

Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation

Doctoral Program in Electrical, Electronics and Communications Engineering

(36^{th}cycle)

Machine Learning-Driven Trajectory Prediction Models and Motion Control Strategies for Connected Automated Vehicles with Human Adaptation

By

Dinesh Cyril Selvaraj

******

Supervisor(s):

Prof. Carla Fabiana Chiasserini

Doctoral Examination Committee:

Prof. Claudia Campolo, Università Mediterranea di Reggio Calabria

Prof. Michele Segata, Università degli studi di Trento

Prof. Renato Lo Cigno, Università di Brescia

Prof. Maria Luisa Merani, Università degli Studi di Modena e Reggio Emilia

Prof. Fabio Dovis, Politecnico di Torino

Politecnico di Torino

2024

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Dinesh Cyril Selvaraj
2024

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Acknowledgements

I would like to express my sincere gratitude to everyone who supported me throughout this journey.

First and foremost, I want to thank my advisor, Prof. Carla Fabiana Chiasserini, for her support, guidance, and insightful feedback throughout this journey. Her advice and mentorship have been instrumental in shaping this work. I have always admired her dedication and professionalism, which have been an inspiration for me to follow. I would also like to extend my gratitude to Prof. Nicola Amati and Prof. Francesco Paolo Deflorio for their support and feedback during our weekly meetings. Additionally, I am grateful to Prof. Falko Dressler for providing me with the opportunity to conduct research at the Technical University of Berlin; it was a great experience for me.

I would like to thank Dr. Christian Vitale for his valuable input, constructive criticism, and advice over the years. I believe I could write a book based on our Skype chats. Our discussions have enriched my research experience and provided me with a broader perspective on my work. I owe special thanks to Dr. Shailesh Hegde for being there for me, sharing late-night discussions, and staying late at the office with me. Without you, it wouldn't have been possible to reach this stage. Thank you for all the advice, both personal and professional, that you have shared with me.

Next, I want to thank my friends and colleagues at Politecnico di Torino for providing memorable experiences and making me feel at home.

On a personal note, I am deeply grateful to my parents, cousins, and family for their encouragement and endless support. Thank you for your constant presence and support.

Thank you all for being part of this journey.

PhD Activity

Journal publications

- **Selvaraj, D. C.**; Vitale, C.; Panayiotou, T.; Kolios, P.; Chiasserini, C. F.; Ellinas, G., Edge-Assisted ML-Aided Uncertainty-Aware Vehicle Collision Avoidance At Urban Intersections, in: *IEEE Transactions on Intelligent Vehicles*, 2023
- **Selvaraj, D. C.**; Hegde, S.; Amati, N.; Deflorio, F.; Chiasserini, C. F., An ML-Aided Reinforcement Learning Approach for Challenging Vehicle Maneuvers, in: *IEEE Transactions on Intelligent Vehicles*, 2023
- **Selvaraj, D. C.**; Hegde, S.; Amati, N.; Deflorio, F.; Chiasserini, C. F., A Deep Reinforcement Learning Approach for Efficient, Safe and Comfortable Driving, in: *Applied Sciences*, 2023

Conference publications

- **Selvaraj, D. C.**; Hegde, S.; Chiasserini, C. F.; Amati, N.; Deflorio, F.; Zennaro, G., A Full-fledge Simulation Framework for the Assessment of Connected Cars, in: *Transportation Research Procedia*, 2021
- **Selvaraj, D. C.**; Vitale, C.; Panayiotou, T.; Kolios, P.; Chiasserini, C. F.; Ellinas, G., Edge Learning of Vehicular Trajectories at Regulated Intersections, in: *IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, 2021
- **Selvaraj, D. C.**; Dressler, F.; Chiasserini, C. F. (2023), Human-Centered Traffic Management Supporting Smart Cities and the Metaverse, in: *IEEE*

International Conference on Metaverse Computing, Networking and Applications (MetaCom), 2023

Attended classes and passed tests

Name	Hours
Big data processing and programming	20
Connected vehicles	20
Data mining concepts and algorithms	20
Advanced scientific programming in matlab	30
Advanced techniques for optimization	20
Advanced topics in energy storage system and electric vehicle drivetrain design	20
Statistical data processing	20
Entrepreneurship and start-up creation	40
Time management	2
Writing scientific papers in english	15

External training activity

Name	Location	Hours
AI4NETS –AI/ML for data communication Networks	Politecnico di Milano	3

Abstract

Vehicles are currently undergoing a digital transformation, increasingly equipped with advanced sensing devices, electronic control units, and communication interfaces that enhance their capabilities. The richness of information generated by these systems fosters the development of data-driven approaches that facilitate context-aware decision-making. The thesis focuses on developing data-driven frameworks in two key research directions: vehicle trajectory prediction and vehicle motion control. The thesis aims to enhance safety, traffic efficiency, comfort, and the overall driving experience through the implementation of these frameworks.

Initially, for vehicle trajectory prediction, a machine learning-aided, uncertainty-aware framework is introduced to detect collisions in complex urban environments. The proposed framework leverages Long Short-Term Memory (LSTM)-based models to predict vehicles' future trajectories and estimate the uncertainties associated with them. Subsequently, a random forest classifier leveraging both the trajectory predictions and their associated uncertainties is employed to detect collisions preemptively. The framework was evaluated using mobility traces from two urban intersections, and the results demonstrated that it could detect all collisions at the intersections preemptively, unlike its alternatives. In addition, it has fewer false positives, emphasizing the framework's effectiveness. The use of prediction intervals along with the predicted trajectories led to a 61% improvement in the median reaction time available for drivers. Given their timely detection, automated vehicles can avoid all collisions at both intersections.

Subsequently, for vehicle motion control, two Deep Reinforcement Learning (DRL)-based frameworks have been introduced: 2-Layer Learning Cooperative Adaptive Cruise Control (2LL-CACC) and Adaptive Autopilot. The 2LL-CACC framework utilizes a two-layer learning strategy designed to tackle challenging maneuvers with the objective of enhancing safety, comfort, and traffic efficiency. The

top layer hosts a context recognition model to appropriately weigh the target metrics such as headway, jerk, and longitudinal wheel slip. The bottom layer leverages a DRL-based control strategy that optimizes vehicle acceleration to achieve the objectives. To evaluate this approach, CoMoVe (Communication, Mobility, and Vehicle dynamics), a virtual validation framework, is employed to realistically simulate the challenging cut-in and cut-out scenarios using domain-specific simulators. The results show that the framework achieved a better trade-off among the objectives and outperformed its counterparts. In terms of traffic efficiency, the approach maintained the desired headway for 73% of the simulation time on average across two scenarios, while its counterparts achieved only 30%. Additionally, SCALEXIO AutoBox is used for rapid control prototyping to experimentally validate the simulation results.

Modern vehicles, despite being equipped with Advanced Driver Assistance Systems (ADAS), often encounter their ADAS functionalities disengaged by drivers due to their limitations in adapting to driver preferences. To tackle the disengagement rate, the adaptive autopilot framework focuses on achieving human-like driving behavior by tailoring control actions according to the driver's preferred driving style. It addresses three interconnected sub-problems in the car-following scenario. Initially, a rule-based (RB) approach categorizes driving styles into three types: aggressive, normal, and conservative. Subsequently, deep neural network-based regressor models are trained on this categorized data to predict human driver-like acceleration values for each driving style. Finally, a Constrained Deep Reinforcement Learning (C-DRL) approach is trained to mimic human-like driving behavior by controlling the vehicle's acceleration. The C-DRL agent is trained to reduce the difference between its action and the predicted human driver-like acceleration, while also guaranteeing safety. Performance assessments show that each sub-problem has achieved its respective objectives. The RB classifier effectively categorizes drivers' driving styles, while the regressor models predict human driver-like accelerations, with 80% of the absolute errors falling below 0.21 m/s^2, outperforming its alternatives. Following that, C-DRL agents can safely mimic human-like driving across all styles, thereby enhancing the driving experience and reducing the disengagement rate of ADAS functionalities.

Contents

Nomenclature **xi**

1 Introduction **1**

1.1 Research Questions and Main Contributions 3

1.1.1 Vehicle Trajectory Prediction 4

1.1.2 Vehicle Motion Control 6

1.2 Outline 8

2 Edge-assisted ML-aided Uncertainty-aware Vehicle Collision Avoidance at Urban Intersections **10**

2.1 Introduction 11

2.2 Related Work 13

2.2.1 Vehicle Trajectory Prediction 14

2.2.2 Collision Detection/Avoidance 15

2.2.3 Novel Contribution 17

2.3 Network-Assisted Collision Detection: Overview of our Framework 18

2.4 Vehicle Trajectory Prediction and Uncertainty in Deep LSTM-ED . 20

2.4.1 Least Squares Regression for Trajectory Prediction: Estimating Mean Response Values 22

2.4.2 Deep Quantile Regression for Trajectory Prediction: Estimating Lower and Upper Response Values 22

2.4.3 Dataset Formulation 24
2.4.4 The LSTM-ED Model 26
2.5 An ML-aided Uncertainty-aware Collision Detection Mechanism . . 29
2.5.1 RFC's Inputs 30
2.6 Avoiding Collisions at Urban Intersections 32
2.7 Performance Evaluation 34
2.7.1 Data Collection and Pre-processing 34
2.7.2 Performance Results - Trajectory Prediction 37
2.7.3 Performance Results - Trajectory Uncertainty Estimation . . 41
2.7.4 Performance Results - Collision Detection 42
2.7.5 Performance Results - Collision Avoidance 51
2.8 Chapter Conclusion 52

3 An ML-aided Reinforcement Learning Approach for Challenging Vehicle Maneuvers 53
3.1 Introduction 54
3.2 Related Work 57
3.3 The 2LL-CACC Framework 60
3.3.1 Top Layer: Context Recognition Model 63
3.3.2 Bottom Layer: The DRL Model 66
3.4 Integrating the DRL Model in CoMoVe 75
3.5 Hardware-in-the-loop Implementation 76
3.6 Performance Evaluation 79
3.6.1 Reference Scenario and Test Cases 80
3.6.2 Results 81
3.7 Chapter Conclusion 87

4 Adaptive Autopilot: Constrained DRL for Diverse Driving Behaviors 88

4.1 Introduction 89
4.2 Related Work 91
4.3 Adaptive Autopilot Framework 92
4.3.1 Rule-based Classifier 93
4.3.2 Human-like Action Predictor 97
4.3.3 C-DRL Framework 99
4.4 Performance Evaluations 103
4.4.1 Dataset 103
4.4.2 Performance Results - Rule-based Classifier 103
4.4.3 Performance Results - Acceleration prediction 104
4.4.4 Performance Results - Human-like driving 108
4.5 Chapter Conclusion 118

5 Conclusions **119**
5.1 Future Works 122

References **124**

Nomenclature

Acronyms / Abbreviations

2LL-CACC 2-Layer Learning Cooperative ACC

AA Adaptive Autopilot

ACC Adaptive Cruise Control

ADAS Advanced Driver Assistance Systems

C-DRL Constrained Deep Reinforcement Learning

C-ITS Cooperative Intelligent Transportation System

C-RL Constrained Reinforcement Learning

CACC Cooperative Adaptive Cruise Control

CAM Cooperative Awareness Message

CAN Controller Area Network

CAV Connected Autonomous Vehicles

CF Car-Following

CI-CWS Cooperative Intersection - Collision Warning System

CoMoVe Communication Mobility Vehicle dynamics

CV Connected Vehicle

DDPG Deep Deterministic Policy Gradient

DDQN Double Deep Q Network

DENM Decentralized Environmental Notification Message

DNN Deep Neural Network

DRL Deep Reinforcement Learning

ED Euclidean Distance

GI Gini Index

HIL Hardware-In-the-Loop

I2I Infrastructure-to-Infrastructure

IDM Intelligent Driver Model

IM Intersection Manager

ITS Intelligent Transportation System

KF Kalman Filter

LSTM Long Short-Term Memory

LSTM-ED Long Short-Term Memory-Encoder Decoder

LV Lead Vehicle

MAE Mean Absolute Error

MDP Markov Decision Process

MEC Mobile Edge Computing

ML Machine Learning

MPC Model Predictive Control

MSBE Mean-Squared Bellman Error

OBU On-Board Unit

ONNX Open Neural Network Exchange

PD Proportional-Derivative

RFC Random Forest Classifier

RL Reinforcement Learning

RMSE Root Mean Square Error

RNN Recurrent Neural Network

RTI Real-Time Interface

SAC Soft Actor-Critic

TTC Time-To-Collision

V2I Vehicle-to-Infrastructure

V2V Vehicle-to-Vehicle

V2X Vehicle-to-Everything

WHO World Health Organization

Chapter 1

Introduction

In recent decades, over half of the global population has moved to urban areas, a proportion projected to rise to two-thirds by 2050 [1]. Such rapid urbanization significantly impacts several key domains, such as transportation and healthcare, which may struggle to manage the increase in urban populations. The transportation domain becomes particularly vital in addressing the complex challenges of urbanization. Traffic congestion, fuel consumption, and commute times all significantly impact public health, the environment, and the economy [2, 3]. The use of technological solutions to address the challenges posed by rapid urbanization in the transportation domain is collectively represented as Intelligent Transportation Systems (ITS). ITS encompasses multiple applications related to traffic management, such as adaptive traffic signals, enhancements to public transportation like smart ticketing systems, and safety improvements like collision avoidance systems. Furthermore, cooperative ITS solutions have been proposed [4], facilitating communication and real-time exchange of data among vehicles, infrastructure, and other road traffic actors. It fosters the development of data-centric approaches that aid in the advancement of automated vehicles, aiming to improve traffic efficiency, safety, fuel economy, and reduction of carbon footprint [5]. With the number of vehicles on the road anticipated to exceed 2 billion in the upcoming decades [6], alongside strict regulations on road safety and emissions [7, 8], the importance of data-driven solutions in the automotive industry is becoming ever more crucial.

The automotive industry is currently undergoing a digital transition aimed at enhancing vehicle capabilities by equipping them with sensing devices, electronic

control units, and driver assistance algorithms. As a result, vehicles are becoming sophisticated systems that extend their functionality beyond mere modes of transportation [9]. With the help of the information collected through sensors and communication frameworks, vehicles can digitally recreate their surrounding environment. This helps them to make informed decisions using data-driven approaches that encompass perception, planning, and vehicle control [10]. To standardize the automation of vehicle functionalities, organizations such as SAE have categorized different levels of vehicle automation [11]. Currently, there is an increasing trend of equipping vehicles with Level 2 automation features, incorporating Advanced Driver Assistance Systems (ADAS) such as parking assist, Adaptive Cruise Control (ACC), lane departure warning, and blind spot monitoring. This thesis focuses on addressing challenges associated with ADAS functionalities through machine learning-based frameworks aimed at enhancing safety, traffic efficiency, comfort, and the driving experience. In this context, a state-of-the-art literature survey reveals several research directions aimed at advancing vehicle capabilities and achieving relevant objectives [10]. Among them, two key research directions emerge as significant: vehicle movement prediction and vehicle motion control. Specifically, this thesis concentrates on addressing the challenges associated with these areas through data-driven frameworks, namely: (i) vehicle trajectory prediction and (ii) vehicle motion control.

Vehicle trajectory prediction is a critical aspect of ADAS and autonomous driving, involving predicting the vehicles' future trajectories using current and historical data. This becomes particularly challenging in urban areas, where a multitude of traffic actors like pedestrians, cyclists, vehicles and factors like traffic signals must be considered. The use of C-ITS technologies enables real-time data exchange between traffic actors and infrastructures, providing a holistic view of the environment and enhancing prediction accuracy. Accurate trajectory prediction is crucial for improving road safety, as these predictions can be used to foresee potential collisions.

Vehicle motion control encompasses control strategies that regulate vehicle movements, including both longitudinal (brake and acceleration commands) and lateral (steering commands). These control strategies rely on data from sensors and Vehicle-to-Everything (V2X) communication frameworks, which encompass information related to the surrounding environment and vehicle dynamics,

to make decisions regarding speed, acceleration, and steering control. The objective of these strategies is to efficiently control vehicle movements to guarantee safety, stability, comfort, and the desired driving experience.

In this context, Machine Learning (ML) techniques have proven to be highly effective in developing context-aware, data-driven strategies. These techniques enable the creation of models that use complex, multivariate data to make accurate predictions, as well as the formulation of control algorithms that can learn the intricate patterns of vehicle movements in a dynamic driving environment. In this thesis, we will investigate different ML techniques, examining their application in both vehicle trajectory prediction and vehicle motion control. We will also explore how communication between vehicles and infrastructure contributes to enhancing safety, efficiency, and comfort. Furthermore, we will explore the advantages of hosting safety-critical services, such as collision avoidance applications, on Mobile Edge Computing (MEC) servers, which require real-time data processing and low latency response times. We will also explore a safety-constrained machine learning technique that tailors control strategies to accommodate various driving styles, addressing drivers' preferences while maintaining a focus on safety. Incorporating driver preferences into control strategies is vital because drivers tend to disengage automated systems that fail to align with their driving styles, diminishing the advantages provided by such systems. Finally, we will demonstrate the benefits of employing data-driven approaches in ensuring safety, improving traffic efficiency, and providing a desired driving experience.

The remainder of this chapter will first present the research questions that have driven our research activity and the main contributions of this work, then provide a detailed outline of the thesis.

1.1 Research Questions and Main Contributions

The main contribution of this thesis is the design, development, and evaluation of ML-based data-driven strategies aimed at addressing the challenges associated with traditional algorithms, thereby enhancing safety, comfort, and traffic efficiency. The thesis has two research directions: vehicle trajectory prediction and vehicle motion control. In the first case, we use MEC servers located at urban intersections to

collect data from vehicles in the surrounding area, predict their trajectories with high accuracy, and estimate the uncertainty around these predictions. Furthermore, the MEC server hosts a collision detection algorithm that analyzes the vehicles' predicted positions to determine if they are on a collision path, taking into account the uncertainty region around the predicted positions to improve the accuracy and detection time of potential collisions. Conversely, in the second case, we use the data obtained from the sensor array and communication frameworks to control the ego vehicle movements, i.e., the vehicle hosting the control strategy, with respect to the lead vehicle to maintain a safe distance between them while ensuring passenger comfort and vehicle stability. Additionally, we have explored ML techniques that tailor vehicle control strategies according to driver preferences. We demonstrated that the proposed approach can safely emulate human-like control actions under various driving styles, ensuring an adaptive approach that accommodates different styles while prioritizing safety. In the first case, we monitor the vehicle movements, predict their trajectories, analyze the predicted data, and take control only if the vehicle is going to collide—a reactive measure. In the second case, we proactively control the vehicle movement to ensure a safe distance between the vehicles and address challenging vehicle maneuvers. Both research directions fall under the broader goal of advancing vehicle capabilities, contributing to the inception of autonomous vehicles.

In the subsequent sections, we will discuss these two research directions in detail, highlighting the research questions and outlining our contributions towards addressing them. Additionally, we will present information regarding our dissemination activities.

1.1.1 Vehicle Trajectory Prediction

Trajectory prediction is essential for both human-driven and autonomous vehicles. For human-driven vehicles, it plays a crucial role in safety-related applications such as collision detection and avoidance. Conversely, for autonomous vehicles, it assists in predicting the movements of surrounding vehicles and doing path planning accordingly. However, trajectory prediction in a highway environment, characterized by straight stretches of road, is relatively less complex compared to urban areas, especially at intersections without traffic lights. In this context, developing a trajectory

prediction model that accounts for the complexities of urban areas becomes critical. We address the following main research question (RQ) in the first research direction:

RQ1: How to provide enhanced safety related services to vehicles in urban environments?

Specifically, how to ensure the accuracy of the predicted trajectories required for reliable safety service applications? What advantages do ML approaches offer over traditional algorithms? What is the performance of the proposed system in a safety-critical application such as collision avoidance?

To address RQ1, we design and develop an ML-aided, uncertainty-aware framework that utilizes Recurrent Neural Network (RNN)-based models to predict both vehicle trajectories and their associated uncertainty estimations. This framework enables preemptive collision detection in complex urban environments, where, even if predicted trajectories do not indicate a collision, the uncertainty associated with them will trigger the alarms if they fall within a collision range. We compare the performance of our proposed framework with state-of-the-art approaches and demonstrate that our ML-based approach outperforms traditional algorithms. Furthermore, evaluation with real-world data shows that our framework achieves better performance in key metrics such as trajectory prediction accuracy, preemptive collision detection, and false positive rate.

This part is covered in detail in Chapter 2. Our contributions to international conferences and journals can be found in:

1. **Selvaraj, D. C.**; Vitale, C.; Panayiotou, T.; Kolios, P.; Chiasserini, C. F.; Ellinas, G., Edge Learning of Vehicular Trajectories at Regulated Intersections, in: *IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, 2021

2. **Selvaraj, D. C.**; Vitale, C.; Panayiotou, T.; Kolios, P.; Chiasserini, C. F.; Ellinas, G., Edge-Assisted ML-Aided Uncertainty-Aware Vehicle Collision Avoidance At Urban Intersections, in: *IEEE Transactions on Intelligent Vehicles*, 2023

1.1.2 Vehicle Motion Control

The second part of the research work focuses on vehicle motion control. This part aims to proactively control the vehicle movements according to the surrounding environment, prioritizing safety, traffic efficiency, comfort, and driving experience. In this context, we introduce two frameworks: 2-Layer Learning Cooperative Adaptive Cruise Control (2LL-CACC) and Adaptive Autopilot (AA), both employing reinforcement learning-based techniques to control vehicle movements. Using these frameworks, we address research questions critical to improving vehicle motion control.

RQ2: How to control the vehicle in a efficient way under various environmental conditions and handle challenging maneuvers in a safe, efficient and comfortable manner?

In particular, how can communication technologies be leveraged to handle challenging maneuvers like cut-ins and cut-outs? How can various reward components, each associated with distinct objectives, be appropriately weighed? How to accurately assess the performance and validate the deployment capability of the data-driven control strategies?

To address RQ2, we introduce an ML-aided Deep Reinforcement Learning (DRL) framework, named 2LL-CACC, that monitors the surrounding vehicle movements and effectively controls the ego vehicle to ensure safety, traffic efficiency, and comfort. This framework employs a two-layer approach: the top layer hosts an ML model to determine the weights of each reward component based on the current context, while the lower layer features a DRL agent that learns the optimal policy for vehicle control. To tackle challenging scenarios, we integrate data from sensors and communication interfaces to monitor vehicle movements, which is vital, especially if the vehicle falls into the onboard sensors' blind spot. Furthermore, we employ a novel simulation framework for virtually validating the proposed approach through domain-specific simulators, which realistically account for all relevant aspects such as communication, vehicle dynamics, and traffic mobility. Furthermore, we validate these control strategies through rapid control prototyping with dSPACE SCALEXIO AutoBox using auto-code generation. It demonstrates the feasibility of implementing the proposed strategies in real-time vehicle control units.

Chapter 3 covers this part in detail and our contributions on this work can be found in:

1. **Selvaraj, D. C.**; Hegde, S.; Amati, N.; Deflorio, F.; Chiasserini, C. F., An ML-Aided Reinforcement Learning Approach for Challenging Vehicle Maneuvers, in: *IEEE Transactions on Intelligent Vehicles*, 2023

2. **Selvaraj, D. C.**; Hegde, S.; Amati, N.; Deflorio, F.; Chiasserini, C. F., A Deep Reinforcement Learning Approach for Efficient, Safe and Comfortable Driving, in: *Applied Sciences*, 2023

RQ3: How to develop a framework that is capable of adjusting the control strategy to accommodate various driving styles, while ensuring safety?

In particular, can an ML framework develop a control strategy that mimics human-like driving behavior without compromising safety constraints? Additionally, what techniques can be employed to decrease the disengagement rate of automated systems and tailor the vehicle control actions to align with the drivers preferred driving style?

The answer to RQ3 is the AA framework, which leverages a constrained DRL (C-DRL) technique to safely mimic human-like driving behavior by tailoring vehicle control actions to accommodate various driving styles. The framework uses regressor models trained on real-world driving traces to predict human-like control actions that encompass diverse driving styles. Subsequently, the framework aims to minimize the difference between the control actions taken by C-DRL agents and regressor models. Additionally, during training, the framework enforces safety constraints relating to the distance between vehicles. We show that the regressor models outperform traditional approaches in predicting human-like acceleration values, while the C-DRL agents can safely emulate human-like driving behavior tailored to various driving styles. By safely imitating the human driver, the framework increases drivers' satisfaction, potentially reducing the disengagement rates of the autopilot system.

Chapter 4 presents the detailed analysis and evaluation of the AA framework and the work is submitted to the *IEEE Intelligent Transportation Systems Conference (IEEE ITSC 2024)*.

1.2 Outline

This thesis is structured into five chapters, including this chapter on the introduction. The rest of the thesis is organized as follows.

Chapter 2 presents a network-assisted, ML-aided, uncertainty-aware framework designed to preemptively detect collisions between vehicles at urban intersections. Long Short-Term Memory (LSTM) models are employed to predict the vehicles' trajectories and to estimate the uncertainties associated with these predictions. Subsequently, it employs an ensemble approach, especially a random forest classifier-based model, to detect potential vehicle collisions by considering both the trajectory predictions and their associated uncertainties. Once a collision is detected, the Intersection Manager (IM) triggers an alarm to alert the drivers to take evasive actions. The framework is assessed using real-world data, where it is able to effectively detect collisions with few false positives and provide sufficient time for the drivers to react. Additionally, the results demonstrate that combining trajectory predictions and their uncertainties not only reduces false positives but also increases the available reaction time.

Chapter 3 proposes the 2LL-CACC framework, leveraging deep reinforcement learning techniques to develop a data-driven cooperative ACC application. This framework aims to control vehicles during challenging maneuvers, such as cut-ins and cut-outs, with a focus on optimizing traffic flow, passenger comfort, and vehicle stability. It comprises two layers: the top layer hosts the context recognition model that appropriately weighs all relevant factors by considering the current contextual information, while the bottom layer employs an actor-critic-based DRL approach to learn the optimal policy for controlling the vehicles. The performance of the proposed approach is evaluated using two road traffic scenarios that are designed to simulate the challenging cut-in and cut-out conditions. The quantitative results demonstrate that the proposed approach could achieve a better trade-off between the objectives when compared to its counterparts. Finally, the hardware-in-the-loop validation confirms the framework's ability to be deployed in actual vehicle control units.

Chapter 4 introduces AA, a constrained deep reinforcement learning based framework that aims to control the vehicle to emulate the varied driving styles of

human drivers. To achieve this objective, the proposed framework addresses three interconnected sub-problems: it categorizes the driver's driving style using a rule-based approach, trains neural network-based regressor models to predict human-like control actions for each driving style, and implements a C-DRL-based controller that mimics these control actions while ensuring safety. The performance of the proposed approach demonstrates its effectiveness in adapting vehicle control actions to align with different driving styles, achieving human-like driving behavior without sacrificing safety. Additionally, a comparative analysis with non-constrained DRL methods emphasizes the critical role of safety constraints in accurately imitating human-like driving, especially in scenarios involving aggressive driving behaviors.

Chapter 5 provides a summary of the conducted research, presents concluding remarks and outlines future research directions.

Chapter 2

Edge-assisted ML-aided Uncertainty-aware Vehicle Collision Avoidance at Urban Intersections

Intersection crossing represents one of the most dangerous sections of the road infrastructure and Connected Vehicles (CVs) can serve as a revolutionary solution to the problem. In this chapter, we present a novel framework that detects preemptively collisions at urban crossroads, exploiting the Multi-access Edge Computing (MEC) platform of 5G networks. At the MEC, an Intersection Manager (IM) collects information from both vehicles and the road infrastructure to create a holistic view of the area of interest. Based on the historical data collected, the IM leverages the capabilities of an encoder-decoder recurrent neural network to predict, with high accuracy, the vehicles' future trajectories. As, however, accuracy is not a sufficient measure of how much we can trust a model, trajectory predictions are additionally associated with a measure of uncertainty towards confident collision forecasting and avoidance. Hence, contrary to any other approach in the state of the art, an uncertainty-aware collision prediction framework is developed that is shown to detect well in advance (and with high reliability) if two vehicles are on a collision course. Subsequently, collision detection triggers a number of alarms that signal the colliding vehicles to brake. Under real-world settings, thanks to the preemptive capabilities of the proposed approach, all the simulated imminent dangers are averted.

The part of the work presented in this chapter has already been published in Selvaraj, D. C.; Vitale, C.; Panayiotou, T.; Kolios, P.; Chiasserini, C. F.; Ellinas, G., "Edge-Assisted ML-Aided Uncertainty-Aware Vehicle Collision Avoidance At Urban Intersections," in IEEE Transactions on Intelligent Vehicles, doi: 10.1109/TIV.2023.3296190, ©2023 IEEE.

2.1 Introduction

Recent studies have shown that accidents on the road infrastructure result in more than 1.19 million worldwide casualties annually [12], highlighting that safety is still of the utmost concern in the automotive sector. As a possible solution, new intelligent functionalities have been recently considered for the new generation of connected (and automated) vehicles that are about to enter the market [13]. Specifically, several efforts exploit the fact that these vehicles can exchange information and create a better understanding of their surroundings, which, in turn, can help to improve driving assistance systems and avoid dangerous situations. As an example, an enriched view of the driving environment helps in refining the vehicles' trajectory predictions, which could be later used as the main tool to foresee possible collisions on the road infrastructure.

As a consequence, predicting the trajectories of Connected Vehicles (CVs) is a well-researched topic. Nevertheless, predicting future collisions starting from trajectory predictions has been scarcely investigated, and still remains a challenging task. For a successful collision detection framework, indeed, trajectory predictions must be compared effectively, and, for any possible real-time safety application exploiting such a framework, in a time-efficient manner. Hence, *when it comes to collision avoidance*, simplified trajectory predictions have been proposed, namely: (i) constant speed location projections [14]; (ii) Kalman Filter (KF) predictions, with a Time-To-Collision (TTC) metric being used as a risk factor [15]; and (iii) collision time estimation via polynomial approximations [16]. However, the aforementioned approaches do not achieve the level of precision required for reliable and preemptive collision detection when a complex driving environment is considered, i.e., when human-driven vehicles at urban intersections are considered [17]. Indeed, due to the human factor, these simplified trajectory predictions cannot cope with the higher

levels of uncertainty (i.e., input uncertainty is present) and with the more evident non-linearities.

This work aims at filling this gap by presenting a framework that preemptively detects collisions, with short execution time, exploiting a precise model able to capture the highly non-linear and uncertain behavior of drivers, even at dense urban crossroads. Our framework is based on the ability of the Multi-access Edge Computing (MEC) platform of 5G networks to collect, thanks to both Vehicle-to-Infrastructure (V2I) and Infrastructure-to-Infrastructure (I2I) communications, relevant and extensive data sampled in real-time (e.g., regarding vehicle's location and speed, but also traffic light phases) at an edge entity, named the Intersection Manager (IM). Such data is processed directly at the edge to obtain not only accurate trajectory predictions, but also an estimation of the associated prediction uncertainty. It is worth mentioning that, even if a few trajectory uncertainty estimation models exist in the literature, e.g., [18], these are obtained in a computationally expensive way, which is not appropriate for real-time decision-making. In this work, instead, trajectory uncertainty estimation is obtained through an estimation of the prediction intervals [19], which capture indirectly the various, possible, modes of vehicle's future trajectories without the need of defining the set of possible driver actions at the intersection.

More in detail, trajectory predictions and uncertainty estimations are obtained through the use of two lightweight independent encoder-decoder regression architectures with recurrent units, specifically designed for sequence-to-sequence prediction problems, i.e., using LSTM Encoder Decoder (LSTM-ED) models [20]. In particular, for trajectory prediction, an LSTM-ED model predicts the future positions of a vehicle over several subsequent timesteps. For uncertainty estimation, a separate LSTM-ED model leverages a deep quantile regression technique to determine the lower and upper bounds for each predicted future position of the vehicle. Then, a Random Forest Classifier (RFC), working as an ensemble technique, is used to recognize patterns of trajectory predictions and uncertainty estimations that led to dangerous situations in the past, to timely determine if two vehicles are on collision course. Thanks to the proposed approach, the IM can transmit alarms to the vehicles involved, so that the imminent danger is avoided.

The performance of the proposed approach is validated through extensive testing using a diverse set of simulations based on real-world intersection data. Specifically,

the SUMO mobility simulator is utilized to simulate real-world mobility traces at two intersections in Monaco and Luxembourg. These mobility traces were employed to train the trajectory prediction and uncertainty estimation models, which are further used to detect collisions. The results are of high quality in terms of: (i) detection rate; (ii) low false positive rate; and (iii) time between the alarm sent by the Incident Management system and the collision time, i.e., the available reaction time. In the obtained results, accounting for trajectory prediction uncertainty estimation allows for: (i) reducing the number of false collision detections when the point trajectory predictions suggest a collision but a large prediction uncertainty is associated with them; and (ii) improving the available reaction time to possible threats where a large uncertainty is associated with safe trajectory predictions. As a result, all dangerous situations are predicted largely in advance by the proposed framework, allowing to efficiently avoid the imminent collisions with simple maneuvers that can be applied also by human drivers, e.g., constant braking.

In this chapter, Sec. 2.2 reviews the state of the art on trajectory prediction and on collision avoidance algorithms at urban intersections, and summarizes the main contributions of this work (Sec. 2.2.3). Sec. 2.3 illustrates an overview on the proposed collision forecast framework, Sec. 2.4 shows the trajectory prediction and the uncertainty estimation models, while Sec. 2.5 and Sec. 2.6 present the RFC technique that is used to forecast collisions, and how it can be used to avoid imminent dangers at urban intersections, respectively. Sec. 2.7 showcases the obtained performance on a realistic real-world intersection data, including comparisons with a collision avoidance technique that only exploits point-based vehicle trajectory predictions and with the state-of-the-art approach in [16]. Finally, Sec. 2.8 concludes the chapter.

2.2 Related Work

As a vital safety application in the automotive domain, diverse methodologies have been presented for preemptive vehicle collision detection [21]. As in our approach, in most of the proposed solutions, e.g., [22, 15, 16, 23], a collision is detected thanks to the use of vehicle trajectory predictions. As such, we first discuss existing trajectory prediction approaches that can be applied to urban intersections (Sec. 2.2.1), then we discuss previous works that propose collision avoidance frameworks (Sec. 2.2.2),

and finally, we summarize how our proposed approach advances the state of the art (Sec. 2.2.3).

2.2.1 Vehicle Trajectory Prediction

The vehicle trajectory prediction problem is by itself a popular topic with significant research activity [24]. Related works mostly focus on predicting vehicle trajectories for simple scenarios, e.g., in highway scenarios, or they exploit only a partial view of the road infrastructure ([25–28]) when more complex ones are analyzed.

This type of solution can hardly be extended to urban intersections. As an example, Xing et al. [29] proposes a vehicle trajectory prediction framework where its output is based on a "static" categorization of the drivers' driving styles. Assuming that driving styles remain constant over time works well in a highway setting, but such an assumption does not hold at intersections, where the drivers' decisions depend on the specific scenario encountered. On the other hand, Liu et al. [30] use Gaussian Processes to define both the vehicle's expected trajectory and the uncertainty associated with the prediction, enumerating the probabilities associated with all possible driver intentions in a highway scenario. Considering the countless possible maneuvers at intersections, a similar approach is difficult to tune and does not scale.

Fewer works are present if the urban intersection use case is taken into account. A first result involves the prediction of drivers' maneuver intentions, i.e., predicting if a vehicle proceeds straight or turns at the intersection, rather than examining full trajectory predictions [31, 32]. More evolved approaches provide a "group" trajectory prediction framework, following the intuition that vehicles' interaction is critical to improve performance [33, 34]. Similarly, single-vehicle trajectory predictions accounting for interaction-aware architectures based on Graphical Neural Networks (GNNs) have been proposed recently [35, 36]. Herein, both interactions among vehicles and between vehicles and the road infrastructure are modeled and accounted for to improve performance. Furthermore, a multimodal trajectory prediction framework generating multiple possible trajectories for each vehicle, and their corresponding likelihood, was proposed in [37]. Even if the support of wireless communications has been envisioned in [37], and despite the fact that incorporating surrounding vehicle movements improves the prediction accuracy, the computation complexity increases

multi-fold when all the possible maneuvers and interactions are modeled individually. Building on the same concept, hence suffering from the same limitations, Strohbeck et al. [38] makes use of Temporal Convolutional Networks and of Kalman Filter outputs, to also obtain a trajectory uncertainty estimation.

Alternatively, Liu and Shoji [39] proposes an approach similar to the one presented in this work. A trajectory prediction model is obtained through the monitoring of a specific area by collecting information at a MEC server. A general trajectory prediction model is then transmitted back to the vehicles, to update the model based on their driving style. Even though interesting, this approach considers the area of interest divided into "squares", among which the movement of vehicles is predicted, hence suffering from the quantization effect. Furthermore, Huang et al. [18] obtains a set of predictors based on different input sources (i.e., camera, odometry, and other sensor data). Herein, uncertainty is obtained as a confidence score and it is used to select, among the predictors obtained, the best trajectory prediction. Nevertheless, trajectory predictions based on a single source, even if the best at any point in time, are not as precise as building a trajectory prediction from multiple sources and properly accounting for the associated uncertainty.

Finally, it must be noted that none of the approaches presented above integrates the trajectory prediction framework with an approach that preemptively forecasts collisions among vehicles; hence, it is unclear if any of the advanced aforementioned approaches are able to fulfill the necessary critical time-sensitive requirements of a collision avoidance application.

2.2.2 Collision Detection/Avoidance

The approaches present in the state of the art for collision detection/avoidance can be divided into two groups, i.e., in frameworks that forecast collisions based: (i) on information relative to the present state of the vehicles; and (ii) on information that involves predictions of vehicles' future states.

For the first class of collision prediction algorithms, two metrics are used to identify imminent dangers: (i) distance [14, 40, 41], and (ii) time [42, 23, 15, 22]. In [14, 40, 41] distance-based collision detection methodologies utilize both real-time in-vehicle sensors' measurements and/or information received through a communication framework. In [42], while similar input data are considered, a time-based

approach is used to detect and avoid collisions. Specifically, authors in [42] utilize a set of virtual collision points between any two lanes in an intersection and a local database where vehicles' intentions are collected. Then, while approaching the intersection, each vehicle queries the database to check if any of the surrounding vehicles will simultaneously be at the same virtual collision point. All the aforementioned approaches use the vehicle's current information to estimate the vehicles' future pair-wise distances or collision times, assuming, however, only constant movement, without any speed or direction changes. As a consequence, these methods are reactive in nature and are inherently affected by a delay in detecting imminent collisions at urban intersections.

Very few approaches, e.g., [22, 15, 16, 23], belong to the second class of collision detection algorithms. As in our work, in this class of solutions, more precise trajectory predictions are used as a mean to foresee or prevent collisions. In [23, 15], a KF is used to predict the future trajectories of the vehicle, with TTC used as a risk factor to identify the potential collisions. Similarly, Tao et al. [22] uses a non-linear Square-Root Unscented Kalman Filter (SR-UKF) to predict vehicles' trajectories, and utilizes the Monte Carlo sampling method to define the uncertainty region around the predicted trajectories. Then, a set of collision risk factors, such as the probability of the current trajectory, TTC, and conflict points in vehicles' bounding boxes, are used to detect collisions.

A different approach is followed in [43], where a decentralized collision avoidance method is proposed. Specifically, this method forbids a vehicle to enter specific sets of speed and position configurations that eventually lead to collisions. Based on the assumption that the vehicle path and collision area are known, the study predicts if a vehicle is about to enter such a set, based on a Kalman filter prediction technique, and, if this is the case, it ensures that the automated collision avoidance mechanism located on-board the vehicle executes specific strategies to avoid the imminent danger. Similarly, De Campos et al. [44] and Brannstrom et al. [45] account for constant turn rate and acceleration-based unscented Kalman filter to predict collisions. Accounting for human drivers, herein the automated collision avoidance mechanism enters in action only if the danger is imminent, and the driver has not intervened as of yet. Furthermore, Malinverno et al. [16] approximates vehicles' future trajectories by solving a fourth-degree equation concerning the vehicles' derivatives (position, velocity, and acceleration). Collisions are then determined based on the time (time-to-collision) and spatial (space-to-collision) domain thresholds across all vehicle pairs.

Finally, following a completely different philosophy, in the state of the art there exists a set of approaches that, instead of predicting a collision, they plan safe trajectories for the vehicles based on an estimation of risk from their surroundings ([46, 47]). Nevertheless, this class of solutions only applies to autonomous vehicles, hence it is outside of the scope of this work.

2.2.3 Novel Contribution

Overall, even though there exists a large body of work concerning vehicle trajectory prediction and LSTM models have already been used for this task, e.g., [25], our framework differs from prior art as it exploits a holistic view of the area of interest, including the past and the present status of both the infrastructure and all the vehicles crossing that area. Such holistic view is obtained thanks to the adoption of 5G and MEC technologies. Furthermore, differently from any other work in the field, prediction intervals are used to measure the reliability of the trajectory predictions. This allows capturing the presence of multiple possible future driver choices in a single metric, which can be computed efficiently, since it does not require modeling separately all maneuvers.

To the best of our knowledge, this work presents, for the first time, an ML-aided uncertainty-aware framework that allows obtaining efficient preemptive collision detection, even when considering human-driven vehicles at urban intersections. Indeed, contrary to all proposed approaches, our collision forecast method exploits both precise trajectory predictions and the associated uncertainty estimations, which allow raising alarms, in specific scenarios, even if the vehicles' predicted trajectories appear to be safe. Thanks to the use of the RFC, the adopted framework is able to easily recognize real-time patterns of trajectory predictions and uncertainty estimations that led to dangerous situations in the past, raising the corresponding alarms within tight latency constraints. As a result, the proposed framework is able to predict an imminent danger well in advance, so that human drivers can avoid the collision by performing simple maneuvers.

Using real-world data, the proposed framework shows improved performance in all monitored metrics (i.e., in terms of trajectory prediction errors, collision detection false positives, available reaction time to drivers to avoid an imminent collision) when compared to: (i) the most relevant work on collision detection at

urban intersections, i.e., [16]; and (ii) a collision detection technique based only on trajectory predictions. In particular, when compared to a conventional technique that considers only point-based trajectory predictions, our framework improves the median available reaction time by 61%. Hence, given the obtained performance, our framework satisfies all the requirements (highly accurate output and short runtime) of the collision avoidance application at hand.

