Learning-enabled Decision-making for Autonomous Driving: Framework and Methodology

Zhiyu Huang

School of Mechanical and Aerospace Engineering

A thesis submitted to the Nanyang Technological University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

2023

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

27/06/2023

Date

Zhiyu Huang

Zhiyu Huang

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accordance with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

27/06/2023

Date

Prof. Chen Lyu

Authorship Attribution Statement

This thesis contains materials from 4 papers published in peer-reviewed journals and 1 paper accepted to a conference in which I am listed as an author.

Chapter 2 is published as Z. Huang, J. Wu and C. Lv, "Efficient Deep Reinforcement Learning With Imitative Expert Priors for Autonomous Driving," in IEEE Transactions on Neural Networks and Learning Systems, doi: 10.1109/TNNLS.2022.3142822.

The contributions of the co-authors are as follows:

- I conceived the study, designed the research methodology, conducted the simulation experiments, and wrote the manuscript.
- Jingda Wu contributed to the interpretation of results and provided critical revisions to the manuscript.
- Prof. Chen Lyu supervised the research, obtained funding for the project, and contributed to editing the manuscript.

Chapter 3 is published as Z. Huang, J. Wu and C. Lv, "Driving Behavior Modeling Using Naturalistic Human Driving Data With Inverse Reinforcement Learning," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 8, pp. 10239-10251, doi: 10.1109/TITS.2021.3088935.

The contributions of the co-authors are as follows:

- I developed the initial concept, designed the experiments, and prepared drafts of the manuscript.
- Jingda Wu and I collaborated in carrying out the simulation experiments and analyzing the resulting data.
- Prof. Chen Lyu offered guidance for the project's direction and contributed to refining the manuscript drafts.

Chapter 4 is published as Z. Huang, H. Liu, and C. Lv, "GameFormer: Game-theoretic Modeling and Learning of Transformer-based Interactive Prediction and Planning for Autonomous Driving," in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2023.

The contributions of the co-authors are as follows:

- I was responsible for conceiving the initial concept, designing the experiments, and drafting the manuscript.

- Haochen Liu and I collaborated on the execution of experiments related to interaction prediction and open-loop planning, as well as the subsequent analysis of the obtained data.
- Prof. Chen Lyu provided valuable guidance in directing the project.

Chapter 5 is published as Z. Huang, H. Liu, J. Wu and C. Lv, "Conditional Predictive Behavior Planning With Inverse Reinforcement Learning for Human-Like Autonomous Driving," in IEEE Transactions on Intelligent Transportation Systems, doi: 10.1109/TITS.2023.3254579.

The contributions of the co-authors are as follows:

- I designed the study, conducted the experiments, analyzed the results, and drafted the initial manuscript.
- Haochen Liu and Jingda Wu provided technical assistance and expertise in the experimental setup and contributed to the interpretation of the results and their implications.
- Prof. Chen Lyu provided guidance on the overall research direction, supervised the project, and reviewed and approved the final manuscript.

Chapter 6 is published as Z. Huang, H. Liu, J. Wu and C. Lv, "Differentiable Integrated Motion Prediction and Planning With Learnable Cost Function for Autonomous Driving," in IEEE Transactions on Neural Networks and Learning Systems, doi: 10.1109/TNNLS.2023.3283542.

The contributions of the co-authors are as follows:

- I formulated the research problem, designed the study, developed the framework, conducted the experiments, and wrote the manuscript.
- Haochen Liu and Jingda Wu provided technical assistance in data collection, preprocessing, model evaluation, and result analysis, and contributed to the refinement of the manuscript.
- Prof. Chen Lyu supervised the project, offered guidance on the overall research direction, and provided feedback and revisions to the manuscript.

27/06/2023

Zhiyu Huang

Date

Zhiyu Huang

Acknowledgements

I would like to express my deepest gratitude to my advisor, Professor Chen Lyu, for his continuous guidance, support, and encouragement throughout my research journey. His invaluable insights and expertise have played a significant role in shaping this thesis, and his wisdom, patience, and passion for research have been instrumental in my development as a scholar. The AutoMan lab has become like a family to me, with the nurturing care, freedom, and support provided by my supervisor.

I am deeply grateful for the friendships and support of my exceptional peers and colleagues, who have made this journey both enjoyable and rewarding. Their camaraderie, shared experiences, and laughter have brightened even the most challenging moments. I am particularly thankful for the invaluable assistance from Jingda Wu, Xiaoyu Mo, Haochen Liu, Yanxin Zhou, Haohan Yang, and Wenhui Huang, who have been steadfast companions during my four-year journey. I appreciate their collaborative efforts, insightful discussions, and the supportive environment they helped create. It has been an immense privilege to work alongside these talented individuals.

Lastly, I dedicate this work to my loving family, who have provided me with a strong foundation and unwavering support. Their love and faith in me have been the pillars of my strength and determination. To my family, I love you.

Abstract

The growing adoption of autonomous vehicles (AVs) holds the promise of transforming transportation systems, enhancing traffic safety, and supporting environmental sustainability. Despite significant advancements in advanced perception systems for AVs, the effectiveness of these vehicles in real-world scenarios depends critically on advanced decision-making systems. Therefore, a critical challenge lies in developing safe, robust, and human-like decision-making systems for achieving higher levels of vehicle autonomy. Harnessing the power of artificial intelligence (AI) and machine learning (ML) is pivotal in developing such advanced decision-making systems. These techniques offer promising solutions for tackling key challenges in autonomous driving systems, including scalability, adaptability, and alignment with human driver behaviors.

This thesis presents a comprehensive framework and a series of learning-based methodologies for decision-making in AVs, with the objective of improving the scalability, adaptability, and alignment of their decision-making systems. The learning-enabled decision-making framework consists of three essential modules: world model, actor, and cost. These modules work collaboratively to provide a unified solution for autonomous machine intelligence and can be learned individually or jointly from human driving data. The methodologies proposed in this thesis focus on three key areas: (1) creating ML-driven actors and cost functions that align with human values, (2) developing ML-based world models capable of accurately forecasting complex human interactions, and (3) designing ML-based decision-making architectures that seamlessly integrate these components.

Initially, we explore each module individually. For the actor module, we propose an approach based on reinforcement learning (RL) that incorporates human prior knowledge into the training process. By combining imitation learning (IL) from human expert demonstrations with RL, a novel learning framework is developed to align agent behavior with human expectations. The results show improved sample efficiency, enabling the agent to navigate complex urban environments safely,

while also generating distinct driving styles that closely resemble human driving behaviors. In the cost module, we focus on learning personalized cost functions underlying human driving behaviors using inverse reinforcement learning (IRL). We propose a maximum entropy IRL algorithm that infers the reward function based on a structural assumption about discrete latent intentions guiding continuous low-level control actions. The personalized cost learning method outperforms general cost modeling methods, leading to a more human-like driving experience. Regarding the world model module, the goal is to accurately represent the complexity and dynamics of human behaviors in real-world driving environments. We introduce a joint multi-agent prediction model based on Transformers, leveraging hierarchical game theory to model interactions among agents in driving scenarios. The model employs a novel Transformer decoder structure and a learning process that regulates agent behavior to respond to other agents' behaviors. Extensive experiments validate the model's state-of-the-art prediction accuracy and its capability to jointly reason about ego agent motion plans and other agents' behaviors.

Following individual assessments, unified predictive decision-making architectures are presented, harmonizing the three modules. A comprehensive behavior planning framework that integrates all three core modules is proposed. It generates diverse trajectory proposals, forecasts multi-modal futures for other agents, and evaluates candidate plans using a learned cost function. Importantly, this framework includes a conditional motion predictor (CMP) that forecasts other agents' responses to the potential actions of the AV. The CMP improves prediction accuracy and facilitates the selection of human-like behaviors. Moreover, we propose a differentiable and integrated prediction-planning framework that enables end-to-end training of the entire decision-making system. The framework incorporates a differentiable nonlinear optimizer as the motion planner, allowing the gradients to flow between the three core modules and directly optimizing the planning performance. The experimental results demonstrate superior performance in both open-loop and closed-loop testing, effectively handling complex urban driving scenarios.

This thesis not only makes significant contributions to the advancement of autonomous driving systems but also lays the foundation for future research in learning-based decision-making for a wide range of intelligent systems. It paves the way for safe and human-centered autonomy, while driving the evolution of intelligent transportation systems.

Contents

List of Figures

List of Tables

List of Abbreviations

ADE	Average Displacement Error
AI	Artificial Intelligence
AV	Autonomous Vehicle
BC	Behavioral Cloning
CMP	Conditional Motion Prediction
CNN	Convolutional Neural Network
CIOC	Continuous Inverse Optimal Control
CQL	Conservative Q-Learning
CTRV	Constant Turn Rate and Velocity
DIM	Deep Imitative Model
DIPP	Differentiable Integrated Prediction and Planning
DRL	Deep Reinforcement Learning
EM	Expectation Maximization
FDE	Final Displacement Error
GAIL	Generative Adversarial Imitation Learning
GMM	Gaussian Mixture Model
GNN	Graph Neural Network
IDM	Intelligent Driver Model
IL	Imitation Learning
IRL	Inverse Reinforcement Learning
ITS	Intelligent Transporation System
KL	Kullback-Leibler
LSTM	Long Short-Term Memory
MAP	Mean Average Precision
MATS	Mixtures of Affine Time-varying Systems
MDP	Markov Decision Process
ML	Machine Learning

MLP	Multi-Layer Perceptron
MOBIL	Minimizing Overall Braking Induced by Lane changes
MPC	Model Predictive Control
MSBE	Mean Squared Bellman Error
MSE	Mean Squared Error
MTR	Motion Transformer
NGSIM	Next Generation Simulation
NLL	Negative Log-Likelihood
NN	Neural Network
OOD	Out-of-Distribution
PPO	Proximal Policy Optimization
RBF	Radial Basis Function
RELU	Rectified Linear Unit
RIP	Robust Imitative Planning
RL	Reinforcement Learning
SAC	Soft Actor-Critic
SL	Supervised Learning
V2V	Vehicle-to-Vehicle
V2I	Vehicle-to-Infrastructure
WOMD	Waymo Open Motion Dataset

Chapter 1

Introduction

1.1 Background

The rapid growth of urbanization, along with increasing concerns about environmental sustainability and traffic safety, has spurred a global drive towards the development of intelligent transportation systems (ITS). At the forefront of this revolution are autonomous vehicles (AVs), which hold the potential to profoundly transform the way people travel and goods are transported. The advancement of AVs has experienced remarkable progress in recent years [1], with successful deployments across various applications, including ride-hailing, freight transportation, and beyond [2]. Although AVs have demonstrated their competence in navigating urban environments, their level of autonomy is limited, and they persistently face challenges in adapting to complex scenarios involving human traffic participants. It is imperative to develop the capacity of AVs to navigate complex environments and interact with other road users with the same level of proficiency as an experienced human driver. Nonetheless, achieving high-level vehicle autonomy constitutes a significant challenge.

Machine learning (ML), a subfield of artificial intelligence (AI) focused on algorithms capable of learning from and evolving with data, offers a promising solution to this challenge. Recently, the integration of ML techniques into autonomous driving systems has substantially improved AV capabilities [3–5]. ML has emerged as a powerful tool for addressing challenges across the entire AV technology stack,

including perception and decision-making. ML-based perception systems, particularly deep learning, have considerably improved the ability of AVs to recognize and understand their environment [6–9]. Neural networks have been applied to tasks such as object detection, semantic segmentation, and scene understanding, enabling AVs to process sensor data from cameras, Lidar, and radar and extract relevant information for decision-making and control. However, the decision-making process in AVs constitutes an even more critical component, as it directly determines the vehicle's actions in response to the perceived environment and makes consequences to the surrounding environment. Traditional rule-based and model-based approaches for decision-making have limitations in handling the complexities of real-world driving scenarios, particularly when dealing with uncertainties and the interactions between multiple agents. Machine learning offers promising solutions to address these challenges and enhance AV decision-making. By learning from human driving data, ML-based decision-making systems can capture the nuances of human driving behavior, adapt to different situations, and make more informed decisions.

There are several key challenges associated with the development of decision-making systems for autonomous driving:

- **Robustness (Safety)**: This is an imperative, fundamental criterion for all autonomous systems. Decision-making systems for AVs must be robust to uncertainties, including perception errors (e.g., sensor noise and occlusions) and uncertain agent behaviors. The system should be able to handle these uncertainties and make safe decisions even in challenging conditions.
- **Scalability**: Real-world driving scenarios manifest considerable complexity, involving a diverse array of situations (e.g., urban intersections or highways) with distinct traffic rules, road geometries, and agent behaviors. Decision-making systems must scale effectively to handle all kinds of scenarios in the real world and provide reliable performance.
- **Adaptability (Interactivity)**: Driving conditions can vary significantly, with numerous interacting agents (vehicles, pedestrians, and cyclists) and dynamic environments. Decision-making systems should be able to adapt to these variations and interactions and continuously enhance their performance through learning from a wide range of driving situations.

- **Alignment (Human-likeness)**: To ensure the safe and efficient integration of AVs into existing transportation systems, decision-making systems must align with human driver behaviors. This necessitates understanding the actions of human drivers and making decisions that are consistent with human expectations or values.

In this thesis, we endeavor to tackle these challenges in AV decision-making systems using ML-based methodologies. In particular, the primary focus of this thesis is to address the scalability issue for AV decision-making by employing and improving learning-based methods, as well as interactivity and alignment issues by learning from human driving data. We employ a general cognitive framework [10] designed for constructing autonomous intelligent agents and adapt this framework to facilitate the development of decision-making systems in AVs. The methodologies and solutions to these challenges will be systematically presented and discussed throughout the course of this thesis.

1.2 Motivation

The motivation behind this thesis lies in tackling the aforementioned challenges and contributing to the evolution of decision-making systems for autonomous driving, with the objective of advancing the state-of-the-art in AV decision-making and enhancing the overall performance of decision-making systems.

First and foremost, it is imperative to answer a fundamental question: **why should we utilize ML in the development of decision-making systems for AVs?** The primary advantage of ML, *scalability*, sets it apart from traditional rule-based and model-based approaches. Traditional methods have demonstrated limitations in coping with the complexities inherent in real-world driving scenarios. For example, a game theoretic decision-making model may work well in specifically designed scenarios and when interacting with a single agent [11], whereas a real-world driving scene may involve numerous interacting agents. Using a finite-state machine to determine the decisions of AVs may prove successful in scenarios with simple road structures and limited interactions [12], but they fail in more complex situations, and the rules themselves become increasingly difficult to design and maintain.

The behaviors of AVs derived from these methods often contradict human expectations or social norms, potentially leading to unpredictable and even hazardous actions within traffic flows involving human participants. In short, the most significant issue with conventional methods is their inadequate *scalability*, which is of paramount importance in autonomous driving. In contrast, ML presents a more scalable, flexible, and adaptive solution that can leverage vast amounts of human driving data, thereby enabling the system to learn and generalize from diverse experiences. Data-driven approaches empower the decision-making system to effectively capture the nuances of human driving behavior, predict future human behaviors, and make more informed and context-aware decisions. Consequently, ML-based methodologies have the potential to significantly enhance the performance and scalability of autonomous vehicles, justifying their application in the development of decision-making systems.

The second question to consider is: **what to learn in the AV's decision-making system?** To answer this question, it is crucial to identify the key elements and principles that govern human (or intelligent agent) decision-making. To this end, we adopt an established framework proposed by Yann LeCun [10] for autonomous machine intelligence. In this framework, there are three fundamental modules for decision-making: *world model*, *actor*, and *cost*, in addition to the *perception* module, which is also an essential capability of intelligent agents but will not be the focus of this thesis. The *world model* encodes essential knowledge about the environment and predicts plausible future states of the world, enabling the system to reason about the actions of other agents. This component is the most challenging and complex piece of the framework, which requires learning-based methods to ensure its accuracy and scalability. The *actor* module is responsible for generating and selecting actions or trajectories that optimize the estimated cost. The actor can be either reactive, directly producing an action from the state produced by the perception, or predictive, requiring the world model to predict and compute the costs of actions. The *cost* module, on the other hand, is responsible for quantifying the desirability of potential actions or trajectories based on various factors. By learning the cost that aligns with human values and preferences, the decision-making system can ensure that its actions are consistent with human expectations. These modules can work together to produce intelligent behaviors, thereby laying a solid foundation for the development of intelligent decision-making systems in AVs.

This thesis will focus on learning the three modules for decision-making within the context of autonomous driving, and the framework will be detailed in Section 1.3.

The final question to address is: **how to design and learn these modules?** To address this, a variety of ML techniques, including supervised learning (SL), reinforcement learning (RL), imitation learning (IL), and inverse reinforcement learning (IRL), can be employed. These methodologies can be applied to different modules (world model, actor, and cost) in the context of AV decision-making systems. SL is a widely used approach, which is particularly effective when a large amount of data is available, allowing the decision-making system to scale efficiently across different scenarios. Therefore, we implement state-of-the-art deep learning models to develop the *world model* that can accurately predict the interaction dynamics between agents and their future trajectories, and strive to integrate them into the decision-making process. For the *actor* module, we will explore the combination of IL and RL and propose novel solutions to leverage their strengths while mitigating their limitations, enabling the AV to learn more efficiently and achieve better performance. For the *cost* module, we employ IRL to learn the cost from human driving data, attempting to align the decision-making processes with human preferences or values. In this thesis, all proposed decision-making frameworks adhere to these three modules, and some of them may be substituted with simpler components to focus on individual modules. Furthermore, we will present a framework that can jointly learn all these modules for enhanced performance.

1.3 Framework

Fig. 1.3 illustrates the architecture for autonomous intelligent agents as proposed in [10], along with its adaptation to the domain of autonomous driving systems. In this context, we rephrase the functions of the core modules, which are **perception, world model, cost, and actor**, to address the requirements of autonomous driving systems specifically. For a comprehensive understanding of the modules' original functions pertaining to general intelligent agents, please refer to [10].

The **perception** module receives signals from sensors and estimates the current state of the world. For AVs, this module is responsible for processing raw sensory input, such as camera images, lidar point clouds, and radar signals, to extract

meaningful semantic information about the vehicle's surroundings. The scope encompasses the estimation of poses for other objects (e.g., vehicles, pedestrians, static obstacles), detection of traffic signs and signals, and identification of road structures. Deep neural networks have been extensively employed in the perception module to accurately detect and classify objects [6] and even generate high-definition maps (detailed road network) [13] to support downstream decision-making modules.

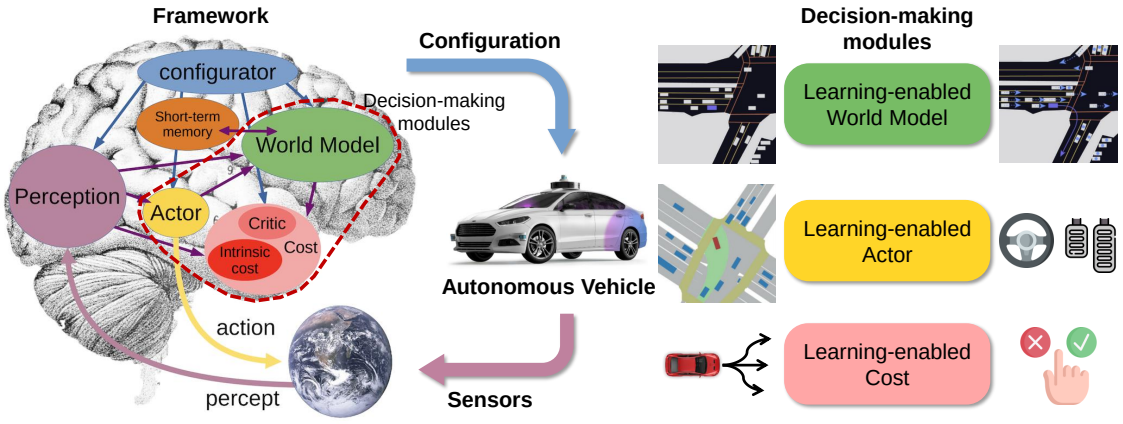


FIGURE 1.1: A system architecture for autonomous intelligence and its adaption to the autonomous driving system

The **world model** module predicts possible future states of the world, taking as input the current state of the world from the perception module. It may either predict the natural evolution of the world or future world states resulting from actions proposed by the actor module. Within the context of AVs, the world model can be considered as a prediction model employed to forecast the future states of surrounding agents. Deep learning-based models have increasingly been adopted for trajectory or motion prediction tasks with considerable success, achieving considerable success and leading to significant enhancements in prediction accuracy and scalability across a myriad of scenarios [14–16]. While most motion prediction models forecast the future trajectories of agents in a "natural" way, wherein the agents are not influenced by the AV, [17] proposes a conditional motion prediction model that anticipates other agents' responses to the AV's potential actions, aligning more closely with the concept of world models. In addition to trajectories, some learning-based prediction models can estimate the future occupancy grid of the driving scene [18], which can effectively handle driving scenarios with arbitrary elements.

The **cost** module measures the agent's level of "discomfort" as a scalar value, and the overall objective of the agent is to minimize the cost over the long run. Within the context of AVs, the cost module quantifies the desirability of potential actions by evaluating their implications on safety, efficiency, adherence to traffic regulations, and conformity to social norms. The input for this module is either the current world state, as generated by the perception module, or possible future states predicted by the world model. By considering multiple objectives, such as minimizing travel time, maintaining a safe distance, and complying with speed limits, the cost module enables the autonomous driving system to make well-balanced decisions. However, the challenge lies in the often latent and difficult-to-quantify nature of these objectives, particularly with regard to social interactions between agents. Furthermore, even when the objectives are quantifiable, achieving a balance that aligns with human values is challenging, especially given that individuals assign different levels of importance to these objectives. Consequently, learning-based methods should be employed to assess the cost or balance of these sub-objectives, in order to align the actions of AVs with human expectations and values [19].

The **actor** module generates actions and conveys them to the controller. The actor may have two paradigms: (1) a policy that directly generates an action from the world state estimate produced by the perception and retrieved from the short-term memory, or (2) an action optimizer, which computes an optimal action sequence by utilizing the world model to predict future world state sequences, potentially based on the proposed action sequence, and using the cost module to estimate the costs of actions, ultimately outputting the initial action (or a short sequence of actions). Within the context of AVs, both paradigms are prevalent in research. The first paradigm is often referred to as model-free policy, where learning-based methods are predominantly employed. This paradigm circumvents the need for building complex world models or even the cost module by directly learning a policy from expert demonstrations (imitation learning) [20] or through self-exploration and interactions with the environment (reinforcement learning) [21]. The second paradigm, akin to model predictive control (MPC), is considered more robust and generalizable. However, the challenging aspect is constructing performant world model and cost modules to enable optimal performance, which necessitates greater effort on these two modules.

In this thesis, we assume that the perception module is perfect and provides an

accurate estimation of the world state, including the states of surrounding objects, maps, and traffic signals, without any errors. In the original framework, all the modules are assumed to be differentiable, allowing gradients to flow between these modules for joint learning of the framework. In our proposed methodologies, we do not maintain this assumption and instead examine each module in the framework separately to achieve better interpretability, modularity, and transparency within the autonomous driving system. However, we will also present integrated and differentiable frameworks capable of jointly optimizing all the modules within the framework.

1.4 Major Contributions

The primary objective of this thesis is to address three critical challenges that currently exist in the area of decision-making for AVs. Firstly, the actor and cost modules often fail to align with human values and expectations. This is primarily due to the fact that the cost/reward functions are typically hand-engineered, which leads to misalignment and scalability issues. Secondly, the world model employed for decision-making is not sophisticated enough to capture the complex interaction dynamics between humans, impairing the final decision-making process. Lastly, the modules in the decision-making framework are often separately learned or designed, which could result in suboptimal performance. To overcome these challenges, this thesis presents significant contributions within the general decision-making framework. Fig. 1.2 illustrates the function of each module in the framework with respect to the different proposed methodologies. The primary contributions of each module in the proposed methodologies are highlighted.

Contribution 1: An improved reinforcement learning-based actor incorporating human prior knowledge.

In this research, we employ the first paradigm of actor, which avoids the necessity of explicitly building a world model. Despite its efficiency and lack of cost module requirements, IL often exhibits suboptimal performance during deployment. As a result, we apply RL to train the actor module, which is parameterized by a deep neural network, to directly generate driving decisions. However, there are two fundamental problems with RL: low data efficiency and difficulty in specifying a

	Actor	Cost	World Model
Chapter 2	***RL Policy***	Divergence from Human Policy	——
Chapter 3	Trajectory Sampler	***IRL Scoring***	Simplistic Rule-based Model
Chapter 4	——	——	***NN Model***
Chapter 5	Trajectory Sampler	IRL Scoring	***NN Conditional Model***
Chapter 6	***Differentiable Optimizer***	Learnable Cost Weights	NN Model

FIGURE 1.2: Illustration of the decision-making modules in proposed methodologies and the outline of the thesis

cost function that aligns with human behavior. To address these issues, we propose a novel method incorporating human prior knowledge in the training process of RL actors. Specifically, we combine RL and IL methods to adjust the actor's decisions in accordance with human expectations. Our proposed approach not only enables the actor to safely and efficiently navigate complex urban driving environments (in simulation) that might challenge a human driver, but also customizes driving styles (e.g., conservative or aggressive) based on particular human demonstrations. A comprehensive description of the framework is presented in Chapter 2.

Contribution 2: An inverse reinforcement learning-based cost for modeling human behaviors.

In this study, we focus on the cost module and the application of IRL to learn personalized cost functions of driving behaviors using large-scale human-driving datasets. Within the decision-making framework, the actor is a sampling-based trajectory planner, and the world model employed to simulate the outcomes of action proposals is simplified. In the cost module, we implement a maximum entropy IRL algorithm that captures the uncertainty inherent in human decision-making. The objective is to model the internal cost function evaluations of various human drivers when making decisions, aiming to deliver a personalized driving experience. Our proposed method demonstrates that the decision-making system,

enabled by our learned cost module, is capable of making driving decisions that align with the values of individual drivers. A thorough exposition of the framework is provided in Chapter 3.

Contribution 3: A deep learning-based prediction model for joint multi-agent interaction modeling.

A core aspect of this thesis involves the development of world models that can accurately represent the complexity and dynamics of human behaviors in real-world driving environments. We propose a joint multi-agent trajectory prediction model utilizing Transformers, an advanced neural network architecture, to model interactions among agents in driving scenarios. A crucial aspect of our proposed method is the fusion of the concept of hierarchical game theory (level-k). More specifically, we employ the Transformer encoder-decoder structure, in which the encoding stage is responsible for extracting information from agents and map elements, while the decoding stage involves a series of Transformer decoders designed to iteratively refine the interaction prediction process. The model is trained on extensive real-world traffic data to ensure the accuracy of predicted human behaviors and their alignment with practical applications. Our proposed prediction model has achieved state-of-the-art performance on the WOMD interaction prediction benchmark and demonstrates superior performance when used for motion planning in both open-loop and closed-loop settings. A comprehensive description of the model is provided in Chapter 4.

Contribution 4: A behavior planning framework utilizing conditional prediction model and cost learning.

In behavior planning or decision-making tasks, the majority of research efforts employ a prediction model to first anticipate other agents' future trajectories and then act accordingly. In contrast, we explore alternative approaches to utilizing world models for improved and more proactive decision-making, shifting from preemptively predicting other agents' behaviors to forecasting their responses to the AV's potential decisions. Within the decision-making framework, the actor is a trajectory sampler that can generate a diverse set of behavior proposals, while the world model is based on deep learning models (Transformers). We propose a novel conditional prediction model that takes the AV's potential actions as input and outputs other agents' possible reactions. In addition, we pair the conditional prediction

model with an IRL-based cost module to more accurately evaluate the costs of behavior proposals in alignment with human behaviors. Our method outperforms the conventional non-conditional prediction model in terms of both prediction accuracy and planning performance, which indicates that the conditional prediction model can better reflect the interaction dynamics in the real world. An in-depth explanation of the framework can be found in Chapter 5.

Contribution 5: A learnable and integrated framework for interactive prediction and planning.

We investigate methodologies that make all decision-making modules within the framework differentiable and jointly learnable, allowing the entire framework to be optimized end-to-end with the direct objective of enhancing the final planning performance. Specifically, within the decision-making framework, we design the actor as a differentiable trajectory optimizer that takes as input the prediction results from the world model and the weights of the cost function to explicitly plan a trajectory for the AV. Since the trajectory planner is differentiable, the error of the final planning trajectory can be backpropagated to the upstream modules (i.e., prediction model and cost). As a result, we enable the entire framework to be modular yet end-to-end trainable using human driving data by imitating the interaction process between humans. We demonstrate that our proposed framework outperforms other actor learning methods or separate prediction-planning methods, because it is capable of generating planning-centric or planning-aware prediction results that enable the planner to deliver human-like driving trajectories. A comprehensive description of the framework is given in Chapter 6.

1.5 Thesis Outline

This thesis is structured into seven chapters, including the introduction and conclusions. The outline of the chapters is as follows:

Chapter 1 provides an overview of the research background, the motivation for the study, the universal decision-making framework, and the main contributions of the thesis. It presents the challenges of AV decision-making and highlights the significance of developing learning-based methodologies for improved and more human-like driving behavior.

Chapter 2 introduces an efficient RL-based actor that incorporates human prior knowledge. We discuss the combination of IL and RL methods for training the actor module and showcase the effectiveness of our approach in generating customized driving styles in complex urban environments.

Chapter 3 focuses on the development and implementation of an IRL-based cost module for modeling and aligning human driving behavior. The maximum entropy IRL algorithm, which captures uncertainty in human decision-making, is introduced. We demonstrate the personalized driving experience provided by the decision-making system, enabled by the learned cost module.

Chapter 4 proposes a joint multi-agent prediction model that utilizes Transformers to model interactions among agents in driving scenarios. We explore the fusion of hierarchical game theory with the Transformer encoder-decoder structure and demonstrate the state-of-the-art performance of our prediction model in interaction prediction, as well as open-loop and closed-loop planning settings.

Chapter 5 explores the development of a behavior planning framework that utilizes a conditional prediction model and IRL cost learning. We discuss the advantages of forecasting agents' responses to the AV's potential decisions and demonstrate the superior performance of our proposed method compared to conventional non-conditional prediction models.

Chapter 6 investigates establishing a differentiable and integrated framework for interactive prediction and planning. We discuss the design of a differentiable trajectory optimizer, its integration into the planning module, and the end-to-end training of the entire framework using human driving data. The effectiveness of our proposed framework in planning human-like driving trajectories, owing to planning-aware predictions provided by our method, is also demonstrated.

Chapter 7 summarizes the main contributions and findings of the thesis. The implications of our research are discussed, and potential avenues for future work in the realm of human-centered AV development and related fields are proposed.

Chapter 2

Learning Actor with Human Prior Knowledge

Learning the actor represents the most direct approach to decision-making, circumventing the complexities associated with learning cost functions and intricate world models. Deep reinforcement learning (DRL) provides a promising avenue for accomplishing this objective. Nevertheless, its limited sample efficiency and the challenges in formulating reward functions for DRL impede its practical applications. In this chapter, we introduce a novel framework aimed at incorporating human prior knowledge into DRL to enhance sample efficiency and reduce the intricacies of designing sophisticated reward functions. This framework consists of three core components: expert demonstration, expert policy derivation, and reinforcement learning. During the expert demonstration phase, a human expert performs the task, and their behaviors are captured as state-action pairs. Subsequently, in the policy derivation step, the imitative expert policy is derived using behavioral cloning and uncertainty estimation. In the reinforcement learning stage, the imitative expert policy guides the DRL agent's learning by regularizing the Kullback-Leibler (KL) divergence between the RL agent's policy and the expert policy.

To evaluate the proposed method in the context of autonomous driving applications, we have designed two simulated urban driving scenarios: unprotected left turn and roundabout navigation. The efficacy of our approach is demonstrated by

the training results, which exhibit superior performance and a significant improvement in sample efficiency compared to baseline algorithms. In testing conditions, the agent trained with our method achieves the highest success rate and displays diverse, human-like driving behaviors. We observe that utilizing an imitative expert policy trained with an ensemble method, which estimates both policy and model uncertainties, as well as augmenting the training sample size, results in improved training and testing performance, particularly for more complex tasks. As a consequence, the proposed method showcases its potential for advancing the practical implementation of DRL-enabled autonomous driving systems.

2.1 Introduction

Reinforcement learning (RL) has seen successful applications in various domains including autonomous driving [22] and intelligent transportation systems [23]. Utilizing an RL-based decision-making system can deliver a human-like driving experience, characterized by interaction awareness and personalization, by learning to interact with other agents on the road without explicitly modeling the complex driving environment. Nevertheless, RL faces two significant challenges. The first major issue is the notoriously low data efficiency, which means an RL agent requires a massive amount of interactions with the environment to learn a functional policy. This problem becomes exacerbated when addressing increasingly demanding and diverse autonomous driving tasks, such as navigating unsignalized intersections or executing unprotected left turns in dense traffic. Another critical challenge lies in designing the reward function, which specifies the desired behaviors for the RL agent. A poorly designed reward function may cause the agent to mistakenly exploit the reward function and stick to unexpected behaviors, and thus it often takes a significant amount of time and effort to design an appropriate reward function and fine-tune parameters. Although the reward functions can be learned from human driving data using techniques such as inverse reinforcement learning [24, 25], assumptions regarding the structure of the reward function (e.g., a linear combination of different hand-crafted features) might not be valid in practice.

Facing these two major challenges, we propose a novel RL framework that can transfer human prior knowledge to the RL agent with a small amount of human expert demonstration, aiming to improve sample efficiency and mitigate the need for

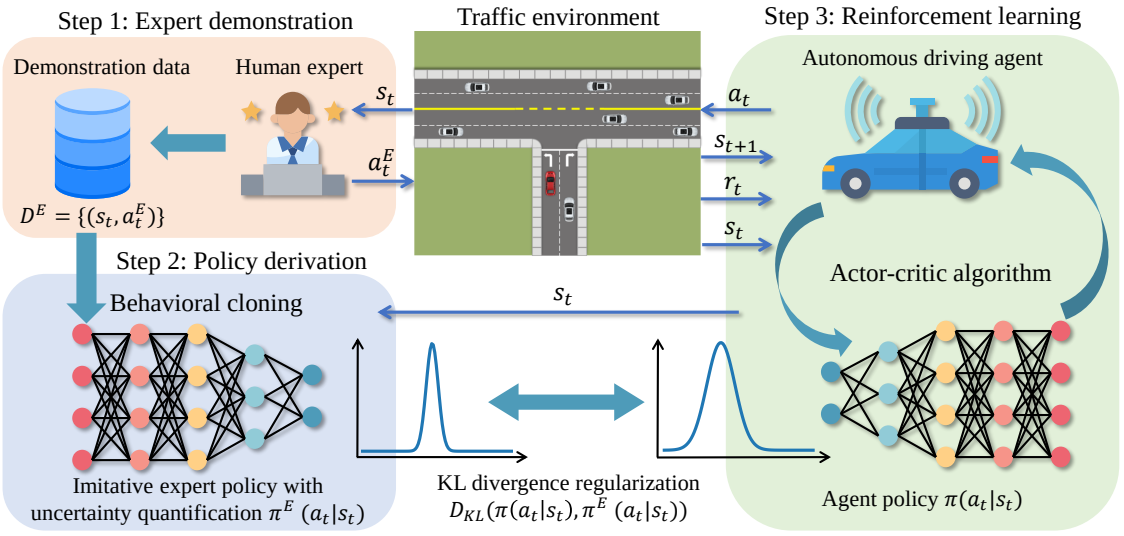


FIGURE 2.1: The framework of RL-enabled decision-making with imitative expert priors

designing sophisticated reward functions. First of all, we distill human prior knowledge from demonstrations into an imitative expert policy using imitation learning and uncertainty estimation. Subsequently, the imitative expert policy guides the RL agent's learning process by regularizing the RL policy to approximate the expert policy. The RL agent's exploration ability is preserved when the uncertainty of the imitative expert policy is high. We also demonstrate that human-like driving behaviors can be achieved with only a sparse reward function. In detail, as shown in Fig. 2.1, we first derive a stochastic expert policy using imitation learning from expert demonstrations. By incorporating the Kullback–Leibler (KL) divergence between the imitative expert priors and agent policy into the RL framework, we can regularize the agent's behaviors within the desired space, thereby significantly enhancing learning efficiency. The main contributions of this chapter are as follows:

- An RL framework that incorporates imitative expert priors from expert demonstrations is proposed. The imitative expert priors are learned using imitation learning and an uncertainty quantification method, which can regularize the RL agent's behavior while also maintaining its exploration ability.
- Two different approaches, value penalty and policy constraint, are proposed to integrate the imitative expert policy within the actor-critic algorithm framework. The superior sample efficiency of our proposed method is revealed in comparison with other RL algorithms.

- The proposed method is rigorously validated in challenging urban driving scenarios, demonstrating that human-like driving behaviors emerge even when the agent receives sparse reward feedback from the environment. The effects of expert policies trained with different sample sizes and different uncertainty estimation methods are also examined.

2.2 Literature Review

2.2.1 Imitation learning and reinforcement learning

Imitation learning (IL) and reinforcement learning (RL) are two promising techniques that can be applied in autonomous driving, which can help develop adaptive, flexible, and human-like decision-making systems. Imitation learning, which leverages an efficient supervised learning mechanism, has seen widespread use in autonomous driving research due to its simplicity and effectiveness. For example, imitation learning has been used in end-to-end autonomous driving systems that directly output control signals from raw sensor inputs [20, 26] or modular driving systems that output planned trajectories from processed perception information [27]. However, one intractable limitation of imitation learning is the distribution shift, which means compounding errors lead to system failure due to the mismatch of training and inference distributions. Conversely, RL does not encounter this problem because it relies on online interactions with the environment. Numerous studies have utilized RL to tackle complex tasks in autonomous driving, such as urban driving [28], multi-agent interactions [29], and handling near-accident scenarios [30]. However, RL's drawbacks are apparent and difficult to resolve, potentially impeding its practical applications. These drawbacks include low training efficiency, even in simulated environments, and the challenges associated with specifying a reward function and time-consuming parameter tuning.

2.2.2 Learning with human prior knowledge

To improve the sample efficiency of RL training or inject desired properties (e.g., safety, smoothness, and risk-aversion) into RL-based policies, many studies propose incorporating human prior knowledge into the RL framework. Human prior

knowledge may include the understanding of the task, guidance, or demonstrations provided to the RL agent, and it has been shown that combining human understanding with RL methods can help solve autonomous driving tasks safely and efficiently. For example, [31, 32] proposed adding a safety check module encoding the expert-defined safety rules to the RL-based control system, guiding the exploration process and excluding dangerous actions to prevent unsafe exploration and expedite training. [33] employed learning useful skills first (driving-in-lane, left lane change, and right lane change), followed by a master policy to switch between different skills, to hierarchically and efficiently solve driving tasks. In addition, utilizing human guidance or demonstration helps constrain the solution space and direct the RL agent's exploration, to eliminate unwanted interactions and accelerate the learning process. For instance, [34] proposed a real-time human-guidance-based learning method that allows human experts to intervene in the training process in real-time and provide guidance, enabling the RL agent to learn from both human guidance and self-exploration. However, such methods are heavily dependent on online human guidance, which may impose a heavy workload on human experts. On the other hand, offline human demonstrations are easier to obtain and do not require interactions between the RL agent and the human expert, making it more advantageous to incorporate human expert demonstrations into the RL framework to enhance performance.

2.2.3 Reinforcement learning with demonstrations

To make use of human demonstrations in RL, one approach is to pre-train the policy using imitation (supervised) learning with expert demonstrations to initialize the policy at a reasonable performance level, and then apply RL to achieve a better policy with respect to the reward function [35, 36]. However, this method often does not perform well, especially for the maximum entropy RL [37], which encourages initial policy randomness to facilitate exploration. An alternate approach involves adding expert demonstrations to the experience replay buffer for off-policy RL algorithms and sampling the experience from both expert demonstrations and the agent's interactions to update the policy [38, 39]. However, this method requires annotating rewards for each state-action pair to comply with the format of transition tuples in RL, which can be intractable in some cases due to the inaccessibility of the expert rewards. [34, 40] proposed incorporating imitation

learning (minimizing the discrepancy between agent actions and expert actions) into policy updates in addition to reinforcement learning (maximizing Q values), allowing the agent to learn from both human demonstrations and self-exploration. However, a sophisticated design of reward function or reward shaping is still indispensable for these methods. In contrast, our proposed method is inspired by recent advances in offline RL [41, 42], which encourages the learned policy to be close to the behavior policy. This idea was originally introduced to avoid value overestimation for actions outside of the training distribution, since the agent learns purely offline without online data collections. We propose to employ this method in an online RL setting where the behavior policy is learned from expert demonstrations and then used to guide exploration and constrain the RL agent's behaviors. To maintain the RL agent's exploration capability, we propose using uncertainty estimation in deriving the expert policy with imitation learning. As a result, the agent can explore more when the uncertainty of the expert policy is high, indicating that the action provided by the imitative expert policy is not reliable. Furthermore, the reward function design can be significantly simplified since the agent's behavior is regularized by the expert policy, thus allowing a sparse reward to work effectively even in complicated situations.

2.3 Preliminaries

2.3.1 Reinforcement learning

Reinforcement learning (RL) addresses the problem of learning to control a dynamic system, which can be defined by a Markov decision process (MDP) denoted as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, \gamma)$. In this tuple, $\mathcal{S}$ represents the state space $\mathbf{s} \in \mathcal{S}$, $\mathcal{A}$ denotes the action space $\mathbf{a} \in \mathcal{A}$, T is the transition probability distribution that defines the system dynamics $T(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, r is the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and $\gamma \in (0, 1]$ is a discount factor. The objective of RL is to find a policy, defined as a distribution over actions conditioned on states $\pi(\mathbf{a}_t|\mathbf{s}_t)$, in order to maximize the long-term discounted cumulative reward:

$$\max_{\pi} \mathop{\mathbb{E}}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right], \tag{2.1}$$

where τ represents a trajectory, $p_\pi(\tau)$ is the distribution of the trajectory under policy π, and T is the time horizon.

One approach to optimize the objective is to estimate the state or state-action value function and then recover a policy. The state value function (V-function) and state-action value function (Q-function) are recursively defined as:

$$V^\pi(\mathbf{s}_t) = \mathop{\mathbb{E}}_{\mathbf{a}_t \sim \pi(\cdot\,|\mathbf{s}_t)} \left[Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \right], \tag{2.2}$$

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim T(\cdot\,|\mathbf{s}_t, \mathbf{a}_t)} \left[V^\pi(\mathbf{s}_{t+1}) \right]. \tag{2.3}$$

The optimal policy π^* can be obtained by maximizing $V^{\pi^*}(\mathbf{s})$ at all states $\mathbf{s} \in \mathcal{S}$. In the actor-critic method, a policy π is optimized by alternately learning the V-function and Q-function through minimizing the Bellman errors over one-step transitions and learning the policy by maximizing the Q-values.

In addition to maximizing the cumulative discounted reward, the soft actor-critic (SAC) [37, 43] method employs a stochastic policy and adds an entropy term of the policy $\mathcal{H}$ to the objective to facilitate exploration:

$$\max_\pi \mathop{\mathbb{E}}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{T} \gamma^t \left(r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}\left(\pi(\cdot\,|\mathbf{s}_t) \right) \right) \right]. \tag{2.4}$$

The entropy term is equivalent to the KL divergence between the agent's policy and a uniformly random action prior. However, in our method, we regularize the learned policy π towards the non-uniform and informative expert behavior policy π^E. Furthermore, we propose an alternative approach to directly constrain the discrepancy between the learned policy π and the expert behavior policy π^E. Some methods for obtaining the expert policy π^E from demonstration data are described below.

2.3.2 Behavioral cloning

Since the expert policy is not directly accessible, one feasible and widely-used approach is to approximate the expert policy through imitation learning from expert demonstrations. Formally, given the expert demonstration dataset $\mathcal{D}^E : \{\tau_i\}_N$,

which consists of N trajectories, and each trajectory is a sequence of state-action pairs $\tau = \{\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T\}$, the imitative expert policy $\pi^E : \mathcal{S} \rightarrow \mathcal{A}$ can be obtained by maximizing the log-likelihood over $\mathcal{D}^E$:

$$\pi^E := \underset{\pi}{\operatorname{argmax}} \underset{(\mathbf{a},\mathbf{s})\sim\mathcal{D}^E}{\mathbb{E}} \left[\log \pi(\mathbf{a}|\mathbf{s})\right]. \tag{2.5}$$

In a general behavioral cloning setup, the expert policy is a neural network parameterized by θ, which takes a vector of state inputs and generates a single deterministic output representing the action, $\mathbf{a}_t = \pi_\theta(\mathbf{s}_t)$. Typically, the policy network is trained by minimizing the mean squared error (MSE) between the network outputs and expert actions, as shown:

$$\mathcal{L}(\theta) = \underset{(\mathbf{a},\mathbf{s})\sim\mathcal{D}^E}{\mathbb{E}} \parallel \pi_\theta(\mathbf{s}) - \mathbf{a} \parallel^2. \tag{2.6}$$

However, this method can only yield a deterministic policy that outputs a point estimation of actions. This is insufficient as a distribution of actions is needed to regularize the action distribution of the stochastic RL policy.

2.3.3 Policy uncertainty

In order to obtain a stochastic expert policy, we first consider the uncertainty of human (expert) behaviors, which implies that one can generate different feasible actions in the same state. We term this kind of uncertainty as policy uncertainty, which originates from the potential intrinsic randomness of human experts generating actions. To account for this uncertainty, we opt for outputting the parameters of a parametric probability distribution over actions instead of deterministic actions. We assume that the action is subject to a Gaussian distribution, i.e., $\mathbf{a}_t \sim \mathcal{N}(\mu_\theta(\mathbf{s}_t), \sigma^2_\theta(\mathbf{s}_t))$. Subsequently, we employ the maximum likelihood approach to train the imitative expert policy, which is equivalent to minimizing the negative log-likelihood (NLL) of the Gaussian distribution [44]:

$$\mathcal{L}(\theta) = \underset{(\mathbf{a},\mathbf{s})\sim\mathcal{D}^E}{\mathbb{E}} \left[\frac{\log \hat{\sigma}^2_\theta(\mathbf{s})}{2} + \frac{(\mathbf{a} - \hat{\mu}_\theta(\mathbf{s}))^2}{2\hat{\sigma}^2_\theta(\mathbf{s})} + c\right], \tag{2.7}$$

where θ is the parameters of the policy network, $\hat{\mu}_\theta$ and $\hat{\sigma}^2_\theta$ are the predictive mean and variance respectively, and c is a constant.

2.3.4 Model uncertainty

Although we have derived a stochastic expert policy, the predictive mean and the variance of the policy are still uncertain and unreliable for data not included in the training dataset. We term this kind of uncertainty as model uncertainty, resulting from a lack of training data in certain areas of the state space. Model uncertainty quantifies whether the model is confident about its action outputs. Estimating model uncertainty is crucial for the imitative expert policy in our proposed method because the RL agent can often encounter states not included in the demonstration dataset, thus requiring confidence quantification on the out-of-distribution states. Some approaches have been put forward to estimate this kind of uncertainty, such as the Monte Carlo dropout method [45, 46] and deep ensemble method [44, 47]. In this chapter, we use the deep ensemble method because it is more computationally efficient and simple to implement. We adopt an ensemble of M networks (stochastic policies) trained with different random initializations and orders of data using Eq. (2.7). We take all networks (θ_i referring to the parameters of i-th network) and combine their results into a Gaussian mixture distribution. The mixture mean and variance can be calculated as:

$$\mu_{\pi^E}(\mathbf{s}) = \frac{1}{M}\sum_{i=1}^{M}\hat{\mu}_{\theta_i}(\mathbf{s}), \tag{2.8}$$

$$\sigma^2_{\pi^E}(\mathbf{s}) = \frac{1}{M}\sum_{i=1}^{M}\hat{\sigma}^2_{\theta_i}(\mathbf{s}) + \left[\frac{1}{M}\sum_{i=1}^{M}\hat{\mu}^2_{\theta_i}(\mathbf{s}) - \mu^2_{\pi^E}(\mathbf{s})\right]. \tag{2.9}$$

We can treat the ensemble model as a Gaussian expert policy with mean $\mu_{\pi^E}(\mathbf{s})$ and variance $\sigma^2_{\pi^E}(\mathbf{s})$ that can capture both the policy uncertainty and model uncertainty.

2.4 Methodology

2.4.1 Framework

Fig. 2.1 illustrates the conceptual framework of our proposed method, which comprises three key steps: expert demonstration, policy derivation, and reinforcement learning. Initially, a human expert demonstrates the execution of a task, and their behaviors are encoded as sequences of state-action pairs. Next, the imitative expert policy is derived from the demonstration data using methods from Section 2.3. The imitative expert policy estimates the action distribution a human expert would follow in a given state, and it can be queried with a state encountered by the RL agent during training to respond with the reference action distribution. Ultimately, the imitative expert policy guides the RL agent's learning process, either by adding a discrepancy penalty to the value function or by regularizing the distributional discrepancy between the agent's policy and the expert's policy.

In the policy derivation step, we employ the stochastic policy setting and propose to estimate both the policy uncertainty and model uncertainty of the expert policy, which are necessary and important. The rationale is that when the uncertainty (variance) of the expert policy is small, indicating the policy is confident about its outputs, the RL agent's action should be close to the expert policy to avoid unnecessary exploration. In contrast, if the uncertainty (variance) is large, the mean value of the expert policy may not be sensible, but it can admit a high variance, allowing the resulting distribution to cover reasonable actions. In this situation, the RL agent's action should also be close to the expert policy to explore more and find better actions.

2.4.2 Actor-critic algorithm

For the agent learning step, we propose a modified actor-critic RL algorithm that can regularize the agent policy towards the imitative expert policy, and we present two methods to achieve this. The algorithm concurrently learns four networks: two Q-function networks (Q_{ϕ_1} and Q_{ϕ_2}), a value function network (V_ψ), and a stochastic policy network (π_θ). We separately maintain a target value network (V_{targ}) obtained by polyak averaging the V_ψ network parameters over the course of

training. In addition, we introduce a pre-trained imitative expert policy network (π^E), which can be an ensemble of networks or a single network, to generate the mean and standard deviation of the reference action distribution given a state input.

Both Q-function networks are learned with a single shared target, which is to minimize the mean-squared Bellman error (MSBE) as shown:

$$\mathcal{L}(\phi_i) = \underset{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}) \sim \mathcal{D}}{\mathbb{E}} \left[\left(Q_{\phi_i} (\mathbf{s}_t, \mathbf{a}_t) - (r_t + \gamma V_{targ} (\mathbf{s}_{t+1})) \right)^2 \right], \tag{2.10}$$

where $i = 1, 2$, $\mathcal{D}$ represents the experience replay buffer containing transition tuples, and V_{targ} is the target value function, which is updated once per the value network V_ψ update by polyak averaging.

The value function network V_ψ updates by the following loss function:

$$\mathcal{L}(\psi) = \underset{\substack{\mathbf{s}_t \sim \mathcal{D} \\ \tilde{\mathbf{a}}_t \sim \pi_\theta(\,\cdot\,|\mathbf{s}_t)}}{\mathbb{E}} \left[\left(V_\psi (\mathbf{s}_t) - \min_{i=1,2} Q_{\phi_i} (\mathbf{s}_t, \tilde{\mathbf{a}}_t) \right)^2 \right], \tag{2.11}$$

where the actions are sampled fresh from the policy $\tilde{\mathbf{a}}_t \sim \pi_\theta (\,\cdot\,|\mathbf{s}_t)$, whereas the states should come from the replay buffer $\mathbf{s}_t \sim \mathcal{D}$. Only the minimum Q-value between the two Q-function networks is taken, to mitigate the overestimation in the value function.

