668	0.6666

Table 4.1: Comparison between different subsets of features

In Table 4.1, the model performs best when using the best feature subset in all categories. This indicates that feature selection can significantly improve the model performance. In addition, the worst data subset performs worse than the sample data subset, indicating that irrelevant or noisy features can negatively affect model predictions. In most cases, using all features results in a model that fails to converge or has the worst performance. This suggests that feature redundancy and noise increase model complexity and decrease performance instead.

4.1.2 Ranking Criteria

Proper feature ranking is crucial as it directly influences the model's performance. We conduct experiments to assess the impact of different sets of ranked features on the model's effectiveness.

4.1.2.1 Ranking whole 175 data features

In this experiment, we rank all 175 data features with different criteria. Table 4.2 shows the model performance of different top data features ranked by different criteria, "relative 6s 3s" includes the relative rate of change over 6 seconds and 3 seconds, "relative 2s 1s" includes the relative rate of change over 2 seconds and 1 second, and "absolute 2s 1s" includes the absolute rate of change over 2 seconds and 1 second.

Ranking criteria	top 10 features	top 20 features	top 30 features
Classification relative 6s 3s	0.6953	**0.7041**	**0.7115**
Classification relative 2s 1s	**0.7052**	0.6950	0.6856
Classification absolute 2s 1s	0.6776	0.6653	0.6469
Trajectory relative 6s 3s	0.1822	**0.1626**	**0.1312**
Trajectory relative 2s 1s	**0.1755**	0.1730	0.1670
Trajectory absolute 2s 1s	0.1758	0.1703	0.1491
Distance relative 6s 3s	**0.4881**	**0.4576**	**0.3538**
Distance relative 2s 1s	0.5461	0.5309	0.5171
Distance absolute 2s 1s	0.5308	0.4866	0.4695

Table 4.2: Model performance of different ranking criteria ranking all data features

Comparing "relative 2s 1s" and "absolute 2s 1s", it is noticeable that data features sorted by relative rate of change criteria perform better on the classification task, while data features sorted by absolute rate of change criteria perform better on the regression task. Comparing "relative 6s 3s" and the other two criteria, we see that data features sorted by long-term relative rate of change generally perform better than those sorted by short-term criteria.

4.1.2.2 Ranking selected 30 data features

In this experiment, we rank the selected 30 features in 4.1.1 with different criteria. These 30 features exclude the noisy data features. It is noticeable that the "absolute 2s 1s" criterion performs better than the "relative 6s 3s" criterion on the regression task, see Table 4.3 and Table 4.4.

The bad performance of the "absolute 2s 1s" criterion in the last experiment and the good performance in this experiment show that the absolute rate of change may wrongly evaluate the bad features highly. However, if the bad features are excluded, it can rank the good features more accurately. In contrast, the "relative 6s 3s" criterion can help rule out the noisy features, but it's not precise when it comes to sorting out good data features.

Different ranking	top 10 features	top 20 features
relative 6s 3s	0.1822	0.1626
relative 2s 1s	0.1832	0.1700
absolute 2s 1s	**0.1595**	**0.1515**

Table 4.3: Different criteria for regression trajectory prediction

Different ranking	top 10 features	top 20 features
relative 6s 3s	0.4881	0.4576
relative 2s 1s	0.5262	0.5108
absolute 2s 1s	**0.4869**	**0.4527**

Table 4.4: Different criteria for regression distance prediction

4.1.3 Impact of Correlation Analysis

After using the non-dominated sorting on the different data features, the top 5 features are listed as follows:

1. Lateral velocity (sensor 1).
2. Distance to the right road edges.
3. Distance to the left road edges.
4. Lateral velocity (sensor 2).
5. Velocity to the right road edges.

We can see that lateral velocity counts as different data features twice. This is because vehicles are equipped with many sensors, and different sensors may record the same data. Therefore, correlation analysis is used to exclude such redundancy. Table 4.5 compares the model performance using the top 30 features with different correlation analysis thresholds. For example, if the threshold is 0.9, then if two data features have more than 0.9 correlation, the lower-ranked feature is excluded.

Thresholds	1.0	0.9	0.8
Regression trajectory MSE	0.1771	0.1745	**0.1467**
Regression distance MSE	0.5708	0.5180	**0.4689**
Classification F1	**0.6912**	0.6886	0.6853

Table 4.5: Performance of feature subsets with different correlation thresholds

Table 4.5 shows that when the threshold drops from 1.0 to 0.8, the performance of the regression tasks improves. This shows that in the regression problem, when we remove the redundant information among the features, more useful data features are inputted into the model, making the model perform better in the regression task.

Unlike regression tasks, the performance of classification tasks slightly worsens when the threshold drops. This is because classification tasks require fewer data features. In section

4.2.1, it is shown that the classification tasks perform better when 10 or 20 data features are inputted rather than the 30 data features. When the threshold is 1, an input of 30 features containing redundant information may be approximately equal to an input of fewer features that do not contain redundant information. Therefore, the most relevant correlated features are more effectively captured by the classification model, reducing the interference of noisy data features (for classification task) and leading to a better model performance.

4.1.4 Best Features

After applying the non-dominated sorting and correlation analysis with the 0.8 threshold, the top 10 important features are listed below:

1. Vehicle lateral velocity.
2. Distance to the right road edges.
3. Distance to the left road edges.
4. Velocity to the right road edges.
5. Velocity to the right lane markers.
6. Rate of change of acceleration to the left road edges.
7. Velocity to the left road edges.
8. Acceleration to the second left lane markers.
9. Rate of change of acceleration to the right road edges.
10. Vehicle longitudinal acceleration.

It is noticeable that the selected features cover various aspects of vehicle dynamics (lateral and longitudinal movements), interactions with road boundaries, and lane positioning, providing a holistic view of the vehicle's behavior and surroundings.