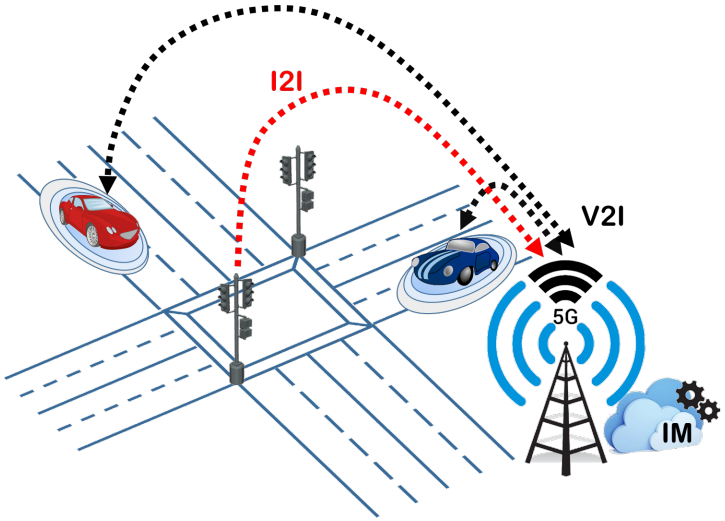

Fig. 2.1 An edge computing-based scenario, including an IM collecting data generated by the vehicles and the road infrastructure and transmitting alarms in case of imminent collisions.

2.3 Network-Assisted Collision Detection: Overview of our Framework

In this work, we focus on a network-assisted preemptive collision detection framework that allows vehicles to prevent dangerous situations. Although the use of such an application can be extended to any dangerous area of the road infrastructure, here we focus on automated and human-driven CVs crossing urban intersections, as depicted in Fig. 2.1.

In our framework, an IM is hosted at the MEC [48] in the proximity of a gNodeB (gNB) covering the geographical area around the intersection. The appropriate number of computing resources to allocate to the IM to respect the time-sensitive nature of the application at hand can be determined through strategic scheduling even

in a MEC platform running multiple services, as shown in [49]. Through integrated V2I and V2V communications, the IM can gather data from multiple sources, namely, (i) onboard sensor measurements sent by the vehicles crossing the intersection through Cooperative Awareness Messages (CAMs), and (ii) infrastructure-based information transmitted by city smart sensors and cameras. Note that CAMs can include several pieces of information provided by the vehicle's Controller Area Network (CAN) bus, including speed, direction, steering angle, acceleration, braking, yaw rate, and relative distances with surrounding vehicles obtained through LiDAR. As for the data collected by the road infrastructure, these can include traffic light phases, vehicle lane information, and number of vehicles in each lane. As a result, the benefits of using a central entity, i.e., the IM, at the network edge for road safety are multi-fold: (i) at any time, the IM has a significantly richer view of the intersection than the individual vehicles, (ii) the IM can collect CAMs and historical data (e.g., collisions/dangerous situations) in a local database to train precise models, and (iii) unlike cloud implementations, the IM meets the tight latency constraints of real-time safety applications. In this work, with the obtained extended view of the intersection, the IM accurately predicts potential future dangerous situations. Moreover, upon detecting two or more vehicles on a collision course, it promptly alerts them by transmitting a Decentralized Environmental Notification Message (DENM). Upon receiving such an alarm, CVs will start braking; thus, the objective of the IM is to detect a possible upcoming collision early enough to let vehicles avoid the imminent risk.

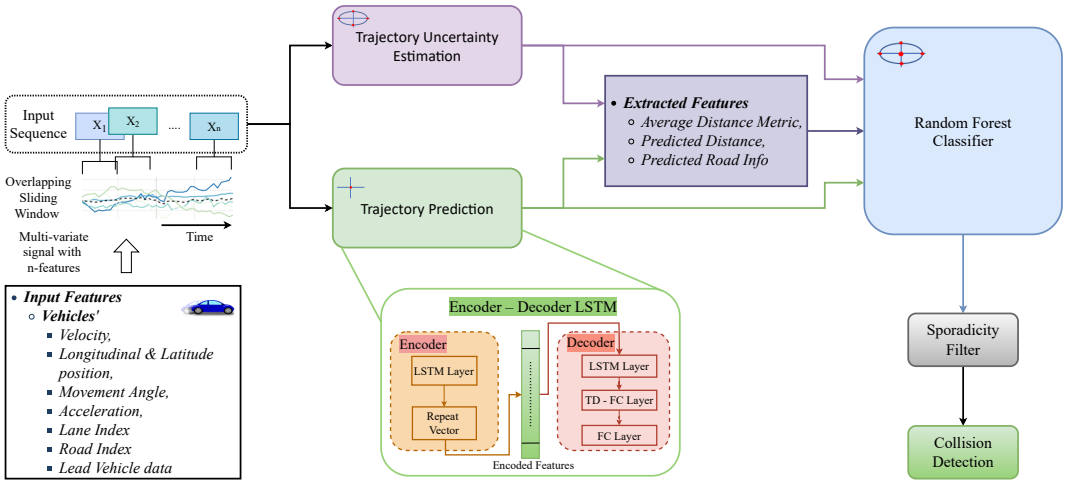

Fig. 2.2 An overview of the proposed network-assisted collision detection methodology.

To this end, we envision the implementation at the IM of the collision detection framework whose structure is depicted in Fig. 2.2. Our detection framework searches for a possible collision as soon as a new CAM, updating the information relative to a vehicle status, is received. Using a sliding window approach, recent past information originating from vehicles and the road infrastructure are used as *input data* to our framework. Collision detection at intersections is in essence composed of two complementary, yet distinct, sub-problems that operate in cascade, i.e., the latter takes as input the output of the former: (i) trajectory prediction and (ii) detection of vehicles on a collision course. *Vehicle trajectory prediction* consists of point-based estimates, inevitably containing some uncertainty. Thus, to develop a trustworthy collision detection mechanism, *trajectory prediction uncertainty* is appropriately measured, interpreted, and processed before decision-making. For both trajectory prediction and uncertainty estimation (even though with different loss functions), LSTM-ED models are exploited to: (i) solve the problem of storing long time-steps in the learning memory [50]; (ii) capture the intrinsic non-linear nature of vehicle trajectories; and (iii) scale to high-dimensional data (unlike Bayesian learning).

The obtained vehicles' trajectory predictions, along with the associated uncertainty estimation and other relevant information, are subsequently used by the IM to train an RFC that, based on historical data, recognizes patterns of dangerous inputs so to detect in real-time the event of a collision or of situations that are deemed dangerous. Finally, to avoid spurious detections, the IM employs a *sporadicity filter* that transmits to the vehicles involved an actual alarm only if, for a specific pair of vehicles, the RFC detects a dangerous situation for a number of consecutive predictions. Overall, a novel uncertainty-aware collision detection mechanism is developed, showing, as presented in Sec. 2.7, to be capable of accurately and timely identifying future collisions. Interestingly, adding to the proposed framework a measure of the trajectories' certainty level is crucial for improving the obtained performance.

2.4 Vehicle Trajectory Prediction and Uncertainty in Deep LSTM-ED

The trajectory prediction sub-problem can be in general described as a sequence-to-sequence problem where the vehicle's past trajectory is used as input, together with

other relevant information, for predicting the vehicle's future trajectory for several time-steps ahead. LSTM-ED models are a promising modeling approach (as shown in [17]) for the trajectory prediction problem. As in similar use cases, a least squares loss function is minimized to find a predictive model that, in essence, approximates the relationship between the input data collected by the IM and the vehicle's future positions. Hence, the output of the model is an estimate of the vehicle's future positions, inevitably containing some uncertainty. This estimate is in fact the mean response value, with the variability around that mean representing the uncertainty of the model itself and of the noisy input data.

When collision avoidance is the ultimate goal of the trajectory prediction sub-problem, a point-estimate is not enough, as it may lead to erroneous collision detections or more severely to the underestimation of a possible collision event. While the latter situation is more severe as the vehicles cannot be notified timely towards collision avoidance, the former should not be ignored, since it may lead to inappropriate decisions/actions with many undesired effects (i.e., frustration of drivers or even collisions). Measuring the certainty level of trajectory estimates, essentially building prediction intervals that quantify the existence of multimodal future trajectories [19], helps reducing the errors associated with our framework since: (i) it reduces the number of false collision detections when the average trajectory estimations suggest a collision, but a large prediction uncertainty is associated with them; and (ii) it avoids underestimating a possible threat when a large uncertainty is associated with safe trajectory predictions.

To this end, in this section we focus on trajectory prediction (Sec. 2.4.1) and uncertainty estimation (Sec. 2.4.2) with the purpose of defining a set of appropriate, uncertainty-aware trajectory estimates, subsequently utilized by the collision detection mechanism developed in this work. Furthermore, the set of inputs used by the presented model is introduced in Sec. 2.4.3. The LSTM-ED model used for both trajectory prediction and uncertainty estimation is then described in Sec. 2.4.4.

2.4.1 Least Squares Regression for Trajectory Prediction: Estimating Mean Response Values

For trajectory prediction, a least squares LSTM-ED model is trained to minimize the Mean Squared Error (MSE) loss function given by:

$$L_{mse} = \frac{1}{NL}\sum_{i=1}^{N}\sum_{j=1}^{L}||\mathbf{y}_{i,t+j} - \hat{\mathbf{y}}_{i,t+j}||_2^2, \tag{2.1}$$

where N is the number of sequences used for MSE evaluation (i.e., during training/testing), L are the time-steps used for prediction, t is the present time, $\mathbf{x}_i \in \mathbb{R}^d$ is a vector describing vehicle i's input data, $\mathbf{y}_{i,t+j} \in \mathbb{R}^2$ is a vector describing the true location (i.e., latitude and longitude information) of vehicle i at time slot $t+j$, and $\hat{\mathbf{y}}_{i,t+j} \in \mathbb{R}^2$ is the estimated vector inferred by the trained least squares LSTM-ED model $\hat{F}$. In essence, $\hat{F}$ is formally defined as $\hat{F}(\mathbf{x}_i, \{\hat{\mathbf{y}}_{i,t+s}\}_{s=0}^{j-1})$ as it utilizes as inputs both present and past data, i.e., $\mathbf{x}_i$, and previously predicted trajectory values to infer the trajectory value for the next step ahead. Hence, when $j=1$, then $\hat{F}(\mathbf{x}_i)$ predicts $\hat{\mathbf{y}}_{i,t+1}$, with this estimate subsequently used as input to $\hat{F}(\mathbf{x}_i, \hat{\mathbf{y}}_{i,t+1})$ to predict $\hat{\mathbf{y}}_{i,t+2}$, and so on.

2.4.2 Deep Quantile Regression for Trajectory Prediction: Estimating Lower and Upper Response Values

As previously mentioned, our uncertainty estimation method is based on building prediction intervals. By definition, a prediction interval is a range of values in which a single future observation will fall in, with a certain probability, based on the estimated model (i.e., given the previous observations used for training the model). While in ML several approaches exist for building prediction intervals, such as conducting Bayesian or Monte Carlo (MC) dropout inference [51], in this work, a deep quantile regression framework is adopted [52], providing a principled way of controlling the certainty level of the estimates. Specifically, a deep LSTM-ED model is trained to estimate a set of conditional quantile functions capable of providing lower and upper estimates for the vehicle's each future position.

To this end, we apply deep quantile regression to approximate the conditional quantile function $y_{i,t+j}^q = Q_q(\mathbf{x}_i, \{y_{i,t+s}^q\}_{s=1}^{j-1})$, also known as the q-quantile, where q

is the probability that the random variable $y_{i,t+j} \in \mathbf{y}_{i,t+j}$ (representing the vehicle's true location at the $t+j$-th future time step) conditional on the value of the random variable $\mathbf{x}_i$ (representing the input data) is less than or equal to $y^q_{i,t+j}$, i.e., $\mathbb{P}(y_{i,t+j} \leq y^q_{i,t+j} | \mathbf{x}_i, \{y^q_{i,t+s}\}_{s=1}^{j-1}) = q$ [53].

To derive lower and upper vehicle trajectory estimates, two q-quantiles need to be estimated. Since the objective is to create a prediction interval able to contain most of the vehicle's true future locations given the input data $\mathbf{x}_i$, a low enough quantile (i.e., with $q \to 0$) and a high enough quantile (i.e., with $q \to 1$) must be used for the lower and upper quantiles, respectively (denoted as l and u, respectively). It is worth mentioning, that while for each q value different quantile models can be trained, in this work we opted to train one single model for both lower and upper estimates of the vehicle's future position (i.e., longitude and latitude). Hence, an LSTM-ED model is trained to return upper and lower estimates for the longitude future locations of the vehicle, and an additional LSTM-ED model is trained to return upper and lower estimates for the latitude future locations of the vehicle. Considering that, in a deep learning framework a q-quantile is estimated by minimizing the asymmetrically weighted sum of absolute errors [52, 53], the loss function minimized during training is given by:

$$L_q = \frac{1}{2NL} \sum_{i=1}^{N} \sum_{j=1}^{L} \rho_l \left(y_{i,t+j} - \hat{y}^l_{i,t+j} \right) + \rho_u \left(y_{i,t+j} - \hat{y}^u_{i,t+j} \right), \tag{2.2}$$

where q denotes any of the two quantiles u and l:

$$\rho_q(z) = \begin{cases} qz, & \text{if } z \geq 0, \\ (q-1)z, & \text{if } z < 0, \end{cases} \tag{2.3}$$

and where $\hat{y}^q_{i,t+s} \in \hat{\mathbf{y}}^q_{i,t+s}$ is the estimation of the trained q-quantile LSTM-ED model $\hat{Q}_q$. Similarly to $\hat{F}$, in this work the approximation of Q_q obtained with the LSTM-ED model, i.e., $\hat{Q}_q$, sequentially utilizes as inputs previous quantile values $\{\hat{y}^q_{i,t+s}\}_{s=1}^{j-1}$. Hence, when $j = 1$, then $\hat{Q}_q(\mathbf{x}_i)$ returns $\hat{y}^q_{i,t+1}$, subsequently used as input to $\hat{Q}_q(x_i, \hat{y}^q_{i,t+1})$ to return $\hat{y}^q_{i,t+2}$, and so on.

2.4.3 Dataset Formulation

For both the considered least squares and quantile loss functions, the objective is to optimize the unknown parameters of an LSTM-ED model such that a non-linear function (i.e., the F or Q_q function, respectively) is approximated. In the former case, the approximation allows accurately predicting the vehicle's future location:

$$\mathbf{y}_i = [\mathbf{y}_{i,t+1}, \mathbf{y}_{i,t+2}, \cdots, \mathbf{y}_{i,t+L}] \tag{2.4}$$

where

$$\mathbf{y}_{i,t+j} = [y^{(\lambda)}_{i,t+j}, y^{(\phi)}_{i,t+j}] \tag{2.5}$$

is a vector including latitude vehicle information $y^{(\lambda)}_{i,t+j}$ and longitude vehicle information $y^{(\phi)}_{i,t+j}$ at the j-th future time step. In the latter case, instead, the approximation allows to accurately estimate the prediction intervals of the vehicle's future location:

$$\mathbf{y}^q_i = [\mathbf{y}^q_{i,t+1}, \mathbf{y}^q_{i,t+2}, \cdots, \mathbf{y}^q_{i,t+L}] \tag{2.6}$$

where

$$\mathbf{y}^q_{i,t+j} = [y^l_{i,t+j}, y^u_{i,t+j}] \tag{2.7}$$

is a vector that includes the lower and upper estimates of the vehicle's latitude/longitude location at the j-th future time step (i.e., $y^l_{i,t+j}$ and $y^u_{i,t+j}$, respectively).

To make predictions, the model is given as input the vehicle's past and present information:

$$\mathbf{x}_i = [\mathbf{x}_{i,t-T}, \mathbf{x}_{i,t-T+1}, \cdots, \mathbf{x}_{i,t-1}, \mathbf{x}_{i,t}]. \tag{2.8}$$

In the aforementioned expression, T is the number of past and present observations, t is the present time instant, L is the number of next vehicle observations (i.e., the prediction horizon), and $\mathbf{x}_i$, $\mathbf{y}_i$, and $\mathbf{y}^q_i$ form a time series (i.e., are sequential in time), with each time instant of interest being τ seconds apart (i.e., τ is the time scale of sampling/estimating time instants). Regarding the vehicle's past input data $\mathbf{x}_i$, each $\mathbf{x}_{i,j}$, $\forall j = t-T, ..., t$, is a vector containing d measurements for both the on-board and infrastructure components. Specifically, each $\mathbf{x}_{i,j} \in \mathbb{R}^d$ consists of the following set of observations:

$$\mathbf{x}_{i,j} = \{x^{(\lambda)}_{i,j}, x^{(\phi)}_{i,j}, x^{(r)}_{i,j}, x^{(v)}_{i,j}, x^{(a)}_{i,j}, x^{(l)}_{i,j}, x^{(e)}_{i,j}, x^{(p,\lambda)}_{i,j}, x^{(p,\phi)}_{i,j}, x^{(p,v)}_{i,j}\}, \tag{2.9}$$

where:

- $x_{i,j}^{(\lambda)}$ is the latitudinal position of vehicle i at time instant $t-j$;
- $x_{i,j}^{(\phi)}$ is the longitudinal position of vehicle i at time instant $t-j$;
- $x_{i,j}^{(r)}$ is the movement angle of vehicle i at time instant $t-j$;
- $x_{i,j}^{(v)}$ is the velocity of vehicle i at time instant $t-j$;
- $x_{i,j}^{(a)}$ is the acceleration of vehicle i at time instant $t-j$;
- $x_{i,j}^{(l)}$ is the lane index where vehicle i is traveling on at time instant $t-j$;
- $x_{i,j}^{(e)}$ is the road index on which vehicle i is traveling at time instant $t-j$;
- $x_{i,j}^{(p,\lambda)}$ is the latitudinal position of the vehicle preceding vehicle i in the same lane, at time instant $t-j$;
- $x_{i,j}^{(p,\phi)}$ is the longitudinal position of the vehicle preceding vehicle i in the same lane, at time instant $t-j$;
- $x_{i,j}^{(p,v)}$ is the velocity of the vehicle preceding vehicle i in the same lane, at time instant $t-j$.

Amongst the input features, the lane index information ($x_{i,j}^{(l)}$) and road index information ($x_{i,j}^{(e)}$) are converted into a one-hot encoded vector to benefit the learning process. Rather than giving semantic information on the road structure, in this work, lane and road index help the LSTM-based trajectory prediction to identify the intention of drivers of turning or of proceeding straight at the intersection. Indeed, given the structure of an intersection, drivers can turn only when approaching the intersection on specific lanes of the road. Furthermore, apart from features describing the past movement of vehicle i, e.g., its location, speed, and direction, a set of information regarding the preceding vehicle on the same lane is also included. The rationale is that, at an intersection, the correlation between vehicle i's future locations and the present location of the preceding vehicle is very high. Information about the preceding vehicles is provided by the IM, which maintains a holistic view of the intersection. By projecting all vehicle locations onto the intersection map, the IM system can ascertain the details of the preceding vehicle for each vehicle and utilize

this information to formulate the input features. Finally, it is worth mentioning that additional input features were also considered and tested during model training and inference, without however contributing to further improving model accuracy. Even though our model achieves a very high accuracy, as shown in the following sections, a further analysis of input features naturally present in CAM messages (or at the infrastructure) constitutes an interesting future research direction. In general, CAMs are divided into high-frequency and low-frequency containers. The high-frequency container includes frequently changing vehicle data such as acceleration, steering angle, and speed, while the low-frequency container holds more static or infrequently changing data such as turn signals and fog lights. Using turn signals as an input feature can help determine whether a vehicle is turning at an intersection. However, using human-activated signals as an input feature is less reliable, and accurately assessing potential performance improvements is challenging without real data.

2.4.4 The LSTM-ED Model

The LSTM-ED architecture we use is illustrated in Fig. 2.3, where index i is omitted to simplify the notation and the output $\mathbf{y}_t$ is alternatively equal to the trajectory prediction or the uncertainty estimation depending on the use of the LSTM model. Note that, for simplicity, encoders and decoders of one layer only are depicted, but the model can include more hidden layers (i.e., stack of LSTM layers). Both encoder and decoder components are formed by LSTM cells, which in Fig. 2.3 appear unfolded through time to depict their operation over the input and output sequences. In reality, inputs are processed sequentially, and outputs are obtained one after the other (and reintroduced at the input of the decoder for predicting the next time step).

As shown in Fig. 2.3, the basic components of the model are the LSTM cells. Figure 2.4 shows a representation of the LSTM cell when the information at a generic time instant $t' \in [t-T,...,t,..,t+L]$ is processed. The key of LSTM is the cell state $\mathbf{c}_{t'}$, that acts as the cell memory, storing a summary of the past input sequence. At the output, the LSTM cell combines such accumulated memory with the input received, so as to obtain the hidden state $\mathbf{h}_{t'}$, also representing the output vector of the LSTM cell. Note that $\mathbf{h}_{t'}$ is also passed to any upper layer, if present. Furthermore, at the decoder cell, the output $\mathbf{h}_{t'}$ is transformed to generate $\mathbf{y}_{t'-1}$. Further details on the LSTM cell structure can be found in [54].

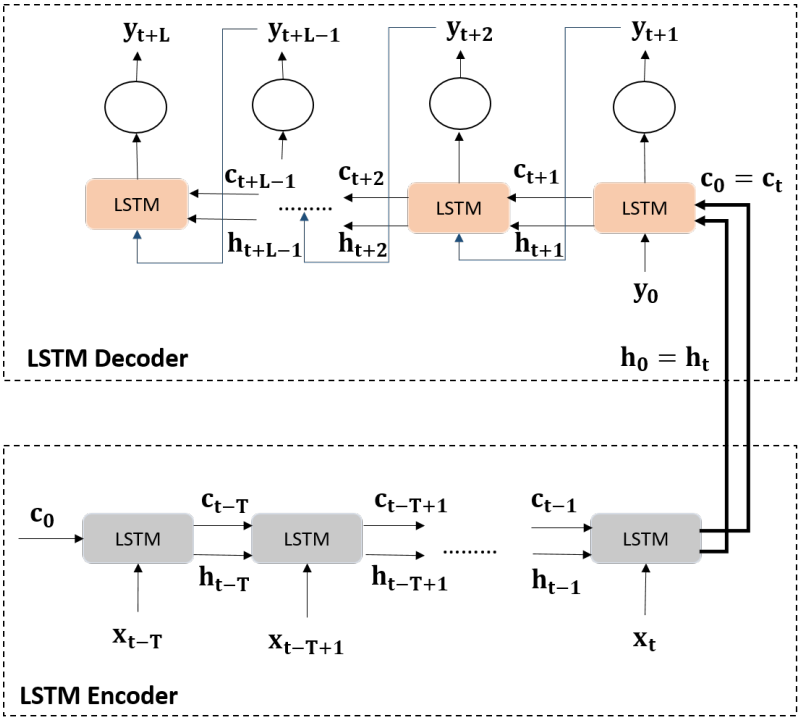

Fig. 2.3 The encoder-decoder LSTM architecture used for vehicle trajectory estimation and uncertainty estimation.

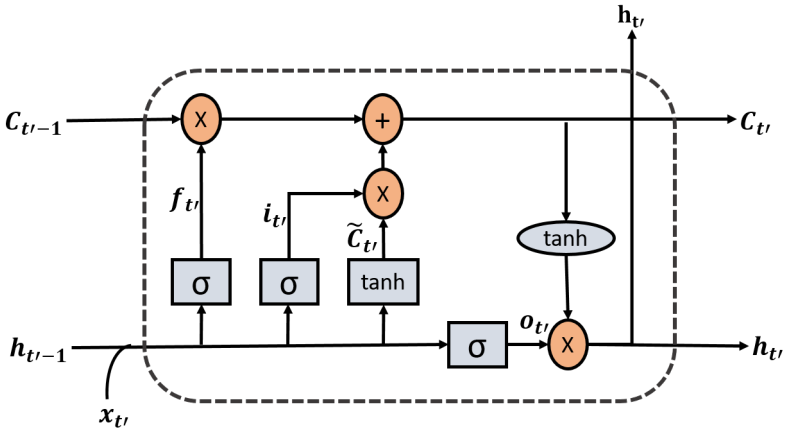

Fig. 2.4 General LSTM cell structure.

An LSTM cell is composed of an input gate vector $\mathbf{i}_{t'}$, an output gate vector $\mathbf{o}_{t'}$, and a forget gate vector $\mathbf{f}_{\mathbf{t}'}$. The three gate vectors control the flow of information in and out of the LSTM cell. Specifically, with the input gate vector the LSTM cell decides whether to update the cell state or not, with the forget gate vector the LSTM cell can erase cell state memory, and with the output gate vector it decides whether to make the output information available.

The following recursive equations describe in detail how the LSTM cell works:

$$\mathbf{i}_{t'} = \sigma(\mathbf{x}_{t'}\mathbf{U}_i + \mathbf{h}_{t'-1}\mathbf{W}_i), \tag{2.10}$$

$$\mathbf{f}_{t'} = \sigma(\mathbf{x}_{t'}\mathbf{U}_f + \mathbf{h}_{t'-1}\mathbf{W}_f), \tag{2.11}$$

$$\mathbf{o}_{t'} = \sigma(\mathbf{x}_{t'}\mathbf{U}_o + \mathbf{h}_{t'-1}\mathbf{W}_o), \tag{2.12}$$

$$\tilde{\mathbf{c}}_{t'} = \tanh(\mathbf{x}_{t'}\mathbf{U}_g + \mathbf{h}_{t'-1}\mathbf{W}_g), \tag{2.13}$$

$$\mathbf{c}_{t'} = \mathbf{f}_{t'} \circ \mathbf{c}_{t'-1} + \mathbf{i}_{t'} \circ \tilde{\mathbf{c}}_{t'}, \tag{2.14}$$

$$\mathbf{h}_{t'} = \tanh(\mathbf{c}_{t'}) \circ \mathbf{o}_{t'}\,, \tag{2.15}$$

where $\mathbf{x}_{t'}$ is a generic input vector, $\sigma(\cdot)$ is the sigmoid function, $\circ$ denotes the element-wise product, $\mathbf{W}_i \in \mathbb{R}^{u\times u}$, $\mathbf{W}_f \in \mathbb{R}^{u\times u}$, $\mathbf{W}_o \in \mathbb{R}^{u\times u}$, $\mathbf{U}_i \in \mathbb{R}^{d\times u}$, $\mathbf{U}_f \in \mathbb{R}^{d\times u}$, $\mathbf{U}_g \in \mathbb{R}^{d\times u}$ are linear transformation matrices (i.e., unknown parameters obtained during learning), and u is the number of hidden units.

In the LSTM-ED, the encoder reads each vector $\{\mathbf{x}_{t-k'}\}_{k'=0}^{T}$ of an input data $\mathbf{x}$ sequentially. As it reads each vector, the cell state of the LSTM changes according to Eq. (2.14) and the hidden state changes according to Eq. (2.15). After reading the end of the input data, the encoder summarizes the entire input sequence into the final vectors $\mathbf{c}_t$ and $\mathbf{h}_t$. The decoder is trained to generate an output trajectory $\{\mathbf{y}_{t+k}\}_{k=1}^{L}$ or quantile estimation $\{\hat{\mathbf{y}}_{t+k}^{q}\}_{k=1}^{L}$ sequentially, given $\mathbf{c}_t$ and $\mathbf{h}_t$ as the initial cell and hidden states, respectively (Fig. 2.3). The decoder is also initialized according to a dummy input $\mathbf{y}_0$. At every update k, the decoder feeds each output $\mathbf{y}_{t+k-1}$ obtained in the previous update, to the input of the current update. However, the generic output $\mathbf{y}_{t+k}$ is not the output of the decoder cell; in fact the generic output is obtained as $\mathbf{y}_{t+k} = g(\mathbf{h}_{t+k})$, where $g(\cdot)$ can be the activation function of the output layer and $\mathbf{h}_{t+k}$ is given by Eq. (2.15). In our trajectory prediction problem, the output of the decoder is followed by a fully connected dense layer (in essence, a linear transformation of the decoder output) to fit our regression task.

To summarize, the LSTM-ED model, parameterized by the set of unknown parameters of the model, learns a nonlinear function that predicts the current output value $\mathbf{y_{t+k}}$ as:

$$\mathbf{y}_{t+k} = \hat{\Theta}(\mathbf{x}_{t-T}, \cdots, \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{y}_{t+1}, \mathbf{y}_{t+2}, \cdots, \mathbf{y}_{t+k-1}), \tag{2.16}$$

with the encoder providing the summary of the input data $\{\mathbf{x}_{t-k'}\}_{k'=0}^{T}$ through the cell and hidden states $\mathbf{c}_t$ and $\mathbf{h}_t$, respectively, and $\hat{\Theta}$ being $\hat{F}$ or $\hat{Q}$, depending on the regression task at hand.

The two components of the LSTM-ED are trained on a dataset $D = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$ where N is the number of labeled vehicle sequences, to minimize the MSE or quantile loss function. Note that, for what concerns the prediction intervals, the LSTM-ED model is trained for obtaining the uncertainty estimation on both latitude and longitude. Hence, dataset D is further partitioned into two datasets $D^{(\lambda)} = \{\mathbf{x}_i, y_i^{(\lambda)}\}_{i=1}^{N}$ and $D^{(\phi)} = \{\mathbf{x}_i, y_i^{(\phi)}\}_{i=1}^{N}$. For training, the Adam optimization algorithm [55] is used.

2.5 An ML-aided Uncertainty-aware Collision Detection Mechanism

Because of our LSTM-ED approach, for any vehicle in the area of interest, both the trajectory prediction and the associated uncertainty estimation are available at the IM for a prediction window of L time-steps. The IM exploits such estimation of the vehicles' future position, along with other relevant data, to make an informed decision on whether a pair of vehicles is on a collision course. To this end, the IM trains an RFC [56, 57] with the aim to determine if two vehicles, described by an input pattern $\chi = \{\chi_1, \chi_2, ..., \chi_k\}$, are expected to collide within the prediction window, in which case the RFC returns classification outcome $\upsilon = 1$, otherwise it returns $\upsilon = 0$.

In a nutshell, RFC is an ensemble of multiple independent decision trees, where the classification of unseen patterns is performed through majority voting. Thanks to its structure, RFC is one of the most successful ensemble learning techniques that is robust to overfitting (as compared to individual decision trees [56]), and proven to be very powerful for problems characterized by high dimensionality and by skewed

data. Thus, it suits very well our scenario where real-world data is expected to be skewed, due to the presence of outlying driving behaviors.

Specifically, we apply feature subsampling to build the trees by splitting the training dataset into sub-datasets according to the value taken by the input features. Eventually, each decision tree is organized into branches composed of two types of nodes, i.e., the decision nodes (at which splits are made) and the leaf nodes, where the classification outcome takes place. Finding the best split at the decision nodes involves choosing the feature and the split value (i.e., a threshold) for that feature that will result into the highest improvement to the model, given the desired outcome v. That is, at each split, two sub-datasets with lower "impurity" are obtained when compared to the original dataset at the decision node. In this work, similarity within each sub-dataset is computed following the Gini Index (GI) [57], which measures the portion of samples in each sub-dataset having different outcomes. At each decision node, since the number of split options is large, only a subset of potential split values is considered per feature [58].

Splitting at the decision nodes continues until either all values in the sub-datasets of that node are pure (i.e., all samples describe a collision or a non-collision) or some other conditions are met (e.g., maximum tree depth).

In the inference phase, to classify any unseen pattern, the tree is navigated up to reaching a leaf node containing an output for the objective binary variable v. A specific leaf node is reached if the unseen pattern falls into specific ranges for the input features, as learned during training.

2.5.1 RFC's Inputs

As it is clear from the random forest approach presented above, the classification of the input data relative to two vehicles i and k in the "collision" class happens if the chosen input is within specific features' intervals. Hence, the choice of the input features is critical for improving the obtained performance.

Among the different possible input features, vehicle's present and future locations certainly play a crucial role in determining whether the two vehicles are on a collision course. Indeed, such features help to determine specific locations on the road infrastructure where more often vehicles tend to collide/experience dangerous situations. Accordingly, also the road direction traveled by the vehicles under anal-

ysis carries important information, since collisions are usually prone to happen to vehicles heading towards a specific route.

Intuitively, the future predicted distance between vehicles is also an indicator of a possible imminent collision. Indeed, independently from their location, a dangerous situation is certain in case two vehicles are predicted to be too close to each other. From the experiments performed, two metrics of proximity between vehicles helped to obtain the best performance, one based on the trajectory prediction and one based on its uncertainty estimation. That is, the first one is the Euclidean Distance (ED) $d_{ik,t+j}$ between the predicted location of vehicles i and k at time $t+j$. The second one is based on transforming the obtained prediction intervals into a distribution around the trajectory prediction described in Sec. 2.4.1, and on computing in distribution (and not based on the point estimates) the squared expected distance $E[d^2_{ik,t+j}]$ between the predicted location of vehicles i and k at time $t+j$. Specifically, it is assumed that the uncertainty associated with the trajectory prediction follows a Gaussian distribution, an assumption that is typically valid when the uncertainty is due to a combination of several independent factors, as in the use case under consideration. Then, the future location of vehicle i follows the multivariate Gaussian distribution $\mathcal{N}(\mathbf{y}_{i,t+j}, \Sigma_{i,t+j} = [\sigma^{2\ (\lambda)}_{i,t+j} \ \ 0;0 \ \ \sigma^{2\ (\phi)}_{i,t+j}])$. Given that the length of the prediction intervals is known from the method introduced in Sec. 2.4.2, the covariance matrix of the Gaussian distribution can be easily obtained as follows[59]:

$$\sigma^{2\ (\lambda)}_{i,t+j} = \frac{(y^{u(\lambda)}_{i,t+j} - y^{l(\lambda)}_{i,t+j})^2}{K_\varepsilon}; \quad \sigma^{2\ (\phi)}_{i,t+j} = \frac{(y^{u(\phi)}_{i,t+j} - y^{l(\phi)}_{i,t+j})^2}{K_\varepsilon} \tag{2.17}$$

where $\sigma^{2\ (\lambda)}_{i,t+j}$ and $\sigma^{2\ (\phi)}_{i,t+j}$ are the variances associated with the predicted latitude and longitude location, respectively, of vehicle i at time $t+j$, and K_ε is the inverse of the Cumulative Density Function (CDF) of the chi-squared distribution having two degrees of freedom computed at $1-\varepsilon = 1-(u-l)$. As a result, the expected squared ED between the two vehicles i and k can be computed as:

$$E[d^2_{ik,t+j}] = ||\mathbf{y}_{i,t+j} - \mathbf{y}_{k,t+j}||^2 + \mathsf{Tr}[\Sigma_{i,t+j} + \Sigma_{k,t+j}] \tag{2.18}$$

where $\mathsf{Tr}[\cdot]$ denotes the trace operator.

Finally, we also include in the input data the size of the prediction intervals and the covariance matrices $\Sigma_{i,t+j}$ and $\Sigma_{k,t+j}$ as a measure of the uncertainty estimation

in order to account in the classification task for the reliability level of the information based on the trajectory predictions, i.e., the future positions of vehicles i and k and their ED.

To summarize, the input data χ concerning a pair of vehicles i and k and a specific future time instant $t+j$, includes the following features:

- $y^{(\lambda)}_{z,t+j}$ ($y^{(\phi)}_{z,t+j}$) is the latitudinal (longitudinal) position of either vehicle i or k at future time instant $t+j$;
- $x^{(l)}_{z,t+j}$ is the road direction where either vehicle i or k are predicted to be traveling on at the future time instant $t+j$;
- $d_{ik,t+j}$ is the ED between vehicles i and k at future time instant $t+j$, based on the predicted point estimates;
- $E[d^2_{ik,t+j}]$ is the expected squared ED between vehicles i and k at future time instant $t+j$, based on the predicted distribution estimates;
- $y^{u(\lambda)}_{z,t+j} - y^{l(\lambda)}_{z,t+j}$ ($y^{u(\phi)}_{z,t+j} - y^{l(\phi)}_{z,t+j}$) is the latitudinal (longitudinal) prediction interval of either vehicle i or k at future time instant $t+j$;
- $\Sigma_{z,t+j}$ is the covariance matrix associated with the predicted location $\mathbf{y}_{z,t+j}$ of either vehicle i or k at future time instant $t+j$.

2.6 Avoiding Collisions at Urban Intersections

Due to the proposed approach, the IM can exploit an uncertainty-aware collision detection mechanism to predict in advance whether two vehicles are on a collision course. Once a collision is detected, the IM informs the vehicles in danger through unicast DENMs, so that corrective actions can be undertaken. In order to reduce the number of alarms relative to situations that do not lead to an actual collision or to an actual dangerous circumstance (i.e., to reduce the number of unnecessary and sudden corrective actions), the IM transmits DENMs only after the RFC has detected a collision over a number of consecutive observations.

Selecting the right number of consecutive observations, which leads to a DENM message transmission, represents a crucial trade-off between detecting a collision

on time, so that corrective actions are effective, and reducing the number of generated false alarms. In this context, it has to be considered that, from the moment the DENM is transmitted to the moment a vehicle starts braking, several latency components need to be accounted for. Figure 2.5 illustrates an overview of such latency components. As an alarm is raised by a CAM that triggers the detection of a collision at the IM, the end-to-end communication latency component, i.e., T_x, includes both CAM and DENM transmission latencies. To reduce this latency, as mentioned earlier, the IM runs at the edge on the 5G-MEC platform, geographically close to the intersection where the collision detection application is intended to work. The detection latency component, i.e., T_d, models the time required by the IM to process the information received from the vehicles. Thereafter, the DENM is received at the vehicle side and before a corrective action is actually undertaken, two further latency components need to be considered, i.e., the in-vehicle processing time T_p and the human reaction time T_r. Clearly, these latency components are modeled differently if a vehicle is assumed to be exclusively human-driven or if it is assisted by an automated braking system. In the former case, the two components include the fact that the vehicle processes the DENM and displays on the dashboard a visual alarm to the driver, which reacts to it with some delay. In the latter case, instead, the vehicle processes the DENM and automatically starts braking, without additional delay, i.e., $T_r = 0$. As a corrective action, we assume that vehicles brake as soon as the alarm is correctly analyzed, with the intention of reaching a halt position (reached after braking for T_a seconds) before colliding with any other vehicle in the intersection. Hence, in our system, a collision is considered as successfully detected if:

$$T_x + T_d + T_p + T_r + T_a < T_c, \tag{2.19}$$

where T_c is the time interval between the moment a CAM is transmitted by a vehicle and the moment such vehicle is involved in a collision. Specifically, (i) for the communication end-to-end latency component, T_x, we used a beta distribution, $B(\alpha = 2.0, \beta = 5.0)$, scaled between 2.4 (minimum) and 18 (maximum) ms [60]; (ii) for the in-vehicle processing data upon receiving the DENM, For in-vehicle data processing upon receiving the DENM, T_p, a constant latency of 400 ms is assumed [16]. Given the advanced computational capabilities of new-generation vehicles [61], T_p can also account for the preparation of actuators for effective braking when necessary; while (iii) for the driver reaction time, T_r, we have used a truncated normal distribution, $\mathcal{N}(\mu = 680, \sigma = 145, a = -1.24, b = 1.52)$, for human-driven

vehicles and $T_r = 0$ for automated vehicles [62]. The processing time, T_d, was determined through extended simulations. The LSTM models' computation time, for both vehicle trajectory prediction and uncertainty estimation, was measured to be approximately 11 ms. Likewise, the RFC model, which checks whether a pair of vehicles is on a collision course, was evaluated to take around 12 ms. In both cases, computation time is almost deterministic, with no deviation from the average. Hence, further assuming that collision tests are performed in parallel, T_d is set to 23 ms.

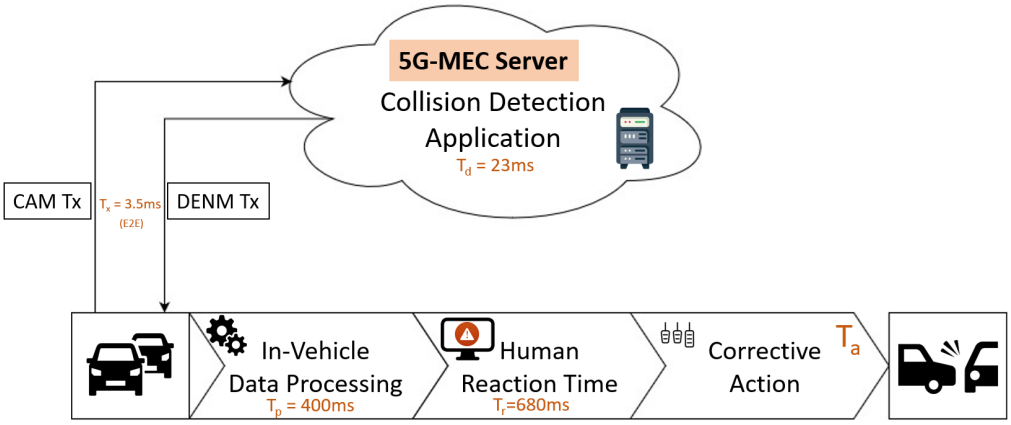

Fig. 2.5 Collision avoidance - An overview of the latency components involved.

2.7 Performance Evaluation

In this section, we evaluate the proposed approach using realistic simulated mobility data traces for the cities of Luxembourg and Monaco.

2.7.1 Data Collection and Pre-processing

To evaluate the proposed methodology, we used synthetic mobility traces from pre-built and validated traffic scenarios in the SUMO simulator representing realistic traffic demands and mobility patterns. Specifically, in this work, we extracted the mobility traces from two specific unregulated intersections in Luxembourg [63] and Monaco [64] through the SUMO's Traffic Control Interface (TraCI) library. The two

intersections from different cities enable us to validate the proposed approach under distinct multi-modal traffic demands, vehicle movements, and physical topology. Figure 2.6 illustrates the layout of the Luxembourg and Monaco intersections.