The policy network $\pi_\theta(\,\cdot\,|\mathbf{s}_t)$ outputs the parameters of a Gaussian distribution, i.e., the mean μ_θ and standard deviation σ_θ, such that the final output action can be generated by sampling from this distribution, $\mathbf{a}_t \sim \mathcal{N} (\mu_\theta (\mathbf{s}_t), \sigma_\theta^2 (\mathbf{s}_t))$. The policy network is trained to maximize the value function at all states, which is equivalent to maximizing the Q-value at any given state, and thus the loss function for the policy network becomes:

$$\mathcal{L}(\theta) = \underset{\substack{\mathbf{s}_t \sim \mathcal{D} \\ \tilde{\mathbf{a}}_t \sim \pi_\theta(\,\cdot\,|\mathbf{s}_t)}}{\mathbb{E}} \left[- \min_{i=1,2} Q_{\phi_i} (\mathbf{s}_t, \tilde{\mathbf{a}}_t) \right]. \tag{2.12}$$

Having constructed the Q-learning and policy learning processes, we will introduce two approaches to incorporate imitative expert priors into the actor-critic algorithm, in order to regularize the learned agent policy π_θ towards the expert policy π^E. Generally, there are two different ways: adding a divergence penalty to the

value function (value penalty) or constraining the divergence between the agent policy and expert policy (policy constraint). The details of these two ways are provided below.

2.4.3 Expert priors as value penalty

To encourage the learned policy to be close to the expert policy, we can add a penalty term into the reward function, resulting in the refactored reward function:

$$\bar{r}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) - \alpha D\left(\pi_\theta\left(\cdot\,|\mathbf{s}_t\right), \pi^E\left(\cdot\,|\mathbf{s}_t\right)\right), \tag{2.13}$$

where α is a temperature parameter, and D is a probability metric to measure the divergence between two distributions over actions and is chosen to be the Kullback–Leibler (KL) divergence, denoted as D_{KL} hereafter.

This is equivalent to directly incorporating the penalty term into the value function. Therefore, the updated loss function of the penalized value function can be rewritten as:

$$\mathcal{L}(\psi) = \mathop{\mathbb{E}}_{\substack{\mathbf{s}_t\sim\mathcal{D} \\ \widetilde{\mathbf{a}}_t\sim\pi_\theta(\,\cdot\,|\mathbf{s}_t)}}\left[\left(V_\psi\left(\mathbf{s}_t\right) - \left(\min_{i=1,2} Q_{\phi_i}\left(\mathbf{s}_t, \widetilde{\mathbf{a}}_t\right) - \alpha\hat{D}_{KL}\left(\pi_\theta\left(\cdot\,|\mathbf{s}_t\right), \pi^E\left(\cdot\,|\mathbf{s}_t\right)\right)\right)\right)^2\right], \tag{2.14}$$

where $\hat{D}_{KL}$ denotes the sample estimation of the KL divergence, $\pi_\theta\left(\cdot\,|\mathbf{s}_t\right)$ is the learned policy, and $\pi^E\left(\cdot\,|\mathbf{s}_t\right)$ is the expert policy obtained through imitation learning. Both are modeled as Gaussian distributions over actions conditioned on states.

Accordingly, the policy should maximize the penalized value function in each state, and thereby the loss function for the policy network should be modified as:

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{\substack{\mathbf{s}_t\sim\mathcal{D} \\ \widetilde{\mathbf{a}}_t\sim\pi_\theta(\,\cdot\,|\mathbf{s}_t)}}\left[-\min_{i=1,2} Q_{\phi_i}\left(\mathbf{s}_t, \tilde{\mathbf{a}}_t\right) + \alpha\hat{D}_{KL}\left(\pi_\theta\left(\cdot\,|\mathbf{s}_t\right), \pi^E\left(\cdot\,|\mathbf{s}_t\right)\right)\right]. \tag{2.15}$$

The update of the Q-function networks remains the same as Eq. (2.10), and now the actor-critic algorithm has been tailored to incorporate the behavior regularization through a penalty in the value function.

2.4.4 Expert priors as policy constraint

Instead of adding a penalty term to the value function, another approach is to explicitly constrain the deviation between the learned policy and expert policy within a small value during policy optimization. Hence, the constrained optimization problem for policy learning can be formulated as:

$$\begin{aligned} &\min_{\pi} \mathop{\mathbb{E}}_{\substack{\mathbf{s}_t \sim \mathcal{D} \\ \tilde{\mathbf{a}}_t \sim \pi_\theta(\cdot|\mathbf{s}_t)}} \left[-\min_{i=1,2} Q_{\phi_i}\left(\mathbf{s}_t, \tilde{\mathbf{a}}_t\right) \right], \\ &\text{s.t. } \hat{D}_{KL}\left(\pi_\theta\left(\cdot|\mathbf{s}_t\right), \pi^E\left(\cdot|\mathbf{s}_t\right)\right) \leq \epsilon, \end{aligned} \tag{2.16}$$

where ϵ is a small positive constant or the tolerance of the KL divergence, which represents the closeness of the learned policy to the imitative expert policy.

To solve this constrained optimization problem, we can construct its corresponding Lagrangian dual problem:

$$\max_{\lambda} \min_{\pi} \mathop{\mathbb{E}}_{\substack{\mathbf{s}_t \sim \mathcal{D} \\ \tilde{\mathbf{a}}_t \sim \pi_\theta(\cdot|\mathbf{s}_t)}} \left[-\min_{i=1,2} Q_{\phi_i}\left(\mathbf{s}_t, \tilde{\mathbf{a}}_t\right) + \lambda \left(\hat{D}_{KL}\left(\pi_\theta\left(\cdot|\mathbf{s}_t\right), \pi^E\left(\cdot|\mathbf{s}_t\right)\right) - \epsilon \right) \right], \tag{2.17}$$

where λ is the non-negative Lagrangian multiplier $\lambda \geq 0$.

We can employ the dual gradient descent method to optimize the unconstrained objective by alternating gradient descent steps on θ and λ respectively [48]. The core idea is to first solve the Lagrangian function (Eq. (2.17)) with respect to the policy π_θ under the current Lagrangian multiplier λ (starting with a random guess), and then update the Lagrange multiplier λ if the constraint is violated. Therefore, the modified loss function for the policy network becomes:

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{\mathbf{s}_t \sim \mathcal{D}} \left[\mathop{\mathbb{E}}_{\tilde{\mathbf{a}}_t \sim \pi_\theta(\cdot|\mathbf{s}_t)} -\min_{i=1,2} Q_{\phi_i}\left(\mathbf{s}_t, \tilde{\mathbf{a}}_t\right) + \lambda \left(\hat{D}_{KL}\left(\pi_\theta\left(\cdot|\mathbf{s}_t\right), \pi^E\left(\cdot|\mathbf{s}_t\right)\right) - \epsilon \right) \right]. \tag{2.18}$$

The Lagrangian multiplier λ gets updated according to the following loss function with the constraint $\lambda \geq 0$:

$$\mathcal{L}(\lambda) = \mathop{\mathbb{E}}_{\mathbf{s}_t \sim \mathcal{D}} \left[-\lambda \left(\hat{D}_{KL}\left(\pi_\theta\left(\cdot|\mathbf{s}_t\right), \pi^E\left(\cdot|\mathbf{s}_t\right)\right) - \epsilon \right) \right]. \tag{2.19}$$

For Q-learning, the update of the value function network remains the same as Eq. (2.11), and the update of the Q-function networks is consistent with Eq. (2.10). By following this approach, the actor-critic algorithm can be modified to incorporate behavior regularization through a constraint on the divergence between the learned policy and the expert policy.

2.5 Experiments

2.5.1 Driving scenarios

Our proposed algorithm is validated in the context of urban autonomous driving problems. We design two challenging urban driving scenarios using the SMARTS simulation platform [49]. Fig. 2.2(a) presents a scenario where the autonomous driving agent must execute an unprotected left turn in heavy traffic. This requires the agent to turn left onto a major road, traverse a two-way four-lane road, and finally reach the rightmost lane, without the regulation of traffic lights. Fig. 2.2(b) shows a first-person view from the ego vehicle in the left turn scenario, allowing a human expert to observe the environment. In Fig. 2.2(c), a roundabout scenario is designed, and the autonomous driving vehicle needs to safely navigate from the start point to the destination. The green area in Fig. 2.2 shows the accessible routes of the ego vehicle, and the waypoints ahead of the ego vehicle (up to 50 meters) are shown. These two scenarios are sufficiently different and challenging to validate the capabilities of our proposed method from different angles. The left turn scenario is more complex in terms of the traffic situations, yet the vehicle maneuver is simpler (focusing on speed control) and it requires less time to complete. Conversely, the roundabout scenario has less complex traffic but demands more time to finish and involves subtler maneuvers, such as avoiding collisions and changing lanes to evade congestion.

To simulate real-world situations, we generate diverse driver behaviors for environment vehicles, given that different types of drivers may be present on the road. Specifically, we randomly choose the parameters of the driving behavior models (e.g., speed distribution, maneuver imperfection, impatience of waiting at intersections, and willingness to cooperate) for a specific type of actor within a reasonable

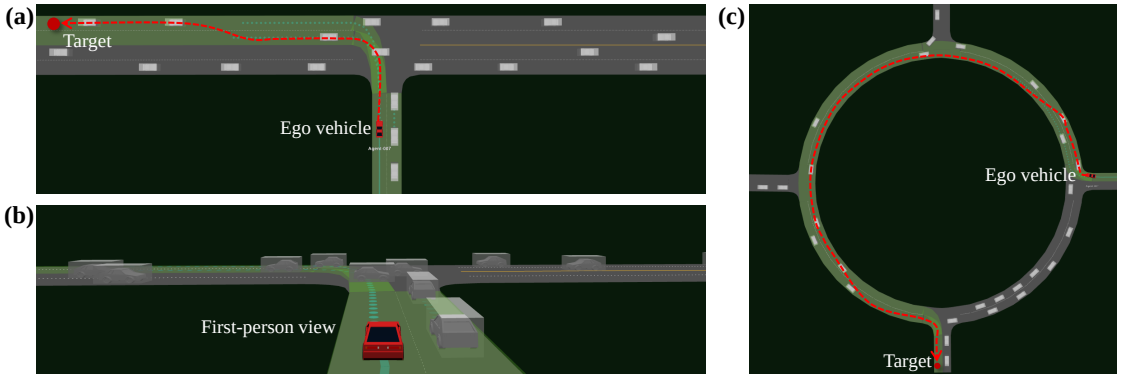


FIGURE 2.2: The designed urban driving scenarios in the SMARTS platform

range, and each traffic flow consists of a mix of different types of actors. We generate 20 different traffic flows for training, and the initial conditions (e.g., vehicle spawn points) are randomly generated by the simulator at the beginning of each episode. In testing, we generate additional 50 traffic flows to test the generalization capability of our trained policy.

2.5.2 MDP formulation

State space. We use a bird-eye view image encoding the environment information within a square area of 32×32 meters, with the ego vehicle at the center. This image is assumed to be obtained from an upstream perception module. The bird-eye view image is an RGB image of the abstracted driving scene (see Fig. 2.3), where the grey area represents the road, the red rectangle denotes the ego vehicle, and the white rectangles are surrounding vehicles. Since this representation encodes a rich amount of information about the road structure and interaction situations and can cope with an arbitrary number of objects, it is suitable for highly interactive urban driving scenarios. The size of the bird-eye view image is $80 \times 80 \times 3$, resulting in a detection area resolution of 0.4 m/pixel. We use a stack of three consecutive bird-eye-view images from the current time step and two previous time steps to include a history of observations, yielding a state space with dimensions of $80 \times 80 \times 9$.

Action space. We adopt a lane-following controller that allows a high-level target input (i.e., target speed V_T and lane change L_T) to control the vehicle rather than direct vehicle control commands (e.g., steering and accelerator). The longitudinal motion control is a continuous target speed V_T within the range of $[0, 10]$ m/s,

which is then normalized into $[-1, 1]$. The lateral motion is determined by the discrete lane change action L_T, specifically changing to the left lane $L_T = -1$, lane-keeping $L_T = 0$, and changing to the right lane $L_T = +1$. To adapt the discrete action into a continuous value for the lane-change control, we discretize a continuous range $[-1, 1]$ into three equal-sized bins, each representing a discrete action. For example, $[-1, -1/3]$ corresponds to the action of changing to the left lane, while $[-1/3, 1/3]$ represents the lane-keeping action. The lane-changing action executed at a decision-making step will not complete the entire lane-change maneuver; thus, the policy must output the lane-change action during consecutive decision-making steps to finish a lane-change maneuver. We choose this control setting because longitudinal motion control necessitates more refined and precise actions, while lateral motion control can be achieved through discrete lane-following or lane-changing actions, as the vehicle typically follows road structure and traffic rules and maintains its lane.

Reward function. We use a sparse reward function, which only emits a meaningful value at the end of a training episode:

$$r(s, a) = r_{\text{collision}} + r_{\text{goal}}, \tag{2.20}$$

where $r_{\text{collision}} = -1$ is the penalty for colliding with other objects, and $r_{\text{goal}} = 1$ is the reward for reaching the goal position. Both values are zero otherwise.

However, we find that the sparse reward makes it very difficult for the baseline RL algorithms to achieve at least an acceptable performance. To improve the performance of baseline RL algorithms for a fair comparison, we employ the reward shaping technique, and the shaped reward function $r'(s, a)$ is:

$$r'(s, a) = 0.001 r_{\text{speed}} + r_{\text{collision}} + r_{\text{goal}}, \tag{2.21}$$

where r_{speed} is the speed of the vehicle. It is important to note that this shaped reward function is only used in other RL algorithms to enhance their performance, and to prevent interference from the reward-shaping term with our proposed method.

2.5.3 Expert demonstration

We request a human expert to execute the designed urban driving tasks in the simulator and collect their actions as expert demonstration data. The human expert observes the driving environment in a first-person view (see Fig. 2.2(b)) and provides their actions, with four discrete actions available for manipulation via keyboard input. For the longitudinal direction, two discrete actions, speed up (increase the speed by 2 m/s) and slow down (decrease the speed by 2 m/s), are used to set the target speed for the vehicle. For the lateral movement, two discrete actions (change to the left lane and the right lane) are used, with the default maneuver being to maintain the current lane. The human expert can execute the longitudinal and lateral control actions simultaneously, but need not provide actions at every timestep. The vehicle will continue executing the last set target until the expert alters the target. We also ask the expert to demonstrate various behaviors in the unprotected left turn task, such as aggressive behavior like nudging forward in the intersection, and conservative behavior like stopping to find an acceptable gap. The demonstration data is stored as state-action pairs, and only successful task executions are counted. The states are normalized to $[0, 1]$ and the actions are normalized to $[-1, 1]$. We collect 40 trajectories of human expert demonstration in the roundabout scenario, and for the left turn scenario, we collect 40 trajectories each for two distinct driving behaviors. We also investigate the effects of sample size in training imitative expert policy on the training and testing results of the RL agent.

2.5.4 Comparison baselines

We implement several RL and IL methods as baselines to benchmark the performance and efficiency of our proposed method. The baseline methods are listed below.

- Soft actor-critic (SAC) [43]. SAC is a state-of-the-art off-policy RL algorithm, which optimizes a trade-off between the expected return and entropy. We adopt the version that can automatically tune the entropy parameter.
- Proximal policy optimization (PPO) [50]. PPO is a state-of-the-art on-policy RL algorithm that can stabilize policy training by limiting the new policy

from deviating too far from the old policy, using a clipped surrogate objective function.

- Generative adversarial imitation learning (GAIL) [51]. GAIL employs expert demonstration to recover a policy using generative adversarial training. The policy serves as the generator, trained with an RL algorithm using a surrogate reward given by the discriminator, which measures the similarity between generated and demonstration samples.

- Behavioral cloning (BC). BC is a supervised learning method aiming to reproduce the expert demonstration actions at given states. We train a stochastic policy using the behavioral cloning method with uncertainty estimation for comparison, and for convenience, we directly use the imitative expert policy as the BC baseline. Since the learning mechanism of BC is different from RL training, it is only compared against our method during the testing phase.

2.5.5 Implementation details

The structures of the neural networks are shown in Fig. 2.3. All networks share the same state encoder structure, which consists of four convolutional layers and a global average pooling layer, followed by different structures of fully connected layers for different networks to generate desired outputs. The hyperparameters of the convolutional layers are provided in the figure in a tuple format, following the meaning of (number of convolution kernels, kernel size, and moving stride). The numbers of hidden units of the fully connected layers are also given. The action distribution, which consists of two continuous actions with respect to longitudinal and lateral motion control, is modeled as a multivariate Gaussian distribution. The policy network generates the parameters of the Gaussian distribution, while the value and Q-value networks output a single value, respectively. All the layers, unless explicitly notated in Fig. 2.3, use the ReLU activation function. All the networks are trained with the Adam optimizer in Tensorflow using an NVIDIA RTX 2080 Super GPU.

For training the ensemble expert policy model, we train five single policy networks with different random seeds, both in the initialization of the weights of the network and shuffle of the dataset, and each network in the ensemble is trained for 100

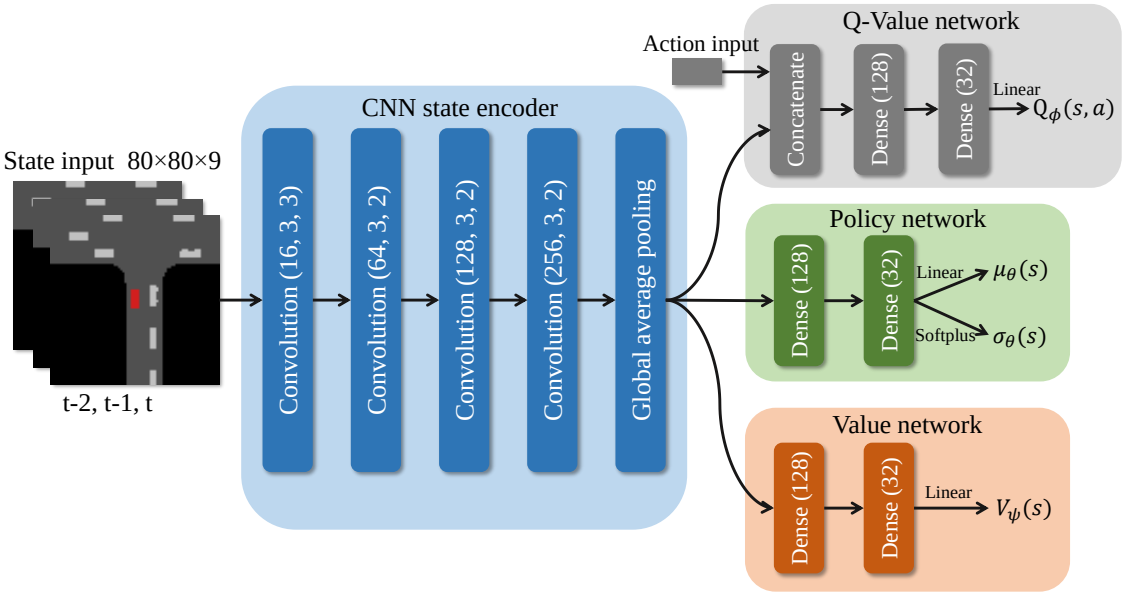


FIGURE 2.3: The structures of the neural networks in RL

epochs. The policy network in the ensemble adopts the same structure as the policy network in Fig. 2.3. Since the expert demonstration actions (target speed, lane selection) are recorded in discrete values, directly regressing on these discrete values makes the network difficult and unstable to train and performs very poorly. Therefore, we add a small normally distributed random number to the discrete actions, which does not significantly change the actual action but makes the neural network easier and more stable to train. The standard deviation of the Gaussian distribution of the obtained expert policy model is added by 0.1 to form a final output action distribution, which is to ensure that the distribution is wide enough to cover feasible actions.

The simulation runs on an AMD Ryzen 3900X CPU and takes approximately one hour to complete 100,000 steps of interaction with the simulated environment. The simulation interval is 0.1 seconds, which means the RL agent makes decisions every 0.1 seconds. The hyperparameters used in this paper are carefully tuned to achieve the best training performance, and the final settings are listed in Table 2.1.

TABLE 2.1: Hyperparameters used in the experiment

Symbol	Meaning	Value
M	Number of the ensemble models	5
α	KL divergence parameter	0.002
λ_0	Initial Lagrange multiplier	0.01
ϵ	KL divergence tolerance	0.8
N_B	Size of the replay buffer	20000
$\mathcal{B}$	Training batch size	32
lr	Learning rate	0.0003
γ	Discount rate	0.99
N_{warm}	Warm-up steps	5000
N_{train}	Total training steps	100000

2.6 Results and Discussions

2.6.1 Training results

We apply the proposed method along with baseline methods to train an autonomous driving agent in the designed driving scenarios. In the unprotected left turn scenario, the human expert purposely demonstrates two distinct driving styles, namely conservative and aggressive. Conservative behavior is reflected by a driver always stopping at the intersection, waiting for an adequate gap to cross without interrupting other vehicles, whereas aggressive behavior means that a driver would nudge forward in the intersection to show aggressiveness, forcing other vehicles on the main road to yield. In the roundabout scenario, the goal of the human expert is to pass through the roundabout as quickly as possible and avoid collisions, to ensure both efficiency and safety. The imitative expert policy here is derived with policy and model uncertainty estimations (ensemble model), and the effects of uncertainty estimation will be discussed subsequently. For each method, ten trials are conducted with different random seeds to reflect the average training performance of that method. Because the proposed method and baseline methods use different reward functions, the scales of the average episodic reward are different; thus, we use the average success rate to gauge the training performance. It is defined as the number of successfully finished episodes over the last 20 episodes divided by 20. Success is counted only when the autonomous driving vehicle reaches the goal within a limited time. Failure conditions include colliding

with other vehicles, going off the drivable road, and exceeding the maximum allowed time. The maximum allowed time for each episode in the left turn scenario is 40 seconds and 60 seconds for the roundabout scenario.

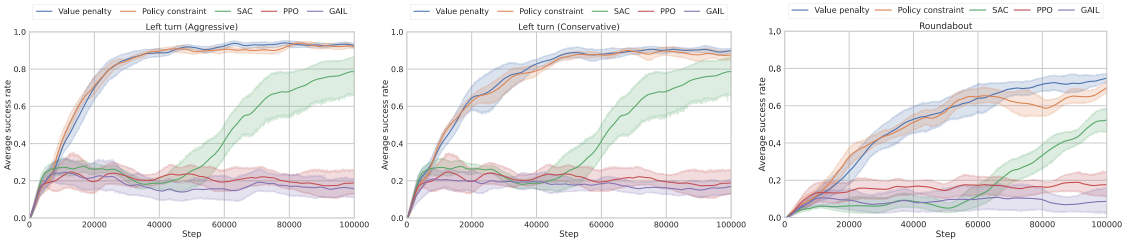


FIGURE 2.4: The training processes of different learning methods in the urban driving scenarios

Fig. 2.4 shows the training results of different methods in the urban driving scenarios, in which the training curve is represented by a solid line of the mean value and an error band of the 95% confidence interval. To smooth the training curve, the average success rate at each step is the exponentially weighted moving average over the immediate 3000 steps. In the unprotected left turn scenario, our proposed methods achieve the best performance at the end of the training process, and the value penalty method and policy constraint method perform equally well in this scenario. In addition to the boost in training performance, our method shows a significant improvement in sample efficiency. It only takes about 30% of interaction steps to reach the same performance as that of SAC, improving the sample efficiency by 70%. This is because while SAC is randomly searching for better actions, the imitative expert policy used in our method could provide a reasonable direction for searching for desired actions, thus remarkably enhancing the sample efficiency. Furthermore, the training process of SAC is unstable across different trials, as reflected in the wide error band (high deviation). In contrast, the superior performance of our proposed method is consistent across different trials. The other two baseline methods, PPO and GAIL, do not perform very well. Their performance hardly improves during the training process, and the training curve almost remains flat. This is because PPO can easily get trapped in local optima, and for GAIL, the adversarial training setting is difficult and parameter-sensitive, which may cause it to perform poorly with high-dimensional state input. In terms of the difference in learning from different driving behaviors, we can observe that learning conservative behavior is more challenging than learning aggressive behavior. This is probably because stopping to find enough gaps is a very subtle behavior and hard to learn, and there is ambiguity about when to start crossing. Moreover, we

can find out aggressive driving behavior can lead to a higher success rate. One possible reason is that the environment vehicles are intended to yield to the ego vehicle even if it violates the safety requirement, and aggressive driving behavior may exploit this feature.

In the roundabout scenario, our proposed method still demonstrates its advantages, achieving a higher success rate and better sample efficiency than the other methods. Unlike the left turn scenario where decisions are made in a small fragment of the scene (the intersection), the roundabout scenario is larger, involving more complicated situations, and requires a longer time to finish the task. Therefore, the value penalty method is more favorable than the policy constraint method in this situation. This is because the reward feedback from the environment is sparse, and adding additional feedback to the reward could help better estimate the value of actions and states. Compared to SAC, our proposed method sees a 60% improvement in sample efficiency and a better success rate at the end of training. PPO and GAIL still perform poorly, and they can hardly improve the policy during the training.

2.6.2 Testing results

In this subsection, we evaluate the testing performance of the learned driving policies. We first generate 50 distinct traffic flows that differ from the training situations. We then employ the driving policy with the best training return to control the ego vehicle, navigating it from the starting point to the target point. During testing, we only use the mean value of the stochastic policy (Gaussian distribution policy) as the output action, instead of sampling from it. We test all learned driving policies obtained from different learning methods for 50 episodes, and use the testing success rate, defined as the percentage of successful episodes divided by the number of testing episodes, to measure the performance and generalization capability. The results are presented in Table 2.2 (unit: %, A: aggressive, C: conservative).

As indicated by the data in Table 2.2, the testing performance of each method is generally consistent with their respective training performance. Our proposed value penalty method achieves the highest testing success rate in all three situations, encompassing various driving scenarios and demonstration behaviors. The

TABLE 2.2: Testing success rate of different learning methods in the urban driving scenarios

	Unprotected left turn (A)	Unprotected left turn (C)	Roundabout
BC	72	70	36
GAIL	26	24	32
PPO	64	64	34
SAC	78	78	74
Ours (policy constraint)	92	90	80
Ours (value penalty)	**96**	**96**	**84**

basic behavioral cloning method outperforms GAIL and PPO, partially due to the low sample efficiency of on-policy RL algorithms. However, our proposed method attains superior performance with the same number of training steps, demonstrating higher sample efficiency. Although SAC performs relatively well in both unprotected left turn and roundabout scenarios in terms of success rate, it falls short compared to our proposed method.

In addition, we present the duration of the successful episodes for each learning method in Table 2.3 (unit: seconds, A: aggressive, C: conservative), which illustrates the different outcomes of driving behaviors when dealing with the unprotected left turn scenario. The duration of the driving policies with conservative driving behavior is four to five seconds longer than that of aggressive behavior, as the vehicle needs to wait for enough gap in the intersection. For the SAC agent, the duration is significantly shorter than other methods, because it is motivated to reach the target as quickly as possible at a higher speed. However, it is important to note that the behavior of SAC is much more aggressive (rushing through the intersection), and safety is somewhat compromised, resulting in more collisions at the intersection. In contrast, our method can learn from a human expert that shows a balance between safety and efficiency, as well as different driving styles. In the roundabout scenario, the duration of GAIL and PPO agents is considerably longer than SAC and our proposed method, because the vehicle only moves at a slower speed. Although the SAC agent can move faster than GAIL and PPO, it often sticks on the outer lane and does not learn to change lanes to gain a higher speed, resulting in a longer time to complete the task compared to our proposed method. Conversely, our proposed method with human demonstration can learn to change lanes to run faster and complete the more quickly.

TABLE 2.3: Testing duration of different learning methods in the urban driving scenarios

	Unprotected left turn (A)	Unprotected left turn (C)	Roundabout
BC	14.89 ± 1.54	19.97 ± 2.12	36.53 ± 2.85
GAIL	15.04 ± 0.67	17.88 ± 0.92	57.12 ± 0.89
PPO	13.65 ± 0.83	–	53.48 ± 0.76
SAC	11.84 ± 0.95	–	43.55 ± 4.70
Ours (policy constraint)	14.72 ± 1.13	19.29 ± 2.24	37.04 ± 3.97
Ours (value penalty)	14.26 ± 1.24	19.62 ± 2.05	36.62 ± 2.84

As depicted in Fig. 2.5, we analyze the vehicle dynamic states of different driving policies in some representative cases. Fig. 2.5(a), (b), and (c) display the speed, acceleration, and curvature of the vehicle in the unprotected left turn scenario for our proposed methods and SAC, respectively. We can observe the different behaviors of aggressive and conservative policies. For aggressive policy, the vehicle will cross the intersection at a low speed (nudging forward); for conservative policy, the vehicle will stop at the intersection, waiting for enough clearance to cross. The stability of the vehicle is maintained, as the acceleration and curvature are within a reasonable and comfortable range. For the SAC agent, it will cross the intersection at a higher speed, forcing other vehicles on the main road to yield, and its acceleration profile is more volatile. Fig. 2.5(d) and (e) exhibit the vehicle dynamic states of our proposed method and SAC in the roundabout scenario. For our proposed method, the vehicle is able to decelerate when entering the roundabout, change to the inner lane to gain a higher speed, and slow down to wait for other vehicles when exiting the roundabout. The acceleration and curvature are smooth and within a comfortable range. However, for the SAC agent, since it only stays in the outer lane, it has to frequently adjust its speed to yield to other vehicles, and the vehicle may decelerate harshly to avoid a collision.

We also provide supplementary videos[1] to illustrate the behaviors of different driving policies in detail. Overall, our proposed method demonstrates human-like driving behaviors, such as nudging forward in the intersection to show its aggressiveness, stopping and waiting for gaps, changing lanes to gain a higher speed, and decelerating to reduce risk. Although the SAC agent can learn to accomplish the driving tasks, it does not display human-like behavior, as it shows reckless moves

[1] https://mczhi.github.io/Expert-Prior-RL/

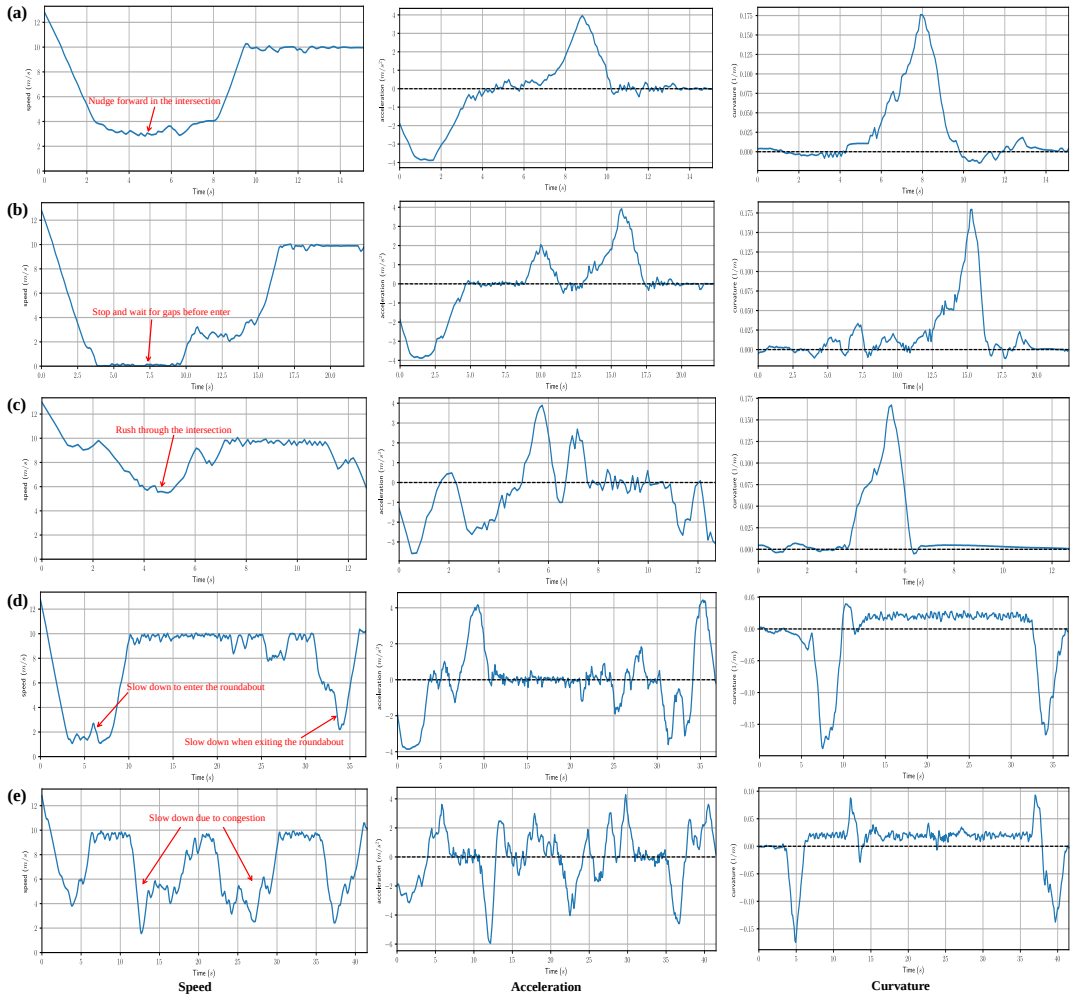


FIGURE 2.5: Vehicle dynamic states of different driving policies in the urban driving scenarios

like directly cutting into the main road without negotiation and sticking to the slow lane. This is mainly because it learns only to avoid collisions, attain a higher speed, and reach the target according to the reward function, without the guidance of human prior knowledge. The PPO and GAIL methods are inefficient and perform poorly in the test, as they have only learned to maintain a low speed and cannot stop to avoid a collision.

2.6.3 Effects of imitative expert policy

Uncertainty estimation. We introduce the ensemble model that can quantify both policy and model uncertainties to obtain the expert policy. In this subsection, we discuss the effects of different uncertainty estimations for expert policy

derivation on the training and testing performance of our proposed method. In addition to the proposed ensemble model, we use two other methods to acquire the expert policy. One is a Gaussian distribution policy that considers only policy uncertainty, meaning only a single policy network in the ensemble model is used. The other method is a Gaussian distribution policy with fixed uncertainty, which entails taking the mean value from a single policy network and setting the standard deviation of the distribution as a constant. The fixed standard deviation is set to 0.2, slightly higher than the mean standard deviation provided by the ensemble method during training. We employ the value penalty method since it delivers better learning results, and the average success rate is employed as the metric.

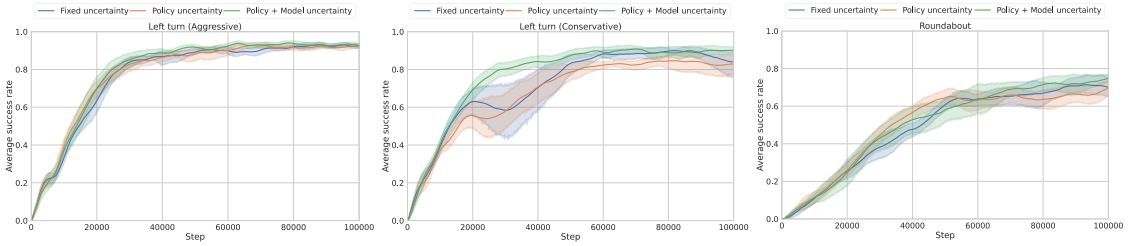


FIGURE 2.6: The training processes with different uncertainty estimations for the expert policy

The training results in Fig. 2.6 indicate that our proposed method would benefit from the expert policy that can estimate both policy and model uncertainty through the ensemble method. This advantage becomes more pronounced as the difficulty of the task increases. There is no significant difference between these methods in the unprotected left turn scenario learning aggressive behavior, because the behavior is simple and solution space for this task is limited. However, the ensemble method can stabilize the training process and obtain a higher success rate when learning conservative behavior, while the other two methods could cause an unstable training process and high variances across different trials. This is likely because conservative behavior is more subtle and hard to learn for the expert policy, and a single Gaussian policy network estimating only policy uncertainty may not generate a high uncertainty in uncertain states. On the other hand, the ensemble method can emit a larger uncertainty (representing the overall policy and model uncertainty) in uncertain states, which could enhance the exploration of the RL agent. The ensemble method also outperforms the fixed uncertainty method since the fixed uncertainty cannot provide a sensible search range search in different states. In the roundabout scenario, the ensemble method also shows a better result than the other two methods. The data in Table 2.4 (unit: %, A:

aggressive, C: conservative) concerning the testing success rate of different expert policy derivation methods also reveals that the ensemble expert policy with both policy and model uncertainty estimations is advantageous, providing better testing performance and leading to improved robustness and generalization ability. Therefore, we conclude that the ensemble method is capable of estimating both policy and model uncertainties and is thus more effective at learning the expert policy (generating expert action distributions), resulting in better performance in guiding the training of RL agents, especially for difficult tasks.

TABLE 2.4: Testing success rate of the proposed method with different uncertainty estimations for the expert policy

	Unprotected left turn (A)	Unprotected left turn (C)	Roundabout
Fixed uncertainty	90	84	66
Policy uncertainty	**96**	90	72
Policy+model uncertainty	**96**	**96**	**84**

Training sample size. Another factor that could affect the quality of the expert policy and training of the RL agent is the sample size of the expert demonstration. In this subsection, we investigate the influence of the training sample size for the expert policy on the training and testing performance of our proposed method. Specifically, we select 10, 20, and 40 demonstration trajectories to train the expert policy, respectively, and use them to guide the training of RL agents. The expert policies are trained with the ensemble method, and the RL agents are trained using the value penalty method.

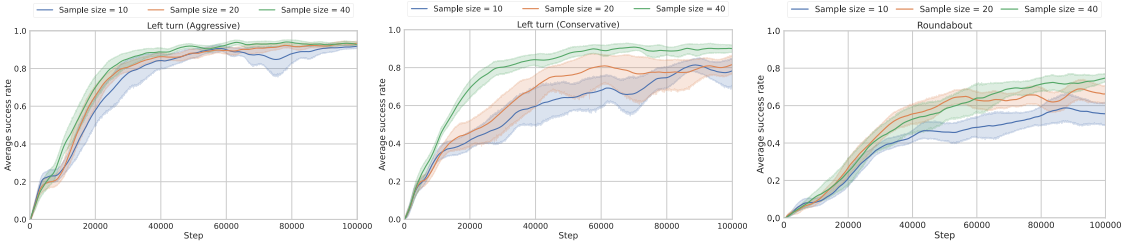


FIGURE 2.7: The training processes with different training sample sizes for the expert policy

The results in Fig. 2.7 reveal that increasing the training sample size is beneficial for training the expert policy and consequently for training RL agents. The difference in sample size becomes more significant with the increase in task complexity. In the unprotected left turn scenario learning aggressive behavior (relatively simple),

a small number of demonstration trajectories can effectively guide RL training, improving the policy to a satisfactory level, although the learning efficiency and stability are compromised. However, when learning conservative behavior (relatively difficult), a larger training sample size can lead to a better performance in terms of average success rate and learning speed. In the roundabout scenario, the difference between the sample sizes is more notable regarding the obtained success rate at the end of training. The results in Table 2.5 (unit: %, A: aggressive, C: conservative) about the testing success rate of different sample sizes demonstrate that extending training samples is also beneficial for the test performance, especially for more difficult tasks. The possible reason is that increasing training samples with diverse demonstration data enables the expert policy to generate reference actions more accurately and, thus, provide more accurate action distribution to regularize the RL agent's behavior.

TABLE 2.5: Testing success rate of the proposed method with different training sample sizes for the expert policy

	Unprotected left turn (A)	Unprotected left turn (C)	Roundabout
Sample size = 10	92	70	60
Sample size = 20	94	80	78
Sample size = 40	**96**	**96**	**84**

2.6.4 Discussions

The two most significant challenges of RL can be categorized into two areas: one is the low sample efficiency, which requires a massive amount of interactions, and the other is the difficulty in specifying the reward function. Our proposed method offers insights into addressing these issues by incorporating human prior knowledge. Specifically, we propose distilling human knowledge, in the form of human demonstration of executing a driving task, into an imitative expert policy, which is subsequently used to guide the training of RL agents. As evidenced by the results presented in the previous subsections, the strengths of our method include improved sample efficiency (60% compared to the state-of-the-art RL algorithm) and reduced requirement for designing reward functions (only using sparse reward), which could potentially facilitate the application of RL-enabled human-like autonomous driving systems. That being said, some limitations of the proposed method should be

acknowledged. The major drawback is that we use human experts' high-level decisions as demonstration data, which does not scale to real-world scenarios when only the trajectories are available. Additionally, more hyperparameters are introduced in the proposed algorithm, and it is time-consuming to tune them for optimal performance. Therefore, future work will focus on utilizing naturalistic human driving data to learn expert policies and guide the training of RL agents. Furthermore, exploring more methods for deriving the expert policy and generalizing the proposed method to other scenarios is warranted.

2.7 Conclusions

In this chapter, we propose a framework that combines human prior knowledge and deep reinforcement learning and apply it in autonomous driving scenarios. Our proposed method consists of three key steps: expert demonstration, policy derivation, and reinforcement learning. First, a human expert demonstrates their execution of the task, potentially with their own preferences, in the expert demonstration step. In the policy derivation step, the expert policy is obtained using imitation learning and uncertainty estimation based on expert demonstration data. Finally, in the reinforcement learning step, we propose an actor-critic-based RL algorithm that can incorporate the imitative expert policy into RL training. Specifically, we introduce two methods to regularize the KL divergence between the RL agent's policy and the imitative expert policy, namely value penalty and policy constraint. To validate our method, we design two driving scenarios and implement two distinct driving behaviors demonstrated by a human expert in the left turn scenario. The training results reveal that our method can not only achieve the best performance but also significantly improve the sample efficiency compared to baseline algorithms. In terms of testing the learned driving policies in different traffic conditions, the results also verify the performance of our method, with the highest success rate. Additionally, the agent is able to perform different driving behaviors provided by the human expert, as well as exhibit some human-like driving features. Moreover, we find that the value penalty method generally performs better in environments with sparse rewards. Using the ensemble expert policy with both policy and model uncertainty estimations, as well as increasing training examples, can lead to better performance, especially for more challenging tasks.

Chapter 3

Learning Cost Aligned with Human Values

Acquiring cost functions for driving behaviors that aligned with human values is of great importance for the development of intelligent and personalized autonomous driving systems. In this chapter, a driving model based on internal reward functions, which emulates human decision-making mechanisms, is employed. To infer the reward function parameters from naturalistic human driving data, we propose a structural assumption about human driving behavior, centering on discrete latent driving intentions. This approach converts the continuous behavior modeling problem to a discrete setting, thus rendering maximum entropy inverse reinforcement learning (IRL) tractable for learning reward functions. Specifically, a polynomial trajectory sampler is adopted to generate candidate trajectories that account for high-level intentions while approximating the partition function within the maximum entropy IRL framework. An environment model considering interactive behaviors between the ego and adjacent vehicles is constructed to better estimate the generated trajectories. The proposed methodology is applied to learn personalized reward functions for individual human drivers from the NGSIM highway driving dataset. Qualitative findings indicate that the learned reward functions are able to explicitly represent the preferences of different drivers and interpret their decisions. Quantitative results reveal the robustness of the learned reward functions, as evidenced by a negligible decrease in proximity to human driving trajectories when employing the reward function under test conditions. For the testing performance,

the personalized modeling method outperforms the general modeling approach, significantly reducing modeling errors in human likeness (a custom metric for accuracy assessment), and these two methods deliver better results compared to other baseline methods. Furthermore, the study finds that predicting the response actions of neighboring vehicles and incorporating their potential decelerations caused by the ego vehicle are critical in estimating the generated trajectories, and the accuracy of personalized planning using the learned reward functions relies on the accuracy of the forecasting model.

3.1 Introduction

Achieving human-like driving is a critical objective for autonomous vehicles (AVs) targeting widespread deployment in the real world. It is conceivable that AVs and human drivers share the roads in the near future, which requires AVs to behave in a human-like manner, ensuring predictability and interpretability for other human drivers. However, current AVs show shortcomings in these aspects, which would lead to conservative and unnatural decisions that may confuse and even endanger other human drivers [52]. This is primarily attributed to their inability to interact with other human traffic participants or, more specifically, to reason about other agents' possible behaviors and make proactive decisions accordingly. This problem underscores the importance of studying and understanding human driving behaviors to enable a safe and efficient autonomous driving system. Moreover, personalization constitutes another essential facet of human-like driving, which means AVs should make decisions based on user-specific preferences [53]. This consideration motivates our research into individual driving behavior, with the aim of explicitly and individually expressing human driving styles.

Recent advancements in artificial intelligence [54] have provided us with powerful tools in learning human driving behaviors for AV decision-making, with imitation learning emerging as a state-of-the-art method for learning decision-making processes by emulating human demonstrative actions [20, 55]. We adopt a similar technique; however, instead of directly learning decision-making policies, we learn the internal reward function used in decision-making. The underlying assumption is that rational drivers choose actions that optimize their internal reward functions. We adopt the internal reward function approach to model driving behavior

for the following reasons. Firstly, this approach formulates the motivations behind agents' actions and is believed to better reflect human internal decision-making processes. Secondly, the form of a reward function (typically linear) is highly succinct and interpretable because of its explicit physical meanings [56], allowing the weights of the reward function to be adjusted to explicitly reflect the preferences of various human drivers. Inverse reinforcement learning (IRL) has emerged as a prominent approach for determining the parameters of the reward function that optimally explain human behavior, with maximum entropy IRL [57] attracting considerable attention in driving behavior modeling owing to its ability to address the stochasticity of driving behavior and the ambiguity arising from multiple reward functions that can explain human behavior. However, the original setting of maximum entropy IRL is limited to discrete and small-scale problems, as it relies on value iteration to evaluate the reward and state visitation frequency for feature expectation calculation, which is intractable to solve in high dimensional problems with large and continuous state spaces.

In our driving behavior modeling task, we observe that human actions are indeed governed by their latent high-level tactical intentions. Thus, we propose a structural assumption of human driving behavior that focuses on the discrete latent states, rather than continuous states and actions, enabling the maximum entropy IRL framework to recover the underlying reward functions. Specifically, we assume that a human driver makes long-term decisions, rather than instantaneous actions, based on three sequential processes: generation, evaluation, and selection. In essence, a human driver first generates multiple candidate trajectories in mind, anticipates their outcomes, evaluates the rewards, and finally selects one to follow. This structured assumption facilitates the calculation of the partition function in the maximum entropy IRL framework and also significantly reduces the computation complexity. We validate the effectiveness of the proposed method and apply it to learn diverse and mixed reward functions of different human drivers from a naturalistic human driving dataset on a highway. The main contributions of this chapter are listed as follows.

- We apply maximum entropy inverse reinforcement learning combined with the proposed structural assumption for driving behavior modeling using naturalistic highway driving data. We demonstrate the effectiveness and efficiency

of the proposed method in driving behavior modeling both qualitatively and quantitatively.

- We investigate two modeling assumptions: personalized modeling, in which each driver possesses a distinct cost function, and general modeling, in which all drivers share a common cost function. The personalized modeling method exhibits superiority over the general modeling method in terms of robustness and modeling accuracy.
- We examine the effects of simulating interactive behaviors in the environment model when evaluating the generated trajectories and incorporating the agent's interaction awareness into the reward function. The results indicate that both factors significantly influence the estimation of rewards for generated trajectories and improve modeling accuracy. Furthermore, we investigate the accuracy of applying the learned reward functions in the personalized planning process.

3.2 Literature Review

3.2.1 Driving behavior modeling

There is a large body of literature on the topic of modeling human driving behavior, encompassing a broad range of problem formulations, model assumptions, and methodologies [58]. Driving behavior modeling tasks primarily fall into three categories: intention estimation, motion prediction, and pattern analysis. Intention estimation aims to identify a driver's immediate future intentions. Classic methods include the parametric models such as the intelligent driver model (IDM) [59] and minimizing overall braking induced by lane changes (MOBIL) model [60], as well as data-driven models like the hidden Markov model [61]. For motion prediction tasks, which predict the future physical states of a vehicle, neural network models are the dominant method, such as convolutional neural networks [62] and long short-term memory networks [63]. Parametric models generally postulate some structure about the problem, resulting in high interpretability and computational efficiency. However, these models are not very expressive in reflecting complex dynamics, and the parameters are hard to specify. Data-driven methods do not

make strong structural assumptions, relying instead on a wealth of data to extract underlying patterns in agent behaviors and make predictions. While these methods boast strong performance, their lack of interpretability and generalization limits their application to safety-critical problems. We utilize the generalized form of agents' internal decision-making schemes, which is mathematically formulated as the reward function. It is easier to integrate into various control and planning frameworks, and we propose extracting the parameters of the reward functions from naturalistic human driving data.

Pattern analysis is another important branch in driving behavior research, focusing on extracting features or patterns from human driving data to help understand the traits of driving behaviors. For example, Siami *et al.* developed an unsupervised pattern recognition framework to extract driving patterns from mobile telematics data, identifying 29 unique driving styles from the data [64]. Birrell *et al.* examined the correlation between certain parameters of human driving behavior and good fuel economy in real-world driving scenarios [65]. More recently, with the advent of the assisted driving system, research on driving behavior modeling in the driver-vehicle system at a micro level has gained great interest. For instance, Na and Cole proposed using game theory to model a human driver's steering control behavior in response to vehicle automated steering intervention [66]. Xing *et al.* presented a deep learning-based joint driver behavior reasoning system to recognize both the driver's physical and mental states [67].

3.2.2 Inverse reinforcement learning and its applications in driving behavior modeling

Many human driving behavior models employ an optimization setting, which postulates that human behavior optimizes the expected reward of actions over time. Therefore, IRL has been widely used in many works as a tool to infer reward functions from expert demonstrations. The core idea of IRL is to adjust the weights of the reward function to produce a policy that matches the expert demonstrations (trajectories). Several IRL algorithms have been applied in driving behavior modeling, including the maximum margin method to learn driving styles and maneuver preferences [68], and the maximum entropy method for learning individual styles

[69]. The maximum entropy method is more popular as it can address the ambiguity that multiple reward functions can explain the expert's behavior. On the other hand, Wulfmeier *et al.* proposed deep maximum entropy IRL capable of learning highly nonlinear cost maps from raw high dimensional state input and applied it to large-scale vehicle navigation tasks [70]. Although the network parameterization can be very expressive, the interpretability and generalization capability could be impaired. Since we aim to explicitly represent and interpret human driving behavior, the maximum entropy method with a linear reward function setting is more favorable.

The most significant challenges of maximum entropy IRL in driving behavior modeling involve continuous and large state spaces and the computationally expensive RL process required to evaluate the reward function at each iteration. To this end, many works opt to optimize a sequence of actions or a trajectory instead of stepwise actions during the evaluation process. The remaining challenge is the continuous and high dimensional state space, which makes computing the partition function intractable. [71, 72] utilized Laplace approximation to reshape the reward function of a trajectory, considering only local optimal as in [73]. This approach enables the partition function to be solved analytically, but the assumption of local optima may not hold in real-world cases. [69] proposed optimizing the spline trajectory with the updated reward function, and only the optimal trajectory is considered to calculate the feature expectation to update the reward function. Similarly, [74] utilized a spatiotemporal state lattice planner to search for an optimal trajectory, which is then used to calculate feature expectations in the IRL framework. However, the assumption of using a single optimal trajectory may be too strong, as human demonstrations can be sub-optimal and multi-optimal, and optimization algorithms can be very time-consuming in optimizing long-horizon trajectories.

Instead of directly optimizing a trajectory, another course is to sample trajectories, which can also be used to approximate the partition function. [25] proposed sampling a set of actions at each step in a planning cycle, resulting in a set of policies (trajectories) that encodes multiple behaviors. However, this approach only deals with static environments with only a vehicle dynamics model as the environment model, and thus more complex human driving behaviors from naturalistic driving are not captured. [75] considered generating a set of trajectories instead of sampling low-level actions and learned the cost function by minimizing the discrepancy

between expert and planned trajectories rather than the feature expectation in the general IRL setting. Our method is closely related to [76], where the authors suggested generating a trajectory set with elastic band path planning and speed profile sampler to estimate the partition function. However, these works do not focus on driving behavior modeling and lack structural assumptions about human driving behavior that can facilitate reward learning. Moreover, the interaction behaviors of the surrounding agents in the environment are not well established in these works. To address driving behavior modeling, we propose a reasonable structural assumption on human driving behavior that can seamlessly integrate into the maximum IRL framework.