Features related to distances and velocities to road edges and lane markers are crucial for maintaining safety and stability, ensuring that the vehicle remains within safe bounds and responds appropriately to environmental changes.

4.2 Model Performance

More tests are used to select the best model parameter settings, including the number of input features, neural network architecture, and prediction horizon.

4.2.1 Number of Input Features

Table 4.6 shows the different model performances on different sets of input features. Here, 10 indicates the 10 best vehicle sensor features from the previous feature selection, 20 indicates the 20 best features, and so on. For the classification problem, the 10 best features have an average F1 score of 0.6950, the 20 best features have 0.7004, the 30 best features have 0.7031, the 40 best features have 0.6934, and the 50 best features have 0.6755. As a result, the top 30 data features are the best option for the classification task.

Input feature	10	20	30	40	50
SimpleMLP 3 blocks F1	0.6570	0.7160	0.7008	0.7069	0.7045
SimpleMLP 5 blocks F1	0.6878	0.7097	0.7178	0.6773	0.69
SimpleMLP 7 blocks F1	0.7242	0.7144	0.7036	0.7065	0.6958
SimpleMLP 9 blocks F1	0.6995	0.6947	0.7081	0.708	0.6888
ResNetMLP 3 blocks F1	0.6900	0.7008	0.7008	0.6804	0.6238
ResNetMLP 5 blocks F1	0.7010	0.6728	0.6885	0.6786	0.6634
ResNetMLP 7 blocks F1	0.6952	0.7165	0.7079	0.6866	0.6874
ResNetMLP 9 blocks F1	0.7053	0.6786	0.6974	0.7033	0.6505
Average	0.6950	0.7004	**0.7031**	0.6934	0.6755

Table 4.6: Classification performance on test dataset

For the regression trajectory problem, see Table 4.7, the 10 best features have an average loss of 0.2016, the 20 best features have 0.1991, the 30 best features have 0.1861, the 40 best features have 0.1863, and the 50 best features have 0.1866. For the regression distance problem, see Table 4.8, the 10 best features have an average loss of 0.6016, the 20 best features have 0.5956, the 30 best features have 0.5492, the 40 best features have 0.5542, and the 50 best features have 0.5464.

Input feature	10	20	30	40	50
SimpleMLP 3 blocks MSE	0.2023	0.1963	0.1946	0.1926	0.1932
SimpleMLP 5 blocks MSE	0.2023	0.2011	0.1878	0.1948	0.1931
SimpleMLP 7 blocks MSE	0.2012	0.1997	0.1948	0.1952	0.1919
SimpleMLP 9 blocks MSE	0.1999	0.2016	0.1955	0.2008	0.1932
ResNetMLP 3 blocks MSE	0.2082	0.2032	0.1791	0.1926	0.1986
ResNetMLP 5 blocks MSE	0.1996	0.1958	0.1822	0.1740	0.1639
ResNetMLP 7 blocks MSE	0.1981	0.1970	0.1783	0.1655	0.1835
ResNetMLP 9 blocks MSE	0.2014	0.1983	0.1770	0.1752	0.1759
Average	0.2016	0.1991	**0.1861**	0.1863	0.1866

Table 4.7: Regression trajectory performance on test dataset

4.2.2 Neural Network Architecture and Parameters

The performance differences between the SimpleMLP and the ResNetMLP are shown in Table 4.9. ResNetMLP behaves slightly worse than SimpleMLP on classification problems, as ResNetMLP achieves an average F1 score of 0.6987, while SimpleMLP achieves an average F1 score of 0.7075. However, ResNetMLP performs better than SimpleMLP on the two regression problems, where SimpleMLP gets an average loss of 0.1932 and 0.5599, and ResNetMLP has an average loss of 0.1791 and 0.5384, respectively.

Another trend is that the SimpleMLP model does not consistently perform better as the number of block layers increases. In contrast, the ResNetMLP model performs better and

Input feature	10	20	30	40	50
SimpleMLP 3 blocks MSE	0.5876	0.5895	0.5432	0.5698	0.5531
SimpleMLP 5 blocks MSE	0.5943	0.5923	0.5617	0.5864	0.5493
SimpleMLP 7 blocks MSE	0.5933	0.5820	0.5674	0.5836	0.5953
SimpleMLP 9 blocks MSE	0.6020	0.5984	0.5674	0.6014	0.5862
ResNetMLP 3 blocks MSE	0.6030	0.6149	0.5591	0.5287	0.5323
ResNetMLP 5 blocks MSE	0.6214	0.5983	0.5437	0.5429	0.5333
ResNetMLP 7 blocks MSE	0.6069	0.5982	0.5300	0.5075	0.5061
ResNetMLP 9 blocks MSE	0.6049	0.5911	0.5209	0.5137	0.5162
Average	0.6016	0.5956	**0.5492**	0.5542	0.5464

Table 4.8: Regression distance performance on test dataset

better on the regression tasks. Especially on the regression distance task, the loss of the ResNetMLP falls from 0.5591 to 0.5209.

Model Setting	Classification F1	Trajectory MSE	Distance MSE
SimpleMLP 3 blocks	0.7008	0.1946	0.5432
SimpleMLP 5 blocks	**0.7178**	0.1878	0.5617
SimpleMLP 7 blocks	0.7036	0.1948	0.5674
SimpleMLP 9 blocks	0.7081	0.1955	0.5674
ResNetMLP 3 blocks	0.7008	0.1791	0.5591
ResNetMLP 5 blocks	0.6885	0.1822	0.5437
ResNetMLP 7 blocks	0.7079	0.1783	0.5300
ResNetMLP 9 blocks	0.6974	**0.1770**	**0.5209**

Table 4.9: Model performance comparison between SimpleMLP and ResNetMLP for the 30 input features.

The difference between SimpleMLP and ResNetMLP is the shortcut connection in the block, which confirms its effectiveness. Shortcuts help models better propagate information from deep to shallow layers, which aligns with the current academic consensus.