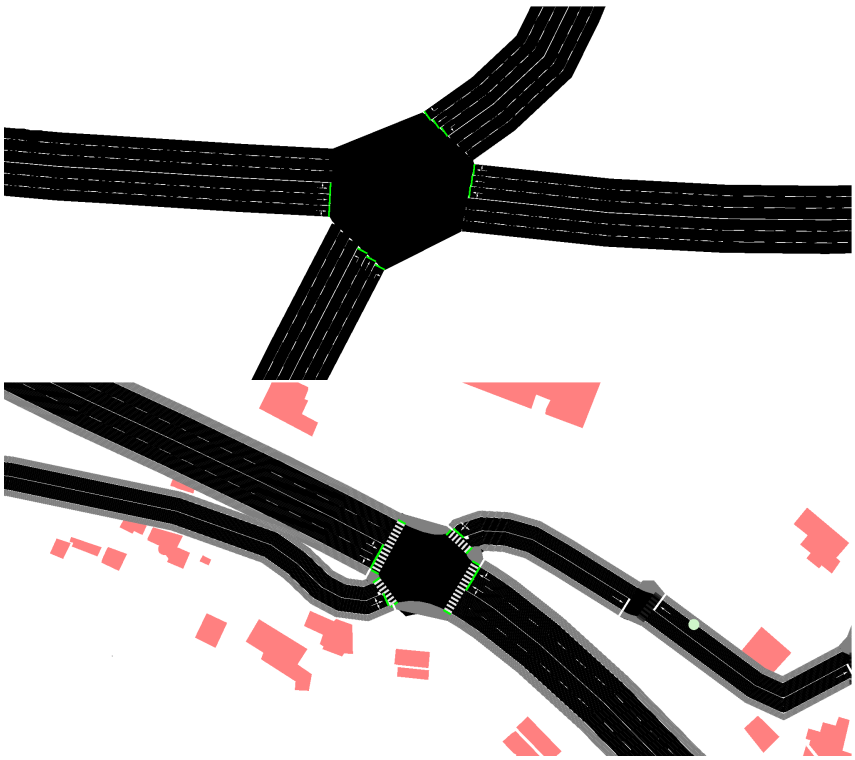

Fig. 2.6 Physical topology of the Luxembourg (top) and Monaco intersections (bottom). The layout highlights the distinct nature of the presented four-way intersections.

Luxembourg Intersection: It is a four-way intersection with a surface area of 1284 m^2 connecting roads with three lanes. Using TraCI, we collected the vehicle movements at the intersection during a time period including the early morning peak hour, i.e., 6-11 AM, to create the dataset. The dataset contains 6,132 vehicles, among which 2,035 turned at the intersection while 4,097 were traveling straight. We have also registered 197 vehicle pairs that collided at the intersection.

Monaco Intersection: It is a two-lane, four-way intersection with a surface area of 386 m^2 and with a long-stretch highway road on one side connecting the urban driveways at an intersection. Like the Luxembourg intersection, we extracted the vehicle movements during a time period that includes the early morning peak hour, i.e., 7 AM-12 PM. The Monaco dataset contains the mobility traces of 5,950 vehicles,

with 1,105 turning and 4,845 non-turning at the intersection. In addition, we have recorded 244 vehicle collision pairs at the intersection.

Pre-processing: For both datasets, the TraCI library facilitates collecting all the features mentioned in Sec. 2.4.3 at a sampling period of 100 ms. Furthermore, to regularize the input features, a feature scaling technique is used. As a result of the scaling technique, all inputs are standardized with zero mean and standard deviation equal to one, allowing the LSTM ED model to converge to a model of high accuracy. The proposed LSTM network takes the last 3 seconds of data, i.e., 30 data points, as input to predict the vehicles' future position for the next 3 seconds, i.e., 30 data points. Alternative input and output windows were also tested, with no significant improvement in the obtained results. Additionally, we are using a sliding window technique to share 29 data points between two consecutive data samples. With this setup, we have 2.4×10^6 and 5.1×10^6 sequences for the Luxembourg and Monaco intersection datasets, respectively. The datasets use 65%, 15%, and 20% as a split ratio to divide the vehicle trajectories into training, validation, and testing sets, respectively. Among the vehicle collision pairs, the testing set for the Luxembourg and Monaco intersections comprises 39 and 51 vehicle collision pairs, respectively. Both the Luxembourg and Monaco testing sets include various collision types. In this work, we induced collisions by setting all traffic lights to green, effectively turning intersections into unregulated areas where vehicles move freely without considering other traffic actors. Table 2.1 presents a summary of the number of collisions belonging to the rear, front, side, and corner categories. In Table 2.1, we define a "side collision" as a scenario where at least one of the two vehicles involved proceeds straight at the intersection, and the other vehicle collides on its side. Conversely, we classify a collision as a "corner collision" if both vehicles are turning at the intersection when the crash occurs.

Table 2.1 Collision classifications.

	Front	Rear	Side	Corner
Luxembourg	0	0	34	5
Monaco	3	6	34	8

2.7.2 Performance Results - Trajectory Prediction

Luxembourg Intersection: The proposed LSTM model comprises a single encoder and decoder layer with 300 hidden units each. The decoder layer is followed by a Time Distributed (TD) dense layer with the same hidden units as in the encoder/decoder layers. Finally, the output layers comprise two units representing the predicted (longitudinal and latitudinal) positions. The model is trained with a batch size equal to 48 and a learning rate of 0.0001. Additionally, to avoid overfitting, we have employed the early stopping criteria to end the training when the error is less than 0.001 in the validation set for two epochs. It should be noted that extensive trials with different hidden layers and batch sizes were performed, with the chosen hyperparameters presenting the best performance in terms of prediction errors. Since we are working at an unregulated intersection, the dataset contains a few traffic slowdowns for a short period of time. However, the performance of our trajectory prediction model is not majorly affected by this heterogeneous traffic behavior.

The ED metric is used to evaluate the model's performance by comparing the predicted positions with the vehicle's true positions for the next 3 seconds. As suggested in [26], the performance of Encoder-Decoder LSTM is also compared with a constant speed KF technique, which performs quite well in highways. Furthermore, to evaluate how the proposed technique compares against one leveraging information on multiple surrounding vehicles, we assess the performance of an LSTM-ED approach that not only considers the preceding vehicle, but also the closest lateral (left and right) and following vehicles. Hereinafter, we refer to such an approach as Interaction-Aware LSTM-ED. Table 2.2 presents the average ED error in predicting the vehicle position at $t+1$, $t+2$, and $t+3$ seconds. The resulting ED error shows that the proposed LSTM-ED model can predict the future trajectories for the horizon of 3 seconds with minimal deviations from the true trajectories. The proposed approach outperforms also the KF benchmark, showing that, assuming little to no modifications of the vehicles' behavior at intersections, leads to very large errors both in short and long prediction windows. When considering the best hyperparameter selection for both approaches, the proposed LSTM-ED also performs slightly better than the Interaction-Aware LSTM-ED approach. The obtained results confirm that: (i) given the small error, the performance obtained by the proposed approach is difficult to improve, even including further input data; (ii) in a busy intersection, where a change of lanes is rare, the information on the preceding vehicle is the

Table 2.2 Euclidean prediction errors for the Luxembourg dataset.

	Prediction Time [s]	Eucl. Error [m]	Interaction-Aware Eucl. Error [m]	KF Eucl. Error [m]
All vehicles	$t+1$	0.541	0.574	1.39
	$t+2$	0.684	0.748	4.06
	$t+3$	1.126	1.168	7.03
Non-turning Vehicles	$t+1$	0.528	0.541	1.35
	$t+2$	0.639	0.706	3.85
	$t+3$	1.032	1.112	6.56
Turning Vehicles	$t+1$	0.563	0.631	1.48
	$t+2$	0.763	0.82	4.45
	$t+3$	1.289	1.265	7.9

most relevant one. Specifically, in the datasets at hand, only 8.71% and 2.8% of the vehicles change lane in the surrounding of the intersection in the Luxembourg and the Monaco testing set, respectively. For all other vehicles, and for all vehicles changing lanes in all time periods during which they are not changing lanes, the surrounding vehicle information can be considered as noisy input data. Being in most cases unnecessary, surrounding vehicle information can introduce additional complexity to the LSTM model and might hinder the model's ability to generalize well, leading to overfitting on the training data, negatively impacting performance on unseen data.

Finally, Fig. 2.7 presents the CDF of the prediction errors at time $t+1$, $t+2$ and $t+3$. The CDF of the prediction errors reveals that not only the average prediction errors but also the distribution of errors are indeed small. Specifically, 91.67% of the 1-s look ahead predictions exhibit an error of less than 1 m, while 2-s look ahead presents 96.81% of errors below 2 m and 3-s look ahead presents 94.09% of errors less than 3 m.

Monaco Intersection: For the Monaco dataset, the best model utilizes a single-layer encoder and decoder with 300 hidden units each. The TD dense layer contains a hidden layer of size 300 neurons. Like the Luxembourg dataset, we set the output of the final dense layer to 2 units, representing the predicted longitudinal and latitudinal positions. We trained the model with a batch size of 32 and 0.0001 as a learning rate. Again, to avoid overfitting, we employed the early stopping criteria to stop

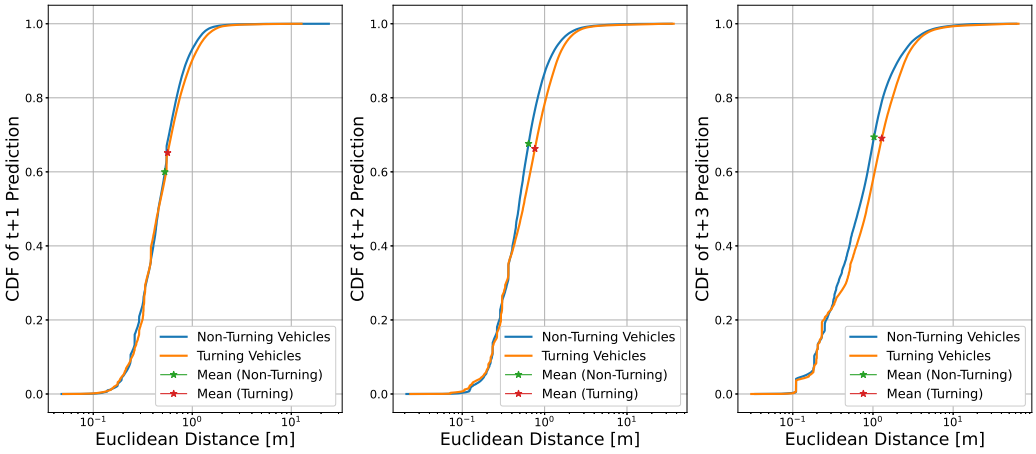

Fig. 2.7 Mean error CDF for predicted trajectories at $t+1$ (left), $t+2$ (center) and $t+3$ (right) (Luxembourg dataset).

training when the error is less than 0.001 in the validation set for two epochs. As this is also an unregulated intersection, the dataset presents quite a few gridlock situations restricting the vehicles' movements on different occasions in the five hours of collected mobility traces. This traffic scenario poses an additional challenge to the model, as it has to consider congestion while predicting traffic stoppages. Table 2.3 presents the average ED error between true and predicted trajectories for a 3-second horizon of the proposed LSTM-ED approach, of a constant speed KF [26], and of the Interaction-Aware LSTM-ED benchmark introduced above. Even with the additional complexity of increased traffic congestion, the results show that the model is able to achieve sub-meter prediction errors in all prediction windows and outperform both benchmark approaches. Figure 2.8 presents the CDF of the prediction errors for $t+1$, $t+2$ and $t+3$ seconds look ahead. In all cases, the CDF shows that the distribution of errors is small and similar to the Luxembourg dataset. Specifically, 95.41% of 1-s look ahead predictions exhibit less than 1 m error, 98.25% of 2-s look ahead have prediction error less than 2 m, and 96.49% of 3-s look ahead predictions maintain less than 3 m error.

The results from both datasets indicate that the proposed approach can excel in diverse scenarios and mobility conditions. Several other comparisons (also including real-world datasets) have been reported in our preliminary work [17]. Specifically, the proposed approach was found to perform better than (i) a Vanilla LSTM model, (ii) a Stacked LSTM-ED model, and (iii) a Bidirectional LSTM model. The results

Table 2.3 Euclidean prediction errors for the Monaco dataset.

	Prediction Time [s]	Eucl. Error [m]	Interaction-Aware Eucl. Error [m]	KF Eucl. Error [m]
All vehicles	$t+1$	0.437	0.843	1.46
	$t+2$	0.496	0.902	4.24
	$t+3$	0.751	1.178	7.26
Non-turning Vehicles	$t+1$	0.434	0.838	1.60
	$t+2$	0.491	0.864	4.59
	$t+3$	0.733	1.099	7.74
Turning Vehicles	$t+1$	0.442	0.85	1.17
	$t+2$	0.505	0.965	3.54
	$t+3$	0.782	1.308	6.28

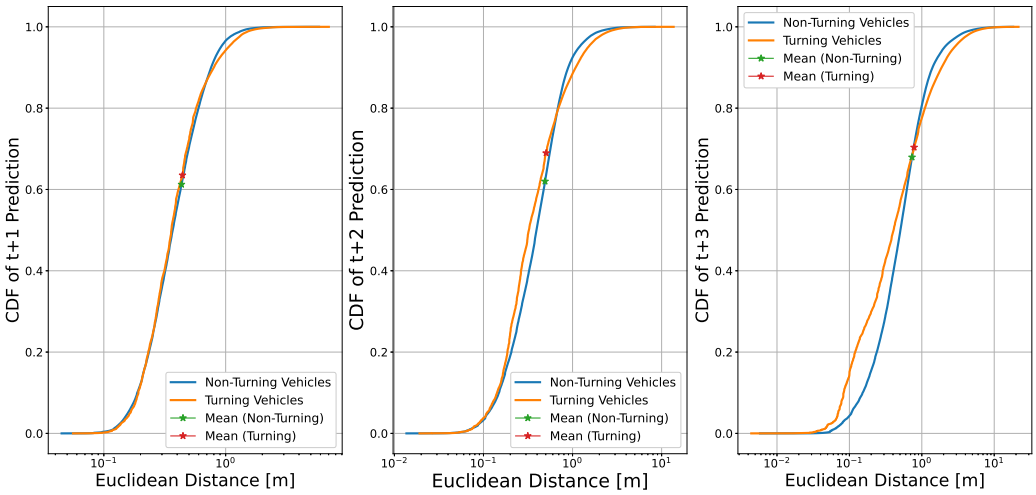

Fig. 2.8 Mean error CDF for predicted trajectories at $t+1$ (left), $t+2$ (center) and $t+3$ (right) (Monaco dataset).

indicated that the presented encoder-decoder framework outperforms other LSTM variants, although the performance of stacked LSTM variants is slightly better, it comes with additional complexity.

2.7.3 Performance Results - Trajectory Uncertainty Estimation

A prediction interval estimates the uncertainty of a point prediction with probabilistic upper and lower bounds on the output variable. For this task, as discussed earlier in Sec. 2.4.1, an LSTM-ED model trained to minimize the asymmetrically weighted sum of absolute errors is used. With the objective of characterizing the tails of the trajectory prediction's distribution, for this work, we have utilized the 0.9-quantile as the upper bound u and the 0.1-quantile as the lower bound l. Hence, the geographical regions individuated by the proposed trajectory uncertainty estimation are expected to contain in total 80% of the vehicle's future positions. Unlike the point prediction model, the deep quantile regression approach uses two models to separately predict the vehicle's longitudinal and latitudinal positions. This solution is motivated by the obtained performance, which approximates closely the expected number of the vehicles' future locations included in the prediction intervals. Further, as in point predictions, the models utilize early stopping criteria controlling the training epochs.

For both the Luxembourg and Monaco datasets, the best LSTM-ED model comprises a single encoder/decoder layer with 320 LSTM units. The decoder output is linked to a TD dense layer using 128 neurons, while the final output dense layer presents two values, one for the lower and one for the upper quantile estimation. The model uses 48 as a batch size with 0.0001 as a learning rate to minimize the quantile loss function for a specific number of epochs.

Table 2.4 presents the deep quantile regression results for the Luxembourg and Monaco datasets regarding the percentage of the vehicles' true locations captured by the computed quantiles for $t+1$, $t+2$, and $t+3$ prediction horizons on the latitudinal (λ) and longitudinal (ϕ) directions. The 0.9 and 0.1 quantiles, i.e., upper and lower bounds, present the ratio of vehicles' true positions below the predicted output, while the "in-between" column presents the percentage of vehicles' true locations in-between the two quantiles. From the results of both datasets, we can determine that the 0.9 and 0.1 quantiles included, respectively, slightly more and slightly less samples than expected. Notably, even though the tested samples are

Table 2.4 Trajectory uncertainty estimation: The percentage of vehicles' true trajectories captured by the 0.9 and 0.1 quantiles and the percentage of vehicles' true trajectories in-between the quantiles.

Dataset	Prediction Time [s]	Quantile Estimation [%]					
		0.9		0.1		In-between	
		ϕ	λ	ϕ	λ	ϕ	λ
Luxembourg	$t+1$	92.47	95.29	8.58	8.38	83.89	86.91
	$t+2$	90.58	93.38	5.14	7.09	85.44	86.29
	$t+3$	89.94	92.56	5.23	7.52	84.71	85.04
Monaco	$t+1$	99.60	99.04	1.33	7.37	98.27	91.67
	$t+2$	98.00	98.70	2.16	7.42	95.84	91.28
	$t+3$	96.00	97.67	4.42	7.69	91.57	89.98

unseen in the testing dataset, the number of vehicles' true locations included between the obtained quantiles approximate closely its expected value. Indeed, 89.24% of vehicle trajectories are between the two quantiles' output on average, as compared to the expected 80%.

2.7.4 Performance Results - Collision Detection

To identify any collision at the intersection, our detection algorithm hosted at the IM utilizes an RFC exploiting as input, at any time t, both the vehicles' predicted trajectories and the vehicles' location uncertainty estimations for the following 3 seconds. Hence, the objective of our approach is to detect a collision 3 seconds before it actually happens, i.e., as soon as the predicted data includes also the moment of collision. Even though, due to the output provided by SUMO, all pairs of colliding vehicles can be easily identified, it is possible that a collision is not easily identifiable when looking only at the true locations of the vehicles. Indeed, given the fact that the location of the vehicles corresponds to the vehicles' front bumper, it is possible that two vehicles will collide even though the distance between their true locations is always greater than zero. This may happen, e.g., if the front bumper of a vehicle collides with the rear bumper of another vehicle. In order to make all collisions uniform from the RFC's perspective, during training an input sequence concerning a pair of colliding vehicles i and k is manually assigned to the "collision" class if one the following three events occurs:

- within three seconds, vehicles i and k collide;
- $d_{ik,t+j}$ is less than d_c, with d_c being the 0.9-quantile of the minimum distance between the true locations of training colliding vehicles' pairs;
- $E[d^2_{ik,t+j}]$ is less than d_c^2, with d_c^2 being the 0.9-quantile of the minimum squared distance between the true locations of training colliding vehicles' pairs.

For the Luxembourg and Monaco datasets, d_c is set to 4.87 m and 3.51 m, while d_c^2 is specified as 23.73 m^2 and 12.34 m^2, respectively. As detailed later, this choice is driven by the fact that the ED and the expected squared ED between the estimated vehicles' locations play a crucial role in collision detection. Hence, a diverse characterization of such metrics during training would produce unbalanced results, depending on the type of collision at hand. Furthermore, during inference, a DENM is sent to the vehicles expected to collide if a collision is predicted for three consecutive time-steps, i.e., for a time interval of 300 ms. This choice for the sporadicity filter represents the numerical optimal solution to restrict the number of False Positives (FPs), while detecting all pairs of vehicles on a collision course. The performance of the obtained solution is compared to two benchmark solutions:

- an approach (named "Relative Distance") based only on the point estimates obtained in Sec. 2.4.1, that transmits an alarm to the corresponding pair of vehicles, as soon as the ED between the predicted locations of vehicles is below d_c;
- a state-of-the-art solution (named "Cooperative Intersection - Collision Warning System (CI-CWS)") [16] that uses the vehicles' velocity, acceleration, and position to predict their collision time by solving a fourth-degree polynomial approximation and that consequently transmits an alarm to the vehicles in danger.

Collision Detection Examples

Before analyzing the performance of the proposed approach in all the collisions present in the testing set, we first choose two collisions that exemplify the role

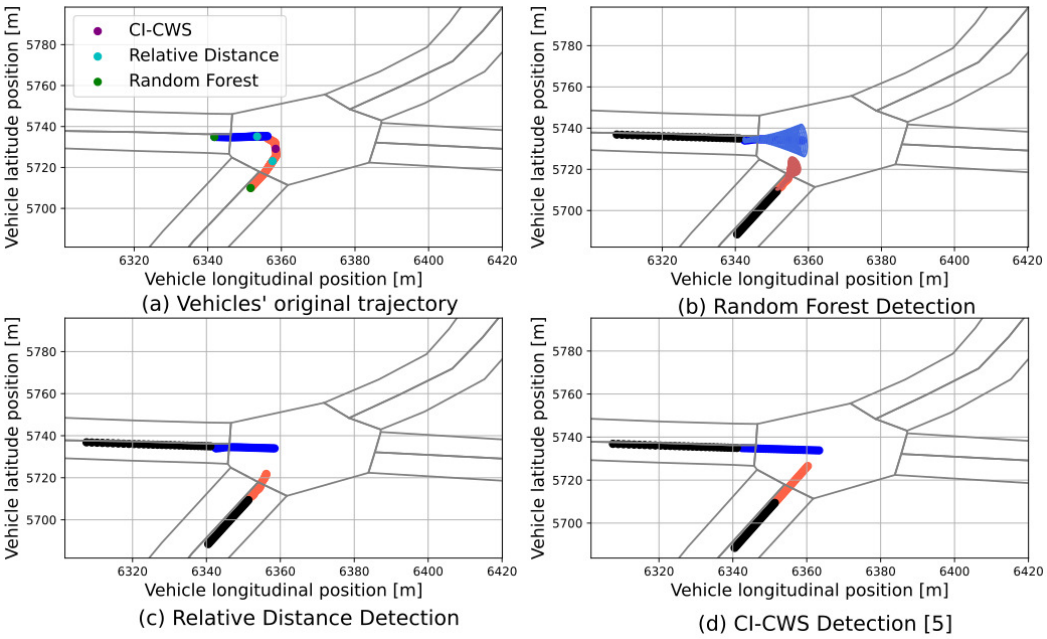

Fig. 2.9 Collision detection analysis for the Luxembourg intersection.

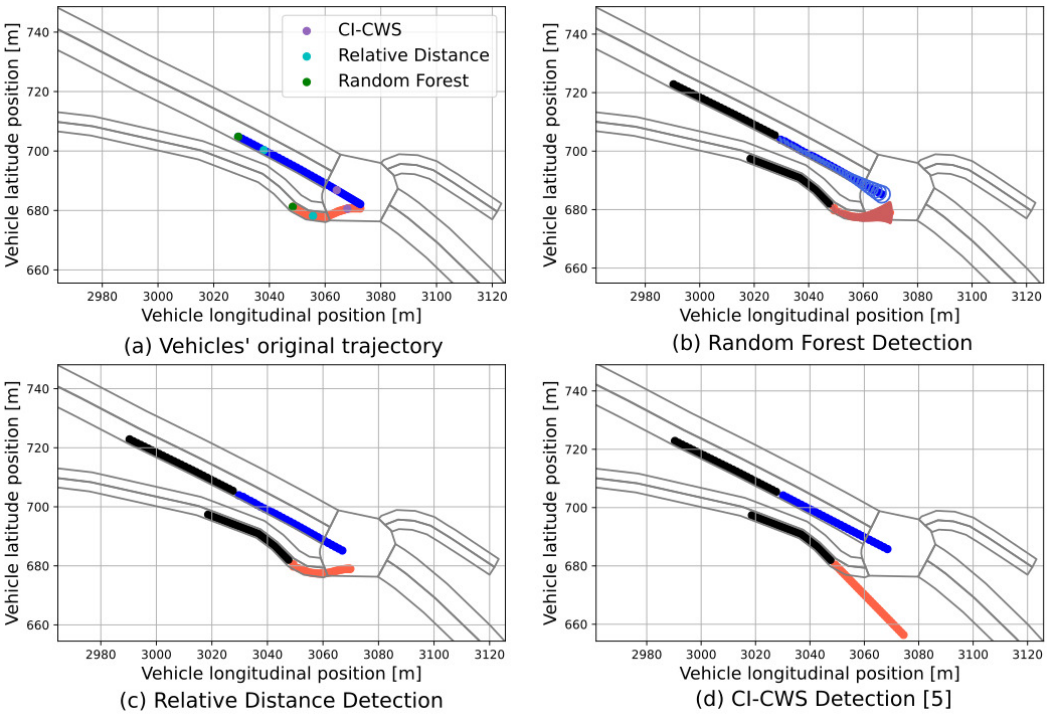

Fig. 2.10 Collision detection analysis for the Monaco intersection.

of the uncertainty estimation in the proposed framework. Figures 2.9 and 2.10 represent the two selected collisions for the Luxembourg and the Monaco intersection, respectively. Both figures refer to the moment at which our proposed methodology, based on the RFC, identifies a possible dangerous situation and sends a DENM message to the two vehicles involved. The top-left plot of each figure illustrates the trajectory traveled in the dataset by the two colliding vehicles, starting from the moment where the RFC triggers the alarm (detection and collision are interleaved by 8.8 seconds in the Luxembourg intersection, and by 3.9 seconds in the Monaco intersection). Herein, green, blue, and purple, mark the location of the vehicles when the RFC, the detection performed exploiting relative distance only, and the approach in [16], respectively, detect the imminent danger. The remaining three plots, instead, depict the 3-second trajectory predictions used by our approach, by the detection performed exploiting relative distance only, and by the approach in [16], to predict a possible collision. Furthermore, the plot referring to our approach (Fig. 2.9(b) and Fig. 2.10(b)), also reports the uncertainty estimation associated with each of the future locations predicted for the vehicles involved. Such uncertainty is represented graphically as an ellipse, with major and minor axes equal to the prediction intervals obtained with the LSTM model using a quantile loss function. In both Fig. 2.9(a) and Fig. 2.10(a), our proposed collision detection technique reacts earlier than any other approach to the imminent danger. Even though the trajectory predictions obtained by the LSTM are not showcasing a possible collision (the front bumpers are still at least 8 m away in both intersections), the associated prediction intervals are large, indicating that multiple possible driver maneuvers are available in both cases (as depicted in Fig. 2.9(b) and Fig. 2.10(b)). Based on the knowledge of dangerous situations present in the training set of the Monaco and Luxembourg intersection, the RFC used in our approach identifies as dangerous the inputs given, i.e., based on the speed of the vehicles, the predicted trajectories, and the associated uncertainty, the RFC correctly detects that the two vehicles may collide. Hence, our proposed approach can react to imminent dangers even if the vehicles trajectories appear to be safe, allowing for a smooth collision avoidance, even in the presence of human drivers. Simple trajectory predictions, as the ones used in [16] and showcased in Fig. 2.9(d) and Fig. 2.10(d), fail to provide a reliable predicted location for vehicles at intersections, and, as a result, detect possible collisions very late.

Statistical Comparative Analysis

Tables 2.5 and 2.6 summarize the predicted collision detection results for the Luxembourg and Monaco intersections, respectively. We have performed a collision check for each pair of vehicles that came as close as 50 meters, i.e., 45,674 (Luxembourg) and 69,712 (Monaco) collision checks for the two intersections analyzed. For the Luxembourg intersection, our approach, named herein "Random Forest", and the "Relative Distance" collision detection method were able to identify all the collisions that occurred in the testing set, i.e., zero False Negatives were obtained, irrespective of the type of collision. Notwithstanding, for the Monaco intersection, the "Relative Distance" approach was not able to detect all collisions with the 0.9-quantile as a distance threshold while "Random Forest" was able to identify all of them. Contrary to the introduced methods, the CI-CWS algorithm [16] is not able to detect all collisions at either intersection. This is due to the less accurate trajectory prediction used, which is also reflected by a much higher number of FPs in the simulated time frames.

Furthermore, Tables 2.7 and 2.8 show the FP count distribution with respect to the minimum distance between vehicles. For this set of results, the actual minimum distance between the vehicles' bodies is reported, rather than the distance between the true location of the vehicles. Almost all FPs are related to vehicles that, in the simulation, are actually getting very close to each other. Among the Random Forest model's FPs, at least 66% of the vehicles are in a critical region with less than a 5-meter distance between them. In some cases, especially in the Luxembourg dataset, our proposed Random Forest approach detects possible collisions, even though the true minimum distance between them never goes below 15 m. In three of the four such predicted collisions, one of the two drivers in the simulation applied the brakes to evade the collision, leading to a longer distance for the FP. As a confirmation that the trajectories used by the CI-CWS algorithm are less accurate than the ones used

Table 2.5 Comparison of collision detection methods (Luxembourg dataset).

	Collision Detection Method		
	Random Forest	Relative Distance	CI-CWS
False Positive	30	18	186
False Negative	0	0	1
True Positive	39	39	38

Table 2.6 Comparison of collision detection methods (Monaco dataset).

	Collision Detection Method		
	Random Forest	Relative Distance	CI-CWS
False Positive	34	24	419
False Negative	0	2	2
True Positive	51	49	49

Table 2.7 Distribution of distance range for FP detections (Luxembourg dataset).

Vehicle body distance range [m]	Number of occurrences		
	Random Forest	Relative Distance	CI-CWS
$0-5$	12	16	103
$5-10$	10	1	53
$10-15$	4	1	17
≥ 15	4	0	13

Table 2.8 Distribution of distance range for FP detections (Monaco dataset).

Vehicle body distance range [m]	Number of occurrences		
	Random Forest	Relative Distance	CI-CWS
$0-5$	31	23	319
$5-10$	2	0	87
$10-15$	1	1	12
≥ 15	0	0	1

by our approach, Tables 2.7 and 2.8 also show that, in many cases, the alarms raised with the considered benchmark approach concerns vehicles whose distance never drops below 15 m.

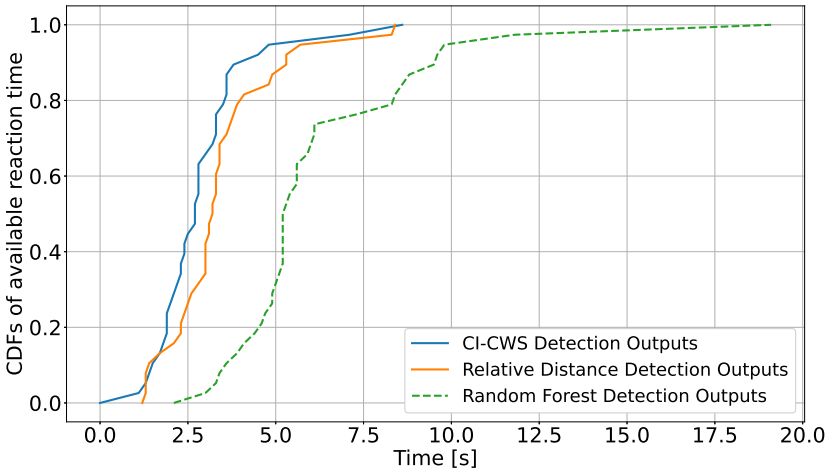

Fig. 2.11 CDF of the available reaction time (Luxembourg dataset).

The effect of accounting for uncertainty in the detection mechanism is clear when a critical metric for avoiding a collision is considered, i.e., the time between the detection of a collision and the time of the collision, which we refer to as "available reaction time" and denote with T_c in Sec. 2.6. Figures 2.11 and 2.12 showcase the performance of the Random Forest and Relative Distance approaches through the CDF of the available reaction time. For the Luxembourg intersection, Fig. 2.11 shows that the Relative Distance approach allows as low as 1.2 s to react to a collision, which is insufficient for any driver to perform a corrective action. However, with the Random Forest model, we can detect all the collisions 2 s before they occur, and the distribution of the available reaction time is better compared to the Relative Distance method for any time instance. Similarly, for the Monaco intersection, Fig. 2.12 indicates that the Random Forest model detected all but one collisions before the 2-second mark, outperforming the Relative Distance method, which could not identify all the collisions.

It is worth noting that, in this dataset, our Random Forest model detects one collision (out of 69,712 cases) 20 s or more earlier than the actual moment of collision. This happens at the time instant when two vehicles are queued at the intersection, and get on a collision course as they accelerate to cross the intersection. In the simulation,

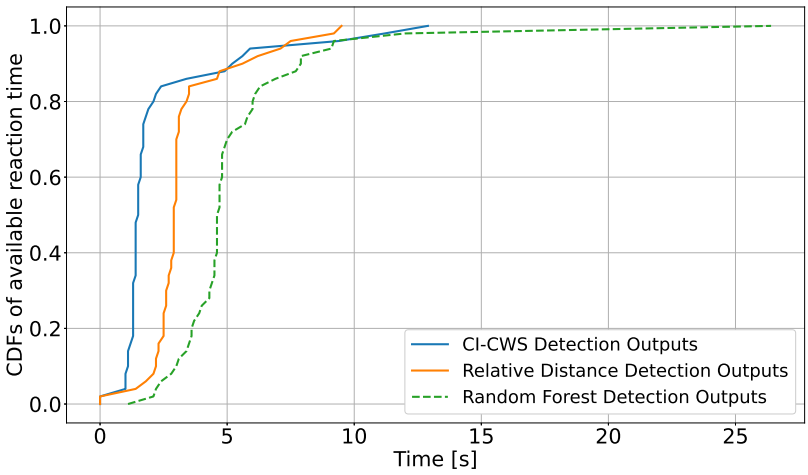

Fig. 2.12 CDF of the available reaction time (Monaco dataset).

thanks to a corrective action, i.e., braking, the vehicles are initially able to avoid the collision. Nevertheless, when they actually restart a few seconds later, they in fact collide. The Random Forest approach can detect both the dangerous situations described above. Overall, it can be concluded that the Random Forest detection method is efficient and outperforms any of the presented baseline techniques. In addition to detecting all collisions, the proposed Random Forest approach allows for a larger available reaction time for the drivers to take corrective action. This holds true for the CI-CWS algorithm as well, which also yields a much higher rate of FPs.

Finally, Figs. 2.13 and 2.14 demonstrate the importance of the input features in detecting collisions. As discussed in Sec. 2.5, the Random Forest model comprises a set of decision trees in which each decision tree is built by splitting data into subsets based on the GI. The feature importance specifies the average reduction of the GI by each feature across all trees in the Random Forest model. Amongst all features of the Random Forest model, the predicted distance and the average square distance metrics play a crucial role in detecting collisions for both datasets. This result (i) justifies the introduced manual labeling mentioned earlier; and (ii) indicates that the Random Forest model exploits both the trajectory prediction and uncertainty estimation in order to detect collisions.

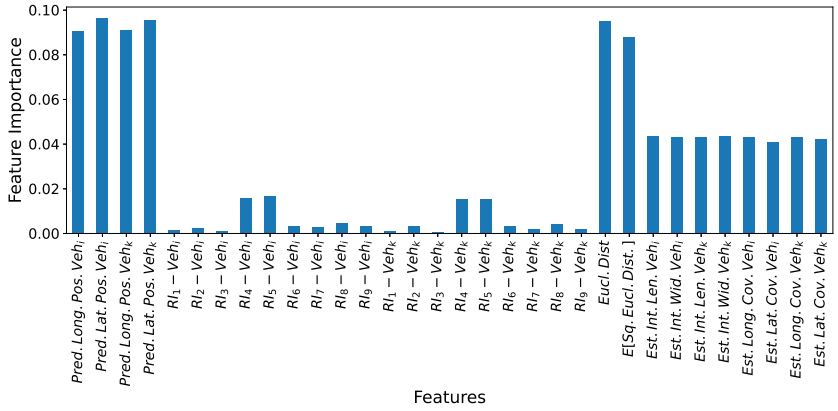

Fig. 2.13 Input feature importance for the RFC (Luxembourg dataset).

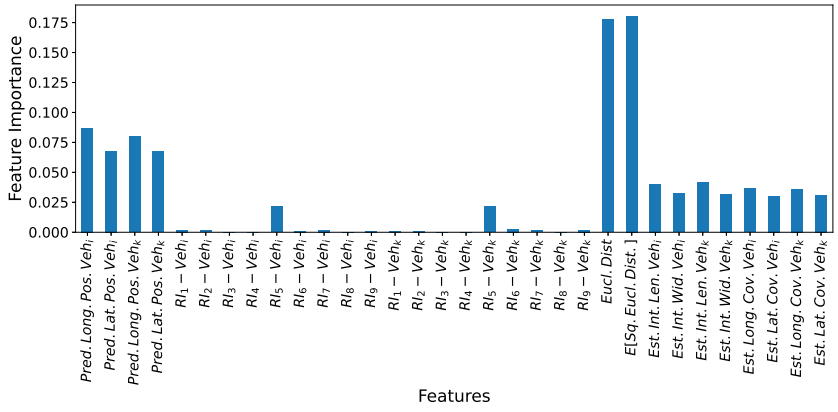

Fig. 2.14 Input feature importance for the RFC (Monaco dataset).

2.7.5 Performance Results - Collision Avoidance

As mentioned earlier, the performance of the proposed approaches in what regards collision avoidance mainly depends on the available reaction time. With the proposed approach, we can identify the collision timely compared to other approaches. Nevertheless, to assess if such detection is enough to avoid a collision, we must consider the latency pertaining to different components, such as communication, in-vehicle data processing, and driver reaction time (Sec. 2.6). Through post-processing, we tested the collision avoidance application by enforcing a continuous braking action for both vehicles till they came to a halt. For the braking action, we have used normal (4.5 m/s^2) and critical (9 m/s^2) deceleration rates to stop the vehicles. In addition, we have repeated the process for 20 simulation trials with randomly distributed latency values for each component.

Table 2.9 Vehicle's collision speed with and without corrective action (Luxembourg dataset).

Deceleration Rate	Vehicle Pair	Vehicle Speed (w/o corrective action) [m/s]	Vehicle Speed (with corrective action) [m/s]
Normal	1	$Veh_1 = 12.09$ $Veh_2 = 6.26$	$Veh_1 = 5.56$ $Veh_2 = 4.53$

Table 2.10 Vehicle's collision speed with and without corrective action (Monaco dataset).

Deceleration Rate	Vehicle Pair	Vehicle Speed (w/o corrective action) [m/s]	Vehicle Speed (with corrective action) [m/s]
Critical	1	$Veh_1 = 12.16$ $Veh_2 = 8.20$	$Veh_1 = 11.48$ $Veh_2 = 7.86$
Normal	1	$Veh_1 = 12.16$ $Veh_2 = 8.20$	$Veh_1 = 11.82$ $Veh_2 = 8.20$

At both intersections, all automated vehicles involved in a collision were able to safely reach a halt position with critical deceleration. In the case of human-driven vehicles, as shown in Table 2.9 for Luxembourg (with normal deceleration rate) and Table 2.10 for Monaco (considering both critical and normal deceleration rates), we successfully avoided all collisions but one (a side collision at each intersection). Notably, the immediate danger in both cases stemmed from an abrupt change in the driver's behavior, specifically hard braking, rather than from the type of collision

itself. Table 2.9 shows that in all cases our detection algorithm helps to reduce considerably the collision intensity. For example, the vehicle speeds are sensibly reduced when collisions happen by applying a normal deceleration rate, i.e., on average the speed was reduced by 23.1% with respect to the case when vehicles collide with no corrective action taken (i.e., without the assistance of our collision detection algorithm).

2.8 Chapter Conclusion

In this chapter, we addressed the critical problem of collision detection in complex road segments, e.g., at intersections, and presented a novel framework that effectively leverages an Intersection Manager hosted at the MEC of a 5G network, as well as V2I and I2I communications, to collect and process the relevant data. Due to its advantageous location, the Intersection Manager can obtain a holistic view of the system and, exploiting LSTM RNNs, it can provide accurate trajectory predictions and an estimation of the prediction uncertainty. Exploiting for the first time a ML-aided network-assisted collision detection methodology yielded high-quality performance, i.e., a very-low number of unnecessary FPs and an ability to detect dangerous situations well in advance, even under real-world conditions. Performance comparisons against state-of-the-art solutions and point-estimates show that the exploitation of a holistic view of the system and the inclusion of prediction intervals in the collision detection application are the critical factors for achieving the showcased results.

Chapter 3

An ML-aided Reinforcement Learning Approach for Challenging Vehicle Maneuvers

The richness of information generated by today's vehicles fosters the development of data-driven decision-making models, with the additional capability to account for the context in which vehicles operate. In this work, we focus on Adaptive Cruise Control (ACC) in the case of such challenging vehicle maneuvers as cut-in and cut-out, and leverages Deep Reinforcement Learning (DRL) and vehicle connectivity to develop a data-driven cooperative ACC application. Our DRL framework accounts for all the relevant factors, namely, passengers' safety and comfort as well as efficient road capacity usage, and it properly weights them through a two-layer learning approach. We evaluate and compare the performance of the proposed scheme against existing alternatives through the CoMoVe framework, which realistically represents vehicle dynamics, communication and traffic. The results, obtained in different real-world scenarios, show that our solution provides excellent vehicle stability, passengers' comfort, and traffic efficiency, and highlight the crucial role that vehicle connectivity can play in ACC. Notably, our DRL scheme improves the road usage efficiency by being inside the desired range of headway in cut-out and cut-in scenarios for 69% and 78% (resp.) of the time, whereas alternatives respect the desired range only for 15% and 45% (resp.) of the time. We also validate the proposed solution through a *hardware-in-the-loop implementation*, and demonstrate that it achieves similar performance to that obtained through the CoMoVe framework.

The part of the work presented in this chapter has already been published in Selvaraj, D. C.; Hegde, S.; Amati, N.; Deflorio, F.; Chiasserini, C. F., "An ML-Aided Reinforcement Learning Approach for Challenging Vehicle Maneuvers," in IEEE Transactions on Intelligent Vehicles, doi: 10.1109/TIV.2022.3224656, ©2023 IEEE.

3.1 Introduction

Recent report by the WHO indicates that nearly 1.19 million people die in road accidents annually [12]. Also, traffic congestion takes a substantial toll on public health and economy because of the polluted air, people's commuting time, and fuel consumption [65, 3]. In this context, Connected Autonomous Vehicles (CAV) can play an essential role, as they can mitigate traffic externalities, especially safety and traffic efficiency. Both vehicles and road infrastructures are increasingly equipped with sensing computational equipment to assist the driver, as well as with vehicle-to-everything (V2X) communication devices to facilitate data exchange. As a result, a CAV can gather an enormous amount of data promoting the development of Machine Learning (ML) models to further improve passengers' safety and comfort.

Among the Advanced Driver Assistance Systems (ADAS), the Adaptive Cruise Control (ACC) is one of the most popular applications in new vehicles generations, and it seemingly performs well under most car-following scenarios. However, there are few challenging scenarios where the human has to be alert and take control of the vehicle to perform a safe maneuver over the ACC [66]. One such scenario is given by the lane change maneuvers which are more common on the roads, and responsible for 7.6% of the car crashes in the US [67]. To overcome such limitations of the traditional ACC, we propose an ML-based ACC application that leverages the information collected through both sensors and communication devices. Such application can improve not only safety but also comfort and traffic efficiency since it substantially reduces the traffic shock waves that usually occur during challenging maneuvers.