Additionally, one limitation of previous studies is the assumption that all vehicles in the human driving dataset share a common cost function, which contradicts the reality that human driving behaviors are diverse and personalized. In real-world scenarios, human drivers can have distinct preferences, which entails learning reward functions that account for multiple intentions involving multiple agents [77, 78]. We focus on personalized driving behaviors with the assumption that each driver has a unique, personalized reward function.

3.3 Methodology

3.3.1 Problem formulation

Consider a human driver in an arbitrary traffic scene, the state $\mathbf{s}_t \in \mathcal{S}$ the driver observes at timestep t comprises the positions, orientations, and velocities of the driver's vehicle and the surrounding vehicles. The action $\mathbf{a}_t \in \mathcal{A}$ taken by the driver is composed of speed and steering controls of the ego vehicle. Assuming a discrete-time setup and a finite length L, a trajectory $\zeta = [\mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2, \mathbf{a}_2, \ldots, \mathbf{s}_L, \mathbf{a}_L]$ is formed by organizing the state and action at each timestep within the decision horizon. Note that the trajectory encompasses multiple vehicles in the driving scene, since we consider interactions between agents.

The state $\mathbf{s}_t$ represents a physical or partial observation that can be directly obtained from sensors. However, the latent intention, which may include attributes such as the driver's navigational goals and driving intentions, actually governs

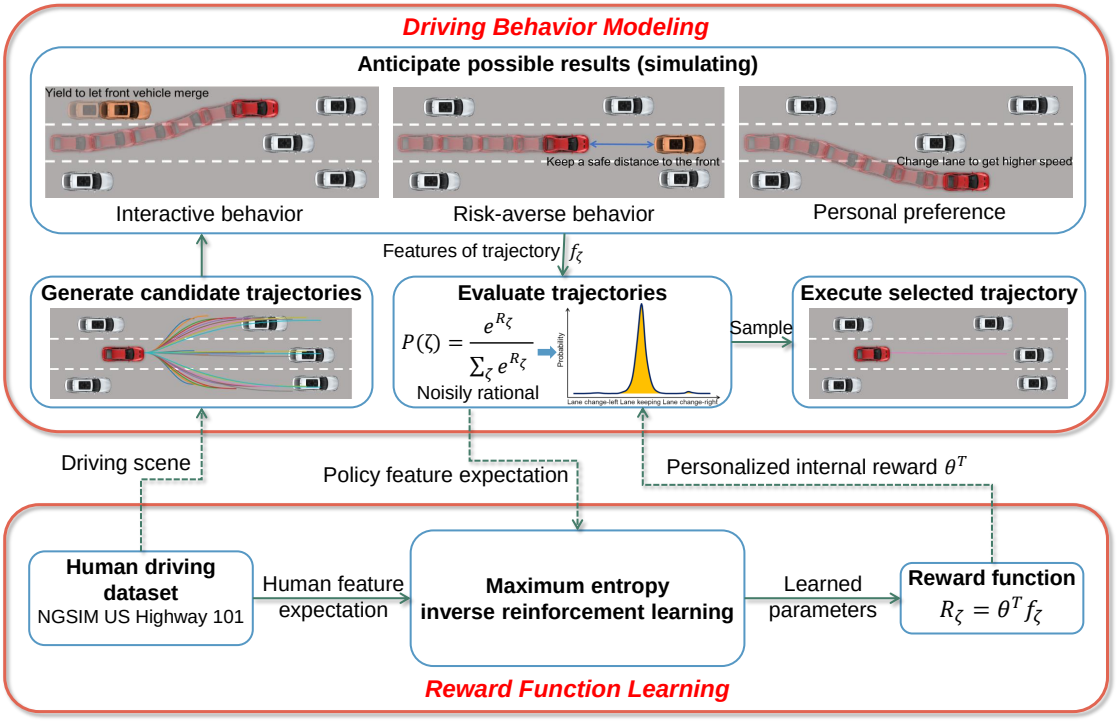


FIGURE 3.1: The framework of internal-reward-function-based driving behavior modeling with maximum entropy inverse reinforcement learning to infer the reward

the driver's actions. Consequently, we propose a structural assumption about human driving behavior focusing on high-level intentions instead of low-level control actions, as illustrated in Fig. 3.1. This assumption posits that human driving behavior consists of three processes: trajectory generation, trajectory evaluation, and trajectory selection. In a given driving scenario, a human driver initially creates multiple candidate trajectories in mind, which pertain to high-level decisions (e.g., lane-changing and lane-keeping) and speed requirements. At the same time, the driver should anticipate the outcomes of the trajectories (involving interactive and risk-averse behaviors) and evaluate the returns of different trajectories using their internal reward functions (involving personal preference). The potential plans are assigned with probabilities according to the Boltzmann noisily-rational model (i.e., the probability of a trajectory is exponential to the reward of the trajectory), and the driver finally executes one of the trajectories subjecting to the distribution. This assumption is justifiable and intuitive and can aptly explain the stochasticity of human driving behavior. Furthermore, the probabilistic setting can address the suboptimal and multi-optimal policies existing in naturalistic human driving datasets.

We assume a linear-structured reward function, which is a weighted sum of the selected features. With a focus on highway scenarios with simple road structures, where driving patterns are relatively stable and the drivers' preferences or behaviors do not change significantly over time, we assume the weights of the reward function remain consistent. Therefore, the reward function $r(\mathbf{s}_t)$ at a specific state $\mathbf{s}_t$ is defined as:

$$r(\mathbf{s}_t) = \boldsymbol{\theta}^T \mathbf{f}(\mathbf{s}_t), \tag{3.1}$$

where $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_K]$ is the K-dimensional weight vector and $\mathbf{f}(\mathbf{s}_t)$ is the extracted feature vector $\mathbf{f}(\mathbf{s}_t) = [f_1(\mathbf{s}_t), f_2(\mathbf{s}_t), \ldots, f_K(\mathbf{s}_t)]$ that characterizes the state $\mathbf{s}_t$. Therefore, the reward of a trajectory $R(\zeta)$ is given as:

$$R(\zeta) = \sum_t r(\mathbf{s}_t) = \boldsymbol{\theta}^T \mathbf{f}_\zeta = \boldsymbol{\theta}^T \sum_{\mathbf{s}_t \in \zeta} \mathbf{f}(\mathbf{s}_t), \tag{3.2}$$

where $\mathbf{f}_\zeta$ denotes the accumulative features along the trajectory ζ. The selection of features is presented in Section 3.4.2.

Formally, given the human driving demonstration dataset $\mathcal{D} = \{\zeta_1, \zeta_2, \ldots, \zeta_N\}$, consisting of N trajectories, the objective is to obtain the reward weights $\boldsymbol{\theta}$ that can generate a driving policy consistent with the human demonstration trajectories. We employ the maximum entropy IRL algorithm to infer the reward weights $\boldsymbol{\theta}$, leveraging our proposed structural assumption on human driving behavior.

3.3.2 Maximum entropy inverse reinforcement learning

In line with our assumption, a human driver follows a stochastic policy, which results in a distribution over generated candidate trajectories. We presume the distribution to be a Boltzmann distribution related to the returns of trajectories. Such a distribution also possesses maximum entropy among all distributions that match the feature expectations of expert demonstrations, which corresponds to the maximum entropy IRL [57, 73]. Formally, the probability of a trajectory is proportional to the exponential of the reward of that trajectory, as expressed by

$$P(\zeta|\boldsymbol{\theta}) = \frac{e^{R(\zeta)}}{Z(\boldsymbol{\theta})} = \frac{e^{\boldsymbol{\theta}^T \mathbf{f}_\zeta}}{Z(\boldsymbol{\theta})}, \tag{3.3}$$

where $P(\zeta|\boldsymbol{\theta})$ denotes the probability of a trajectory ζ given the reward parameter $\boldsymbol{\theta}$, and $Z(\boldsymbol{\theta})$ is the partition function.

However, the partition function $Z(\boldsymbol{\theta})$ is intractable for continuous and high dimensional spaces, because it requires integrating over the entire class of possible trajectories. Referring to our assumption, the space of possible trajectories can be reduced to several smaller subspaces. Therefore, we generate a limited number of feasible trajectories to approximate the partition function, and thus the probability of a trajectory becomes:

$$P(\zeta|\boldsymbol{\theta}) \approx \frac{e^{\boldsymbol{\theta}^T \mathbf{f}_\zeta}}{\sum_{i=1}^{M} e^{\boldsymbol{\theta}^T \mathbf{f}_{\tilde{\zeta}^i}}}, \tag{3.4}$$

where $\tilde{\zeta}^i$ is a generated trajectory that has the same initial state as ζ, $\mathbf{f}_{\tilde{\zeta}^i}$ the feature vector of the trajectory, and M the number of generated trajectories. Through this approximation, we transform $P(\zeta|\boldsymbol{\theta})$ into a probability mass, which is much easier to compute.

The goal of maximum entropy IRL is to adjust the weights $\boldsymbol{\theta}$ to maximize the likelihood of expert demonstrations under the trajectory distribution in Eq. (3.4):

$$\max_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \sum_{\zeta \in \mathcal{D}} \log P(\zeta|\boldsymbol{\theta}), \tag{3.5}$$

where $\mathcal{D} = \{\zeta_i\}_{i=1}^{N}$ denotes the trajectory set of human demonstrations.

By substituting $P(\zeta|\boldsymbol{\theta})$ in Eq. (3.5) with Eq. (3.4), we can rewrite the objective function $\mathcal{J}(\boldsymbol{\theta})$ as:

$$\mathcal{J}(\boldsymbol{\theta}) = \sum_{\zeta \in \mathcal{D}} \left[\boldsymbol{\theta}^T \mathbf{f}_\zeta - \log \sum_{i=1}^{M} e^{\boldsymbol{\theta}^T \mathbf{f}_{\tilde{\zeta}^i}} \right]. \tag{3.6}$$

Though it cannot be solved analytically, Eq. (3.6) can be optimized using a gradient-based method. The gradient of the objective function $\mathcal{J}(\boldsymbol{\theta})$ is:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \sum_{\zeta \in \mathcal{D}} \left[\mathbf{f}_\zeta - \sum_{i=1}^{M} \frac{e^{\boldsymbol{\theta}^T \mathbf{f}_{\tilde{\zeta}^i}}}{\sum_{i=1}^{M} e^{\boldsymbol{\theta}^T \mathbf{f}_{\tilde{\zeta}^i}}} \mathbf{f}_{\tilde{\zeta}^i} \right], \tag{3.7}$$

where $\mathbf{f}_\zeta$ is the feature vector of a human demonstrated trajectory, $\tilde{\zeta}^i$ is one of the generated trajectories that share the initial state of ζ, and $\mathbf{f}_{\tilde{\zeta}^i}$ is the feature vector of that trajectory.

The gradient can be viewed as the difference in feature expectations between the human demonstration trajectories and the generated ones:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \sum_{\zeta \in \mathcal{D}} \left[\mathbf{f}_{\zeta} - \sum_{i=1}^{M} P(\tilde{\zeta}^i | \boldsymbol{\theta}) \mathbf{f}_{\tilde{\zeta}^i} \right]. \tag{3.8}$$

We can employ the gradient ascent method to iteratively update the reward function until the loss converges. In practice, to prevent overfitting, we add L2 regularization on the weights into the objective function:

$$\mathcal{J}(\boldsymbol{\theta}) = \sum_{\zeta \in \mathcal{D}} \left[\boldsymbol{\theta}^T \mathbf{f}_{\zeta} - \log \sum_{i=1}^{M} e^{\boldsymbol{\theta}^T \mathbf{f}_{\tilde{\zeta}^i}} \right] - \lambda \boldsymbol{\theta}^2, \tag{3.9}$$

where $\lambda > 0$ is the regularization parameter. Thus, the gradient becomes the difference of the feature expectations along with the gradient of the regularization term:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \sum_{\zeta \in \mathcal{D}} \left[\mathbf{f}_{\zeta} - \sum_{i=1}^{M} P(\tilde{\zeta}^i | \boldsymbol{\theta}) \mathbf{f}_{\tilde{\zeta}^i} \right] - 2\lambda \boldsymbol{\theta}. \tag{3.10}$$

3.3.3 Trajectory generation

To efficiently generate feasible trajectories in a structured environment (e.g., a highway), we assume a human driver makes a short-term plan from the current state to a target state, considering both the longitudinal and lateral targets. In the longitudinal direction, the driver decides the target speed, and in the lateral direction, the driver makes a tactical decision on lane-changing and lane-keeping. Therefore, it is very convenient to use polynomial curves to represent the planned trajectories.

We use the local coordinate system, with reference to the origin and path of the road, to represent a vehicle's trajectory along the longitudinal and lateral axes. The generated trajectory should be smooth and dynamically feasible, which requires continuous acceleration and jerk over time. For the lateral y-coordinate, we need to specify the target position, velocity, and acceleration, resulting in six boundary conditions along with the initial state, which entails a quintic polynomial function. For the longitudinal x-axis, only the target velocity and acceleration are needed, and a quartic polynomial function can ensure smoothness. Consequently,

the trajectory is represented by two polynomial functions (continuous functions of time) with respect to x and y coordinates:

$$\begin{cases} \mathbf{x}(\tau) = a_0 + a_1\tau + a_2\tau^2 + a_3\tau^3 + a_4\tau^4 \\ \mathbf{y}(\tau) = b_0 + b_1\tau + b_2\tau^2 + b_3\tau^3 + b_4\tau^4 + b_5\tau^5 \end{cases}, \tag{3.11}$$

where τ is the time, and $\{a_0, \cdots, a_4\}$ and $\{b_0, \cdots, b_5\}$ are the coefficients of the polynomial functions.

Given the initial state of the ego vehicle and the target state, as well as the required time T to reach the target, the boundary conditions of the polynomial functions on the longitudinal and lateral axis are:

$$\begin{cases} \mathbf{x}(\tau = 0) = x_s \\ \dot{\mathbf{x}}(\tau = 0) = v_{xs} \\ \ddot{\mathbf{x}}(\tau = 0) = a_{xs} \\ \dot{\mathbf{x}}(\tau = T) = v_{xe} \\ \ddot{\mathbf{x}}(\tau = T) = a_{xe} \end{cases}, \begin{cases} \mathbf{y}(\tau = 0) = y_s \\ \dot{\mathbf{y}}(\tau = 0) = v_{ys} \\ \ddot{\mathbf{y}}(\tau = 0) = a_{ys} \\ \mathbf{y}(\tau = T) = y_e \\ \dot{\mathbf{y}}(\tau = T) = v_{ye} \\ \ddot{\mathbf{y}}(\tau = T) = a_{ye} \end{cases}, \tag{3.12}$$

where $(x_s, v_{xs}, a_{xs}, y_s, v_{ys}, a_{ys})$ is the start state ($\tau = 0$) including position, velocity, and acceleration in the longitudinal and lateral directions; $(v_{xe}, a_{xe}, y_e, v_{ye}, a_{ye})$ is the target state ($\tau = T$).

By solving the boundary equations, we can determine the coefficients of the polynomial functions, and thus generate a trajectory. The position, velocity, and acceleration of the ego vehicle on the trajectory can be derived for any given time τ ($\tau \leq T$), and we set the horizon to 5 seconds ($T = 5\ s$). Given the instants (with a time interval of 0.1 seconds) at which the acceleration, velocity, and position values are computed, a polynomial trajectory can be generated, which is a sequence of these values aligning with the timesteps of the human driving trajectories.

We can generate multiple polynomial trajectories by sampling the target states from the target space $\mathbf{\Phi} = \{v_{xe}, a_{xe}, y_e, v_{ye}, a_{ye}\}$, which covers a range of possible maneuvers. Fig. 3.2 shows an example of the trajectory generation process, in which only v_{xe} and y_e are variable while others are constant to 0. In Fig. 3.2, multiple candidate trajectories are generated, covering decisions on lane-changing and lane-keeping as well as the desired longitudinal speed.

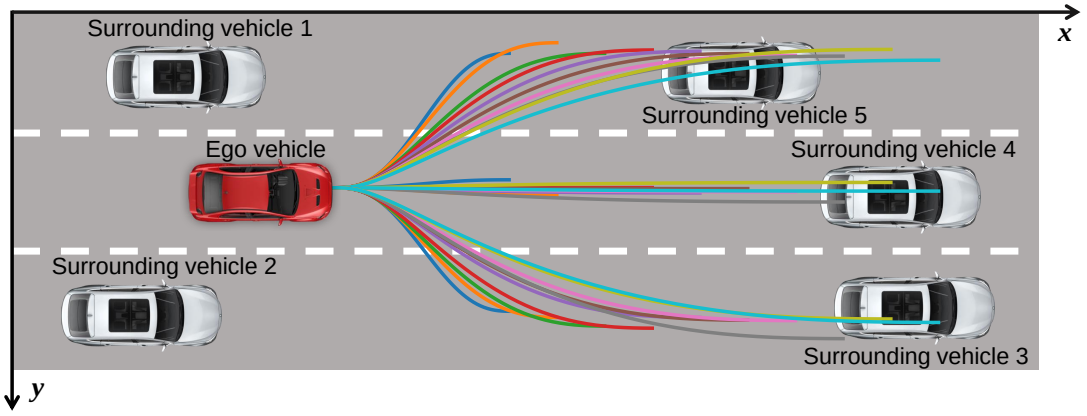


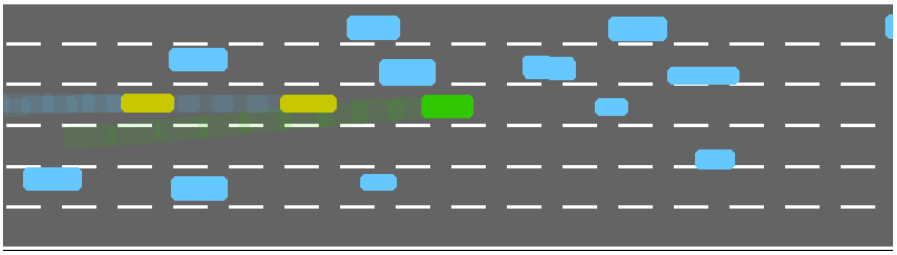
FIGURE 3.2: Trajectory generation process considering the longitudinal and lateral targets

3.3.4 Environment model

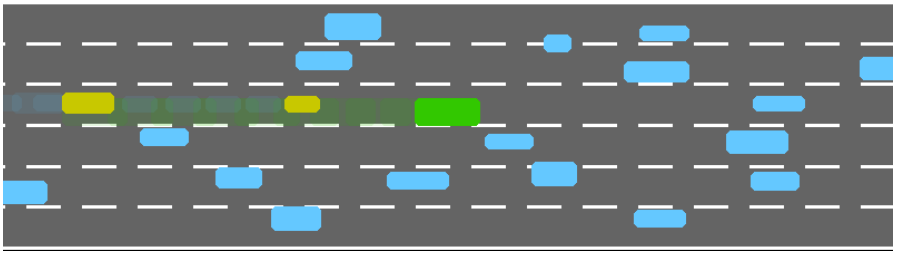
To simulate the outcomes of the generated trajectories and the reactions of other agents to the ego's actions, particularly those that deviate from the ground truth, an environment model is essential. The model serves as a simulated mental world of human drivers, helping to anticipate other agents' reactions to the ego movements and estimate the reward of the generated trajectory. We first construct a multi-lane highway road with the same structure as the study area in the NGSIM US-101 dataset, which consists of five mainline lanes throughout the section and an auxiliary lane between the on-ramp and the off-ramp. More details about the road structure can be found in Section 3.4.1. Then, we spawn vehicles on the road as recorded in the dataset at a specific instant. One of these vehicles is designated as the observation object and controlled by a pure-pursuit controller to track the generated trajectory, ensuring the final trajectory is dynamically feasible. The kinematic state of the vehicle is propagated according to the bicycle model.

Next, we predict the future trajectories of the surrounding vehicles in response to the ego vehicle's actions. The general idea is that the surrounding vehicles follow their original trajectories from the dataset and react by maintaining a safe distance from the ego vehicle or the influenced vehicle. The underlying assumption is that humans are best-response agents who can accurately anticipate other agents' reactions to their planned actions. It is worth noting that this setting may introduce some bias in estimating the transition function. Specifically, only vehicles within a 50-meter range of the ego vehicle in the environment are considered. Surrounding vehicles will initially follow their recorded trajectories and continually check the

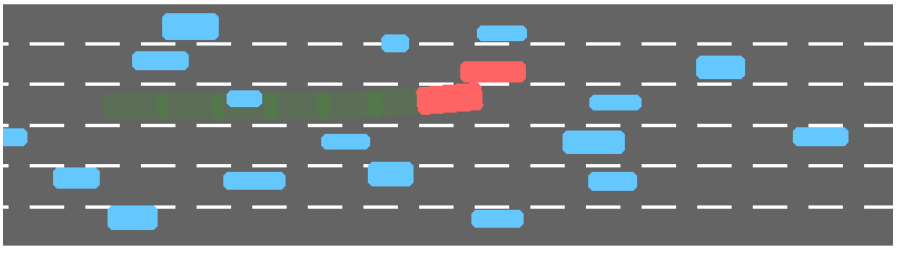
gap between themselves and the vehicle in front. If the front is the ego vehicle and the gap between them is smaller than the desired gap determined by IDM, the environment vehicle will be overridden by IDM and will no longer follow its original trajectory. Similarly, if the front vehicle is an environment vehicle overridden by IDM, the environment vehicle behind it will also be overridden if the gap between them is too small. IDM is a parametric car-following model for highway traffic simulation that models the driver's desire to achieve the target speed and maintain a safe distance from the vehicle in front. More details about IDM can be found in [59]. We only consider the longitudinal responses of surrounding vehicles to simulate the influence of the ego vehicle's change-of-course behavior. Using these simple rules, multiple surrounding vehicles can be sequentially affected by the ego vehicle's decisions.



(A)



(B)



(C)

FIGURE 3.3: Illustrations of interactive behaviors in the environment model

Fig. 3.3 shows some exemplar scenarios where the green vehicle is the ego vehicle, blue vehicles are surrounding vehicles following their original trajectories, yellow ones are vehicles affected by the ego vehicle's actions and overridden by IDM, and

red vehicles indicate collisions. Fig. 3.3(a): the ego vehicle tries to change to the right lane, which makes the affected vehicles decelerate to yield; Fig. 3.3(b) the ego vehicle runs too slow, which makes the rear vehicles decelerate to avoid a collision; Fig. 3.3(c) the ego vehicle's generated trajectory causes a collision.

3.3.5 Summary of the IRL algorithm

The algorithm of maximum entropy IRL with trajectory sampling is summarized in Algorithm 1. We first initialize the reward parameters randomly and compute the feature expectation of human driving trajectories. Given that the sampling process requires the majority of the computation time, a buffer is created to store the feature vectors of all generated trajectories, thereby avoiding iterative sampling in the environment. For each driving scene provided by the demonstration data, a set of trajectories is generated and rolled out in the environment model to obtain the feature vectors. The size of generated trajectory set $\tilde{\mathcal{D}}_i$ for a single driving scene is determined by the size of the target sampling space (i.e., the product of the number of longitudinal targets and the number of lateral targets). Upon completing the sampling process and obtaining the buffer, the gradient can be calculated, and the gradient ascent method is employed to iteratively update the reward parameters. This process ensures that the feature expectation of the generated trajectories matches that of the human trajectories.

3.4 Experiments

3.4.1 Naturalistic human driving dataset

To validate the proposed method for driving behavior modeling, we employ the Next Generation Simulation (NGSIM) dataset [79], focusing on a segment of data recorded between 7:50 a.m. and 8:05 a.m. on US Highway 101. The recording area is a section of the highway with approximately 640 meters (2,100 feet) in length, consisting of five main lanes throughout the section and an auxiliary lane situated between an on-ramp and an off-ramp, as illustrated in Fig. 3.4(a). In addition to the global positions, the dataset also provides local positions of vehicles with respect to the local coordinate system. Based on this data, we reconstruct the

Algorithm 1: Maximum entropy IRL with trajectory sampling

Input : Human demonstration trajectory dataset $\mathcal{D} = \{\zeta_i\}_{i=1}^{N}$, environment model P, learning rate α, regularization parameter λ, number of epochs E

Output: Optimized reward function parameters $\boldsymbol{\theta}^*$

Initialize $\boldsymbol{\theta} \leftarrow \mathcal{N}(0, 0.05)$;
Compute human feature expectation $\overline{\mathbf{f}} \leftarrow \sum_{i=1}^{N} \mathbf{f}_{\zeta_i}$;
Initialize buffer $\mathcal{B} \leftarrow [\,]$;
foreach ζ_i *in* $\mathcal{D}$ **do**
  Determine the sampling space $\boldsymbol{\Phi}$ and planning horizon T;
  Generate a trajectory set $\tilde{\mathcal{D}}_i = \{\tilde{\zeta}_i^j\}$ with the same initial state as ζ_i according to the sampling space $\boldsymbol{\Phi}$ and horizon T;
  foreach $\tilde{\zeta}_i^j$ *in* $\tilde{\mathcal{D}}_i$ **do**
    Rollout the trajectory $\tilde{\zeta}_i^j$ in the environment model P and calculate the feature vector of the trajectory $\mathbf{f}_{\tilde{\zeta}_i^j}$;
    Add trajectory and its feature vector to buffer $\mathcal{B} \xleftarrow{+} \tilde{\zeta}_i^j,\ \mathbf{f}_{\tilde{\zeta}_i^j}$;
  end
end
for *epoch* $\leftarrow$ *1 to* E **do**
  Calculate the feature expectation with the collected samples from $\mathcal{B}$:
  $\tilde{\mathbf{f}} \leftarrow \sum_{i=1}^{N} \sum_j \frac{\exp\left(\boldsymbol{\theta}^T \mathbf{f}_{\tilde{\zeta}_i^j}\right)}{\sum_j \exp\left(\boldsymbol{\theta}^T \mathbf{f}_{\tilde{\zeta}_i^j}\right)} \mathbf{f}_{\tilde{\zeta}_i^j}$;
  Calculate the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) \leftarrow \overline{\mathbf{f}} - \tilde{\mathbf{f}} - 2\lambda\boldsymbol{\theta}$;
  Update reward parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$;
end
$\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$

road structure in our environment model, as shown in Fig. 3.4(b). The coordinate system's origin is positioned at the top-left corner vertex of the study area. The longitudinal x axis extends along the road, while the lateral y axis is perpendicular to the road's direction. The road spans 640 meters and includes five main lanes (Lanes 1 to 5), each with a width of 3.66 meters. The on-ramp (Lane 7) and off-ramp (Lane 8), as well as the auxiliary lane between them (Lane 6), are also considered but ramp merging scenarios are not the focus. The locations of each vehicle are recorded at a rate of 10 frames per second, resulting in detailed vehicle trajectories from the area's beginning to its end. However, the originally collected vehicle trajectories in the dataset are full of observation noise. To address this issue, we employ the Savitzky-Golay filter with a third-order polynomial over 2-second windows to smooth the original trajectories and obtain the demonstration

trajectories for reward learning. Fig. 3.4(c) displays the processed trajectories of 300 randomly selected vehicles in the dataset as examples, and speeds and accelerations can be estimated from position data. The naturalistic human driving dataset encompasses the trajectories of nearly 3000 vehicles and provides abundant information on interactions between human drivers, which is essential for our study to uncover the diverse, personalized, and highly interactive human driving behaviors.

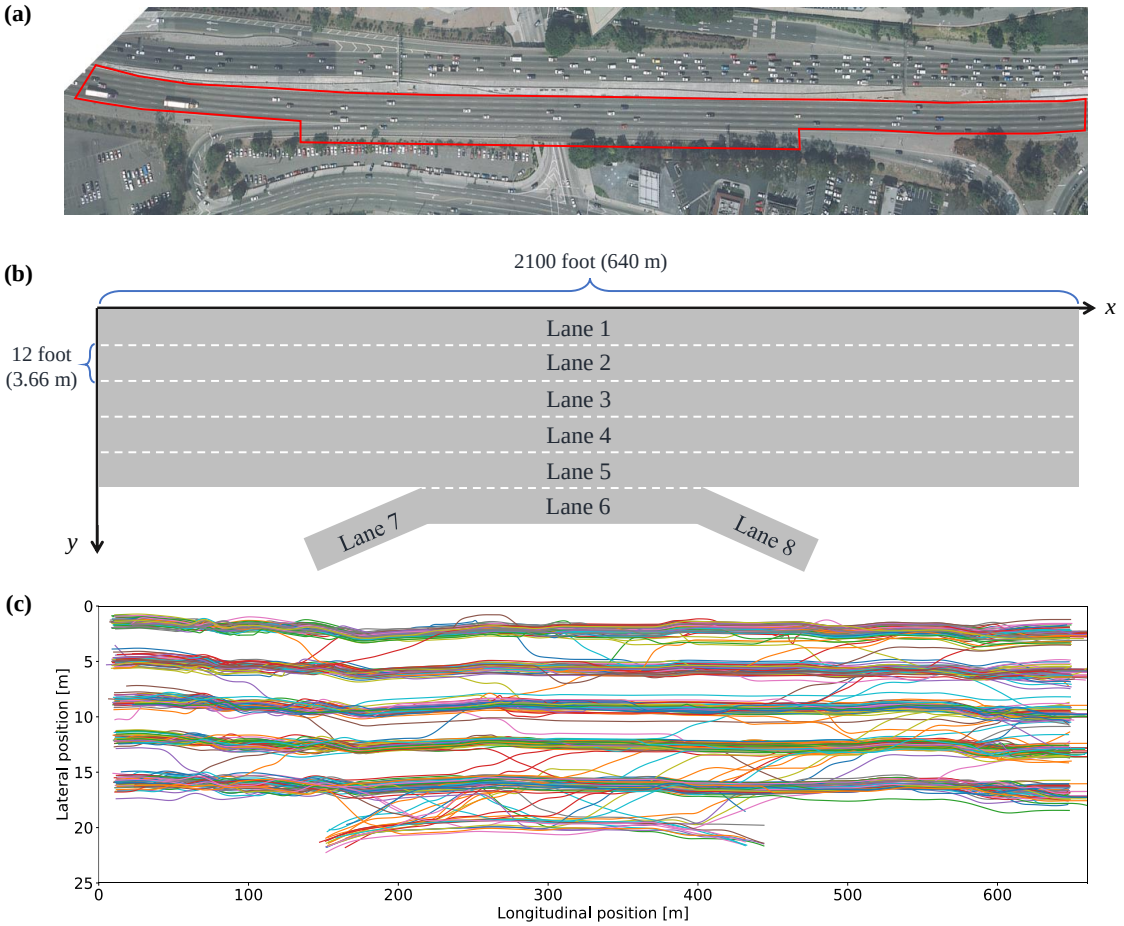


FIGURE 3.4: Illustrations of the dataset, simulated road structure, and processed vehicle trajectories

3.4.2 Feature selection

Features are mappings from states to real values that capture important properties of the state. In this study, we group the features of the driving state into the following four main aspects that are important to human drivers [80].

Travel efficiency. This feature is designed to reflect the human's desire to reach the destination as quickly as possible, defined as the speed of the vehicle:

$$f_v(\mathbf{s}_t) = v(t). \tag{3.13}$$

Comfort. Ride comfort is another factor that human drivers prefer, and the metrics to gauge comfort are longitudinal acceleration a_x, lateral acceleration a_y, and longitudinal jerk j_x:

$$\begin{cases} f_{a_x}(\mathbf{s}_t) = |a_x(t)| = |\ddot{x}(t)| \\ f_{a_y}(\mathbf{s}_t) = |a_y(t)| = |\ddot{y}(t)| \\ f_{j_x}(\mathbf{s}_t) = |\dot{a}_x(t)| = |\dddot{x}(t)| \end{cases}, \tag{3.14}$$

where $x(t)$ and $y(t)$ represent the longitudinal and lateral coordinates, respectively.

Risk aversion. Human drivers tend to maintain a safe distance from the surrounding vehicles and this distance varies across different human drivers, which reflects their individual levels of risk perception. We define the risk level to the front vehicle as an exponential function related to the time headway from the ego vehicle to the front vehicle, assuming a constant speed movement:

$$f_{risk_f}(\mathbf{s}_t) = e^{-\left(\frac{x_f(t)-x_{ego}(t)}{v_{ego}(t)}\right)}, \tag{3.15}$$

where $x_f(t)$ is the longitudinal position of the nearest front vehicle, $x_{ego}(t)$ is that of the ego vehicle, and $v_{ego}(t)$ is the speed of the ego vehicle.

Similarly, the risk level to the rear end is defined as an exponential function related to the time headway from the rear vehicle to the ego vehicle:

$$f_{risk_r}(\mathbf{s}_t) = e^{-\left(\frac{x_{ego}(t)-x_r(t)}{v_r(t)}\right)}, \tag{3.16}$$

where $x_r(t)$ and $v_r(t)$ are the longitudinal position and speed of the nearest rear vehicle, respectively.

It should be noted that collisions may occur when evaluating the generated trajectories in our environment model, including collisions with other vehicles or road

curbs. Consequently, collision is also considered a risk indicator, defined as:

$$f_{collision}(\mathbf{s}_t) = \begin{cases} 1, & \text{if collision,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.17}$$

Interaction. A fundamental property of human driving behavior is that humans are aware of the influence of their actions on the surrounding vehicles, or more specifically, whether their plans would impose additional inconvenience on others (e.g., sharp decelerate to yield) [81]. We introduce the following feature to explicitly represent such influence. It is defined as the sum of predicted decelerations of the environment vehicles that have been affected by the ego vehicle's behavior according to our environment model (chain deceleration reactions of the following vehicles), indicating that the ego vehicle's change of the original course has caused direct influence on them.

$$f_I(\mathbf{s}_t) = \sum_i |a_i(t)|, \text{ if } a_i(t) < 0, \tag{3.18}$$

where $a_i(t)$ is the acceleration of vehicle i that has been influenced by the ego vehicle. When applying the reward function in real-world scenarios, we can estimate this feature with a prediction module that forecasts other agents' actions due to the planned actions using driving models such as IDM.

All the above features are calculated at every timestep and accumulated over time to obtain the feature of the trajectory. The trajectory features are then normalized to the range $[0, 1]$ by dividing by the maximum value in the dataset to cancel out the influence of their different units and scales. Additionally, we assign a fixed large negative weight (-10) to the collision feature, because this could improve the modeling accuracy compared to making this weight learnable.

3.4.3 Experiment design

- **Driving behavior analysis**. We utilize the proposed method to analyze the driving behaviors of different human drivers. We first demonstrate the reward learning process of a human driver from the dataset as an example to highlight the effectiveness of our proposed method. Then, the learned reward

function is used to determine the probabilities of the candidate trajectories in testing conditions and to interpret some driving behaviors.

- **Robustness**. We test the learned reward functions in scenes that are not included in the training phase to find out if there is a significant drop in the similarity between the learned and human policies. This assessment investigates the robustness of the proposed method.

- **Modeling accuracy**. We present the quantitative results of modeling accuracy in the testing conditions by comparing the learned policy to the ground-truth human driving trajectory. We investigate the personalized modeling assumption that each human driver has different preferences (driving styles), resulting in different weights over the reward function. The general modeling assumption that all drivers share an identical cost function is adopted as a comparison. Two other baseline models are also employed: IDM and MOBIL for longitudinal and lateral movement, respectively, and the constant velocity model.

- **Interaction factors**. We analyze the effects of interaction factors on the modeling accuracy, including the interaction feature in the reward function and simulating the reactions of surrounding vehicles to the change of course of the ego vehicle in the environment model.

3.4.4 Implementation details

For simplicity, the target sampling space is reduced to $\mathbf{\Phi} = \{v_{xe}, y_e\}$, in which only the longitudinal speed and lateral position are variables and other targets are set to 0. The sampling range of the longitudinal speed is $[v-5, v+5]$ m/s with an interval of 1 m/s, where v is the initial speed of the vehicle. The sampling set of the lateral position is $\{y, y_L, y_R\}$ m, where y is the initial lateral position, and y_L and y_R are the positions of the left lane and right lane, respectively, if available. The planning horizon is 5 s and the simulation interval is 0.1 s. The parameters of IDM are: desired velocity $v_0 = v_{current}$ m/s, maximum acceleration $a_{max} = 5$ m/s^2, desired time gap $\tau = 1$ s, comfortable braking deceleration $b = 3$ m/s^2, minimum distance $s_0 = 1$ m. A problem arises as the longitudinal and lateral jerk of human trajectories and generated trajectories can hardly match, because the

polynomial curves are smooth while the human driving trajectories exhibit noisy movements. Therefore, we process the human driving trajectory to be represented by a polynomial curve, given the initial state and end condition of the original trajectory.

To stabilize the training process, we implement the Adam optimizer instead of the vanilla gradient ascent method. Three hyperparameters require tuning: the regularization parameter λ, the learning rate α, and the number of training epochs E. We use the grid search method with the parameter space as $\lambda \in \{0.1, 0.01, 0.001\}$, $\alpha \in \{0.1, 0.05, 0.01\}$, and $E \in \{100, 200, 300\}$, and the performance metric as the average likelihood of demonstration trajectories from 10 drivers. The final setting of the hyperparameters is $\lambda = 0.01, \alpha = 0.05, E = 200$. Completing the learning process with 35 human demonstration trajectories from a vehicle takes approximately 18 minutes using an AMD Ryzen 3900X CPU. In the testing phase, it takes about 30 seconds to implement a sampling and evaluation process (with the size of sampled trajectories as 30). It is important to note that parallel sampling is not used, as real-time planning is not the focus of this paper, but it could significantly speed up the learning and testing processes.

3.5 Results and Discussions

3.5.1 Driving behavior analysis

For a vehicle in the dataset, its original trajectory throughout the highway section, which spans approximately 50 to 70 seconds in duration, is evenly partitioned into 50 short-term trajectories, each lasting 5 s. Each trajectory represents a driving scene involving different situations and various interactions with the surrounding vehicles. Among these trajectories, 35 are randomly selected and used as the training data for reward function learning. The remaining 15 trajectories serve as testing conditions, where the learned reward function is applied to select the candidate trajectories. An example of the training process is shown in Fig. 3.5, which plots the curves of average feature difference (L2 norm) between the learned policy and human driver, average log-likelihood of the human demonstrated trajectories, and average human likeness. Human likeness is a custom metric designed to gauge the accuracy of the model, i.e., closeness to the ground-truth human

driving behavior. Since our model is probabilistic, human likeness is defined as the minimal final displacement error of the three trajectories with the highest probabilities in the distribution over generated trajectories. It is calculated as the L2 norm between the position at the end of the ground truth trajectory and that of the closest prediction among the three most likely trajectories. Formally, $HL = \min\{\left\|\hat{\zeta}_i(L) - \zeta_{gt}(L)\right\|_2\}_{i=1}^{3}$, where $\hat{\zeta}_i$ $(i = 1, 2, 3)$ are the selected trajectories with the highest probabilities, ζ_{gt} is the ground-truth trajectory driven by the human driver, and L is the end of the time horizon. Therefore, smaller human likeness values indicate better modeling accuracy.

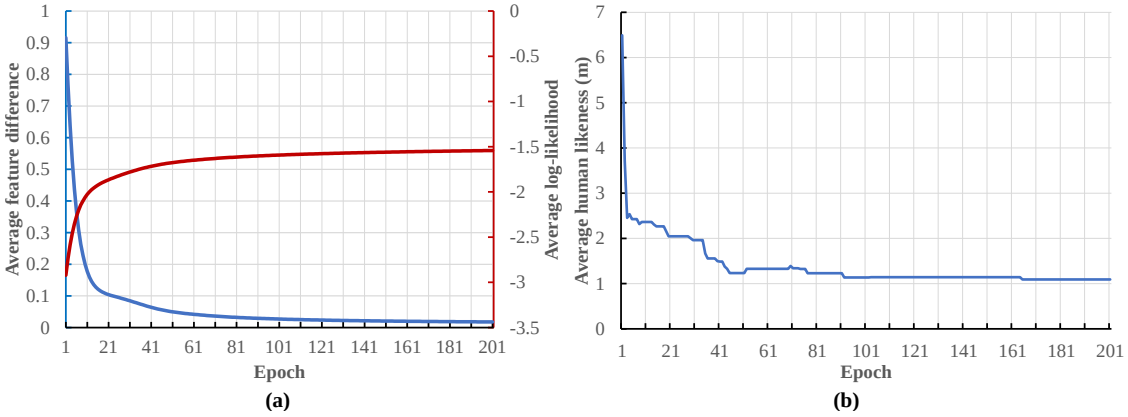


FIGURE 3.5: Example of the training process: plots of the average feature difference, log-likelihood, and average human likeness

As illustrated in Fig. 3.5(a), the average log-likelihood of the human demonstrated trajectories gradually increases and converges, recalling that the goal of the maximum entropy IRL is to maximize the likelihood of human demonstrations. Meanwhile, the average feature difference between the learned policy and human driver consistently decreases to a small value. This leads to the decrease in human likeness, as shown in Fig. 3.5(b), which implies that the probability of choosing the trajectories close to human driving behavior increases under the learned reward function. These results validate the effectiveness of the proposed maximum entropy IRL algorithm with trajectory sampling.

We select various human drivers and associated driving trajectories from the dataset and apply the proposed method to infer their individual reward functions, and subsequently use the learned reward to interpret their decisions. Fig. 3.6 shows some representative cases of different vehicles from the US-101 highway dataset. The candidate trajectories and their associated probabilities and the ground-truth

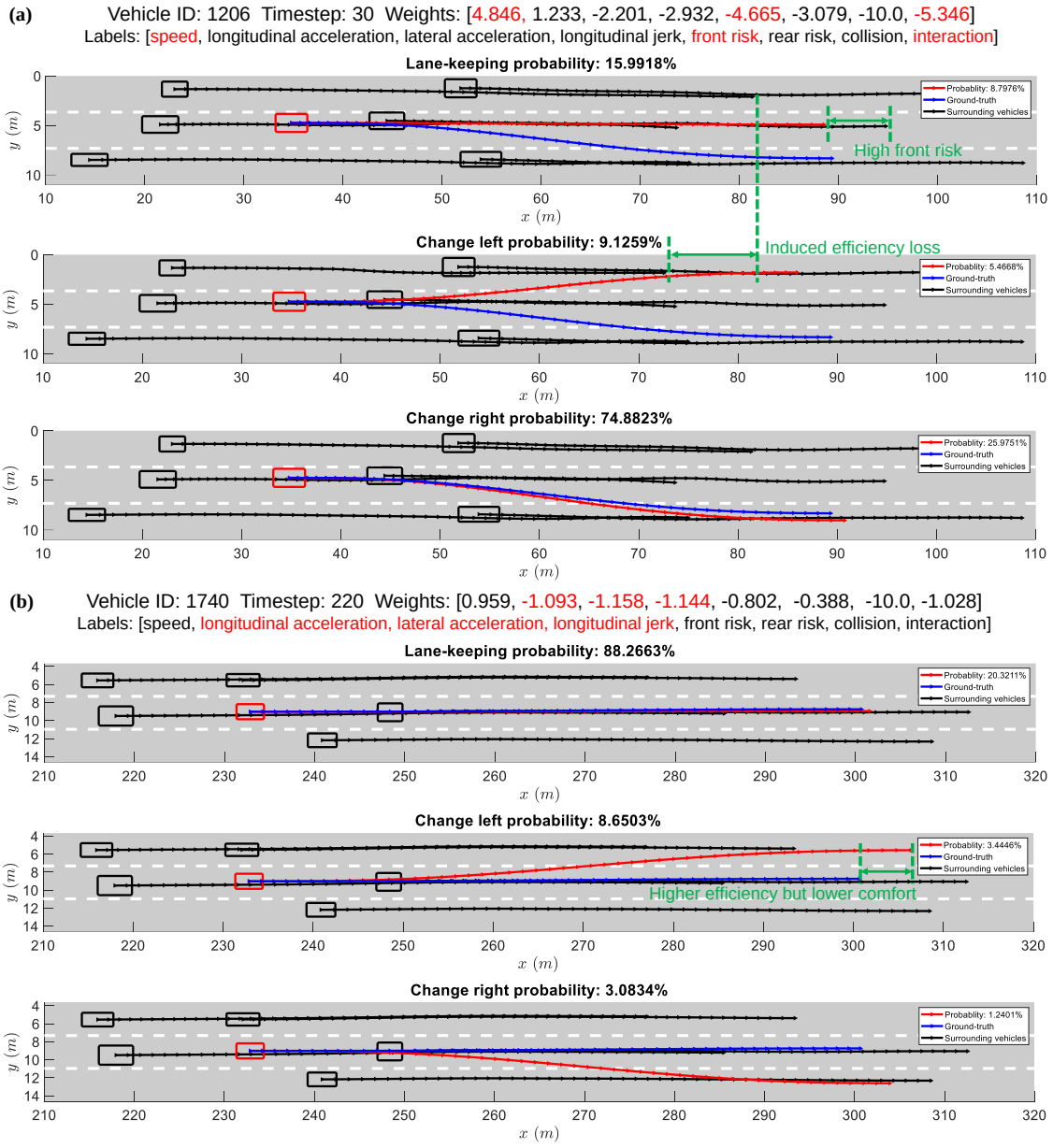


FIGURE 3.6: Driving behavior analysis of some typical cases from the US 101 highway dataset

human driving trajectories are displayed, along with the trajectories of the surrounding vehicles as the interaction context. Only the most likely trajectory in the three discrete lateral decision groups (lane-keeping, change left, and change right) is displayed in Fig. 3.6. Generally, minimizing the risk to both the front end and rear end is a critical factor shared by most human drivers, while the importance of other factors (speed, ride comfort, and interaction) varies among different drivers. Fig. 3.6(a) shows an overtaking scenario in which the human driver, as represented

by the recovered reward function, prioritizes speed over ride comfort (both longitudinally and laterally). If the driver remains in the current lane and wants to keep the speed, the distance to the front vehicle would be shorter, which is not likely to happen since it would result in higher front risk. Furthermore, if the driver chooses to change to the left lane, a significant speed loss can be imposed on the rear vehicle, which is an undesirable outcome as the driver is averse to influencing others, given a higher weight on the interaction term. Therefore, the driver would choose to change to the right lane to overtake, as predicted by our model with a probability of nearly 75%. A detailed trajectory is also provided, which closely matches the ground-truth human driving trajectory. This example signifies that our model can accurately predict lane-changing behavior and generate detailed trajectories. In another instance, as shown in Fig. 3.6(b), where the human driver regards ride comfort (both longitudinally and laterally) as the main concern, the driver would maintain the current lane to avoid acceleration and jerk, even if changing to the left lane could yield higher efficiency. This decision is predicted by the model with a probability of 88%.

3.5.2 Testing of accuracy and robustness

We randomly select 100 vehicles from the dataset, with 50 vehicles experiencing lane changes and the rest only performing lane-keeping, as the target objects. For the personalized modeling assumption that each driver has a unique cost function, the proposed IRL method is applied to infer their individual reward functions. For each individual vehicle, we examine the robustness of the learned reward function in the on-hold 15 driving scenes, which are different from the training conditions. For the general modeling assumption, a total of 150 trajectories from 20 vehicles are used to learn a general cost function, which is assumed to be shared by all human drivers. The learned cost function is utilized to select the candidate trajectories in the same testing conditions as the personalized modeling method and compared against the ground-truth human driving trajectories to investigate the robustness of the learned reward function. The results of the robustness testing are presented in Fig. 3.7(a). To evaluate the accuracy of the proposed reward function modeling method, two other models are selected as comparison baselines, i.e., the IDM and MOBIL for longitudinal and lateral behaviors, respectively, and the constant velocity model. For IDM, the tuned parameters are: maximum acceleration

$a_{max} = 1.3\ m/s^2$, desired time gap $\tau = 1.2\ s$, comfortable braking deceleration $b = 0.7\ m/s^2$, and minimum distance $s_0 = 1.5\ m$; the parameters for MOBIL are: safe deceleration limitation $b_{safe} = 2\ m/s^2$, politeness factor $p = 0.01$, and lane-changing decision threshold $a_{th} = 0.2\ m/s^2$. These models are applied in the same testing conditions as the reward function modeling method, and only one trajectory is generated, as they are deterministic methods. The metric to quantify the modeling accuracy is the average human likeness on trajectories. The results of the modeling accuracy of different models are shown in Fig. 3.7(b), in which the boxplots display the summary of the 100 different vehicles in the testing conditions.

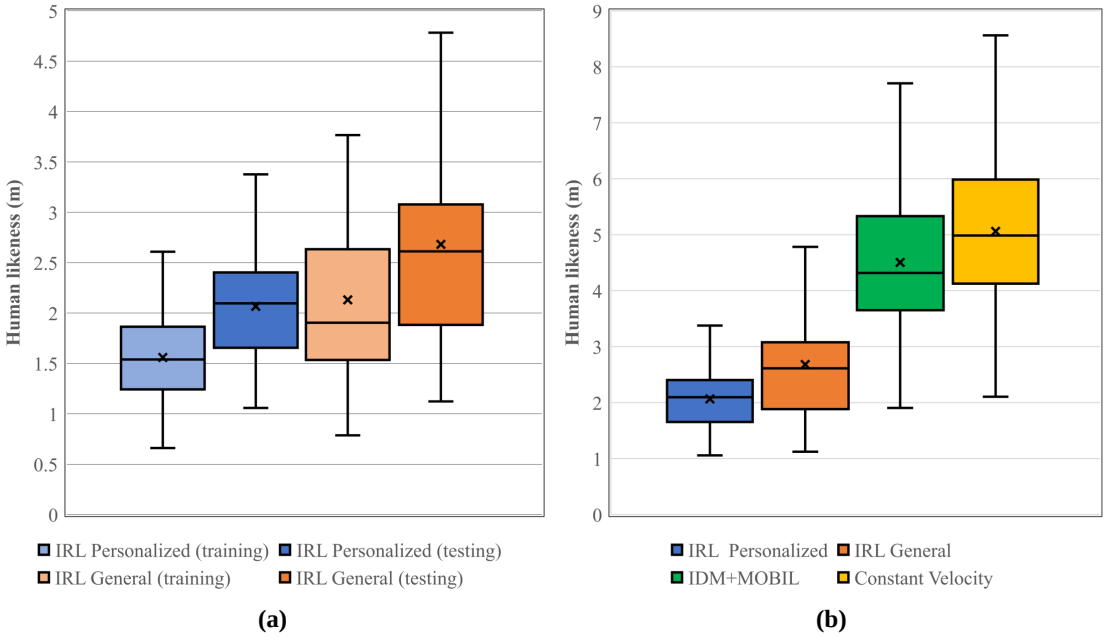


FIGURE 3.7: Comparison of robustness and modeling accuracy of different models

The results in Fig. 3.7(a) indicate that the learned reward functions show acceptable robustness in the testing conditions. There is only a slight deterioration in human likeness in the testing conditions, suggesting that the learned reward function is robust in selecting the candidate trajectories close to human driving ones in the untrained conditions. The personalized reward modeling method outperforms the general modeling method in terms of robustness, as the general modeling method shows lower mean accuracy and higher variance in the testing conditions. Fig. 3.7(b) reveals that personalized modeling more closely resembles human driving behavior and thus demonstrates smaller errors compared to the ground-truth trajectories ($Mean = 2.066\ m$). A notable reduction in human likeness is observed

when turning to general modeling ($Mean = 2.681\ m$), but it still performs significantly better than the IDM+MOBIL model ($Mean = 4.504\ m$) and the constant velocity model ($Mean = 4.986\ m$). Additionally, the difference in human likeness between personalized modeling and general modeling is found to be statistically significant according to the t-test ($p < 0.001$). These findings suggest that a general reward function can encode the basic requirements and common preferences of human driving behaviors, whereas the personalized reward function is capable of expressing diverse human driving preferences and thereby achieves better performance in fitting personalized driving behavior.

3.5.3 Effects of interaction factors

We have considered the interaction factors among human drivers in this study, which are reflected in the speed loss on other vehicles caused by the ego vehicle's movement and the modeling of behaviors of surrounding vehicles. We now investigate how these two interaction factors could affect the modeling accuracy of human driving behaviors in terms of training performance and generalization capability. The former aspect is achieved by removing the interaction feature from the reward function, while the latter is to maintain the original trajectories of the surrounding vehicles instead of overriding them with reactive behaviors. The same 100 target vehicles in the previous subsection are selected, and the results are shown in Table 3.1. The metrics, human likeness and training log-likelihood, are averaged first by the trajectories of a vehicle and then by different vehicles.