4.3 Prediction Horizon

This section compares the model's performance when the prediction horizon is set to 2 and 3 seconds. Table 4.10-4.12 shows the model performance of 2- and 3-second prediction horizons. Although there is no significant difference in the classification tasks, see Table 4.10, the 2-second prediction model performs better on the regression task than the 3-second prediction, as seen in Table 4.11 and Table 4.12.

4.4 Driver-state Information

This section focuses on how driver intention information affects model performance.

Input feature	10(2s)	10(3s)		20(2s)	20(3s)		30(2s)	30(3s)
SimpleMLP 3 blocks F1	**0.6795**	0.6570		0.6897	**0.7160**		0.6897	**0.7008**
SimpleMLP 5 blocks F1	0.6769	**0.6878**		0.6972	**0.7097**		0.6729	**0.7178**
SimpleMLP 7 blocks F1	0.6757	**0.7242**		0.6717	**0.7144**		0.6752	**0.7036**
SimpleMLP 9 blocks F1	0.6758	**0.6995**		0.6619	**0.6947**		0.6489	**0.7081**
ResNetMLP 3 blocks F1	0.6875	**0.6900**		0.6930	0.7008		**0.7054**	0.7008
ResNetMLP 5 blocks F1	0.6987	**0.7010**		**0.6935**	0.6728		**0.7137**	0.6885
ResNetMLP 7 blocks F1	**0.7044**	0.6952		0.6898	**0.7165**		0.7009	**0.7079**
ResNetMLP 9 blocks F1	0.6953	**0.7053**		**0.7041**	0.6786		**0.7115**	0.6974

Table 4.10: Comparison between different prediction horizons in classification performance

Input feature	10(2s)	10(3s)		20(2s)	20(3s)		30(2s)	30(3s)
SimpleMLP 3 blocks MSE	0.2138	**0.2023**		0.2242	**0.1963**		**0.1496**	0.1946
SimpleMLP 5 blocks MSE	**0.1752**	0.2023		0.2360	**0.2011**		**0.1441**	0.1878
SimpleMLP 7 blocks MSE	**0.1799**	0.2012		**0.1756**	0.1997		**0.1529**	0.1948
SimpleMLP 9 blocks MSE	**0.1814**	0.1999		**0.1749**	0.2016		**0.1543**	0.1955
ResNetMLP 3 blocks MSE	**0.1893**	0.2082		**0.1843**	0.2032		**0.1492**	0.1791
ResNetMLP 5 blocks MSE	**0.1750**	0.1996		**0.1683**	0.1958		**0.1286**	0.1822
ResNetMLP 7 blocks MSE	**0.1760**	0.1981		**0.1716**	0.1970		**0.1217**	0.1783
ResNetMLP 9 blocks MSE	**0.1822**	0.2014		**0.1626**	0.1983		**0.1321**	0.1770

Table 4.11: Comparison between different prediction horizons in regression trajectory performance

Input feature	10(2s)	10(3s)		20(2s)	20(3s)		30(2s)	30(3s)
SimpleMLP 3 blocks MSE	**0.4971**	0.5876		**0.5127**	0.5895		**0.3834**	0.5432
SimpleMLP 5 blocks MSE	**0.5117**	0.5943		**0.4853**	0.5923		**0.4280**	0.5617
SimpleMLP 7 blocks MSE	**0.5171**	0.5933		**0.4975**	0.5820		**0.4166**	0.5674
SimpleMLP 9 blocks MSE	**0.4867**	0.6020		**0.4895**	0.5984		**0.4308**	0.5674
ResNetMLP 3 blocks MSE	**0.5169**	0.6030		**0.4821**	0.6149		**0.3769**	0.5591
ResNetMLP 5 blocks MSE	**0.4954**	0.6214		**0.4555**	0.5983		**0.3759**	0.5437
ResNetMLP 7 blocks MSE	**0.5212**	0.6069		**0.4782**	0.5982		**0.3389**	0.5300
ResNetMLP 9 blocks MSE	**0.4881**	0.6049		**0.4576**	0.5911		**0.3538**	0.5209

Table 4.12: Comparison between different prediction horizons in regression distance performance

4.4.1 Driver-state Pattern

Table 4.13 and 4.14 show the areas most viewed by drivers, 6 and 1 second before a lane change, respectively. The similarity between the two time periods is that gaze zone 1 has the highest number of gazes. This indicates that the driver's primary gaze area is the front at different times before the lane change, while the secondary area is area 0. When a lane change is about to occur, the number of gazes in zone 1 decreases significantly, while the number of gazes in zone 0 and zone 7 increases. As mentioned earlier, zones 0 and 7 represent the driver- and passenger-side mirror.

Category	Percentage
0	0.156
1	**0.718**
2	0.031
3	0.012
4	0.006
5	0.016
6	0.015
7	0.013
8	0.025
9	0.009

Table 4.13: Most probable gaze zone 6s before the lane change

Category	Percentage
0	0.173
1	**0.654**
2	0.028
3	0.011
4	0.006
5	0.015
6	0.025
7	0.038
8	0.031
9	0.020

Table 4.14: Most probable gaze zone 1 second before the lane change

4.4.2 Experiment

ResNetMLP with 9 blocks and 30 ego car inputted data features was previously proven suitable for the classification task and regression tasks for trajectory and distance prediction. To investigate whether driver-state information improves performance, the coming experiment adds driver information to the model's inputs, i.e., gaze zones and the driver forward unawareness information.

Tasks	Normal model	With gaze zone	gaze zone and unawareness
Classification F1	0.7091	**0.7121**	0.6916
Regression trajectory MSE	0.1339	**0.1280**	0.1382
Regression distance MSE	0.4377	**0.3935**	0.4223

Table 4.15: Model performance with different driver-states

As shown in Table 4.15, introducing the gaze zone improves the model performance in all tasks. This suggests that information about the driver's gaze zone is beneficial. However, when unawareness information is added, the model performs worse. In this case, driver forward unawareness harms improving model performance, so we must distinguish between valid information and noise when introducing driver-state information.