More specifically, the framework we propose, called 2-Layer Learning Cooperative ACC (2LL-CACC), accounts for CAVs' road efficiency, safety, and comfort, as follows. Efficiency is measured by the headway metric – a proxy way to measure the inter-vehicle distance in a traffic stream [68–70]. Safety is expressed in terms of the longitudinal slip ratio and the Time-To-Collision (TTC), where the former is the

amount of slip experienced by pneumatic tires on the road surface, while the latter represents the time it takes for two vehicles to collide. Finally, comfort is measured through the jerk metric, defined as a rate of change in the vehicle's acceleration.

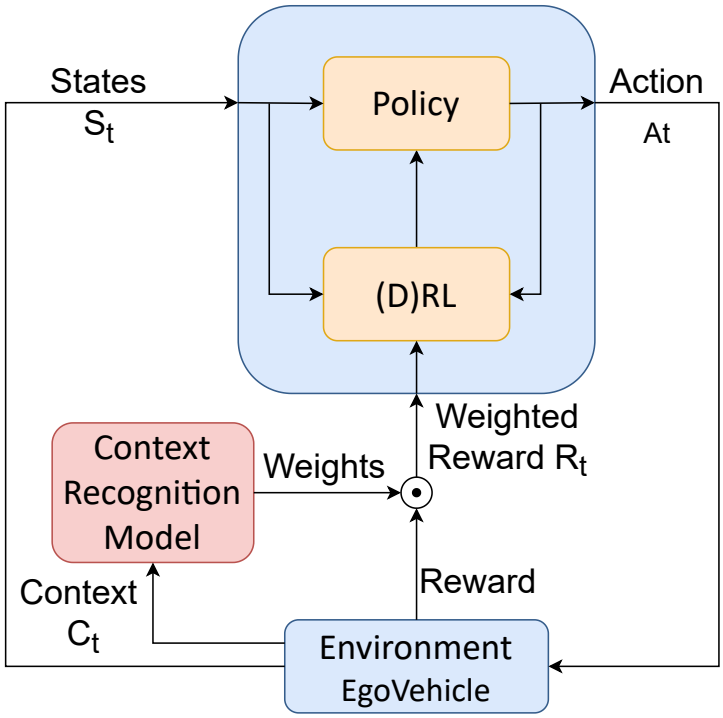

Fig. 3.1 Architecture of 2LL-CACC scheme.

As sketched in Fig. 3.1, 2LL-CACC aims at finding the best tradeoff among road efficiency, safety, and comfort by using a Deep Reinforcement Learning (DRL) where the reward function is an ML-driven weighted sum of the three metrics. The top layer hosts a Random Forest Classifier [56] to assess the current contextual information, while the lower one includes a Deep Deterministic Policy Gradient (DDPG) [71] algorithm that aims to maximize the cumulative reward by mapping the states and action through an optimal policy. Thanks to such a 2-layer ML-based approach, 2LL-CACC can adapt to the operational context and effectively selects the acceleration to adopt, thus overcoming the limitations of the traditional ACC in coping with challenging traffic situations. To demonstrate this, we primarily address road traffic scenarios where the ego vehicle follows a short-distance/low-velocity lead vehicle, as it typically occurs during cut-in and cut-out scenarios.

To evaluate the performance of our 2LL-CACC framework, we integrate it into the CoMoVe architecture, a virtual validation tool, which provides a realistic representation of communication, traffic mobility, and vehicle dynamics [5]. Specifically,

SUMO, a mobility simulator, is utilized to model traffic mobility, while vehicle dynamics are realistically modeled using the MathWorks Simulink model, and V2X communication functionalities are simulated using the ns-3 network simulator. Each simulator, specialized for its respective domain, is effectively integrated within the CoMoVe framework, providing realistic simulations. Within this framework, the Python engine serves as middleware to facilitate communication between the simulators and hosts the 2LL-CACC framework's control strategies for vehicle control. By leveraging such a comprehensive simulation tool, we obtain performance metrics related to vehicle stability, comfort, and traffic flow efficiency in handling challenging scenarios.

Our main contributions can thus be summarized as follows:

(i) We present the 2LL-CACC, an ML-aided DRL framework that employs a two-layered learning strategy to accomplish road efficiency, safety, and comfort objectives. The two layers host the Context Recognition Model and the DRL model, respectively. The role of the Context Recognition Model is to recognize the current contextual information and appropriately weigh the reward components, i.e., road efficiency, safety, and comfort. Subsequently, the weighted reward components assist the DRL model convergence by providing valuable feedback on the DRL learning process.

(ii) To achieve the above objectives and adequately represent the environment, the DRL states exploit information about the lead vehicle and its relation to the ego vehicle in terms of its lead vehicle's acceleration, headway, and relative velocity. Furthermore, vehicle stability-related states, such as longitudinal slip and road friction coefficient, are used to evaluate the vehicle's stability. As rewards, we use headway as a traffic efficiency indicator, jerk to assess comfort, and slip to ensure vehicle stability. The reward components are modeled to provide positive/negative reinforcement to the agent as feedback.

(iii) Specifically, for the aggressive driving scenarios, we have introduced a V2X-supported gradual-switching technique that facilitates the ego vehicle to change focus on the lane-changing vehicle safely and steadily. Unlike the car-following scenario, gradual switching is crucial for the early identification of lane-changing vehicles and smooth transition between the vehicles to prevent the deterioration of the target key performance indicators.

(iv) We present a detailed process flow of the Hardware-In-the-Loop (HIL) implementation that facilitates the real-time deployment of the PyTorch-based DRL agent in the dSPACE SCALEXIO AutoBox through the MathWorks environment. Also, the HIL validation demonstrates that 2LL-CACC can be actually implemented in a real-world vehicle and that it achieves a similar outcome as in the CoMoVe simulations.

Overall, the proposed system uses a content recognition model to assess the contextual information, the DRL model to drive the ego vehicle in an efficient, safe, comfortable way, and finally, the gradual switching to identify the lane-changing vehicles and adequately manipulate the DRL states to take suitable decisions.

The rest of the chapter is organized as follows: Sec. 3.2 discusses relevant previous work and highlights our novel contributions. Sec. 3.3 describes the 2LL-CACC framework and explains how V2X communication is exploited, while Sec. 3.4 and Sec. 3.5 detail, respectively, the integration with the CoMoVe framework and the process flow of the HIL implementation. Sec. 3.6 presents the performance of 2LL-CACC against state-of-the-art alternatives. Finally, Sec. 3.7 draws our conclusions and discusses future work.

3.2 Related Work

The Adaptive Cruise Control application efficiently controls the longitudinal speed of the vehicle for simple car-following scenarios, while such complex conditions like cut-in or cut-out maneuvers can be highly challenging [66, 72], as the inter-vehicle distance may change dramatically. In particular, a defensive response to the cut-in/cut-out vehicles may greatly affect traffic efficiency [73], while an overly aggressive reaction leads to collision with very high probability [67]. It is thus critical that automated/autonomous vehicles overcome the current limitations to ensure safety. To assist vehicles in such complex situations, ML techniques are widely adopted. In particular, (D)RL algorithms have been preferred to other ML approaches, since they effectively deal with uncertain and partially observable environments [74]. Several works [75–79] have explored the usage of (D)RL-based algorithms to improve vehicle performance in complex scenarios. In particular, [75] leverages a DRL-based CACC algorithm that exploits information from vehicle's RADAR and vehicle-to-vehicle (V2V) communication to maintain the desired headway with the

lead vehicle. Even though V2V communication can help to identify lane-changing scenarios beforehand, [75] only focuses on optimizing headway, thus overlooking the passenger comfort or vehicle stability. The traditional ACC also suffers from similar inadequacies, as it does not consider the environmental factors while controlling the longitudinal vehicle movements. The DRL-based framework in [77] addresses some drawbacks of [75], by using a multi-objective reward function to optimize vehicle's safety, comfort and efficiency. It also considers a continuous action space, unlike the DRL framework in [75] which can only select an action from a pre-defined discrete action space. However, Zhu et al. [77] only considers a linear model to simulate the vehicle behavior, which is often not suitable to represent a vehicle in real-world conditions. Furthermore, prior art has not considered vehicle stability under different road conditions as an objective, which is an integral part of the passenger's safety. To address this limitation, our framework utilizes longitudinal slip ratio and road friction coefficient to ensure the vehicle's stability. Even though we obtain these parameters from the simulation models, one can estimate them in real-life situations by leveraging the estimation techniques proposed in, e.g., [80–82].

Looking at the cut-in scenario, Chen et al. [78] presents a DRL framework tailored to deal with cut-in events and car-following scenarios. Chen et al. [78] uses a two-step process: (i) a deep neural network trained to predict the cut-in maneuver, and (ii) a Double Deep Q Network (DDQN) to train the DRL model for the cut-in scenario. As part of the second step, the authors develop an Experience Screening, a pre-training process where multiple DRL simulations are performed for a set of pre-defined scenarios, and the best experiences (states, actions, rewards, transition states) of each scenario are stored in an experience pool. Later, the DDQN samples the data from the experience pool for faster training convergence and generalization across different scenarios. We take this study as one of the benchmarks against which we compare our scheme. However, since the dataset used in [78] is not publicly available, we had to compared our framework to the vanilla DDQN algorithm, which is the core component of the algorithm proposed in [78]. It is also worth stressing that, differently from the framework we propose, the DDQN algorithm only supports a discrete action space. Thus, the degree of freedom to choose an appropriate action is limited, compared to 2LL-CACC.

As for non-ML-based methods, Carvalho et al. [83], Milanés and Shladover [84], Chen et al. [85] focus on improving the traditional ACC to handle the cut-in vehicles. Milanés and Shladover [84] presents a (C)ACC algorithm for platooning in

vehicle cut-in/cut-out situations. It uses a Proportional-Derivative (PD) controller, which takes relative velocity and distance between the lead and ego vehicle as input and outputs the desired ego vehicle acceleration. Nevertheless, in our preliminary work [5], we compared the performance of a similar (C)ACC controller to our proposed DRL framework, and the results showed that the DRL framework can achieve better results than the approach in [84]. Chen et al. [85] proposes instead Model Predictive Control (MPC) for cut-in maneuvers with safety and comfort objectives. However, Lin et al. [86] shows that MPC suffers high computation and time complexity, while a model-free DRL model can be trained offline and provide results promptly.

In a (D)RL framework, the representation of a reward function is critical, as it quantifies the value associated with each state and action pair and assists the agent in learning an optimal policy. Matheron et al. [87] remarks that (D)RL with sparse rewards can lead to instability and suboptimal policy convergence. Likewise, each reward component should be weighted optimally in a multi-objective DRL agent to achieve the desired outcome and faster convergence. Few studies [88, 89] use supervised reward shaping techniques to assist the sparse rewards setup, which is a different approach from ours, as we focus on predicting optimal weights for each reward component according to the current contextual information.

In general, (D)RL frameworks employ recognized simulators to validate their agent's performance [90]. In our work, we employ the CoMoVe (Communication, Mobility, Vehicle dynamics) simulation framework [91]: a sophisticated validation tool that can realistically simulate both detailed vehicle dynamics and communication models. Finally, we mention that in the preliminary version of this work [5], we assessed the framework's performance across multiple scenarios, including both urban and highway environments with varying road conditions.

With respect to [5], (i) we now develop a two-layer ML-based approach, with an ML classifier dynamically determining the setting of the weights for the three reward components in the DRL agent; (ii) the DRL framework is enhanced to identify lane-changing scenarios in advance and actuate gradual-switching strategy to the new lead vehicle assuring comfort, safety, and efficiency, and (iii) the HIL implementation demonstrates the deployability of 2LL-CACC in actual vehicles.

3.3 The 2LL-CACC Framework

In this section, we describe the 2LL-CACC scheme, which aims to learn an optimal decision-making strategy for the ego vehicle, ensuring an efficient, safe, and comfortable driving experience. As depicted in Fig. 3.1, 2LL-CACC comprises two layers: the top one hosts an ML model to access the current context and scenario characteristics; the lower one focuses on the DRL agent attributes to learn an optimal policy.

At any given time t, the ego vehicle traveling through a road traffic scenario provides information about the *environment*, specifically, neighboring vehicles and road conditions, to the DRL agent and Context Recognition model as state $s(t) \in \mathscr{S}$ and context $c(t) \in \mathscr{C}$ (resp.). Given $s(t)$, the role of the DRL framework is to attain efficient, safe, and comfortable driving 2LL-CACC by maintaining optimal speed, according to headway, slip, and jerk values through the agent's decision-making policy. Based on the input state, $s(t) \in \mathscr{S}$, the DRL agent takes action ($\mathscr{A}$) to change the behavior of the ego vehicle by either accelerating or decelerating it. As a response to the action, the agent gets a reward from the environment. The representation of states, actions, and reward in the DRL framework assists the agent in learning the optimal policy.

In our study, the reward comprises three components, namely, headway, slip, and jerk, to model efficient, safe, and comfortable driving. However, equally weighted reward components may not provide optimal feedback to the DRL agent, as, depending on the situation experienced by the ego vehicle, a component may be more important and hence should be weighted more. Examples include the case where road pavement conditions are particularly slippery and vehicle stability has to be ensured with highest priority, or the case where the ego vehicle has high driving speed and should maintain a sufficient headway. Thus, each reward component should be weighted depending upon the current context, as the latter impacts the learning process directly. To do so, it is necessary to derive the relation between features that impact the reward components and the corresponding weights. To this end, we introduce the Context Recognition Model, which leverages a Random Forest Classifier to infer such a relationship and determine the weight to be associated with each reward component based on the current context $c(t)$. Subsequently, the predicted weights are used to regulate their corresponding reward components, and

the sum of the weighted rewards facilitates the DRL model in learning an optimal policy. Fig. 3.2 depicts an overview of the proposed 2LL-CACC framework.

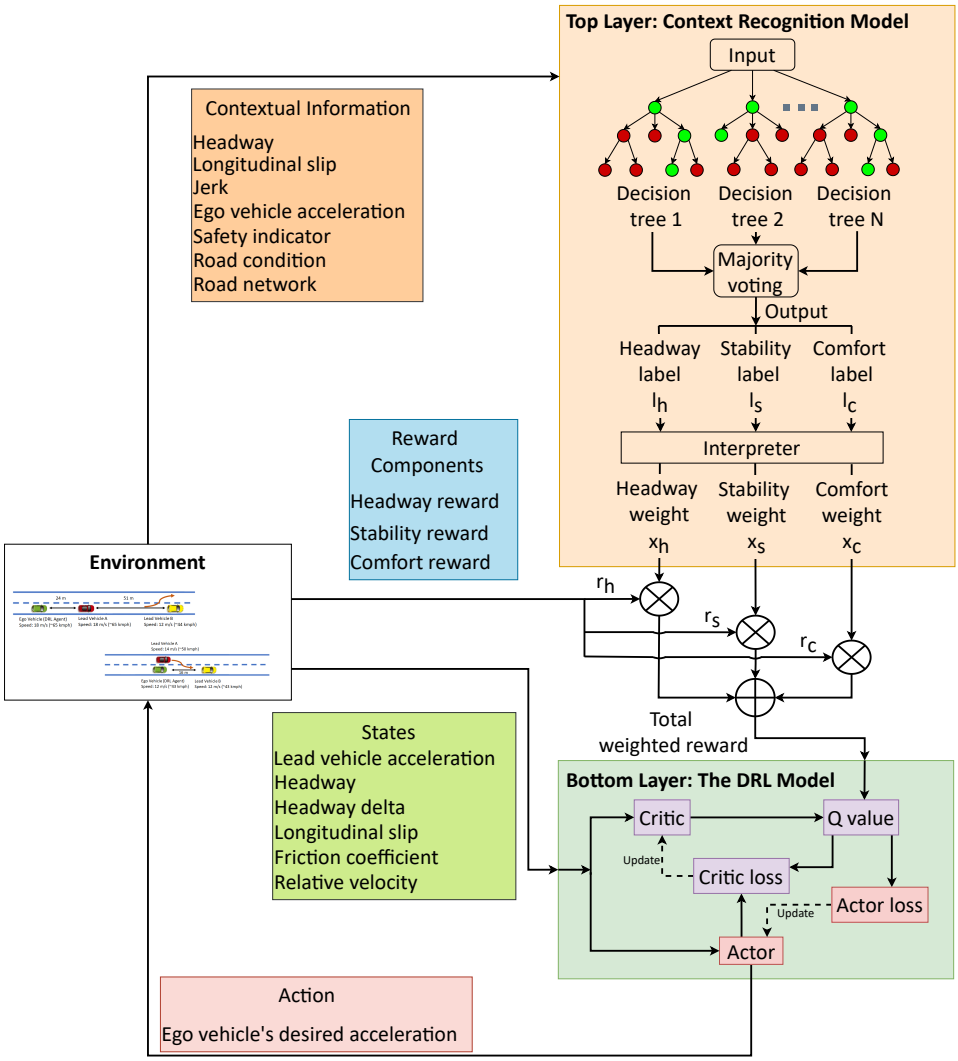

Fig. 3.2 An overview of the proposed 2LL-CACC framework.

In the following, Sec. 3.3.1 details the top layer hosting the Context Recognition model, while Sec. 3.3.2 presents the lower layer hosting the DRL framework. The notations used in Sec. 3.3.1 and Sec. 3.3.2 are summarized in Tab. 3.1.

Table 3.1 Notations.

Symbols	Description
$\mathscr{C}$	Contextual Information
$\mathscr{S}$	DRL State Space
$\mathscr{A}$	DRL Action Space
$\mathscr{Z}$	Experience Replay Buffer
x_h	Headway Reward Weight Coefficient
r_h	Headway Reward Component
x_s	Stability Reward Weight Coefficient
r_s	Stability Reward Component
x_c	Comfort Reward Weight Coefficient
r_c	Comfort Reward Component
α	Lead Vehicle Acceleration
ϑ	Headway
$\Delta\vartheta$	Headway Derivative
ξ	Longitudinal Slip
μ	Road Friction Coefficient
ν	Relative Velocity
j	Jerk
$\ddot{x}$	Ego Vehicle's Acceleration
χ	Safety Indicator, corresponding to the Time-to-Collision (TTC) value
ψ	Road Condition, based on the road friction coefficient (μ)
ω	Road Network, describes the current road traffic scenario
$\mathscr{Q}(s,a\|\beta)$	Parameterized State-Action value function with β as parameters
$\pi(s\|\eta)$	Parameterized Policy function with η as parameters

3.3.1 Top Layer: Context Recognition Model

As mentioned above, we use an ML model to predict the weights of each reward component, based on the current situation. Specifically, we uses a Random Forest Classifier (RFC) [56, 57] and train it to interpret the current contextual information through selected input features ($\mathscr{C}$), and output the optimal weight class for the headway (l_h), stability (l_s), and comfort (l_c) reward components.

Preliminaries

In general, RFC is a powerful ensemble algorithm that has been proved to handle high dimensionality problems efficiently. It suits the problem at hand particularly well, since we have a set of input features that must be mapped into three weight labels accordingly. In a nutshell, the RFC builds a set of independent decision trees and aggregates them together to get accurate predictions. Notably, the Random Forest model utilizes the bootstrap aggregation method to mitigate the high variance issue observed in single decision tree techniques. With the help of bootstrap aggregation, each decision tree samples a random subset of data from the original dataset and it uses a random subset of input features to build the tree. Because of the randomness, the decision trees are less correlated and produce better prediction outputs than a single decision tree. Essentially, a decision tree aims to split the data into homogeneous branches to determine the outcome. The tree includes two types of nodes (i) a decision splitting the data into two subsets (branches), and (ii) a leaf node representing an outcome decision. With the help of the Gini Index (GI) [57], each decision node determines a splitting criterion based on a specific feature and a threshold, which results in fewer samples of heterogeneous classes in each subset $\mathscr{H}$. At each subset, the GI is calculated as:

$$GI(\mathscr{H}) = 1 - \sum_{i=1}^{n} p_i^2 \tag{3.1}$$

where n is the total number of classes, and p_i the number of samples in subset $\mathscr{H}$ that belong to the $i-$th class normalized to $|\mathscr{H}|$'s' cardinality. Then, the weighted average of each subset's GI is used to identify the best criteria to split the data. Given

subsets $\mathscr{H}_1$ and $\mathscr{H}_2$, the weighted GI is given by:

$$GI_w(\mathscr{H}_1, \mathscr{H}_2) = \frac{n_1}{n} GI(\mathscr{H}_1) + \frac{n_2}{n} GI(\mathscr{H}_2) \tag{3.2}$$

where n_1 and n_2, represent the number of samples in subsets $\mathscr{H}_1$ and $\mathscr{H}_2$, respectively, and n is the total number of rows in $\mathscr{H}_1 \cup \mathscr{H}_2$. Similarly, GI_w is calculated for different splitting criteria, and the criterion with minimum GI_w is used to split the data. Indeed, the smaller GI value means better splitting criteria with a higher percentage of homogeneous classes in the subsets. The decision node continues to split the data till all values in each subset are homogeneous, i.e., belong to the same class. In the decision-making (i.e., inference) phase, the input data traverse the decision tree from decision nodes to the leaf node, where the majority class in the leaf node is predicted as the output label. The output label is decided based on the majority vote of all decision trees.

Context Recognition Model

The context recognition model employs the Random Forest Classifier to predict the weights of the reward components based on the current contextual information. Therefore, the RFC takes input features that concern the ego vehicles' objectives to choose the corresponding labels for the reward components. We have three reward components representing headway (r_h), stability (r_s), and comfort (r_c). To support the classification model, we discretize the weight values into 20 bins and numerically labeled each bin (e.g., label 1 represents [0.0, 0.05], label 2 [0.05, 0.1], etc.). We chose classification rather than regression algorithms because the predicted values vary considerably in regression, causing the DRL model to map a similar state-action pair with different rewards, and, hence, slowing down convergence. At a given time t, the input features headway ($\vartheta(t)$), jerk ($j(t)$), and longitudinal slip ($\xi(t)$) play a crucial role, as they represent the three main objectives of the ego vehicle. They are

formulated as:

$$\vartheta(t) = \frac{\Delta P_{lead}(t)}{\mathscr{V}_{ego}(t)} \tag{3.3}$$

$$j(t) = \frac{\ddot{x}(t) - \ddot{x}(t-1)}{\tau}, \tag{3.4}$$

$$\xi(t) = \begin{cases} \frac{\mathscr{V}^R_{ego}(t) - \mathscr{V}^W_{ego}(t)}{\mathscr{V}^R_{ego}(t)}, \ \ddot{x}(t) >= 0 \\ \frac{\mathscr{V}^R_{ego}(t) - \mathscr{V}^W_{ego}(t)}{\mathscr{V}^W_{ego}(t)}, \text{ otherwise} \end{cases} \tag{3.5}$$

where $\Delta P_{lead}(t)$ is the relative distance between lead and ego vehicle, $\mathscr{V}_{ego}(t)$ is the ego vehicle's velocity, $\mathscr{V}^R_{ego}(t)$ is the ego vehicle's tire tangential velocity, $\mathscr{V}^W_{ego}(t)$ is the ego vehicle's wheel ground point velocity, $\ddot{x}(t)$ is the ego vehicle's acceleration at time t and τ is the sampling interval of the framework. Apart from them, the ego vehicle's acceleration ($\ddot{x}(t)$) at time t helps identify the vehicle's current operating range (braking or speeding up) , which affects the objectives.

Likewise, a safety indicator is used to determine critical situations and further discount the comfort factor in case of imminent danger. In this work, we use Time-To-Collision (TTC) to identify potential collision situations, representing the time it takes for two vehicles to collide, if their speed is not modified. The TTC at time t is formulated as:

$$TTC(t) = \frac{\Delta P_{lead}(t)}{v(t)} \tag{3.6}$$

where $v(t)$ represents the relative velocity between the lead and ego vehicles at time t. Therefore, the safety indicator at time t ($\chi(t)$) is computed as:

$$\chi(t) = \begin{cases} 1, \ TTC(t) > 4s \\ 0, \ TTC(t) \leq 4s \end{cases} \tag{3.7}$$

where the 4-s threshold is set based on [92].

Furthermore, the road friction coefficient ($\mu(t)$) at time t is directly related to vehicle stability, where an abrupt acceleration change often leads to instability in

low-friction roads. The road condition ($\psi(t)$) is defined as:

$$\psi(t) = \begin{cases} 1,\ 0.7 \leq \mu(t) \leq 1 \\ 2,\ 0.4 \leq \mu(t) < 0.7 \\ 3,\ \mu(t) < 0.4\,. \end{cases} \tag{3.8}$$

Finally, ($\omega(t)$) represents the road traffic scenario at time t, as the ego vehicle' behavior may significantly vary, e.g., from urban intersections to highway scenarios. In addition, the road traffic scenarios are expressed as unique discrete values to benefit the learning process.

To summarise, at the generic time-step t, context ($c(t)$) is represented by the following features:

- headway ($\vartheta(t)$) representing the distance between ego and lead vehicle;
- longitudinal slip ($\xi(t)$) representing the ego vehicle's stability;
- jerk ($j(t)$), i.e., the comfort factor;
- ego vehicle's acceleration ($\ddot{x}(t)$);
- $\chi(t)$, a binary value that indicates whether the TTC drops below a fixed safety threshold or not;
- road condition ($\psi(t)$) describing the road friction coefficient ($\mu(t)$) in the form of slippery, wet, or dry conditions;
- road network ($\omega(t)$) representing the current road traffic scenario.

At every time-step, the RFC model takes the current context ($c(t)$) as input and predicts the optimal weight label for headway, stability, and comfort reward components. Then, labels (l_h, l_s, l_c) are converted into values (x_h, x_s, x_c) according to their discretized bins. For example, with reference to the above example about the bins' labels, if the RFC model predicts 1 as headway label l_h, the corresponding headway weight is a random uniform value between 0 to 0.05.

3.3.2 Bottom Layer: The DRL Model

We now provide a detailed description of the DRL model, starting with some preliminaries on DRL and then introducing the solution we designed.

Preliminaries

The goal of a RL model is to learn an optimal decision-making strategy by repeatedly interacting with an environment that provides positive or negative feedback as a reward for the current behavior. Its main components are: state-space ($\mathscr{S}$), i.e., a representation of the environment; action space ($\mathscr{A}$), a set of actions an agent can take to interact with the environment; rewards ($\mathscr{R}$), numerical feedback from the environment; policy ($\pi(s)$), a decision-making strategy that characterizes the mapping from states to actions; value function ($\mathscr{Q}_\pi(s,a)$), which indicates the expected future return from the state-action pair. The policy and value function facilitate the agent to take a sequence of actions that maximizes the cumulative discounted reward received from the environment.

The RL problem is generally modeled as a Markov Decision Process (MDP). At any given time step t, the MDP is represented through a quintuple, $< s(t) \in \mathscr{S}, a(t) \in \mathscr{A}, \mathscr{K}, r(s(t),a(t)) \in \mathscr{R}, \gamma >$ where $\mathscr{K}$ is the state transition probability matrix, and $\gamma \in [0,1]$ is a discount factor for future rewards. $\mathscr{K}$, specifies the probability of being in $s(t+1)$ due to action $a(t)$ taken at state $s(t)$. However, it is difficult to model the state transitions for complex problems such as vehicle dynamics, thus we adopt an actor-critic method, which is model free and exhibits low computational complexity.

In the actor-critic framework, the critic uses a function approximator to learn the value function parameters β optimizing the value function ($\mathscr{Q}(s,a|\beta)$), while the actor adopts a function approximator as well to update the policy parameter (η) in the direction suggested by the critic to optimize $\pi(s|\eta)$. In general, RL algorithms employ deep neural networks as function approximators to achieve the optimal solution, and such techniques are collectively called Deep Reinforcement Learning (DRL) methods.

In our work, we use Deep Deterministic Policy Gradient (DDPG) [71], a DRL algorithm that follows the actor-critic framework to learn both the value function and policy. The critic network with parameters β takes care of the value function estimation ($\mathscr{Q}(s,a|\beta)$) while the actor network with parameters η represents the agent's policy ($\pi(s|\eta)$). Notably, the DDPG algorithm supports the continuous action space and suits the 2LL-CACC scheme to learn the optimal acceleration profile of the ego vehicle. As the name signifies, it learns a deterministic policy

where the policy predicts the action directly, rather than predicting a set of probability distributions over the action space $\mathscr{A}$. Since the policy is deterministic, a standard normal noise with zero mean and a standard deviation of 0.1 is added to the predicted action value during training, to ensure the continued exploration of the action space.

Further, it leverages experience replay buffer and target network techniques to ensure a stable and efficient learning process. The experience replay buffer ($\mathscr{Z}$) stores the agent's experience samples as a tuple $(s(t), a(t), s(t+1), r(s(t), a(t)), d(t))$ at every step, where $d(t)$ is a binary value indicating whether the state $s(t+1)$ is a terminal state or not. In the replay buffer $\mathscr{Z}$, the next state ($s(t+1)$) is represented as s' as it holds transitions from several time steps. The algorithm then randomly draws the experience samples from the buffer during the learning process. Since the replay buffer allows using the same transitions multiple times, the experience replay buffer improves the sample efficiency and removes the correlation between them by random sampling. The target network, instead, helps stabilize the learning. In general, the value function tries to minimize the Mean-Squared Bellman Error (MSBE), which indicates the difference between the current value function and the value function with greedy policy (taking actions with maximum expected return). It is represented as:

$$L(\beta, \mathscr{Z}) = \mathbb{E}_{\mathscr{Z}}[(\mathscr{Q}(s,a|\beta) - y_t)^2]), \text{with} \tag{3.9}$$

$$y_t = r(s,a) + \gamma(1-d)\max_{a'}(\mathscr{Q}(s',a'|\beta) \tag{3.10}$$

$$a' = \pi(s|\eta) \tag{3.11}$$

$$(s,a,s',r,d) \in \mathscr{Z}\,. \tag{3.12}$$

Note that both terms in (3.9) depend on the same value function parameters β. Eventually, it causes instability in the learning process as both the terms in the (3.9) keep changing. Thus, the structure of the main actor and critic network is cloned as a target actor-and-critic network ($\mathscr{Q}'$ and π') with different parameters (β^*, η^*) to overcome training instability. The target network parameters are used in the (3.10)–(3.11) to calculate y_t and later, the MSBE. As the training progresses, the main network parameters (β, η) are gradually updated to the target network (β^*, η^*) through the Polyak averaging technique. Essentially, the critic network is trained to minimize the mean square error of the target network's expected return and value predicted by the critic network, while the actor network aims to maximize the critic

network's mean value for the actions predicted by the actor. Subsequently, the model learns to predict the actions with maximum critic value for the current state.

The DRL-based Acceleration Control

The DRL-based ACC application we develop seeks to optimally determine the ego vehicle's acceleration through system state information gathered from the ego vehicle's sensors and neighboring vehicles. The pseudo-code of the proposed scheme is presented in Algorithm 1.

States and Action: At a certain time-step t, the state space of the environment is represented by: (i) the lead vehicle's acceleration $\alpha(t)$, (ii) the headway $\vartheta(t)$, (iii) the headway derivative $\Delta\vartheta(t)$, (iv) the longitudinal slip $\xi(t)$, (v) the friction coefficient $\mu(t)$, and (vi) the relative velocity $\nu(t)$. In the state space, the preceding vehicle's acceleration is obtained through V2X communication, which is simulated with the help of the CoMoVe framework, and we assume the road friction coefficient is provided by an external estimation method running in the ego vehicle. The headway ($\vartheta(t)$) and longitudinal slip $\xi(t)$ variables are formulated through Eq. 3.3 and Eq. 3.5, respectively. The remaining state variables are formulated as:

$$\Delta\vartheta(t) = \vartheta(t) - \vartheta(t-1) \tag{3.13}$$

$$\nu(t) = \mathscr{V}_{lead}(t) - \mathscr{V}_{ego}(t) \tag{3.14}$$

where $\vartheta(t)$ and $\vartheta(t-1)$ are the headway values at time t and $t-1$ (resp.), and $\mathscr{V}_{ego}(t)$ and $\mathscr{V}_{lead}(t)$ represent ego and lead vehicle's velocity (resp.) at time t. In our model, and in contrast to prior art [75, 77], we also account for the wheel longitudinal slip ratio and road friction coefficient, to represent the vehicle stability.

Since our DRL model aims to control the ego vehicle's acceleration, action $a(t) \in \mathscr{A}$ is defined as a continuous variable. Further, the action values are bounded, in regular conditions, between [-2, 1.47] to provide a comfortable travel experience [93]. The DRL agent receives a numerical value from the environment as feedback on the agent's behavior, a numerical reward that motivates the DRL agent to satisfy the desired objective. The sampling interval of our framework is $\tau = 100$ ms long; the state observation and action decision routine are performed every τ seconds.

Algorithm 1 DRL-based Acceleration Control

Randomly initialize critic network $\mathscr{Q}(s,a|\beta)$ and actor $\pi(s|\eta)$ with weights β and η
Initialize target network $\mathscr{Q}'$ and π' with weights $\beta^* \leftarrow \beta$ and $\eta^* \leftarrow \eta$
Initialize replay buffer $\mathscr{Z}$
for episode$= 1, M$ **do**
  Receive initial observation state $s(1)$
  for $t = 1, T$ **do**
    Select action $a(t) = \pi(s(t)|\eta) +$ random normal noise according to the current policy and exploration noise
    Execute action a_t and observe new state $s(t+1)$, rewards of each component ($r_h(s(t),a(t))$, $r_s(s(t),a(t))$, $r_c(s(t),a(t))$), environment status $d(t)$
    Get weights (x_h, x_s, x_c) from Context Recognition Model
    Calculate the reward $r(s(t),a(t))$ based on the weights and their reward components
    Store transition $(s(t),a(t),s(t+1),r(s(t),a(t)),$ $d(t))$ in $\mathscr{Z}$
    Sample a random mini batch of N transitions $(s_i,a_i,s_{i+1},r_i,d_i)$ from $\mathscr{Z}$
    Set $y_i = r_i + \gamma\mathscr{Q}'(s_{i+1},\pi'(s_{i+1}|\eta*)|\beta^*)$
    Update critic by minimizing the loss:
    $L = \frac{1}{N}\sum_i (y_i - \mathscr{Q}(s_i,a_i|\beta))^2$
    Update the actor policy using the sampled policy gradient:
    $\nabla_\eta J \approx \frac{1}{N}\sum_i \nabla_a \mathscr{Q}(s,a|\beta)|_{s=s_i,a=\pi(s_i)} \cdot \nabla_\eta \pi(s|\eta)|s_i$
    Update the target networks:
    $\beta^* \leftarrow \rho\beta + (1-\rho)\beta^*$
    $\eta^* \leftarrow \rho\eta + (1-\rho)\eta^*$

Reward Components: The reward function comprises three components: headway (representing traffic flow efficiency), stability (representing safety), and comfort, each component's value ranging in [-1,1]. More formally, we have:

$$\begin{aligned} r(s(t),a(t)) = x_h \cdot r_h(s(t),a(t)) + x_s \cdot r_s(s(t),a(t)) \\ + x_c \cdot r_c(s(t),a(t)) \end{aligned} \tag{3.15}$$

where x_h, x_s, x_c are the weight coefficients obtained from the Context Recognition Model, which dynamically vary according to the current state, and $r_h(s(t),a(t))$, $r_s(s(t),a(t))$, $r_c(s(t),a(t))$ are, respectively, the headway, vehicle stability, and comfort reward component at time step t. The three reward components are detailed below.

Headway reward component: Headway is a proxy way to measure the gap between two successive vehicles, i.e., ego and lead vehicle [75, 77], and it can be calculating using Eq. 3.3. Following [75], we set the ideal headway to secure a safe and efficient inter-vehicle distance to 1.3 s, while headway values lower than 0.5 s imply a possible risky situation between the ego and the lead vehicle. Traffic efficiency is further ensured by adding the relative velocity between ego and lead vehicle to the headway term, as in Eq. 3.17. The headway term remains unchanged if the ego and lead vehicle travel at the same speed. If instead the ego vehicle travels faster or slower than the lead vehicle, its velocity affects the relative distance, hence the headway. Thus, the addition of the relative velocity helps regulate the ego vehicle's acceleration proactively. Compared to our preliminary work [5], the addition of relative velocity to the state space ($\mathscr{S}$) and reward calculation assists the DRL model to consider the neighboring vehicles traveling at different velocities effectively.

The headway reward component ($r_h(s(t),a(t))$) is modeled as a Log-Normal distribution function with mean ε and variance σ, equal to 0.285 and 0.15, respec-

tively:

$$r_h(s(t),a(t)) = 2\cdot(F_h - 0.5), \quad \text{with} \tag{3.16}$$

$$\varphi(t) = \vartheta(t) + \vartheta(t)\cdot v_{norm}(t) \tag{3.17}$$

$$v_{norm}(t) = \frac{v(t) - \mathscr{V}_{min}}{\mathscr{V}_{max} - \mathscr{V}_{min}} \tag{3.18}$$

$$F_h = 0.4944 \cdot f_{lognorm}(\varphi(t)|\varepsilon,\sigma), \tag{3.19}$$

$$f_{lognorm}(x|\varepsilon,\sigma) = \frac{1}{x\cdot\sigma\sqrt{2\cdot pi}}\exp\frac{(-(\ln x - \varepsilon)^2)}{2\sigma^2} \tag{3.20}$$

where $\mathscr{V}_{min}$, $\mathscr{V}_{max}$, pi are the parameters of the headway reward component and their respective values are defined in Tab. 3.4. Such headway reward function reaches $+1$ for $\varphi(t) = 1.3$ s, and -1 for $\varphi(t) = 0.5$ s with the specified parameter values.

Comfort reward component: It is associated with the rate of change of acceleration with time, i.e., jerk $j(t)$. According to [93], the best comfort is observed when the absolute jerk value is below 0.9 m/s^3, while values above $1.3\,m/s^3$ indicate aggressive driving. Therefore, the reward function decreases gradually with the jerk value rising from $0.6\,m/s^3$ to $2\,m/s^3$, and it saturates with the minimum reward of -1. These requirements are formulated as a mathematical expression using Polynomial Curve Fitting. It is worth noting that the passengers' safety supersedes the comfort factor during critical situations. Thus, we consider the TTC as a safety indicator to identify dangerous situations. The comfort reward is neglected when TTC≤ 4 s, to prioritize safety during such situations. The comfort reward is formulated as:

$$r_c(s(t),a(t)) = \chi(t)\cdot f(jerk), \quad \text{with} \tag{3.21}$$

$$f(jerk) = polyfit(|j(t)|, 9) \tag{3.22}$$

where $\chi(t)$ is the safety indicator declared in (3.7), which is used to discount comfort in the case of danger.

Stability reward component: It is valued in terms of the slip, i.e., the maximum tractive force of a pneumatic tire on road surfaces. Based on experimental data, an absolute longitudinal slip value below 0.2 is considered a stable condition. Thus, the stability reward gives a maximum reward of $+1$ for zero slip, and a negative reward for slip values over 0.2, indicating that the vehicle is not in the stable region. The

stability reward is given by a tanh function as:

$$r_s(s(t), a(t)) = 2.0099 \cdot F_s + 1 \quad \text{with} \tag{3.23}$$

$$F_s = \tanh(-3 \cdot |\xi(t)|) \tag{3.24}$$

Simulation Environment: To learn the desired behavior, the DRL agent has to interact with an environment that simulates the neighboring vehicle's behaviors and road conditions. Our study uses CoMoVe, a comprehensive simulation environment that can accurately simulate all vehicles' dynamics and sensor arrays, V2X communication, and road conditions to facilitate the DRL agent's learning process. Sec. 3.4 explains in detail the integration of the DRL model with the CoMoVe simulation framework.

Importance of V2X Communication: The role of communication in the 2LL-CACC scheme is crucial as it is responsible for collecting lead vehicle's acceleration to form the state space ($\mathscr{S}$) in the DRL model. Furthermore, in the cut-in and cut-out scenarios, the lead vehicle often falls in the blind spot of the ego vehicle's sensor array, resulting in late detection of the lead vehicle's presence and in uncomfortable maneuvers to avoid a potential collision. Through V2X communications, instead, the ego vehicle can periodically receive information on the lead vehicle's movements (e.g., yaw rate and position) and recognize in advance its intention to change lane, even before the sensor array can perceive it.

To fully benefit from such additional information, we introduce a *gradual switching technique* that allows the ego vehicle to gradually switch the attention to the cut-in vehicle, or the vehicle ahead of the cut-out vehicle, to perform moderate evasive maneuvers without hindering the passengers' safety and comfort. Denoting with $\mathscr{Y}_e$ and $\mathscr{Y}_c$, respectively, the lateral position of the ego vehicle and that of the generic lane-changing vehicle, we define $\Delta\mathscr{Y}_c$ as:

$$\Delta\mathscr{Y}_c = \mathscr{Y}_c - \mathscr{Y}_e\,. \tag{3.25}$$

Then we let the ego vehicle trigger a lead-vehicle switch whenever $\Delta\mathscr{Y}_c$ crosses a certain threshold. Specifically, in the cut-out scenario, the ego vehicle switches to the new lead vehicle when $\Delta\mathscr{Y}_c > M_1 \cdot \mathscr{Y}_0$ with $\mathscr{Y}_0$ being the lane width (i.e., 3.3 m) and M_1 a scaling factor. In the cut-in scenario, instead, the ego vehicle takes as new lead vehicle the one cutting-in when $\Delta\mathscr{Y}_c < M_2 \cdot \mathscr{Y}_0$ with M_2 being a scaling

factor. The values of the scaling factors we used are presented in Tab. 3.4. Since the gradual switching indicates slowly shifting the focus from one vehicle to another, this scaled variable suits well our methodology and actual implementation. Next, let us introduce a normalized variable $\wp$ scaled between 0 and 1 according to the specified thresholds.

As long as $\Delta\mathscr{Y}_c$ is less than the threshold value in the cut-out scenario, we compute the headway to be fed to the DRL model as:

$$\vartheta(t)=\frac{\wp\Delta P_c(t)+(1-\wp)\Delta P_n}{\mathscr{V}_{ego}} \tag{3.26}$$

where n is the new lead vehicle, identified by the ego vehicle based on the values of yaw rate received from its neighbors through V2X communication. Similarly, as long as $\Delta\mathscr{Y}_c$ is greater than the threshold in the cut-in scenario, the headway input to the DRL model is:

$$\vartheta(t)=\frac{\wp\Delta P_p+(1-\wp)\Delta P_c}{\mathscr{V}_{ego}} \tag{3.27}$$

where p is the previous lead vehicle. Specifically for cut-in situations, the *gradual switching technique* incorporates an adaption of the Automated Lane Keeping System (ALKS), UN Regulation No. 157 [94]. As per the regulation suggestions, the *gradual switching technique* is refined to wait for at least 0.72 seconds before reacting to the cut-in vehicle to avoid considering any temporary lateral position changes in the social vehicle. Subsequently, if the social vehicle's lateral position continues to change for more than the specified threshold, the proposed switching technique will change the focus gradually to the lane-changing vehicle, considering it a cut-in situation. In addition, we monitor the ego vehicle's Time-to-Collision (TTC) concerning the social vehicle and the social vehicle's lateral position during the lane-changing phase to handle aggressive cut-in situations. The ego vehicle will switch its focus entirely to the lane-changing social vehicle if any of the conditions are met:

- TTC becomes lower than the $TTC_{LaneIntrusion}$ [94] threshold, defined as:

$$TTC_{LaneIntrusion}=\frac{v}{2\cdot M_3}+M_4 \tag{3.28}$$

  where v is the relative velocity between the lane-changing social vehicle and the ego vehicle, while M_3 and M_4 are scaling factors accounting for maximum deceleration rate and reaction time, respectively;

- TTC is less than 4 s [92];

- The social vehicle is 30 cm [94] inside the ego vehicle's lane.