TABLE 3.1: Comparison of training and testing performance for personalized modeling with regard to interaction factors

Method	Human likeness (training) [m]	Log-likelihood (training)	Human likeness (testing) [m]
Proposed	1.572	-2.062	**2.066**
W/o interaction awareness	1.708	-2.145	2.199
W/o reactive response	**1.533**	**-2.016**	2.145

As evident from Table 3.1, removing the interaction awareness in the reward function impairs modeling accuracy, which suggests the importance of interaction or courtesy factors in modeling naturalistic human driving behaviors. Another finding is that although not simulating the reactive behaviors of surrounding vehicles may

produce better training performance in terms of both human likeness and likelihood, its generalization ability is compromised, and testing performance is worse than the proposed method. The issue is possibly caused by the biased estimation of the partition function. For the sampling-based IRL method, generating accurate and plausible samples that agree with realistic human behaviors is crucial to approximate the partition function. If all other vehicles follow their fixed paths in the environment model, some sampled trajectories become less likely, as they may cause a crash or become too risky. However, those trajectories are still possible in real-world settings because other drivers can adapt to the ego vehicle's actions, and thus, the stochasticity of human driving behavior is ignored. This could underestimate some sampled trajectories when approximating the partition function, and thus bias the estimation to fit the training data and consequently leading to compromised generalization ability. Therefore, it is reasonable to simulate other vehicles' responses to the sampled trajectories to approximate the partition function and learn the parameters of the cost function more accurately.

For the general modeling assumption, a total of 150 trajectories from 20 vehicles are selected as training data to learn a general reward function. The learned cost function is then utilized to select the candidate trajectories in the same testing conditions as the personalized modeling method. The human likeness and log-likelihood of the training process are displayed in Table 3.2. Note that the training log-likelihood is averaged over all training trajectories.

TABLE 3.2: Comparison of training and testing performance for general modeling with regard to interaction factor

Method	Human likeness (training) [m]	Log-likelihood (training)	Human likeness (testing) [m]
Proposed	2.231	-2.411	**2.681**
W/o interaction awareness	2.473	-2.474	3.410
W/o reactive response	**2.161**	**-2.381**	3.174

The findings in Table 3.2 are consistent with those in Table 3.1, suggesting that ignoring interaction awareness would lower the modeling accuracy and likelihood of human demonstrations both in training and testing. Not simulating the responses of other vehicles could produce better training performance but undermine the generalization capability.

TABLE 3.3: Evaluation of the learned reward function with the forecasting model in planning

Method	Human likeness (training) [m]	Log-likelihood (training)	Human likeness (testing) [m]
Proposed (personalized)	–	–	**2.316**
Proposed (general)	–	–	2.722
W/ forecasting model (personalized)	1.938	-2.181	2.354
W/ forecasting model (general)	2.266	-2.429	3.158

Moreover, we now consider another setting where the ground truth of the future behavior of the surrounding vehicles is not available and use IDM and MOBIL models to forecast the trajectories of the nearby vehicles without relying on the log-replay data. First, we use the same training set to learn the reward function with the forecasting model instead of log-replay data to investigate the effect of the environment model. Second, using the same testing set, we test the previously learned reward function for trajectory planning along with the forecasting model in an open-loop manner, i.e., comparing the human-likeness of the planned (generated) trajectories and ground truth human driving trajectories. This reflects how the learned reward functions can be used in planning and decision-making processes in real-world application scenarios. For calculating the interaction feature, the influenced surrounding vehicles will be the ones following behind the ego vehicle in the same lane, which can also react sequentially to the action of the ego vehicle. The results are given in Table 3.3, which reveal that the human-likeness of applying the personalized reward function degrades in this scenario but still outperforms that of using the general reward function. This is primarily due to the inaccurate forecasting of the surrounding vehicles and thus the inaccurate estimation of the risk features, which often have higher weights for most individuals. However, the general reward function is shown to be more robust against the change of the environment model, as the human-likeness metric remains at nearly the same level. It indicates that the accuracy of using the personalized reward function in the planning module is sensitive to the accuracy of the forecasting model, while the general reward function is more robust to the accuracy of the forecasting model. Additionally, using only the forecasting model instead of log-replay data to learn the reward function would result in a decline in accuracy during training but could still deliver similar performance in testing.

3.5.4 Discussions

The primary application of the proposed driving behavior modeling method is in the planning and decision-making modules of AVs to provide personalized driving experiences. It is promising to learn a personalized cost function from naturalistic human driving data offline and integrate the learned cost function into the trajectory planning module, ultimately delivering a personalized driving experience. Another application lies in predicting the motion of surrounding vehicles. The reward functions of other vehicles can be inferred online through an offline dataset containing a distribution of cost functions for different driving styles [82] or even acquired via vehicle-to-vehicle communications. Due to the mutual influence of agents, the trajectories of all interacting vehicles should be predicted. Although this would considerably increase the computation time, the prediction process can be accelerated by reducing the sampling space or the number of target vehicles and parallelizing the sampling process.

It is important to note that the recovered reward function in this study may differ from the reward function in classical reinforcement learning, which directly drives the behavior of an agent interacting with the environment. Since we leverage the assumption on human driving behaviors, the reward function is only used to score the generated candidate trajectories, and thus the learned reward function is somehow tailored to fit our problem setting. Investigating whether the recovered reward function is applicable in the classical sense of reinforcement learning requires further research.

Moreover, several limitations need to be acknowledged. First, the assumption of a linear reward function with time-invariant weights may not hold in real-world scenarios, and the hand-crafted features may not fully represent the factors involved in human driving behaviors. Second, the trajectory sampling space in this study may not sufficiently cover all possible maneuvers. This can be improved by increasing the targets in the sampling space and diversifying the planning horizon to include more complex driving behaviors. For example, we can segment the 5-second planning horizon into several intervals and sample a target state for each interval, adding more target states to generate complex actions. Future work may focus on using a neural network to parameterize the reward function, mapping raw sensory states to reward values, which could help address nonlinear reward function modeling and improve expression ability. Another focus is refining the trajectory

sampling method to capture more diverse behavior driving behaviors and achieve a more accurate estimation of the partition function.

3.6 Conclusions

In this chapter, we employ an internal reward function-based approach to model driving behavior from naturalistic human driving data in highway driving scenarios. To enable the maximum entropy IRL algorithm for inferring the reward function behind human driving, we propose a structural assumption about human driving behaviors, focusing on discrete latent intentions that govern continuous low-level control actions. Based on this assumption, we first use a polynomial trajectory sampler to generate candidate trajectories covering high-level decisions and desired speeds, while the generated trajectories are used to approximate the partition function in the maximum entropy IRL framework. An environment model is constructed to predict the trajectories of the surrounding vehicles and evaluate the outcomes of the generated trajectories. We apply the proposed method to learn the personalized reward functions of different human drivers in the NGSIM dataset and qualitatively interpret their driving decisions using the learned reward functions. The quantitative results for 100 vehicles demonstrate that the personalized modeling method outperforms the general modeling method in terms of both robustness and human likeness. Moreover, the reward-function-based models significantly outperform the IDM+MOBIL model and constant velocity model. We also find that not simulating the response actions of vehicles influenced by the ego vehicle's generated trajectory could produce better training results but compromise the generalization ability, and lacking interaction awareness could also reduce the modeling accuracy. Furthermore, we investigate applying personalized reward functions with a forecasting model in the trajectory planning process and find that the accuracy of open-loop planning depends on the accuracy of the forecasting model but still outperforms that with a general reward function.

Chapter 4

Learning World Model for Human Interactions

In this chapter, we delve into neural network-based world models for predicting human interactions in real-world traffic scenarios. Accurate predictions of interactive behaviors between traffic participants are crucial for AVs operating in complex environments. We address the interaction prediction problem by formulating it within a hierarchical game theory framework and introducing the GameFormer framework for its implementation. Specifically, our prediction model adopts a Transformer encoder-decoder structure. In the encoder, all agents and map polylines are encoded, and their relationships are extracted using self-attention Transformers. We introduce a novel Transformer decoder structure that uses the prediction results from the previous level together with the common environment background to iteratively refine the interaction process. Moreover, we propose a learning process that regulates an agent's behavior at the current level to respond to other agents' predicted behaviors from the previous level. Through experiments conducted on large-scale real-world driving datasets, we demonstrate the state-of-the-art prediction accuracy of our model in the interaction prediction task. We validate the model's ability to jointly reason about the motion plans of the ego agent and the behaviors of other agents in both open-loop and closed-loop planning tests. Our model outperforms a variety of baseline methods, thereby showcasing its effectiveness in modeling human interactions in traffic scenarios. Furthermore, we evaluate the efficacy of our model on the nuPlan planning benchmark, where it achieves high-quality planning performance in all tasks.

4.1 Introduction

Accurately forecasting the future behaviors of surrounding traffic participants and making safe and socially-compatible decisions are critical capabilities for modern autonomous driving systems. However, predicting a traffic participant's future behavior is a highly challenging task due to the heavy reliance of their behaviors on road structures, traffic norms, and underlying interactions among road users. In recent years, deep neural network-based approaches have enjoyed tremendous growth and exhibited significant improvements in prediction accuracy and scalability [83–86]. In particular, Transformer modules have gained widespread popularity in motion prediction models [87–92] because of their flexibility and effectiveness in processing information about the driving scene with a heterogeneous mix of modalities (e.g., road maps, traffic signals, and historical states of agents), as well as their ability to capture relationships among these heterogeneous elements.

Despite the success of existing prediction models in encoding the driving scene and representing interactions through agents' past trajectories, they often fail to explicitly model agents' future interactions and their interaction with the autonomous vehicle (AV). This limitation results in a passive reaction from the AV's planning module to the prediction results. However, in critical situations such as merges, lane changes, and unprotected turns, the AV needs to move actively to seek coordination with other agents. Therefore, joint prediction and planning are required to deliver more human-like and socially compatible decisions. One typical approach to address this issue is the recently-proposed conditional prediction model [93–98]. This model uses the AV's internal plans to forecast other agents' responses to the AV. Although the conditional prediction model mitigates the interaction issue, such a one-way interaction scheme still neglects the dynamic mutual influences between the AV and other road users. From a game theory perspective, the current prediction/planning models can be regarded as leader-follower games with limited interaction levels among agents.

In this study, we utilize a hierarchical game-theoretic framework [99–101] to model the mutual interactions among agents and propose a new Transformer-based model named *GameFormer* for interactive prediction and planning. Stemming from insights in cognitive science, level-k game theory offers a structured approach to modeling interactions among agents. At its core, the theory introduces a hierarchy

of reasoning depths denoted by k. A level-0 agent acts independently without considering the possible actions of other agents. As we move up the hierarchy, a level-1 agent considers interactions by assuming that other agents are level-0 and predicts their actions accordingly. This process continues iteratively, where a level-k agent predicts others' actions assuming they are level-$(k-1)$ and responds based on these predictions. Our model aligns with the spirit of level-k game theory by considering agents' reasoning levels and explicit interactions.

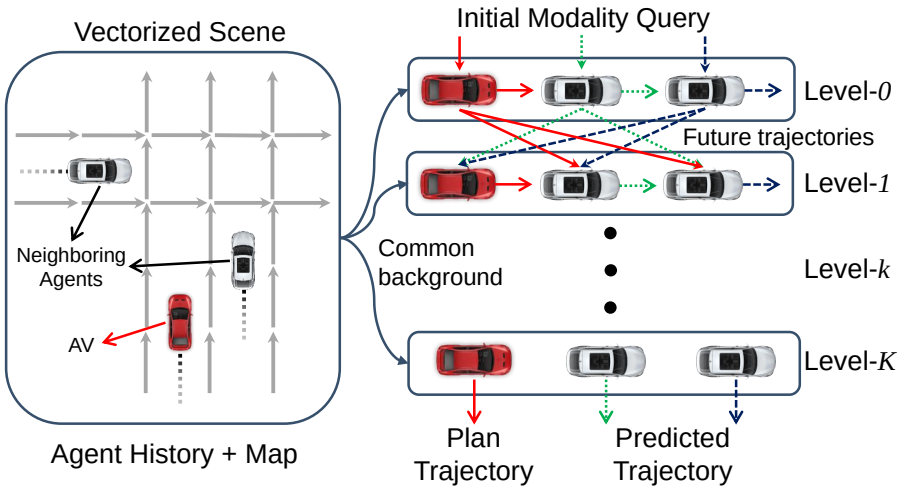


FIGURE 4.1: Hierarchical game theoretic modeling of agent interactions

As illustrated in Fig. 4.1, we employ hierarchical reasoning games to model the future interactions among agents in an iterative manner. First, we encode the driving scene as background information, which includes vectorized maps and observed agent states, using Transformer encoders. In the future decoding stage, we follow the level-k game theory [102, 103], which assumes that an agent performs a limited number of iterations of strategic reasoning. Concretely, we set up a series of Transformer decoders to implement level-k reasoning. The level-0 decoder uses only the initial modality query and encoded scene context as key and value to predict the agent's multi-modal future trajectories. Then, at each iteration k, the level-k decoder takes as input the predicted trajectories from the level-$(k-1)$ decoder, along with the background information, to predict the agent's trajectories at the current level. Moreover, we design a learning process that regulates the agents' trajectories to avoid collision with the trajectories of other agents from the last level while also staying close to human driving data. We train the proposed framework on large-scale real-world driving datasets to evaluate our approach's prediction performance and planning performance in both open-loop and closed-loop settings. The main contributions are summarized as follows:

- We propose *GameFormer*, a Transformer-based interactive prediction and planning framework. The model employs a hierarchical decoding structure to capture agent interactions, iteratively refine predictions, and is trained based on the level-k game formalism.
- We demonstrate the state-of-the-art prediction performance of our *GameFormer* model on the Waymo interaction prediction benchmark.
- We validate the planning performance of the *GameFormer* framework in open-loop driving scenes and closed-loop simulations using the Waymo open motion dataset and the nuPlan planning benchmark.

4.2 Literature Review

4.2.1 Motion prediction for autonomous driving

A rapidly growing line of research on motion prediction studies a new paradigm of forecasting traffic participants' future trajectories with deep neural networks. These models have shown powerful capabilities in encoding the scene context, which typically involves road maps and agent history states, resulting in improved prediction accuracy. Early works in this area utilized long short-term memory (LSTM) networks [104] to encode the agent's past states and convolutional neural networks (CNNs) to process the rasterized image of the scene [15, 83, 95, 105]. To explicitly model the interaction between agents, graph neural networks (GNNs) [106–109] have been widely used to process the scene graph and capture the relationship between agents. More recently, the unified Transformer encoder-decoder structure for motion prediction has gained traction due to its compact model description and superior performance, exemplified by models such as SceneTransformer [87] and WayFormer [88]. However, most Transformer-based prediction models tend to focus more on the encoding part, with less attention paid to the decoding part. Motion Transformer [89] addresses this limitation by proposing a well-designed decoding stage that leverages iterative local motion refinement to enhance prediction accuracy. Inspired by the idea of iterative refinement and hierarchical game theory, we propose a novel Transformer-based decoder for interaction prediction, providing an explicit way to model the interactions between agents.

4.2.2 Joint prediction and planning

Regarding the utilization of prediction models for planning tasks, numerous works focus on multi-agent joint motion prediction frameworks that enable efficient and consistent prediction of multi-modal multi-agent trajectories. An inherent issue in existing motion prediction models is that they often ignore the influence of the AV's actions, rendering them unsuitable for downstream planning tasks. To tackle this problem, several conditional multi-agent motion prediction models [93, 97, 98] have been proposed by integrating AV planning information into the prediction process. However, these models still exhibit one-way interactions, neglecting the mutual influence among agents. In contrast, our approach aims to jointly predict the future trajectories of surrounding agents and facilitate AV planning through iterative mutual interaction modeling.

4.2.3 Learning for decision-making

The primary goal of the motion prediction module is to enable the planning module to make safe and intelligent decisions. This can be achieved through the use of offline learning methods that can learn decision-making from large-scale driving datasets. Imitation learning is the most widely used approach, which aims to learn a driving policy that can replicate expert demonstrations [20, 110, 111]. Offline reinforcement learning [112] has also gained interest as it combines the benefits of reinforcement learning and large collected datasets. However, directly learning a policy lacks interpretability and safety guarantees, and often suffers from distributional shifts. In contrast, planning with a learned motion prediction model is believed to be more interpretable and robust [113–116], making it a more desirable way for autonomous driving. Our proposed approach aims to enhance the capability of prediction models that can improve interactive decision-making performance.

4.3 Methodology

We introduce our interactive motion prediction and planning framework, called *GameFormer*, which adopts the Transformer encoder-decoder architecture (refer

to Fig. 4.2). The encoding of the scene context is achieved through a Transformer-based encoder. The level-0 decoder utilizes the modality embedding and agent history encodings as queries, producing level-0 future trajectories and scores. The level-k decoder employs a self-attention module to model the level-$(k-1)$ future interaction and appends it to the scene context encoding. In the subsequent sections, we first define the problem and discuss the level-k game theory that guides the design of the model and learning process in Section 4.3.1. Next, we describe the encoder component of the model, which encodes the scene context, in Section 4.3.2, and the decoder component, which incorporates a novel interaction modeling concept, in Section 4.3.3. Finally, we present the learning process that accounts for interactions among different reasoning levels in Section 4.3.4.

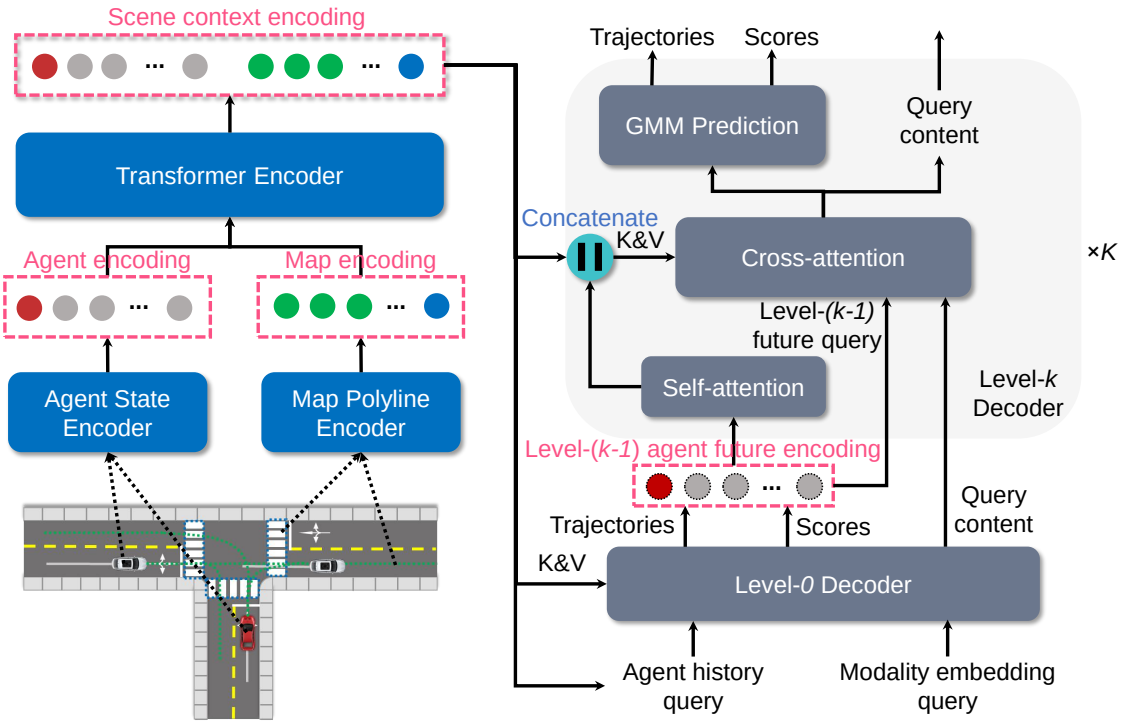


FIGURE 4.2: Overview of our proposed *GameFormer* framework

4.3.1 Game-theoretic formulation

We consider a driving scene with N agents, where the AV is denoted as A_0 and its neighboring agents as $A_1, \cdots, A_{N-1}$ at the current time $t = 0$. Given the historical states of all agents (including the AV) over an observation horizon T_h, $\mathbf{S} = \{\mathbf{s}_i^{-T_h:0}\}$, as well as the map information $\mathbf{M}$ including traffic lights and road waypoints, the goal is to jointly predict the future trajectories of neighboring agents $\mathbf{Y}_{1:N-1}^{1:T_f}$ over

the future horizon T_f, as well as a planned trajectory for the AV $\mathbf{Y}_0^{1:T_f}$. In order to capture the uncertainty, the results are multi-modal future trajectories for the AV and neighboring agents, denoted by $\mathbf{Y}_i^{1:T_f} = \{\mathbf{y}_j^{1:T_f},\ p_j | j = 1 : M\}$, where $\mathbf{y}_j^{1:T_f}$ is a sequence of predicted states, p_j the probability of the trajectory, and M the number of modalities.

We leverage level-k game theory to model agent interactions in an iterative manner. Instead of simply predicting a single set of trajectories, we predict a hierarchy of trajectories to model the cognitive interaction process. At each reasoning level, with the exception of level-0, the decoder takes as input the prediction results from the previous level, which effectively makes them a part of the scene, and estimates the responses of agents in the current level to other agents in the previous level. We denote the predicted multi-modal trajectories (essentially a Gaussian mixture model) of agent i at reasoning level k as $\pi_i^{(k)}$, which can be regarded as a policy for that agent. The policy $\pi_i^{(k)}$ is conditioned on the policies of all other agents except the i-th agent at level-$(k-1)$, denoted by $\pi_{\neg i}^{(k-1)}$. For instance, the AV's policy at level-2 $\pi_0^{(2)}$ would take into account all neighboring agents' policies at level-1 $\pi_{1:N-1}^{(1)}$. Formally, the i-th agent's level-k policy is set to optimize the following objective:

$$\min_{\pi_i} \ \mathcal{L}_i^k \left(\pi_i^{(k)} \mid \pi_{\neg i}^{(k-1)} \right), \tag{4.1}$$

where $\mathcal{L}(\cdot)$ is the loss (or cost) function. It is important to note that policy π here represents the multi-modal predicted trajectories (GMM) of an agent and that the loss function is calculated on the trajectory level.

For the level-0 policies, they do not take into account probable actions or reactions of other agents and instead behave independently. Based on the level-k game theory framework, we design the future decoder, which we elaborate upon in Section 4.3.3.

4.3.2 Scene encoding

Input representation. The input data comprises historical state information of agents, $S_p \in \mathbb{R}^{N \times T_h \times d_s}$, where d_s represents the number of state attributes, and local vectorized map polylines $M \in \mathbb{R}^{N \times N_m \times N_p \times d_p}$. For each agent, we find N_m nearby map elements such as routes and crosswalks, each containing N_p waypoints with d_p attributes. The inputs are normalized according to the state of the ego agent, and any missing positions in the tensors are padded with zeros.

Agent history encoding. We use LSTM networks to encode the historical state sequence S_p for each agent, resulting in a tensor $A_p \in \mathbb{R}^{N\times D}$, which contains the past features of all agents. Here, D denotes the hidden feature dimension.

Vectorized map encoding. To encode the local map polylines of all agents, we use the multi-layer perceptron (MLP) network, which generates a map feature tensor $M_p \in \mathbb{R}^{N\times N_m\times N_p\times D}$ with a feature dimension of D. We then group the waypoints from the same map element and use max-pooling to aggregate their features, reducing the number of map tokens. The resulting map feature tensor is reshaped into $M_r \in \mathbb{R}^{N\times N_{mr}\times D}$, where N_{mr} represents the number of aggregated map elements.

Relation encoding. We concatenate the agent features and their corresponding local map features to create an agent-wise scene context tensor $C^i = [A_p, M_p^i] \in \mathbb{R}^{(N+N_{mr})\times D}$ for each agent. We use a Transformer encoder with E layers to capture the relationships among all the scene elements in each agent's context tensor C^i. The Transformer encoder is applied to all agents, generating a final scene context encoding $C_s \in \mathbb{R}^{N\times(N+N_{mr})\times D}$, which represents the common environment background inputs for the subsequent decoder network.

4.3.3 Future decoding with level-k reasoning

Modality embedding. To account for future uncertainties, we need to initialize the modality embedding for each possible future, which serves as the query to the level-0 decoder. This can be achieved through either a heuristics-based method, learnable initial queries [88], or through a data-driven method [89]. Specifically, a learnable initial modality embedding tensor $I \in \mathbb{R}^{N\times M\times D}$ is generated, where M represents the number of future modalities.

Level-0 decoding. In the level-0 decoding layer, a multi-head cross-attention Transformer module is utilized, which takes as input the combination of the initial modality embedding I and the agent's historical encoding in the final scene context C_{s,A_p} (by inflating a modality axis), resulting in $(C_{s,A_p}+I) \in \mathbb{R}^{N\times M\times D}$ as the query and the scene context encoding C_s as the key and value. The attention is applied to the modality axis for each agent, and the query content features can be obtained after the attention layer as $Z_{L_0} \in \mathbb{R}^{N\times M\times D}$. Two MLPs are appended to the

query content features Z_{L_0} to decode the GMM components of predicted futures $G_{L_0} \in \mathbb{R}^{N\times M\times T_f\times 4}$ (corresponding to $(\mu_x, \mu_y, \log\sigma_x, \log\sigma_y)$ at every timestep) and the scores of these components $P_{L_0} \in \mathbb{R}^{N\times M\times 1}$.

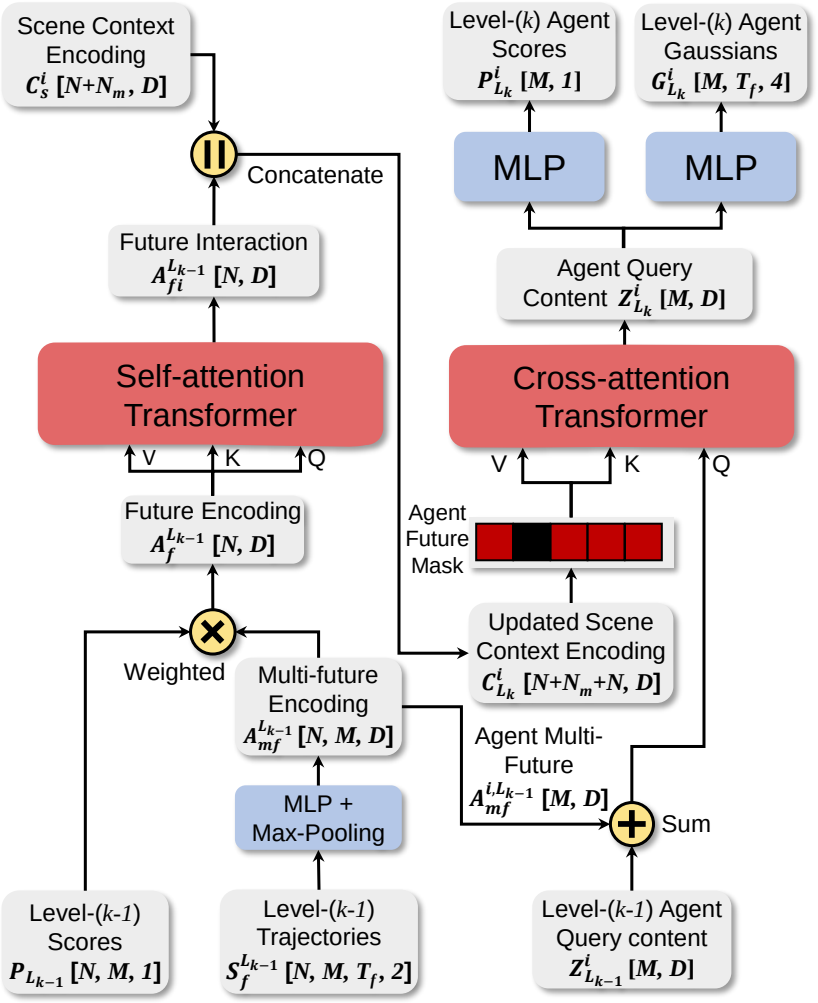


FIGURE 4.3: The detailed structure of a level-k interaction decoder

Interaction decoding. The interaction decoding stage contains K decoding layers corresponding to K reasoning levels. In the level-k layer ($k \geq 1$), it receives all agents' trajectories from the level-$(k-1)$ layer $S_f^{L_{k-1}} \in \mathbb{R}^{N\times M\times T_f\times 2}$ (the mean values of the GMM $G_{L_{k-1}}$) and use an MLP with max-pooling on the time axis to encode the trajectories, resulting in a tensor of agent multi-modal future trajectory encoding $A_{mf}^{L_{k-1}} \in \mathbb{R}^{N\times M\times D}$. Then, we apply weighted-average-pooling on the modality axis with the predicted scores from the level-$(k-1)$ layer $P_{L_{k-1}}$ to obtain the agent future features $A_f^{L_{k-1}} \in \mathbb{R}^{N\times D}$. We use a multi-head self-attention Transformer module to model the interactions between agent future trajectories $A_{fi}^{L_{k-1}}$ and concatenate the resulting interaction features with the scene context encoding

from the encoder part. This yields an updated scene context encoding for agent i, denoted by $C^i_{L_k} = [A^{L_{k-1}}_{fi}, C^i_s] \in \mathbb{R}^{(N+N_m+N)\times D}$. We adopt a multi-head cross-attention Transformer module with the query content features from the level-$(k-1)$ layer $Z^i_{L_{k-1}}$ and agent future features $A^{L_{k-1}}_{mf}$, $(Z^i_{L_{k-1}} + A^{i,L_{k-1}}_{mf}) \in \mathbb{R}^{M\times D}$ as query and the updated scene context encoding $C^i_{L_k}$ as key and value. We use a masking strategy to prevent an agent from accessing its own future information from the last layer. For example, agent A_0 can only get access to the future interaction features of other agents $\{A_1, \cdots, A_{N-1}\}$. Finally, the resulting query content tensor from the cross-attention module $Z^i_{L_k}$ is passed through two MLPs to decode the agent's GMM components and scores, respectively. Fig. 4.3 illustrates the detailed structure of a level-k interaction decoder. Note that we share the level-k decoder for all agents to generate multi-agent trajectories at that level. At the final level of interaction decoding, we can obtain multi-modal trajectories for the AV and neighboring agents G_{L_K}, as well as their scores P_{L_K}.

4.3.4 Learning process

We present a learning process to train our model using the level-k game theory formalism. First, we employ imitation loss as the primary loss to regularize the agent's behaviors, which can be regarded as a surrogate for factors such as traffic regulations and driving styles. The future behavior of an agent is modeled as a Gaussian mixture model (GMM), where each mode m at time step t is described by a Gaussian distribution over the (x, y) coordinates, characterized by mean μ^t_m and covariance σ^t_m. The imitation loss is computed using the negative log-likelihood loss from the best-predicted component m^* (closest to the ground truth) at each timestep, as formulated:

$$\mathcal{L}_{IL} = \sum_{t=1}^{T_f} \mathcal{L}_{NLL}(\mu^t_{m^*}, \sigma^t_{m^*}, p_{m*}, \mathbf{s}_t). \tag{4.2}$$

The negative log-likelihood loss function $\mathcal{L}_{NLL}$ is defined as follows:

$$\mathcal{L}_{NLL} = \log \sigma_x + \log \sigma_y + \frac{1}{2}\left(\left(\frac{dx}{\sigma_x}\right)^2 + \left(\frac{dx}{\sigma_x}\right)^2\right) - \log(p_{m*}), \tag{4.3}$$

where $d_x = \mathbf{s}_x - \mu_x$ and $d_y = \mathbf{s}_y - \mu_y$, $(\mathbf{s}_x, \mathbf{s}_y)$ is ground-truth position; p_{m*} is the probability of the selected component, and we use the cross-entropy loss in practice.

For a level-k agent $A_i^{(k)}$, we design an auxiliary loss function inspired by prior works [108, 117, 118] that considers the agent's interactions with others. The safety of agent interactions is crucial, and we use an interaction loss (applicable only to decoding levels $k \geq 1$) to encourage the agent to avoid collisions with the possible future trajectories of other level-$(k-1)$ agents. Specifically, we use a repulsive potential field in the interaction loss to discourage the agent's future trajectories from getting too close to any possible trajectory of any other level-$(k-1)$ agent $A_{\neg i}^{(k-1)}$. The interaction loss is defined as follows:

$$\mathcal{L}_{Inter} = \sum_{m=1}^{M} \sum_{t=1}^{T_f} \max_{\substack{\forall j \neq i \\ \forall n \in 1:M}} \frac{1}{d\left(\hat{\mathbf{s}}_{m,t}^{(i,k)}, \hat{\mathbf{s}}_{n,t}^{(j,k-1)}\right) + 1}, \tag{4.4}$$

where $d(\cdot, \cdot)$ is the L_2 distance between the future states ((x, y) positions), m is the mode of the agent i, n is the mode of the level-$(k-1)$ agent j. To ensure activation of the repulsive force solely within close proximity, a safety margin is introduced, meaning the loss is only applied to interaction pairs with distances smaller than a threshold.

The total loss function for the level-k agent i is the weighted sum of the imitation loss and interaction loss.

$$\mathcal{L}_i^k(\pi_i^{(k)}) = w_1 \mathcal{L}_{IL}(\pi_i^{(k)}) + w_2 \mathcal{L}_{Inter}(\pi_i^{(k)}, \pi_{\neg i}^{(k-1)}), \tag{4.5}$$

where w_1 and w_2 are the weighting factors to balance the influence of the two loss terms.

4.4 Experiments

4.4.1 Dataset

We set up two different model variants for different evaluation purposes. The prediction-oriented model is trained and evaluated using the Waymo open motion

dataset (WOMD) [119], specifically addressing the task of predicting the joint trajectories of two interacting agents. For the planning tasks, we train and test the models on both WOMD with selected interactive scenarios and the nuPlan dataset [120] with a comprehensive evaluation benchmark.

4.4.2 Prediction-oriented model

This model variant focuses on improving prediction accuracy, especially in interaction prediction. We adopt the setting of the WOMD interaction prediction task, where the model predicts the joint future positions of two interacting agents 8 seconds into the future, based on their one second of historical data and high-definition maps. The neighboring agents within the scene will serve as the background information in the encoding stage, while only the two labeled interacting agents' joint future trajectories are predicted. The model is trained on the entire WOMD training dataset, and we adhere to the official evaluation metrics, which include minimum average displacement error (minADE), minimum final displacement error (minFDE), miss rate, and mean average precision (mAP). We investigate two different interaction prediction settings. Firstly, we consider the joint prediction setting, where only 6 joint trajectories of the two agents are predicted [87]. Secondly, we examine the marginal prediction setting and train our model to predict 64 marginal trajectories for each agent in the interaction pair. During inference, the EM method proposed in MultiPath++ [86] is employed to generate a set of 6 marginal trajectories for each agent, from which the top 6 joint predictions are selected for these two agents.

Training. In the training dataset, each scene contains several agent tracks to predict, and we consider each track sequentially as the focal agent, while the closest track to the focal agent is chosen as the interacting agent. The task is to predict six possible joint future trajectories of these two agents. To improve the prediction accuracy, we employ only imitation loss at each decoding level. In the joint prediction model, we aim to predict the joint and scene-level future trajectories of the two agents. Therefore, we backpropagate the loss through the joint future trajectories of the two agents that most closely match the ground truth (i.e., have the least sum of displacement errors). In the marginal prediction model, we backpropagate the imitation loss to the individual agent through the positive GMM component

that corresponds to the closest intention point to the endpoint of the ground-truth trajectory. Our models are trained for 30 epochs using the AdamW optimizer with a weight decay of 0.01. The learning rate starts with 1e-4 and decays by a factor of 0.5 every 3 epochs after 15 epochs. We also clip the gradient norm of the network parameters with the max norm of the gradients as 5. We train the models using 4 NVIDIA Tesla V100 GPUs, with a batch size of 64 per GPU.

Testing. There are three types of agents in the testing dataset, which are vehicle, pedestrian, and cyclist. For the vehicle-vehicle interaction, we randomly select one of the two vehicles as the focal agent. For other types of interaction pairs (e.g., cyclist-vehicle and pedestrian-vehicle), we consider the cyclist or pedestrian as the focal agent. For the marginal prediction model, we employ the Expectation-Maximization (EM) method to aggregate trajectories for each agent. Specifically, we use the EM method to obtain 6 marginal trajectories (along with their probabilities) from the 64 trajectories predicted for each agent. Then, we consider the top 6 joint predictions from the 36 possible combinations of the two agents, where the confidence of each combination is the product of the marginal probabilities.

4.4.3 Planning-oriented model

We introduce another model variant that is designed for planning. Specifically, the observation horizon is 1 second and the prediction/planning horizon is set to 5 seconds, and all the neighboring agents' future interactions are considered to predict their future trajectories. We randomly select 10,000 20-second scenarios from the WOMD dataset, where 9,000 scenarios are used for training and the remaining 1,000 for validation. Furthermore, we evaluate the model's joint prediction and planning performance on 400 9-second scenarios[1], which includes interesting interactions such as lane-change, merge, and left-turn. To conduct closed-loop testing, we utilize a log-replay simulator to replay the original scenarios and override the AV with our planner. In addition, to improve the closed-loop planning performance, we employ a motion planner [113] to refine the trajectory provided by the model by explicitly optimizing the trajectory's cost. In open-loop testing, we use distance-based error metrics, including planning ADE, collision rate, miss rate, and prediction ADE. In closed-loop testing, we focus on evaluating the planner's

[1] `https://github.com/smarts-project/smarts-project.offline-datasets/blob/master/waymo_candid_list.csv`

performance in a realistic driving scenario by measuring metrics including collision rate, off-route rate, progress along the route, longitudinal acceleration and jerk, lateral acceleration, and position errors.

Training. In data processing, we filter those scenes where the AV's moving distance is less than 5 meters (e.g. when stopping at a red light). Similarly, we perform joint future prediction and calculate the imitation loss through the joint future that is closest to the ground truth. The weights for the imitation loss and interaction loss are set to $w_1 = 1, w_2 = 0.1$. Our model is trained for 20 epochs using the AdamW optimizer with a weight decay of 0.01. The learning rate is initialized to 1e-4 and decreases by a factor of 0.5 every 2 epochs after the 10th epoch. We train the model using an NVIDIA RTX 3080 GPU, with a batch size of 32.

Testing. In open-loop testing, we check collisions between the AV's planned trajectory and other agents' ground-truth future trajectories, and we count a miss if the distance between AV's planned state at the final step and the ground-truth state is larger than 4.5 meters. The planning errors and prediction errors are calculated according to the most likely trajectories scored by the model. In closed-loop testing, the AV plans a trajectory at every timestep and executes the first step of the plan.

4.4.4 GameFormer planner

For the nuPlan dataset, we design a comprehensive planning framework called *GameFormer Planner* and adhere to the nuPlan challenge settings[2] to evaluate the framework's planning performance.

The planning process comprises the following steps: 1) feature processing: relevant data from the observation buffer and map API undergoes preprocessing to extract input features for the prediction model; 2) path planning: candidate route paths for the ego vehicle are computed, from which the optimal path is selected as the reference path; 3) model query: the prediction model is queried to generate an initial plan for the ego vehicle and predict the trajectories of surrounding agents; and 4) trajectory refinement: a nonlinear optimizer is employed to refine the ego vehicle's

[2] `https://eval.ai/web/challenges/challenge-page/1856/overview`

trajectory on the reference path and produce the final plan. For computational efficiency, we use a compact version of the GameFormer model, configuring it with 3 encoding layers and 3 decoding layers (1 initial decoding layer and 2 interaction decoding layers). Additionally, we introduce an extra decoding layer after the last interaction decoding layer to separately generate the ego vehicle's plan. The ego plan is then projected onto the reference path as an initialization of the refinement planner. The output of the GameFormer model consists of multimodal trajectories for the surrounding agents. For each neighboring agent, we select the trajectory with the highest probability and project it onto the reference path using the Frenet transformation, subsequently calculating spatiotemporal path occupancy. More details about this planning framework can be found in this dedicated report[3].

We evaluate the planner's performance in three tasks: open-loop planning, closed-loop planning with non-reactive agents, and closed-loop with reactive agents. These tasks are evaluated using a comprehensive set of metrics provided by the nuPlan platform, and an overall score is derived based on these tasks.

4.4.5 Baseline methods

To validate the planning capability of our model, we introduce the following imitation learning-based planning baselines.

Vanilla Imitation Learning (IL): A simplified version of our model that directly outputs the planned trajectory of the AV without explicitly reasoning other agents' future trajectories. The plan is only a single-modal trajectory. The original encoder part of our model is utilized, but only one decoder layer with the ego agent's historical encoding as query is used to decode the AV's plan.

Deep Imitative Model (DIM) [121]: A probabilistic planning method that aims to generate expert-like future trajectories $q\left(\mathbf{S}_{1:T}|\phi\right) = \prod_{t=1}^{T} q\left(\mathbf{S}_t|\mathbf{S}_{1:t-1}, \phi\right)$ given the AV's observations ϕ. We follow the original open-source DIM implementation and use the rasterized scene image $\mathbb{R}^{200\times200\times3}$ and the AV's historical states $\mathbb{R}^{11\times5}$ as the observation. We use a CNN to encode the scene image and an RNN to encode the agent's historical states. The AV's future state is decoded (as a multivariate Gaussian distribution) in an autoregressive manner. In testing,

[3] `https://opendrivelab.com/e2ead/AD23Challenge/Track_4_AID.pdf`

DIM requires a specific goal $\mathcal{G}$ to direct the agent to the goal, and a gradient-based planner maximizes the learned imitation prior $\log q(\mathbf{S}|\phi)$ and the test-time goal likelihood $\log p(\mathcal{G}|\mathbf{S}, \phi)$.

Robust Imitative Planning (RIP) [122]: An epistemic uncertainty-aware planning method that is developed upon DIM and shows good performance in conducting robust planning in out-of-distribution (OOD) scenarios. Specifically, we employ the original open-source implementation and choose the worst-case model that has the worst likelihood $\min_d \log q(\mathbf{S}_{1:T}|\phi)$ among $d = 6$ trained DIM models and improve it with a gradient-based planner.

Conservative Q-Learning (CQL) [112]: A widely-used offline reinforcement learning algorithm that learns to make decisions from offline datasets. We implement the CQL method with the d3rlpy offline RL library. The RL agent takes the same state inputs as the DIM method and outputs the target pose of the next step $(\Delta x, \Delta y, \Delta\theta)$ relative to the agent's current position. The reward function is the distance traveled per step plus an extra reward for reaching the goal, i.e., $r_t = \Delta d_t + 10 \times 1\,(d(s_t, s_{goal}) < 1)$. Since the dataset only contains perfect driving data, no collision penalty is included.

Differentiable Integrated Prediction and Planning (DIPP) [113]: A joint prediction and planning method that uses a differentiable motion planner to optimize the trajectory according to the prediction result. We adopt the original open-source implementation and the same state input setting. We increase the historical horizon to 20 and the number of prediction modalities from 3 to 6. In open-loop testing, we utilize the results from the DIPP prediction network without trajectory planning (refinement).

MultiPath++ [86, 123]: A high-performing motion prediction model that is based on the context-aware fusion of heterogeneous scene elements and learnable latent anchor embeddings. We utilize the open-source implementation of MultiPath++ that achieved state-of-the-art prediction accuracy on the WOMD motion prediction benchmark. We train the model to predict 6 possible trajectories and corresponding scores for the ego agent using the same dataset. In open-loop testing, only the most-likely trajectory will be used as the plan for the AV.

Motion Transformer (MTR)-e2e [89]: A state-of-the-art prediction model that occupies the first place on the WOMD motion prediction leaderboard. We follow

the original open-source implementation of the context encoder and MTR decoder. However, we modified the decoder to use an end-to-end variant of MTR that is better suited for the open-loop planning task. Specifically, only 6 learnable motion query pairs are used to decode 6 possible trajectories and scores. The same dataset is used to train MTR-e2e model, and the data is processed according to the MTR context inputs.

4.5 Results and Discussions

4.5.1 Interaction prediction

For the prediction-oriented model, we use a stack of $E = 6$ Transformer encoder layers, and the hidden feature dimension is set to $D = 256$. We consider 20 neighboring agents around the two interacting agents as background information and employ $K = 6$ decoding layers. The model only generates trajectories for the two labeled interacting agents. Moreover, the local map elements for each agent comprise possible lane polylines and crosswalk polylines.

Quantitative results. Table 4.1 summarizes the prediction performance of our model in comparison with state-of-the-art methods on the WOMD interaction prediction (joint prediction of two interacting agents) benchmark. The metrics are averaged over different object types (vehicle, pedestrian, and cyclist) and evaluation times (3, 5, and 8 seconds). Our joint prediction model (GameFormer (J, M=6)) outperforms existing methods in terms of position errors. This can be attributed to its superior ability to capture future interactions between agents through an iterative process and to predict future trajectories in a scene-consistent manner. However, the scoring performance of the joint model is limited without predicting an over-complete set of trajectories and aggregation. To mitigate this issue, we employ the marginal prediction model (GameFormer (M, M=64)) with EM aggregation, which significantly improves the scoring performance (better mAP metric). The overall performance of our marginal model is comparable to that of the ensemble and more complicated MTR model [89]. Nevertheless, it is worth noting that marginal ensemble models may not be practical for real-world applications due to their substantial computational burden. Therefore, we utilize the joint prediction

model, which provides better prediction accuracy and computational efficiency, for planning tests.

TABLE 4.1: Comparison with state-of-the-art models on the WOMD interaction prediction benchmark

Model	minADE (↓)	minFDE (↓)	Miss rate (↓)	mAP (↑)
LSTM baseline [119]	1.9056	5.0278	0.7750	0.0524
Heat [106]	1.4197	3.2595	0.7224	0.0844
AIR^2 [124]	1.3165	2.7138	0.6230	0.0963
SceneTrans [87]	0.9774	2.1892	0.4942	0.1192
DenseTNT [85]	1.1417	2.4904	0.5350	0.1647
M2I [96]	1.3506	2.8325	0.5538	0.1239
MTR [89]	0.9181	2.0633	**0.4411**	**0.2037**
GameFormer (M, M=64)	0.9721	2.2146	0.4933	0.1923
GameFormer (J, M=6)	**0.9161**	**1.9373**	0.4531	0.1376

Table 4.2 displays the per-category performance of our models on the WOMD interaction prediction benchmark, in comparison with the MTR model. The GameFormer joint prediction model exhibits the lowest minFDE in all object categories, indicating the advantage of our model and joint training of interaction patterns. Our GameFormer model surpasses MTR in the cyclist category and achieves comparable performance with MTR in other categories, albeit with a much simpler structure than MTR.

TABLE 4.2: Per-class performance of interaction prediction on the WOMD interaction prediction benchmark

Class	Model	minADE (↓)	minFDE (↓)	Miss rate (↓)	mAP (↑)
Vehicle	MTR	**0.9793**	2.2157	0.3833	**0.2977**
	GF (J)	0.9822	**2.0745**	**0.3785**	0.1856
	GF (M)	1.0499	2.4044	0.4321	0.2469
Pedestrian	MTR	**0.7098**	1.5835	**0.3973**	**0.2033**
	GF (J)	0.7279	**1.4894**	0.4272	0.1505
	GF (M)	0.7978	1.8195	0.4713	0.1962
Cyclist	MTR	1.0652	2.3908	**0.5428**	0.1102
	GF (J)	**1.0383**	**2.2480**	0.5536	0.0768
	GF (M)	1.0686	2.4199	0.5765	**0.1338**

Qualitative results. Fig. 4.4 illustrates the performance of our approach in predicting interactions in various typical scenarios. The red boxes represent the

interacting agents to be predicted, while the magenta boxes depict the neighboring background agents. Our model predicts six joint trajectories for the interacting agents. In the vehicle-vehicle interaction scenario, two distinct situations are captured by our model: vehicle 2 accelerates to pass the intersection first and vehicle 2 yields to vehicle 1. In both situations, our model predicts that Vehicle 1 creeps forward to observe the actions of Vehicle 2 before completing a left turn. In the vehicle-pedestrian scenario, our model predicts that the vehicle will stop and wait for the pedestrian to pass before resuming movement. In the vehicle-cyclist interaction scenario, where the vehicle intends to merge into the right lane, our model predicts the vehicle will decelerate and follow behind the cyclist in that lane. Overall, the results manifest that our model can accurately predict the possible joint futures of interacting agents.

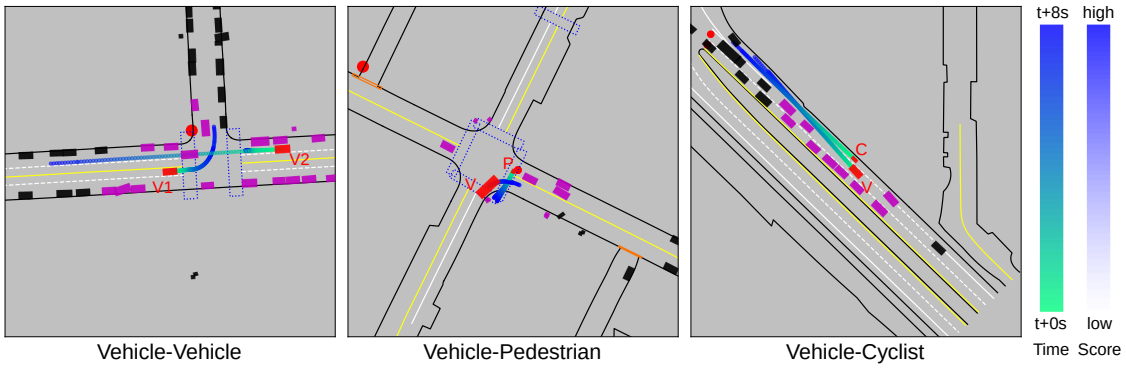


FIGURE 4.4: Qualitative results of the proposed method in interaction prediction

Additional qualitative results of our GameFormer framework in the interaction prediction task are presented in Figure 4.5. These results showcase the capability of our method to handle various interaction pairs and urban driving scenarios.

Level-k Prediction. Fig. 4.6 illustrates the most-likely joint trajectories of the target agents at different interaction levels. Only the most-likely joint trajectories of the target agents are displayed for clarity. The results demonstrate that our proposed framework is capable of refining the prediction results in the iterated interaction process. At level-0, the target agents are predicted rather independently, and their trajectories may collide. However, through iterative refinement, our model can output reasonable and close-to-human trajectories at the final level.