4.5 Integrated Model

The proposed integrated neural network model combines the regression and classification tasks, utilizes the data information, and reduces the number of neurons.

Tables 4.6 to 4.9 illustrate the distinct patterns in the classification and regression tasks. Increasing the number of ResNetMLP blocks from 3 to 9 leads to better performance in the regression models. For instance, the regression distance MSE decreases from 0.5591 to 0.5299. However, increasing the number of ResNetMLP blocks has minimal or even adverse impacts on the classification models, with the F1 score dropping from 0.7008 to 0.6974. Additionally, regression models benefit from using 40 or 50 inputted features compared to 10 or 20, while the classification model experiences a decline in performance when using 50 features, achieving an average F1 score of 0.6755. These comparisons suggest that regression tasks are more complex, requiring more inputted features and neural network blocks for generalization ability. In contrast, the classification model needs fewer data features and network blocks to avoid over-fitting. As a result, the regression module requires more blocks than the classification module in the integrated model.

However, if we over-compress the number of blocks in the classification models, the classification models may fail to capture complex data patterns. Thus, the core idea of the integrated model is to use the regression module to transform complex inputted features into hidden features that are easier for the classification module to interpret. These hidden features allow the classification module to express complex data features with fewer inputted features and network blocks.

By setting the number of the main block (regression module) to 9 and the number of the side block (classification module) to 1, the performance during one training process of such model is shown in Table 4.16. As shown in Table 4.16, with increased training epochs, the regression performance improves, whereas the classification performance decreases. This indicates that regression tasks are more complex and require more training epochs, while over-training can cause the classification module over-fitting, leading to performance degradation. 10 epochs seem to balance well, producing good results for the classification and regression modules.

Epoch	Regression trajectory loss	Classification F1
5	0.1536	0.7204
9	0.1488	0.7185
6	0.1382	0.7151
10	0.1363	0.7115
11	0.1258	0.7091

Table 4.16: Model performance during one training process with 9 main blocks and 1 side block

The experiment also indicates that the classification module converges very fast, thus we need to reduce its learning rate appropriately. Table 4.17 shows the results when the learning rate for the regression module is set to 1e-3, and different learning rates for the classification module are tested. It shows that a classification learning rate should be around 1e-4 to 2e-4 for good performance.

Classification learning rate	Regression trajectory loss	Classification F1
1e-3	0.1601	0.6908
5e-4	0.1337	0.6909
2e-4	**0.1288**	**0.7175**
1e-4	**0.1241**	**0.7134**
5e-5	0.1272	0.7054

Table 4.17: Best model performance for different classification learning rate

The number of learnable parameters of the 9 main block 1 side block integrated model is 137567. Its best loss for the regression task is 0.1288, and its nest F1 score for the classification task is 0.7175. This can also be compared to the number of learnable parameters for the regression 9 block model, 117315, with a loss of 0.1217, and the classification 5 block model, 82456, and an F1 score of 0.7137. Furthermore, the proposed integrated model uses only 68.86% of the computational resources to achieve what the previous two models can.

5

Discussion and Conclusion

This section first discusses the research questions posed in Chapter 1, including the findings found during the experiment and the potential reasons behind them. Then, it lists the experiment's challenges and suggests possible ways to improve them in the future. Finally, it concludes the thesis by listing the contributions.

5.1 Discussion

5.1.1 Findings

First, we use the experimental data in Chapter 4 to discuss the research questions posed in Chapter 1:

(1) How long should the prediction horizon be?

Section 4.3 investigates different prediction horizons. It tests two-second and three-second horizons. For the regression task, a prediction horizon of two seconds improves the model's performance, but for the classification task, a prediction horizon of three seconds improves the model's prediction accuracy. As the prediction horizon decreases, the uncertainty in the vehicle's trajectory is reduced. However, for the classification problem, reducing the prediction horizon makes the difference between the data of the positive and negative classes smaller, leading to a decrease in the model performance.

Overall, a 2-second prediction horizon is a good setting for both the classification and the regression tasks.

(2) Which data features are required to be input into the system?

The top 10 features selected by non-dominated sorting and correlation analysis are:

1. Vehicle lateral velocity.
2. Distance to the right road edges.
3. Distance to the left road edges.
4. Velocity to the right road edges.
5. Velocity to the right lane markers.
6. Rate of change of acceleration to the left road edges.
7. Velocity to the left road edges.
8. Acceleration to the second left lane markers.
9. Rate of change of acceleration to the right road edges.

10. Vehicle longitudinal acceleration.

From the list, we can observe that the lateral dynamics, i.e., the lateral distance, velocity, and acceleration, are more important than longitudinal dynamics, which aligns with common sense in lane-changing problems. Lateral dynamics' data features should be included in lane-changing problems.

(3) What selection criteria should be used when making feature selections?

The experiments presented in Section 4.1.2 show that the short-term change rate is more suitable for selecting the best features among good features. In addition, the absolute rate of change is better than the relative rate of change. The long-term rate of change and relative rate of change are more suitable for ruling out the worst data features.

(4) What model should be used to solve the problem?

The experiments presented in Section 4.2.1 show that the system performance increases with the number of features used, but the increase is less after 30 features. This holds for both the regression and the classification tasks. Furthermore, regarding network architecture, the performance of the SimpleMLP and the ResNetMLP are similar. In Section 4.2.2, we can see that the ResNetMLP solves the degradation problem well in the regression task by using the shortcut in the block. However, for the classification problem, ResNetMLP does not outperform the SimpleMLP. This is because the classification problem is simpler, so a more complex model and a larger number of layers cause overfitting.