The relative velocity between the ego vehicle and the lead vehicle (v) is computed similarly, and further used by the DRL framework to control the ego vehicle movements. In summary, the gradual switching technique is specifically introduced to handle challenging vehicle maneuvers such as cut-in and cut-out. In fact, it influences the DRL model variables to advise the agent to accommodate the lane-changing maneuvers efficiently. The results reported in Sec. 3.6.2 further validate the importance of V2X communication in the proposed framework.

3.4 Integrating the DRL Model in CoMoVe

The CoMoVe framework [91], depicted in Fig. 3.3, combines widely used simulators in each domain (mobility, communication, and vehicle dynamics) and makes them to interact efficiently. It combines: (i) SUMO, a traffic simulator for vehicle mobility, (ii) ns-3, a network simulator to model V2X communications, (iii) the MATLAB/Simulink module modeling the vehicle dynamics and the vehicle on-board sensors while Driving scenario designer converts the vehicle information from SUMO to MATLAB format to support the on-board sensing, (iv) a Python Engine as a middle-man to handle the information flow between the modules and the host control strategies.

CoMoVe leverages SUMO's TraCI library, ns3's Python bindings, and MATLAB's Python Engine to write complete Python simulation scripts and ensure efficient interactions between them. Consequently, the Python Engine is the CoMoVe's core: it can access information from each simulator and hosts the 2LL-CACC framework to control the ego vehicle movement. As for the DRL state components, the lead vehicle's acceleration value ($\alpha(t)$) is received through the ns3 V2X communication model, while the vehicle sensor model output helps calculate the headway ($\vartheta(t)$), headway derivative ($\Delta\vartheta(t)$), and relative velocity $v(t)$ values. The longitudinal slip ($\xi(t)$) and friction coefficient ($\mu(t)$) are obtained through the Simulink Vehicle Dynamic model. The DRL model's action (desired acceleration) is used as a reference signal to the ego vehicle's lower level controller in the Vehicle Dynamics Model. A pure electric vehicle with a 14-Degree-of-Freedom (DoF) mathematical model and rear in-wheel motors are utilized to characterize the vehicle dynamics.

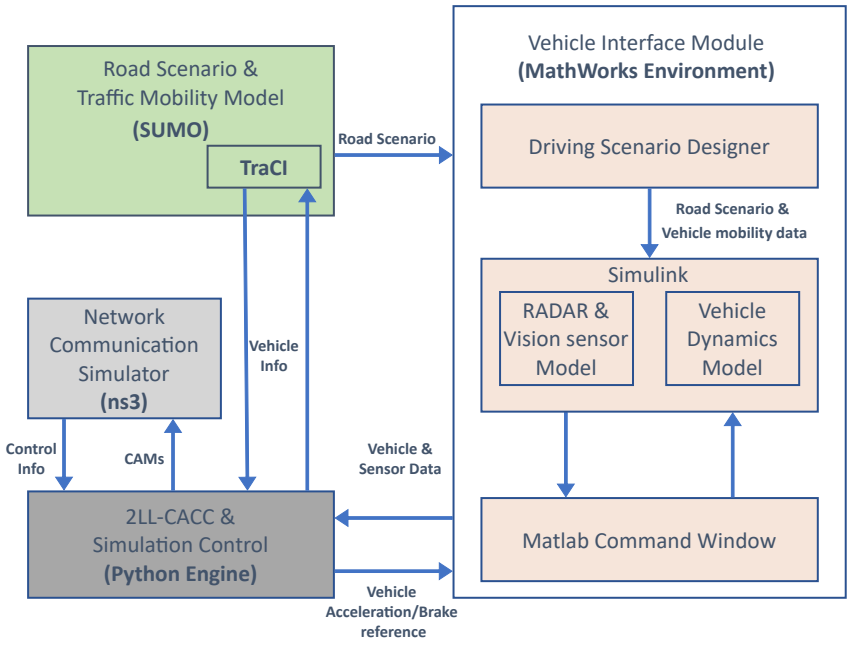

Fig. 3.3 Architecture of the CoMoVe framework.

Using the CoMoVe framework, in Sec. 3.6 we show how 2LL-CACC provides a safe, comfortable, and efficient driving experience in challenging road scenarios.

3.5 Hardware-in-the-loop Implementation

Testing and validating ADAS subsystems in assembled vehicles incurs significant overhead in terms of time, safety, and cost. Thus, HIL simulations have emerged as a convenient way to virtually validate the system in a wide range of test scenarios during the vehicle development process. In general, HIL simulations validate control algorithms through a real-time virtual environment encompassing the vehicle's functionalities.

To validate our approach, we perform HIL simulations using dSPACE real-time systems comprising modular and robust platforms for testing autonomous driving. Notably, HIL simulations demonstrate the deployable nature of the proposed controller with a similar outcome in the actual vehicle.

More specifically, in the proposed framework, the implementation of HIL simulation involves two main steps:

(i) Conversion of the pre-trained Python DRL model into MATLAB/Simulink supported DRL model, and

(ii) Generation of the DRL agent.

Since the vehicle sensor and dynamics models are simulated in the MathWorks environment, the Python-based DRL agent must be converted into the MATLAB-supported DRL agent for auto code generation. Note that the network simulator (ns3) and traffic mobility model (SUMO) are not part of the HIL implementation, as they do not support the auto code generation process. Instead, the HIL simulation uses the mobility traces of the lead vehicles' and is assumed to be equipped with the vehicle's V2X communication On-Board Unit (OBU) to receive the lead vehicle information.

As specified in Sec. 3.3, we embedded the DDPG algorithm into the CoMoVe framework through the Python Engine. Specifically, the PyTorch machine learning framework is used to build and train the DDPG algorithm's neural network model. In general, the Open Neural Network Exchange (ONNX) format is used to achieve interoperability between different ML frameworks like TensorFlow, PyTorch, and MATLAB. However, the support of the ONNX format in MATLAB is limited to the 3D input layers, i.e., images, so the direct usage of the PyTorch model in MATLAB is unattainable. As a workaround, we replicated the PyTorch neural network structure in MATLAB, and its learnable parameter values are transferred to the MATLAB model. In essence, the learnable parameters are the optimized weights and biases of the neural network that are learned to achieve the desired outcome.

Then, the MATLAB DRL model is converted into a function to evaluate the learned policy of the DRL agent. At a given time step t, the generated function can predict the action ($a(t)$) based on the state ($s(t)$), as per the trained optimal policy. Subsequently, the function is integrated into the Simulink model through the "MATLAB Function" block, so that it can directly predict the control action for the ego vehicle in Simulink. Notice that the generated function does not support further learning and can only be used to perform inference.

Finally, we validated the performance of PyTorch and MATLAB DRL agents by transferring multiple model parameters from PyTorch to MATLAB. Tab. 3.2 presents the observed Root Mean Square Error (RMSE) of the headway parameter concerning the PyTorch and MATLAB DRL agents. The validation results indicate that the effect

of the model conversion on the output values is negligible, thus firmly confirming the correct transfer of the PyTorch DRL model to MATLAB. In the second step, the dSPACE's Real-Time Interface (RTI) links the Simulink software with the dSPACE hardware. In particular, the RTI extends Simulink's C code generator to execute the Simulink software model in real-time hardware. Later, the generated C code is loaded into the dSPACE SCALEXIO AutoBox to perform the HIL simulation.

Table 3.2 Model Conversion validation.

Headway RMSE (Ideal = 1.3 s)		
Validation Models	PyTorch	MATLAB
1	0.1766	0.1799
2	0.184	0.186
3	0.2165	0.2187
4	0.1926	0.1951
5	0.1645	0.1665

dSPACE simulates two main subsystems: Controller and Plant. The controller subsystem provides the desired acceleration for the ego vehicle based on the current states; the Plant subsystem instead simulates the ego vehicle dynamics, sensors, and lead vehicles' mobility traces. In this work, we used a dSPACE SCALEXIO AutoBox hardware equipped with Intel Core i7-6820EQ, the quad-core processor. The results of the HIL simulations are discussed later in Sec. 3.6.2.

Fig. 3.4 shows the process flow of our HIL implementation (left) and the structure of the HIL simulation platform in dSPACE SCALEXIO (right).

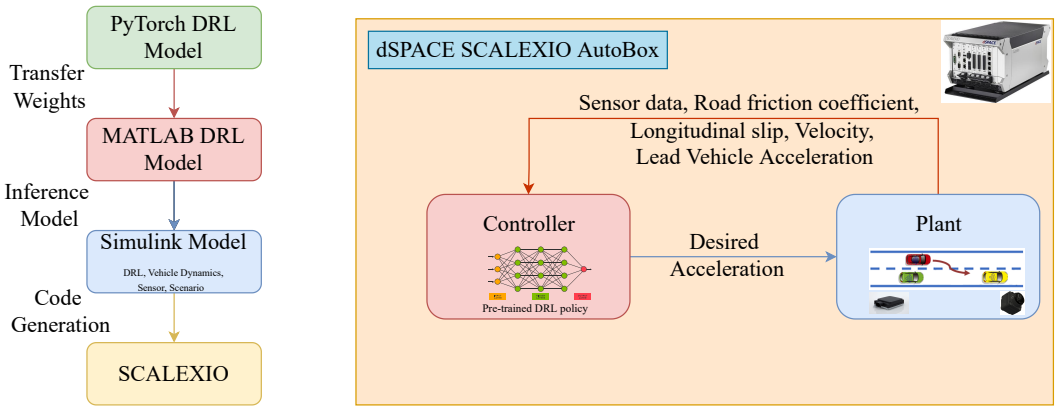

Fig. 3.4 Process flow of HIL implementation (left); dSPACE SCALEXIO Simulation process (right).

3.6 Performance Evaluation

This section first introduces the realistic settings, under which we derive the performance of the 2LL-CACC, as well as the state-of-the-art technique that we consider as benchmark (Sec. 3.6.1). Then, using both the CoMoVe framework and the HIL implementation, it presents the obtained performance results in relevant, practical scenarios (Sec. 3.6.2).

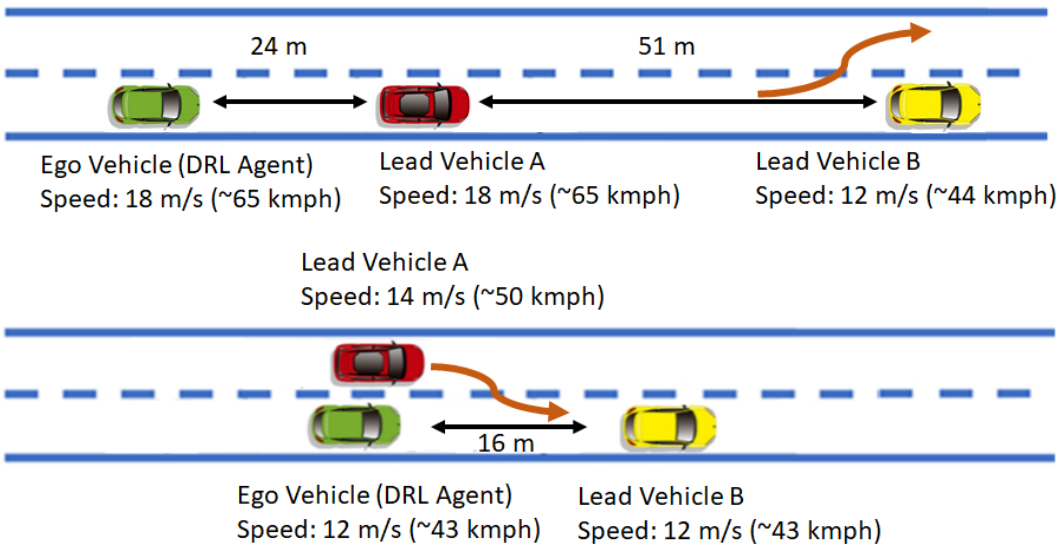

Fig. 3.5 Cut-out (top) and cut-in (bottom) scenarios.

3.6.1 Reference Scenario and Test Cases

We explore two highly challenging highway driving scenarios where a lead vehicle cut in and out from its current lane, exposing the ego vehicle to unclear or critical situations. In both scenarios, as shown in Fig. 3.5, the lead vehicle (LV), i.e., LV-A in the cut-in and LV-B in the cut-out scenario, is in the sensor array's blind spot. Thanks to V2X communication, the ego vehicle becomes aware of the LV's lateral movements and employs the gradual switching technique to change its focus between the two lead vehicles. Note that the ego vehicle's sensor array takes over the perception control only once the new LV is in its field of view. Fig. 3.6 illustrates the lateral movement of the LVs in the considered mobility scenario.

Table 3.3 DRL Hyperparameter values.

	DDPG	DDQN
Action Space	(-2, 1.47)	[-2.0, -1.6, -1.2, -0.8, -0.4, 0.09, 0.4, 0.8, 1.2, 1.47]
Hidden Layers	3 (actor, critic each)	6
Neurons	64	64
Actor Learning Rate	0.0001	0.0001
Critic Learning Rate	0.001	-
Target Network Update	0.001 (ρ)	100 (steps)
Replay Buffer Size	50000	500000
Mini-Batch Size	48	64

Table 3.4 Parameter values.

Parameters	Values
$\mathscr{V}_{min}$	$0\,m/s$
$\mathscr{V}_{max}$	$36\,m/s$
τ	100 ms
pi	3.142
M_1	0.5
M_2	0.8
M_3	$6\,ms^{-2}$
M_4	0.35 s

In addition, we compare the proposed framework to the state-of-the-art method in [78], whose implementation cannot be entirely reproduced because their dataset is not available for public use. We therefore consider the vanilla DDQN algorithm [95], which is the core component of the method used in [78]. Tab. 3.3 shows the

hyperparameter values we used for the DDPG and DDQN algorithms and Tab. 3.4 presents the parameter values of the different reward components. Sec. 3.6.2 discusses the behavior of the ego vehicle equipped with the 2LL-CACC, and its relative performance with the case where at the bottom layer we use the state-of-the-art vanilla DDQN algorithm instead of the proposed DRL model, in the challenging cut-in and cut-out maneuvers. For brevity, in the plots shown in the following we refer to the considered benchmark as DDQN.

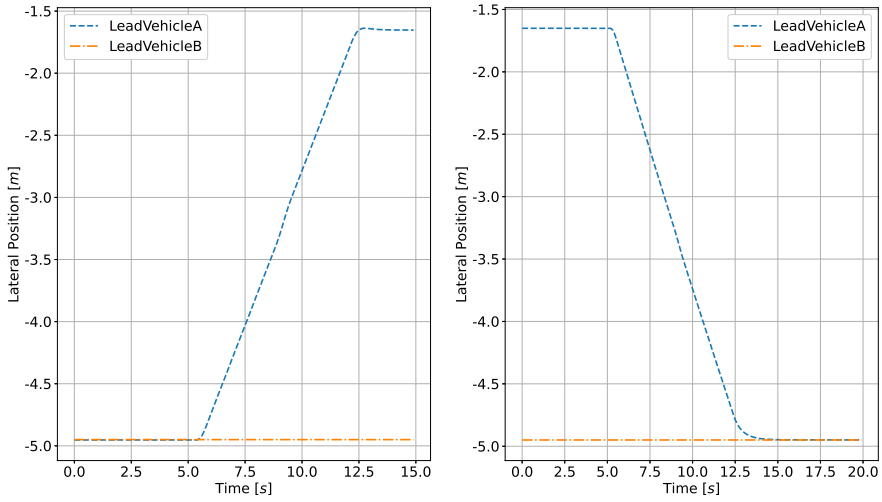

Fig. 3.6 Lead vehicles' lateral movement in the cut-out (left) and cut-in (right) scenarios.

3.6.2 Results

We start by discussing the performance of 2LL-CACC in the cut-out scenario. The top left and top right plots of Fig. 3.7 present the velocity and acceleration profile of the vehicles. The bottom left and right plots show instead the vehicles' headway and jerk, i.e., the efficiency and comfort factor of the objectives. In Fig. 3.7, the black line indicates the ego vehicle's desired operating range to maintain safe inter-vehicle distance, provide adequate comfort, and improve road usage efficiency. As mentioned, in the cut-out scenario LV-A changes lane at the last moment, to avoid collision with the slow-moving vehicle in front of it. As the ego vehicle monitors the LVs' lateral movements, it responds to the lane-changing behavior by gradually switching its focus to LV-B.

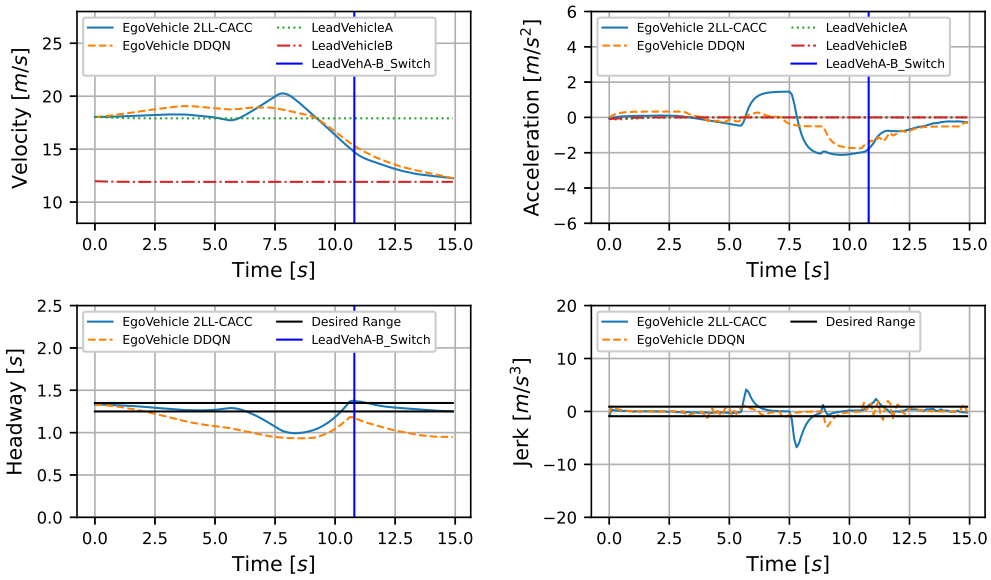

Fig. 3.7 Cut-out scenario. Left: Vehicles' velocity (top) and ego vehicle's headway (bottom). Right: Vehicles' acceleration (top) and ego vehicle's jerk (bottom), under 2LL-CACC and DDQN.

Notice that the ego vehicle's DRL model is designed to maintain a headway of 1.3 s, but the headway increases as the ego vehicle gradually switches its focus to LV-B. Initially, the ego vehicle speeds up to compensate for the rise in headway; however, it also keeps track of the TTC with LV-A to ensure it does not collide with it before it completes the lane-changing maneuver. Once LV-A's lateral position is far enough, LV-B becomes a primary focus for the ego vehicle. Subsequently, the ego vehicle decelerates to maintain zero relative velocity with LV-B, as the latter travels at a lower velocity.

From the bottom left plot of Fig. 3.7, we can see that 2LL-CACC maintains the headway inside the desired range for about 69% of the simulation time. Also, although the ego vehicle cannot keep the headway inside the desired range during the cut-out maneuver, the left plot of Fig. 3.8 shows that the TTC never drops below the critical threshold of 4 s, thus always guaranteeing safety. For better visualization, Fig. 3.8 presents the TTC with an upper bound of 100 s and highlights the lower critical threshold of 4 s with a black horizontal line.

In terms of comfort, one can notice a few spikes in the jerk from the bottom right plot of Fig. 3.7, which however are necessary to maintain in a safe range the

TTC between the vehicles, which have clearly higher priority. Also, the context recognition model consider the safety indicator (χ) as one of the input features to appropriately assign weights to the reward components, so as to prioritize passenger safety. The blue line in the figure denotes the time when the sensor array detects the presence of the new LV. One can infer that the detection is indeed very late, and it could lead to unsafe conditions if the ego vehicle responded to the new LV just from that moment.

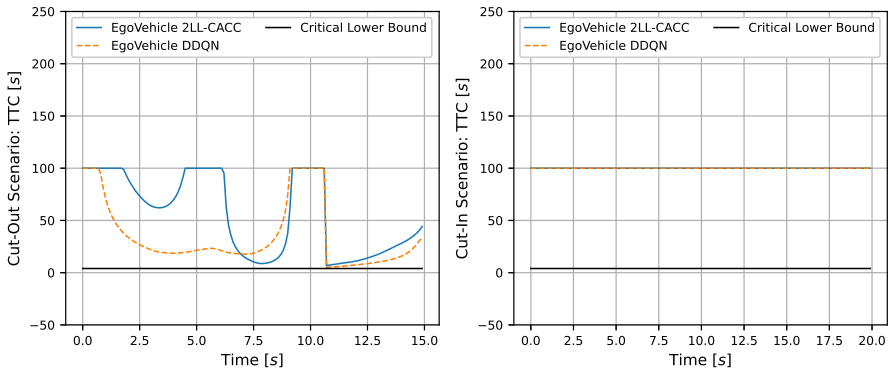

Fig. 3.8 Time-To-Collision of cut-out (left) and cut-in (right) scenarios, under 2LL-CACC and DDQN.

As we can see from Fig. 3.7, the DDQN-assisted ego vehicle does not provide satisfactory results compared to the 2LL-CACC. The bottom left plot of Fig. 3.7 highlights that the DDQN model maintains the headway inside the desired range only for 15% of the total simulation time, while the counterpart maintains it for 69% of the time, providing much better road usage efficiency. Regarding comfort, DDQN could keep the jerk within the desired range for most of the time, but one can notice the oscillatory behavior showing a frequent change in the acceleration and, hence, resulting in sub-optimal performance. In particular, the DDQN suffers from the usage of discrete action space as in this case the ego vehicle has only a limited set of accelerations to choose from. Since the vehicle travels on a dry road and exhibits non aggressive driving behavior, Fig. 3.9 confirms that the ego vehicle's longitudinal slip is always within the desired range, thus ensuring the vehicle's stability.

Further, it is worth mentioning that we tested the DDQN model with an extended set of actions comprising 19 equally spaced actions between -2 m/s^2 and 1.47 m/s^2. However, the DDQN model could not converge even after 480 episodes since the larger action space increases the model complexity and demands more time to learn

the optimal behavior. In contrast, 2LL-CACC learns an optimal policy within 340 episodes and delivers better results than the DDQN.

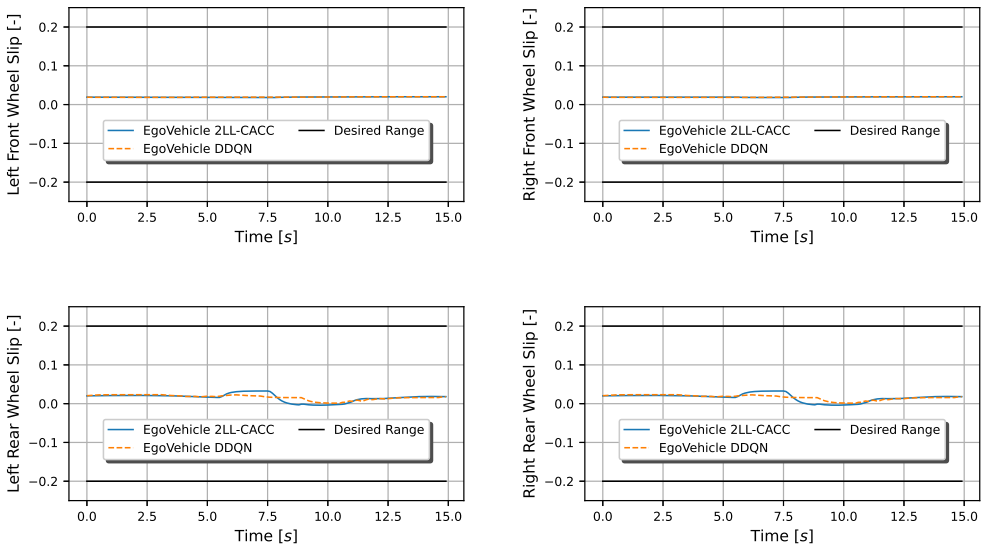

Fig. 3.9 Cut-out scenario: Vehicle wheel slip under 2LL-CACC and DDQN. Left: left front wheel (top) and left rear wheel (bottom). Right: right front wheel (top) and right rear wheel (bottom).

We now move to the cut-in scenario. The top plots of Fig. 3.10 show the vehicles' velocity (left) and acceleration (right) involved in the scenario. The lead vehicle (LV-A) traveling at higher speed overtakes the ego vehicle and then starts a cut-in maneuver to enter the ego vehicle's lane. Also, LV-A decelerates in order to squeeze into the gap between the ego vehicle and LV-B. As before, the ego vehicle promptly recognizes the lane-changing maneuver thanks to V2X communications, and it starts to monitor the lateral movement of the social vehicle. Once the gradual switching determines it is a cut-in situation, the ego vehicle starts decelerating as the distance between them reduces rapidly. Note that in this situation, the distance between the ego vehicle and LV-A does not represent the gap between them. Instead, the relative distance is calculated according to the turning angle and length of LV-A, as it also incorporates the ego vehicle's collision point on LV-A. The calculated relative distance gives additional time to the ego vehicle to handle the maneuver effectively. The camera sensor quickly identifies the LV-A presence and takes control to define the relative distance between the cars. Nevertheless, the role of V2X communication

is still crucial, as it recognizes the maneuver proactively and allows the ego vehicle to decelerate gradually.

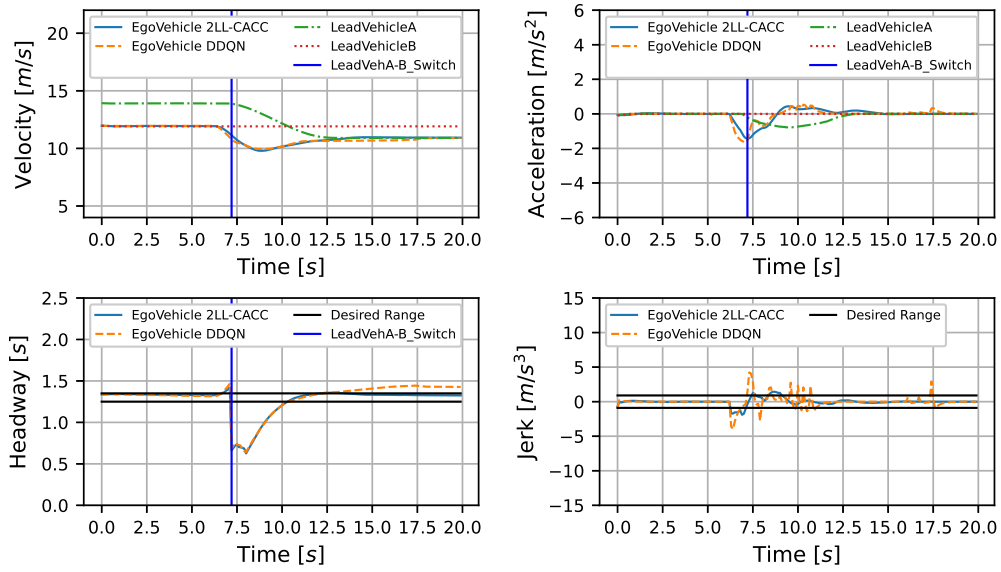

Fig. 3.10 Cut-in scenario. Left: Vehicles' velocity (top) and ego vehicle's headway (bottom). Right: Vehicles' acceleration (top) and ego vehicle's jerk (bottom), under 2LL-CACC and DDQN.

As for the headway, the bottom left plot of Fig. 3.10 shows that the 2LL-CACC can maintain such metric within the desired range for a more extended time period compared to the DDQN (78% versus 45% of the total simulation time). Also, the right plot of Fig. 3.8 shows that the TTC never drops below the critical safety threshold of 4 s: this shows that, even in the closer cut-in situation, the gradual switching can assist the ego vehicle to ensure safety and better road usage efficiency.

The bottom right plot of Fig. 3.10 underlines that the jerk is temporarily outside the desired range during the cut-in maneuver, but this is again inevitable, as the scenario demands such a response to maintain a safe distance between the cars for the whole simulation period. Furthermore, the ego vehicle's longitudinal slip presented in Fig. 3.11 shows that the vehicle remains stable with both DDPG- and DDQN-based models, given that the cut-in scenario is carried out on dry road conditions.

In addition, we have executed the cut-in scenario in the dSPACE Scalexio AutoBox and verified the HIL system's performance. As can be seen in Fig. 3.12, the HIL implementation achieves similar performance to that obtained with the standard

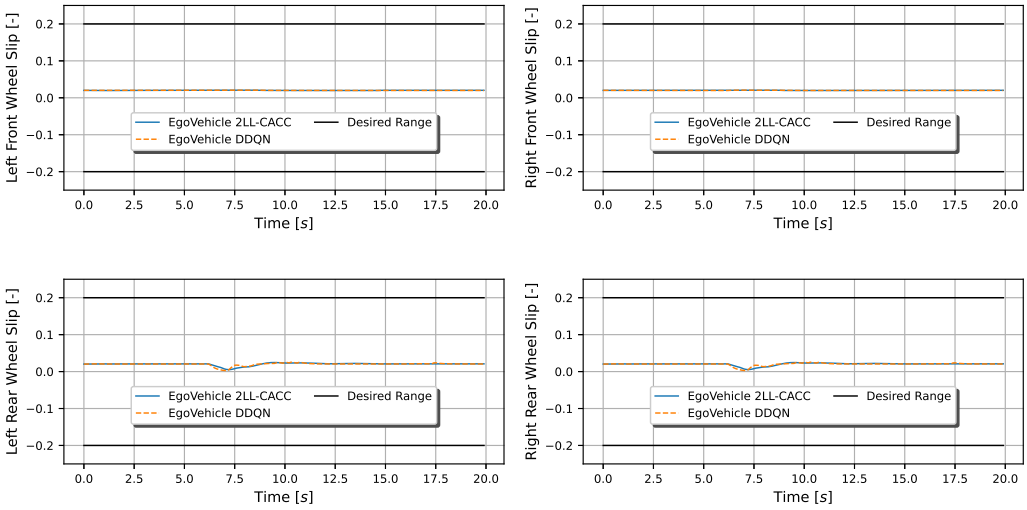

Fig. 3.11 Cut-in scenario: Vehicle wheel slip under 2LL-CACC and DDQN. Left: left front wheel (top) and left rear wheel (bottom). Right: right front wheel (top) and right rear wheel (bottom).

model in the loop setup. This result further strengthens the 2LL-CACC's ability, as it validates the deployable nature of the trained DRL model in actual vehicles.

Table 3.5 Comparison between 2LL-CACC and DDQN in terms of RMSE for headway, jerk, and slip.

RMSE			
Metrics	Scenarios	2LL-CACC	DDQN
Headway (Ideal = 1.3 s)	Cut-out	0.1184	0.2515
	Cut-in	0.1767	0.1872
Jerk (Ideal = 0 m/s^3)	Cut-out	1.2405	0.6976
	Cut-in	0.4715	0.8812
Slip (Ideal = 0)	Cut-out	0.018	0.018
	Cut-in	0.02	0.02

Finally, Tab. 3.5 presents the RMSE of the obtained results to highlight the quantitative performance of the proposed framework. 2LL-CACC achieves very good results with respect to all objectives, and notably outperforms the DDQN-based model in all scenarios. In terms of comfort, 2LL-CACC has higher jerk RMSE values; however this is inevitable, to promptly react to new conditions, as passengers' safety has to be prioritized over comfort in these critical scenarios. Still, 2LL-CACC achieves an excellent trade-off among the three objectives, providing safe inter-

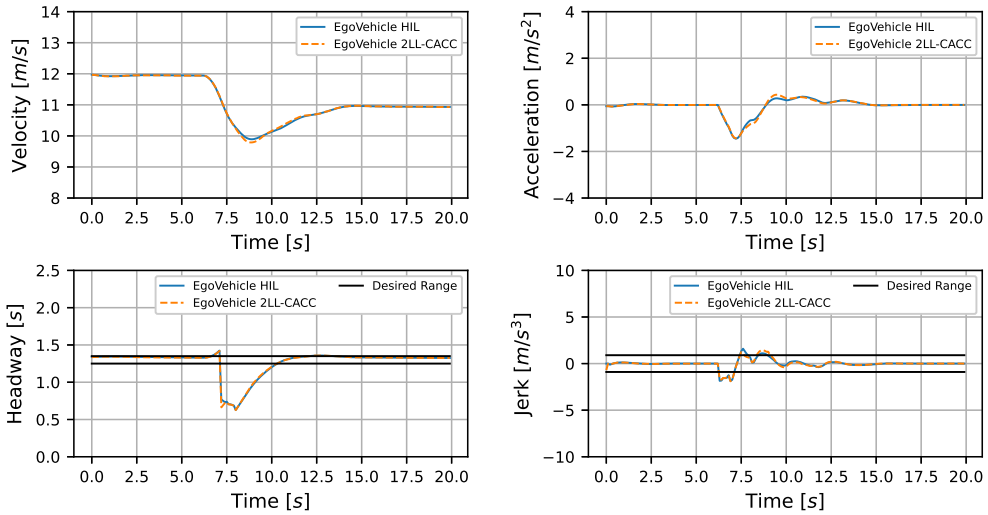

Fig. 3.12 Cut-in scenario. Left: Ego vehicle's velocity (top) and headway (bottom). Right: acceleration (top) and jerk (bottom), under 2LL-CACC and HIL.

vehicle distance, improved road usage efficiency, and satisfactory comfort to the passengers.

3.7 Chapter Conclusion

In this chapter, we addressed Adaptive Cruise Control (ACC) for connected autonomous vehicles in such challenging traffic scenarios as the cut-in and cut-out maneuvers. We proposed a 2-layer, ML-assisted deep reinforcement learning (DRL) approach that weighs the target metrics such as headway, jerk, and longitudinal wheel slip properly and achieves the best trade-off among safety, road efficiency, and comfort objectives. When compared with state-of-the-art alternatives, our framework provides substantially better performance. Notably, it achieves 54% and 33% better headway than its alternatives, thus ensuring better traffic flow efficiency. Particularly, the V2X communication enables the ego vehicle to be timely and gradually switch its focus to the neighboring vehicles, significantly boosting safety performance.

Chapter 4

Adaptive Autopilot: Constrained DRL for Diverse Driving Behaviors

In pursuit of autonomous vehicles, achieving human-like driving behavior is vital. In this chapter, we introduce Adaptive Autopilot (AA), a unique framework utilizing Constrained-Deep Reinforcement Learning (C-DRL). AA aims to safely emulate human driving to reduce the necessity for driver intervention. Focusing on the car-following scenario, the process involves (i) extracting data from the highD natural driving study and categorizing it into three driving styles using a rule-based classifier; (ii) employing Deep Neural Network (DNN) regressors to predict human-like acceleration across styles; and (iii) using C-DRL, specifically the soft actor-critic Lagrangian technique, to learn human-like safe driving policies. Results indicate effectiveness in each step, with the rule-based classifier distinguishing driving styles, the regressor model accurately predicting acceleration, outperforming traditional car-following models, and C-DRL agents learning optimal policies for human-like driving across styles.

Part of the work described in this chapter is submitted to the IEEE Intelligent Transportation Systems Conference (IEEE ITSC 2024).

4.1 Introduction

In recent years, the automotive industry has experienced a digital transformation, enhancing vehicles with sensing devices, electronic control units, and advanced driver assistance algorithms, including features like blind spot detection and ACC [10]. This evolution aims to improve safety, traffic efficiency, and the overall travel experience [96]. However, consumer adoption relies on trust in automated systems and considerations of legal issues [97]. To improve trust, multiple additional features like head-up displays and comprehensive information displays can be used to help the driver understand the functionality of the supportive systems [98, 99].

The acceptance of these systems is also influenced by their ability to emulate human-like driving styles [100, 101]. Toward this, distinct driver categories, like aggressive drivers prioritizing smaller gaps with abrupt maneuvers and conservative drivers favoring larger gaps with smoother behavior [102], require tailored controllers. Due to their inherent differences, a controller calibrated to an aggressive driving style is not suitable for conservative drivers, and vice versa. Current car-following controllers [103–106], despite attempts to differentiate among driving styles, depend on predefined parameters, lacking real-world adaptability [107]. The remedy lies in data-driven controllers utilizing real-world data, emulating diverse driving styles, and, theoretically, having the potential to reduce disengagement rates [100]. In this context, ML plays a pivotal role in developing models capable of making informed decisions by analyzing complex and multi-variate data. Among ML paradigms, RL is well-suited for this intricate task [108, 109], as RL agents learn by interacting with the environment through a trial-and-error mechanism, aiming to maximize cumulative rewards. However, traditional RL agents often overlook safety constraints critical for real-world applications like autonomous driving. C-DRL addresses this limitation as, unlike traditional RL, it incorporates constraints through cost functions, ensuring safe driving by minimizing them during the learning process [110].

Building on C-DRL, our work introduces the Adaptive Autopilot (AA) framework. This framework employs a C-DRL approach to effectively accommodate diverse driving styles by integrating rewards based on a human-like acceleration predictor, alongside constraints to enforce a minimum headway among vehicles. The three main steps of the framework are: (i) categorizing real-world driving data from the highD dataset [111] into aggressive, normal, and conservative styles, (ii) training deep neural network-based regressors to predict human driver-like vehicle

acceleration tailored to each driving behavior, and (iii) implementing the C-DRL framework to take human-like safe actions. The trained agents, corresponding to each driving style and based on the soft actor-critic Lagrangian algorithm [112], are validated using real-world driving data from the highD dataset.

The AA framework primarily focuses on car-following scenarios, and therefore, the mobility traces that reflect these situations are extracted from the highD dataset. Initially, these traces are filtered and categorized into three distinct driving styles using a rule-based approach. The categorized data is then used to train a human-like acceleration predictor, which estimates the acceleration a human driver would take under similar conditions. Finally, the C-DRL framework, which leverages a straightforward car-following simulation environment, manages the vehicle movements in a way that is both safe and aligned with human preferences. Results demonstrate the framework's ability to drive the vehicle in line with corresponding human drivers under different styles, with the headway highlighting the prioritization of safety constraints. To summarize, the main contributions of this work are as follows:

(i) Real-world driving data is classified into aggressive, normal, and conservative styles using a rule-based approach. Separate neural network-based regressors are then trained for each style to predict human-like vehicle accelerations.

(ii) A novel C-DRL framework is introduced, adapting vehicle acceleration to different driving styles by (safely) mimicking human drivers. This is achieved through minimizing the difference between C-DRL and predicted human actions by neural network-based regressors at each step. Further, a headway-based safety constraint is imposed during training, where multiple real-world driving traces are used to enhance generalization.

(iii) Performance results demonstrate the proposed framework's ability to adapt to diverse driving styles while adhering to safety constraints.

In this chapter, Sec. 4.2 reviews the related research works on this topic, Sec. 4.3 presents the preliminaries on constrained RL and introduces the Adaptive Autopilot framework, Sec. 4.4 discusses the obtained results on realistic driving traces, and Sec. 4.5 presents concluding remarks.

4.2 Related Work

In recent years, a significant amount of research has been conducted in the automotive domain, particularly on the topic of autonomous vehicle driving [10, 108], driven by technological advancements as well as improvements in data-based learning algorithms. In this study, we specifically focus on the aspect of longitudinal vehicle control, which ensures a safe following distance between vehicles while adapting to diverse driving styles, mimicking human-like driving behavior to achieve context-aware and personalizable version of the ACC application. While commercial ACC systems were introduced in the early 2000s to enhance safety and driving experiences [113], they still offer limited customization options with few user-defined parameters like desired gap and velocity. The rigidity of these systems hampers their ability to accurately replicate human driving styles, leading to reduced trust and increased instances of driver intervention, thereby affecting safety benefits. Various research directions [114, 115, 109] have been explored to address these limitations and enhance ACC systems.

One research direction involves car-following (CF) models to provide optimal control actions in response to lead vehicle movements. Relevant models include the Gipps model [106], which prioritizes safety, ensuring a safe inter-vehicle distance, and incorporates human factors, considering reaction time and comfort along with the braking capabilities of both vehicles. The Intelligent Driver Model (IDM) [104], a widely known car-following model, considers desired velocity and inter-vehicle distance as key factors to calculate suitable vehicle acceleration. Further, IDM also utilize different parameter values accounting for different driving styles to achieve a realistic representation of driver behavior and analyze their impact on traffic dynamics [103]. However, these CF models struggle to accurately represent real-world driving behavior due to oversimplification, and their parameters are calibrated for traffic scenarios and safety rather than human-like driving behavior [114]. Our framework, compared to IDM, employs a data-driven approach demonstrating safe and human-like acceleration behavior across different styles.

Another research direction explores data-driven models, optimizing vehicle control using real-world mobility traces. For example, Papathanasopoulou and Antoniou [115] employs Particle Swarm Optimization with Bi-directional Long Short-Term Memory (PSO–Bi–LSTM) to enhance IDM model parameters and predict human driving behavior. IDM's learned fixed parameters limit however its ability to ac-

curately model driving behavior. Other works use traditional and recurrent neural networks (NNs) for acceleration/velocity predictions [116, 117]. Such NNs face challenges in personalized driver behavior modeling due to training data influences. Similarly, DRL has been utilized [77, 109] for improved car-following behavior, emphasizing safety, traffic efficiency, and comfort. Nevertheless, these DRL approaches focus solely on generic driving behaviors, lacking consideration for a human in their training process to achieve human-like driving behavior.