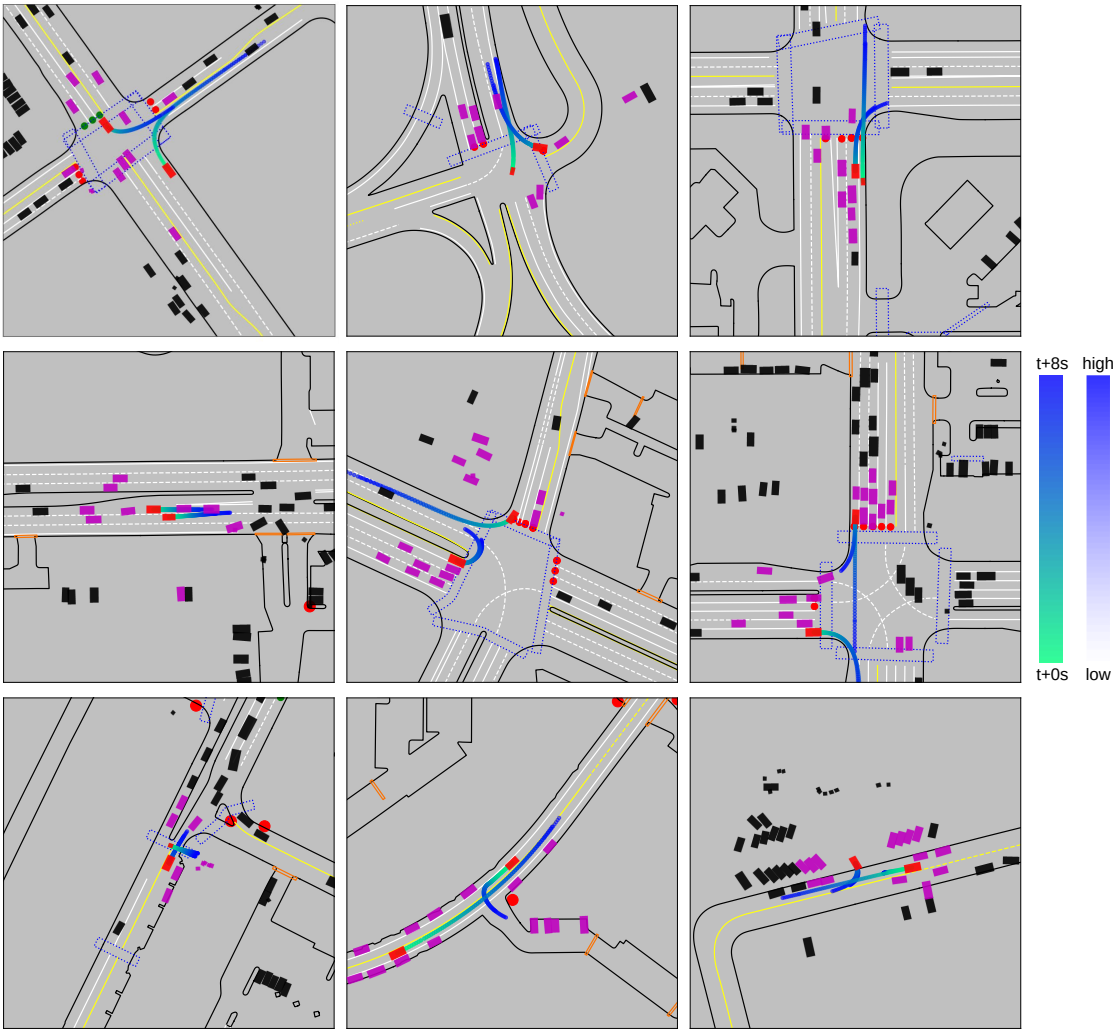


FIGURE 4.5: Additional qualitative results of interaction prediction

4.5.2 Open-loop planning

We first conduct the planning tests in selected WOMD scenarios with a prediction/planning horizon of 5 seconds. The model uses a stack of $E = 6$ Transformer encoder layers, and we consider 10 neighboring agents closest to the ego vehicle to predict $M = 6$ joint future trajectories for them.

Determining the decoding levels. To determine the optimal reasoning levels for planning, we analyze the impact of decoding layers on open-loop planning performance, and the results are presented in Table 4.3. It can be inferred that the performance reaches its peak when the number of interaction decoding layers is increased to 4. Although the planning ADE and prediction ADE exhibit a slight decrease with additional decoding layers, the miss rate and collision rate are at

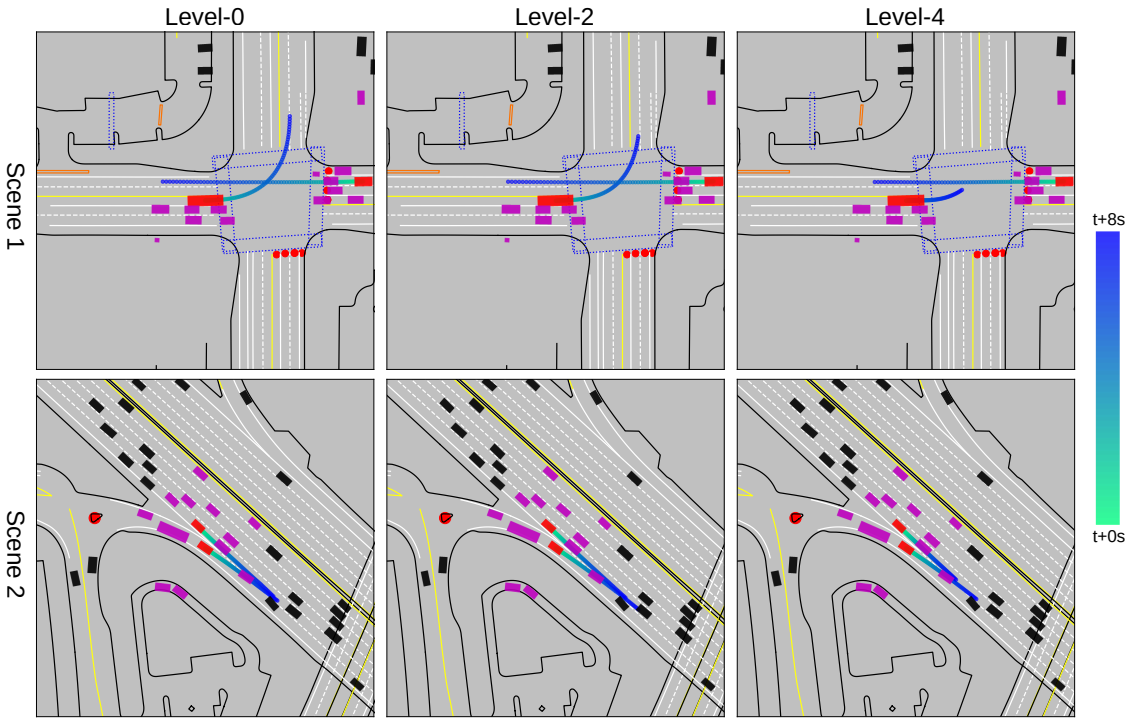


FIGURE 4.6: Prediction results of the two interacting agents at different reasoning levels

their lowest when the decoding level is 4. The intuition behind this observation is that humans are capable of performing only a limited depth of reasoning. In our open-loop planning experiments, the optimal iteration depth empirically appears to be $K = 4$.

TABLE 4.3: Influence of decoding levels on open-loop planning

Level	Planning ADE	Collision Rate	Miss Rate	Prediction ADE
0	0.9458	0.0384	0.1154	1.0955
1	0.8846	0.0305	0.0994	0.9377
2	0.8529	0.0277	0.0897	0.8875
3	0.8423	0.0269	0.0816	0.8723
4	0.8329	**0.0198**	**0.0753**	0.8527
5	**0.8171**	0.0245	0.0777	0.8361
6	0.8208	0.0238	0.0826	**0.8355**

Quantitative results. Our joint prediction and planning model employs 4 decoding layers, and the results of the final decoding layer (the most-likely future evaluated by the trained scorer) are utilized as the plan for the AV and predictions for other agents. We set up some imitation learning-based planning methods as baselines, which are: 1) vanilla imitation learning (IL), 2) deep imitative

model (DIM), 3) MultiPath++ (which predicts multi-modal trajectories for the ego agent), 4) MTR-e2e (end-to-end variant with learnable motion queries), and 5) differentiable integrated prediction and planning (DIPP). Table 4.4 reports the open-loop planning performance of our model in comparison with the baseline methods. The results reveal that our model performs significantly better than vanilla IL and DIM, because they are just trained to output the ego's trajectory while not explicitly predicting other agents' future behaviors. Compared to performant motion prediction models (MultiPath++ and MTR-e2e), our model also shows better planning metrics for the ego agent. Moreover, our model outperforms DIPP (a joint prediction and planning method) in both planning and prediction metrics, especially the collision rate. These results emphasize the advantage of our model, which explicitly considers all agents' future behaviors and iteratively refines the interaction process.

TABLE 4.4: Evaluation of open-loop planning performance of the proposed method against baseline methods

Method	Collision rate (%)	Miss rate (%)	Planning error (m)			Prediction error (m)	
			@1s	@3s	@5s	ADE	FDE
Vanilla IL	4.25	15.61	0.216	1.273	3.175	–	–
DIM	4.96	17.68	0.483	1.869	3.683	–	–
MultiPath++	2.86	8.61	0.146	0.948	2.719	–	–
MTR-e2e	2.32	8.88	0.141	0.888	2.698	–	–
DIPP	2.33	8.44	0.135	0.928	2.803	0.925	2.059
Ours	**1.98**	**7.53**	**0.129**	**0.836**	**2.451**	**0.853**	**1.919**

Qualitative results. Fig. 4.7 displays qualitative results of our model's open-loop planning performance in complex driving scenarios. For clarity, only the most-likely trajectories of the agents are displayed. The red box is the AV and the magenta boxes are its neighboring agents; the red trajectory is the plan of the AV and the blue ones are the predictions of neighboring agents. These results demonstrate that our model can generate a plausible future trajectory for the AV and handle diverse interaction scenarios, and predictions of the surrounding agents enhance the interpretability of our planning model's output.

Fig. 4.8 provides additional qualitative results of our framework in the open-loop planning task, which depict the ability of our model to plan the trajectory of the AV and predict its neighboring agents. The red box is the AV and the magenta boxes are its neighboring agents; the red trajectory is the plan of the AV and the blue ones are the predictions of neighboring agents

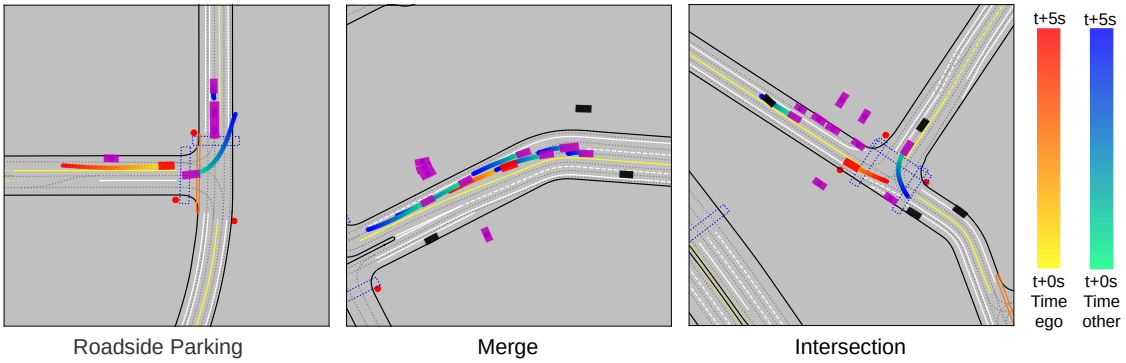


FIGURE 4.7: Qualitative results of the proposed method in open-loop planning

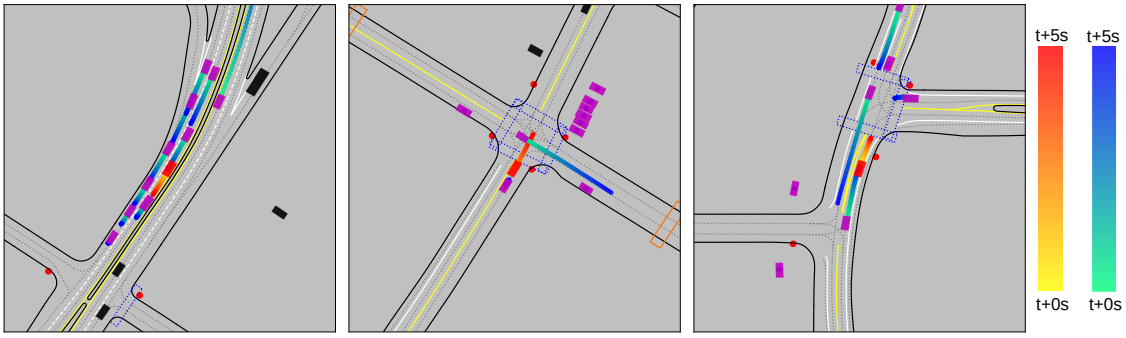


FIGURE 4.8: Additional qualitative results of open-loop planning

4.5.3 Closed-loop planning

We evaluate the closed-loop planning performance of our model in selected WOMD scenarios. Within a simulated environment [113], we execute the planned trajectory generated by the model and update the ego agent's state at each time step, while other agents follow their logged trajectories from the dataset. Since other agents do not react to the ego agent, the success rate is a lower bound for safety assessment. For planning-based methods (DIPP and our proposed method), we project the output trajectory onto a reference path to ensure the ego vehicle's adherence to the roadway. Additionally, we employ a cost-based refinement planner [113], which utilizes the initial output trajectory and the predicted trajectories of other agents to explicitly regulate the ego agent's actions. Our method is compared against four baseline methods: 1) vanilla IL, 2) robust imitative planning (RIP) [122], 3) conservative Q-learning (CQL) [112], and 4) DIPP [113]. We report the means and standard deviations of the planning-based methods over three training runs (models trained with different seeds). The quantitative results of closed-loop testing are summarized in Table 4.5. The results show that the IL and offline RL methods exhibit subpar performance in the closed-loop test, primarily due to

distributional shifts and casual confusion. In contrast, planning-based methods perform significantly better across all metrics. Without the refinement step, our model outperforms DIPP because it captures agent interactions more effectively and thus the raw trajectory is closer to an expert driver. With the refinement step, the planner becomes more robust against training seeds, and our method surpasses DIPP because it can deliver better predictions of agent interactions and provide a good initial plan to the refinement planner.

TABLE 4.5: Evaluation of closed-loop planning performance of the proposed method against baseline methods

Method	Success rate (%)	Progress (m)	Acceleration (m/s^2)	Jerk (m/s^3)	Lateral acc. (m/s^2)	Position error to expert driver (m) @3s	@5s	@8s
Vanilla IL	0	6.23	1.588	16.24	0.661	9.355	20.52	46.33
RIP	19.5	12.85	1.445	14.97	0.355	7.035	17.13	38.25
CQL	10	8.28	3.158	25.31	0.152	10.86	21.18	40.17
DIPP	68.12±5.51	41.08±5.88	1.44±0.18	12.58±3.23	0.31±0.11	6.22±0.52	15.55±1.12	26.10±3.88
Ours	73.16±6.14	44.94±7.69	1.19±0.15	13.63±2.88	0.32±0.09	5.89±0.78	12.43±0.51	21.02±2.48
DIPP (w/ refinement)	92.16±0.62	51.85±0.14	0.58±0.03	**1.54**±0.19	0.11±0.01	2.26±0.10	5.55±0.24	12.53±0.48
Ours (w/ refinement)	**94.50**±0.66	**52.67**±0.33	**0.53**±0.02	1.56±0.23	**0.10**±0.01	**2.11**±0.21	**4.87**±0.18	**11.13**±0.33

4.5.4 nuPlan benchmark evaluation

To handle diverse driving scenarios in the nuPlan platform [120], we develop a comprehensive planning framework *GameFormer Planner*. It fulfills all important steps in the planning pipeline, including feature processing, path planning, model query, and motion refinement. We increase the prediction and planning horizon to 8 seconds to meet benchmark requirements. The evaluation is conducted over three tasks: open-loop (OL) planning, closed-loop (CL) planning with non-reactive agents, and closed-loop planning with reactive agents. The score for each individual task is calculated using various metrics and scoring functions, and an overall score is obtained by aggregating these task-specific scores. It is important to note that we reduce the size of our model (encoder and decoder layers) due to limited computational resources on the test server. The performance of our model on the nuPlan test benchmark is presented in Table 4.6, in comparison with other competitive learning-based methods and a rule-based approach (IDM Planner). The results reveal the capability of our planning framework to achieve high-quality planning results across the evaluated tasks. Moreover, the closed-loop visualization results

illustrate the ability of our model to facilitate the ego vehicle in making interactive and human-like decisions[4].

TABLE 4.6: Results on the nuPlan planning benchmark

Method	Overall	OL	CL non-reactive	CL reactive
Hoplan	0.8745	0.8523	0.8899	0.8813
Multi_path	0.8477	0.8758	0.8165	0.8506
GameFormer	0.8288	0.8400	0.8087	0.8376
Urban Driver	0.7467	0.8629	0.6821	0.6952
IDM Planner	0.5912	0.2944	0.7243	0.7549

4.5.5 Ablation study

Effects of agent future modeling. We investigate the impact of different agent future modeling settings on the open-loop planning performance. We compare our base model to three ablated models, which are: 1) *No future* (where agent future trajectories from the last level are not incorporated in decoding the current level), 2) *No self-attention* (where agent future trajectories are incorporated but not processed by a self-attention module), and 3) *No interaction loss* (where the model is trained without the proposed interaction loss). The results presented in Table 4.7 indicate that our game-theoretic approach can significantly improve the planning and prediction accuracy by utilizing the future trajectories of agents from the last level as the scene context in the current level. Additionally, incorporating a self-attention module to represent future interactions among agents improves the accuracy, while using the proposed interaction loss to train the model significantly reduces the collision rate.

TABLE 4.7: Influence of agent future modeling on open-loop planning

	Planning ADE	Collision Rate	Miss Rate	Prediction ADE
No future	0.9210	0.0295	0.0963	0.9235
No self-attention	0.8666	0.0231	0.0860	0.8856
No interaction loss	0.8415	0.0417	0.0846	**0.8486**
Base	**0.8329**	**0.0198**	**0.0753**	0.8527

Influence of decoder structures. We investigate the influence of decoder structures on the open-loop planning task in WOMD scenarios. Specifically, we examine

[4] `https://mczhi.github.io/GameFormer/`

two ablated models. First, we assess the importance of incorporating k independent decoder layers, as opposed to training a single shared interaction decoder and iteratively applying it k times. Second, we explore the impact of simplifying the decoder into a multi-layer Transformer that does not generate intermediate states. This translates into applying the loss solely to the final decoding layer, rather than all intermediate layers. The results presented in Table 4.8 demonstrate better open-loop planning performance for the base model (independent decoding layers with intermediate trajectories). This design allows each layer to capture different levels of relationships, thereby facilitating hierarchical modeling. In addition, the omission of intermediate trajectory outputs can degrade the model's performance, highlighting the necessity of regularizing the intermediate state outputs.

TABLE 4.8: Influence of decoder structures on open-loop planning

	Planning ADE	Collision Rate	Miss Rate	Prediction ADE
Base	**0.8329**	**0.0198**	**0.0753**	**0.8547**
Shared decoder	0.9196	0.0382	0.0860	0.9095
Multi-layer decoder	0.9584	0.0353	0.0988	0.9637

Ablation results on the interaction prediction task. We investigate the influence of the decoder on the WOMD interaction prediction task. Specifically, we vary the decoding levels from 0 to 8 to determine the optimal decoding level for this task. Moreover, we remove either the agent future encoding part from the decoder or the self-attention module (for modeling agent future interactions) to investigate their influences on prediction performance. We train the ablated models using the same training set and evaluate their performance on the validation set. The results in Table 4.9 reveal that the empirically optimal number of decoding layers is 6 for the interaction prediction task. It is evident that fewer decoding layers fail to adequately capture the interaction dynamics, resulting in subpar prediction performance. However, using more than 6 decoding layers may introduce training instability and overfitting issues, leading to worse testing performance. Similarly, we find that incorporating predicted agent future information is crucial for achieving good performance, and using self-attention to model the interaction among agents' futures can also improve prediction accuracy.

Effects of decoding levels on closed-loop planning. We investigate the influence of decoding levels on closed-loop planning performance, using the success rate (without collision) as the main metric. Additionally, we report the inference

TABLE 4.9: Decoder ablation results on interaction prediction

Decoding layers	minADE	minFDE	Miss Rate	mAP
K=0	1.0505	2.2905	0.5113	0.1226
K=1	1.0169	2.1876	0.5061	0.1281
K=3	0.9945	2.1143	0.5026	0.1265
K=6	**0.9133**	**1.9251**	**0.4564**	**0.1339**
K=8	0.9839	2.1515	0.5003	0.1255
K=6 w/o future	0.9862	2.0848	0.4979	0.1256
K=6 w/o self-attention	0.9263	1.9931	0.4599	0.1281

time of the prediction network (without the refine motion planner) in closed-loop planning, which is executed on an NVIDIA RTX 3080 GPU. The results in Table 4.10 reveal that increasing the decoding layers could lead to a higher success rate, and adding one layer of interaction modeling can bring significant improvement compared to level-0. In closed-loop testing, the success rate plateaus when the decoding level reaches 2, but the computation time continues to increase. Therefore, using two reasoning levels in our model may achieve a good balance between performance and efficiency in practical applications.

TABLE 4.10: Effects of decoding levels on closed-loop planning

Level	Closed-loop testing	
	Success rate (%)	Inference (ms)
0	89.5	31.8
1	92.25	44.1
2	94	56.7
3	94.5	66.5
4	94.5	79.2

4.6 Conclusions

We introduce *GameFormer*, a Transformer-based model that utilizes hierarchical game theory for interactive prediction and planning. By leveraging level-k game theory, our approach introduces level-k decoders that iteratively enhance the prediction of future trajectories for interacting agents. Additionally, we have developed a learning process that effectively regulates the predicted behaviors of agents

based on the prediction results obtained from the previous level. Through extensive experimentation on the WOMD, we demonstrate the superior performance of our proposed model. Our results indicate that GameFormer achieves state-of-the-art accuracy in interaction prediction, and outperforms baseline methods in both open-loop and closed-loop planning tests. This highlights the efficacy of our model in capturing and forecasting complex interactive behaviors in various scenarios. Moreover, our proposed planning framework delivers leading performance on the nuPlan planning benchmark.

The practical implications of our work are significant, as the exceptional performance of GameFormer underscores its potential for enhancing the capabilities of AVs, making them more capable of understanding and responding to the behaviors of other agents on the road. Moving forward, there are several avenues for future research and improvement, particularly in the closed-loop planning tests. In particular, more rigorous methods should be explored to handle the multi-modal predictions generated by the model, instead of simply selecting the prediction with the highest probability. This would allow for a more comprehensive consideration of uncertainty and improve the robustness of the planning process.

Chapter 5

Learning World Model to Predict Human Response

In this chapter, the primary focus is on the development of world models that, rather than forecasting the trajectories of other human road users, anticipate human responses to the vehicle's decisions, in order to facilitate proactive, interactive, and human-like decision-making for autonomous driving systems. The application of such world models to learning-based behavior planning is emphasized. In contrast to existing learning-based approaches that directly generate decisions, this chapter introduces a predictive behavior planning framework that learns to predict and evaluate based on human driving data. The proposed framework comprises three integral components: a behavior generation module responsible for generating a diverse array of candidate behaviors in the form of trajectory proposals; a conditional motion prediction network tasked with predicting future trajectories of other agents conditioned on each proposal; and a scoring module that assesses the candidate plans using maximum entropy IRL. The effectiveness of the proposed framework is ascertained via comprehensive experiments on a large-scale, real-world urban driving dataset. The results indicate that the conditional prediction model is capable of predicting distinct and plausible future trajectories given different trajectory proposals, while the IRL-based scoring module can select plans closely resembling human driving patterns. In terms of similarity to human driving trajectories, the proposed framework surpasses other baseline methods. Moreover, the conditional prediction model demonstrates improvements in both prediction and planning performance compared to the non-conditional model. Finally, the

importance of learning the scoring module for aligning evaluations with those of human drivers is highlighted.

5.1 Introduction

Making human-like decisions is crucial for AVs because they need to operate among humans in a safe and socially-compliant manner. The modern autonomous driving stack divides the decision-making task into two sub-modules: behavior planning and trajectory planning [125]. Behavior planning determines high-level decisions such as lane changes, overtaking, and yielding, while trajectory planning generates smooth trajectories to accomplish these high-level objectives. Behavior planning plays a crucial role in the decision-making process because it informs downstream trajectory planning and directs the final control outputs. However, widely-used behavior planners rely on hand-engineered rules or finite-state machines to specify the vehicle's behaviors under various situations [12, 126]. This approach, although prevalent, is not scalable, as the rules become difficult to maintain and can conflict with each other as more scenarios are taken into consideration. Additionally, for complex real-world scenarios, such as intersections with complex road structures and many diverse traffic participants, it is challenging to design rules that align with human expectations. To address these limitations, end-to-end learning-based methods (e.g., imitation learning and reinforcement learning) that directly make decisions from perception results or raw sensors have gained popularity. However, these methods lack reliability, interpretability, and safety guarantees. On the contrary, this study proposes a predictive behavior planning framework [127] that predicts the behaviors of surrounding road users and evaluates the goodness of synthesized decisions using a cost function explicitly considering speed, comfort, and safety. The framework is believed to be closer to the human decision-making process, rendering it more scalable, robust, and suitable for real-world scenarios. Our framework aims to address the scalability issue by learning from a variety of real-world scenarios and making decisions that closely resemble human driving. The main metric used to measure human likeness is the displacement error compared to human driving trajectories.

We focus on learning the prediction model and evaluation model in the framework, and we try to address two specific challenges. The first challenge is to

improve the prediction accuracy and make the prediction results more suitable for the downstream task. Deep learning-based approaches [16, 85, 128] have demonstrated significant improvements in accuracy compared to conventional methods (e.g., kinematic models and intelligent driver model). However, most motion prediction models are only passively used in behavior planning, meaning the prediction model often outputs fixed results for other agents and ignores the potential impact of the ego vehicle's future actions. This can often lead to overly conservative and even dangerous decisions. To overcome this issue, we leverage the conditional motion prediction (CMP) method [87, 94, 129] in our behavior planning framework, which jointly predicts the future motions of multiple interacting agents in a scene based on the AV's candidate decisions. This provides the evaluation module with more accurate information, enabling improved evaluation of the consequences of different decisions. The second challenge is to design the cost function, which determines which behaviors are desirable, as human driving behaviors exhibit many hard-to-specify nuances (e.g., speed profiles, comfort, and risk preference). Manually tuning the cost function can be laborious and may not reflect the actual human preferences, resulting in unintended behaviors. To resolve this, we employ a maximum entropy inverse reinforcement learning (IRL) framework [57] to automatically learn the cost function from human driving data.

Our proposed conditional predictive behavior planning framework consists of three main components: generation, prediction, and scoring. The trajectory **generation** module uses a performant and interpretable approach to create diverse trajectory proposals for the AV, taking into account factors such as lane, traffic rules, and speed profiles. The conditional motion **prediction** (CMP) module then forecasts the future trajectories of surrounding agents based on each trajectory proposal and generates different scene-compliant trajectories. The CMP module in our framework is inspired by [93], but we propose a novel Transformer-based structure in observation encoding and plan fusion. The **scoring** module with a learnable cost function evaluates these trajectory proposals to make final decisions. Specifically, the factors considered in the cost function encompass travel efficiency (speed), ride comfort (longitudinal jerk and acceleration and lateral acceleration), risk (headway and lateral distance), and safety (collision probabilities with other vehicles). The main contributions of this study are summarized as follows:

- We propose a learning-based behavior planning framework that learns to

predict conditional multi-agent future trajectories and evaluate decisions from real-world human driving data.

- We propose a novel Transformer-based conditional motion prediction model that utilizes the attention mechanism for effective environment encoding and plan fusion.
- We propose a two-stage learning process where the conditional prediction model is trained first and used as an environment model in the learning of the cost function with maximum entropy IRL.
- We conduct comprehensive experiments to demonstrate the ability of our framework to make conditional and multi-modal predictions and human-like decisions.

5.2 Literature Review

5.2.1 Decision-making for autonomous driving

In addition to perception ability [130], decision-making has become a research hotspot for autonomous driving in recent years, owing to its importance as a bottleneck for the widespread deployment of autonomous vehicles. In particular, learning-based decision-making methods such as imitation learning (IL) and reinforcement learning (RL) have shown promising results and potential [131]. IL attempts to directly imitate the actions of a human driver through the use of holistic neural networks and massive offline driving datasets [20, 110, 132]. However, an evident issue with IL is the distribution shift from training to deployment, which is difficult to mitigate. Conversely, RL tries to optimize a reward function by interacting with the environment and learning from trial and error but still faces obstacles such as sample efficiency, accurate environment modeling, and proper reward function design [21, 133–137]. Despite these ongoing efforts on learning-based methods, IL and RL have their inherent flaws, and a fundamental problem is that their safety, interpretability, and generalizability are somehow compromised. As a result, there has been a shift towards classic planning methods (e.g., graph search, sampling, and optimization) [138], which provide stronger safety guarantees, rule compliance, and interpretability. Nevertheless, their performance heavily relies on

the prediction accuracy of surrounding agents and proper evaluation of the planned behaviors, which are the main focuses of this study.

5.2.2 Motion prediction

There has been a growing body of deep learning-based motion prediction approaches thanks to the wide availability of large-scale driving datasets [15, 139]. They have demonstrated excellent accuracy and scalability due to their ability to handle high-dimensional map data and model agent interactions. [84] proposed a vectorized high-definition map representation, which is more compact and easy to process than an image-based map. To model the agent-agent and agent-map interactions, the agents and map vectors are often abstracted into a graph and processed by graph neural networks (GNNs) [106, 140] or Transformer networks [16, 87]. In addition, the prediction model should be capable of outputting multiple possible futures (to address uncertainty) for multiple surrounding agents in the scene (for efficient inference and better scene consistency) [87, 141].

However, most of the motion prediction models ignore the influence of the AV's future actions on other agents, and the decision-making module has to act passively. This is problematic because other agents may react differently if the AV makes different decisions, and ignoring this may result in overly conservative or aggressive behaviors. More recently, conditional motion prediction (CMP) has emerged to address this issue and enable more interactive prediction. [94] proposed a CMP model that predicts future trajectories for other agents conditioned on a query future trajectory for an ego agent, which shows a 10% improvement in accuracy over non-conditional prediction. M2I [96] extended CMP to multi-agent interactive prediction, where an influencer is selected and reactors' future trajectories are predicted using a CMP model according to the influencer's marginal prediction result. Scene Transformer [87] proposed a unified Transformer-based architecture with a masking strategy, enabling one to predict other agents' behaviors conditioned on the future trajectory of the AV. However, these works still focus on the prediction task, and CMP models are not integrated into planning. PiP [93], which conditions the prediction process on multiple candidate future trajectories of the AV, is the most related to our study. However, the planning performance was not

investigated, probably due to the difficulty in evaluating the candidate trajectories, and the model is only validated on highway datasets. Our method is more focused on improving the downstream planning performance and validated on more challenging urban driving datasets.

5.2.3 Inverse reinforcement learning

Inverse reinforcement learning (IRL) methods aim to learn underlying cost functions from expert demonstrations, thereby avoiding manual specification. The maximum entropy IRL approach [57], which addresses ambiguities or uncertainties inherent in human demonstrations, has become popular in autonomous driving applications. [69] applied maximum entropy IRL to learn individual driving styles from highway driving demonstrations and reproduce distinct driving policies. [25] combined maximum entropy IRL with a planning algorithm to automatically tune the cost function, which exceeds the level of manual expert-tuned cost functions. However, they either ignored other agents (which is not practical) or used very simple models (e.g., constant speed) to predict other agents' actions. In Chapter 3, we propose using sampling-based IRL to infer cost functions for human drivers and construct a simple environment model that uses log replay and an intelligent driver model to simulate other agents' actions when they are influenced. In this chapter, we improve the prediction module with a neural network-based conditional prediction model and integrated it into a decision-making module based on IRL scoring, which allows the framework to extend to complex urban driving scenarios.

5.3 Methodology

The proposed behavior planning framework is illustrated in Fig. 5.1 and consists of three core components: behavior generation, conditional motion prediction, and IRL scoring. The behavior generation component synthesizes a diverse set of trajectory proposals by sparsely sampling plausible maneuvers and velocity profiles, based on the current state and reference route of the AV. The conditional motion prediction component predicts the trajectories of surrounding agents under each planned trajectory, providing multiple possible futures for both the AV and other agents. The features of the different trajectory proposals (including cumulative

step-wise features over the time horizon and trajectory-level features) are combined with learnable weights to calculate the costs, and the probabilities of trajectory proposals are obtained according to maximum entropy IRL. The behavior (trajectory proposal) with the highest probability or sampled from the distribution can be used as a reference trajectory in the downstream trajectory planning, which further refines the coarse trajectory and generates a well-defined trajectory for the autonomous vehicle to follow. In this study, we omit the trajectory planning module since it is well-established. In the following, we will first formulate the behavior planning problem and then elaborate on the key components of our framework.

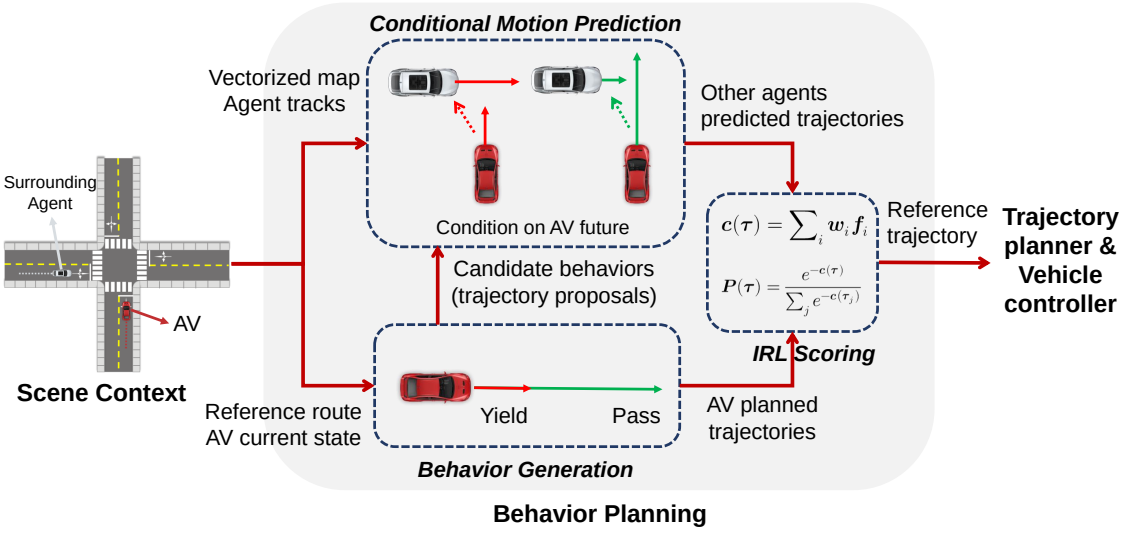


FIGURE 5.1: The proposed learning-based behavior planning framework

5.3.1 Problem formulation

Consider an arbitrary driving scene consisting of $N+1$ agents, where the AV is denoted as A_0 and other surrounding agents are denoted as A_i $(i = 1, \cdots, N)$. Let $\mathbf{s}_t^i$ represent the physical state of agent i at time step t. Assuming the current time step $t = 0$, $\mathbf{S}^i_{-T_h:0} \doteq \{\mathbf{s}^i_{-T_h}, \cdots, \mathbf{s}^i_0\}$ denotes the historical states over a past horizon of length T_h. Given the joint historical states of surrounding agents $\mathbf{X} \doteq \mathbf{S}^{1:N}_{-T_h:0}$, as well as the map information $\mathcal{M}$, the conventional motion prediction task is to model the posterior distribution $p(\mathbf{Y}|\mathbf{X}, \mathcal{M})$, where $\mathbf{Y} \doteq \hat{\mathbf{S}}^{1:N}_{1:T_f}$ denotes the joint states of surrounding agents over a future time horizon T_f. For conditional motion prediction, we incorporate additional information on the AV's potential future trajectory $\mathbf{P} \doteq \{\hat{\mathbf{s}}^0_1, \cdots, \hat{\mathbf{s}}^0_{T_f}\}$, which is to model the conditional distribution $p(\mathbf{Y}|\mathbf{X}, \mathcal{M}, \mathbf{P})$.

Given the planning and conditional prediction results $\mathbf{P}$ and $p(\mathbf{Y}|\mathbf{P})$, we utilize a cost function f to score the results and make decisions. In particular, let $r(\mathbf{P})$ represent the reward (negative cost) of the planned trajectory and $r(\mathbf{P}) = -f(\mathbf{P}, p(\mathbf{Y}|\mathbf{P}); \mathbf{w})$, where $\mathbf{w}$ is the learnable weights. The probability of a planned trajectory being selected according to the maximum entropy principle is $P(\mathbf{P}_i) = \frac{\exp r(\mathbf{P}_i)}{\sum_j \exp r(\mathbf{P}_j)}$. We adopt the IRL algorithm to learn the weights $\mathbf{w}$ from expert demonstrations, which is to maximize the log-likelihood of the expert demonstration trajectory $\mathbf{P}^*$ being selected: $\max_{\mathbf{w}} \log P(\mathbf{P}^*|\mathbf{w})$.

5.3.2 Behavior generation

We generate different candidate behaviors for the AV in the format of trajectory proposals. This is done by considering the road structure (available lane centerlines), semantics (changing lanes and lane-keeping, as well as different speed profiles), and traffic rules (speed limit). The trajectory proposals are generated by utilizing polynomial curves given the target lane and speed. This process ensures that the generated trajectories are dynamically feasible and the behaviors are correct, interpretable, and compliant with traffic rules. To adapt to complex road structures in urban scenarios, we represent a trajectory in the Frenet Frame of the reference path (e.g., the lane centerline) [142] and then translate it back to the Cartesian coordinate system. We consider a fixed time horizon $T = 5s$ because it can balance prediction accuracy and long-term decisions, and satisfy the requirement of generating coarse trajectory proposals. Specifically, for the longitudinal s-axis, we specify a set of target velocities at the terminal state ranging from braking to stop ($\dot{s}(T) = 0$) to accelerating to the speed limit ($\dot{s}(T) = v_{limit}$), while the target acceleration is fixed to be 0 ($\ddot{s}(T) = 0$). We use a quartic polynomial to parameterize the longitudinal states along the trajectory:

$$s(\tau) = a_0 + a_1\tau + a_2\tau^2 + a_3\tau^3 + a_4\tau^4, \tag{5.1}$$

where τ is the time, $\{a_0, \cdots, a_4\}$ are the coefficients. Given the initial state $[s(0), \dot{s}(0), \ddot{s}(0)]$ and target state $[\dot{s}(T), \ddot{s}(T)]$, we can calculate the coefficients and get the longitudinal state at each time step on the trajectory.

For the lateral d-axis, we assume the AV completes the maneuver at the end of the time horizon, i.e., $d(T) = 0$, $d(T) = D$, or $d(T) = -D$, where D is the distance to

the neighbor lanes, and fix the end conditions $\dot{d}(T)$ and $\ddot{d}(T)$ to be 0. We use a quintic polynomial to represent the lateral states:

$$l(\tau) = b_0 + b_1\tau + b_2\tau^2 + b_3\tau^3 + b_4\tau^4 + b_5\tau^5, \tag{5.2}$$

where $\{b_0, \cdots, b_5\}$ are the coefficients of the polynomial. Likewise, given the initial state $[d(0), \dot{d}(0), \ddot{d}(0)]$ and target state $[d(T), \dot{d}(T), \ddot{d}(T)]$, we can calculate the coefficients and obtain the lateral states.

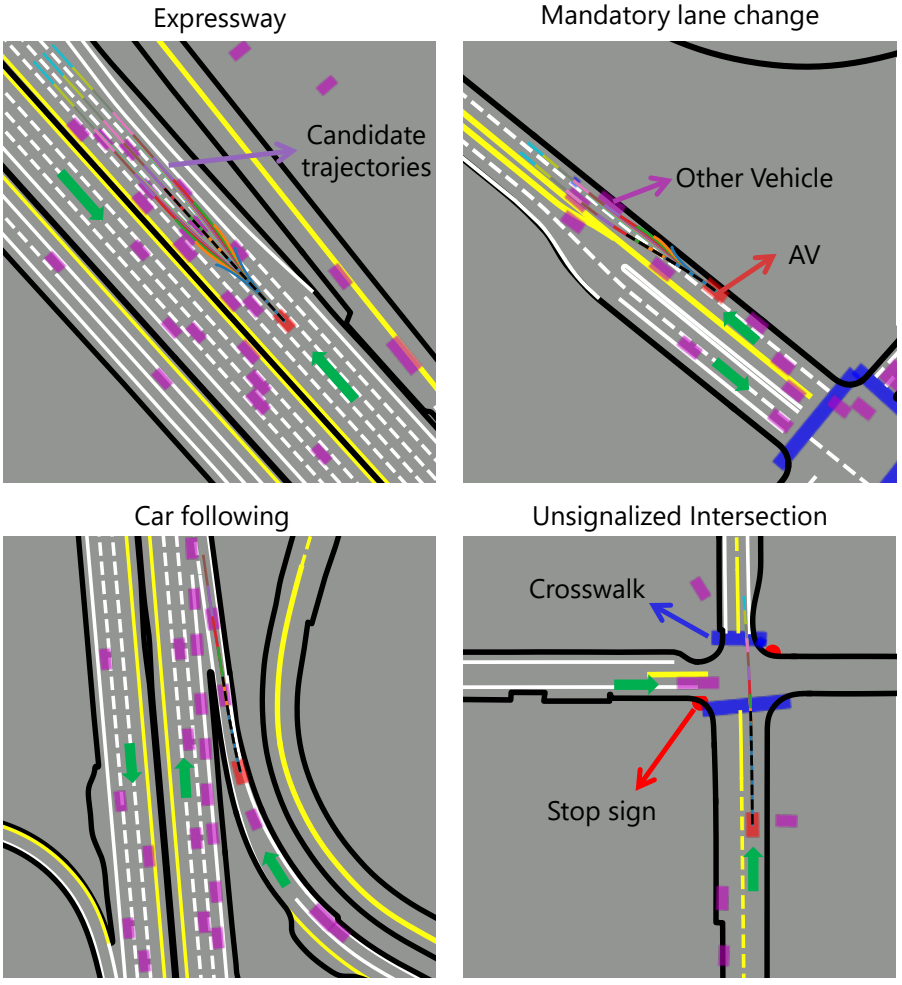


FIGURE 5.2: Representative examples of the behavior generation process in a variety of urban driving scenarios

The final trajectory is represented by a sequence of states with regard to the s and d axis $[s(\tau), d(\tau)]$, which is then translated back to the Cartesian coordinate $[x(\tau), y(\tau)]$ given the reference route's coordinate, heading angle, and curvature [142]. Approximately $10 - 30$ candidate behaviors (trajectory proposals) are generated depending on the AV's initial state and road structure. Fig. 5.2 shows

some representative cases of the module generating candidate behaviors for the AV according to different road structures. The examples provided demonstrate the capabilities of our method in handling various scenarios in urban driving. Given the candidate lane centerlines, our proposed method is capable of addressing nearly all scenarios on structured roads. More details about the behavior generation process can be found in section 5.4.2.

5.3.3 Conditional motion prediction

We utilize a conditional motion prediction module to predict the future motions of other agents conditioned on the AV's planned trajectory. The prediction module needs to address the inherent uncertainties (multi-modality) of other agents' intentions, and consequently predict a diverse set of future outcomes for other agents. Particularly, we represent the future outcomes as a Gaussian mixture model (GMM), where each mixture component corresponds to a joint future state sequence for all other agents. The state of a multi-agent joint future at each time step is represented as a Gaussian distribution:

$$\phi(\hat{\mathbf{S}}_t^{1:N}|\mathbf{P}) = \mathcal{N}(\mu_t(\mathbf{P}), \Sigma_t(\mathbf{P})), \tag{5.3}$$

where μ_t and Σ_t are the mean and covariance, respectively. The agents' historical states $\mathbf{X}$ and map information $\mathcal{M}$ are omitted for conciseness.

We model a probabilistic distribution (GMM) over multiple joint future outcomes, which represents the probability over each predicted future:

$$p(\mathbf{Y}|\mathbf{P}) = \sum_{k=1}^{K} p(\mathbf{Y}^k|\mathbf{P}) \prod_{t=1}^{T_f} \phi(\hat{\mathbf{S}}_t^{1:N}|\mathbf{P}, \mathbf{Y}^k), \tag{5.4}$$

where K is the number of the mixture components, $p(\mathbf{Y}^k)$ is the probability of the k-th component.

The multi-step means and covariances of each future, as well as its probability, are learned parameters. We design a Transformer-based neural network to fulfill the conditional prediction task. The architecture of the prediction network is shown in Fig. 3.5, and the details of the building blocks are described below.

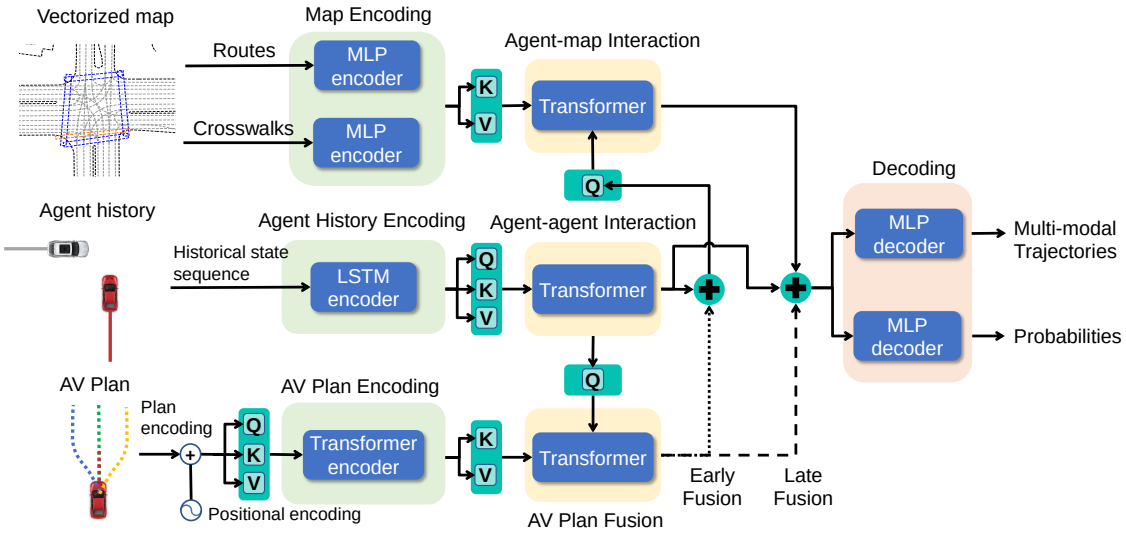


FIGURE 5.3: The structure of the conditional motion prediction network

Input Representations. The prediction network utilizes three diverse sources of information: vectorized map, agent history, and AV plan. In practice, these types of information are readily available in a modern autonomous driving system. The agent history tracks can be obtained by the object detection and tracking function in the perception system, and the vectorized map can either be obtained from a high-definition map created offline or from an onboard mapping system. The candidate plans for the AV are generated using the proposed method. For each agent, we find its possible driving routes stretching a predefined length (100 meters) and extract the waypoints with a fixed interval (1 meter), as well as its nearby crosswalk polylines.

Agent History and Map Encoding. The historical states of all agents (including the AV) are encoded by different dual-layer long short-term memory (LSTM) networks for different types of agents (i.e., vehicle, pedestrian, and cyclist). For each agent, we apply two multi-layer perceptrons (MLPs) to encode the driving routes and crosswalks respectively. A map waypoint is comprised of spatial attributes, such as its position and heading angle, as well as additional attributes like speed limit and traffic signals, which are effectively encoded by the MLP as the latent feature of that waypoint. More details about the encoders can be found in [113].

AV Plan Encoding. A candidate behavior is represented as a trajectory proposal, which is filled into a tensor with shape (T_f, D_p). The future state D_p of a planned trajectory consists of the x and y coordinates, heading angles, and speed.

Specifically, the trajectory is projected to a high dimension with an MLP and the positional encoding is added, and then we employ a self-attention Transformer encoder layer [143] to extract the temporal relation of the trajectory, obtaining the encoding of a planned trajectory with shape (T_f, D). The same encoder is applied to all trajectory proposals.

Interaction Modeling. The agent-agent interaction is modeled by a two-layer self-attention Transformer encoder, which takes as input the agents' historical state encoding and outputs the feature of relations between them, and we remove the feature of the AV and obtain the agent interaction encoding with shape (N, D). For each agent in the interaction encoding, we propose an agent-map encoder (two cross-attention layers and a multi-modal attention layer [16]) to generate different modes of agent-map interaction features with shape (K, D). All the surrounding agents share the same agent-map interaction encoder. More details about the interaction modeling encoders can be found in [113].

AV Plan Fusion. Injecting the AV's planned trajectory as conditional information into other agents' encoded feature to predict their future trajectories is an essential part of the model. To fuse that information, we design two viable approaches, namely early fusion and late fusion. We first use a cross-attention Transformer layer with the agent-agent interaction features as query and AV plan encoding as key and value and obtain a tensor with shape (N, D) representing the influence of the AV's future plan on different agents. In the early fusion approach, an attention-based fusion mechanism is designed by combining the agent-agent interaction feature and AV's future influence as the query to the agent-map interaction module. In contrast, the late fusion approach simply concatenates the feature of the AV's future plan with the final encoding, which is used to decode the future trajectories. We have also combined early fusion and late fusion (denoted as early + late fusion), and the effectiveness of the different fusion methods will be evaluated through experiments.

Decoding. We repeat the agent interaction encoding K times along the zeroth dimension and concatenate it with the agent-map interaction encoding of all agents (K, N, D), to generate a final latent representation tensor with shape $(K, N, 2D)$. It contains the necessary information to predict an agent's future motion, including its historical physical states, interaction with other agents, relation with the map, and the influence of the AV's plan if using early fusion. For late fusion, the influence

of the AV's future plan on other agents is also concatenated to the latent representation tensor, yielding a tensor with shape $(K, N, 3D)$. We use an MLP to decode the Gaussian parameters $\mathcal{N}(\mu_x, \sigma_x; \mu_y, \sigma_y)$ for every agent at every timestep in the future from the latent representation, outputting a tensor with shape $(K, N, T_f, 4)$. To predict the probability of different futures, which is a tensor with shape $(K, 1)$, we first use max-pooling to aggregate the information of all agents and then pass the obtained tensor with shape $(K, 2D)$ or $(K, 3D)$ through another MLP.

5.3.4 Inverse reinforcement learning

Although we have obtained the candidate behaviors and other agents' predicted reactions through the CMP module, it is still challenging to appropriately evaluate the behaviors and make human-like decisions. This is because driving behaviors involve many hard-to-specify nuances, and different individuals may take different actions even in the same situation. Therefore, we adopt the maximum entropy IRL method [57] to learn to evaluate these behaviors from human driving data.

Maximum entropy IRL aims to recover the underlying reward functions from demonstrations of human behaviors. It can address the ambiguity of multiple solutions and the stochasticity of expert behaviors by recovering a distribution over all trajectories. In essence, according to the principle of maximum entropy, the resulting probability distribution over candidate behaviors (trajectories) is:

$$P(\zeta_i) = \frac{e^{-c(\zeta_i)}}{\sum_j e^{-c(\zeta_j)}}, \tag{5.5}$$

where ζ_i is a trajectory among the set of candidate trajectory proposals, and $c(\zeta_i)$ is the cost of that trajectory.

To maintain interpretability, we use a linear cost function, which is a linear combination of different features that characterize the driving behavior, and the cost of a candidate trajectory is the weighted sum of these features:

$$c(\zeta) = \mathbf{w}^\top \mathbf{f}(\zeta) = \sum_i w_i f_i(\zeta), \tag{5.6}$$

where $\mathbf{w}$ is the weights of the cost function and $\mathbf{f}(\xi)$ is the feature vector of the trajectory. The features of a trajectory are designed to cover the major concerns

of autonomous driving, such as travel speed, ride comfort, traffic rules, and most importantly safety, which are detailed in section 5.4.3.

The objective of IRL is to optimize the cost function weights in order to maximize the log-likelihood of the expert demonstration trajectories $\zeta^* \in D$ in the dataset:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \sum_{\zeta^* \in D} \log P(\zeta^* | \mathbf{w}). \tag{5.7}$$

We can use a gradient-based optimization method with automatic differentiation tools to optimize the weights.

5.3.5 Learning process

The learning process is divided into two stages. The first stage deals with conditional motion prediction, which involves learning to predict other agents' behaviors conditioned on the AV's future trajectory from a massive amount of interactions among human drivers. Since we can only get access to the ground-truth trajectories for the AV and other agents, we use the AV's ground-truth future trajectory as the planned trajectory, and other agents' joint future trajectories are predicted and conditioned on the information. Although we cannot exactly learn how other agents would react to the AV's different plans in a specific scenario from an offline dataset, the CMP model is capable of predicting reactive and different behaviors of other agents given different plans. This is because the model can learn the reactions of other agents in a wide variety of scenarios where the AV takes different plans, and then generalize to predict other agents' reactions to different plans of the AV in a specific scenario.