(5) Does the introduction of driver-state information improve the system performance?

In Section 4.4, it is found that drivers spend most of their time looking forward and to the side mirrors, but before making the lane change, the proportion of looking forward decreases, and the proportion of looking at the mirrors on both sides rises by a small amount. Introducing the data feature "most probable gaze zone" to the neural network helps improve the system's performance. However, the "driver forward unawareness" data feature worsens the system's performance.

(6) Is it possible to combine the regression problem and the classification problem into a single problem to be solved?

Section 4.5 shows an integrated neural network model that predicts both the regression and classification tasks. The regression task is more complex and requires more neural network blocks to achieve good results, while the classification task is less complex. A classification task module can use the extracted hidden features of the regression module to reduce the number of parameters and improve model performance.

Table 4.16 shows the training process of the integrated model. The regression module improves as the number of training epochs increases, while the classification module's performance slightly declines. This illustrates that the regression task requires more fine-tuning and benefits from additional training rounds, whereas the classification task is relatively simpler. If trained for too many epochs, the classification module may suffer

from over-fitting, reducing its generalization ability. Choosing an appropriate number of training rounds can balance the performance of both tasks. In our experiment, 10 epochs prove optimal for achieving this balance.

The learning rate plays a crucial role in optimizing the model training. Specifically, a classification learning rate between 1e-4 and 2e-4, combined with a regression learning rate of 1e-3, leads to the best overall performance, as evidenced by the improvements in both regression trajectory loss and classification F1 scores. The higher learning rate in the regression module allows for quicker weight adjustments, optimizing the loss function within the given training time. Meanwhile, the lower learning rate in the classification module ensures a more stable learning process, preventing interference from the rapid updates in the regression module and avoiding over-adjustment or unstable gradient changes.

The standalone classification model requires 5 blocks to achieve an F1 Score of 0.7137. However, the classification module in the integrated model benefits from shared information with the regression module, achieving a slightly higher F1 Score of 0.7175 using only 1 block. This improvement suggests that the integrated model effectively leverages shared parameters between tasks, enhancing overall efficiency. By combining tasks within a single framework, the integrated model reduces computational resource usage to 68.86% of the learnable parameters while maintaining comparable performance.

5.1.2 Challenges

Despite the promising results, several challenges are identified during the work.

Firstly, the class imbalance in the dataset poses a challenge, as risky driving instances are only a small fraction of the total data. The data pre-processing part needs to balance the positive and negative classes so the model does not become biased towards the majority class.

Secondly, while the feature selection process effectively reduces dimension, selecting the most relevant features remains challenging. This is due to the unique characteristics of each data feature, which complicate their ranking based on a single criterion. Some features provide the model with short-term information, while others offer long-term insights. As a result, the ranking of features represents a relative order.

Thirdly, the current history depth is only 1 second, which may not capture the long-term driving patterns. History depth refers to the amount of past data inputted into the model for making predictions. In our case, we use data from the previous second to predict future states in two seconds. This limitation can affect the system's effectiveness in real-world situations.

Lastly, adding the driver-state information may introduce noise and complexity that degrade model performance. One possible way to overcome this challenge is to choose models better suited to handle complex features, such as ensemble learning models [83] and deep learning models. Moreover, feature importance analysis methods, such as feature importance scores [84], can be used to evaluate the contribution of each feature to the

model's predictions.

5.1.3 Suggestions for future Work

Building on the findings of this thesis, several directions for future work are proposed:

It is advised to apply more advanced neural network architectures to the problem to improve prediction accuracy, such as Transformer-based models [81].

Further refinement of the feature selection process through advanced machine-learning techniques can more effectively identify and rank features to reduce redundancy and enhance model accuracy.

Collecting and integrating more useful driver-state information can help improve the model's performance. For example, we can monitor the eye closure rate. Calculating the blinking frequency helps the model determine whether the driver is fatigued. The model can also monitor whether the driver's hands are on the steering wheel and analyze whether the driver's steering wheel operation is normal. This kind of information can further improve the model's robustness.

Finally, conduct real-world trials and validations with the developed model to assess its practical applicability and reliability. Collaborating with automotive companies or utilizing driving simulators can provide valuable data and insights to refine and validate the proposed method.

5.2 Conclusions

This thesis presents a comprehensive approach addressing the AES-BLIS problem. The focus is on developing an effective system to analyze the current traffic situation and predict future vehicle states to prevent collisions.

The main conclusions of this thesis can be summarized as follows: using the NSGA-II algorithm and correlation analysis for feature selection helps reduce the data dimension while maintaining prediction accuracy, and an artificial neural network model structure inspired by the ResNet model structure with shortcuts is suitable.

Furthermore, tests have been carried out to improve the model's performance on the regression and classification tasks. In addition, the "most probable gaze zone" data feature enhances the model's performance.

An integrated model is also proposed to combine the regression and classification tasks. Integrating the main block and side block models leverages the strengths of both the regression and classification tasks. The regression model excels at processing continuous features like distance, speed, and acceleration, providing robust features for the classification model, which handles discrete labels (safe/danger). This integration mitigates noisy data issues and over-fitting in classification tasks, improving overall performance by using the regression model's accurate features to enhance classification accuracy. It performs well on both tasks and reduces the number of learnable parameters.

Bibliography

[1] Toroyan, Tamitza. “Global status report on road safety.” Injury Prevention 15 (2009): 286 - 286.

[2] González-Saavedra, Juan F., Miguel Figueroa, Sandra Céspedes Umaña and Samuel Montejo Sánchez. “Survey of Cooperative Advanced Driver Assistance Systems: From a Holistic and Systemic Vision.” Sensors (Basel, Switzerland) 22 (2022): n. pag.

[3] J. Nidamanuri, C. Nibhanupudi, R. Assfalg and H. Venkataraman, "A Progressive Review: Emerging Technologies for ADAS Driven Solutions," in IEEE Transactions on Intelligent Vehicles, vol. 7, no. 2, pp. 326-341, June 2022, doi: 10.1109/TIV.2021.3122898.