Given the ability of the DRL to model dynamic behaviors, Human-in-the-loop reinforcement learning [118] is gaining traction to integrate the human factor into the training loop to further improve the adaptability of the DRL framework to produce context-aware and personalized behavior. The human guidance can be accounted for either by offline [107] or online [119] during the training process. In the offline setup, human experience can be utilized to frame the reward functions during training, while the online setup requires real-time human assistance while training the RL agent. Since, RL operates through a trial-and-error learning process and the agent learns optimal strategies through exploration, it is time-consuming to have real-time human assistance while training. This typically results in longer episode durations compared to offline human-in-the-loop frameworks, employed in this proposed framework. For instance, Zhu et al. [107] uses Shanghai naturalistic driving study data to mimic human-like driving behavior by designing reward functions to reduce errors between simulated and empirical values in spacing and velocity. It outperforms traditional neural network models in capturing driver behavior, although safety concerns arise as aggressive human behaviors are replicated without considering safety. Additionally, Tian et al. [120] employs behavior cloning, an imitation learning method, to achieve human-like driving behavior. A major drawback of imitation learning is the accumulation of errors over time, leading to adverse control actions.

To the best of our knowledge, our work is the first to present a comprehensive human-in-the-loop C-DRL framework designed to adapt vehicle driving behavior across diverse driving styles along with safety constraints.

4.3 Adaptive Autopilot Framework

In this work, the focus is on achieving human-like driving behavior through an Adaptive Autopilot controller designed to accommodate various driving styles while

ensuring safe conditions, especially in car-following scenarios. The proposed AA framework addresses three interconnected problems: (i) identifying and classifying the driver's style into aggressive, normal, or conservative using a rule-based approach (Sec. 4.3.1); (ii) training a neural network-based regressor to predict human-like control actions, particularly acceleration rates, of the same driving style of the driver (Sec. 4.3.2); (iii) implementing a C-DRL framework for the controller, considering vehicle's states as input and ensuring safety while minimizing the difference between the control action and human-like acceleration predicted by the regressor (Sec. 4.3.3).

4.3.1 Rule-based Classifier

In general, drivers tend to develop a habit of driving their vehicle in a certain way over time, reflecting their preferred style of driving. The terminology used to differentiate driving styles is diverse, represented by multiple terms such as aggressive, risky, unsafe, defensive, timid, normal, and so on. Inspired by Kesting et al. [103], we categorize driving styles into aggressive, normal, and conservative. Such classification typically utilizes indicators related to longitudinal movements, including speed, acceleration, headway, relative velocity, as well as steering input and lateral acceleration [102]. Nevertheless, given the focus on car-following scenarios, only longitudinal control-related indicators are employed for style categorization.

Designing a model to accurately classify a driver's entire data trace into a unique driving style is challenging due to potential variations within a driver's behavior. For instance, aggressive drivers may exhibit normal or conservative driving at times. In this work, each control action of the driver is tagged with a specific driving behavior. Subsequently, the ratio of each tagged behavior across the entire trace is calculated to categorize drivers as aggressive, normal, or conservative. Driver actions are tagged with a specific driving behavior based on a rule-based approach, which utilizes headway as a key factor in differentiating driving styles. As suggested by [103], aggressive drivers aim to maintain a headway of 1 second or below, Normal drivers aim for headways of around 1.5 seconds, while conservative drivers aim for a headway of 1.8 seconds and above. Based on longitudinal indicators, the classifier's objective is to tag the driver's intention, analyzing how the driver's action will change the headway and toward which of the three goal headways it will lead in the long term.

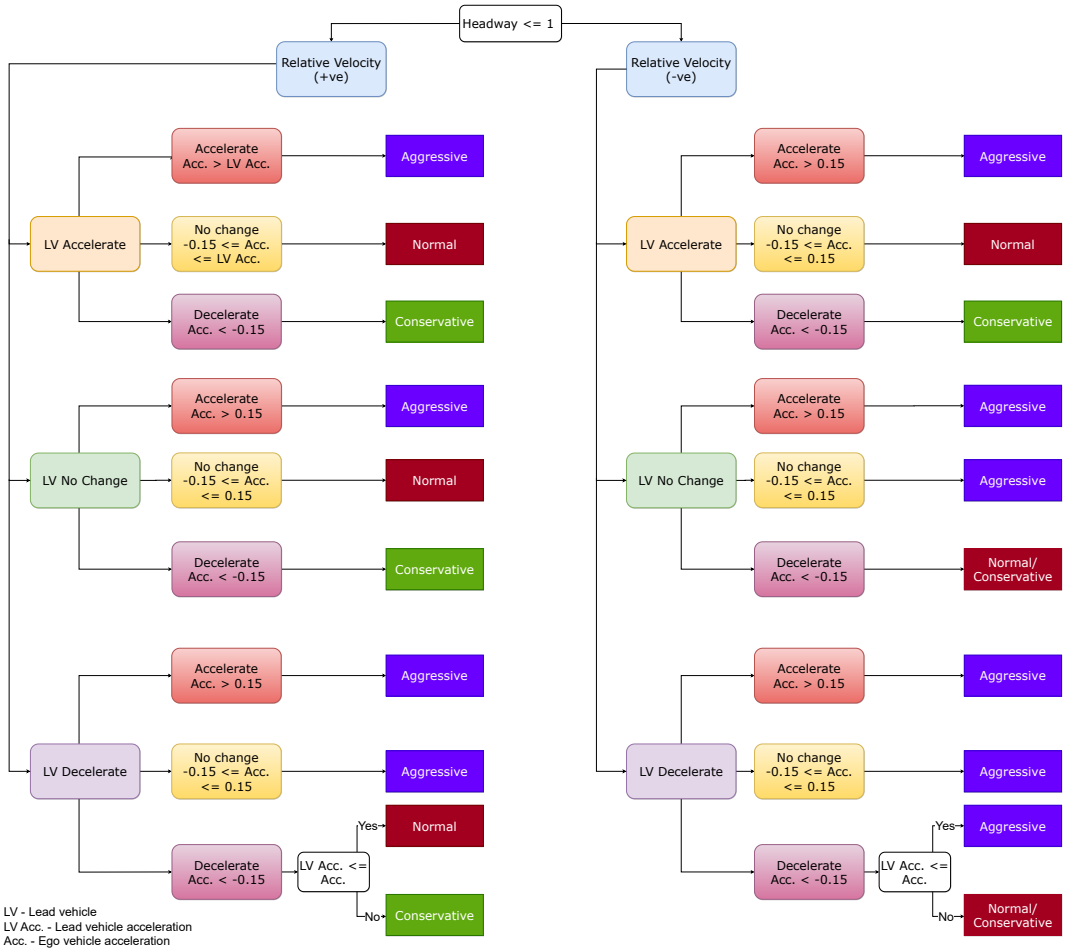

Fig. 4.1 A hierarchical rule-based classifier that aims to label the driving data into Aggressive driving style category.

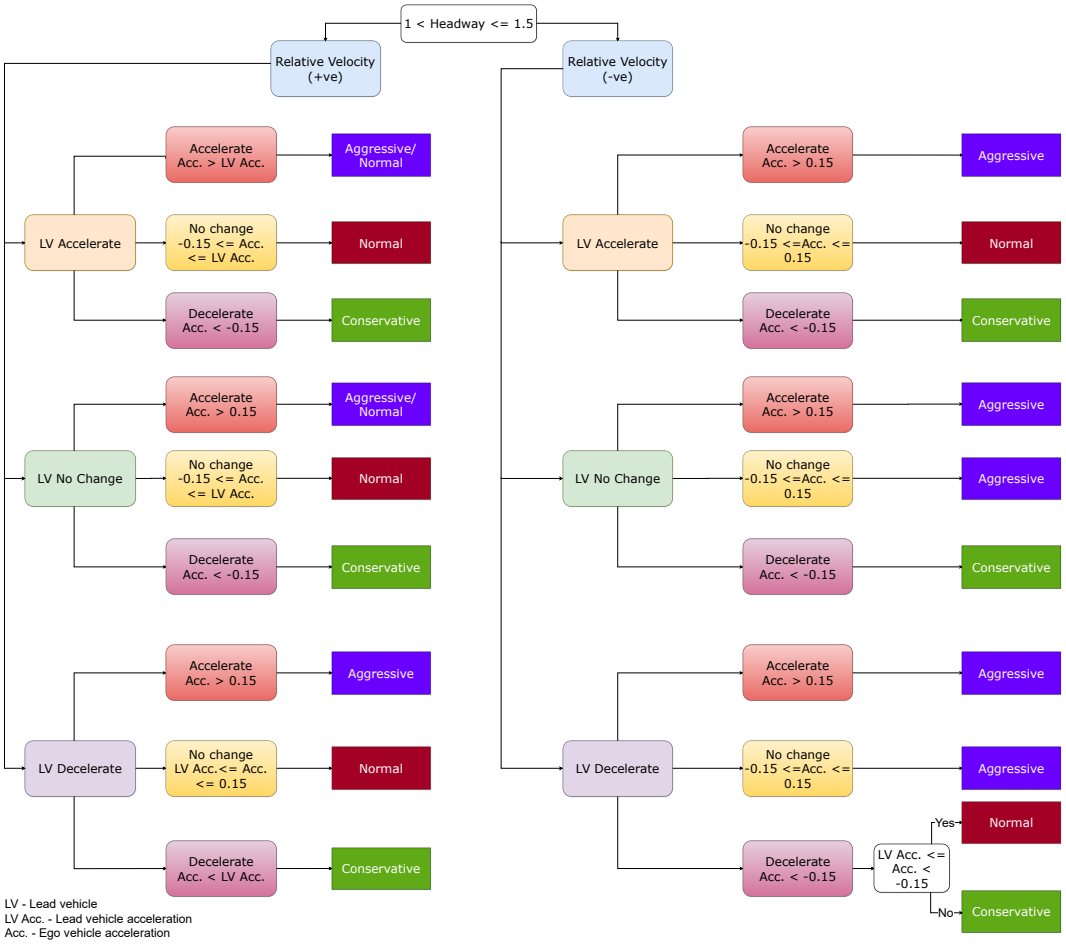

Fig. 4.2 A hierarchical rule-based classifier that aims to label the driving data into Normal driving style category.

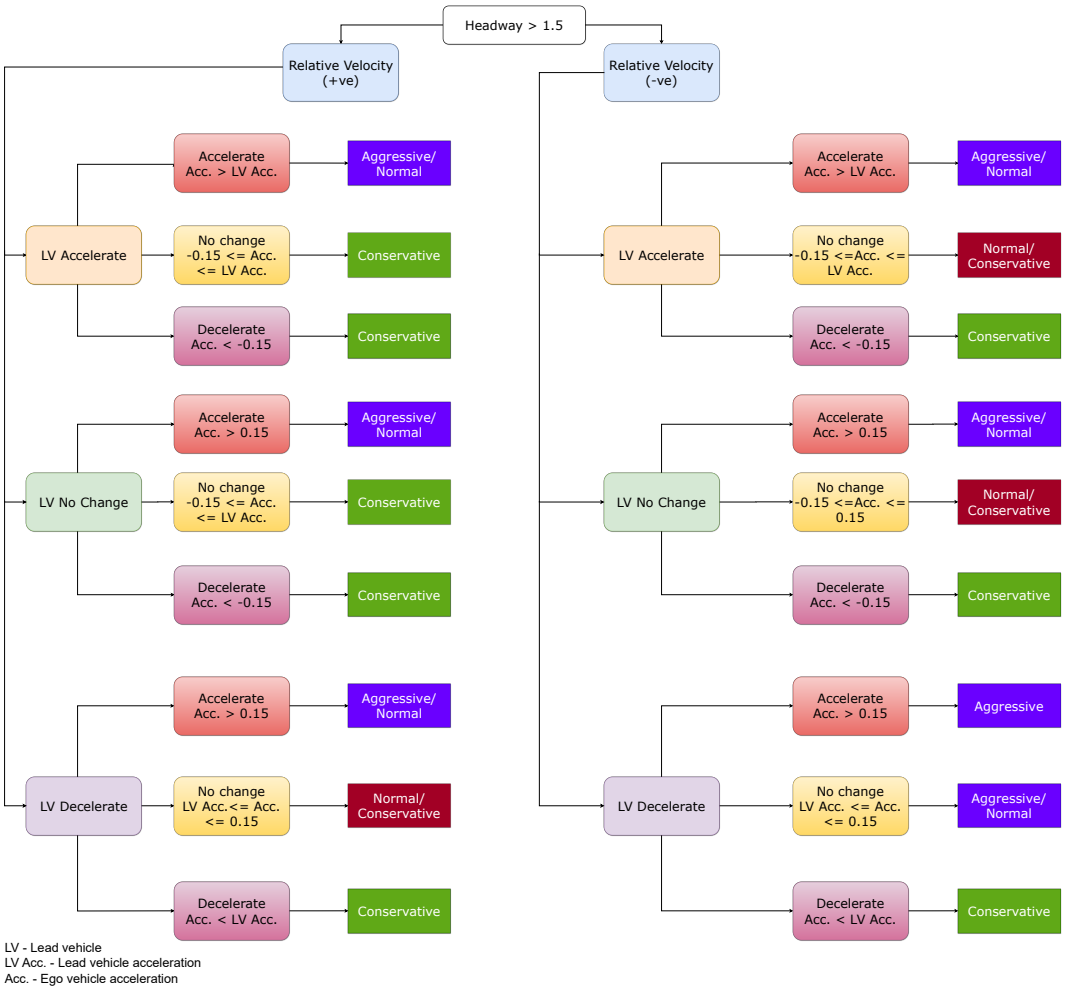

Fig. 4.3 A hierarchical rule-based classifier that aims to label the driving data into Conservative driving style category.

Considering these aspects, Figs. 4.1-4.3 outlines the hierarchical rule-based classification approach employed in this work, where the leaf nodes represent the assigned driving style. Specifically, the classifier considers information related to both lead and ego vehicles at a given time t: $X(t)=\{\vartheta(t), v(t), \ddot{x}_{ego}(t), \ddot{x}_{lead}(t)\}$, representing headway, relative velocity, ego vehicle's acceleration, and lead vehicle's acceleration, respectively, to classify the behavior: $y(t)=\{Aggressive, Normal, Conservative\}$. Headway and relative velocity are formulated as:

$$\vartheta(t)=\frac{\Delta x_{lead}(t)}{\dot{x}_{ego}(t)} \tag{4.1}$$

$$v(t)=\dot{x}_{lead}(t)-\dot{x}_{ego}(t), \tag{4.2}$$

where $\Delta x_{lead}(t)=x_{lead}(t)-x_{ego}(t)$ is the relative distance between lead and ego vehicles, and $\dot{x}_{lead}(t)$ and $\dot{x}_{ego}(t)$ represent the lead and ego vehicle's speed, respectively.

Among the input features, headway serves as the primary criterion for splitting the data into three leaf nodes. Subsequently, for each leaf node, relative velocity becomes a crucial factor in anticipating potential headway changes in subsequent time steps, acting as a secondary criterion for data segmentation. Considering the current headways and relative velocities, the driver's action, specifically the applied acceleration, is categorized into one of the three driving styles. Acceleration differences among the lead and ego vehicles enable an understanding of future changes in relative velocity before they manifest in the data points. This ability facilitates the identification of future achieved headways. Matching future achieved headways with desired headways (as per [103]) provides a straightforward mean to categorize driving actions.

4.3.2 Human-like Action Predictor

The human-like action predictor operates as a regressor model, using relevant input data to forecast the vehicle's acceleration. Its purpose is to learn a non-linear function that approximates the relationship between input data and the next vehicle's acceleration. To circumvent relying on past human actions, which might be unavailable in autopilot scenarios as the one of the AA framework, the regressor incorporates both historical and current data related to the lead vehicle, while utilizing only the

present ego vehicle data as input. The input dataset consists of the following set of observations:

$$\begin{aligned} X(t)=\{&\ddot{x}_{lead}(t-2\Delta t),\ddot{x}_{lead}(t-\Delta t),\ddot{x}_{lead}(t),\\ &\dot{x}_{lead}(t-2\Delta t),\dot{x}_{lead}(t-\Delta t),\dot{x}_{lead}(t),\\ &\dot{x}_{ego}(t),\vartheta(t)\}, \end{aligned} \tag{4.3}$$

to obtain prediction $\hat{y}(t)=\ddot{x}_{pred}(t)$ corresponding to the ego vehicle's acceleration $y(t)=\ddot{x}_{ego}(t)$, where t, $t-\Delta t$, and $t-2\Delta t$ represent the present and two past time instants, respectively.

In this work, a DNN-based regressor, a deep learning technique, is utilized to predict human-like acceleration values. The highD dataset, [111], serves as the training dataset, following the segmentation into the three driving styles mentioned above, achieved through the rule-based classifier outlined in Sec. 4.3.1. Hence, a separate model is obtained for each driving style. Throughout the training process, the model is optimized to minimize the mean absolute error (MAE) loss function:

$$L_{mae}=\frac{1}{N}\sum_{i=1}^{N}|y_i-\hat{y}_i|, \tag{4.4}$$

where N is the number of observations used for MAE loss minimization, $\hat{y}_i$ is the predicted value of the ith observation, and y_i is the actual value of the ith observation.

It is worth mentioning that predicting vehicles' acceleration typically formulated as a sequence-to-sequence approach, employing LSTM or other RNN-related techniques, where past acceleration values are used to predict future acceleration values for the next few time steps. However, as mentioned earlier, the AA framework is an interconnected problem where the regressor's predictions facilitate the later C-DRL framework's control actions to achieve human-like driving behavior. Therefore, considering vehicle's acceleration as an input creates a loop, directly influencing subsequent acceleration predictions. To address this issue, we solely rely on environmental-related features and utilize past history from the lead vehicles' information independent of the control actions, maintaining the integrity of the regressor model.

4.3.3 C-DRL Framework

We now introduce our C-DRL framework, inspired by a previously proposed algorithm [112]. The C-DRL framework utilizes pertinent vehicle data as input to decide the vehicle's longitudinal control action, focusing specifically on vehicle's acceleration. The control action applied guides the vehicle, earning rewards based on its ability to emulate the desired human-like driving behavior. Moreover, the framework integrates safety constraints to guarantee that the applied control actions maintain a safe distance between vehicles. Fig. 4.4 provides an overview of the proposed AA framework.

Background: Constrained RL extends traditional RL by introducing constraints on the actions taken by the agent. C-RL is formalized as a Constrained Markov Decision Process (CMDP) [110], an extension of the standard MDP framework. In this form, C-RL is characterized by the tuple $(\mathscr{S}, \mathscr{A}, \mathscr{P}, \mathscr{R}, \mathscr{C}, b, \gamma)$, representing the state space, action space, transition probabilities, reward, cost function, safety threshold, and discount factor, respectively. The goal of C-RL is to solve the CMDP by learning the optimal policy $\pi : \mathscr{S} \rightarrow \mathscr{A}$ that maximizes the expected cumulative discounted reward while satisfying the constraints. The problem addressed by C-RL is:

$$\begin{aligned} &\max_{\pi:(\mathbf{s}(t),a(t))\sim\rho_\pi} \mathbb{E}\left[\sum_t \gamma^t \mathscr{R}(\mathbf{s}(t),a(t))\right] \\ &\text{subject to:} \quad \mathbb{E}\left[\sum_t \gamma^t \mathscr{C}(\mathbf{s}(t),a(t))\right] \leq b \end{aligned} \tag{4.5}$$

where ρ_π denotes the trajectory distribution following policy π, and $\mathscr{R}(\mathbf{s}(t),a(t))$ and $\mathscr{C}(\mathbf{s}(t),a(t))$ represent the reward and cost functions associated with state $(\mathbf{s}(t))$ and action $(a(t))$ at a specific time step t, respectively. As modeling the state transition probabilities for intricate problems can be challenging, a model-free approach is typically used, with the relationship between action and reward/cost implicitly learned by interacting with the environment.

In constrained optimization problems, such as Eq. (4.5), C-RL can utilize an equivalent formulation with Lagrangian multipliers (λ) for optimization. The Lagrangian's saddle point is determined through iterative gradient ascent steps for the policy function π and gradient descent on the Lagrangian multipliers λ [121, 122, 112]. Notably, the gradient step related to λ emphasizes the loss function associated with the constraint. If the constraint is violated, the gradient update increases the multiplier's value, prioritizing the constraint over the reward function, and vice versa. C-DRL advances upon C-RL by incorporating deep neural network-based function approximators to model the policy function $\pi(\mathbf{s}|\theta)$, where θ denotes the neural network parameters. This augmentation significantly improves the C-RL framework's capability to navigate intricate, high-dimensional real-world environments. In this study, we specifically adopt the Soft Actor-Critic Lagrangian (SAC-Lagrangian) technique [112] as the C-DRL methodology to achieve the desired outcome.

States and Action space: The C-DRL state, $\mathbf{s}(t)\in\mathscr{S}$, represents the vehicle state at any time t and is given by:

$$\mathbf{s}(t)=\{\ddot{x}_{lead}(t), \vartheta(t), \dot{x}_{ego}(t), v(t), \psi(t)\} \tag{4.6}$$

where $\psi(t)$ represents the value obtained from the indicator cost function associated with the safety constraint. The cost function is formulated as:

$$\psi(t) = I(\vartheta(t) > \omega), \tag{4.7}$$

where ω represents the safety threshold for the distance between the vehicles. As mentioned earlier, the action space, $a(t)\in\mathscr{A}$, corresponds to the vehicle's acceleration (a continuous variable bounded within the range $[-4,4]\,\mathrm{ms}^{-2}$).

Reward components: The reward signal is a scalar value provided by the environment after each action, offering insights into the agent's performance concerning the framework objectives. Our reward function comprises two key components: (i) human similarity reward, which assesses the disparity between C-DRL control actions and human-like actions predicted by the regressor model; and (ii) comfort, ensuring smooth acceleration changes between time steps. The functions expressing

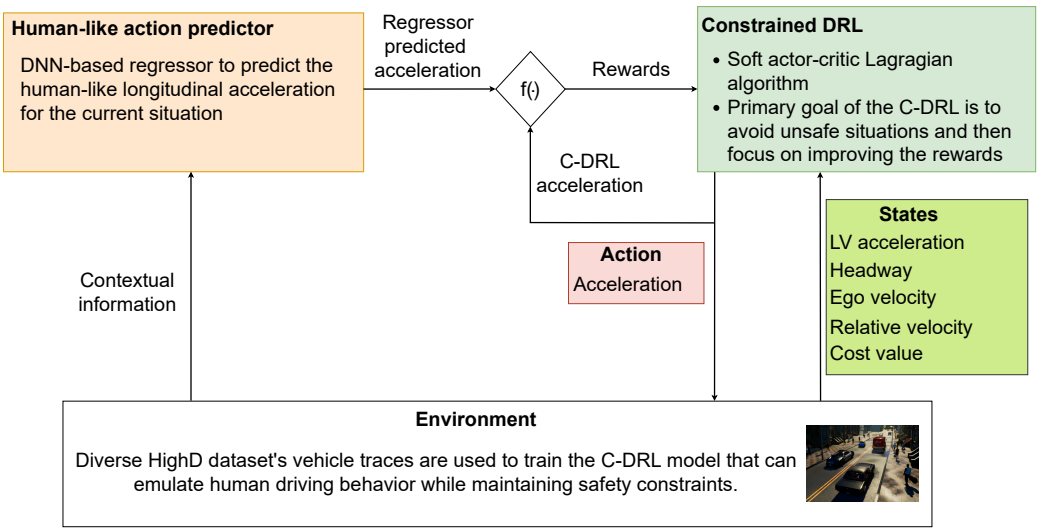

Fig. 4.4 An overview of the proposed Adaptive Autopilot methodology.

these reward components are illustrated in Fig. 4.5. Formally, the reward is expressed as:

$$r(\mathbf{s}(t),a(t)) = r_h(\mathbf{s}(t),a(t)) + r_c(\mathbf{s}(t),a(t)), \tag{4.8}$$

where $r_h(\mathbf{s}(t),a(t))$ and $r_c(\mathbf{s}(t),a(t))$ represent human similarity and comfort rewards at time step t, respectively. The reward components are further detailed below.

Human similarity reward component: The human similarity reward component assesses the similarity between the driving style of the vehicle and that of a human. It quantifies the disparity between the predicted acceleration values by the DNN regressor and the ones applied by C-DRL, encouraging the agent to minimize this difference. Specifically, the function offers a reward that is maximum (+1) for zero error and decreases significantly as the difference between predictions increases. The reward formulation incorporates a tanh function for this purpose:

$$r_h(\mathbf{s}(t),a(t)) = 2 \cdot F_h + 1, \quad \text{with} \tag{4.9}$$

$$F_h = \tanh(-2 \cdot \xi(t)), \tag{4.10}$$

$$\xi(t) = |a(t) - \ddot{x}_{preg}(t)| \tag{4.11}$$

Comfort reward component: Sudden acceleration changes can lead to discomfort for passengers. To address this, the comfort reward component considers the rate of change of acceleration with time, known as jerk ($j(t)$). The reward function is designed to decrease gradually as the absolute jerk value increases, ranging from a maximum reward value of 0 to a minimum of -1. This desired reward is crafted using a curve-fitting function, specifically a 4PL model, as illustrated in Fig. 4.5 (right), depicting the comfort reward component.

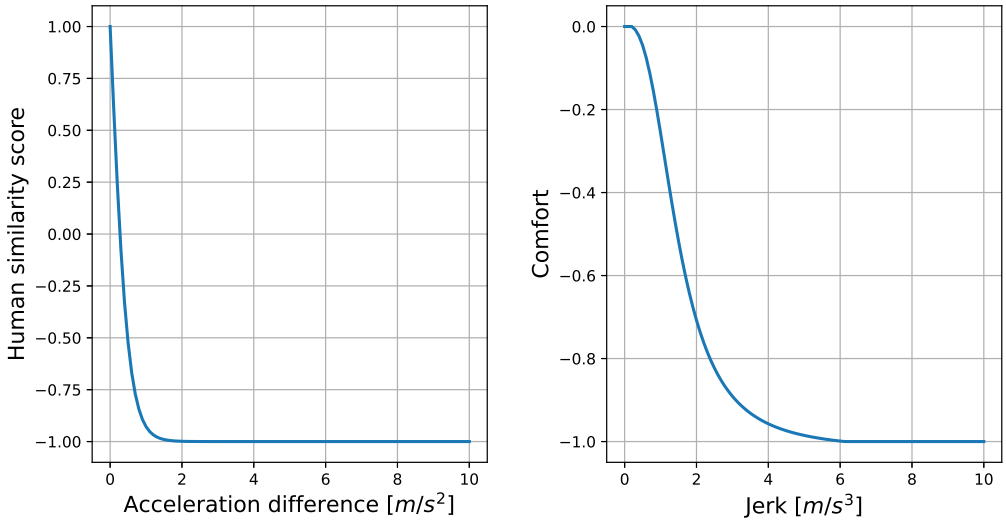

Fig. 4.5 Human similarity (left) and comfort (right) reward functions.

Simulation environment: A straightforward car-following simulation environment is created to replicate vehicle movements, enabling the C-DRL agent to learn the desired behavior. Utilizing the highD dataset, the simulation environment incorporates movements for the lead vehicle, while simulating the ego vehicle's motions using a linear motion model. The C-DRL agent's predicted acceleration serves as the control action to drive the vehicle, with a set sampling interval of Δt=80 ms. The ego vehicle movements follow:

$$\begin{aligned} \dot{x}_{ego}(t+\Delta t) &= \dot{x}_{ego}(t) + a(t)\cdot\Delta t, \\ x_{ego}(t+\Delta t) &= x_{ego}(t) + \dot{x}_{ego}(t)\cdot\Delta t + 0.5\cdot a(t)\cdot\Delta t^2. \end{aligned} \tag{4.12}$$

4.4 Performance Evaluations

In this section, the experimental results of the AA framework are presented. The dataset used for training and evaluation is discussed, followed by a detailed performance overview of the proposed framework.

4.4.1 Dataset

This work utilizes the highD dataset, comprising vehicle trajectories recorded via a drone on German highways at six locations, each covering 420 meters [111]. The dataset consists of 110,500 vehicle trajectories across 60 recordings, with an average recording length of 17 minutes, encompassing free-driving, car-following, and lane-changing events. To focus on the car-following scenario, vehicle traces were filtered based on criteria including duration (minimum 10 s of data), absence of lane changes, consistent lead vehicle, minimum speed (6 ms^{-1}), and vehicle type classification. The sampling frequency used in this work is Δt=80 ms. Among the 60 recordings, 32 were selected for pre-processing, resulting in approximately 2.6 million rows of data, balancing accuracy in training regressor models with computational efficiency.

4.4.2 Performance Results - Rule-based Classifier

Here, we showcase the performance of the rule-based classifier on the highD dataset. Based on the rules presented in Sec. 4.3.1, the data points are classified into three categories: Aggressive, Normal, and Conservative. Fig. 4.6 depicts the key characteristics, in terms of longitudinal acceleration and time headway, of each category using this rule-based setup. Overall, aggressive, normal, and conservative driving behaviors comprise 924k, 1.4M, and 863k data points, respectively, with some data double-tagged because certain behaviors coincide with more than one driving styles.

The performance results of the classifier are consistent with expectations, showing large differences between driving styles. Looking at the probability density function (PDF) (Fig. 4.6 (top)) of the applied longitudinal acceleration, conservative drivers tend to brake to increase the distance from the lead vehicle. Specifically, the PDF mode is at -0.2 ms^{-2} and 85% of the conservative actions represent a braking action. On the contrary, aggressive drivers aim to close the gap with the lead vehicle

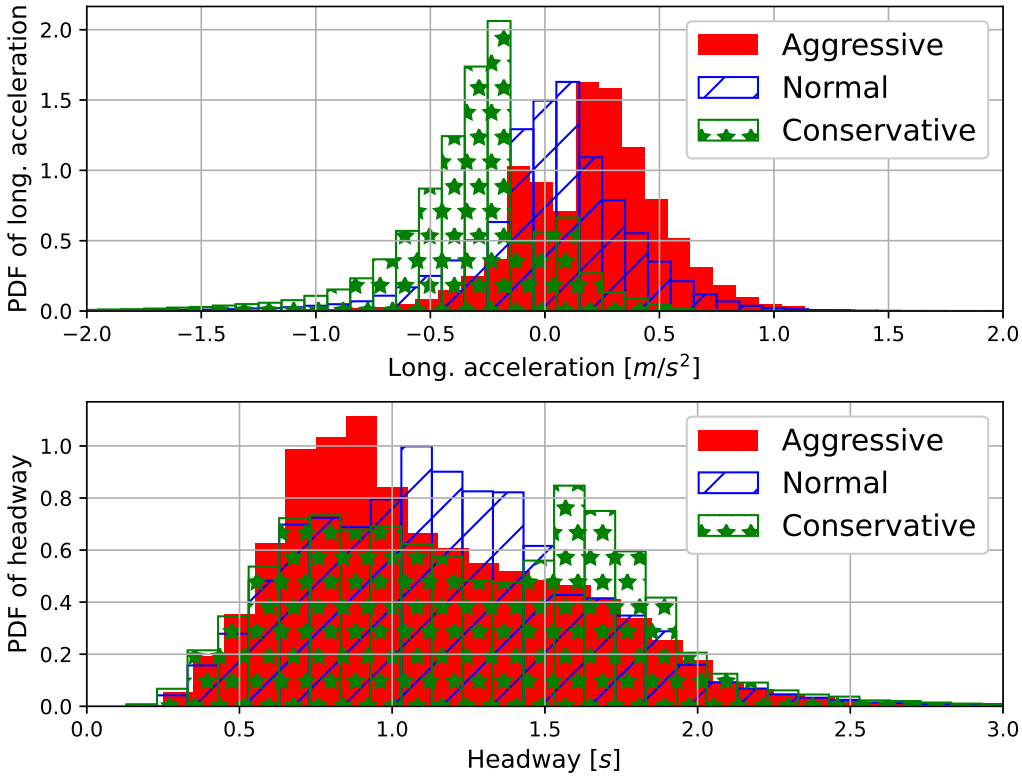

Fig. 4.6 Analysis of driving styles classification using the highD dataset.

as much as possible (the PDF mode is at 0.2 ms^{-2}, and 73% of the aggressive actions represent acceleration). Normal driving follows a hybrid pattern, with the PDF mode at around 0 ms^{-2}.

Further, although classification happens based on the driver's intention to change its headway, the headway PDF plots (Fig. 4.6 (bottom)) show that the mode behavior of a specific driving style corresponds to the ones envisioned in [103] (i.e., the PDF mode of aggressive drivers just below the 1-s mark, of normal drivers between 1 and 1.5 s, and of conservative drivers just before the 2-s mark).

4.4.3 Performance Results - Acceleration prediction

This section discusses the performance of the regressor models corresponding to the three driving behaviors. As mentioned earlier, we employed a traditional DNN

regressor to train the models for predicting vehicles' acceleration based on the input data. Using the categorized data obtained from the rule-based classifier, each driver behavior dataset was divided into training (65%), validation (15%), and testing (20%) sets. Given the varied nature of the input, we employed a standardization technique to scale the input features, with zero mean and standard deviation equal to one to help the training. Also, to mitigate overfitting, an early stopping technique was employed to stop training if the error improvement was less than 0.001 in the validation set for five epochs. It should be noted that this section presents the best configurations for each model after extensive hyperparameter trials (Tab. 4.1).

MAE was used to compare the results obtained during inference. In Tab. 4.2, the results obtained by the proposed DNN regressor model are compared with the well-known car-following algorithm, IDM, with parameters suggested in [103]. Additionally, the IDM model was enhanced to match as closely as possible the highD dataset. The fixed parameters used in [103] were modified so as to obtain the best possible fitting with the dataset data points, i.e., the IDM fixed parameters were selected so that the MAE was minimized. The results obtained demonstrate that the proposed regressor models outperform the car-following algorithms for all driving styles. Further, analyzing the CDF of the MAE (Fig. 4.7), the optimal performance of the DNN predictor is also showcased by the fact that the absolute error of the prediction, i.e., $|\ddot{x}_{ego}(t) - \ddot{x}_{pred}(t)|$, is less than 0.21 ms^{-2} in 80% of the data points for all driving styles.

To assess the predictor's performance on individual driver traces, Fig. 4.8 displays three traces representing aggressive (top), normal (middle), and conservative (bottom) driving behaviors. To evaluate their long-term performance, for both the DNN predictor and the benchmarks, the vehicles in the traces are moved by applying

Table 4.1 DNN regressor's hyperparameter values.

	Driving style		
	Aggressive	Normal	Conservative
Number of hidden units (3 layers)	256, 128, 64	256, 256, 128	256, 128, 64
Dropout rate between hidden layers		0.2, 0.15, 0.1	
Learning rate		0.0001	
Batch size	32	64	64

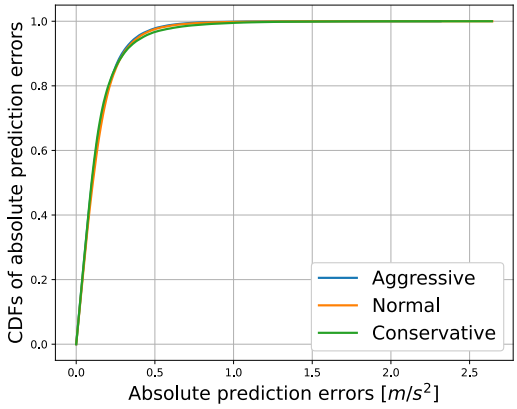

Fig. 4.7 CDF of the mean absolute error for predicted and actual acceleration.

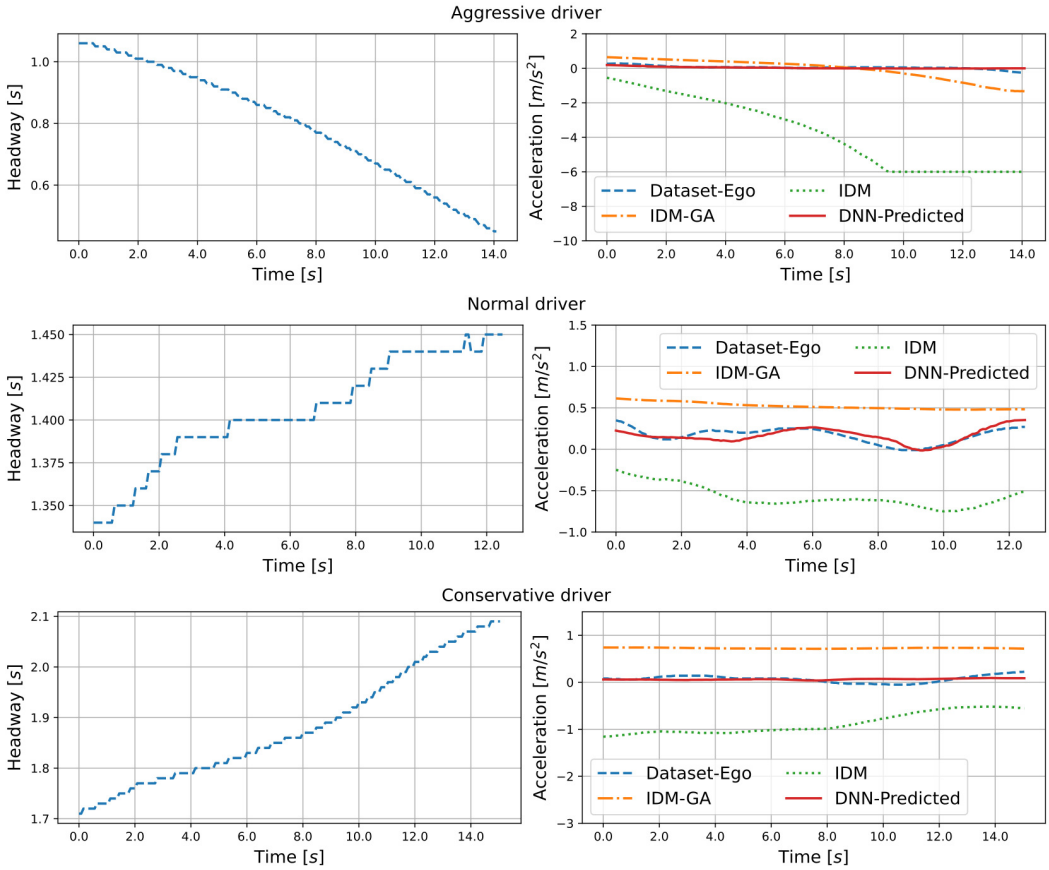

Fig. 4.8 Driver-wise regressor model predictions for the three driving styles.

Table 4.2 Mean absolute prediction errors $[m/s^2]$.

Driving style	DNN	IDM	IDM-GA
Aggressive	0.1356	2.0357	0.3936
Normal	0.1413	2.4309	0.4584
Conservative	0.1415	4.3752	0.6151

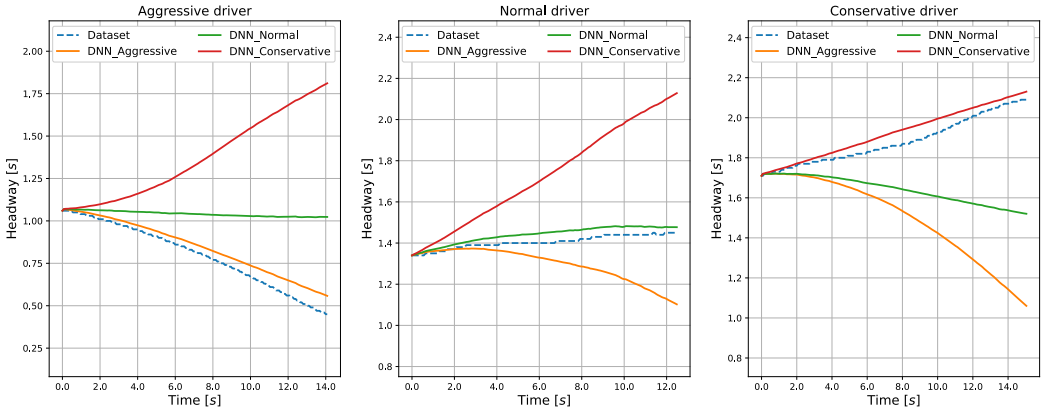

Fig. 4.9 Differentiating regressor model behavior across the three driving styles.

the predicted accelerations using the motion model in Eq. 4.12. That is, while the lead vehicle traces correspond to those in the highD dataset, the ego vehicle trace disregards the driver's applied accelerations but incorporates the acceleration predicted by the DNN and benchmarks. Despite some deviations, the DNN-based predicted acceleration closely matches the actual driver behavior and outperforms existing benchmarks. The slight discrepancies observed can be attributed to the DNN predictor leveraging data from thousands of drivers to learn how to predict acceleration in specific situations, while individual driver styles may vary slightly even within the same driving behavior.

Furthermore, to ensure the models can differentiate among the three driving styles under identical conditions, three regressor model predictors were employed to control the vehicle. Fig. 4.9 presents the resulting headways for the different driving styles using regressor models. For instance, the aggressive model (the left plot in Fig. 4.9) follows the same headway trend as the actual, whereas the normal and conservative models predict acceleration values leading to a wider gap between the vehicles, reflecting their corresponding behaviors. Similarly, the remaining, i.e., normal and conservative, regressor models (illustrated in the middle (normal)

and right (conservative) plots of Fig. 4.9) demonstrate that the models effectively distinguish between driving styles.

4.4.4 Performance Results - Human-like driving

This section discusses the results of the C-DRL models, aimed at mimicking human behavior safely. One model was trained for each driving style, with hyperparameter values similar to those in SAC-Lagrangian [112] (except for the differences listed in Tab. 4.3). Diverse traces were used for each driving style to ensure learning generalized behavior. Figs. 4.10-4.12 represent the driving traces utilized to train the agents. For each agent, a driving trace was randomly selected from the ones chosen for training for each episode.

Table 4.3 C-DRL hyperparameter values.