To train the CMP module (a deep neural network parameterized by θ), for each agent at a specific future time step, we adopt the negative log-likelihood (NLL) loss on the GMM parameters:

$$\begin{aligned} \mathcal{L}_{CMP} &= -\log \mathcal{N}_{\hat{k}}(Y_x - \mu_x, \sigma_x; Y_y - \mu_y, \sigma_y; \rho = 0) \\ &= \log \sigma_x + \log \sigma_y + \frac{1}{2}\left(\left(\frac{Y_x - \mu_x}{\sigma_x}\right)^2 + \left(\frac{Y_y - \mu_y}{\sigma_y}\right)^2\right) - \log p_{\hat{k}}, \end{aligned} \tag{5.8}$$

where (Y_x, Y_y) represents the ground-truth position, $\hat{k}$ is the selected predicted future with the closest joint trajectories to the ground-truth ones measured by the L2 distance, $p_{\hat{k}}$ is the predicted probability of the selected Gaussian component. Note that we take the joint loss formulation by aggregating the losses of all agents jointly in the best-predicted future, meaning all agents have the same best prediction mode $\hat{k}$. We adopt the cross-entropy loss in the above equation to maximize the probability of the selected Gaussian component.

In the second stage, we concentrate on learning the cost function weights for evaluating the candidate plans. After obtaining the well-trained CMP module, which outputs the possible trajectories of other agents given a planned trajectory, we can query the module for all the generated plans and obtain a set of future predictions. We then calculate the features and costs of all these futures (including features of the AV's trajectory and safety features considering other agents), and consequently, the distribution of planned trajectories. We impose the NLL loss on the distribution, which favors the trajectory that most closely matches the expert demonstration in feature space:

$$\mathcal{L}_{IRL} = -\sum_{m=1}^{M} \mathbf{1}(m = \hat{m}) \log P(\mathbf{P}_m|\mathbf{w}), \tag{5.9}$$

where $\mathbf{P}_m$ is a planned trajectory, $\mathbf{w}$ is the learnable weights, M is the number of generated plans, and $\hat{m}$ is the index of the plan with the closest end-point distance to the ground-truth trajectory.

5.4 Experiments

5.4.1 Dataset

We train and validate the proposed framework (CMP and IRL modules) on the Waymo Open Motion Dataset (WOMD) [144], a large-scale real-world driving dataset containing $104,000$ unique scenes (each 20 seconds long at 10 Hz) collected from 570 hours of driving and over 1750 km of roadways. The WOMD dataset provides annotated high-definition map data (e.g., lane polylines, lane connectivity, speed limits, and traffic signal states) and high-accuracy agent track data (e.g.,

coordinates, heading angles, velocities, and bounding box sizes), which is suitable for both prediction and planning tasks.

In our experiments, we select 10,156 scenes from the dataset, allocating 80% for training data and the remaining 20% for testing data. In each scene, we split the 20-second long track data into several 7-second tracks with a sliding window, where the observation horizon is 2 seconds and the prediction/planning horizon is 5 seconds into the future. Each scene includes one track labeled as the self-driving car, which we choose as the AV for behavior planning, while its surrounding traffic participants are the agents to predict. We only utilize the AV track in each scene to train the IRL scoring module because they do not contain aggressive and unsafe behaviors. Eventually, we obtain 113,622 training data points and 28,396 testing data points. For the CMP module, all the data points are used for model training and evaluation of the prediction performance. For the IRL scoring module, we focus on the high-level behavior planning task and thereby filter those data points where the AV's average speed is less than 3 m/s (e.g., waiting at a red light), as well as a portion of data points where the AV cannot make lane changes. The amount of training data for the IRL scoring module after the filtering is 10,564 and testing data for evaluating the planning performance is 2,246.

5.4.2 Behavior generation

To handle the complex road structures in urban areas, we generate candidate behaviors in the Frenet Frame of the given reference route, allowing us to separate the behaviors in the longitudinal and lateral directions. In the longitudinal direction, 10 target speeds are evenly sampled in the range $[0, v_{limit}]$, where v_{limit} is the speed limit of the route. In the lateral direction, according to the road structure, we specify the target lateral displacement to complete a lane change. For example, if there is a left lane adjacent to the reference route that allows lane changing, we set the lateral displacement to be D, which is the distance from the current lane to the centerline of the left lane. The number of target lateral displacements varies with the road structure, ranging from 1 (keep lane) to 3 (keep lane, change left, and change right). The different target speeds and lateral displacements are combined to generate candidate trajectories in Frenet space. These trajectories are then translated back to Cartesian space and fed to the CMP module.

5.4.3 Feature design

To maintain the interpretability of the planner, we design a set of representative features (scalar values) to characterize a candidate decision (trajectory). We compute the features for each candidate trajectory, and the cost of the trajectory is the sum of these features multiplied by the corresponding learnable cost weights: $c(\zeta) = \sum_i w_i f_i(\zeta)$. The designed features are described as follows.

Travel efficiency. We use the difference between the current speed and speed limit to represent travel efficiency while also obeying traffic rules. The difference is normalized by the value of the speed limit to balance high-speed and low-speed cases, and the feature takes the average value over all timesteps along the trajectory:

$$f_{\text{travel}}(\zeta) = \frac{1}{T_f} \sum_{t=1}^{T_f} \frac{|v_t - v_{limit}|}{v_{limit}}, \tag{5.10}$$

where v_t is the speed of the trajectory point at time step t, and v_{limit} is the speed limit of the road.

Maximum acceleration. The maximum acceleration (m/s^2) along the trajectory is used as a measure of ride comfort, which is denoted as:

$$f_{\text{acc}}(\zeta) = \frac{\max_t |a_t^{lon}|}{a_{\max}^{lon}}, \tag{5.11}$$

where a_t^{lon} is the longitudinal acceleration of the trajectory point at time step t, and $a_{\max}^{lon} = 5\ m/s^2$ is used to normalize this feature.

Maximum jerk. In addition to acceleration, we use the maximum jerk (m/s^3) along the trajectory to represent the ride comfort in the longitudinal direction:

$$f_{\text{jerk}}(\zeta) = \frac{\max_t |j_t|}{j_{\max}}, \tag{5.12}$$

where j_t is the longitudinal jerk of the trajectory point at time step t, and $j_{\max} = 10\ m/s^3$.

Maximum lateral acceleration. The maximum lateral acceleration (m/s^2) in the lateral direction is adopted as another measure of ride comfort:

$$f_{\text{lat\_acc}}(\zeta) = \frac{\max_t |a_t^{lat}|}{a_{\max}^{lat}}, \tag{5.13}$$

where a_t^{lat} is the lateral acceleration of the trajectory point at time step t, and $a_{\max}^{lat} = 5\ m/s^2$.

The computation of the following features requires an estimate of other road users' states in the future, which is given by the CMP module. Note that we take the mean values of the predicted Gaussian at each timestep as a road user's state in calculating these features.

Headway. The AV should keep a safe longitudinal distance to the leading vehicle, which is dependent on the speed of the AV. We employ the concept of time headway and define the headway feature as follows.

$$\begin{aligned} HW &= \frac{1}{K}\sum_k p_k \min_t \frac{\Delta s_t^k}{v_t}, \\ f_{\text{hw}} &= e^{-HW^2}, \end{aligned} \tag{5.14}$$

where Δs_t^k is the longitudinal distance between the AV and leading vehicle at timestep t according to k-th predicted future, and v_t is the speed of the AV at timestep t. We take the minimum value of time headway in the time horizon and average across all predicted futures by their probabilities. Then, we use a Gaussian radial basis function (RBF) to compute the headway feature, which aims to penalize the states the AV is too close to the leading vehicle.

Lateral distance. The AV should maintain a safe lateral distance from other vehicles, and thus we set up a feature to represent safety in the lateral direction.

$$\begin{aligned} LD &= \frac{1}{K}\sum_k p_k \min_t \Delta l_t^k, \\ f_{\text{ld}} &= e^{-LD^2}, \end{aligned} \tag{5.15}$$

where Δl_t^k is the lateral distance between the AV and the closest vehicle on the sides of the AV at timestep t. Likewise, the minimum lateral distance in the time horizon from each predicted future is obtained and then weighed by the probability

of the future. We also use a Gaussian RBF to compute the lateral distance feature, and if no other vehicles are on the sides, this feature is set to 0.

Safety. The safety feature explicitly considers the collisions between the AV's planned trajectory and other vehicles' predicted trajectories, as well as uncertainties. At each time step, we calculate if the AV's planned state violates the spatial occupancy of any other agents in a given predicted trajectory.

$$f_{\text{safety}} = \frac{1}{K} \sum_{k} p_k \sum_{t} \max_{i} \mathbf{1}_{overlap}(\hat{\mathbf{s}}_t^{i,k}, \hat{\mathbf{s}}_t^0), \tag{5.16}$$

where $\hat{\mathbf{s}}_t^{i,k}$ is the predicted state of agent i at timestep t in the k-th future, $\hat{\mathbf{s}}_t^0$ is the state of the AV, and $\mathbf{1}_{overlap}$ is an indicator function, which emits 1 if the bounding box of the AV overlaps with an agent's bounding box and 0 otherwise. If the AV collides with any other agents at timestep t, the frame is counted as a collision, and the collisions are summed across all time steps in the time horizon. The final safety feature averages the collision times from each predicted future weighted by their probabilities.

5.4.4 Evaluation metrics

To evaluate the prediction performance, two established metrics for behavior prediction are used, which are the minimum Average Displacement Error (minADE) and minimum Final Displacement Error (minFDE). minADE measures the average displacement of each point in the closest joint trajectories to the ground truth, while minFDE calculates the displacement error between the final point of the joint predicted trajectories and the ground truth. The prediction errors are averaged for all agents in the joint trajectories.

We use a set of metrics to evaluate the behavior planning performance (primarily focusing on the closeness to human driving trajectories): minFDE between the top-3 most likely planned trajectories and the ground-truth one, the accuracy of any of the top-3 most likely planned trajectories matching with the ground-truth one, and intention accuracy. We choose the top-3 accuracy because our planning framework is probabilistic, which can also address the stochasticity of human driving behaviors. In addition, we decrease the granularity of behaviors to discrete intentions, i.e., acceleration and deceleration in the longitudinal direction and lane change in

the lateral direction, and we calculate the accuracy of our model in identifying the intentions.

5.4.5 Implementation details

The parameters of the prediction module are listed in Table 5.1. For all Transformer modules in the network, the number of attention heads is 8, the hidden dimension of the feed-forward network is 1024, and the activation function is ReLU. Every dense layer, except for the output layer, is followed by a dropout layer with a dropout rate of 0.1. The network outputs the displacements relative to an agent's current position instead of the original coordinates, which could significantly improve prediction accuracy. We train the prediction module with the Adam optimizer, and the learning rate with an initial value of 2e-4 decays by a factor of 0.5 every 5 epochs. The batch size is 32, and the total number of training epochs is 30. We clip the gradient norm of the network parameters with the max norm set at 5.

TABLE 5.1: Parameters of the prediction module

Symbol	Meaning	Value
T_h	Length of historical timesteps	20
T_f	Length of future timesteps	50
D_p	Dimension of AV plan features	4
D	Dimension of embedding	256
N	Number of surrounding agents to consider	10
K	Number of predicted futures	3

For training the IRL-based planner (i.e., learning the cost function weights), we use the Adam optimizer with a learning rate that starts at 1e-2 and decays by a factor of 0.9 every 50 steps. We also add L2 regularization on the cost function weights with a weight decay value of 1e-2 to prevent overfitting. The mini-batch size is 64 and the total number of training steps is 500. To check if a collision occurs between the AV and another object at a specific timestep, we approximate each object via a list of circles given their poses. The circles from the AV and the other object are paired, and if the distance between any pair of circles' centers is smaller than a threshold, it is considered that the two objects intersect and thus a collision happens. More details about the collision indicator can be found in [145].

5.5 Results and Discussions

5.5.1 Prediction performance

We first evaluate the performance of the conditional motion prediction module and report the results in both quantitative and qualitative manners.

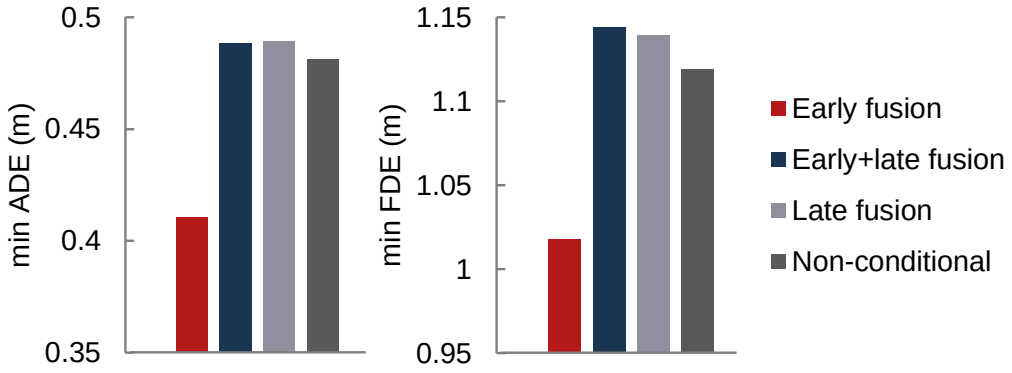


FIGURE 5.4: Quantitative results of the conditional prediction module with different fusion approaches

Quantitative results. Fig. 5.4 shows the quantitative prediction accuracy results in the testing set for different network structures. Here, we use the ground-truth future trajectory of the AV as its planned trajectory and feed it to the CMP network. The structure of the non-conditional prediction model is the same as our proposed prediction model, except that the AV plan encoding and fusion part is removed. The results demonstrate that the early-fusion structure significantly outperforms the others, showing an approximately 10% improvement in prediction metrics compared with the non-conditional prediction. This clearly suggests that leveraging future information of the controllable agent (AV) could allow the prediction module to be more informed and the results more accurate. Nevertheless, the structure of fusing the AV's future information needs careful design. Here, we investigate three fusion structures: the early fusion structure treats the AV's future information as part of the query to the agent-map interaction encoder; the late fusion setup feeds the AV's future information only at the final decoding stage; and the early+late fusion structure employs both early and late fusion approaches. As the results indicate, the late-fusion or early-late-fusion variant performs significantly worse than the early-fusion structure and even worse than the non-conditional prediction network. This implies that early fusion is a more effective structure in CMP modules, and late fusion may negatively influence the prediction results.

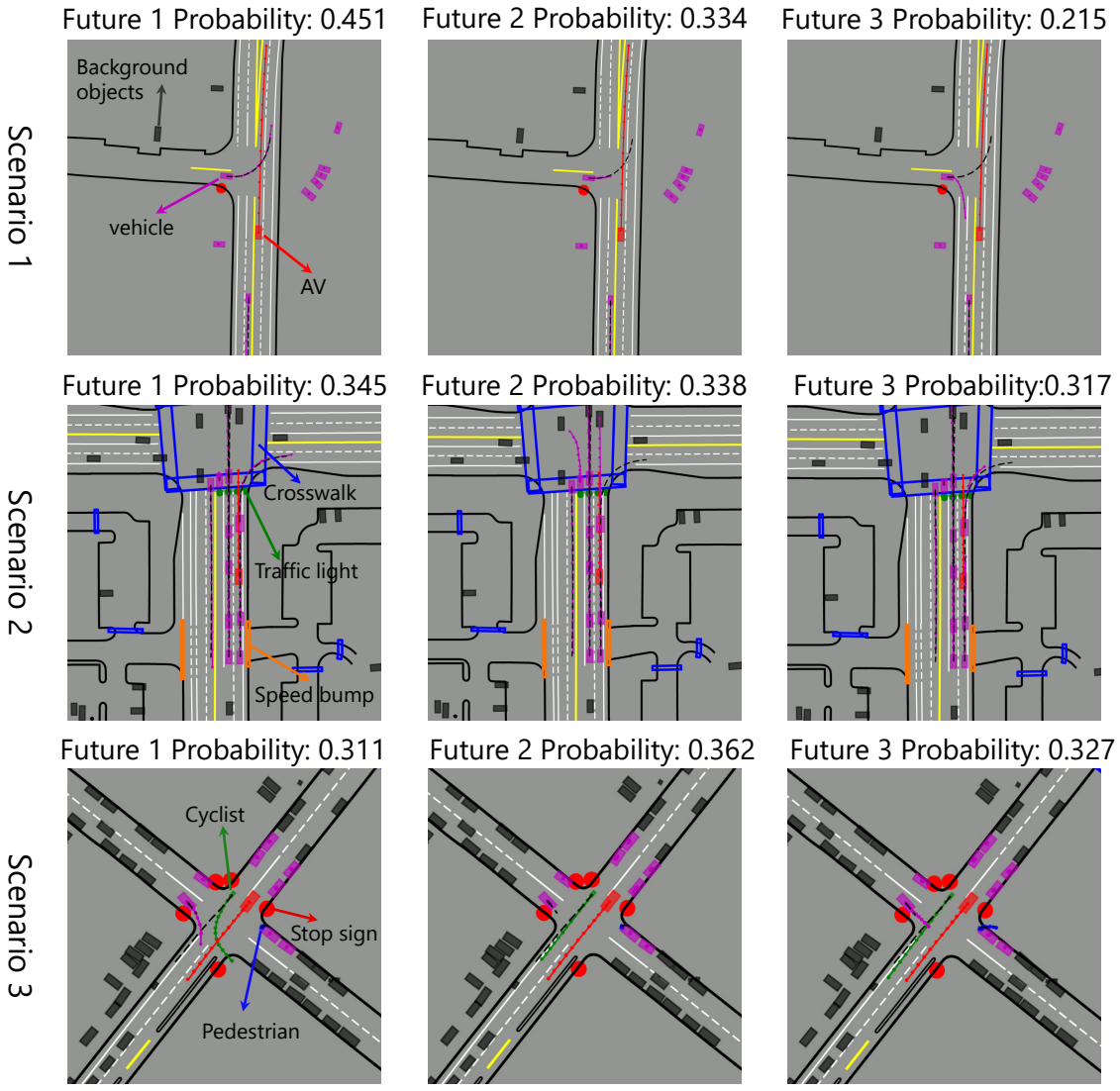


FIGURE 5.5: Qualitative results of the conditional prediction module to predict multiple futures for surrounding agents given a planned trajectory

Multi-future prediction. Since the early fusion CMP module achieves the best prediction performance, we demonstrate the network's ability to jointly predict multiple futures for surrounding agents based on the early fusion structure. Fig. 5.5 shows three representative driving scenarios, each with three possible predicted futures given a single planned AV trajectory, and we use the ground-truth AV trajectory as the conditional information input to the CMP model. The prediction model can capture the multi-modality of agents' behaviors in accordance with the road structure and generate other agents' joint future trajectories in a scene-consistent manner (i.e., no self-collisions between predicted trajectories). The model can also assign a probability to each predicted future, and the predicted future closer to the ground-truth one is assigned with a higher probability. The results reveal the

model's capability to handle the uncertainty of the future under the same future plan, thus enabling the downstream planner better evaluate the plan.

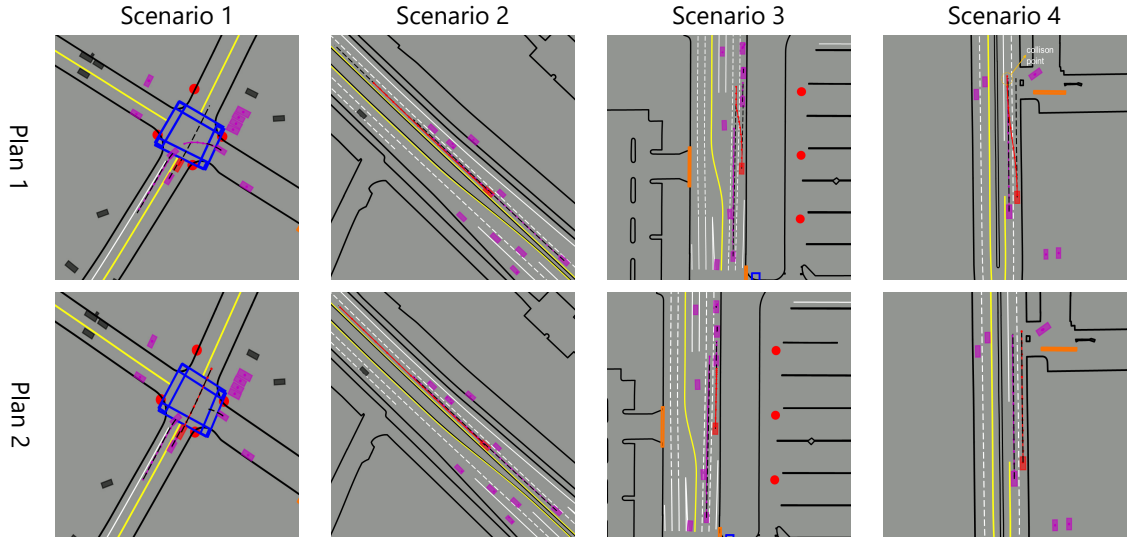


FIGURE 5.6: Qualitative results of the conditional prediction module to predict other agents' behaviors according to the AV's planned trajectory

Conditional prediction. We also adopt the early fusion structure as the prediction model and display the network's ability to predict other agents' behaviors conditioned on the AV's different plans in Fig. 5.6. Note that only the most-likely prediction result given an AV's plan is shown for clarity. In Scenario 1, the AV is interacting with a vehicle that may conflict with the AV's route at an unsignalized intersection. Under the yield decision (Plan 1, target speed is 0), the other vehicle is predicted to pass first and the vehicle behind the AV is predicted to stay still. When giving a pass decision (Plan 2, target speed is high) to the AV, the other vehicle is predicted to yield and the vehicle behind starts moving. In Scenario 2, an expressway, we can see that if the AV chooses to decelerate (Plan 1), the vehicle behind the AV is predicted to decelerate too and keep a safe distance (compared to Plan 2). In Scenario 3, the other vehicle is predicted to slow down if the AV plans a lane change to avoid the congested lane (Plan 1) and maintain its speed if the AV plans to decelerate in its current lane (Plan 2) without interfering with other vehicles. However, the model cannot completely make reactive predictions for other agents, which means other agents may not react to the AV's different plans and cause collisions in some plans, as shown in Scenario 4. This is because some of the AV's candidate plans deviate from the training data or normal behaviors and the model cannot make reactive predictions in such situations. Nonetheless, such

plans will be ruled out by the downstream planner, which encourages the planner to choose plans that comply with the training distribution from real-world data.

5.5.2 Planning performance

Qualitative results. We utilize the testing set with 2,246 urban driving scenes to evaluate the behavior planning performance of our method. We first display the qualitative results in Fig. 5.7, showing the proposed method's capability to predict other agents' trajectories and select appropriate behaviors in some representative driving scenarios. For each scenario, we present four candidate behaviors and their normalized scores. Only the predicted future with the highest probability is shown for clarity. Scenario 1 illustrates a car-following scenario on a multi-lane expressway, where the AV should keep a safe distance from the leading vehicle. Plan 1, which is the closest to the ground truth, has the highest score among candidate trajectories. Other candidate plans (e.g., Plan 2 with lower target speed and Plan 3 with higher target speed) have lower scores because they would lead to a smaller headway to the leading vehicle (unsafe behavior) or unnecessary speed loss. Changing lanes (Plan 4) is also unfavorable as it would induce unnecessary lateral discomfort without increasing the speed. In Scenario 2, the AV needs to deal with a cut-in vehicle from the right lane while interacting with other vehicles. Our method selects the lane-changing behavior (Plan 1) with the highest score because it safely avoids the collision risk with the cut-in vehicle and also speed loss. However, if the AV attempts hard braking to yield to the cut-in vehicle (Plan 2), there is a risk that the vehicle at the rear end could collide with the AV, and thus this plan has a near-to-zero score. Other candidate plans, such as slowing down without lane changing (Plan 3) and accelerating to overtake the cut-in vehicle (Plan 4), have low scores because they sacrifice the safety gap to the leading vehicle. In Scenario 3, slowing down and yielding to the cut-in vehicle (Plan 1 and Plan 2) have the higher scores because either accelerating (Plan 3) or changing lanes (Plan 4) could result in collision risks. The results suggest that the learned cost function can properly score different candidate plans according to the plan itself and also the conditional prediction result, and the behaviors (trajectory proposals) closest to ground truth can be assigned with the highest scores in most cases.

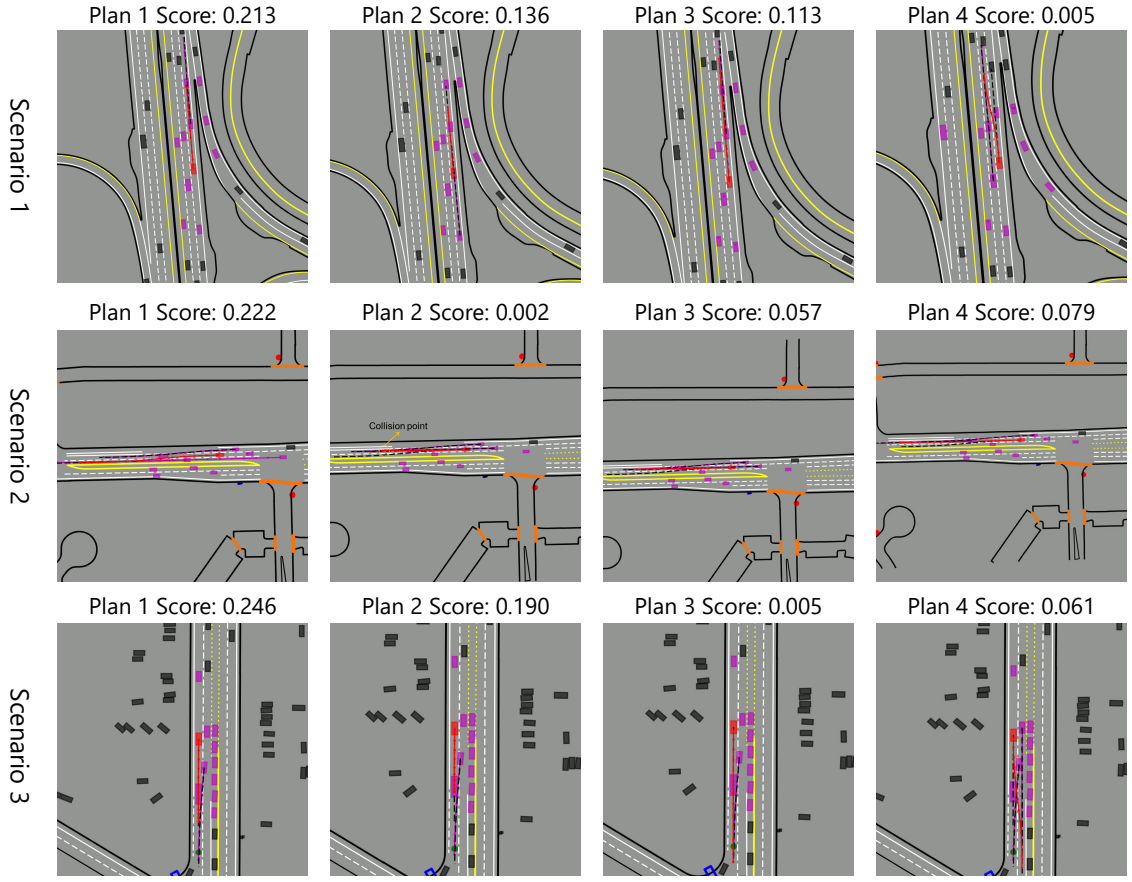


FIGURE 5.7: Qualitative results of the behavior planning module to properly score different candidate plans

Quantitative results. To compare the model's planning performance, we set up several baseline methods and apply them to the behavior planning task in the testing scenes. **Neural network classifier**: we build a neural network that takes as input the planned trajectories and predicted trajectories from the CMP module and directly outputs the score of each planned trajectory. The neural network is trained with the classification (cross-entropy) loss and the same training data for planning. **Neural network regressor**: we construct another neural network to directly output the target speed in the longitudinal direction and a value indicating lane change in the lateral direction. The network utilizes the backbone of the CMP module, which takes as input the information of the AV's and other agents' historical states and the map information. The network is trained with the mean squared error loss between the outputs and label speed and lane change indicator. **Model-based**: we use the intelligent driver model (IDM) to compute the desired speed and the minimizing overall braking induced by lane changes (MOBIL) algorithm to decide the lane-changing maneuver [146]. For the

neural network regressor and model-based methods, a trajectory can be obtained given the target speed and lane change. We compare the obtained trajectories from different methods against the ground-truth trajectories in each scene and report the results in Table 5.2 considering the evaluation metrics previously defined.

TABLE 5.2: Evaluation of the planning performance in comparison with baseline methods

	minFDE	Accuracy (%)	Speed Acc. (%)	Lane Acc. (%)
Model-based	7.59	–	75.15	75.63
NN regressor	5.66	–	91.08	85.56
NN classifier	2.91	68.61	**96.59**	89.91
Ours	**2.78**	**69.88**	95.40	**90.12**

The results in Table 5.2 reveal that our proposed method delivers human-like decision-making ability in terms of the position error to human driving trajectories and the accuracy of choosing closed-to-human behaviors. The performance of our method in making intention-level decisions is superior, reaching over 95% of accuracy in target speed and over 90% of accuracy in target lane compared to ground-truth human driving data. The NN classifier method, which can be regarded as deep IRL, attains similar performance to our approach. However, the interpretability of such a method is compromised. On the other hand, the performance of the NN regressor method is inferior, which shows the drawback of learning-based methods that directly output decision values, lacking interpretability and robustness. The model-based method performs the worst because they are based on simple mathematical formulations and rules and are thereby not applicable to complex urban driving scenarios.

5.5.3 Effects of the prediction model

We investigate the influence of the prediction module on the downstream planning performance of our method. We utilize different prediction models in the behavior planning framework and test the planning performance in the same testing driving scenes. In addition to the proposed conditional prediction model (with early fusion structure), other prediction models used are listed as follows. **Non-conditional**: we remove the AV plan encoding and fusion parts from the proposed conditional prediction model. **Model-based**: we use the constant turn rate and

velocity (CTRV) model to predict the surrounding agents' future trajectories, but the model's prediction accuracy is very limited. **Oracle**: we use the ground-truth future trajectories as the predicted trajectories of surrounding agents to reveal the upper bound of the influence of prediction accuracy. We report the primary planning evaluation metrics, i.e., minFDE and accuracy, and the results are given in Fig. 5.8.

The results indicate that prediction accuracy plays an important role in ensuring the downstream planning performance, and using a learning-based prediction model can significantly improve prediction accuracy and consequently planning performance compared to a kinematic-based prediction model. Moreover, using the conditional prediction model in the planning framework outperforms the non-conditional model, which emphasizes the benefit of leveraging the AV's future plan information in a prediction model. We also find that planning with the proposed conditional prediction model has comparable performance to the oracle method, which suggests that the conditional prediction model could better reflect the real-world interaction dynamics. However, planning errors still exist even when using the oracle model. First, the generated trajectory proposals are limited and coarse and cannot cover all the possible trajectories that a human driver may take. Second, the limitation of evaluation is that the IRL module only uses a linear cost function, which cannot fully reflect a human driver's actual evaluation of costs, and thus the scoring of the generated trajectories may not be accurate in some cases.

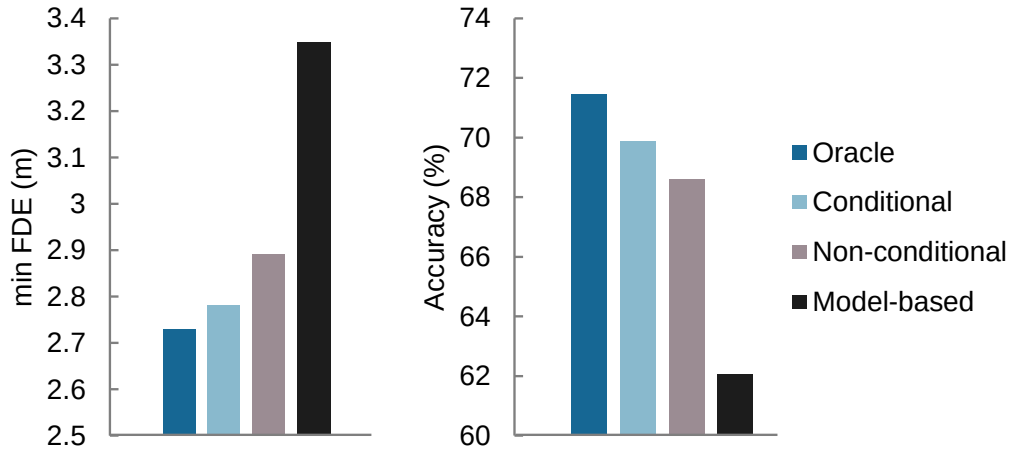


FIGURE 5.8: Evaluation of the influence of different prediction models on the planning performance

5.5.4 Effects of the cost function

We investigate the influence of the cost function to evaluate the candidate plans and the final planning performance. Here, we fix the prediction module as the proposed conditional prediction method and introduce two other baseline methods to obtain the cost function. **Manually tuned**: we manually tune the cost function weights according to a human expert's experience. **Maximum-margin**: we learn the cost function with the same training data using the max-margin method [147], which is also a popular IRL algorithm. The results of planning performance with different cost functions derived from different methods are summarized in Fig. 5.9.

We conclude that learning the cost function from data can significantly improve the evaluation of the candidate behaviors and human-likeness compared to using a manually tuned cost function. In addition, the maximum-entropy IRL method and the maximum-margin IRL method have similar planning performance, but the max-entropy method marginally outperforms the max-margin method. The results underscore the importance of learning the cost function from data rather than tuning it manually.

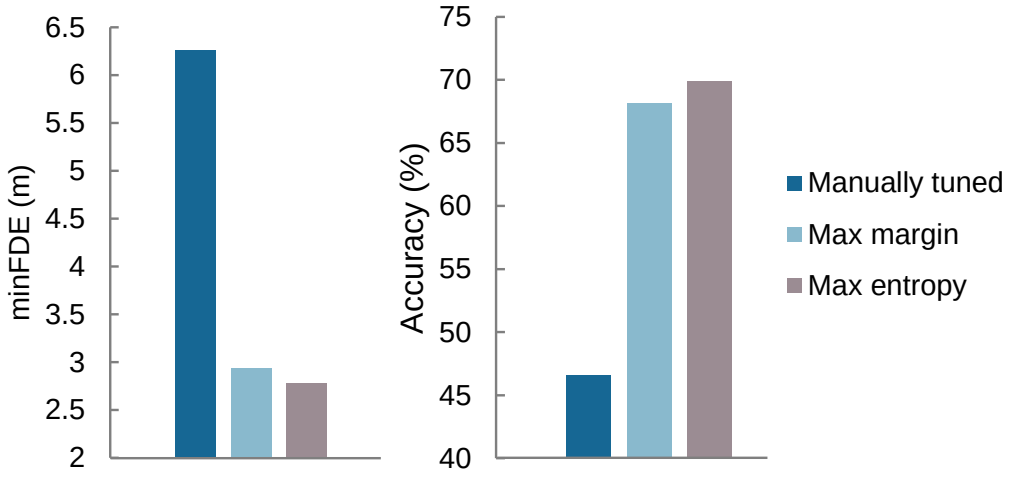


FIGURE 5.9: Evaluation of the influence of cost functions derived from different methods on the planning performance

5.5.5 Computation time

We compare the computation time of different methods, and all methods run on an NVIDIA RTX 3080 GPU. For the conditional prediction methods, there are two

inference approaches: 1) single, which means we query the prediction model for each planned trajectory; 2) batch, which means we organize all the planned trajectories into a batch and repeat the environmental context tensors to match with the plan queries. The results in Table 5.3 reveal that the batch processing method can significantly reduce the computation time by paralleling the conditional prediction process. The computation time is sufficient to satisfy the real-time requirement, as behavior planning runs at a lower frequency ($\leq$ 2 Hz). The single processing method that frequently queries the prediction model for multiple candidate plans has the longest computation time and is not suitable for real-time usage. The non-conditional method has the shortest computation time, but the planning and prediction performance is a trade-off. In addition, the early fusion method runs slightly faster than the late fusion method.

TABLE 5.3: Comparison of computation time of different methods

Method	Inference	Time (ms)
Non-conditional	–	68
Early fusion	Single	928
	Batch	145
Late fusion	Single	963
	Batch	149

We also report the computation time for learning the cost function using IRL. Note that IRL is only conducted offline, and the learned cost function is then directly used in online testing. The computational requirements for IRL are minimal, as only a few parameters in the cost function are learnable, but the computation of features may take up a large amount of time. Specifically, the computing time of a single IRL iteration (64 scenes in a batch) is approximately 28 seconds. Within this time, 9 s are used to query the conditional prediction module for the responses of other agents, and the remaining 19 s are spent on computing the feature vectors of different plans for all scenes across the batch. We iteratively feed different plans to the CMP model and use batch processing to obtain the prediction results and compute the features, to improve the computation efficiency. Nevertheless, computational time is not a major concern in offline learning, and only a few hundred iterations are required to finish the learning process of the cost function weights. When using the scoring module in online testing, computing the feature vector of one candidate AV plan requires 120 ms. To meet the real-time requirement, we can parallelize the computing process to obtain the feature vectors of

all candidate plans. Consequently, considering the computation time of generation, prediction, and scoring, our proposed framework can perform the behavior planning task within an acceptable time frame ($< 400\ ms$).

5.5.6 Discussions

The proposed framework in this study tackles the behavior planning task, which is one of the critical challenges in autonomous driving, by dividing it into prediction and scoring processes. Compared to other learning-based methods that only output the decision values, the proposed framework learns prediction and cost to evaluate candidate plans, which improves the safety, interpretability, and reliability of the system. Based on the overall framework, we propose a conditional motion prediction model that can forecast other agents' future trajectories according to the AV's potential future plan, which tightly couples the prediction and planning modules. Experimental results showed that the proposed framework outperforms traditional model-based methods and neural network-based methods that directly output decision values in terms of human-likeness and similarity to human behavior. The proposed conditional prediction model and learning the cost function with inverse reinforcement learning both play important roles to ensure the proper and human-like evaluation of candidate plans.

However, some limitations of this study should be acknowledged. One limitation is that the validation is performed in an open-loop manner because the low-level trajectory planner and controller are ignored. Future work will test the proposed behavior planner in a closed-loop simulator to fully manifest its capabilities. Additionally, this study did not investigate the performance of the proposed framework in safe-critical scenarios, such as encountering road obstacles or navigating complex intersections, which will be addressed in future work.

5.6 Conclusions

In this chapter, we propose a learning-based predictive behavior planning framework that consists of three core modules: a behavior generation module, a conditional motion prediction module, and a scoring module. Our proposed framework

aims to address one of the major challenges in autonomous driving, namely, behavior planning, by separating it into prediction and scoring processes. Compared to other learning-based methods, our framework learns both prediction and cost functions to evaluate candidate plans, which can lead to better safety, interpretability, and reliability of the system. We conduct extensive experiments on a large-scale real-world urban driving dataset to validate the proposed framework. The qualitative results demonstrate that the conditional prediction module can provide multi-modal futures given a candidate plan and reactive predictions to different plans. Furthermore, the scoring module, with the learned cost function, can select plans that are close to human-driving ones. The quantitative results suggest that the early fusion structure is the most effective for the conditional prediction model. Moreover, we find that the conditional prediction model not only improves the prediction accuracy but also facilitates the downstream scoring module to better evaluate candidate decisions, leading to more human-like behaviors. Lastly, we emphasize that learning the cost function is crucial in correctly evaluating the candidate plans to align with human values. Overall, we believe that our proposed framework represents a significant step forward in addressing the challenges of behavior planning in autonomous driving.

Chapter 6

Learning Differentiable and Integrated Framework

In this section, we present a unified framework that combines the three key modules in the decision-making framework. We propose a differentiable and integrated prediction-planning framework (DIPP) that can learn the cost function from data. Our framework utilizes a differentiable nonlinear optimizer as the motion planner, which takes predicted trajectories of surrounding agents provided by the neural network as input and optimizes the trajectory for the autonomous vehicle. This design allows all operations, including the cost function weights, to be differentiable. To train the proposed framework, we use a large-scale real-world driving dataset to imitate human driving trajectories in various driving scenes. We validate the framework in both open-loop and closed-loop manners. The open-loop testing results demonstrate that the proposed method outperforms the baseline methods across multiple metrics and produces planning-centric prediction results, enabling the planning module to generate trajectories similar to those of human drivers. In closed-loop testing, the proposed method surpasses various baseline methods, showcasing its capability to handle complex urban driving scenarios and its robustness against distributional shifts. Notably, we find that joint training of the planning and prediction modules achieves superior performance compared to planning with a separately trained prediction module, both in open-loop and closed-loop tests. Furthermore, the ablation study highlights the significance of the learnable components in the framework, as they are crucial for ensuring planning stability and optimal performance.

6.1 Introduction

Making safe, socially-compatible, and human-like decisions is a fundamental capability of AVs. While learning-based end-to-end decision-making methods, such as imitation learning and reinforcement learning [3], have enjoyed vigorous growth recently, they still lack interpretability, robustness, and safety compared to classic planning-based methods [138, 148, 149]. Nonetheless, to make safe and smooth plans in complex traffic scenarios, the key is to accurately forecast the future trajectories of surrounding traffic participants [16, 106, 150, 151]. The current autonomous driving stack treats prediction and planning as separate and sequential parts (see Fig. 6.1(a) traditional sequential prediction and planning), which means the prediction module is independent of the planning module. The problem with this setting is that prediction and planning tasks are highly interrelated and interdependent tasks, especially when the AV interacts with other traffic participants in urban areas. Moreover, another challenge with the traditional method is how to specify the cost function that can properly evaluate future motion plans and achieve a delicate trade-off between different requirements, e.g., collision avoidance, travel efficiency, and ride comfort. Manually tuning the parameters of the cost function is laborious and time-consuming, and might only be applicable in certain scenarios. Although some methods have been proposed to learn the cost function from driving data, such as continuous inverse optimal control (CIOC) [73, 152], sampling-based maximum-entropy inverse reinforcement learning (IRL) [19], as well as max-margin method [125], they are not coupled with the prediction module and rely on the impractical assumption that the prediction results are perfect. Some other methods have turned to a pure data-driven setting [87], which utilizes a holistic model to directly output planned AV trajectory while predicting other agents' future trajectories (see Fig. 6.1(b) end-to-end method), implicitly handling the interactions between agents. However, such methods lack safety guarantees and reliability for the decision-making task in safety-critical applications.

In this chapter, we propose a novel framework called differentiable integrated prediction and planning (DIPP), as shown in Fig. 6.1(c). The objective of the DIPP framework is to make the prediction module aware of the downstream planning task, so as to deliver planning-aware prediction results to the motion planner, without altering the established structure of the prediction-planning process. The principle of the DIPP framework is to implicitly change the prediction of other

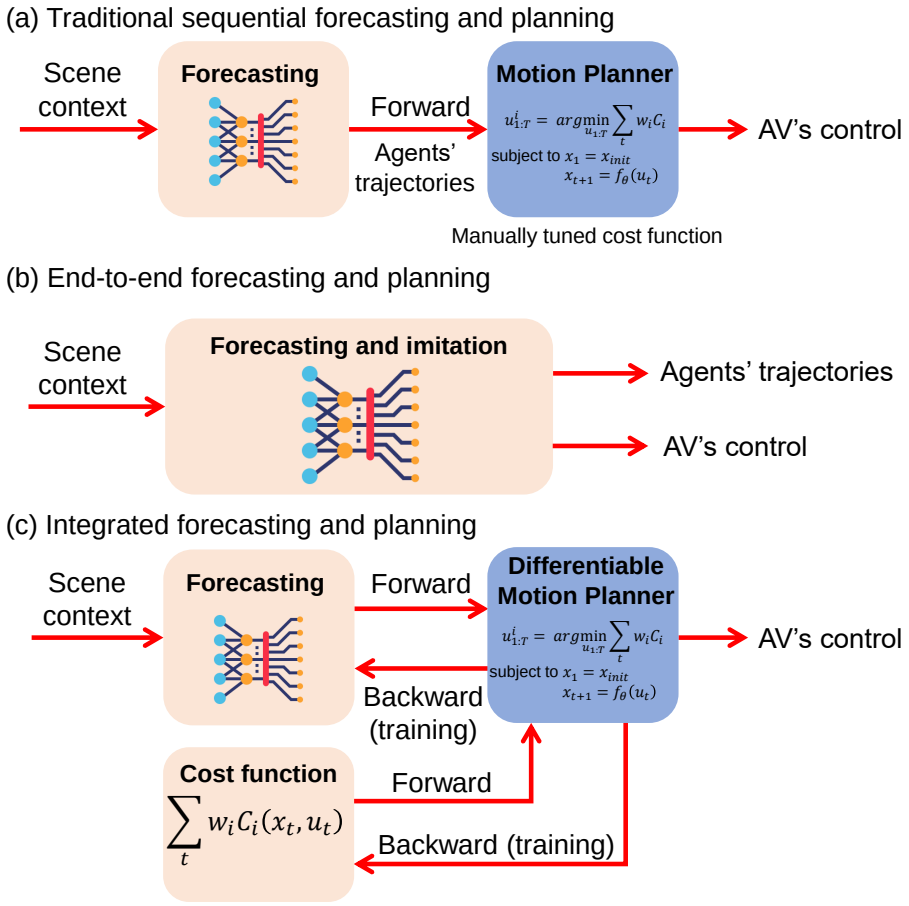


FIGURE 6.1: Three existing different motion planning paradigms

agents' future behaviors in the training process, where the planning error of the AV could also influence the prediction results of other agents, enabling the planner to make better and human-like plans at test time. Specifically, we construct a Transformer-based neural network to forecast the future trajectories of surrounding agents simultaneously. Then, the prediction results are channeled into a differentiable motion planner (optimizer), which explicitly plans a trajectory for the AV. When training the framework, the AV's planned trajectory is compared against the human driving one and the planning loss can be back-propagated to the prediction module and cost function, to alter the prediction results and automatically adjust the cost function with the objective of making human-like plans. Here, we assume the perception results (e.g., detecting and tracking of moving objects [153], localization, and mapping) are accurate and focus on the joint prediction and planning problem. The proposed DIPP framework is trained to match the human driving trajectories (interactions among humans) in the entire driving scene and tested in both open-loop and closed-looped manners. In open-loop evaluations, we

compared the planned trajectories of our framework directly against human trajectories without executing the plan, while in closed-loop evaluations, we unrolled the planned trajectory in simulation to test its performance in deployment scenarios. The results show that our framework is able to deliver more planning-centric prediction results, enabling the system better adapt to the scenarios in which they are deployed. In summary, the main contributions are listed as follows:

- We propose a fully differentiable structured learning framework that integrates prediction and planning modules for autonomous driving, enabling the prediction results better fit to the downstream planning task and the cost function learnable from real-world driving data.
- We train the proposed framework with a large-scale real-world driving dataset that covers a wide variety of complex urban driving scenarios and validate it in both open-loop and closed-loop manners.
- We demonstrate that the proposed framework outperforms the baseline methods in both open-loop and closed-loop tests and conduct an ablation study to investigate the importance of each component.

6.2 Literature Review

6.2.1 Multi-agent prediction

There has been a growing body of learning-based approaches for trajectory prediction to excellent effect because deep neural networks can handle complex environments with multiple interacting agents and various road structures. Some works [16, 84, 85, 154] utilizing vector representation of scene context and Transformer-based networks have advanced the forecasting accuracy even further. However, most of the existing models are formulated to predict each agent's future trajectory independently, which is computationally inefficient and might produce inconsistent results. Thus, some approaches focus on multi-agent joint prediction to generate future trajectories for all agents of interest in a consistent manner [91, 106, 155, 156]. Importantly, using the multi-agent prediction method can allow the model to capture the interactions between agents and facilitate the planning task. Therefore, in

our proposed framework, we employ the multi-agent prediction setting and provide each agent with a local vectorized map that shows the possible lanes and nearby crosswalks. We utilize Transformer modules to model the interactions between agents and their relations to different elements on the local map.

6.2.2 Motion planning

Motion planning is a long-researched area and there are a variety of approaches such as trajectory optimization, graph search, random sampling, and more recently learning-based methods. Learning-based methods employ neural networks as driving policies that generate actions or trajectories directly from sensor inputs or perception results, such as deep imitation learning [20, 157] and reinforcement learning [158–160]. Though simple and efficient, the neural network-based motion planner suffers from poor interpretability and generalization capability and also lacks stability and safety guarantees. Therefore, we turn to classic motion planning methods [138] and a popular choice is an optimization-based method as it is flexible and achieves maximal control of the trajectory. In particular, we employ a differentiable least-squares nonlinear optimizer as the motion planner, so that the optimizer can be integrated into the neural framework and its parameters (e.g., cost function weights) can be learned at the same time. In contrast to other planning methods that often work in static environments or assume a perfect prediction of surrounding obstacles, our proposed approach integrates the prediction of the surrounding agents and jointly trains the predictor and planner to achieve better performance of robustness.

6.2.3 Joint prediction and planning

Driving in dense and interactive traffic requires joint reasoning of other agents' future behaviors and the AV's plans [87, 97, 116] to make active and human-like decisions. One promising framework is to generate a set of candidate trajectories for the AV and predict other agents' future trajectories conditioned on the AV's planned trajectories [93], and the best planning trajectory could be selected after evaluating these candidate plans and corresponding prediction results [114, 116, 117]. For example, an interactive prediction and planning framework is proposed

in [97], which can predict other agents' future states according to a planned state sequence of the ego vehicle. The planner in their approach uses a cross-entropy method to sample a large number of action sequences and has to iteratively query the prediction model to get other agents' responses, which may significantly slow down the planning process. In contrast, our method enables the prediction model to output planning-aware prediction results and implicitly captures the influence of the ego vehicle's future actions during the training process. This allows the planner to generate human-like plans in response to the prediction results during test time without requiring iterative queries to the prediction model.

Another promising framework is the end-to-end holistic neural network model that implicitly captures the prediction-planning interactions in the latent space, such as SceneTransformer [87], which jointly outputs a motion plan for the AV and predicted trajectories for other agents. Although end-to-end models enjoy enhanced accuracy and simple inference, they cannot explicitly compute the feedback loop between planning and prediction from open-loop and offline training and thus cannot guarantee safety. Another challenge of such methods is the distribution shift encountered when the predicted actions are rolled out in deployment. Therefore, some other methods, e.g., MATS (Mixtures of Affine Time-varying Systems) [161], have tried to couple the prediction model with classic optimization methods because of their interpretability and reliability. MATS predicts the parameters of a linear-affine dynamical structure, which are utilized by a model predictive controller (MPC) for motion planning. Though efficient due to a small number of parameters, MATS does not incorporate the scene context in forecasting, and its performance is compromised. We propose to output the trajectories of surrounding agents using a neural network and combine it with differentiable optimization steps that explicitly consider ride comfort, safety, and traffic rules, making the feedback between planning and prediction differentiable and enhancing both safety and human driving similarity.

6.3 Methodology

6.3.1 Problem formulation

We formulate the driving scene with the AV and a varying number of diverse interacting traffic participants as a continuous-space discrete-time system (time discretization is uniform), where the AV is denoted as A_0 and other agents are denoted as $A_1, \ldots, A_N$. Each agent including the AV has a semantic class (i.e., vehicle, bicycle, or pedestrian), and its state at time t is denoted as $\mathbf{s}_t^i$, where i is the agent index. We also introduce the scene context, such as a vectorized high-definition map and traffic light signals, to the system and denote it as $\mathbf{M}$. Assuming the current time step is $t = 0$, given the historical states of all agents for the previous H timesteps $\mathbf{X} = \mathbf{s}_{-H:0}^{0:N}$ and the scene context $\mathbf{M}$, the predictor needs to predict K possible joint future trajectories of all agents for the next T timesteps $\{\mathbf{Y}_k | k = 1, \cdots, K\}$, $\mathbf{Y}_k = \hat{\mathbf{s}}_{1:T}^{0:N}$ where $\hat{\mathbf{s}}_t^i$ is the predicted state of agent i at future timestep t, and the probability of each future $\{p_k | k = 1, \cdots, K\}$. The planner needs to further optimize the AV's future trajectory according to the initial trajectory of the AV $\hat{\mathbf{s}}_{1:T}^0$, the prediction results of other agents, and the cost function.