[4] Scholliers, Johan, Michiel Modijefsky, Rick Janse, Mikko Tarkiainen, Anne Silla and Guus van den Born. “Study on the feasibility, costs and benefits of retrofitting advanced driver assistance to improve road safety.” (2020).

[5] Tonmoy, Achinta Brata Roy, M D Sarwar Zinan, Selim Sultan and Abir Sarker. “A comparative study on LIDAR and Ultrasonic Sensor for Obstacle Avoidance Robot Car.” 2023 International Conference on Advances in Electronics, Communication, Computing and Intelligent Information Systems (ICAECIS) (2023): 582-587.

[6] Dahl, John, Gabriel Rodrigues de Campos, Claes Olsson and Jonas Fredriksson. “Collision Avoidance: A Literature Review on Threat-Assessment Techniques.” IEEE Transactions on Intelligent Vehicles 4 (2019): 101-113.

[7] Ulbrich, Simon and Markus Maurer. “Probabilistic online POMDP decision making for lane changes in fully automated driving.” 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013) (2013): 2063-2067.

[8] Kanarachos, Stratis A. "A new method for computing optimal obstacle avoidance steering manoeuvres of vehicles." International Journal of Vehicle Autonomous Systems 7.1-2 (2009): 73-95.

[9] Yasin, Jawad N., et al. "Low-cost ultrasonic based object detection and collision avoidance method for autonomous robots." International Journal of Information Technology 13 (2021): 97-107.

[10] Hrishikesh M Thakurdesai, Jagannath V Aghav. Autonomous cars: technical challenges and a solution to blind spot[C]. Advances in Computational Intelligence and Communication Technology: Proceedings of CICT 2019, 2021, 533-547.

[11] Yiting Shen, Wei Qi Yan. Blind spot monitoring using deep learning[C]. 2018 International Conference on Image and Vision Computing New Zealand (IVCNZ), 2018, 1-5.

[12] Lee, Jonghun, Youngseok Jin, Bong-seok Kim, Seungeon Song and Sangdong Kim. “Performance Evaluation of 24GHz FMCW Radar-based Blind-spot Detection and Lane-change Assistance under Dynamic Driving Conditions in a Vehicle Proving

Ground." 2023 Fourteenth International Conference on Ubiquitous and Future Networks (ICUFN) (2023): 190-193.

[13] Guiru Liu, Mingzheng Zhou, Lulin Wang, Hai Wang, Xiansheng Guo. A blind spot detection and warning system based on millimeter wave radar for driver assistance[J]. Optik, 2017, 135: 353-365.

[14] Bing-Fei Wu, Hao-Yu Huang, Chao-Jung Chen, Ying-Han Chen, Chia-Wei Chang, Yen-Lin Chen. A vision-based blind spot warning system for daytime and nighttime driver assistance[J]. Computers & Electrical Engineering, 2013, 39(3): 846-862.

[15] Campos-Ferreira, Andres E., Jorge de Jesús Lozoya-Santos, Juan Carlos Tudón-Martínez, Ricardo Ambrocio Ramírez-Mendoza, Adriana Vargas-Martínez, Rubén Morales-Menéndez and Diego Fabián Lozano-García. "Vehicle and Driver Monitoring System Using On-Board and Remote Sensors." Sensors (Basel, Switzerland) 23 (2023): n. pag.

[16] Dahl, John, Gabriel Rodrigues de Campos and Jonas Fredriksson. "Intention-Aware Lane Keeping Assist Using Driver Gaze Information." 2023 IEEE Intelligent Vehicles Symposium (IV) (2023): 1-7.

[17] Horst, Ara van der. "A time-based analysis of road user behaviour in normal and critical encounters." (1990).

[18] Horst, Ara Van Der and Jeroen H. Hogema. "TIME-TO-COLLISION AND COLLISION AVOIDANCE SYSTEMS." (1994).

[19] Dahl, John, Rasmus Jonsson, Anton Kollmats, Gabriel Rodrigues de Campos and Jonas Fredriksson. "Automotive Safety: a Neural Network Approach for Lane Departure Detection using Real World Driving Data." 2019 IEEE Intelligent Transportation Systems Conference (ITSC) (2019): 3669-3674.

[20] "Model Predictive Control: Theory and Design." (2014).

[21] Anderson, Sterling J., Steven C. Peters, Thomas Edward Pilutti and Karl Iagnemma. "An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios." International Journal of Vehicle Autonomous Systems 8 (2010): 190-216.

[22] Hung Duy Nguyen, Dongryul Kim, Young Seop Son, Kyoungseok Han. Linear time-varying mpc-based autonomous emergency steering control for collision avoidance[J]. IEEE Transactions on Vehicular Technology, 2023.

[23] Wang, Dekun, Kaveh Nazem Tahmasebi and Dejiu Chen. "Integrated Control of Steering and Braking for Effective Collision Avoidance with Autonomous Emergency Braking in Automated Driving." 2022 30th Mediterranean Conference on Control and Automation (MED) (2022): 945-950.

[24] Rizaldi, Albert and Matthias Althoff. "Formalising Traffic Rules for Accountability of Autonomous Vehicles." 2015 IEEE 18th International Conference on Intelligent Transportation Systems (2015): 1658-1665.

[25] Ali, Mohammad, Paolo Falcone and Jonas Sjöberg. "Model-based threat assessment for lane guidance systems." Proceedings of the 2011 American Control Conference (2011): 4586-4591.

[26] Blanchini, Franco. "Set invariance in control." Automatica 35, no. 11 (1999): 1747-1767.

[27] de Campos, Gabriel R., Adam H. Runarsson, Fredrik Granum, Paolo Falcone, and Klas Alenljung. "Collision avoidance at intersections: A probabilistic threat-assessment

and decision-making system for safety interventions." In 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 649-654. IEEE, 2014.