	SAC-Lagragian
Number of hidden layers (actor, critic)	3, 2
Number of hidden units (actor)	128, 256, 128
Number of hidden units per layer (critic)	128
Learning rate	0.0003
Replay buffer size	1,000,000
Mini-batch size	128
Discount factor	0.99
Number of random exploration episodes	100
Number of transitions between updates	5
Constraint threshold	0.1

The safety objective is to maintain a minimum headway of ω=1 s between two vehicles. Hence, the final accelerations decided by the C-DRL agent must ensure that the two vehicles are never closer than this minimum headway. To achieve this objective, during training, the agent primarily focuses on finding a policy that minimizes the cost function representing the safety constraint. After ensuring the safety constraint is met, the agent tries to maximize the reward function, aiming to find a comfortable acceleration profile that mimics human-like driving behavior. Fig. 4.13 presents the evolution during training of the rewards and the weights assigned to the cost (λ) and reward (1-λ) functions for the three agents. Specifically, the first row shows the evaluation episode rewards during training (executed every 100 training episodes), which is used to assess the training progress. As depicted,

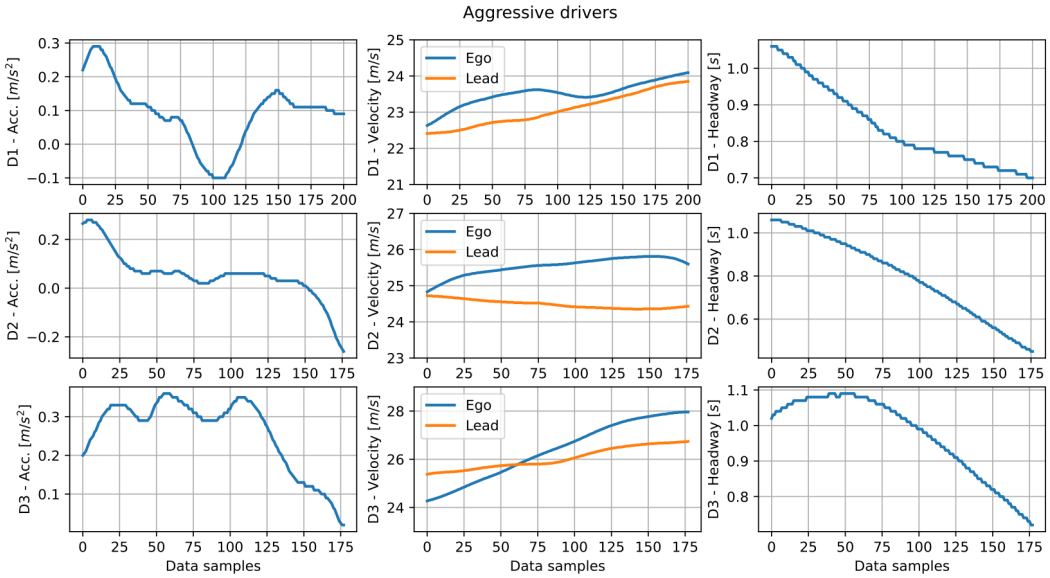

Fig. 4.10 C-DRL training drivers of Aggressive driving style.

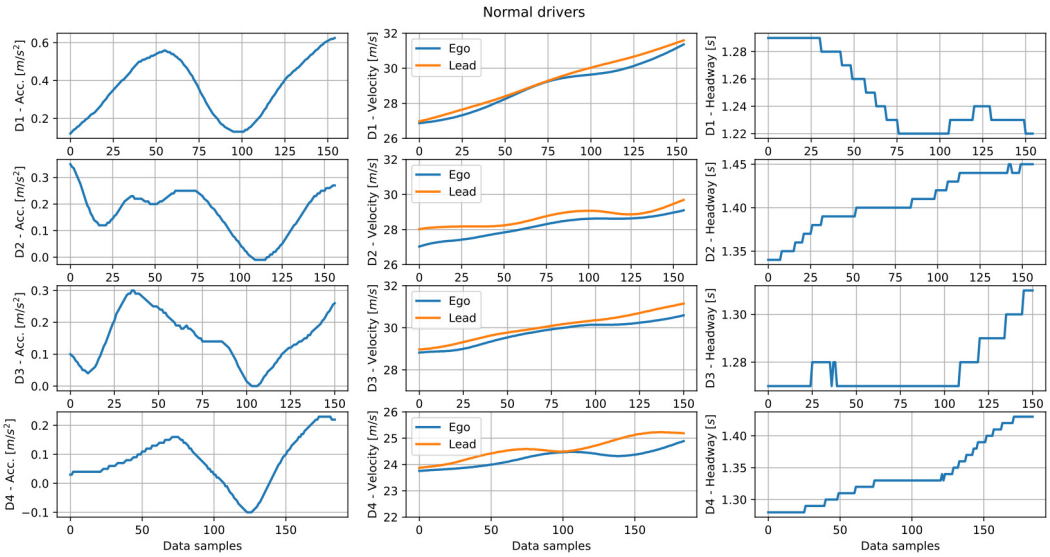

Fig. 4.11 C-DRL training drivers of Normal driving style.

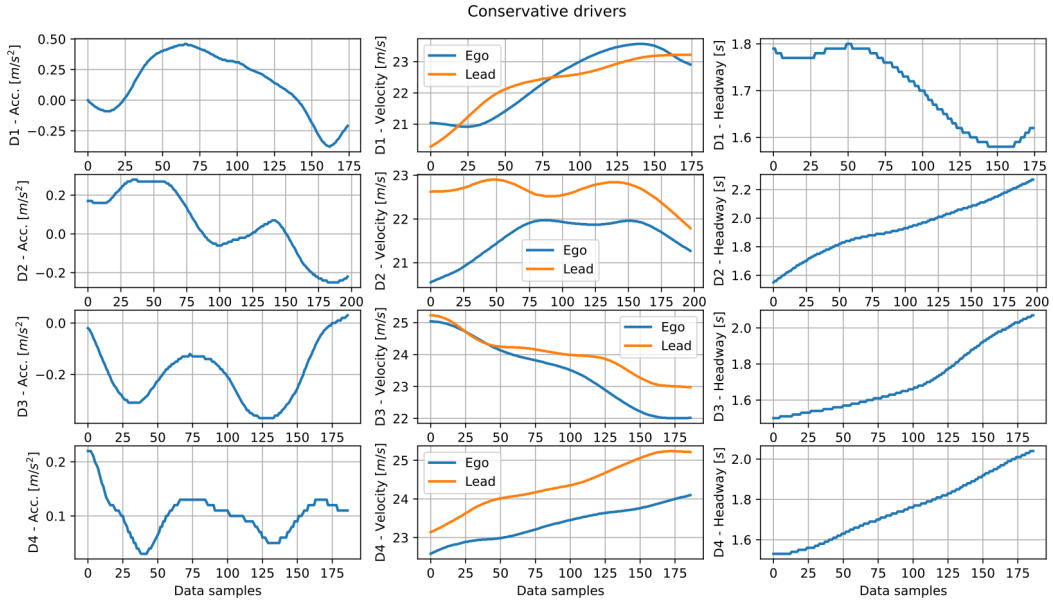

Fig. 4.12 C-DRL training drivers of Conservative driving style.

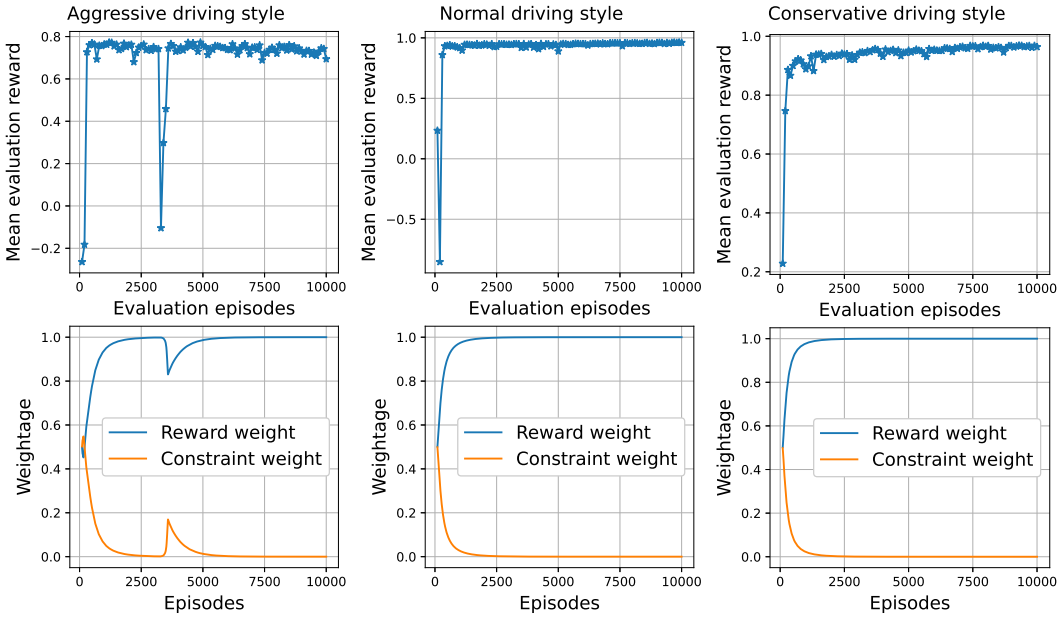

Fig. 4.13 C-DRL training: reward trend (top) and reward vs constraint importance (bottom) for Aggressive, Normal, and Conservative driving.

the reward grows and stabilizes as training progresses. The normal and conservative agents could achieve higher rewards than the aggressive agent because the aggressive agent had to prioritize the safety constraint cost function before maximizing rewards (second row of Fig. 4.13). The conservative and normal driving behavior agents give more importance to the rewards, as the corresponding agents would not breach the safety constraint (according to their driving style). However, for aggressive drivers, who would naturally maintain a headway below the 1-second mark, the optimal cost function weight λ varies until discovering the optimal policy for controlling the vehicle without allowing the headway to drop below the safety threshold. To test the performance of each agent, we selected the best-performing model from the evaluation episodes for the inference phase.

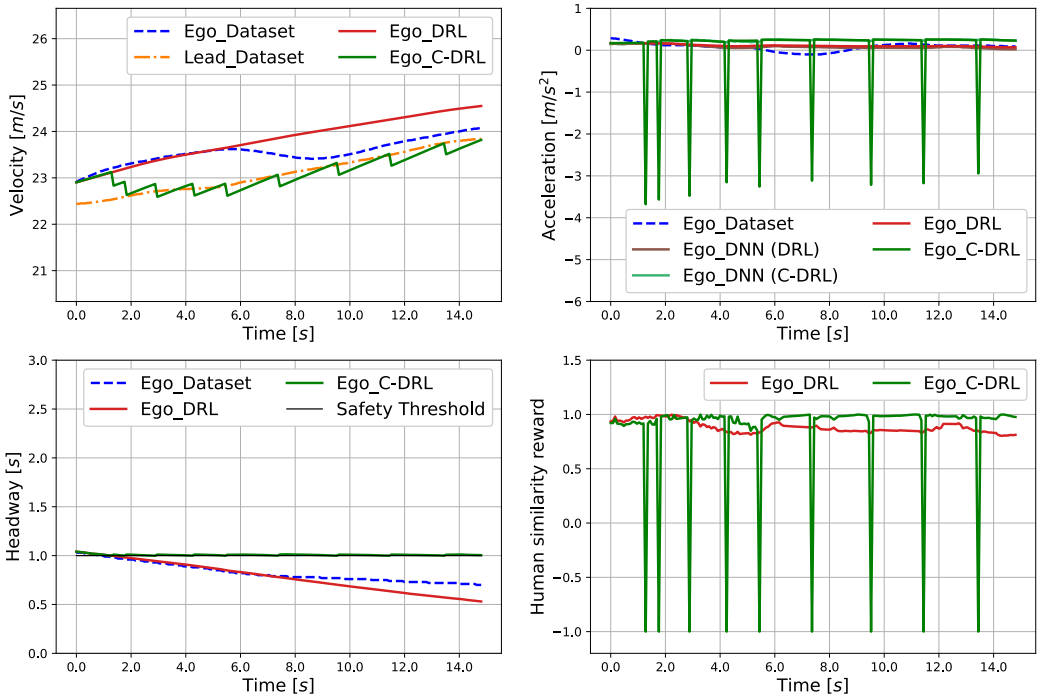

Fig. 4.14 Inference phase: C-DRL - Aggressive driving style.

During inference, the agents are tested on driving traces that were not used during training. Figs. 4.14–4.17 illustrate the agent's performance across the three driving styles, each for a specific trace (with similar results obtained for all tested traces). Fig. 4.14 demonstrates that the aggressive agent could safely drive the vehicle, maintaining the headway around the 1-s mark without violating the safety threshold and also imitating the acceleration predicted by the regressor model whenever possible.

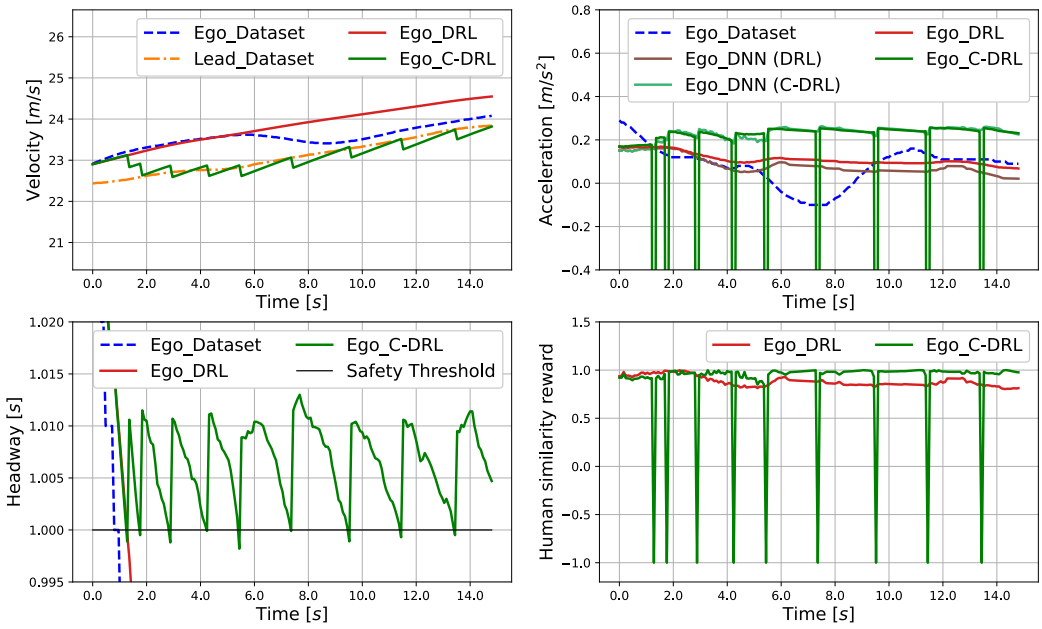

Fig. 4.15 Inference phase: C-DRL - Aggressive driving style (Zoomed).

However, when the headway drops below the safety threshold, even by a very slight margin (as depicted in Fig. 4.15), the agent takes a control action with maximum deceleration. Since the simulation environment utilizes a linear motion model, the control action is immediately reflected in the velocity, consequently increasing the headway above the safety threshold. The rationale behind the aggressive control action stems from the fact that the agent learned to take such actions whenever the headway falls below the safety threshold, even marginally, as inferred from the state space ($\mathscr{S}$). The agent recognizes that taking such decisive control action leads to immediate recovery. Hence, it sacrifices immediate reward, in terms of both human similarity and comfort rewards, as a trade-off to maximize future rewards and prevent constraint violations. Once the agent has successfully satisfied the safety constraint, its focus shifts to mimicking the driver's behavior, as depicted by the vehicle's acceleration (as in Fig. 4.14 (top right)), closely following the DNN regressor model predictions. This is also confirmed by the human similarity reward (see Fig. 4.14 (bottom left)). To emphasize the importance of the C-DRL approach, we compared the proposed framework with a non-constrained DRL technique (referred to as Ego_DRL), where we excluded the cost indicator function during training, confirming that without the safety constraint, the aggressive driving style could

lead to unsafe headway, potentially resulting in dangerous situations. In the next subsection (Sec. 4.4.4), we regulate the control actions predicted by the agent by constraining its subsequent actions within pre-defined bounds.

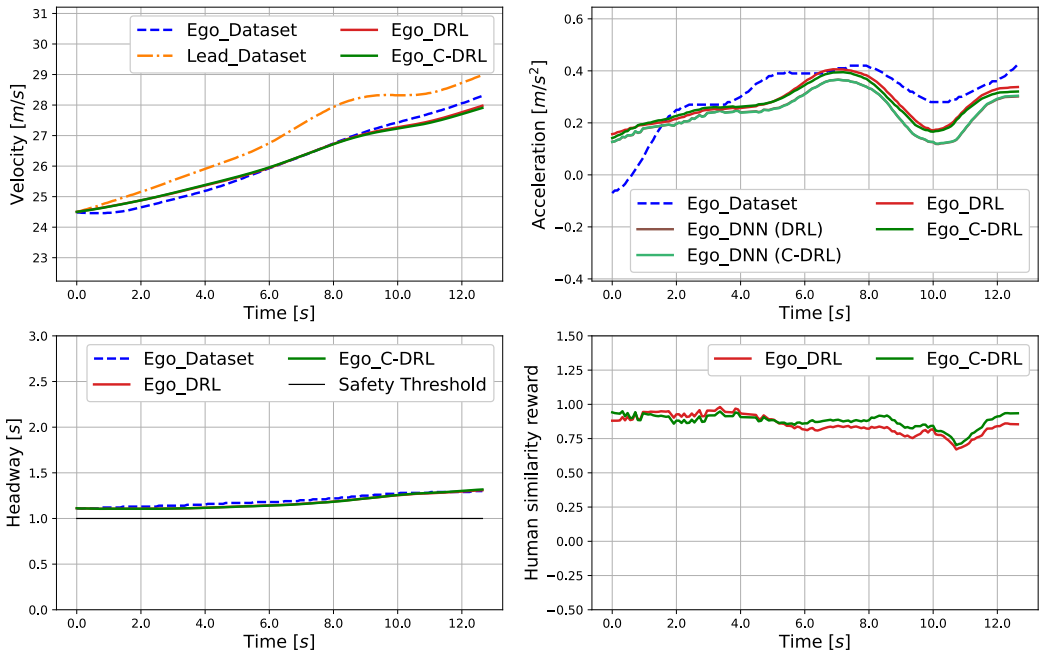

Fig. 4.16 Inference phase: C-DRL - Normal driving style.

In the normal and conservative driving styles, where the safety constraint's role is not crucial, the agents effectively mimicked human driving behavior by following the DNN regressor-predicted accelerations (Figs. 4.16–4.17). To quantitatively evaluate the results, we calculated the root mean square error between the regressor-predicted human-like acceleration and the C-DRL predicted acceleration, resulting in error values of 0.7778, 0.032, and 0.004 for aggressive, normal, and conservative driving behavior, respectively. As anticipated, aggressive driving behavior yields a higher magnitude of error due to safety constraints, while errors for normal and conservative driving behaviors remain minimal. Additionally, it is noteworthy that, although slight discrepancies exist between the DNN regressor and the actual human-applied acceleration, the overall headway profiles generated by the C-DRL agents consistently align closely with those observed in the dataset.

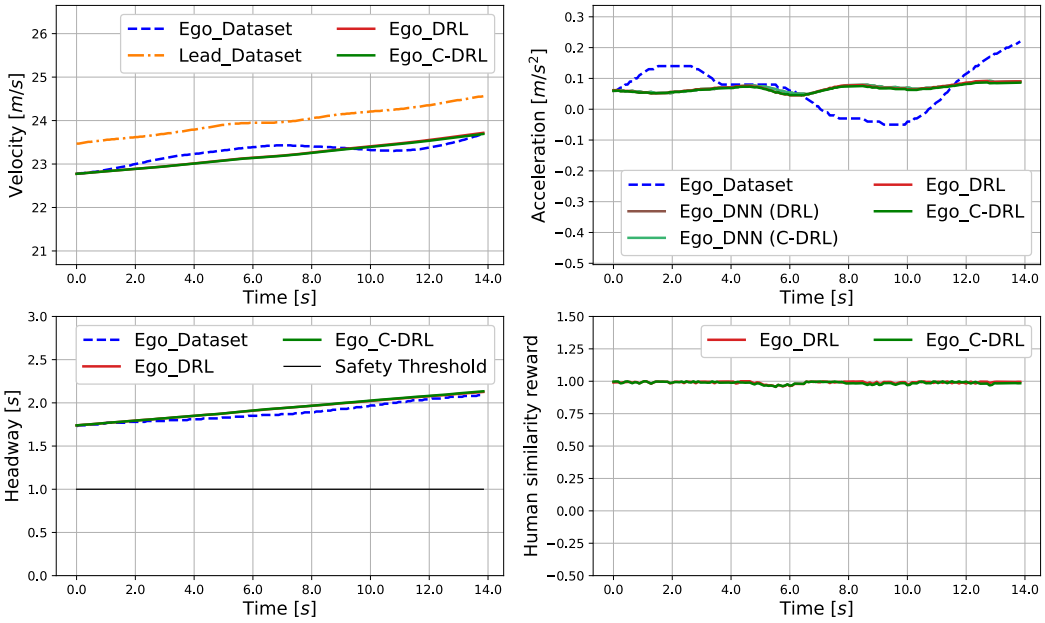

Fig. 4.17 Inference phase: C-DRL - Conservative driving style.

Performance Results - Human-like driving with action bounds

Even though the comfort reward term aims to mitigate abrupt acceleration changes in control actions, whenever the headway falls below 1 second, even marginally (as depicted in Fig. 4.15), the agent takes sharp control action, which immediately affects the velocity, consequently influencing the distance. To prevent these abrupt changes, consecutive acceleration values are restricted to vary by no more than $\pm 0.24\,\mathrm{ms}^{-2}$ [123]. However, in general, RL relies on the exploration-exploitation process, which is crucial to finding a balance between trying new actions and exploiting actions with high rewards. Initial training stages necessitate thorough exploration of the action space to discover those actions that maximize cumulative rewards. However, if action values are restricted, hindering exploration, the agent may miss identifying actions leading to higher rewards. To mitigate this, we adopt a curriculum learning approach [124, 125], gradually increasing difficulty during training. Initial episodes allow unrestricted changes in subsequent actions, with limits introduced once the agent learns the desired behavior. Further, the action taken in the previous time step ($a(t-\Delta t)$) is added to the state space ($\mathscr{S}$) to aid in the learning process. The training process with bounded actions is repeated for the same drivers, and Fig. 4.18 presents the reward and associated weights during training. It is worth noting that

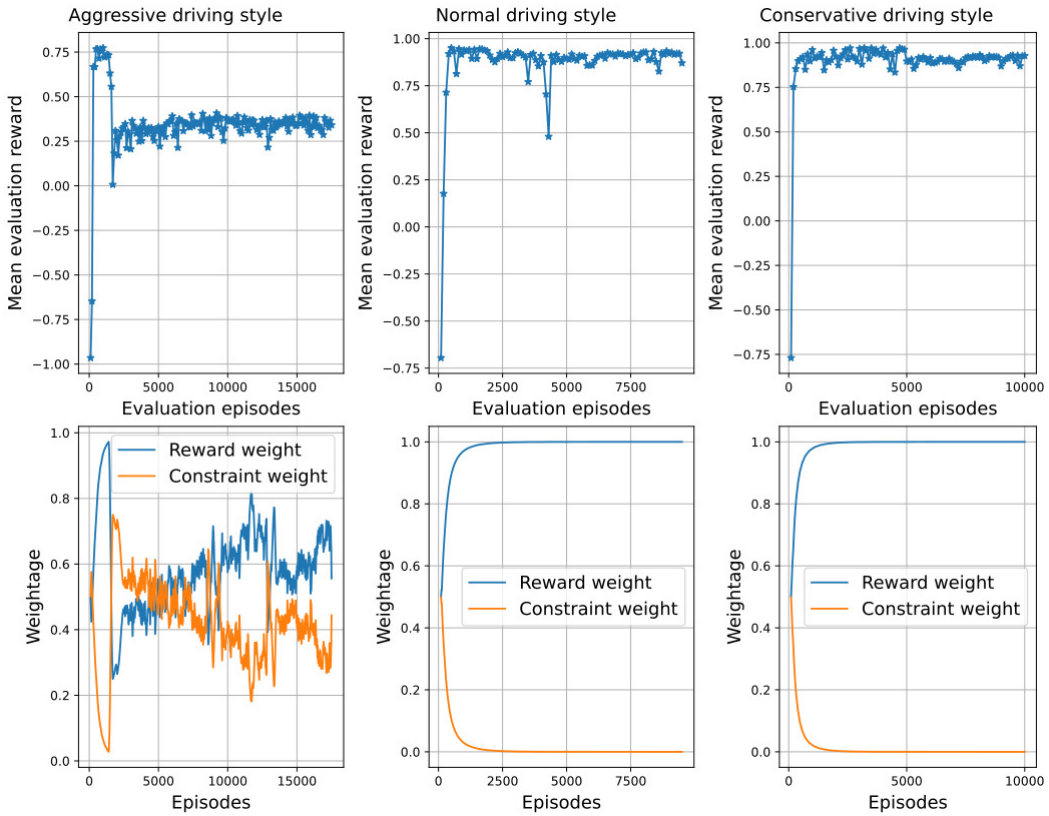

Fig. 4.18 C-DRL training with action bounds: reward trend (top) and reward vs constraint importance (bottom) for Aggressive, Normal, and Conservative driving.

limiting subsequent accelerations is mainly necessary when using a simple linear motion model to drive the vehicle, as drastic changes in acceleration are immediately reflected in the vehicle's movements. As expected, the normal and conservative agents could achieve similar rewards as in the scenario without action bounds, while the aggressive agent encounters difficulties due to the action limits, as it would take time to recover if there was a safety constraint violation. Fig. 4.18 shows that the cost function weight λ is not equal to zero, which indicates that maximum reward maximization would fail to respect the safety constraint, which is undesirable. Hence, the agent trades reward maximization for enhanced safety. Eventually, during training, the aggressive agent learns to prioritize rewards over constraints, as demonstrated by the weight distribution between the reward and constraint (as shown in Fig. 4.18), indicating that the agent has learned to balance satisfying both rewards and constraints during its training process.

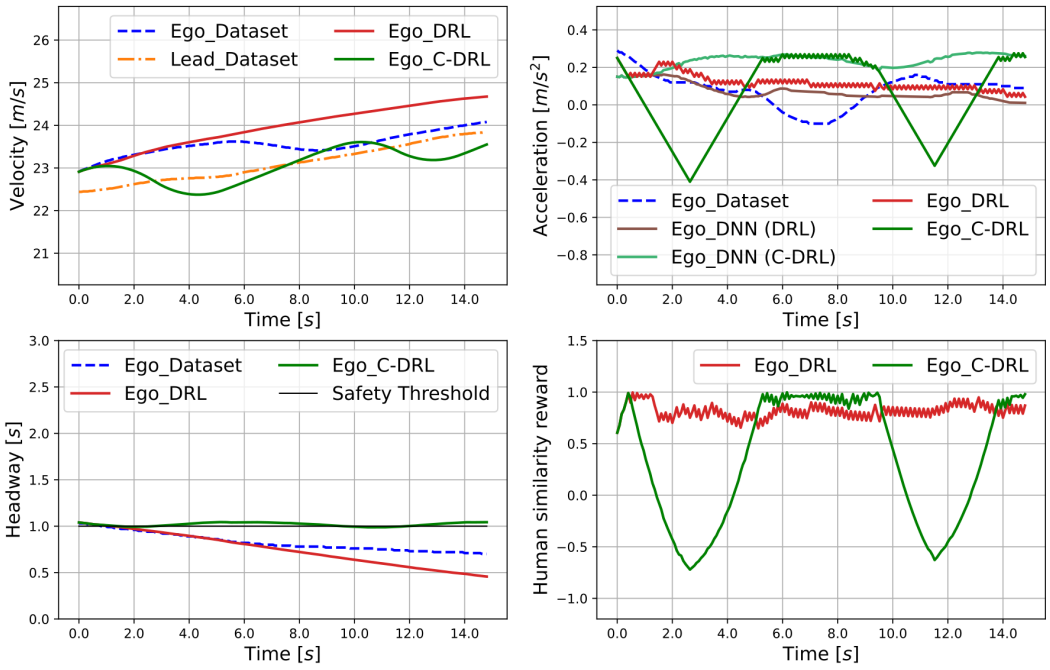

Fig. 4.19 Inference phase: C-DRL with action bounds - Aggressive driving style.

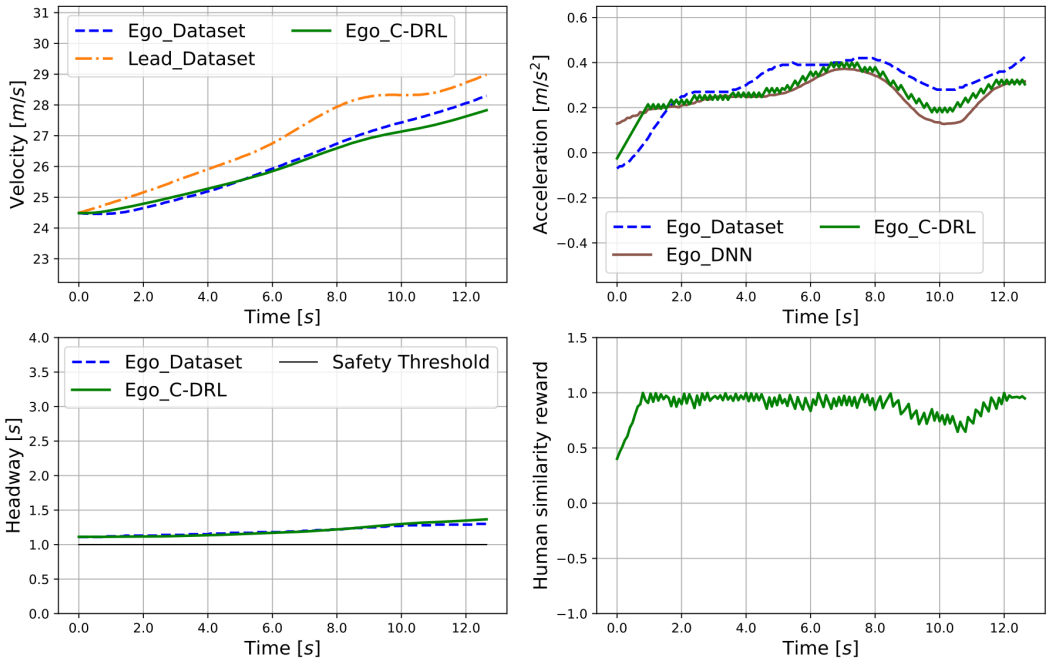

Fig. 4.20 Inference phase: C-DRL with action bounds - Normal driving style.

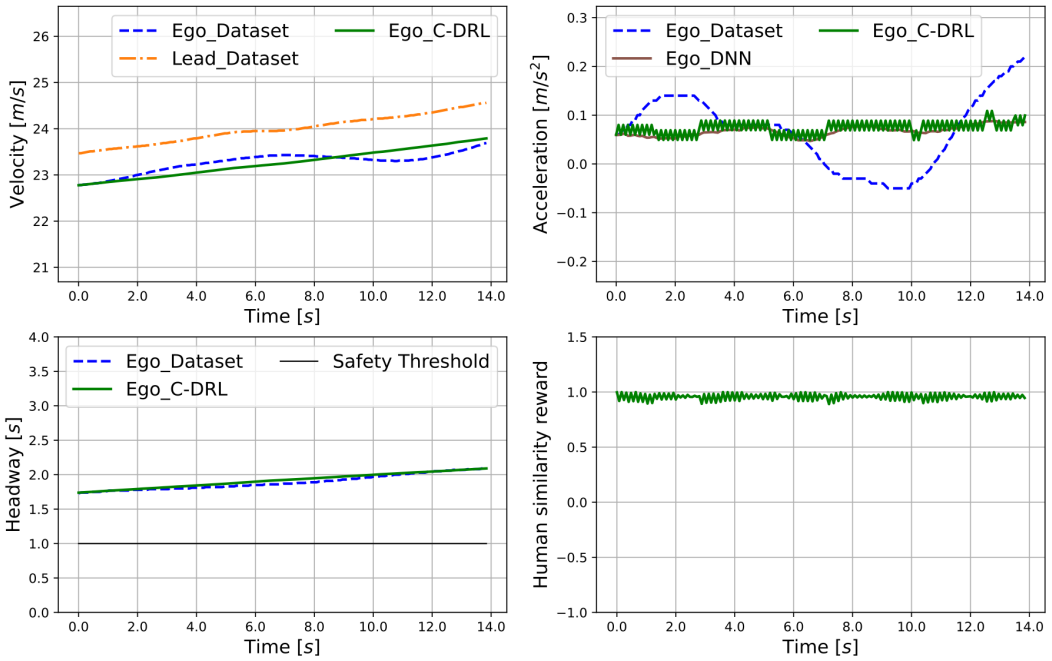

Fig. 4.21 Inference phase: C-DRL with action bounds - Conservative driving style.

For the inference phase, the same driving traces were utilized as earlier, and Figs. 4.19 to 4.21 depict the agent's performance with action bounds for the three driving styles. With action bounds, Fig. 4.19 illustrates that the aggressive agent decelerates smoothly to maintain a safe distance from the lead vehicle when the headway drops below the safety threshold, compared to the agent without action bounds. Once the safety threshold is met, the agent aims to maximize the reward (bottom left plot in Fig. 4.19) by taking control actions similar to the DNN regressor model human-like acceleration predictions. Similarly, Figs. 4.20 and 4.21 demonstrate that the agents effectively mimic human driving behavior. Furthermore, the root mean square error between the regressor and the C-DRL predicted acceleration yields errors with values of 0.2823, 0.0426, and 0.0132 for aggressive, normal, and conservative driving behavior, respectively. The error in aggressive driving behavior slightly reduced comparatively, while errors in normal and conservative driving behavior remain minimal in both versions of control action variations.

4.5 Chapter Conclusion

In this chapter, we focused on improving the reception of automated systems by addressing their limitations to safely mimic human-like driving styles. We proposed an adaptive autopilot (AA) framework that employs a constrained deep reinforcement learning approach to drive vehicles similarly to human drivers, adapting to diverse driving styles. The AA framework tackles three interconnected sub-problems: identifying driving styles using real-world data through a rule-based approach, predicting human-like acceleration across different driving styles using a DNN regressor model, and implementing a C-DRL approach to control vehicles while considering safety constraints and mimicking human-like behavior. The results from the DNN regressor, which serves as a human-like action predictor, indicate that the regressors can capture various driving behaviors and outperform state-of-the-art IDM models in predicting acceleration. Further, the results from the human-like driving C-DRL agents demonstrate its ability to replicate human behavior while adhering to safety constraints, which is particularly crucial when mimicking aggressive driving styles. Hence, the comfortable experience provided by the proposed AA framework has the potential to enhance the satisfaction of human drivers, leading to a reduced disengagement rate of the autopilot driving system.

Chapter 5

Conclusions

A recent WHO report highlights that nearly a million people die in road accidents annually, posing a significant challenge to achieving Vision Zero goals, particularly with the projected increase in the number of vehicles [12, 7, 6]. However, the recent digital transformation of vehicles, equipping them with advanced sensors, computing power, and wireless communication, fosters data-driven decision-making to address the challenges of current transportation systems. In this thesis, we have developed data-driven frameworks prioritizing safety, traffic efficiency, comfort, and driving experience. Specifically, these frameworks offer innovative solutions for preemptive collision avoidance, optimizing vehicle control during challenging maneuvers, and adapting vehicle behavior to diverse driving styles. To address the research questions outlined in Chapter 1, we have designed, developed, and evaluated three ML-aided frameworks.

In detail, Chapter 2 introduced a novel ML-aided, uncertainty-aware framework that was designed to enhance safety-critical collision detection methods in a densely populated urban environment. The 5G MEC hosts the Intersection Manager, which gathers relevant data from the surrounding traffic actors, to forecast vehicle positions using LSTM-based prediction models. Further, to account for dynamic traffic situations in complex urban settings, additional LSTM models were trained to estimate the uncertainty associated with the predicted trajectories. Furthermore, the framework adopted an ensemble ML approach, leveraging both predictions and their uncertainties to detect collisions more effectively. We evaluated the framework's performance at two intersections in Monaco and Luxembourg, with mobility traces

reflecting real-world conditions. The proposed framework outperformed the traditional approaches in predicting vehicles' future positions accurately, which is vital for collision detection. Unlike traditional methods that only use predicted positions, our framework detected collisions well in advance, resulting in a 61% improvement in the median reaction time available for drivers to react. Moreover, apart from detecting the collisions, reducing the likelihood of false positives is crucial, as frequent false alarms could lead to drivers disregarding the alerts. Compared with the state-of-the-art collision detection methods, our ML-aided framework not only successfully identified all collisions but also achieved a significantly lower rate of false positives. Given the timely detection of collisions, automated vehicles could avoid collisions at both intersections, showcasing the proposed framework's ability to address the research questions and contribute to the Vision Zero goals.

In Chapter 3, we addressed the challenges related to vehicle control with data-driven models that enhanced the performance of traditional ACC in challenging traffic scenarios. We introduced 2LL-CACC, an ML-aided DRL framework that employed a two-layered approach to accomplish road efficiency, safety, and comfort objectives. In this framework, maintaining a constant and safe inter-vehicle distance was used as a proxy to represent road efficiency, safety was represented by vehicle stability, and the rate of change in vehicle acceleration was used to represent comfort. The top layer hosts a context recognition model that appropriately weighs the targeted metrics based on the current contextual information. The lower layer hosts the DRL model, trained to find the optimal policy that balances safety, road efficiency, and comfort objectives by effectively controlling vehicle movements. Additionally, the framework exploited V2X communication to detect lane-changing vehicles before they appeared within the range of the onboard vehicle sensors. A gradual switching technique leveraged this early detection, enabling the ego vehicle to smoothly shift its focus to the new target vehicle, a crucial step in ensuring safety. The framework's performance was assessed using simulation scenarios involving vehicles entering into the ego vehicle's lane (cut-in) and exiting the lane, exposing the ego vehicle to slower traffic (cut-out), against state-of-the-art alternatives. Moreover, the framework employed a novel simulation framework (CoMoVe) to ensure realistic simulations of the scenarios. The results showed that the proposed framework achieved the best trade-off among the performance metrics and substantially outperformed its counterparts. The proposed framework achieved better road efficiency by maintaining the desired headway range for 73% of the time, while alternatives only respected

the desired range for 30% of the simulation time. Finally, the hardware-in-the-loop validation ensured the proposed framework could be deployed in vehicles and achieve the same results as the simulation. The assessment of the proposed approach demonstrated that data-driven decision-making can contribute to the advancement of automated vehicles.

In Chapter 4, we focused on addressing the disengagement rate of automated systems caused by their limitations in accommodating individual driving preferences. To address the limitation, we designed an adaptive autopilot framework that employed constrained deep reinforcement learning to emulate human-like driving behavior, tailoring vehicle control actions to accommodate diverse driving preferences. The framework achieved this by addressing three interconnected sub-problems: categorizing driving styles into aggressive, normal, and conservative using a rule-based approach; predicting human-like acceleration for these different styles using DNN-based regressor models; and employing a C-DRL approach to emulate human-like driving behavior across diverse driving styles, taking into account safety constraints. The performance results of the DNN regression models demonstrated that they could predict human driver-like acceleration and could exhibit different driving behaviors under identical conditions based on their driving styles. Moreover, compared to the state-of-the-art IDM approach, our regression models' proved to be better at predicting human-like acceleration values. The C-DRL agents' performance results demonstrated that it could adapt its control action to different driving styles, imitating how a human driver would drive by closely following the regressors' predicted acceleration whenever the safety constraint was satisfied. Furthermore, the non-constrained DRL agent's results revealed that its control actions led to unsafe situations, especially when imitating aggressive driving styles. This emphasizes the importance of the safety-constrained agent in adapting driving behavior to different styles. By safely mimicking human-like driving behavior, the proposed approach has the potential to improve the human driver's satisfaction and driving experience, thereby reducing interruptions in the autopilot driving systems.

With these novel frameworks, the thesis addresses the drawbacks of traditional approaches and contributes to the advancement of vehicular technology, taking a step towards the realization of autonomous vehicles.

5.1 Future Works

In the context of the thesis, the future research directions are outlined below.

Transfer learning: In Chapter 2, we trained the LSTM networks for two intersections, starting from scratch. Considering time efficiency, it is not practical to repeat this process for every intersection. To address this, transfer learning techniques can be explored to retain information from pre-trained trajectory prediction and uncertainty estimation models, thereby reducing training time and improving the model's applicability across various intersections.

Real-time validation: To assess the scalability of the collision detection approach presented in Chapter 2 in extensive scenarios, real-time validation can be conducted using a cellular testbed equipped with the Open Air Interface (OAI). This validation, leveraging real cellular communication in realistic conditions, aims to assess both the scalability of the system and the impact of potential communication losses on the collision avoidance application. Additionally, it will help determine the computational resources required for real-world deployment.

In-vehicle testing: Even though we have validated our implementation through rapid prototyping in Chapter 3, further testing with a vehicle in a controlled environment is essential before deploying in the real world. This will help validate the effectiveness and efficiency of the proposed frameworks under real-world conditions. Additionally, this testing will allow us to identify any potential limitations of the frameworks and further fine-tune them to achieve the desired outcome.

Lateral vehicle control: In this thesis, we primarily focused on the longitudinal control dynamics, particularly in car-following scenarios, to enhance traffic efficiency, safety, and comfort. On the other hand, lateral vehicle control also plays a significant role in maneuvering vehicles through challenging scenarios. A comprehensive control strategy encompassing both longitudinal and lateral controls of the vehicle needs to be explored to further strengthen the proposed frameworks.

Vehicle navigation: Traditional route planning services offer the same routes to all drivers, considering only a few standardized factors such as travel distance or time, without taking into account individual driver preferences. With the inception of autonomous vehicles expected in the coming years, these vehicles will begin employing such planners to determine their routes. Therefore, to enhance the travel experience, routing options based on passenger preferences can be explored.

Effective methodologies to incorporate human preferences in route selection, along with the usual traffic parameters, are crucial to offer personalized routes for drivers.

Comprehensive simulation framework: In this thesis, we have explored various frameworks designed to enhance the functionalities of automated vehicles, focusing on safety, traffic efficiency, comfort, and driving experiences. Future research could focus on developing a comprehensive framework that encompasses perception, planning, and vehicle control. In terms of perception, enhancing object detection through context-aware sensor fusion techniques and employing cooperative sensing could be pivotal. Utilizing Mobile Edge Computing (MEC) servers, multiple vehicles could share their sensor viewpoints along with roadside traffic cameras to construct a comprehensive view of the vehicular environment, thereby improving situational awareness beyond each vehicle's immediate sensory range. Moreover, the proposed uncertainty-aware trajectory prediction approach could be expanded to predict the movements and behaviors of surrounding vehicles more accurately, helping the movement of the autonomous vehicles in a safe and efficient manner. Additionally, control strategies could be developed to support drivers in adverse weather conditions by leveraging cooperative sensing and sensor fusion, further enhancing safety.

Explainable AI: To further strengthen the proposed frameworks, a critical future research direction involves providing human-interpretable AI decisions to gain trust and acceptance among users for widespread adoption. Generally, data-driven models that utilize artificial intelligence (AI) methods are referred to as "black box" models because their decisions are difficult to interpret. The objective of Explainable AI (XAI) is to provide interpretable AI decisions to end-users by summarizing the results of "black box" models with human-comprehensible explanations. This is crucial for building trust, ensuring compliance with regulations, and facilitating informed decision-making based on AI outputs.