Accordingly, the proposed framework consists of two main components, as illustrated in Fig. 6.2. The first component utilizes a neural network predictor to forecast the future states of surrounding agents and generate the initial plan for the motion planner. This predictor embeds the historical trajectories of agents and scene context into high-dimensional spaces, producing key representations ($\mathbf{K}$), value representations ($\mathbf{V}$), and query representations ($\mathbf{Q}$) for the attention mechanism [143] within the interaction modeling modules. By establishing interaction relationships, this module determines the different possible future trajectories for each agent and their associated probabilities. The second component employs a differentiable optimizer as a motion planner, explicitly determining a future trajectory for the autonomous vehicle based on the most probable prediction outcome and the initial motion plan. Specifically, we formulate the motion planning problem as a nonlinear least squares problem. The optimization variables ($\mathbf{u}$) represent the action sequence, while $\mathbf{u}^i \subset \mathbf{u}$ denotes a subset of the action sequence. The predicted states of other agents ($\hat{\mathbf{s}}$), individual costs (c_i), and cost weights (ω_i) are incorporated into the objective function, which is a sum of squared cost terms.

These cost terms encompass various driving factors, such as travel efficiency, collision avoidance, and ride comfort. We utilize a differentiable nonlinear optimizer to solve this optimization problem, enabling back-propagation of gradients from the planner (planning loss) to the prediction module and cost function during training. As a result, the entire framework is end-to-end learnable. The details of the neural network predictor and differentiable motion planner are given in Section 6.3.2 and Section 6.3.3, respectively.

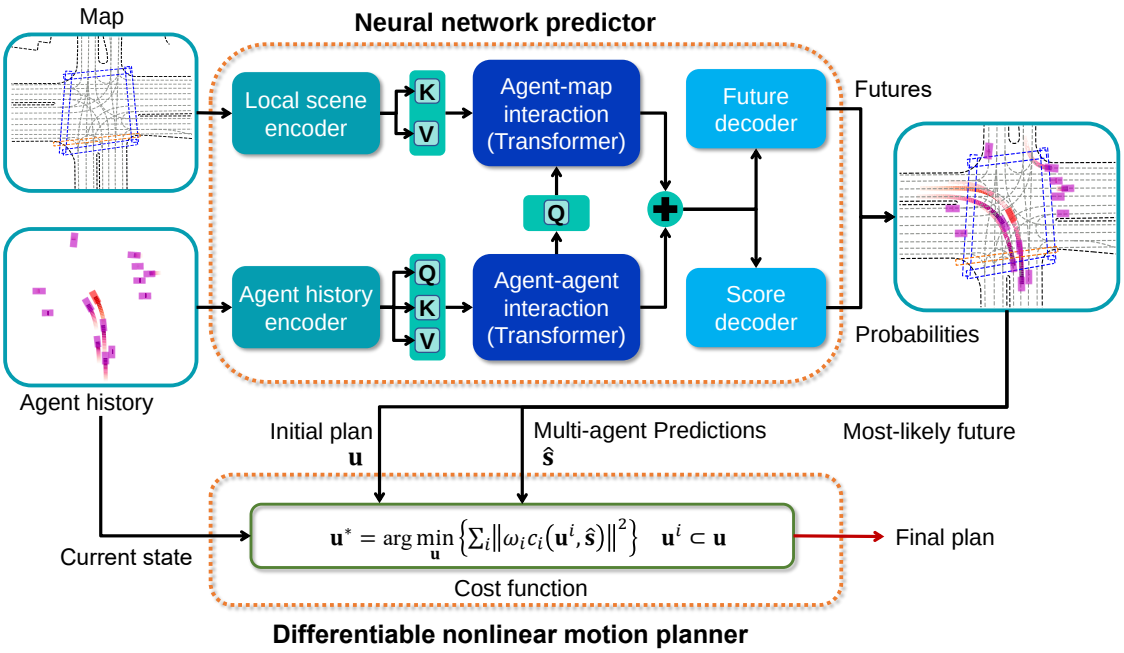


FIGURE 6.2: The proposed differentiable integrated prediction and planning (DIPP) framework

6.3.2 Multi-agent prediction and initial plan

Input representation. The neural network predictor receives two types of input data: historical agent states and scene context. For each agent, its historical state is a sequence of dynamic features for the past timesteps $H = 20$, including its 2D position, heading angle, velocity, and bounding box size. We consider the nearest $N = 10$ agents around the AV, whose observation data is stored in a fixed-size tensor with the missing agents padded as zeros. For the scene context, we consider two types of map elements, which are lanes and crosswalks. Each map element is presented by a vector, which consists of a sequence of waypoints with different associated features. For each agent, we build a local scene context, which

encompasses the possible lanes it might take and surrounding crosswalks. The features of the waypoint in the drivable lane include the positions and headings of the lane center and lane boundaries, as well as the speed limit and traffic signals (e.g., traffic light state and stop sign), and the features of the waypoint in the crosswalk are the positions of points enclosing the crosswalk area. All the agents' and map elements' positional attributes are translated to the AV's local coordinate system.

Encoding agent history and scene context. To encode an agent's observed historical states, we feed the state sequence into a long short-term memory (LSTM) network. LSTM is used because we find it would bring better final prediction performance and computational efficiency than Transformers when dealing with short-term time series, and we can retrieve the encoded state feature at the last timestep from the LSTM as the node feature in the agent interaction graphs. Each type of agent shares an LSTM encoder, and all agents including the AV are stacked to form a tensor of agents' encoded historical states. The local scene context encoder consists of a lane encoder for processing the lanes and a crosswalk encoder for processing the crosswalk vectors. The lane encoder uses a multi-layer perceptron (MLP) to encode numeric features (e.g., positions, directions, and speed limits) and embedding layers to encode discrete features (e.g., traffic light state, lane type, and stop sign), and the crosswalk encoder is another MLP to encode the numeric features (i.e., positions and headings). For each agent, we find its nearby 6 lanes and 4 crosswalks as map vectors, encode them separately, and concatenate the encoded map vectors to form a tensor of the agent's local scene context.

Modeling agent-agent and agent-scene interactions. Capturing relationships or dependencies between agents and the environment is essential in ensuring prediction accuracy. To realize this, we follow the idea of [106] and construct two interaction graphs, which are the agent-agent interaction graph (all agents are fully connected in the graph) and the agent-map interaction graph (a target agent is connected to all local lanes and crosswalks polylines. We use a two-layer self-attention Transformer encoder as the agent-agent interaction encoder to process the graph, where the query, key, and value $(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ are the encoded agents' historical trajectory features. Here, the functionality of the Transformer is similar to the graph attention network [106] but with improved structure and representation capability, and using a few layers of Transformer blocks would not significantly increase

the computational cost. With each agent's encoded local scene context in hand, we employ two cross-attention Transformer encoders as the agent-map interaction encoder: one is for modeling the agent's attention on each lane vector (focusing on waypoints in the vector), and the other is for agent's attention on each cross-walk vector. We utilize an agent's interaction feature as query ($\mathbf{Q}$), and a single map vector (a sequence of encoded waypoints) as the key and value ($\mathbf{K}, \mathbf{V}$). The operation is called multiple times to process all the map vectors from an agent, resulting in a sequence of agent-map vectors attention features. Then we introduce a multi-modal attention Transformer encoder [16], which is essentially an ensemble of cross-attention modules, to output multiple trajectories through the different attention modules, representing the different relations between the agent and map vectors. We still use an agent's interaction feature as query, and agent-map vectors attention as key and value, yielding different encodings of the agent's relationship with the local scene context. Likewise, we apply the multi-modal encoder to all agents to extract their possible interactions with the scene context and stack the results along the future axis.

Decoding multi-modal future. For each agent, its agent-agent interaction encodings are repeated and concatenated with the multi-modal agent-map interaction encoding to form a final feature vector. For the surrounding agents, we decode their possible future trajectories through a shared MLP from the final feature vector. For the AV, we decode its future control actions (i.e., acceleration and steering angle) from the feature vector through an MLP and translate the action sequence to trajectory using a kinematic model (see Section 6.3.3). We also output multiple possible trajectories for the AV to better model the interaction between agents. To predict the probability of each future (joint trajectories for all agents), we use max-pooling to aggregate the information from all agents and map vectors and an MLP to decode the probability. The motion planner will take as input the future (i.e., the initial plan and other agent's prediction) with the highest probability.

6.3.3 Differentiable motion planning

Problem statement

We consider an open-loop finite-horizon optimal planning problem, which is to find a sequence of control inputs $\mathbf{u} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_T\}$ to minimize the cost function

in Equation (6.1). The cost function is a sum of squared residual terms, each represented by a product of a weight ω_i and cost c_i [162], and the state of other surrounding agents $\hat{\mathbf{s}}$ is provided by the neural network predictor.

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \frac{1}{2} \sum_i \parallel \omega_i c_i(\mathbf{u}^i, \hat{\mathbf{s}}) \parallel^2, \tag{6.1}$$

where c_i is a function of subsets of the control sequence $\mathbf{u}^i \subset \mathbf{u}$ and potentially the predicted states of other surrounding agents $\hat{\mathbf{s}}$. In the training process, the gradient of the final planning error can be backpropagated to the predicted states of other agents and cost weights through the optimization problem [163, 164], to implicitly influence other agents' future behaviors and adjust the cost weights.

Kinematic model

When calculating some costs, we need to convert the control action $\mathbf{u}_t = \{a_t, \delta_t\}$ (where a_t is acceleration and δ_t is steering angle) to state $\mathbf{s}_t = \{x_t, y_t, \theta_t, v_t\}$ (where (x_t, y_t) is 2D position, θ_t is heading angle, and v_t is velocity). We adopt the kinematic bicycle model [165] shown in Equation (6.2) to update the states of the AV.

$$\begin{aligned} v_{t+1} &= v_t + a_t \Delta t, \\ x_{t+1} &= x_t + v_t \cos(\theta_t) \Delta t, \\ y_{t+1} &= y_t + v_t \sin(\theta_t) \Delta t, \\ \theta_{t+1} &= \theta_t + \frac{v_t}{L} \tan(\delta_t) \Delta t, \end{aligned} \tag{6.2}$$

where L is the wheelbase of the vehicle and Δt is the time interval. The kinematic bicycle model is differentiable, thus permitting calculating gradients and Jacobians in the optimizer.

Cost function

The cost function encompasses a variety of carefully crafted cost terms that encode different aspects of driving including travel efficiency, ride comfort, traffic rules, and most importantly safety. The details of the different cost terms are given below.

Travel efficiency. We encourage the AV to reach the destination as fast as possible but not run above the speed limit. Therefore, the cost of travel efficiency

is defined as the difference between the AV's speed v_t and speed limit:

$$\mathbf{c}_t^{speed} = v_t - v_{limit}, \tag{6.3}$$

where v_{limit} is the speed limit of the lane.

Ride comfort. Human drivers prefer comfortable maneuvers, and we introduce four sub-costs to represent the ride comfort factors the AV needs to optimize. They are longitudinal acceleration a_t and longitudinal jerk $\dot{a}_t$, as well as steering angle δ_t and steering change rate $\dot{\delta}_t$ for lateral stability and comfort.

$$\begin{aligned} \mathbf{c}_t^{acc} &= a_t, \\ \mathbf{c}_t^{jerk} &= \dot{a}_t, \\ \mathbf{c}_t^{steer} &= \delta_t, \\ \mathbf{c}_t^{rate} &= \dot{\delta}_t, \end{aligned} \tag{6.4}$$

where $\dot{a}_t = \frac{\Delta a_t}{\Delta t}$ and $\dot{\delta}_t = \frac{\Delta \delta_t}{\Delta t}$ are the discrete forms of the longitudinal jerk and steering rate respectively.

Traffic rules. The AV is expected to adhere to the traffic rules and structure of the road. Thus, to promote staying close to the lane (route) centerline and following the lane direction, two sub-costs are designed.

$$\begin{aligned} \mathbf{c}_t^{pos} &= p_t - p_{l,\perp}, \\ \mathbf{c}_t^{head} &= \theta_t - \theta_{l,\perp}, \end{aligned} \tag{6.5}$$

where p_t and $p_{l,\perp}$ are the positions of the AV and closest point from the lane's centerline to the AV respectively, and θ_t and $\theta_{l,\perp}$ are the heading angles of the AV and its closest point on the lane respectively.

In addition, obeying traffic lights should be treated as a hard constraint for the AV. Here, we replace the hard constraint with a soft penalty term, which can be assigned with a large cost weight. We assume the AV runs along a predefined route and its running distance is s_t, which is derived from $s_t = \sum_{t'=1}^{t} v_{t'} \Delta t$, and the stop line position (if encountering a red light) on the route is s_{stop}. We can formulate

the cost of violating traffic lights using the hinge loss:

$$\mathbf{c}_t^{traffic} = \begin{cases} s_t - s_{stop}, & s_t \geq s_{stop} \\ 0, & \text{otherwise} \end{cases}. \tag{6.6}$$

It means that if the AV goes past the stop line at a red light, a large penalty will be induced, thus forcing the AV to stop near the stop point.

Safety. Keeping a safe distance from other traffic participants on the road to avoid collision is a fundamental requirement for AVs. However, optimizing the safe distances to all other agents in the scenario is unnecessary and time-consuming. Therefore, we take advantage of the Frenet frame [166, 167], which decouples the vehicle trajectory into the longitudinal direction along a predefined driving route and the lateral direction perpendicular to the route, to define the interactive agents. As illustrated in Fig. 6.3, all other agents' positions are projected to the Frenet frame with the AV's route as the reference path. At each future timestep, those agents whose predicted positions are within the route's conflict area are defined as the interactive agents. The calculation of safe distance at a specific timestep is given as:

$$d_{safe} = \min_i \parallel p_t - p_t^i \parallel_2, \tag{6.7}$$

where p_t^i is the predicted position of the interactive agent i at future timestep t.

We also employ the hinge loss to formulate the safety cost to ensure the safe distance is large enough.

$$\mathbf{c}_t^{safety} = \begin{cases} \epsilon - d_{safe}, & d_{safe} \leq \epsilon \\ 0, & \text{otherwise} \end{cases}, \tag{6.8}$$

where ϵ is the minimum safety distance requirement, which is defined as the sum of the lengths of two agents and a safety gap.

Weights. The cost terms in Equation (6.1) consist of different features listed before at different timesteps. Here, we only consider the weights for different features and the weights for individual features are the same at different timesteps. For features other than safety and traffic rules, we consider all the timesteps over the planning horizon. For the cost of traffic rule violation, we consider every other

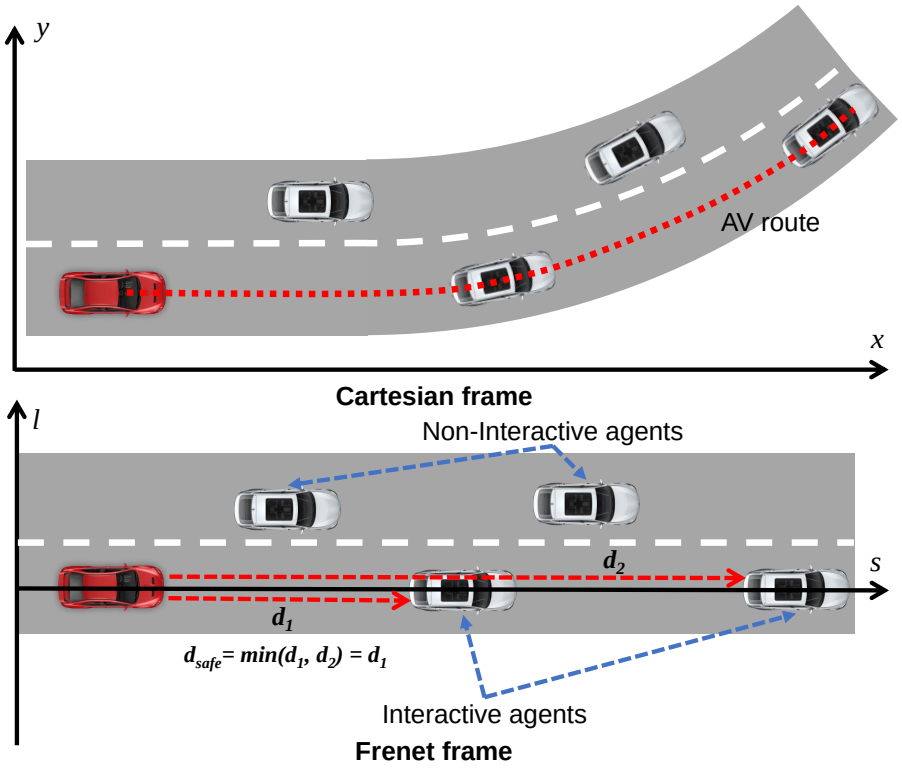


FIGURE 6.3: Illustration of the calculation of safe distance

timestep across the future horizon to reduce computation costs. Because the computation for the safety cost is expensive, we only consider the trajectory waypoints at $[0.1, 0.3, 0.6, 1.0, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0]$ seconds, which could significantly speed up the optimization without compromising safety. To ensure the constraints are satisfied, the weights for the constraints (i.e., safety and traffic light violations) in the cost function, are set to a large enough value and fixed during training.

Differentiable optimization

The Gauss-Newton algorithm is utilized to solve the nonlinear optimization problem [164]. It is an iterative least-squares optimization approach, where at each iteration step, the nonlinear objective is first linearized around the current control variables to derive the linear system:

$$\left(\sum_i J_i^\top J_i\right) \Delta \mathbf{u} = \sum_i J_i^\top \omega_i c_i, \tag{6.9}$$

where $J_i = \frac{\partial(\omega_i c_i)}{\partial \mathbf{u}^i}$ is the Jacobian of the cost with respect to the control variable.

We can explicitly solve the linear system using Cholesky decomposition of the normal equations to find the update $\Delta\mathbf{u}$, and eventually update the control variables:

$$\mathbf{u} \leftarrow \mathbf{u} - \alpha\Delta\mathbf{u}. \tag{6.10}$$

where $0 < \alpha \leq 1$ is the step size. In the original Gauss-Newton method, $\alpha = 1$, but it may cause the nonlinear objective not to decrease at every iteration. Therefore, an improved version is setting $0 < \alpha < 1$ to employ a fraction of the update and thus mitigates the divergence issue.

Since the above procedure is fully differentiable, we can set up other differentiable components (e.g., prediction and cost function weights) and integrate them into the planner, enabling the whole architecture to be differentiable. Moreover, to start the optimization, one has to provide an initial guess for the control variables, which is crucial to the convergence to the global optimum, and we use the control actions given by the prediction network as the initial guess, which is also differentiable.

6.3.4 Learning process

At a high level, we regard the motion planner as the primary component of our framework. During the forward pass, the motion planner takes as input the prediction results of other agents and the initial plan (with the highest probability), as well as the cost function, to start the trajectory optimization and iteratively updates the planned trajectory until passing a convergence check or reaching the maximum number of iterations. At the end of the optimization, the motion planner will output a trajectory for the AV to follow. In the backward pass, we evaluate the loss function on the planned trajectory, compute gradients with respect to the initial plan, cost function weights, as well as the predicted trajectories of other agents, and back-propagate the loss through the neural components and update the parameters, in order to generate better quality trajectories. The learning process of our framework is stable and efficient because all the components (i.e., prediction, initial plan, and cost function) are learned in a supervised manner. Compared to generative adversarial imitation learning (GAIL) [51, 168], which learns proxy reward signals through adversarial learning, our method only learns the weights of the cost function terms. However, the initialization of the cost function weights

is important in our method because this can affect the optimization process. The learning process of our framework is summarized in Algorithm 2.

Since all operations in the framework are differentiable, we can train the entire framework in an end-to-end fashion from real-world driving data, and we implement the framework using PyTorch and Theseus [162]. The loss function for training the framework encompasses four terms: prediction loss for all agents, score loss to accurately predict the probabilities of different futures, as well as imitation loss and planning cost for the AV. The overall loss is defined as:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{prediction} + \lambda_2 \mathcal{L}_{score} + \lambda_3 \mathcal{L}_{imitation} + \lambda_4 \mathcal{L}_{cost}, \tag{6.11}$$

where λ_1, λ_2, λ_3, and λ_4 are the weights to scale these different loss terms. For prediction loss, we treat each future to be coherent individual futures across all agents and back-propagate the smooth L1 losses on the trajectories through the joint future that most closely matches the ground truth:

$$\mathcal{L}_{prediction} = \sum_{k=1}^{K} 1(k = \hat{k}) \sum_{i=1}^{N} \text{smooth}L_1(\xi_i^k - \xi_i^{gt}), \tag{6.12}$$

where 1 is the indicator function, $\hat{k}$ is the index of the best-predicted joint future of multiple agents (the future with the smallest sum of displacement errors across all agents, $\hat{k} = \arg\min_k \sum_i \parallel \xi_i^k - \xi_i^{gt} \parallel_2$), and ξ_i^{gt} is the individual ground-truth trajectory. The scoring loss is defined as:

$$\mathcal{L}_{score} = \sum_{k=1}^{K} 1(k = \hat{k}) \log p_k, \tag{6.13}$$

where p_k is the probability of future k.

The motion planner would take as input the most-likely trajectories of all agents, where the AV's trajectory (original control actions) is used as planning initialization and other agents' future trajectories are used in computing the safety cost. The output of the motion planner ξ_{AV} is compared against the ground-truth trajectory ξ_{AV}^{gt} to calculate the imitation loss.

$$\mathcal{L}_{imitation} = \text{smooth}L_1(\xi_{AV} - \xi_{AV}^{gt}). \tag{6.14}$$

Algorithm 2: Learning of the proposed DIPP framework

Input : Neural network predictor f_θ, initial cost function weights $\{\omega_i\}$, differentiable planner g

Get the environment information (Agent tracks $\mathbf{X}$ and vectorized map $\mathbf{M}$);
Predict the AV and other agents' multi-modal joint trajectories $\{\xi_i^k\}_{k=1:K,i=0:N},\ \{p_k\}_{k=1:K} \leftarrow f_\theta\left(\mathbf{X}, \mathbf{M}\right)$;
Calculate prediction loss $\mathcal{L}_{prediction}$ and score loss $\mathcal{L}_{score}$;
Get the most-likely prediction result $\{\xi_i^{\hat{k}}\}_{i=0:N},\ \hat{k} = \arg\min_k \sum_i \parallel \xi_i^k - \xi_i^{gt} \parallel_2$;
Plan the AV's trajectory using the differentiable planner $\xi_{AV}, \mathcal{L}_{cost} \leftarrow g\left(\xi_0^{\hat{k}}, \{\xi_i^{\hat{k}}\}_{i=1:N}, \{\omega_i\}\right)$;
Calculate imitation loss $\mathcal{L}_{imitation}$;
Calculate total loss $\mathcal{L}$ according to Equation (6.11);
Backpropagate loss and calculate gradients with respect to θ and $\{\omega_i\}$;
Update f_θ and $\{\omega_i\}$ using Adam optimizer;

6.4 Experiments

6.4.1 Dataset

We train and validate our framework on the Waymo Open Motion Dataset [144], a large-scale real-world driving dataset that focuses on urban driving scenarios with complex and diverse road structures and traffic interactions. The dataset contains 103,354 unique driving scenes, each 20 seconds in duration, with detailed map data and agent tracks. We randomly select 10% of the dataset, which corresponds to 100 data files out of 1000 files in total, resulting in a total of 10,156 different driving scenes. Since the driving scenes are randomly stored in the data files and the selected driving scenes are a random subset of the dataset, the data distribution of the reduced dataset remains the same as the original. To segment a 20-second driving scene into frames (each frame contains 7-second object tracks at 10Hz), we set up a 7 s time window with a 2 s history observation horizon and a 5 s planning/prediction horizon. The window slides from the beginning of the scene with a stride of 10 timesteps (1 second), resulting in 14 segments of 7 s frames. For training, we use 80% of the selected scenes, while the remaining 20% are used for validation and open-loop testing. The total number of frames used for training is 113,622, while 28,396 frames are used for validation and open-loop testing.

6.4.2 Evaluation

We evaluate the performance of the framework in both open-loop and closed-loop manners. In the open-loop testing, the motion planner plans a trajectory for the AV based on the current states and prediction results at each frame, and we compare the planned trajectory against the AV's ground-truth trajectory. In the closed-loop testing, we construct a log-replay simulator, where at each step, the AV will take the first action from its planned trajectory and update its state, while the neighboring agents will follow their record trajectories in the dataset. We set up the following metrics to evaluate the performance of the framework.

Safety: the collision rate is employed to measure the safety performance of the system. For open-loop testing, collision is calculated based on the AV's planned trajectory and the ground-truth future trajectories of other agents. If a collision is detected at any step in the plan, then the frame is deemed as a collision. For closed-loop testing, we check if the AV collides with other agents at every time step during the simulation.

Traffic rule violation: we consider deviating from the route and passing over the stop line at a right light as traffic rule violations, and we calculate the total number of frames where the AV violates the traffic rules.

Vehicle dynamics: three metrics are introduced to quantify rider comfort and feasibility of the planned trajectory, which are longitudinal acceleration and jerk, as well as lateral acceleration. They are absolute values averaged over time in a scene and will be compared against human driving metrics.

Human driving similarity: we use the position errors between the planned trajectory in open-loop testing or rollout trajectory in closed-loop testing and the ground truth to quantify the human likeness. We calculate the position errors at different timesteps (i.e., 1 s, 3 s, 5 s in open-loop testing and 3 s, 5 s, 10 s in closed-loop testing).

Prediction: we calculate the average displacement error (ADE) and final displacement error (FDE) of the multi-agent joint trajectories from the most-likely future to reflect the performance of the prediction module. ADE computes the L2 norm between the ground-truth state $s_{a,t}$ and the most-likely joint prediction $\hat{s}^{\hat{k}}_{a,t}$: $\frac{1}{AT}\sum_a \sum_t \parallel s_{a,t} - \hat{s}^{\hat{k}}_{a,t} \parallel_2$, where $\hat{k} = \arg\max p_k$ is the index of the most-likely

predicted joint future. FDE calculates the L2 distance between the ground truth and predicted states at the final step T: $\frac{1}{A}\sum_a \parallel \hat{s}_{a,T} - \hat{s}^{\hat{k}}_{a,T} \parallel_2$.

6.4.3 Comparison baselines

We compare our proposed method with the following baselines to reveal the effects and advantages of the proposed framework.

Vanilla imitation learning: based on the same scene context, we use only imitation learning (IL) to train a network to directly generate the trajectory for the AV. The structure of the network is the same as the proposed method, but the prediction of other agents is removed and only one trajectory instead of multi-modal trajectories is generated. The IL policy is trained on the same dataset with the same hyperparameters as the proposed method, which makes sure the comparison results are fair.

Imitation learning with prediction sub-task: we train a multi-task neural network with the main task of imitation learning and the sub-task of predicting other agents' future trajectories. This is equivalent to the proposed network without the differentiable motion planner, and the training settings and hyperparameters are the same as the proposed method.

Separated prediction-planning: based on the same network, we separately train a prediction module without the integrated motion planner to output the initial trajectory for the AV and predicted trajectories for other agents, which is the same as the IL with the prediction model. In testing, a motion planner, which is the same as the proposed method, is utilized to perform trajectory planning for the AV with the trained prediction model and a pre-specified cost function.

Conservative Q-learning (CQL) [112]: we utilize a widely-used offline reinforcement learning algorithm that can learn to make decisions from offline datasets. We implement the CQL method with the d3rlpy offline RL library [169] using the default hyperparameters. The observation space is a 3-channel RGB bird-eye-view image $256 \times 256 \times 3$ of the driving scene [15], the action is the target pose of the next step $(\Delta x, \Delta y, \Delta\theta)$ relative to the ego vehicle's current position, and the reward function is the distance traveled per step plus an extra reward for reaching the goal. The policy is trained on the same scenarios as the proposed method and

all transition steps (observations, actions, rewards, and terminals) in each scenario will be used. The trained policy is only evaluated in the closed-loop planning test.

Intelligent driver model (IDM) [170]: we utilize a well-known model-based driving model, IDM, to control the longitudinal movement of the ego vehicle along the reference route. The IDM planner takes as input the current state of the ego vehicle (position and speed) and the state of the leading agent (position, speed, and length) under the coordinate of the reference route, and outputs the acceleration of the ego, which is then translated to the coordinate in the scene. The IDM method is only used in the closed-loop planning test.

6.4.4 Implementation details

Network. The network outputs 3 possible futures (joint trajectories of all agents) and their probabilities. For other agents, we predict the displacements relative to their current locations instead of their coordinates, which shows significant improvement in prediction accuracy. The learning of the cost function is also incorporated into the network and implemented by a small MLP, which takes a fixed dummy input and outputs the weights of the cost function. We set the cost weights for traffic light violations and collisions as large values (larger than two orders of magnitude) and unlearnable to ensure the constraints are satisfied.

Training. We use a batch size of 32 and an Adam optimizer with a learning rate that starts from 2e-4 and decays by a factor of 0.5 every 4 epochs. The total number of training epochs is 20 and we pre-train the framework for 5 epochs without the planner to get good initial prediction results and control actions. For speed considerations, we set the max number of iterations for the motion planner to 2, which will also encourage the network to produce high-quality initial plans, and the step size for the Gauss-Newton update is $\alpha = 0.4$. The weights for the loss function are set to $\lambda_1 = 0.5$, $\lambda_2 = 1$, $\lambda_3 = 1$, and $\lambda_4 = 0.001$. The results shown in Section 6.5.1 and Section 6.5.2 are based on this loss function setting and we investigate the effects of the loss function weights in Section 6.5.3. We clip the gradient norm of the network parameters with the max norm of the gradients as 5.

Testing. In the testing process, the max number of iterations for the motion planner is set to 50, the step size is set to 0.2, and the absolute error tolerance is

set as 1e-2, in order to plan a high-quality trajectory for the AV. Acceleration and jerk in the longitudinal and lateral directions are calculated based on the positions and headings on the trajectory. To check if a collision happens, we use a list of circles to approximate an object at each timestep. If the distance between any two circles' origins of the given two objects is lower than a threshold, it is considered as they collide with each other.

The main hyperparameters used are summarized in Table 6.1.

TABLE 6.1: Hyperparameters of the model and training process

Parameter	Value
Number of neighbors N	10
Number of predicted futures K	3
Historical timesteps H	20
Future timesteps T	50
Step size α	0.4
Planning iterations	2
Initial learning rate	2e-4
Total training epochs	20
Pretraining epochs	5
Batch size	32
Weight for prediction loss λ_1	0.5
Weight for score loss λ_2	1
Weight for imitation loss λ_3	1
Weight for planning cost λ_4	1e-3

6.5 Results and Discussions

6.5.1 Open-loop planning test

Multi-modal joint prediction. Fig. 6.4 shows two representative scenarios where the neural network predictor makes multi-future predictions (ten nearest agents to the AV), as well as the probabilities of the predicted futures. In Scenario 1, the AV consistently chooses to stop behind the leading vehicle in all futures, while different interacting behaviors among the two agents in the intersection area are predicted. The future (joint trajectories) closest to ground truth is assigned with a higher probability. In Scenario 2, the predictor delivers different future predictions

for both the AV and the other interacting agent at an unsignalized intersection. For example, the AV would choose to go if the other agent is predicted to turn left and stop if the other agent's trajectory is predicted to conflict with the AV's route. The close-to-ground truth futures are also assigned with higher probabilities. These results indicate that the neural network predictor is able to provide accurate multi-modal joint predictions for all agents in the scene and a good initial guess for the motion planner.

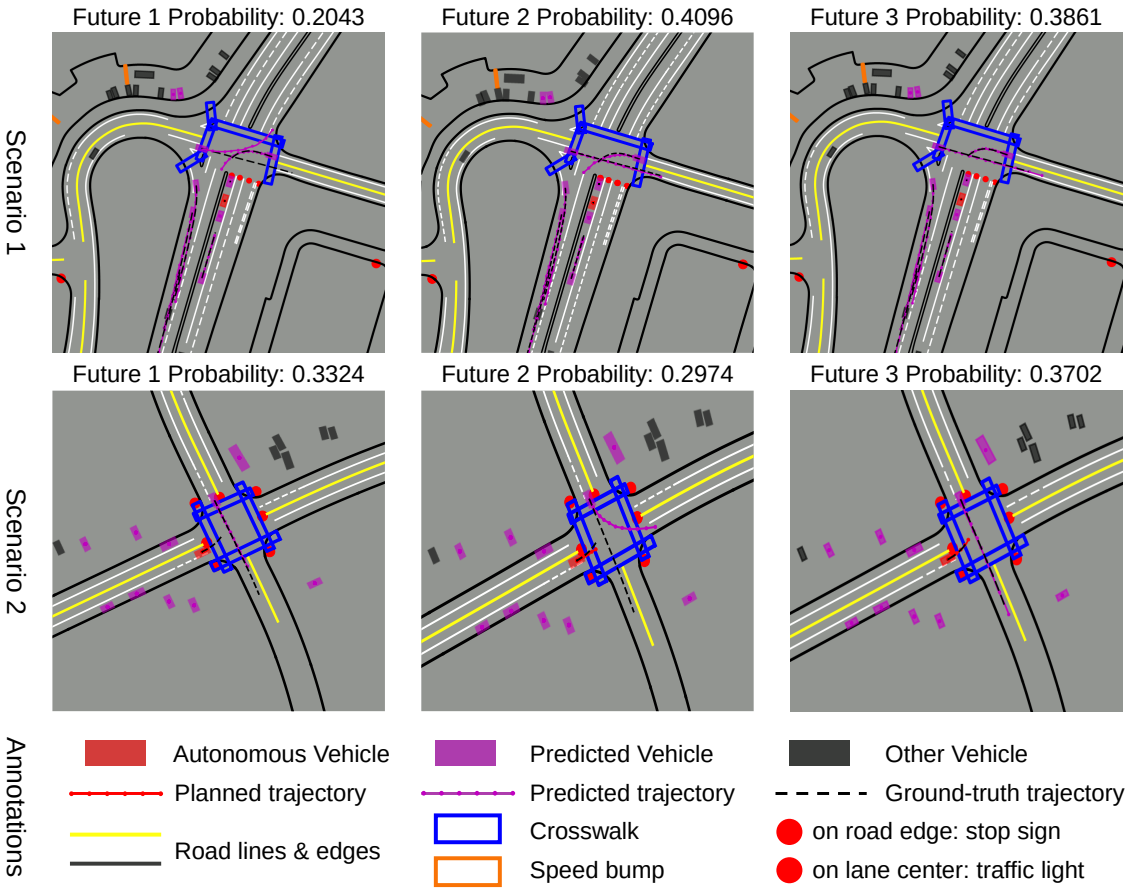


FIGURE 6.4: The multi-modal predictions given by the neural network predictor

Qualitative results. Fig. 6.5 displays some representative interactive scenarios, demonstrating the proposed framework's ability to plan a safe, traffic rule-compliant trajectory for the AV, based on the prediction results of other surrounding agents' future states (including vehicles, cyclists, and pedestrians). The colored solid lines are the planned or predicted trajectories for AV or surrounding agents, and black dotted lines are the ground truth trajectories. The results reveal that the motion planner is capable of handling a variety of urban driving scenarios involving a mixture of interacting traffic participants, including making a smooth right turn,

yielding to pedestrians on the crosswalk, stopping at the red light, and yielding to another vehicle at an unprotected left turn, etc.

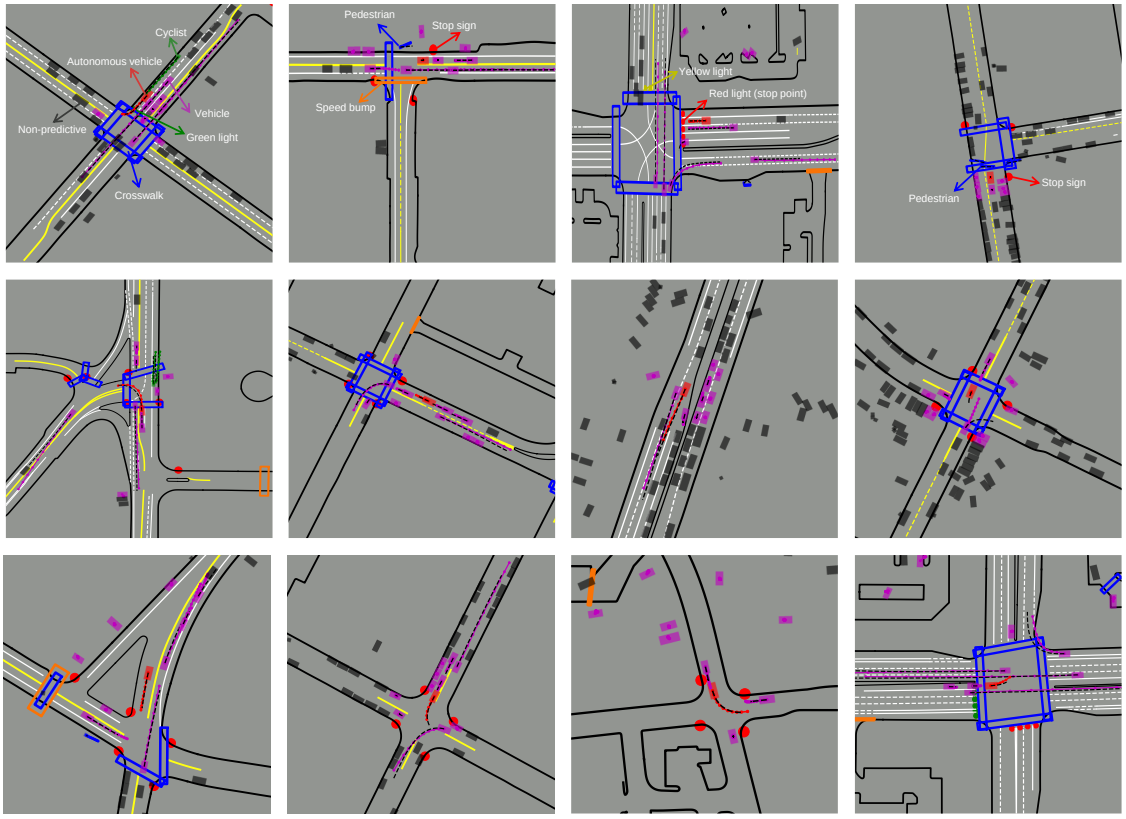


FIGURE 6.5: Qualitative results of the proposed framework in open-loop testing

In Fig. 6.6, we compare the proposed method with other baseline methods in two interactive scenarios (top: interacting with a vehicle; bottom: interacting with a cyclist). We can see that without explicit constraints, the neural network-planned trajectory gradually deviates from the lane centerline, while the motion planner's output trajectory adheres to the road structure. Our proposed method can generate planning-aware or planning-centric prediction results, which can help the planner deliver more close-to-human trajectories compared to using separate prediction results. For example, in the merging scenario, our method predicts that the interacting vehicle should yield to the AV at a stop sign, leading to more accurate prediction and planning results. In the interaction with a cyclist, our method predicts that the cyclist may move in front of the AV (however not really happen in the data) and the motion planner generates a slow-down trajectory to avoid potential risks, which is in accordance with the human-like trajectory. The result indicates that even if the prediction result is not entirely accurate, the planning module would benefit from such planning-centric results and produce

human-like plans. The prediction of other agents in our framework is similar to the driver's belief rather than an exact reflection of the ground-truth dynamics.

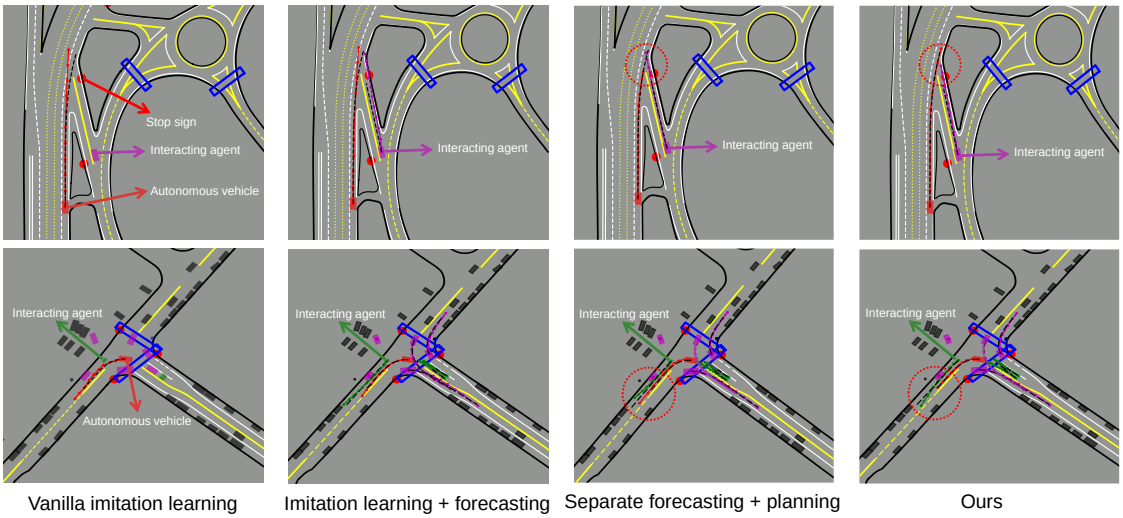


FIGURE 6.6: Qualitative comparison between the proposed method and baseline methods in open-loop testing

Quantitative results. Following the previously defined evaluation metrics, we list the quantitative results of our method and other baseline methods in Table 6.2. The results of open-loop testing indicate that the proposed method significantly outperforms the imitation learning-based methods in terms of collision and traffic rule violation rates. The imitation learning-based methods have a much higher off-route rate due to the lack of explicit constraints on following the route, and without considering the ride comfort, they yield an unacceptable lateral acceleration, which is significantly worse than the planning-based methods and human driving. On the other hand, the planning-based methods deliver much lower acceleration and jerk, which ensures smoothness and ride comfort. Because the neural network is only trained to predict the trajectories by direct regression, imitation learning with the multi-agent prediction sub-task method achieves the smallest planning error (compared with the ground-truth trajectories). However, since imitation learning methods ignore other important factors, they perform very poorly in the closed-loop test and barely finish a task (see Section 6.5.2). One major conclusion from the results is that our proposed DIPP method, which jointly trains the predictor and planner, outperforms the planner with a separate trained prediction module, in terms of both planning and prediction errors. The improvements in DIPP may come from two possible ways: 1) the additional regularization from the planning module enables the prediction module to make fewer prediction errors that would

negatively affect planning (e.g., large deviations from lane center and unnecessary stops), and 2) without significantly impacting raw prediction accuracy, the influence of the planning task enables the prediction module to make planning-centric results that would lead to better and human-like plans (e.g., react to possible cut-in of other agents). Another interesting finding is that though the vanilla imitation learning method performs the worst, adding the prediction sub-task can significantly reduce the collision rate and planning error, which shows the importance of multi-agent joint prediction. Fig. 6.7 shows the boxplots of the prediction error of the proposed method compared against the separated prediction-planning method and the planning error of all methods in open-loop tests. The results further clarify the conclusion that differentiable and integrated training of the prediction and planning modules (DIPP) can improve both the prediction and planning metrics in open-loop tests.

TABLE 6.2: Open-loop evaluation of the proposed method against baselines

Method	Collision (%)	Red light (%)	Off route (%)	Acc. (m/s^2)	Jerk (m/s^3)	Lat. Acc. (m/s^2)	Planning error (m) @1s	@3s	@5s	Prediction error (m) ADE	FDE
IL	5.469	2.772	4.816	0.546	2.881	3.151	0.175	1.416	4.194	–	–
IL+Prediction	3.930	1.670	4.542	0.672	4.004	3.939	**0.127**	**0.892**	**2.901**	0.773	1.916
Sep. Plan+Pred.	1.813	1.327	0.527	0.269	0.149	0.077	0.238	1.251	3.466	0.766	1.908
DIPP (ours)	**1.802**	**1.235**	**0.506**	0.269	0.150	0.079	0.227	1.187	3.335	**0.740**	**1.814**
Human	–	–	–	0.620	1.707	0.101	–	–	–	–	–

Sep. Plan+Pred.: separated plan + prediction, Acc.: acceleration, Lat. Acc.: Lateral acceleration

6.5.2 Closed-loop planning test

Qualitative results. To fully reveal the planning performance of our proposed method, we conduct a closed-loop test with 100 replayed scenes from the testing set. Specifically, we build a log-replay simulator to roll out the AV's planned trajectory, where other agents follow their original tracks in the dataset and only the AV's state gets updated. At each timestep, the AV plans a trajectory according to the prediction result and takes the first action from its planned trajectory, and replans at the next time step. The simulation horizon is 15 seconds and the interval is 0.1 seconds. To evaluate the closed-loop testing performance, we add a progress metric, which measures the running distance of the AV until it reaches the end of the scene, collides with other agents, or goes off route. We measure the position error between the AV's rollout trajectory and its ground-truth one at several time steps to reflect the similarity to the human driver. The results of the closed-loop test are given

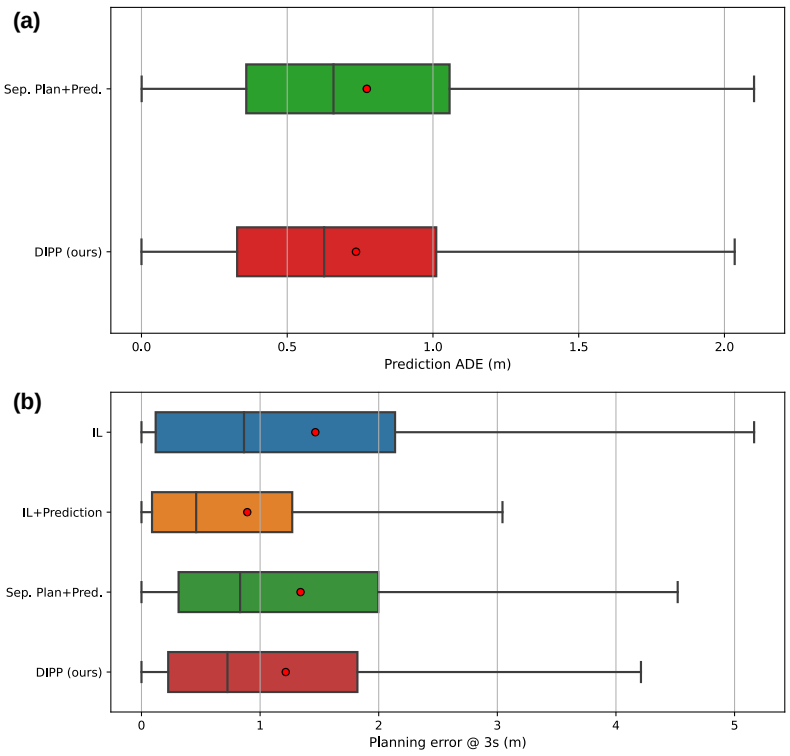


FIGURE 6.7: Boxplots of the prediction and planning errors of the proposed method and other baselines in the open-loop planning test

in Table 6.3 and we provide supplementary videos[1] of our method navigating in a variety of urban scenarios for better visualization and evaluation of our method. It is important to note that the collision rate is just a lower bound of safety since other agents do not react to the AV (e.g., the ego vehicle runs slower than the human driver in the data). The supplementary videos encompass various driving scenes where the proposed framework can handle different tasks such as cruising, obeying traffic lights, making smooth turns, and more importantly interacting with other road users. We also add some scenarios where other vehicles would take adversarial actions toward the AV (not intentionally as the AV has deviated from its original track), but our planning framework can handle such emergencies and avoid collisions.

Qualitative results. The results in Table 6.3 indicate that the imitation learning-based methods cannot finish a single scene without causing a collision or going off route. Although the planning error of the IL+prediction method is the smallest in open-loop testing, the position errors of IL-based methods are significantly worse. The ride comfort is also compromised as IL-based methods deliver much higher jerk

[1]https://mczhi.github.io/DIPP/

TABLE 6.3: Closed-loop evaluation of the proposed method against baselines

Method	Collision (%)	Off route (%)	Progress (m)	Acc. (m/s^2)	Jerk (m/s^3)	Lat. Acc. (m/s^2)	Position error (m) @3s	@5s	@10s
IL	40	60	8.23	1.857	3.249	1.872	11.29	19.42	43.63
IL+Prediction	42	58	11.05	1.345	1.977	0.813	11.03	19.13	42.25
CQL	50	50	16.38	3.456	8.798	2.154	12.85	20.18	45.10
IDM	28	0	49.44	0.885	2.236	0.081	3.895	9.568	30.32
Sep. Plan+Pred.	7	0	76.28	0.638	1.313	0.058	1.869	4.182	10.17
DIPP (Ours)	**5**	**0**	**77.57**	0.624	1.209	0.069	**1.726**	**3.913**	**9.365**
Human	–	–	–	0.621	2.067	0.105	–	–	–

Sep. Plan+Pred: separated plan + prediction, Acc.: acceleration, Lat. Acc.: Lateral acceleration

and acceleration. This is because the IL-based methods suffer from distributional shifts, which means the compounding errors lead the AV to a situation that deviates from the training distribution and degrade its performance. Similar to IL-based methods, the CQL method performs poorly in the test and cannot finish a single task because of the difficulty of training and labeling the correct reward for human driving behaviors. On the other hand, the planning-based methods leveraging structural information and domain knowledge perform significantly better despite learning purely from an offline dataset. The motion planner with a separate trained prediction module shows similar yet worse performance compared to our proposed method, which highlights the advantages of integrated training of the planning and prediction modules. Aside from its simplicity and few parameters, the IDM method is more effective in closed-loop testing since we only regulate the ego vehicle's longitudinal motion on the reference route. However, the performance of the IDM method is inferior to the planning-based approach, which is more flexible and can handle highly interactive and complex scenarios.

6.5.3 Ablation study

Effects of learnable components. To unveil the importance and function of each key component in our proposed framework, an ablation study is conducted. We set up three baselines in which the learnable components (i.e., initial plan, cost function, and prediction) are dropped out one at a time. We also set up an oracle method that uses the ground truth future trajectories of other agents as the prediction results to examine the capabilities of the proposed motion planner. For the non-learnable cost function baseline, the cost function used by the motion planner is manually tuned to balance the different cost terms. For the non-learnable

initialization baseline, the initial guess to the motion planner is fixed as ($a_t = 0, \delta_t = 0$) at all timesteps. For the non-learnable prediction baseline, the motion prediction module is a constant turn rate and velocity model. The results of the ablation study are summarized in Table 6.4 and Table 6.5. In open-loop testing, we select the planning and prediction errors as the main metrics, and the planning error is averaged over three timesteps (1 s, 3 s, and 5 s) and ADE is used as the prediction error. In closed-loop testing, we employ the failure rate (sum of the collision and off-route rates), progress, and average position error (3 s, 5 s, and 10 s) as the main metrics.

TABLE 6.4: Ablation study of each component in open-loop testing

Model	Planning error (m)	Prediction error (m)
Base (ours)	1.583	0.740
Base w/ oracle prediction	1.388	–
No learnable cost function	1.834	0.740
No learnable initialization	2.672	0.740
No learnable prediction	1.811	1.871

TABLE 6.5: Ablation study of each component in closed-loop testing

Model	Failure (%)	Progress (m)	Position error (m)
Base (ours)	5	76.28	4.083
Base w/ oracle prediction	6	76.16	4.153
No learnable cost function	13	69.09	5.881
No learnable initialization	17	57.52	6.571
No learnable prediction	19	44.01	9.971

The open-loop testing results in Table 6.4 indicate that learnable initialization is the most important part of the framework to ensure the planned trajectory is close to the human-driving one. This suggests that a good initial plan for the planner is critical to solving the optimization and learnable initialization could substantially improve the planning performance. Another important factor for planning is the cost function, and we demonstrate that learning the cost function from data can render the planned trajectories closer to humans. Surprisingly, using only a physics-based non-learnable prediction model would not significantly deteriorate the planning performance in an open-loop setting though the prediction error is notably higher. Using the oracle predictor can improve the motion planner's performance in open-loop testing, which suggests that the proposed motion planner performs well and could achieve even better performance with more training data and

improved prediction accuracy. The closed-loop testing results in Table 6.5 reveal that the prediction module plays a more critical role in ensuring safety and human likeness. The gap between open-loop and closed-loop performance is due to that the closed-loop testing set contains many interactive scenarios, where accurately predicting other agents' future states is required. In contrast to open-loop testing, our method can deliver comparable performance to the oracle prediction method in closed-loop testing, marking the advantage of our method (planning-aware prediction) when deploying the system in practice. In addition, learnable initialization and cost function are necessary to stabilize the planning process (solving the optimization problem) and obtain better planning performance, which otherwise could lead to instability to find viable solutions and consequently worse safety and reliability. In summary, all three learnable components in our framework are integral to maintaining the final planning performance.