[28] Szalay, István, Krisztián Enisz, Hunor Medve, and Dénes Fodor. "Localization accuracy improvement of autonomous vehicles using sensor fusion and extended Kalman filter." Hungarian Journal of Industry and Chemistry (2020): 109-115.

[29] Campos, Gabriel Rodrigues de, Adam H. Runarsson, Fredrik Granum, Paolo Falcone and Klas Alenljung. “Collision avoidance at intersections: A probabilistic threat-assessment and decision-making system for safety interventions.” 17th International IEEE Conference on Intelligent Transportation Systems (ITSC) (2014): 649-654.

[30] Armand, Alexandre, David Filliat and Javier Ibanez Guzman. “A Bayesian framework for preventive assistance at road intersections.” 2016 IEEE Intelligent Vehicles Symposium (IV) (2016): 1128-1134.

[31] Xiong, Xiao-xia, Long Chen and Jun Liang. “A New Framework of Vehicle Collision Prediction by Combining SVM and HMM.” IEEE Transactions on Intelligent Transportation Systems 19 (2018): 699-710.

[32] Wang, Wenshuo, Ding Zhao, Wei Han, and Junqiang Xi. "A learning-based approach for lane departure warning systems with a personalized driver model." IEEE Transactions on Vehicular Technology 67, no. 10 (2018): 9145-9157.

[33] Zyner, Alex, Stewart Worrall, and Eduardo Nebot. "Naturalistic driver intention and path prediction using recurrent neural networks." IEEE Transactions on Intelligent Transportation Systems 21, no. 4 (2019): 1584-1594.

[34] Siti Nur Atiqah Halimi, Mohd Azizi Abdul Rahman, Mohd Hatta Mohammed Ariff, Yap Hong Yeu, Nor Aziyatul Izni, Mohd Azman Abas, Syed Zaini Putra Syed Yusoff. Deep Learning Based Distance Estimation Method Using SSD and Deep ANN for Autonomous Braking/Steering[C]. International Conference on Robotics, Vision, Signal Processing and Power Applications, 2024, 581-587.

[35] Arabi, Saeed, Anuj Sharma, Michelle Reyes, Cara Hamann, and Corinne Peek-Asa. "Farm vehicle following distance estimation using deep learning and monocular camera images." Sensors 22, no. 7 (2022): 2736.

[36] Fonder, Michaël, Damien Ernst, and Marc Van Droogenbroeck. "M4Depth: Monocular depth estimation for autonomous vehicles in unseen environments." arXiv preprint arXiv:2105.09847 (2021).

[37] Shao, Hao, Letian Wang, Ruobing Chen, Steven L. Waslander, Hongsheng Li and Y. Liu. “ReasonNet: End-to-End Driving with Temporal and Global Reasoning.” 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2023): 13723-13733.

[38] Liu, Ze, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin and Baining Guo. “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows.” 2021 IEEE/CVF International Conference on Computer Vision (ICCV) (2021): 9992-10002.

[39] Nidamarthi Srinivas, Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms[J]. Evolutionary Computation, 1994, 2(3): 221-248.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep residual learning for image recognition[C]. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, 770–778.

[41] Johan Andersson. A survey of multiobjective optimization in engineering design[J]. Department of Mechanical Engineering, Linköping University. Sweden, 2000.

[42] Vilfredo Pareto. Cours d'économie politique[V]. Librairie Droz, 1964.

[43] J. Brownlee, "Rectified Linear Activation Function for Deep Learning Neural Networks," Machine Learning Mastery, 2021. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[44] Frederic B Fitch. Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity[J]. The Journal of Symbolic Logic, 1944, 9(2): 49–50.

[45] Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017, 60(6): 84–90.

[46] J. Brownlee, "How to Implement GAN Hacks in Keras to Train Stable Models," Machine Learning Mastery, May 2021.

[47] Priyanga, G.S., Pransu, G., Krishna, H., Thomas, T. (2023). Discovery of Novel Photocatalysts Using Machine Learning Approach. In: Joshi, N., Kushvaha, V., Madhushri, P. (eds) Machine Learning for Advanced Functional Materials. Springer, Singapore.

[48] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain[J]. Psychological Review, 1958, 65(6): 386.

[49] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams. Learning representations by back-propagating errors[J]. Nature, 1986, 323(6088): 533–536.

[50] Vinod Nair, Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines[C]. Proceedings of the 27th international conference on machine learning (ICML-10), 2010, 807–814.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification[C]. Proceedings of the IEEE international conference on computer vision, 2015, 1026–1034.

[52] Robbins, Herbert E.. "A Stochastic Approximation Method." Annals of Mathematical Statistics 22 (1951): 400-407.

[53] Kingma, Diederik P. and Jimmy Ba. "Adam: A Method for Stochastic Optimization." CoRR abs/1412.6980 (2014): n. pag.

[54] Dauphin, Yann, Harm de Vries and Yoshua Bengio. "RMSProp and equilibrated adaptive learning rates for non-convex optimization." ArXiv abs/1502.04390 (2015): n. pag.

[55] Duchi, John C., Elad Hazan and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." J. Mach. Learn. Res. 12 (2011): 2121-2159.

[56] Tibshirani, Robert. "Regression Shrinkage and Selection via the Lasso." Journal of the royal statistical society series b-methodological 58 (1996): 267-288.

[57] Ioffe, Sergey and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." ArXiv abs/1502.03167 (2015): n. pag.

[58] Chawla, N., K. Bowyer, Lawrence O. Hall and W. Philip Kegelmeyer. "SMOTE: Synthetic Minority Over-sampling Technique." ArXiv abs/1106.1813 (2002): n. pag.

[59] Elman, Jeffrey L.. "Finding Structure in Time." Cogn. Sci. 14 (1990): 179-211.