References

[1] Department of Economic United Nations and Population Division (2019) Social Affairs. World urbanization prospects: The 2018 revision. Technical report, United Nations. URL https://population.un.org/wup/Publications/Files/WUP2018-Report.pdf.

[2] Health Effects Institute. Traffic-related air pollution: A critical review of the literature on emissions, exposure, and health effects: Executive summary. URL https://www.healtheffects.org/system/files/SR17TrafficReview_Exec_Summary.pdf.

[3] INRIX. INRIX Global Traffic Scorecard: Congestion cost UK economy £6.9 billion in 2019, 2019. URL https://inrix.com/press-releases/2019-traffic-scorecard-uk/. Publication Title: Inrix.

[4] European Commission. COMMUNICATION FROM THE COMMISSION TO THE EUROPEAN PARLIAMENT, THE COUNCIL, THE EUROPEAN ECONOMIC AND SOCIAL COMMITTEE AND THE COMMITTEE OF THE REGIONS A European strategy on Cooperative Intelligent Transport Systems, a milestone towards cooperative, connected and automated mobility. Technical report, European Commission, 2016. URL https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52016DC0766.

[5] Dinesh Cyril Selvaraj, Shailesh Hegde, Nicola Amati, Francesco Deflorio, and Carla Fabiana Chiasserini. A Deep Reinforcement Learning Approach for Efficient, Safe and Comfortable Driving. *Applied Sciences*, 13(9):5272, April 2023. ISSN 2076-3417. doi: 10.3390/app13095272. URL https://www.mdpi.com/2076-3417/13/9/5272.

[6] Michael Gross. A planet with two billion cars. *Current Biology*, 26(8):R307–R310, April 2016. ISSN 09609822. doi: 10.1016/j.cub.2016.04.019. URL https://linkinghub.elsevier.com/retrieve/pii/S0960982216303414.

[7] European Climate, Infrastructure and Environment Executive Agency. *EU road safety: towards "Vision Zero"*. Publications Office, LU, 2022. URL https://data.europa.eu/doi/10.2840/701809.

[8] EUR-Lex - 32019R0631 - EN - EUR-Lex, . URL https://eur-lex.europa.eu/eli/reg/2019/631/oj. Doc ID: 32019R0631 Doc Sector: 3 Doc Title: Regulation

(EU) 2019/631 of the European Parliament and of the Council of 17 April 2019 setting CO2 emission performance standards for new passenger cars and for new light commercial vehicles, and repealing Regulations (EC) No 443/2009 and (EU) No 510/2011 (recast) (Text with EEA relevance.) Doc Type: R Usr_lan: en.

[9] Falko Dressler, Carla Fabiana Chiasserini, Frank H. P. Fitzek, Holger Karl, Renato Lo Cigno, Antonio Capone, Claudio Ettore Casetti, Francesco Malandrino, Vincenzo Mancuso, Florian Klingler, and Gianluca A. Rizzo. V-Edge: Virtual Edge Computing as an Enabler for Novel Microservices and Cooperative Computing. *IEEE Network*, 36(3):24–31, May 2022. ISSN 1558-156X. doi: 10.1109/MNET.001.2100491.

[10] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A Survey of Autonomous Driving: *Common Practices and Emerging Technologies*. *IEEE Access*, 8:58443–58469, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.2983149. URL https://ieeexplore.ieee.org/document/9046805/.

[11] On-Road Automated Driving (ORAD) committee. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. Technical report, SAE International. URL https://www.sae.org/content/j3016_202104.

[12] Global status report on road safety 2023, . URL https://www.who.int/publications-detail-redirect/9789240086517.

[13] Prateek Bansal and Kara M. Kockelman. Forecasting Americans' long-term adoption of connected and autonomous vehicle technologies. *Transportation Research Part A: Policy and Practice*, 95:49–63, January 2017. ISSN 09658564. doi: 10.1016/j.tra.2016.10.013. URL https://linkinghub.elsevier.com/retrieve/pii/S0965856415300628.

[14] Andres E. Gomez, Sebastien Glaser, Yasser Alayli, Arthur De M. Neto, and Denis F. Wolf. Cooperative collision warning for driving assistance. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 990–997, Rio de Janeiro, Brazil, November 2016. IEEE. ISBN 978-1-5090-1889-5. doi: 10.1109/ITSC.2016.7795676. URL http://ieeexplore.ieee.org/document/7795676/.

[15] Yunpeng Wang, E. Wenjuan, Daxin Tian, Guangquan Lu, Guizhen Yu, and Yifan Wang. Vehicle collision warning system and collision detection algorithm based on vehicle infrastructure integration. In *7th Advanced Forum on Transportation of China (AFTC 2011)*, pages 216–220, Beijing, China, 2011. IET. ISBN 978-1-84919-571-3. doi: 10.1049/cp.2011.1407. URL https://digital-library.theiet.org/content/conferences/10.1049/cp.2011.1407.

[16] Marco Malinverno, Giuseppe Avino, Claudio Casetti, Carla Fabiana Chiasserini, Francesco Malandrino, and Salvatore Scarpina. Edge-Based Collision Avoidance for Vehicles and Vulnerable Users: An Architecture Based on

MEC. *IEEE Vehicular Technology Magazine*, 15(1):27–35, March 2020. ISSN 1556-6072, 1556-6080. doi: 10.1109/MVT.2019.2953770. URL https://ieeexplore.ieee.org/document/8943265/.

[17] Dinesh Cyril Selvaraj, Christian Vitale, Tania Panayiotou, Panayiotis Kolios, Carla Fabiana Chiasserini, and Georgios Ellinas. Edge Learning of Vehicular Trajectories at Regulated Intersections. In *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, pages 1–7, Norman, OK, USA, September 2021. IEEE. ISBN 978-1-66541-368-8. doi: 10.1109/VTC2021-Fall52928.2021.9625570. URL https://ieeexplore.ieee.org/document/9625570/.

[18] Xin Huang, Stephen G. McGill, Brian C. Williams, Luke Fletcher, and Guy Rosman. Uncertainty-Aware Driver Trajectory Prediction at Urban Intersections. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9718–9724, Montreal, QC, Canada, May 2019. IEEE. ISBN 978-1-5386-6027-0. doi: 10.1109/ICRA.2019.8794282. URL https://ieeexplore.ieee.org/document/8794282/.

[19] William Q. Jr Meeker, Gerald J. Hahn, and Luis A. Escobar. *Statistical intervals: a guide for practitioners and researchers*. John Wiley et sons, Hoboken, 2nd ed edition, 2017. ISBN 978-1-118-59484-1 978-1-118-59495-7 978-0-471-68717-7.

[20] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179.

[21] John Dahl, Gabriel Rodrigues De Campos, Claes Olsson, and Jonas Fredriksson. Collision Avoidance: A Literature Review on Threat-Assessment Techniques. *IEEE Transactions on Intelligent Vehicles*, 4(1):101–113, March 2019. ISSN 2379-8904, 2379-8858. doi: 10.1109/TIV.2018.2886682. URL https://ieeexplore.ieee.org/document/8574961/.

[22] Lu Tao, Yousuke Watanabe, Yixiao Li, Shunya Yamada, and Hiroaki Takada. Collision Risk Assessment Service for Connected Vehicles: Leveraging Vehicular State and Motion Uncertainties. *IEEE Internet of Things Journal*, 8 (14):11548–11560, July 2021. ISSN 2327-4662, 2372-2541. doi: 10.1109/JIOT.2021.3059222. URL https://ieeexplore.ieee.org/document/9353721/.

[23] Ruifeng Zhang, Libo Cao, Shan Bao, and Jianjie Tan. A method for connected vehicle trajectory prediction and collision warning algorithm based on V2V communication. *International Journal of Crashworthiness*, 22(1):15–25, January 2017. ISSN 1358-8265, 1754-2111. doi: 10.1080/13588265.2016.

1215584. URL https://www.tandfonline.com/doi/full/10.1080/13588265.2016.1215584.

[24] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: a survey. *The International Journal of Robotics Research*, 39(8):895–935, July 2020. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364920917446. URL http://journals.sagepub.com/doi/10.1177/0278364920917446.

[25] Florent Altche and Arnaud De La Fortelle. An LSTM network for highway trajectory prediction. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 353–359, Yokohama, October 2017. IEEE. ISBN 978-1-5386-1526-3. doi: 10.1109/ITSC.2017.8317913. URL http://ieeexplore.ieee.org/document/8317913/.

[26] Nachiket Deo and Mohan M. Trivedi. Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1179–1184, Changshu, June 2018. IEEE. ISBN 978-1-5386-4452-2. doi: 10.1109/IVS.2018.8500493. URL https://ieeexplore.ieee.org/document/8500493/.

[27] ByeoungDo Kim, Chang Mook Kang, Jaekyum Kim, Seung Hi Lee, Chung Choo Chung, and Jun Won Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 399–404, Yokohama, October 2017. IEEE. ISBN 978-1-5386-1526-3. doi: 10.1109/ITSC.2017.8317943. URL http://ieeexplore.ieee.org/document/8317943/.

[28] Yonghwan Jeong, Seonwook Kim, and Kyongsu Yi. Surround Vehicle Motion Prediction Using LSTM-RNN for Motion Planning of Autonomous Vehicles at Multi-Lane Turn Intersections. *IEEE Open Journal of Intelligent Transportation Systems*, 1:2–14, 2020. ISSN 2687-7813. doi: 10.1109/OJITS.2020.2965969. URL https://ieeexplore.ieee.org/document/8957421/.

[29] Yang Xing, Chen Lv, and Dongpu Cao. Personalized Vehicle Trajectory Prediction Based on Joint Time-Series Modeling for Connected Vehicles. *IEEE Transactions on Vehicular Technology*, 69(2):1341–1352, February 2020. ISSN 0018-9545, 1939-9359. doi: 10.1109/TVT.2019.2960110. URL https://ieeexplore.ieee.org/document/8933492/.

[30] Jinxin Liu, Yugong Luo, Hui Xiong, Tinghan Wang, Heye Huang, and Zhihua Zhong. An Integrated Approach to Probabilistic Vehicle Trajectory Prediction via Driver Characteristic and Intention Estimation. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3526–3532, Auckland, New Zealand, October 2019. IEEE. ISBN 978-1-5386-7024-8. doi: 10.1109/ITSC.2019.8917039. URL https://ieeexplore.ieee.org/document/8917039/.

[31] Quan Tran and Jonas Firl. Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 918–923, MI, USA, June 2014. IEEE. ISBN 978-1-4799-3638-0. doi: 10.1109/IVS.2014.6856480. URL http://ieeexplore.ieee.org/document/6856480/.

[32] Derek J. Phillips, Tim A. Wheeler, and Mykel J. Kochenderfer. Generalizable intention prediction of human drivers at intersections. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1665–1670, Los Angeles, CA, USA, June 2017. IEEE. ISBN 978-1-5090-4804-5. doi: 10.1109/IVS.2017.7995948. URL http://ieeexplore.ieee.org/document/7995948/.

[33] Wenquan Xu, Zhikang Chen, Chuwen Zhang, Xuefeng Ji, Yunsheng Wang, Hang Su, and Bin Liu. GlobalInsight: An LSTM Based Model for Multi-Vehicle Trajectory Prediction. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–7, Dublin, Ireland, June 2020. IEEE. ISBN 978-1-72815-089-5. doi: 10.1109/ICC40277.2020.9149261. URL https://ieeexplore.ieee.org/document/9149261/.

[34] Nachiket Deo and Mohan M. Trivedi. Convolutional Social Pooling for Vehicle Trajectory Prediction. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1549–15498, Salt Lake City, UT, USA, June 2018. IEEE. ISBN 978-1-5386-6100-0. doi: 10.1109/CVPRW.2018.00196. URL https://ieeexplore.ieee.org/document/8575356/.

[35] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. VectorNet: Encoding HD Maps and Agent Dynamics From Vectorized Representation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11522–11530, Seattle, WA, USA, June 2020. IEEE. ISBN 978-1-72817-168-5. doi: 10.1109/CVPR42600.2020.01154. URL https://ieeexplore.ieee.org/document/9157331/.

[36] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning Lane Graph Representations for Motion Forecasting. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, volume 12347, pages 541–556. Springer International Publishing, Cham, 2020. ISBN 978-3-030-58535-8 978-3-030-58536-5. doi: 10.1007/978-3-030-58536-5_32. URL https://link.springer.com/10.1007/978-3-030-58536-5_32. Series Title: Lecture Notes in Computer Science.

[37] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096, Montreal, QC, Canada, May 2019.

IEEE. ISBN 978-1-5386-6027-0. doi: 10.1109/ICRA.2019.8793868. URL https://ieeexplore.ieee.org/document/8793868/.

[38] Jan Strohbeck, Vasileios Belagiannis, Johannes Muller, Marcel Schreiber, Martin Herrmann, Daniel Wolf, and Michael Buchholz. Multiple Trajectory Prediction with Deep Temporal and Spatial Convolutional Neural Networks. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1992–1998, Las Vegas, NV, USA, October 2020. IEEE. ISBN 978-1-72816-212-6. doi: 10.1109/IROS45743.2020.9341327. URL https://ieeexplore.ieee.org/document/9341327/.

[39] Wei Liu and Yozo Shoji. Edge-Assisted Vehicle Mobility Prediction to Support V2X Communications. *IEEE Transactions on Vehicular Technology*, 68(10): 10227–10238, October 2019. ISSN 0018-9545, 1939-9359. doi: 10.1109/TVT.2019.2937825. URL https://ieeexplore.ieee.org/document/8818318/.

[40] Chung-Ming Huang and Shih-Yang Lin. An Advanced Vehicle Collision Warning Algorithm over the DSRC Communication Environment: An Advanced Vehicle Collision Warning Algorithm. In *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pages 696–702, Barcelona, March 2013. IEEE. ISBN 978-1-4673-5550-6 978-0-7695-4953-8. doi: 10.1109/AINA.2013.72. URL http://ieeexplore.ieee.org/document/6531822/.

[41] Dewant Katare and Mohamed El-Sharkawy. Embedded System Enabled Vehicle Collision Detection: An ANN Classifier. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0284–0289, Las Vegas, NV, USA, January 2019. IEEE. ISBN 978-1-72810-554-3. doi: 10.1109/CCWC.2019.8666562. URL https://ieeexplore.ieee.org/document/8666562/.

[42] Islam Elleuch, Achraf Makni, and Rafik Bouaziz. Cooperative Intersection Collision Avoidance Persistent System Based on V2V Communication and Real-Time Databases. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pages 1082–1089, Hammamet, October 2017. IEEE. ISBN 978-1-5386-3581-0. doi: 10.1109/AICCSA.2017.20. URL http://ieeexplore.ieee.org/document/8308410/.

[43] Michael R. Hafner, Drew Cunningham, Lorenzo Caminiti, and Domitilla Del Vecchio. Cooperative Collision Avoidance at Intersections: Algorithms and Experiments. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1162–1175, September 2013. ISSN 1524-9050, 1558-0016. doi: 10.1109/TITS.2013.2252901. URL https://ieeexplore.ieee.org/document/6495719.

[44] Gabriel R. De Campos, 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), pages 649–654, Qingdao, China, October 2014. IEEE. ISBN 978-1-4799-6078-1. doi: 10.1109/ITSC.2014.6957763. URL http://ieeexplore.ieee.org/document/6957763/.

[45] Mattias Brannstrom, Fredrik Sandblom, and Lars Hammarstrand. A Probabilistic Framework for Decision-Making in Collision Avoidance Systems. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):637–648, June 2013. ISSN 1524-9050, 1558-0016. doi: 10.1109/TITS.2012.2227474. URL http://ieeexplore.ieee.org/document/6374680/.

[46] Jie Ji, Amir Khajepour, Wael William Melek, and Yanjun Huang. Path Planning and Tracking for Vehicle Collision Avoidance Based on Model Predictive Control With Multiconstraints. *IEEE Transactions on Vehicular Technology*, 66(2):952–964, February 2017. ISSN 0018-9545, 1939-9359. doi: 10.1109/TVT.2016.2555853. URL http://ieeexplore.ieee.org/document/7458179/.

[47] Kibeom Lee and Dongsuk Kum. Collision Avoidance/Mitigation System: Motion Planning of Autonomous Vehicle via Predictive Occupancy Map. *IEEE Access*, 7:52846–52857, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2912067. URL https://ieeexplore.ieee.org/document/8698763/.

[48] Giuseppe Avino, Paolo Bande, Pantelis A. Frangoudis, Christian Vitale, Claudio Casetti, Carla Fabiana Chiasserini, Kalkidan Gebru, Adlen Ksentini, and Giuliana Zennaro. A MEC-Based Extended Virtual Sensing for Automotive Services. *IEEE Transactions on Network and Service Management*, 16(4):1450–1463, December 2019. ISSN 1932-4537, 2373-7379. doi: 10.1109/TNSM.2019.2931878. URL https://ieeexplore.ieee.org/document/8781832/.

[49] Bouziane Brik and Adlen Ksentini. Toward Optimal MEC Resource Dimensioning for a Vehicle Collision Avoidance System: A Deep Learning Approach. *IEEE Network*, 35(3):74–80, May 2021. ISSN 0890-8044, 1558-156X. doi: 10.1109/MNET.011.2000577. URL https://ieeexplore.ieee.org/document/9454593/.

[50] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994. ISSN 1045-9227, 1941-0093. doi: 10.1109/72.279181. URL https://ieeexplore.ieee.org/document/279181/.

[51] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, June 2016. PMLR. URL https://proceedings.mlr.press/v48/gal16.html.

[52] Filipe Rodrigues and Francisco C. Pereira. Beyond Expectation: Deep Joint Mean and Quantile Regression for Spatiotemporal Problems. *IEEE Transactions on Neural Networks and Learning Systems*, 31(12):5377–5389, December 2020. ISSN 2162-237X, 2162-2388. doi: 10.1109/TNNLS.2020.2966745. URL https://ieeexplore.ieee.org/document/8985289/.

[53] Roger Koenker. Fundamentals of Quantile Regression. In *Quantile Regression*, Econometric Society Monographs, pages 26–67. Cambridge University Press, 2005. ISBN 978-0-511-75409-8. URL http://ebooks.cambridge.org/ref/id/CBO9780511754098.

[54] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.1997.9.8.1735. URL https://direct.mit.edu/neco/article/9/8/1735-1780/6109.

[55] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*, January 2017. doi: 10.48550/arXiv.1412.6980. URL http://arxiv.org/abs/1412.6980. arXiv:1412.6980 [cs].

[56] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 08856125. doi: 10.1023/A:1010933404324. URL http://link.springer.com/10.1023/A:1010933404324.

[57] Gilles Louppe. Understanding Random Forests: From Theory to Practice. *ArXiv e-prints*, June 2015. doi: 10.48550/arXiv.1407.7502. URL http://arxiv.org/abs/1407.7502. arXiv:1407.7502 [stat].

[58] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, August 1998. ISSN 01628828. doi: 10.1109/34.709601. URL http://ieeexplore.ieee.org/document/709601/.

[59] Maria Isabel Ribeiro. Gaussian probability density functions: Properties and error characterization. Technical report, Inst. for Syst. and Robotics, Lisbon, Portugal, February 2004. URL http://users.isr.ist.utl.pt/~mir/pub/probability.pdf.

[60] Huisheng Ma, Shufang Li, Erqing Zhang, Zhengnan Lv, Jing Hu, and Xinlei Wei. Cooperative Autonomous Driving Oriented MEC-Aided 5G-V2X: Prototype System Design, Field Tests and AI-Based Optimization Tools. *IEEE Access*, 8:54288–54302, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.2981463. URL https://ieeexplore.ieee.org/document/9040539/.

[61] Sidi Lu and Weisong Shi. Vehicle Computing: Vision and challenges. *Journal of Information and Intelligence*, 1(1):23–35, May 2023. ISSN 29497159. doi: 10.1016/j.jiixd.2022.10.001. URL https://linkinghub.elsevier.com/retrieve/pii/S2949715922000038.

[62] Paweł Droździel, Sławomir Tarkowski, Iwona Rybicka, and Rafał Wrona. Drivers 'reaction time research in the conditions in the real traffic. *Open Engineering*, 10(1):35–47, January 2020. ISSN 2391-5439. doi: 10.1515/eng-2020-0004. URL https://www.degruyter.com/document/doi/10.1515/eng-2020-0004/html.

[63] Lara Codeca, Raphael Frank, Sebastien Faye, and Thomas Engel. Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation. *IEEE Intelligent Transportation Systems Magazine*, 9(2):52–63, 2017. ISSN 1939-1390. doi: 10.1109/MITS.2017.2666585. URL http://ieeexplore.ieee.org/document/7906642/.

[64] Lara Codeca and Jerome Harri. Towards multimodal mobility simulation of C-ITS: The Monaco SUMO traffic scenario. In *2017 IEEE Vehicular Networking Conference (VNC)*, pages 97–100, Torino, November 2017. IEEE. ISBN 978-1-5386-0986-6. doi: 10.1109/VNC.2017.8275627. URL http://ieeexplore.ieee.org/document/8275627/.

[65] Health Effects Institute Panel on the Health Effects of Traffic-Related Air Pollution. Traffic-related air pollution: a critical review of the literature on emissions, exposure, and health effects, 2010. Publisher: Health Effects Institute.

[66] SeHwan Kim, Junmin Wang, Dennis Guenther, Gary Heydinger, Joshua Every, M Kamel Salaani, and Frank Barickman. Analysis of human driver behavior in highway cut-in scenarios. Technical report, SAE Technical Paper, 2017.

[67] Eric Thorn, Shawn C. Kimmel, Michelle Chaka, Virginia Tech Transportation Institute, Southwest Research Institute, and Inc. Booz Allen Hamilton. A Framework for Automated Driving System Testable Cases and Scenarios. Technical Report DOT HS 812 623, United States. Department of Transportation. National Highway Traffic Safety, September 2018. URL https://rosap.ntl.bts.gov/view/dot/38824.

[68] Sanhita Das and Akhilesh Kumar Maurya. Time Headway Analysis for Four-Lane and Two-Lane Roads. *Transportation in Developing Economies*, 3(1):9, April 2017. ISSN 2199-9287, 2199-9295. doi: 10.1007/s40890-017-0039-8. URL http://link.springer.com/10.1007/s40890-017-0039-8.

[69] Yue Li, Huapu Lu, Xinxin Yu, and Ya Gang Sui. Traffic flow headway distribution and capacity analysis using urban arterial road data. In *2011 International Conference on Electric Technology and Civil Engineering (ICETCE)*, pages 1821–1824, Lushan, China, April 2011. IEEE. ISBN 978-1-4577-0289-1. doi: 10.1109/ICETCE.2011.5775326. URL http://ieeexplore.ieee.org/document/5775326/.

[70] A. Shrivastava and P.Y. Li. Traffic flow stability induced by constant time headway policy for adaptive cruise control (ACC) vehicles. In *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*,

pages 1503–1508 vol.3, Chicago, IL, USA, 2000. IEEE. ISBN 978-0-7803-5519-4. doi: 10.1109/ACC.2000.879450. URL http://ieeexplore.ieee.org/document/879450/.

[71] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *ArXiv e-prints*, 2015. doi: 10.48550/ARXIV.1509.02971. URL https://arxiv.org/abs/1509.02971. Publisher: [object Object] Version Number: 6.

[72] EuroNCAP. 2018 Automated Driving Tests, 2018. URL https://www.euroncap.com/en/vehicle-safety/safety-campaigns/2018-automated-driving-tests/.

[73] Beshr Sultan, Mark Brackstone, Ben Waterson, and Erwin R. Boer. Modeling the Dynamic Cut-In Situation. *Transportation Research Record: Journal of the Transportation Research Board*, 1803(1):45–51, January 2002. ISSN 0361-1981, 2169-4052. doi: 10.3141/1803-07. URL http://journals.sagepub.com/doi/10.3141/1803-07.

[74] Sampo Kuutti, Richard Bowden, Yaochu Jin, Phil Barber, and Saber Fallah. A Survey of Deep Learning Applications to Autonomous Vehicle Control. *IEEE Transactions on Intelligent Transportation Systems*, 22(2):712–733, February 2021. ISSN 1524-9050, 1558-0016. doi: 10.1109/TITS.2019.2962338. URL https://ieeexplore.ieee.org/document/8951131/.

[75] C. Desjardins and B. Chaib-draa. Cooperative Adaptive Cruise Control: A Reinforcement Learning Approach. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1248–1260, December 2011. ISSN 1524-9050, 1558-0016. doi: 10.1109/TITS.2011.2157145. URL http://ieeexplore.ieee.org/document/5876320/.

[76] Subramanya Nageshrao, H. Eric Tseng, and Dimitar Filev. Autonomous Highway Driving using Deep Reinforcement Learning. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 2326–2331, Bari, Italy, October 2019. IEEE. ISBN 978-1-72814-569-3. doi: 10.1109/SMC.2019.8914621. URL https://ieeexplore.ieee.org/document/8914621/.

[77] Meixin Zhu, Yinhai Wang, Ziyuan Pu, Jingyun Hu, Xuesong Wang, and Ruimin Ke. Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving. *Transportation Research Part C: Emerging Technologies*, 117:102662, August 2020. ISSN 0968090X. doi: 10.1016/j.trc.2020.102662. URL https://linkinghub.elsevier.com/retrieve/pii/S0968090X20305775.

[78] Qingyun Chen, Wanzhong Zhao, Lin Li, Chunyan Wang, and Feng Chen. ES-DQN: A Learning Method for Vehicle Intelligent Speed Control Strategy Under Uncertain Cut-In Scenario. *IEEE Transactions on Vehicular Technology*, 71(3):2472–2484, March 2022. ISSN 0018-9545, 1939-9359.

doi: 10.1109/TVT.2022.3143840. URL https://ieeexplore.ieee.org/document/9684932/.

[79] Pin Wang, Ching-Yao Chan, and Arnaud De La Fortelle. A Reinforcement Learning Based Approach for Automated Lane Change Maneuvers. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1379–1384, Changshu, June 2018. IEEE. ISBN 978-1-5386-4452-2. doi: 10.1109/IVS.2018.8500556. URL https://ieeexplore.ieee.org/document/8500556/.

[80] Angelo Bonfitto, Stefano Feraco, Andrea Tonoli, and Nicola Amati. Combined regression and classification artificial neural networks for sideslip angle estimation and road condition identification. *Vehicle System Dynamics*, 58(11):1766–1787, November 2020. ISSN 0042-3114, 1744-5159. doi: 10.1080/00423114.2019.1645860. URL https://www.tandfonline.com/doi/full/10.1080/00423114.2019.1645860.

[81] You-Qun Zhao, Hai-Qing Li, Fen Lin, Jian Wang, and Xue-Wu Ji. Estimation of Road Friction Coefficient in Different Road Conditions Based on Vehicle Braking Dynamics. *Chinese Journal of Mechanical Engineering*, 30(4):982–990, July 2017. ISSN 1000-9345, 2192-8258. doi: 10.1007/s10033-017-0143-z. URL http://link.springer.com/10.1007/s10033-017-0143-z.

[82] Sulakshan Rajendran, Sarah K Spurgeon, Georgios Tsampardoukas, and Ric Hampson. Estimation of road frictional force and wheel slip for effective antilock braking system (ABS) control. *International Journal of Robust and Nonlinear Control*, 29(3):736–765, February 2019. ISSN 1049-8923, 1099-1239. doi: 10.1002/rnc.4366. URL https://onlinelibrary.wiley.com/doi/10.1002/rnc.4366.

[83] Ashwin Carvalho, Alek Williams, Stéphanie Lefevre, and Francesco Borrelli. Autonomous cruise control with cut-in target vehicle detection. In *Advanced Vehicle Control*, pages 92–98. CRC Press, 2016.

[84] Vicente Milanés and Steven E. Shladover. Handling Cut-In Vehicles in Strings of Cooperative Adaptive Cruise Control Vehicles. *Journal of Intelligent Transportation Systems*, 20(2):178–191, March 2016. ISSN 1547-2450, 1547-2442. doi: 10.1080/15472450.2015.1016023. URL https://www.tandfonline.com/doi/full/10.1080/15472450.2015.1016023.

[85] Chongpu Chen, Jianhua Guo, Chong Guo, Chaoyi Chen, Yao Zhang, and Jiawei Wang. Adaptive Cruise Control for Cut-In Scenarios Based on Model Predictive Control Algorithm. *Applied Sciences*, 11(11):5293, June 2021. ISSN 2076-3417. doi: 10.3390/app11115293. URL https://www.mdpi.com/2076-3417/11/11/5293.

[86] Yuan Lin, John McPhee, and Nasser L. Azad. Comparison of Deep Reinforcement Learning and Model Predictive Control for Adaptive Cruise Control. *IEEE Transactions on Intelligent Vehicles*, 6(2):221–231, June

2021. ISSN 2379-8904, 2379-8858. doi: 10.1109/TIV.2020.3012947. URL https://ieeexplore.ieee.org/document/9152161/.

[87] Guillaume Matheron, Nicolas Perrin, and Olivier Sigaud. Understanding Failures of Deterministic Actor-Critic with Continuous Action Spaces and Sparse Rewards. In Igor Farkaš, Paolo Masulli, and Stefan Wermter, editors, *Artificial Neural Networks and Machine Learning – ICANN 2020*, volume 12397, pages 308–320. Springer International Publishing, Cham, 2020. ISBN 978-3-030-61615-1 978-3-030-61616-8. doi: 10.1007/978-3-030-61616-8_25. URL https://link.springer.com/10.1007/978-3-030-61616-8_25. Series Title: Lecture Notes in Computer Science.

[88] Farzan Memarian, Wonjoon Goo, Rudolf Lioutikov, Scott Niekum, and Ufuk Topcu. Self-Supervised Online Reward Shaping in Sparse-Reward Environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2369–2375, Prague, Czech Republic, September 2021. IEEE. ISBN 978-1-66541-714-3. doi: 10.1109/IROS51168.2021.9636020. URL https://ieeexplore.ieee.org/document/9636020/.

[89] Zhaohui Hu, Dongbin Zhao, and Institute of Automation, Chinese Academy of Sciences, No.95, Zhongguancun East Road, Beijing 100190, China. Adaptive Cruise Control Based on Reinforcement Leaning with Shaping Rewards. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 15(3):351–356, May 2011. ISSN 1883-8014, 1343-0130. doi: 10.20965/jaciii.2011.p0351. URL https://www.fujipress.jp/jaciii/jc/jacii001500030351.

[90] Yingjun Ye, Xiaohui Zhang, and Jian Sun. Automated vehicle's behavior decision making using deep reinforcement learning and high-fidelity simulation environment. *Transportation Research Part C: Emerging Technologies*, 107: 155–170, October 2019. ISSN 0968090X. doi: 10.1016/j.trc.2019.08.011. URL https://linkinghub.elsevier.com/retrieve/pii/S0968090X19311301.

[91] Dinesh Cyril Selvaraj, Shailesh Hegde, Carla Fabiana Chiasserini, Nicola Amati, Francesco Deflorio, and Giuliana Zennaro. A Full-fledge Simulation Framework for the Assessment of Connected Cars. *Transportation Research Procedia*, 52:315–322, 2021. ISSN 23521465. doi: 10.1016/j.trpro.2021.01.037. URL https://linkinghub.elsevier.com/retrieve/pii/S2352146521000697.

[92] Economic Commission for Europe. Proposal for a new draft UN Regulation on the approval of motor vehicles with regard to their Advanced Emergency Braking System for M1 and N1 vehicles, 2019. URL https://unece.org/DAM/trans/doc/2019/wp29/ECE-TRANS-WP29-2019-61e.pdf.

[93] Il Bae, Jaeyoung Moon, Junekyo Jhung, Ho Suk, Taewoo Kim, Hyungbin Park, Jaekwang Cha, Jinhyuk Kim, Dohyun Kim, and Shiho Kim. Self-Driving like a Human driver instead of a Robocar: Personalized comfortable driving experience for autonomous vehicles. *ArXiv e-prints*, 2020. doi: 10.48550/ARXIV.2001.03908. URL https://arxiv.org/abs/2001.03908. Publisher: [object Object] Version Number: 2.

[94] United Nations Economic Commission for Europe (UNECE). Proposal for a New UN Regulation on Uniform Provisions Concerning the Approval of Vehicles with Regards to Automated Lane Keeping System, 2020.

[95] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-Learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, volume 30 of *AAAI'16*, pages 2094–2100. AAAI Press, 2016. Issue: 1 Place: Phoenix, Arizona.

[96] Liangyao Yu and Ruyue Wang. Researches on Adaptive Cruise Control system: A state of the art review. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 236(2-3):211–240, February 2022. ISSN 0954-4070, 2041-2991. doi: 10.1177/09544070211019254. URL http://journals.sagepub.com/doi/10.1177/09544070211019254.

[97] M. Kyriakidis, R. Happee, and J.C.F. De Winter. Public opinion on automated driving: Results of an international questionnaire among 5000 respondents. *Transportation Research Part F: Traffic Psychology and Behaviour*, 32:127–140, July 2015. ISSN 13698478. doi: 10.1016/j.trf.2015.04.014. URL https://linkinghub.elsevier.com/retrieve/pii/S1369847815000777.

[98] David R. Large, Kyle Harrington, Gary Burnett, Jacob Luton, Peter Thomas, and Pete Bennett. To Please in a Pod: Employing an Anthropomorphic Agent-Interlocutor to Enhance Trust and User Experience in an Autonomous, Self-Driving Vehicle. In *Proceedings of the 11th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 49–59, Utrecht Netherlands, September 2019. ACM. ISBN 978-1-4503-6884-1. doi: 10.1145/3342197.3344545. URL https://dl.acm.org/doi/10.1145/3342197.3344545.

[99] Lia Morra, Fabrizio Lamberti, F. Gabriele Prattico, Salvatore La Rosa, and Paolo Montuschi. Building Trust in Autonomous Vehicles: Role of Virtual Reality Driving Simulators in HMI Design. *IEEE Transactions on Vehicular Technology*, 68(10):9438–9450, October 2019. ISSN 0018-9545, 1939-9359. doi: 10.1109/TVT.2019.2933601. URL https://ieeexplore.ieee.org/document/8789466/.

[100] Zheng Ma and Yiqi Zhang. Drivers trust, acceptance, and takeover behaviors in fully automated vehicles: Effects of automated driving styles and driver's driving styles. *Accident Analysis & Prevention*, 159:106238, September 2021. ISSN 00014575. doi: 10.1016/j.aap.2021.106238. URL https://linkinghub.elsevier.com/retrieve/pii/S0001457521002694.

[101] Zheng Ma and Yiqi Zhang. Investigating the Effects of Automated Driving Styles and Driver's Driving Styles on Driver Trust, Acceptance, and Take Over Behaviors. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 64(1):2001–2005, December 2020. ISSN 2169-5067, 1071-1813. doi: 10.1177/1071181320641484. URL http://journals.sagepub.com/doi/10.1177/1071181320641484.

[102] Fridulv Sagberg, Selpi, Giulio Francesco Bianchi Piccinini, and Johan Engström. A Review of Research on Driving Styles and Road Safety. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 57(7):1248–1275, November 2015. ISSN 0018-7208, 1547-8181. doi: 10.1177/0018720815591313. URL http://journals.sagepub.com/doi/10.1177/0018720815591313.

[103] Arne Kesting, Martin Treiber, and Dirk Helbing. Agents for Traffic Simulation. *ArXiv e-prints*, 2008. doi: 10.48550/ARXIV.0805.0300. URL https://arxiv.org/abs/0805.0300. Publisher: [object Object] Version Number: 1.

[104] Martin Treiber and Arne Kesting. Modeling Human Aspects of Driving Behavior. In *Traffic Flow Dynamics*, pages 205–224. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-32459-8 978-3-642-32460-4. doi: 10.1007/978-3-642-32460-4_12. URL https://link.springer.com/10.1007/978-3-642-32460-4_12.

[105] S Krauss. Microscopic modeling of traffic flow: investigation of collision free vehicle dynamics, April 1998.

[106] P.G. Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111, April 1981. ISSN 01912615. doi: 10.1016/0191-2615(81)90037-0. URL https://linkinghub.elsevier.com/retrieve/pii/0191261581900370.

[107] Meixin Zhu, Xuesong Wang, and Yinhai Wang. Human-like autonomous car-following model with deep reinforcement learning. *Transportation Research Part C: Emerging Technologies*, 97:348–368, December 2018. ISSN 0968090X. doi: 10.1016/j.trc.2018.10.024. URL https://linkinghub.elsevier.com/retrieve/pii/S0968090X1830055X.

[108] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Perez. Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, June 2022. ISSN 1524-9050, 1558-0016. doi: 10.1109/TITS.2021.3054625. URL https://ieeexplore.ieee.org/document/9351818/.

[109] Dinesh Cyril Selvaraj, Shailesh Hegde, Nicola Amati, Francesco Deflorio, and Carla Fabiana Chiasserini. An ML-Aided Reinforcement Learning Approach for Challenging Vehicle Maneuvers. *IEEE Transactions on Intelligent Vehicles*, 8(2):1686–1698, February 2023. ISSN 2379-8904, 2379-8858. doi: 10.1109/TIV.2022.3224656. URL https://ieeexplore.ieee.org/document/9963702/.

[110] Eitan Altman. *Constrained Markov Decision Processes: Stochastic Modeling*. Routledge, Boca Raton, 1 edition, December 2021. ISBN 978-1-315-14022-3. doi: 10.1201/9781315140223. URL https://www.taylorfrancis.com/books/9781315140223.

[111] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125, Maui, HI, November 2018. IEEE. ISBN 978-1-72810-321-1 978-1-72810-323-5. doi: 10.1109/ITSC.2018.8569552. URL https://ieeexplore.ieee.org/document/8569552/.

[112] Julien Roy, Roger Girgis, Joshua Romoff, Pierre-Luc Bacon, and Christopher Pal. Direct Behavior Specification via Constrained Reinforcement Learning. *ArXiv e-prints*, 2021. doi: 10.48550/ARXIV.2112.12228. URL https://arxiv.org/abs/2112.12228. Publisher: [object Object] Version Number: 6.

[113] T. Watanabe, N. Kishimoto, K. Hayafune, K. Yamada, and N. Maede. Development of an intelligent cruise control system. November 1995. URL https://trid.trb.org/View/462217.

[114] Tianya Terry Zhang, Ph. D., Peter J. Jin, Ph. D., Sean T. McQuade, Ph. D., Alexandre Bayen, Ph. D., and Benedetto Piccoli. Car-Following Models: A Multidisciplinary Review. *ArXiv e-prints*, 2023. doi: 10.48550/ARXIV.2304.07143. URL https://arxiv.org/abs/2304.07143. Publisher: [object Object] Version Number: 4.

[115] Vasileia Papathanasopoulou and Constantinos Antoniou. Towards data-driven car-following models. *Transportation Research Part C: Emerging Technologies*, 55:496–509, June 2015. ISSN 0968090X. doi: 10.1016/j.trc.2015.02.016. URL https://linkinghub.elsevier.com/retrieve/pii/S0968090X15000716.

[116] Alireza Khodayari, Ali Ghaffari, Reza Kazemi, and Reinhard Braunstingl. A Modified Car-Following Model Based on a Neural Network Model of the Human Driver Effects. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 42(6):1440–1449, November 2012. ISSN 1083-4427, 1558-2426. doi: 10.1109/TSMCA.2012.2192262. URL http://ieeexplore.ieee.org/document/6193221/.

[117] Xiao Wang, Rui Jiang, Li Li, Yilun Lin, Xinhu Zheng, and Fei-Yue Wang. Capturing Car-Following Behaviors by Deep Learning. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):910–920, March 2018. ISSN 1524-9050, 1558-0016. doi: 10.1109/TITS.2017.2706963. URL http://ieeexplore.ieee.org/document/7970189/.

[118] Huanghuang Liang, Lu Yang, Hong Cheng, Wenzhe Tu, and Mengjie Xu. Human-in-the-loop reinforcement learning. In *2017 Chinese Automation Congress (CAC)*, pages 4511–4518, Jinan, October 2017. IEEE. ISBN 978-1-5386-3524-7. doi: 10.1109/CAC.2017.8243575. URL http://ieeexplore.ieee.org/document/8243575/.

[119] Jingda Wu, Zhiyu Huang, Chao Huang, Zhongxu Hu, Peng Hang, Yang Xing, and Chen Lv. Human-in-the-Loop Deep Reinforcement Learning with

Application to Autonomous Driving. 2021. doi: 10.48550/ARXIV.2104.07246. URL https://arxiv.org/abs/2104.07246. Publisher: [object Object] Version Number: 1.

[120] Yantao Tian, Xuanhao Cao, Kai Huang, Cong Fei, Zhu Zheng, and Xuewu Ji. Learning to Drive Like Human Beings: A Method Based on Deep Reinforcement Learning. *IEEE Transactions on Intelligent Transportation Systems*, 23 (7):6357–6367, July 2022. ISSN 1524-9050, 1558-0016. doi: 10.1109/TITS.2021.3055899. URL https://ieeexplore.ieee.org/document/9352233/.

[121] Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. Learning to Walk in the Real World with Minimal Human Effort. *ArXiv e-prints*, 2020. doi: 10.48550/ARXIV.2002.08550. URL https://arxiv.org/abs/2002.08550. Publisher: [object Object] Version Number: 3.

[122] Qisong Yang, Thiago D. Simão, Simon H Tindemans, and Matthijs T. J. Spaan. WCSAC: Worst-Case Soft Actor Critic for Safety-Constrained Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35 (12):10639–10646, May 2021. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v35i12.17272. URL https://ojs.aaai.org/index.php/AAAI/article/view/17272.

[123] Dominik Salles, Stefan Kaufmann, and Hans-Christian Reuss. Extending the Intelligent Driver Model in SUMO and Verifying the Drive Off Trajectories with Aerial Measurements. *SUMO Conference Proceedings*, 1:1–25, June 2022. ISSN 2750-4425. doi: 10.52825/scp.v1i.95. URL https://www.tib-op.org/ojs/index.php/scp/article/view/95.

[124] Guy Hacohen and Daphna Weinshall. On The Power of Curriculum Learning in Training Deep Networks. *ArXiv e-prints*, 2019. doi: 10.48550/ARXIV.1904.03626. URL https://arxiv.org/abs/1904.03626. Publisher: [object Object] Version Number: 3.

[125] Dena Markudova and Michela Meo. ReCoCo: Reinforcement learning-based Congestion control for Real-time applications. In *2023 IEEE 24th International Conference on High Performance Switching and Routing (HPSR)*, pages 68–74, Albuquerque, NM, USA, June 2023. IEEE. ISBN 978-1-66547-640-9. doi: 10.1109/HPSR57248.2023.10147986. URL https://ieeexplore.ieee.org/document/10147986/.