Effects of weights of the training loss function. We investigate the influence of the weights in the loss function (Equation (6.11)) and test the trained models in the open-loop planning setting. To reveal the sensitivity of the framework performance to these values, we set up the following groups of loss function weights: (1) the default parameter setting in the experiment, (2) the weight for the prediction loss λ_1 is increased by $5\times$, (3) the weight for the prediction loss λ_1 is decreased to 1/5 of the default value, (4) the weight for the score loss λ_2 is increased by $5\times$. The weights for imitation loss λ_3 and planning cost λ_4 are fixed, and the same metrics in Table 6.4 are used to evaluate the framework's performance.

TABLE 6.6: Effects of weights of the loss function in open-loop testing

No.	λ_1	λ_2	λ_3	Planning error (m)	Prediction error (m)
1	0.5	1	1	1.583	0.740
2	2.5	1	1	1.837	0.733
3	0.1	1	1	1.642	1.244
4	0.5	5	1	1.704	0.820

The results in Table 6.6 indicate that the default loss parameters in our experiments (No.1) achieve the best planning performance. Although increasing the weight of prediction loss (No.2) can slightly improve the prediction error, the planning performance is compromised because the loss signal of the initial plan or cost function in motion planning is largely neglected in the total loss function. However, decreasing the weight of prediction loss (No.3) still does not bring better planning

performance because the prediction results are required to be more accurate for both the ego vehicle and other agents. In addition, increasing the weight of score loss (No.4) may not improve probability evaluation and even lead to worse planning or prediction performance. The results suggest that our multi-task learning framework requires a trade-off between the planning and prediction tasks, and the default loss function weights can strike a better balance between the tasks.

Effects of the step size and planning iterations. We investigate the effects of the step size (α in Equation (6.10)) and planning iterations in the motion planner on the final open-loop planning test. Note that the different parameters are only applied in the training process while the parameters of the motion planner in the testing process are the same. We explore different parameter settings in Table 6.7, where the value of the step size α ranges from 0.2 to 1, and the maximum planning iterations is 3 for speed consideration. The same metrics for open-loop testing are used to evaluate the performance of the trained models.

TABLE 6.7: Effects of the planner update step size and planning iterations in open-loop testing

step size α	planning iteration	Planning error (m)	Prediction error (m)
0.4	2	1.583	0.740
1.0	2	1.856	0.867
0.4	3	1.551	0.725
0.2	3	1.549	0.797

The results in Table 6.7 indicate that using the default planner setting provides satisfactory results in the final open-loop planning test. Setting the step size to the maximum value ($\alpha = 1$) could significantly impair the testing performance because the large step size makes the optimization process unstable, which negatively impacts the learning process of the prediction module and the cost function weights. While increasing the number of planning iterations with the default step size ($\alpha = 0.4$) could slightly improve the planning and prediction metrics, this would also result in longer computing time for the planner, making it an unnecessary choice. On the other hand, using a smaller step size ($\alpha = 0.2$) along with additional planning iterations could also achieve comparable performance to the default setting, but may still be an unnecessary choice. Overall, the default planner parameter setting provides a good balance between final performance and computational efficiency.

Results of cost function learning. We report detailed results for the learned cost function and hand-tuned one in Table 6.8. The traffic light violation and safety weights are fixed at 10 because they are considered constraints, and other parameters are learnable. The results indicate that using the learned cost function as opposed to the hand-tuned cost function, significantly reduces planning errors in the open-loop planning test and the failure rate in the closed-loop test. Generally, the learned weights are reasonable from the viewpoint of human drivers. Adhering to the driving route (for both positions and directions) is the most critical factor, and driving comfort (acceleration and change rate of steering) is also an important factor to optimize. The steering angle is not a significant factor in the cost since the vehicle primarily needs to follow the route, but speed and jerk cost terms are assigned with smaller weights due to their relatively large scales. It is important to note that the cost terms have different scales, and the weights are learned to balance the scales to facilitate the optimization process. This means that the learned cost function weights may not reflect their actual weights in human driving and are thereby unsuitable for other frameworks.

TABLE 6.8: Detailed results for the learned planning cost

Cost Weights	ω_1 ($\mathbf{c}^{speed}$)	ω_2 ($\mathbf{c}^{acc}$)	ω_3 ($\mathbf{c}^{jerk}$)	ω_4 ($\mathbf{c}^{steer}$)	ω_5 ($\mathbf{c}^{rate}$)	ω_6 ($\mathbf{c}^{pos}$)	ω_7 ($\mathbf{c}^{head}$)	ω_8 ($\mathbf{c}^{traffic}$)	ω_9 ($\mathbf{c}^{safety}$)	Planning error (m)	Failure (%)
Hand-tuned	0.1	0.5	0.1	0.01	0.5	0.5	5	10	10	1.834	13
Learned	0.055	0.29	0.13	0.014	0.46	0.31	3.23	10	10	**1.583**	**5**

6.5.4 Discussions

Strengths. The direct benefit of our method is to integrate structural information and domain knowledge into machine learning models, or from a different perspective, to apply machine learning models to some hard-to-solve points in traditional optimization problems. We apply our method to the challenging autonomous driving task and focus on solving real-world problems (e.g., how to specify the cost function and how other agents' actions would affect the ego) by learning from real-world data. The integration of prediction, cost function, and planning modules in our framework leads to end-to-end, decision-focused learning, that trains the whole pipeline to optimize the final planning performance directly. Experiments demonstrate that the proposed method has excellent open-loop and closed-loop performance and robustness against distributional shifts and adversarial agents, despite only learning from offline data. In addition, the proposed method outperforms the

separated planning and prediction method in both planning and prediction performance, which underscores the advantage of our method to integrate the prediction and planning modules in an end-to-end differentiable way.

Limitations and future work. Some limitations of this work should be acknowledged. First is the computation time of the framework. The average runtime of the prediction+planning step is 0.012 s per sample at training (with NVIDIA RTX 3080 GPU, 2 iterations) and 1.78 s at testing (with AMD 3900X CPU, max 50 iterations). The bottleneck in the framework is the computation of Jacobin in solving the least-squares optimization. However, we need to note that this work is just a prototype of the proposed framework and the running efficiency has not been optimized. We plan to further improve the efficiency of the computation pipeline to reduce the runtime in solving the optimization, potentially enabling the framework to run near real-time. Another limitation is that we do not thoroughly consider the uncertainty or multi-modality of human behavior prediction, only taking the most probable one as the planner input. In future work, the planner can take all modalities and uncertainties along the timesteps into account to make more informed decisions or perform contingency planning.

6.6 Conclusions

We propose a novel differentiable integrated multi-agent interactive prediction and motion planning framework, which is trained end-to-end from real-world driving data. A Transformer-based predictor is established to predict joint future trajectories of surrounding agents and provide an initial guess for the planner. The predicted trajectories, initial plan, and learnable cost function are channeled to an optimization-based differentiable motion planner, allowing every component in the framework to be differentiable. The framework is validated on a large-scale urban driving dataset in both open-loop and closed-loop testing. The open-loop testing results reveal that our proposed method outperforms traditional pipeline methods and IL-based methods in terms of planning and prediction performance, ride comfort, and safety metrics. The closed-loop testing results indicate that planning-based methods significantly outperform IL-based methods (though have better similarity to human trajectories in open-loop testing), which suggests that our method overcomes the distributional shift problem common in offline learning.

Moreover, the proposed method outperforms the pipeline method in closed-loop testing, emphasizing the benefit of joint training of prediction and planning modules. In the ablation study, we find that all learnable components in the framework are integral to maintaining the stability of the optimizer and desired planning performance. The proposed framework has the potential to accelerate the development of autonomous driving decision-making systems by enabling all components to be learned from real-world data.

Chapter 7

Conclusions

7.1 Summary

We summarize the findings of this thesis, which proposes a series of learning-based methodologies for decision-making in AVs. The primary aim is to bolster the stability, adaptability, and alignment of decision-making systems, which currently constitute significant barriers to the widespread deployment of AVs. Our methodologies are based on a comprehensive autonomous intelligence architecture, comprising three essential components: the actor, the cost, and the world model. We develop various learning frameworks and neural architectures to enhance these modules, subsequently augmenting the decision-making system's performance in terms of safety, interactivity, and human likeness.

In Chapter 2, we propose an RL-based actor module, which bypasses the need for building a world model, the most complex component in our framework. Despite their effectiveness, RL-based methods struggle to align agent behavior with human expectations and require extensive interaction steps to reach functional capacity. To address these issues, we develop a novel learning framework and algorithm that incorporate human prior knowledge into RL training through a combination of IL and RL. The proposed framework operates in three principal steps: expert demonstration, policy derivation, and reinforcement learning. Initially, a human expert demonstrates their approach to a task, potentially with their personal preferences. Subsequently, the policy derivation stage applies IL and uncertainty estimation to derive an expert policy from the expert demonstration data. In the final RL step,

we propose an actor-critic-based algorithm to integrate the imitative expert policy into the RL training process. The results indicate that our proposed approach significantly improves sample efficiency in comparison to baseline algorithms, and successfully generates distinct driving styles in complex urban environments that closely mimic human driving behaviors.

In Chapter 3, we introduce an IRL-based cost module, facilitating the modeling and alignment of human driving behavior. Within the decision-making framework, we simplify the actor and world model modules, focusing on learning the personalized cost functions underlying human driving behaviors. Specifically, to apply the maximum entropy IRL algorithm to infer the reward function within our framework, we propose a structural assumption about human driving behaviors, centered on discrete latent intentions that guide continuous low-level control actions. Based on our assumption, we first use a polynomial trajectory sampler to generate candidate trajectories covering the high-level decisions and the desired speed. These trajectories then serve to approximate the partition function in the maximum entropy IRL framework. Furthermore, we devise a simplistic world model to predict the trajectories of surrounding vehicles and assess the outcomes of the generated trajectories. We validate our proposed method utilizing naturalistic human driving data in highway driving scenarios. The results demonstrate that our IRL-based cost module has enabled a personalized driving experience, closely reflecting human driving preferences and behaviors. Furthermore, the personalized cost learning method notably surpasses the general cost modeling method in terms of robustness and human likeness.

In Chapter 4, we shift our focus to the most complex module of the decision-making framework, the world model, and develop a joint multi-agent prediction model. This model leverages Transformers to model interactions among agents in driving scenarios. We approach the interaction prediction problem by adopting hierarchical game theory and introducing our proposed framework for practical implementation. Specifically, our method presents a novel Transformer decoder structure that iteratively refines the interaction prediction process by utilizing the prediction results from the previous level in tandem with the common environmental context. Moreover, we propose a learning process that regulates an agent's behavior at the current level to respond to other agents' behaviors at the preceding level. We validate our model through rigorous experiments on an extensive

real-world driving dataset, which demonstrates that our model can achieve state-of-the-art prediction accuracy in the interaction prediction task. Furthermore, we validate the model's capability to jointly reason about the ego agent's motion plans and other agents' behaviors in both open-loop and closed-loop planning tests, showcasing performance superior to a variety of baseline methods.

In Chapter 5, we propose a comprehensive behavior planning framework integrating all three core modules. Notably, we emphasize the world model part and introduce a CMP model, which predicts the possible responses of other agents to the AV's potential actions. Our proposed framework comprises three integral modules: behavior generation, CMP, and scoring. The behavior generation module produces a diverse set of trajectory proposals, while the CMP module forecasts other agents' future trajectories jointly conditioned on each candidate plan. The scoring module evaluates these candidate plans using a cost function learned through maximum entropy IRL. We conduct comprehensive experiments to validate the proposed framework on a large-scale real-world urban driving dataset. The results demonstrate that the CMP module can predict multi-modal futures given a candidate plan and provide reactive predictions to different plans. The CMP module not only improves the prediction accuracy but also facilitates the downstream scoring module in more effectively evaluating candidate decisions, leading to the emulation of human-like behaviors. Moreover, the scoring module, equipped with the learned cost function, can properly select plans that align closely with human driving behaviors.

In Chapter 6, we construct a differentiable and integrated framework encompassing the three core modules, allowing for end-to-end training of the entire framework using real-world driving data. Specifically, our framework employs a differentiable nonlinear optimizer as the motion planner, which takes as input the predicted trajectories of surrounding agents produced by the neural model and optimizes the AV's trajectory, thereby enabling all operations to be differentiable, including the cost function weights. The proposed framework is trained on a large-scale real-world driving dataset and validated in both open-loop and closed-loop modes. The open-loop testing results reveal that the proposed method outperforms baseline methods across multiple metrics and provides planning-centric prediction results, allowing the planning module to generate trajectories that closely resemble those of human drivers. In closed-loop testing, our method exhibits superior performance

over various baseline methods, showing its ability to handle complex urban driving scenarios and robustness against distributional shifts.

This thesis represents a significant contribution to the development of learning-based methodologies for AV decision-making, with the aim of creating more human-like driving behavior. As the field of AV development continues to evolve, we hope that the research presented in this thesis will serve as a foundation for further advancements in human-centric AV decision-making, ultimately contributing to the broader goal of safer, more efficient, and more enjoyable transportation experiences for all.

7.2 Future Work

While this thesis has made significant progress, it is important to acknowledge the limitations of the research and identify potential opportunities for future studies in the field of advanced and human-centered vehicle autonomy. The following sections outline some prospective avenues for future research.

Real-world deployment and evaluation. Conducting real-world deployment and evaluation of the proposed methodologies is crucial for assessing their practical viability and impact. However, due to constraints in time and resources, extensive real-world experiments were unfeasible during this research. Nevertheless, to address this limitation and pave the way for future studies, we outline several potential plans for real-world experimentation. *Field trials in controlled environments*: conducting field trials in controlled environments, such as closed test tracks or designated areas, allows for the evaluation of the proposed methodologies under controlled conditions. These trials can provide insights into the system's behavior, performance, and limitations in a more realistic setting while minimizing potential risks associated with public roads. *Collaborative partnerships with industry stakeholders*: establishing collaborations with automotive manufacturers or technology companies can provide access to real-world AV systems and the necessary resources for conducting large-scale evaluations. Such partnerships can facilitate the integration of the proposed methodologies into existing AV platforms, enabling comprehensive performance assessments. Future studies focusing on real-world

experimentation have the potential to bridge the gap between theoretical advancements and practical implementation, thereby enhancing our understanding of the proposed methodologies in real-world settings.

Handling real-world uncertainties. Achieving robust and reliable performance in vehicle autonomy systems necessitates addressing uncertainties that arise in real-world scenarios. These uncertainties stem from various sources, including perception, human behavior, and modeling errors. *Perception Uncertainties*: real-world environments are complex, leading to uncertainties in perception. Factors such as adverse weather, occlusions, sensor limitations, and lighting variations impact the accuracy of perception modules.. *Human Behavior Uncertainties*: human behavior introduces inherent uncertainty, necessitating the development of ML models capable of capturing distinct behavior patterns and enhancing the prediction of human actions. *Modeling Errors*: ML models employed in vehicle autonomy systems are prone to errors, especially when predicting human behaviors. These errors can result in unpredictability and erratic behavior, potentially affecting decision-making and control algorithms. To tackle these challenges, it is imperative to develop algorithms, tools, or methods that effectively handle real-world uncertainties. These solutions should deliver robust and reliable performance even in situations where one or more uncertainties exist in real-world scenarios. Additionally, designing safety-assurance algorithms and tools plays a vital role in ensuring safety in autonomous systems and enabling the confident integration of ML models within the decision-making loop.

Development of decision-focused autonomy architecture. The current AV system follows a more fragmented architecture, where each module is developed and evaluated independently. This fragmented approach may potentially compromise the overall downstream decision-making performance. Moreover, improving the accuracy of specific modules may not necessarily enhance the performance of the decision-making task, leading to wasted efforts. Consequently, there is a pressing need to adopt a more integrated and holistic approach by developing a decision-focused system architecture. Fig. 7.1 illustrates key elements of the next-generation autonomy architecture, which emphasizes integration, modularity, and decision-focused design. The utilization of modular architecture is crucial due to its reusability and interpretability properties, particularly in safety-critical contexts. Additionally, the integrated nature of the architecture, featuring neural

connections and end-to-end trainability, can eliminate information bottlenecks and optimize efficiency, thus directly improving the final decision-making performance. This decision-focused and highly integrated architecture combines the benefits of classical decoupled designs with the advantages of end-to-end architecture. Another significant feature of the next-generation autonomy architecture is the inclusion of humans in the decision-making loop. Recognizing the potential synergies between human expertise and AI capabilities, the architecture aims to incorporate human knowledge into decision-making processes, resulting in more reliable and human-aligned outcomes. Incorporating humans in the loop involves enhancing the intelligence of AI systems through human feedback and effectively communicating the intentions of AI systems. By doing so, the architecture improves the interpretability, transparency, and trustworthiness of the overall system.

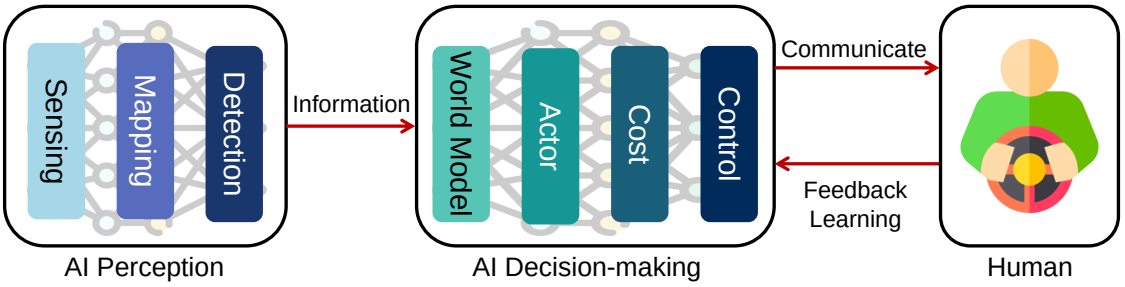


FIGURE 7.1: Key elements of next-generation autonomy architecture: modular, integrated, learnable, and inclusion of humans in the loop

Connected autonomy. Investigating the potential of connected autonomy, wherein AVs communicate with each other and with the surrounding infrastructure, can lead to significant improvements in traffic efficiency, safety, and overall driving experience. By leveraging vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication, AVs can share information about their intentions, trajectories, and environmental observations. This connectivity enables more accurate perceptions of their surroundings and more precise predictions of other agents' behaviors, thereby facilitating safe, coordinated, and cooperative decision-making. Research in connected autonomy can delve into the development of novel algorithms for collaborative perception and planning, enabling AVs to navigate complex traffic situations more effectively while maintaining human-like driving behavior.

List of Author's Awards and Publications

Awards

- **Best Paper Runner-up Award**, "Learning Interaction-aware Motion Prediction Model for Decision-making in Autonomous Driving", *IEEE ITSC*, 2023.
- **Innovation Award**, "nuPlan Planning Challenge", *CVPR Workshop on End-to-End Autonomous Driving*, 2023.
- **Winner (3rd Place)**, "Waymo Open Dataset Motion Prediction Challenge", *CVPR Workshop on Autonomous Driving*, 2023.
- **Winner (3rd Place), Most Innovative Award**, "Driving SMARTS Competition", *NeurIPS Competition Track*, 2022.
- **Winner (2nd Place)**, "Waymo Open Dataset Occupancy and Flow Prediction Challenge", *CVPR Workshop on Autonomous Driving*, 2022.
- **Winner (2nd Place)**, "IEEE VTS Motor Vehicle Challenge", *IEEE Vehicular Technology Society*, 2022.
- **Outstanding Student Paper**, "Digital Twin-enabled Reinforcement Learning for End-to-end Autonomous Driving", *IEEE DTPI*, 2021.
- **Winner (2nd Place)**, "Waymo Open Dataset Motion Prediction Challenge", *CVPR Workshop on Autonomous Driving*, 2021.
- **Winner (1st Place)**, "Waymo Open Dataset Interaction Prediction Challenge", *CVPR Workshop on Autonomous Driving*, 2021.

Journal Articles

- **Z. Huang**, H. Liu, J. Wu, C. Lv, "Differentiable Integrated Motion Prediction and Planning with Learnable Cost Function for Autonomous Driving," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.
- **Z. Huang**, H. Liu, J. Wu, C. Lv, "Conditional Predictive Behavior Planning with Inverse Reinforcement Learning for Human-like Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems*, 2023.
- X. He, J. Wu, **Z. Huang,** Z. Hu, J. Wang, A. L. Sangiovanni-Vincentelli, C. Lv, "Fear-Neuro-Inspired Reinforcement Learning for Safe Autonomous Driving," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.
- J. Wu, Y. Zhou, H. Yang, **Z. Huang**, C. Lv., "Human-Guided Reinforcement Learning With Sim-to-Real Transfer for Autonomous Navigation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.
- J. Wu, **Z. Huang**, Z. Hu, C. Lv, "Toward human-in-the-loop AI: Enhancing deep reinforcement learning via real-time human guidance for autonomous driving," *Engineering 21, 75-91*, 2022.
- J. Wu, **Z. Huang**, C. Lv, "Uncertainty-aware model-based reinforcement learning: methodology and application in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, 2022.
- J. Wu, **Z. Huang**, W. Huang, C. Lv, "Prioritized experience-based reinforcement learning with human guidance for autonomous driving," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- X. Mo, **Z. Huang**, Y. Xing, C. Lv, "Multi-agent Trajectory Prediction with Heterogeneous Edge-enhanced Graph Attention Network," *IEEE Transactions on Intelligent Transportation Systems*, 2022.
- **Z. Huang**, J. Wu, C. Lv, "Efficient deep reinforcement learning with imitative expert priors for autonomous driving," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

- **Z. Huang**, J. Wu, C. Lv, "Driving Behavior Modeling using Naturalistic Human Driving Data with Inverse Reinforcement Learning," *IEEE Transactions on Intelligent Transportation Systems*, 2021.
- **Z. Huang**, C. Lv, Y. Xing, J. Wu, "Multi-modal sensor fusion-based deep neural network for end-to-end autonomous driving with scene understanding," *IEEE Sensors Journal 21 (10), 11781-11790*, 2020.

Conference Proceedings

- **Z. Huang**, H. Liu, C. Lv. "GameFormer: Game-theoretic Modeling and Learning of Transformer-based Interactive Prediction and Planning for Autonomous Driving," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- **Z. Huang**, H. Liu, J. Wu, W. Huang, C. Lv. "Learning Interaction-aware Motion Prediction Model for Decision-making in Autonomous Driving," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2023.
- H. Liu, **Z. Huang**, C. Lv. "Occupancy Prediction-Guided Neural Planner for Autonomous Driving," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2023.
- H. Liu, **Z. Huang**, C. Lv, "Multi-modal Hierarchical Transformer for Occupancy Flow Field Prediction in Autonomous Driving," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
- J. Wu, **Z. Huang**, C. Lv, "Recurrent Neural Network-based Predictive Energy Management for Hybrid Energy Storage System of Electric Vehicles," in *IEEE Vehicle Power and Propulsion Conference (VPPC)*, 2022.
- X. Mo, **Z. Huang**, C. Lv, "Stochastic Multimodal Interaction Prediction for Urban Driving," in *IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022.
- **Z. Huang**, X. Mo, C. Lv, "ReCoAt: A Deep Learning-based Framework for Multi-Modal Motion Prediction in Autonomous Driving Application," in

IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), 2022.

- H. Liu, **Z. Huang**, J. Wu, C. Lv, "Improved deep reinforcement learning with expert demonstrations for urban autonomous driving," in *IEEE Intelligent Vehicles Symposium (IV)*, 2022.

- **Z. Huang**, X. Mo, C. Lv, "Multi-modal motion prediction with transformer-based neural network for autonomous driving," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

- J. Wu, **Z. Huang**, P. Hang, C. Huang, N. De Boer, C. Lv, "Digital Twin-enabled Reinforcement Learning for End-to-end Autonomous Driving," in *IEEE International Conference on Digital Twins and Parallel Intelligence (DTPI)*, 2021.

Bibliography

[1] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, 165:113816, 2021. 1

[2] 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. 1

[3] Ardi Tampuu, Tambet Matiisen, Maksym Semikin, Dmytro Fishman, and Naveed Muhammad. A survey of end-to-end driving: Architectures and training methods. *IEEE Transactions on Neural Networks and Learning Systems*, 2020. 1, 134

[4] Yang Tang, Chaoqiang Zhao, Jianrui Wang, Chongzhen Zhang, Qiyu Sun, and Feng Qian. Perception and decision-making of autonomous systems in the era of learning: An overview. *arXiv preprint arXiv:2001.02319*, 2020.

[5] Yu Huang and Yue Chen. Autonomous driving with deep learning: A survey of state-of-art technologies. *arXiv preprint arXiv:2006.06091*, 2020. 1

[6] Jiageng Mao, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 3d object detection for autonomous driving: A comprehensive survey. *International Journal of Computer Vision*, pages 1–55, 2023. 2, 6

[7] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[8] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34:12077–12090, 2021.

[9] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *Computer*

Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IX, pages 1–18. Springer, 2022. 2

[10] Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62, 2022. 3, 4, 5

[11] Peng Hang, Chen Lv, Yang Xing, Chao Huang, and Zhongxu Hu. Human-like decision making for autonomous driving: A noncooperative game theoretic approach. *IEEE Transactions on Intelligent Transportation Systems*, 22(4): 2076–2087, 2020. 3

[12] Nam Dinh Van, Muhammad Sualeh, Dohyeong Kim, and Gon-Woo Kim. A hierarchical control system for autonomous driving towards urban challenges. *Applied Sciences*, 10(10):3543, 2020. 3, 102

[13] Qi Li, Yue Wang, Yilun Wang, and Hang Zhao. Hdmapnet: An online hd map construction and evaluation framework. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4628–4634, 2022. doi: 10.1109/ICRA46639.2022.9812383. 6

[14] Yanjun Huang, Jiatong Du, Ziru Yang, Zewei Zhou, Lin Zhang, and Hong Chen. A survey on trajectory-prediction methods for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 7(3):652–674, 2022. doi: 10.1109/TIV.2022.3167103. 6

[15] Zhiyu Huang, Xiaoyu Mo, and Chen Lv. Recoat: A deep learning-based framework for multi-modal motion prediction in autonomous driving application. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 988–993. IEEE, 2022. 76, 105, 151

[16] Zhiyu Huang, Xiaoyu Mo, and Chen Lv. Multi-modal motion prediction with transformer-based neural network for autonomous driving. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2605–2611. IEEE, 2022. 6, 103, 105, 112, 134, 136, 142

[17] Zhiyu Huang, Haochen Liu, Jingda Wu, and Chen Lv. Conditional predictive behavior planning with inverse reinforcement learning for human-like autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 2023. 6

[18] Haochen Liu, Zhiyu Huang, and Chen Lv. Multi-modal hierarchical transformer for occupancy flow field prediction in autonomous driving. In *2023 International Conference on Robotics and Automation (ICRA)*, 2023. 6

[19] Zhiyu Huang, Jingda Wu, and Chen Lv. Driving behavior modeling using naturalistic human driving data with inverse reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 7, 134

[20] Zhiyu Huang, Chen Lv, Yang Xing, and Jingda Wu. Multi-modal sensor fusion-based deep neural network for end-to-end autonomous driving with scene understanding. *IEEE Sensors Journal*, 21(10):11781–11790, 2021. doi: 10.1109/JSEN.2020.3003121. 7, 16, 44, 77, 104, 137

[21] Zhiyu Huang, Jingda Wu, and Chen Lv. Efficient deep reinforcement learning with imitative expert priors for autonomous driving. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 7, 104

[22] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–18, 2021. 14

[23] Ammar Haydari and Yasin Yilmaz. Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–22, 2020. 14

[24] Zhiyu Huang, Jingda Wu, and Chen Lv. Driving behavior modeling using naturalistic human driving data with inverse reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–13, 2021. doi: 10.1109/TITS.2021.3088935. 14

[25] Sascha Rosbach, Vinit James, Simon Großjohann, Silviu Homoceanu, and Stefan Roth. Driving with style: Inverse reinforcement learning in general-purpose planning for automated driving. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2658–2665. IEEE, 2019. 14, 48, 106

[26] Jeffrey Hawke, Richard Shen, Corina Gurau, Siddharth Sharma, Daniele Reda, Nikolay Nikolov, Przemysław Mazur, Sean Micklethwaite, Nicolas Griffiths, Amar Shah, et al. Urban driving with conditional imitation learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 251–257. IEEE, 2020. 16

[27] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019. doi: 10.15607/RSS.2019.XV.031. 16

[28] Jianyu Chen, Shengbo Eben Li, and Masayoshi Tomizuka. Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 16

[29] Praveen Palanisamy. Multi-agent connected autonomous driving using deep reinforcement learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020. 16

[30] Zhangjie Cao, Erdem Biyik, Woodrow Wang, Allan Raventos, Adrien Gaidon, Guy Rosman, and Dorsa Sadigh. Reinforcement learning based control of imitative policies for near-accident driving. In *Robotics: Science and Systems*, 2020. 16

[31] Dong Chen, Longsheng Jiang, Yue Wang, and Zhaojian Li. Autonomous driving using safe reinforcement learning by incorporating a regret-based human lane-changing decision model. In *2020 American Control Conference (ACC)*, pages 4355–4361. IEEE, 2020. 17

[32] Hanna Krasowski, Xiao Wang, and Matthias Althoff. Safe reinforcement learning for autonomous lane changing using set-based prediction. In *2020 IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020. 17

[33] Jingliang Duan, Shengbo Eben Li, Yang Guan, Qi Sun, and Bo Cheng. Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data. *IET Intelligent Transport Systems*, 14(5): 297–305, 2020. 17

[34] 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. *arXiv preprint arXiv:2104.07246*, 2021. 17

[35] Xiaodan Liang, Tairui Wang, Luona Yang, and Eric Xing. Cirl: Controllable imitative reinforcement learning for vision-based self-driving. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 584–599, 2018. 17

[36] Mark Pfeiffer, Samarth Shukla, Matteo Turchetta, Cesar Cadena, Andreas Krause, Roland Siegwart, and Juan Nieto. Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations. *IEEE Robotics and Automation Letters*, 3(4):4423–4430, 2018. 17

[37] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870, 2018. 17, 19

[38] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017. 17

[39] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep Q-learning from demonstrations. In *AAAI*, pages 3223–3230, 2018. 17

[40] Haochen Liu, Zhiyu Huang, and Chen Lv. Improved deep reinforcement learning with expert demonstrations for urban autonomous driving. *arXiv preprint arXiv:2102.09243*, 2021. 17

[41] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. 18

[42] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019. 18

[43] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018. 19, 29

[44] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017. 20, 21

[45] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016. 21

[46] Rhiannon Michelmore, Matthew Wicker, Luca Laurenti, Luca Cardelli, Yarin Gal, and Marta Kwiatkowska. Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7344–7350. IEEE, 2020. 21

[47] Panagiotis Tigas, Angelos Filos, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. Robust imitative planning: Planning from demonstrations under uncertainty. In *Machine Learning for Autonomous Driving Workshop at the 33rd Conference on Neural Information Processing Systems*, 2019. 21

[48] Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2020. 25

[49] Ming Zhou, Jun Luo, Julian Villela, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, et al. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. In *Proceedings of the 4th Conference on Robot Learning (CoRL)*, 2020. 26

[50] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 29

[51] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016. 30, 147

[52] Liangzhi Li, Kaoru Ota, and Mianxiong Dong. Humanlike driving: Empirical decision-making system for autonomous vehicles. *IEEE Transactions on Vehicular Technology*, 67(8):6814–6823, 2018. 44

[53] Chao Huang, Hailong Huang, Peng Hang, Hongbo Gao, Jingda Wu, Zhiyu Huang, and Chen Lv. Personalized trajectory planning and control of lane-change maneuvers for autonomous driving. *IEEE Transactions on Vehicular Technology*, 2021. 44

[54] Yifang Ma, Zhenyu Wang, Hong Yang, and Lin Yang. Artificial intelligence applications in the development of autonomous vehicles: a survey. *IEEE/-CAA Journal of Automatica Sinica*, 7(2):315–329, 2020. 44

[55] Parham M Kebria, Abbas Khosravi, Syed Moshfeq Salaken, and Saeid Nahavandi. Deep imitation learning for autonomous vehicles based on convolutional neural networks. *IEEE/CAA Journal of Automatica Sinica*, 7(1): 82–95, 2020. 44

[56] Liting Sun, Wei Zhan, Yeping Hu, and Masayoshi Tomizuka. Interpretable modelling of driving behaviors in interactive driving scenarios based on cumulative prospect theory. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 4329–4335. IEEE, 2019. 45

[57] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008. 45, 51, 103, 106, 113

[58] Kyle Brown, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Modeling and prediction of human driver behavior: A survey. *arXiv preprint arXiv:2006.08832*, 2020. 46

[59] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62 (2):1805–1824, 2000. 46, 56

[60] Arne Kesting, Martin Treiber, and Dirk Helbing. General lane-changing model MOBIL for car-following models. *Transportation Research Record*, 1999(1):86–94, 2007. 46

[61] Wei Yuan, Zhen Li, and Chang Wang. Lane-change prediction method for adaptive cruise control system with hidden markov model. *Advances in Mechanical Engineering*, 10(9):1–9, 2018. 46

[62] Xiaoyu Mo, Yang Xing, and Chen Lv. Interaction-aware trajectory prediction of connected vehicles using CNN-LSTM networks. In *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, pages 5057–5062. IEEE, 2020. 46

[63] 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, 2020. 46

[64] Mohammad Siami, Mohsen Naderpour, and Jie Lu. A mobile telematics pattern recognition framework for driving behavior extraction. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–14, 2020. 47

[65] Stewart Birrell, James Taylor, Andrew McGordon, Joonwoo Son, and Paul Jennings. Analysis of three independent real-world driving studies: A data driven and expert analysis approach to determining parameters affecting fuel economy. *Transportation research part D: transport and environment*, 33: 74–86, 2014. 47

[66] Xiaoxiang Na and David Cole. Theoretical and experimental investigation of driver noncooperative-game steering control behavior. *IEEE/CAA Journal of Automatica Sinica*, 8(1):189–205, 2020. 47

[67] Yang Xing, Zhongxu Hu, Zhiyu Huang, Chen Lv, Dongpu Cao, and Efstathios Velenis. Multi-scale driver behaviors reasoning system for intelligent vehicles based on a joint deep learning framework. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 4410–4415. IEEE, 2020. 47

[68] David Silver, J Andrew Bagnell, and Anthony Stentz. Learning autonomous driving styles and maneuvers from expert demonstration. In *Experimental Robotics*, pages 371–386. Springer, 2013. 47

[69] Markus Kuderer, Shilpa Gulati, and Wolfram Burgard. Learning driving styles for autonomous vehicles from demonstration. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2641–2646. IEEE, 2015. 48, 106

[70] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36(10):1073–1087, 2017. 48

[71] Liting Sun, Wei Zhan, and Masayoshi Tomizuka. Probabilistic prediction of interactive driving behavior via hierarchical inverse reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2111–2117. IEEE, 2018. 48

[72] Yeping Hu, Liting Sun, and Masayoshi Tomizuka. Generic prediction architecture considering both rational and irrational driving behaviors. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3539–3546. IEEE, 2019. 48

[73] Sergey Levine and Vladlen Koltun. Continuous inverse optimal control with locally optimal examples. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 475–482, 2012. 48, 51, 134

[74] David Sierra González, Ozgur Erkent, Víctor Romero-Cano, Jilles Dibangoye, and Christian Laugier. Modeling driver behavior from demonstrations in dynamic environments using spatiotemporal lattices. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018. 48

[75] Donghao Xu, Zhezhang Ding, Xu He, Huijing Zhao, Mathieu Moze, François Aioun, and Franck Guillemard. Learning from naturalistic driving data for human-like autonomous highway driving. *IEEE Transactions on Intelligent Transportation Systems*, 2020. 48

[76] Zheng Wu, Liting Sun, Wei Zhan, Chenyu Yang, and Masayoshi Tomizuka. Efficient sampling-based maximum entropy inverse reinforcement learning with application to autonomous driving. *IEEE Robotics and Automation Letters*, 5(4):5355–5362, 2020. 49

[77] Monica Babeş-Vroman, Vukosi Marivate, Kaushik Subramanian, and Michael Littman. Apprenticeship learning about multiple intentions. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 897–904, 2011. 49

[78] Giorgia Ramponi, Amarildo Likmeta, Alberto Maria Metelli, Andrea Tirinzoni, and Marcello Restelli. Truly batch model-free inverse reinforcement learning about multiple intentions. In *International Conference on Artificial Intelligence and Statistics*, pages 2359–2369. PMLR, 2020. 49

[79] Vassili Alexiadis, James Colyar, John Halkias, Rob Hranac, and Gene McHale. The next generation simulation program. *Institute of Transportation Engineers. ITE Journal*, 74(8):22–26, 2004. 57

[80] Maximilian Naumann, Liting Sun, Wei Zhan, and Masayoshi Tomizuka. Analyzing the suitability of cost functions for explaining and imitating human driving behavior based on inverse reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5481–5487. IEEE, 2020. 59

[81] Niclas Evestedt, Erik Ward, John Folkesson, and Daniel Axehill. Interaction aware trajectory planning for merge scenarios in congested traffic situations. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 465–472. IEEE, 2016. 61

[82] Liting Sun, Zheng Wu, Hengbo Ma, and Masayoshi Tomizuka. Expressing diverse human driving behavior with probabilistic rewards and online inference. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2020–2026, 2020. 71

[83] 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. IEEE, 2019. 74, 76

[84] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020. 105, 136

[85] Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15303–15312, 2021. 90, 103, 136

[86] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi Pang Lam, Dragomir Anguelov, et al. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7814–7821. IEEE, 2022. 74, 84, 88

[87] Jiquan Ngiam, Vijay Vasudevan, Benjamin Caine, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene transformer: A unified architecture for predicting future trajectories of multiple agents. In *International Conference on Learning Representations*, 2021. 74, 76, 84, 90, 103, 105, 134, 137, 138

[88] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. *arXiv preprint arXiv:2207.05844*, 2022. 76, 80

[89] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 2022. 76, 80, 88, 89, 90

[90] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris M Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9813–9823, 2021.

[91] Zikang Zhou, Luyao Ye, Jianping Wang, Kui Wu, and Kejie Lu. Hivt: Hierarchical vector transformer for multi-agent motion prediction. In *Proceedings*

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8823–8833, 2022. 136

[92] Xiaosong Jia, Penghao Wu, Li Chen, Hongyang Li, Yu Liu, and Junchi Yan. Hdgt: Heterogeneous driving graph transformer for multi-agent trajectory prediction via scene encoding. *arXiv preprint arXiv:2205.09753*, 2022. 74

[93] Haoran Song, Wenchao Ding, Yuxuan Chen, Shaojie Shen, Michael Yu Wang, and Qifeng Chen. Pip: Planning-informed trajectory prediction for autonomous driving. In *European Conference on Computer Vision*, pages 598–614. Springer, 2020. 74, 77, 103, 105, 137

[94] Ekaterina Tolstaya, Reza Mahjourian, Carlton Downey, Balakrishnan Vadarajan, Benjamin Sapp, and Dragomir Anguelov. Identifying driver interactions via conditional behavior prediction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3473–3479. IEEE, 2021. 103, 105

[95] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision*, pages 683–700. Springer, 2020. 76

[96] Qiao Sun, Xin Huang, Junru Gu, Brian C Williams, and Hang Zhao. M2i: From factored marginal trajectory prediction to interactive prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6543–6552, 2022. 90, 105

[97] Jose Luis Vazquez Espinoza, Alexander Liniger, Wilko Schwarting, Daniela Rus, and Luc Van Gool. Deep interactive motion prediction and planning: Playing games with motion prediction models. In *Learning for Dynamics and Control Conference*, pages 1006–1019. PMLR, 2022. 77, 137, 138

[98] Zhiyu Huang, Haochen Liu, Jingda Wu, and Chen Lv. Conditional predictive behavior planning with inverse reinforcement learning for human-like autonomous driving. *arXiv preprint arXiv:2212.08787*, 2022. 74, 77

[99] Wenshuo Wang, Letian Wang, Chengyuan Zhang, Changliu Liu, and Lijun Sun. Social interactions for autonomous driving: A review and perspectives. *arXiv preprint arXiv:2208.07541*, 2022. 74

[100] Nan Li, Dave W Oyler, Mengxuan Zhang, Yildiray Yildiz, Ilya Kolmanovsky, and Anouck R Girard. Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems. *IEEE Transactions on control systems technology*, 26(5):1782–1797, 2017.

[101] Peng Hang, Chen Lv, Chao Huang, Yang Xing, and Zhongxu Hu. Cooperative decision making of connected automated vehicles at multi-lane merging zone: A coalitional game approach. *IEEE Transactions on Intelligent Transportation Systems*, 23(4):3829–3841, 2021. 74

[102] Miguel A Costa-Gomes, Vincent P Crawford, and Nagore Iriberri. Comparing models of strategic thinking in van huyck, battalio, and beil's coordination games. *Journal of the European Economic Association*, 7(2-3):365–376, 2009. 75

[103] James R Wright and Kevin Leyton-Brown. Beyond equilibrium: Predicting human behavior in normal-form games. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010. 75

[104] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016. 76

[105] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Home: Heatmap output for future motion estimation. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 500–507. IEEE, 2021. 76

[106] Xiaoyu Mo, Zhiyu Huang, Yang Xing, and Chen Lv. Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network. *IEEE Transactions on Intelligent Transportation Systems*, 2022. 76, 90, 105, 134, 136, 141

[107] Zhiyu Huang, Xiaoyu Mo, and Chen Lv. Multi-modal motion prediction with transformer-based neural network for autonomous driving. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2605–2611. IEEE, 2022.

[108] Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Scept: Scene-consistent, policy-based trajectory predictions for planning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17103–17112, 2022. 83

[109] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Gohome: Graph-oriented heatmap output for future motion estimation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9107–9114. IEEE, 2022. 76

[110] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018. 77, 104

[111] Danfei Xu, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Bits: Bi-level imitation for traffic simulation. *arXiv preprint arXiv:2208.12403*, 2022. 77

[112] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020. 77, 88, 95, 151

[113] Zhiyu Huang, Haochen Liu, Jingda Wu, and Chen Lv. Differentiable integrated motion prediction and planning with learnable cost function for autonomous driving. *arXiv preprint arXiv:2207.10422*, 2022. 77, 85, 88, 95, 111, 112

[114] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14403–14412, 2021. 137

[115] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019.

[116] Alexander Cui, Sergio Casas, Abbas Sadat, Renjie Liao, and Raquel Urtasun. Lookout: Diverse multi-future prediction and planning for self-driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16107–16116, 2021. 77, 137

[117] Jerry Liu, Wenyuan Zeng, Raquel Urtasun, and Ersin Yumer. Deep structured reactive planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4897–4904. IEEE, 2021. 83, 137

[118] Niklas Hanselmann, Katrin Renz, Kashyap Chitta, Apratim Bhattacharyya, and Andreas Geiger. King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. *arXiv preprint arXiv:2204.13683*, 2022. 83

[119] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9710–9719, 2021. 84, 90

[120] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. In *CVPR ADP3 workshop*, 2021. 84, 96

[121] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. In *International Conference on Learning Representations*, 2019. 87

[122] Angelos Filos, Panagiotis Tigkas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? In *International Conference on Machine Learning*, pages 3145–3153. PMLR, 2020. 88, 95

[123] Stepan Konev. Mpa: Multipath++ based architecture for motion prediction. *arXiv preprint arXiv:2206.10041*, 2022. 88

[124] David Wu and Yunnan Wu. Air^2 for interaction prediction. *arXiv preprint arXiv:2111.08184*, 2021. 90

[125] Abbas Sadat, Mengye Ren, Andrei Pokrovsky, Yen-Chen Lin, Ersin Yumer, and Raquel Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3949–3956. IEEE, 2019. 102, 134

[126] Andrea Censi, Konstantin Slutsky, Tichakorn Wongpiromsarn, Dmitry Yershov, Scott Pendleton, James Fu, and Emilio Frazzoli. Liability, ethics, and culture-aware behavior specification using rulebooks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8536–8542. IEEE, 2019. 102

[127] Junqing Wei, Jarrod M Snider, Tianyu Gu, John M Dolan, and Bakhtiar Litkouhi. A behavioral planning framework for autonomous driving. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 458–464. IEEE, 2014. 102

[128] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *European Conference on Computer Vision*, pages 541–556. Springer, 2020. 103

[129] Chen Tang, Wei Zhan, and Masayoshi Tomizuka. Interventional behavior prediction: Avoiding overly confident anticipation in interactive prediction. *arXiv preprint arXiv:2204.08665*, 2022. 103

[130] Honghao Gao, Danqing Fang, Junsheng Xiao, Walayat Hussain, and Jung Yoon Kim. Camrl: A joint method of channel attention and multi-dimensional regression loss for 3d object detection in automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–15, 2022. doi: 10.1109/TITS.2022.3219474. 104

[131] Xiaojin Ma, Huahu Xu, Honghao Gao, Minjie Bian, and Walayat Hussain. Real-time virtual machine scheduling in industry iot network: A reinforcement learning method. *IEEE Transactions on Industrial Informatics*, 19(2): 2129–2139, 2023. doi: 10.1109/TII.2022.3211622. 104

[132] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. Neat: Neural attention fields for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15793–15803, 2021. 104

[133] Haochen Liu, Zhiyu Huang, Jingda Wu, and Chen Lv. Improved deep reinforcement learning with expert demonstrations for urban autonomous driving. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 921–928. IEEE, 2022. 104

[134] Jingda Wu, Zhiyu Huang, and Chen Lv. Uncertainty-aware model-based reinforcement learning: methodology and application in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2022.

[135] Jingda Wu, Zhiyu Huang, Zhongxu Hu, and Chen Lv. Toward human-in-the-loop ai: Enhancing deep reinforcement learning via real-time human guidance for autonomous driving. *Engineering*, 2022.

[136] Haochen Liu, Zhiyu Huang, Xiaoyu Mo, and Chen Lv. Augmenting reinforcement learning with transformer-based scene representation learning for decision-making of autonomous driving. *arXiv preprint arXiv:2208.12263*, 2022.

[137] Jingda Wu, Zhiyu Huang, Wenhui Huang, and Chen Lv. Prioritized experience-based reinforcement learning with human guidance for autonomous driving. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 104

[138] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on intelligent transportation systems*, 17(4):1135–1145, 2015. 104, 134, 137

[139] Xiaoyu Mo, Zhiyu Huang, and Chen Lv. Stochastic multimodal interaction prediction for urban driving. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1000–1005. IEEE, 2022. 105

[140] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Gohome: Graph-oriented heatmap output for future motion estimation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9107–9114. IEEE, 2022. 105

[141] DiJia Andy Su, Bertrand Douillard, Rami Al-Rfou, Cheol Park, and Benjamin Sapp. Narrowing the coordinate-frame gap in behavior prediction models: Distillation for efficient and accurate scene-centric motion forecasting. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 653–659. IEEE, 2022. 105

[142] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *2010 IEEE International Conference on Robotics and Automation*, pages 987–993. IEEE, 2010. 108, 109

[143] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 112, 139

[144] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aurélien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9710–9719, October 2021. 115, 149

[145] Oliver Scheel, Luca Bergamini, Maciej Wolczyk, Błażej Osiński, and Peter Ondruska. Urban driver: Learning to drive from real-world demonstrations using policy gradients. In *Conference on Robot Learning*, pages 718–728. PMLR, 2022. 120

[146] Majid Moghadam and Gabriel Hugh Elkaim. An autonomous driving framework for long-term decision-making and short-term trajectory planning on frenet space. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 1745–1750. IEEE, 2021. 125

[147] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004. 128

[148] Jun Ma, Zilong Cheng, Xiaoxue Zhang, Ziyu Lin, Frank L Lewis, and Tong Heng Lee. Local learning enabled iterative linear quadratic regulator for constrained trajectory planning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 134

[149] Peng Hang, Chen Lv, Chao Huang, Jiacheng Cai, Zhongxu Hu, and Yang Xing. An integrated framework of decision making and motion planning for autonomous vehicles considering social behaviors. *IEEE Transactions on Vehicular Technology*, 69(12):14458–14469, 2020. doi: 10.1109/TVT.2020.3040398. 134

[150] Hongbo Gao, Yechen Qin, Chuan Hu, Yuchao Liu, and Keqiang Li. An interacting multiple model for trajectory prediction of intelligent vehicles in typical road traffic scenario. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 134

[151] Siyuan Chen and Jiahai Wang. Heterogeneous interaction modeling with reduced accumulated error for multiagent trajectory prediction. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 134

[152] Yifei Xu, Jianwen Xie, Tianyang Zhao, Chris Baker, Yibiao Zhao, and Ying Nian Wu. Energy-based continuous inverse optimal control. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 134

[153] Ángel Llamazares, Eduardo J Molinos, and Manuel Ocaña. Detection and tracking of moving obstacles (datmo): a review. *Robotica*, 38(5):761–774, 2020. 135

[154] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi Pang Lam, Dragomir Anguelov, et al. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7814–7821. IEEE, 2022. 136

[155] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. THOMAS: Trajectory heatmap output with learned multi-agent sampling. In *International Conference on Learning Representations*, 2022. 136

[156] Xiaosong Jia, Liting Sun, Hang Zhao, Masayoshi Tomizuka, and Wei Zhan. Multi-agent trajectory prediction by combining egocentric and allocentric views. In *5th Annual Conference on Robot Learning*, 2021. URL `https://openreview.net/forum?id=lAtePxetBNb`. 136

[157] Long Chen, Lukas Platinsky, Stefanie Speichert, Błażej Osiński, Oliver Scheel, Yawei Ye, Hugo Grimmett, Luca Del Pero, and Peter Ondruska. What data do we need for training an av motion planner? In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1066–1072. IEEE, 2021. 137

[158] Zhiyu Huang, Jingda Wu, and Chen Lv. Efficient deep reinforcement learning with imitative expert priors for autonomous driving. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 137

[159] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE, 2019.

[160] Szilárd Aradi. Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2020. 137

[161] Boris Ivanovic, Amine Elhafsi, Guy Rosman, Adrien Gaidon, and Marco Pavone. Mats: An interpretable trajectory forecasting representation for planning and control. In *4th Annual Conference on Robot Learning*, 2020. 138

[162] Luis Pineda, Taosha Fan, Maurizio Monge, Shobha Venkataraman, Paloma Sodhi, Ricky T. Q. Chen, Joseph Ortiz, Daniel DeTone, Austin S Wang, Stuart Anderson, Jing Dong, Brandon Amos, and Mustafa Mukadam. Theseus: A library for differentiable nonlinear optimization. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 143, 148

[163] Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable mpc for end-to-end planning and control. *Advances in neural information processing systems*, 31, 2018. 143

[164] Mohak Bhardwaj, Byron Boots, and Mustafa Mukadam. Differentiable gaussian process motion planning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10598–10604. IEEE, 2020. 143, 146

[165] Rajesh Rajamani. Lateral vehicle dynamics. In *Vehicle Dynamics and control*, pages 15–46. Springer, 2012. 143

[166] Yajia Zhang, Hongyi Sun, Jinyun Zhou, Jiangtao Hu, and Jinghao Miao. Optimal trajectory generation for autonomous vehicles under centripetal acceleration constraints for in-lane driving scenarios. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3619–3626. IEEE, 2019. 145

[167] Yajia Zhang, Hongyi Sun, Jinyun Zhou, Jiacheng Pan, Jiangtao Hu, and Jinghao Miao. Optimal vehicle path planning using quadratic optimization for baidu apollo open platform. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 978–984. IEEE, 2020. 145

[168] Nir Baram, Oron Anschel, Itai Caspi, and Shie Mannor. End-to-end differentiable adversarial imitation learning. In *International Conference on Machine Learning*, pages 390–399. PMLR, 2017. 147

[169] Takuma Seno and Michita Imai. d3rlpy: An offline deep reinforcement learning library. *Journal of Machine Learning Research*, 23(315):1–20, 2022. URL `http://jmlr.org/papers/v23/22-0017.html`. 151

[170] Chengyuan Zhang and Lijun Sun. Bayesian calibration of the intelligent driver model. *arXiv preprint arXiv:2210.03571*, 2022. 152