[60] Hochreiter, Sepp and Jürgen Schmidhuber. “Long Short-Term Memory.” Neural Computation 9 (1997): 1735-1780.

[61] Guyon, Isabelle M and André Elisseeff. “An Introduction to Variable and Feature Selection.” J. Mach. Learn. Res. 3 (2003): 1157-1182.

[62] Jinxin Liu, Yugong Luo, Hui Xiong, Tinghan Wang, Heye Huang, Zhihua Zhong. An Integrated Approach to Probabilistic Vehicle Trajectory Prediction via Driver Characteristic and Intention Estimation[C]. 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2019, 3526-3532.

[63] Zixu Hao, Xing Huang, Kaige Wang, Maoyuan Cui, Yantao Tian. Attention-Based GRU for Driver Intention Recognition and Vehicle Trajectory Prediction[C]. 2020 4th CAA International Conference on Vehicular Control and Intelligence (CVCI), 2020, 86-91.

[64] Koen Vellenga, H. Joe Steinhauer, Alexander Karlsson, Göran Falkman, A. Pehlivan Rhodin, Ashok Chaitanya Koppisetty. Driver Intention Recognition: State-of-the-Art Review[J]. IEEE Open Journal of Intelligent Transportation Systems, 2022, 3: 602-616.

[65] Shilpa Gite, Himanshu Agrawal, Ketan Kotecha. Early anticipation of driver's maneuver in semiautonomous vehicles using deep learning[J]. Progress in Artificial Intelligence, 2019: 1-13.

[66] Abdellatif Moussaid, Ismail Berrada, Mohamed El-Kamili, Khalid Fardousse. Predicting Driver Lane Change Maneuvers Using Driver's Face[C]. 2019 International Conference on Wireless Networks and Mobile Communications (WINCOM), 2019, 1-7.

[67] Ashesh Jain, Hema Swetha Koppula, Bharad Raghavan, Shane Soh, Ashutosh Saxena. Car that Knows Before You Do: Anticipating Maneuvers via Learning Temporal Driving Models[C]. 2015 IEEE International Conference on Computer Vision (ICCV), 2015, 3182-3190.

[68] Mahdi Bonyani, Mina Rahmanian, Simindokht Jahangard, Mahdi Rezaei. Dipnet: Driver Intention Prediction for a Safe Takeover Transition in Automated Vehicles[J]. SSRN Electronic Journal, 2021.

[69] Mark Hamilton, Zhoutong Zhang, Bharath Hariharan, Noah Snavely, William T. Freeman. Unsupervised Semantic Segmentation by Distilling Feature Correspondences[J]. ArXiv, 2022, abs/2203.08414.

[70] Yiming Huang, Ao Jia, Xiaodan Zhang, Jiawei Zhang. Generic Attention-model Explainability by Weighted Relevance Accumulation[J]. ArXiv, 2023, abs/2308.10240.

[71] Boyu Chen, Peixia Li, Baopu Li, Chuming Li, Lei Bai, Chen Lin, Ming Sun, Junjie Yan, Wanli Ouyang. PSViT: Better Vision Transformer via Token Pooling and Attention Sharing[J]. ArXiv, 2021, abs/2108.03428.

[72] Xiaohan Ding, X. Zhang, Yi Zhou, Jungong Han, Guiguang Ding, Jian Sun. Scaling Up Your Kernels to 31×31: Revisiting Large Kernel Design in CNNs[C]. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, 11953-11965.

[73] Yanjun Huang, Jiatong Du, Ziru Yang, Zewei Zhou, Lin Zhang, Hong Chen. A Survey on Trajectory-Prediction Methods for Autonomous Driving[J]. IEEE Transactions on Intelligent Vehicles, 2022, 7: 652-674.

[74] Ruihua Wang, et al. An improved nondominated sorting genetic algorithm for multi-objective problem[J]. Mathematical Problems in Engineering, 2016.

[75] Haoran Song, Di Luan, Wenchao Ding, Michael Yu Wang, Qifeng Chen. Learning to Predict Vehicle Trajectories with Model-based Planning[J]. ArXiv, 2021, abs/2103.04027.
[76] Alex Kuefler, Jeremy Morton, Timothy A. Wheeler, Mykel J. Kochenderfer. Imitating driver behavior with generative adversarial networks[C]. 2017 IEEE Intelligent Vehicles Symposium (IV), 2017, 204-211.
[77] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, Cordelia Schmid. VectorNet: Encoding HD Maps and Agent Dynamics From Vectorized Representation[C]. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, 11522-11530.
[78] John Dahl, Gabriel Rodrigues de Campos, Jonas Fredriksson. Performance and Efficiency Analysis of a Linear Learning-Based Prediction Model Used for Unintended Lane-Departure Detection[J]. IEEE Transactions on Intelligent Transportation Systems, 2022, 23: 9115-9125.
[79] Yunhao Zhang, Junchi Yan. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting[C]. International Conference on Learning Representations, 2023.
[80] Lin, Tsung-Yi, Priya Goyal, Ross B. Girshick, Kaiming He and Piotr Dollár. "Focal Loss for Dense Object Detection." 2017 IEEE International Conference on Computer Vision (ICCV) (2017): 2999-3007.
[81] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin. Attention is all you need[C]. Advances in Neural Information Processing Systems, 2017: 5998–6008.
[82] Ze Liu, Yutong Lin, Yixuan Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows[J]. arXiv preprint arXiv:2103.14030, 2021.
[83] Mienye, Ibomoiye Domor and Yanxia Sun. "A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects." IEEE Access 10 (2022): 99129-99149.
[84] Little, Camille Olivia, Debolina Halder Lina and Genevera I. Allen. "Fair Feature Importance Scores for Interpreting Tree-Based Methods and Surrogates." ArXiv abs/2310.04352 (2023): n. pag.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY