Designing Interaction-aware Prediction and Planning Models for Autonomous Driving

by

Yeping Hu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Spring 2021

Designing Interaction-aware Prediction and Planning Models for Autonomous Driving

Abstract

Designing Interaction-aware Prediction and Planning Models for Autonomous Driving

by

Yeping Hu

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

The ability to interact with other road participants is essential for autonomous vehicles in order to navigate under highly complex or critical driving scenarios. It is an extremely challenging task to enable autonomous vehicles interactively predict behaviors of others, and plan safe and high-quality motions for themselves. In fact, designing interaction-aware prediction and planning models is of great importance for autonomous vehicles to achieve full autonomy.

When designing a model, it is essential to consider our expectations for the model, which can be derived through two perspectives: 1) From the perspective of evaluation metrics (i.e. backward design); 2) From the perspective of task and user preferences (i.e. forward design). In this dissertation, a combination of forward and backward design process is proposed and it is argued that a model should be designed based on desired model properties inferred from metrics, tasks, and user preferences. A total of seven properties are covered in this dissertation: uncertainty, multi-modality, interpretability, flexibility, generalizability, reliability, and efficiency. The fundamental research question we addressed in this dissertation is: *how to design interaction-aware prediction and planning models under different autonomous driving settings in order to achieve desired model properties.*

This dissertation is divided into two parts. In Part I, we focus on the design of two-agent interaction models, which involves fundamental explorations of how certain model property can be achieved through algorithm design by taking the advantage of relatively low-complexity settings. Part II is concerned with the design of multi-agent interaction models, which includes comprehensive analysis and design for each desired model property under more complicated driving scenarios. In each part, we consider different prediction and planning related problems, and motivate our design by identifying desired model properties that correspond to each problem. By utilizing the proposed model design process under different problem settings, we demonstrate that our models are able to not only achieve desired properties, but also have great performances in terms of various evaluation criteria.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

The past five years of pursuing Ph.D. at Berkeley have been such a wonderful and unforgettable journey for me. I would not have come this far without the tremendous help from many people.

My deep and sincere gratitude goes to my Ph.D. advisor Professor Masayoshi Tomizuka, who has been a great mentor to me during the past five years for his profound knowledge, enthusiasm on work, and positiveness towards life. It is because of his recognition and encouragement that have supported me to conquer difficulties and leaded me to where I am right now. What I have learned from him will accompany and influence my entire academic life in the future.

I want to thank Professor Francesco Borrelli and Professor Pieter Abbeel to serve as my dissertation committee members. Meanwhile, I am thankful for Professor Kameshwar Poolla, Professor Mark Mueller, Professor Somayeh Sojoudi and Professor Jonathan Shewchuk for serving in my qualifying exam committee. Without the knowledge that I learned from them, I cannot finish the work in the dissertation.

Being a member in the Mechanical System Control (MSC) laboratory has been wonderful in the past five years. I want to specially thank Dr. Wei Zhan for guiding me into the field of autonomous driving, and providing me great support for my research projects. In addition, I sincerely thank my collaborators for all inspiring discussions: Dr. Wei Zhan, Dr. Liting Sun, and Jiachen Li. I also would like to express my thanks to the MSC lab members: Minghui Zheng, Shiying Zhou, Kevin Haninger, Changliu Liu, Yu Zhao, Xiaowen Yu, Te Tang, Hsien-Chung Lin, Chen-Yu Chan, Cheng Peng, Yongxiang Fan, Yu-Chu Huang, Daisuke Kaneishi, Shuyang Li, Liting Sun, Wei Zhan, Kiwoo Shin, Jianyu Chen, Zining Wang, Jiachen Li, Chen Tang, Zhuo Xu, Yujiao Cheng, Shiyu Jin, Hengbo Ma, Jessica Leu, Changhao Wang, Xinghao Zhu, Yiyang Zhou, Lingfeng Sun, Ge Zhang, Huidong Gao, Zheng Wu, Jinning Li, Catherine Faulkner, Wu-Te Yang, Xiang Zhang, Ting Xu, Chengfeng Xu, Chenran Li, Akio Kodaira, and Ran Tian, for their help during my graduate study and their efforts to maintain a good research atmosphere in the group.

Great thanks also go to all my sponsors through my Ph.D. life. I thank Berkeley Deep Drive and Momenta for their funding support. I would also want to thank Alireza Nakhaei at Honda Research Institute for his mentor and research collaborations for this dissertation.

Last but not least, my deepest love goes to my parents, for your unconditioned love, support and encouragement.

Chapter 1

Introduction

1.1 Model Design Process: An Overview

As the autonomous driving industry has grown rapidly over the past decade, tremendous amount of research and implementation efforts have been devoted to it in order to accelerate the realization of fully autonomous vehicles. Although many progresses have been made by leading companies or research institutes in this field, it is still an extremely challenging task for autonomous vehicles to navigate in dense urban scenarios which involve interactions with other road entities. Under such scenarios, autonomous vehicles need to accurately predict potential behaviors of other road participants, and plan safe and high-quality motions accordingly. In fact, such interactive prediction and planning problem is one of the major blocks to enable full autonomy. Therefore, in this dissertation, we focus on the design of interaction-aware prediction and planning models for autonomous vehicles, which process is illustrated in Fig. 1.1.

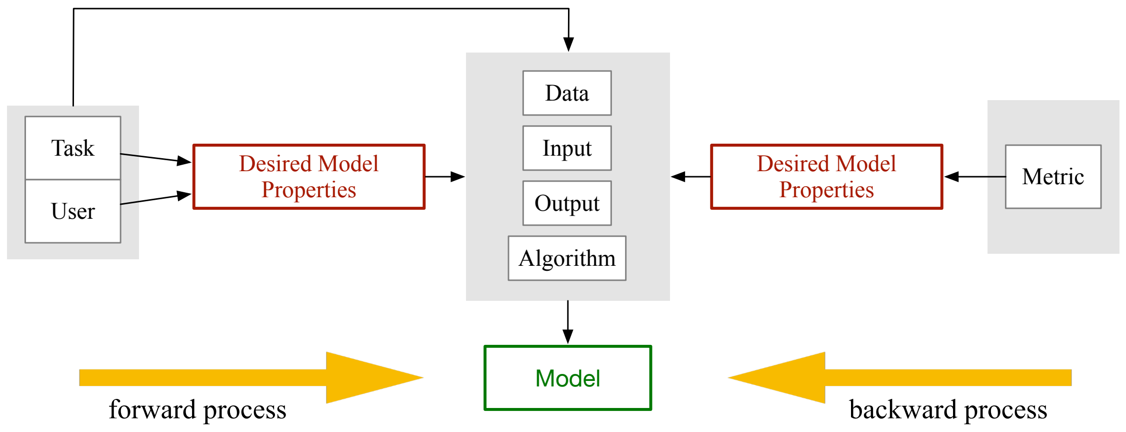


Figure 1.1: Model design process.

It is important to determine our expectations for the model before the design, which can be derived through two perspectives: 1) From the perspective of evaluation metrics (i.e. backward design). 2) From the perspective of task and user preferences (i.e. forward design). It is common to set the design goal as having state-of-the-art results evaluated using certain metrics, but the model could be deficient if only backward design process is considered. For example, when designing a prediction model, if we only concentrate on improving the prediction accuracy, the model might suffer from overfitting. Hence, we propose to take into account both forward and backward process for the model design.

According to Fig. 1.1, the forward design process begins with a task module and an user module. The task module describes *what* the model are designated to perform, such as "predict surrounding vehicles' trajectories" for a prediction model, or "merge into traffic" for a planning model; the user module describes user's preference on the model, for example passengers might prefer time-efficient driving strategies or gaining insight into why certain decision is made by the model. The backward design process, on the other hand, starts with a metric module which contains commonly used metrics for a given task. For instance, a trajectory prediction task usually utilizes mean squared error (MSE) to evaluate the prediction accuracy between the predicted output and the ground-truth. After acquiring information from the aforementioned three modules, we are able to infer desired properties for the model through both forward and backward process. Consequently, based on these properties and problem settings, we could determine the four building blocks of the final model, which include data, input, output, and the overall algorithm.

In this dissertation, we will cover seven model properties for prediction and planning tasks in autonomous driving, which are listed below with a brief introduction for each of them. Specifically, the first three properties are mainly related to prediction models, the last property is related to planning models, and the remaining properties are associated with both prediction and planning models. Moreover, interpretability and efficiency are properties that usually connect to user preferences, while other properties are either related to the task or the metric information. Note that some fundamental model properties such as accuracy for prediction models or feasibility for planning models are not listed below but are considered in the design.

- **Uncertainty**: As what will happen in the future will never be deterministic, prediction models are expected to consider uncertainties in predicted outputs.

- **Multi-modality**: The multi-modal property of a model can be regarded as having different motion patterns in the outputs. As was reviewed and summarized in [1], motion patterns can be categorized hierarchically into route, pass-yield and subtle patterns in various kinds of scenarios.

- **Interpretability**: Interpretability refers to an ability that can let people comprehend why certain decisions or predictions have been made by the model. This ability could help passengers to trust and mitigate fears of using autonomous vehicles. According to [2], interpretability can be viewed in two aspects. Firstly, models can convey some interpretability by themselves and such model are called as transparent models. Secondly, we can use some techniques to enable model interpretability. Such techniques are called post-hoc explainability techniques, which aim at communicating understandable information about how an already developed model produces its predictions for any given input.

- **Flexibility**: Flexibility refers to the model's ability to deal with a time-varying number of homogeneous or even heterogeneous agents in a scenario. Specifically, the model should handle different input sizes and be invariant to input ordering.
- **Generalizability**: Generalizability refers to the degree to which the model can be generalized or transferred to various situations. It is expected that a predictor or planner is generalizable across unseen driving scenarios or domains and such model property is extremely important for enabling autonomous vehicles to navigate in dynamically changing environment in real life.
- **Reliability**: Reliability refers to the quality of being trustworthy or of performing consistently well. In order for an autonomous vehicle to smoothly navigate in the environment, the predictor is expected to continuously produce reasonable results that are at least consistent with common-sense, and the planner should always generate collision-free path with safety guarantees.
- **Efficiency**: Efficiency, or time efficiency specifically, is an important factor in path planning. On one hand, a path planner is expected to find a safe solution path within a reasonable amount of time. On the other hand, passengers would prefer autonomous vehicles have efficient decision makers such that they can spend less time in traffic and arrive their destinations quickly.

1.2 Dissertation Contributions and Outline

The goal of this dissertation is to design interaction-aware prediction and planning models based on different problem settings in order to achieve desired model properties.

Figure 1.2 illustrates the outline of this dissertation. We divide this dissertation into two parts, *two-agent interaction* in Part I and *multi-agent interaction* in Part II. Note that in autonomous driving applications, agents referred to as road participants include vehicles, cyclists, and pedestrians. When traffic is sparse, interaction is most likely occur between two agents; when traffic is dense, each agent needs to simultaneously interact with several other agents, which results in multi-agent interaction. In each part, we consider different prediction and planning related problems, and motivate our design by identifying desired model properties correspond to each problem. Part I involves fundamental explorations of how certain model property can be achieved through algorithm design by taking the advantage of relatively low-complexity two-agent settings. Part II includes comprehensive analysis and designs for each desired model property under more complicated multi-agent settings.

1.2.1 Part I Two-agent Interaction

For two-agent interaction, two different situations need to be considered. The first situation is when both agents are observed by the autonomous vehicle, whose behaviors might have potential influence on the autonomous vehicle itself. In such case, predicting two agents' joint behaviors could help the autonomous vehicle make its own decisions better. The second situation is when one of the agents is autonomous vehicle and it is interacting with another agent on the road. In such case, the autonomous vehicle needs to predict the other agent's behavior conditioned on its planned path. In this part, we will consider both of the two-agent interaction settings and explore a total of five model properties when designing our models (see Fig. 1.2 for details).

Chapter 2

For prediction tasks, as the predicted horizon becomes longer, modeling prediction uncertainties and multi-modal distributions over future sequences turn into a challenging task. In this chapter, we address these challenge by presenting a multi-modal probabilistic prediction approach. The proposed method is based on a generative model and is capable of jointly predicting sequential motions of each pair of interacting agents. Most importantly, our model is interpretable, which can explain the underneath logic as well as obtain more reliability to use in real applications. A complicate real-world roundabout scenario is utilized to implement and examine the proposed method.

Chapter 3

When predicting behaviors of other agents, both rational and irrational behaviors exist, which need to be aware of by autonomous vehicles in their prediction module. Besides, the prediction module is also expected to generate reasonable results in the presence of unseen and corner scenarios. Two types of prediction models are typically used to solve the prediction problem: learning-based model and planning-based model. Learning-based model utilizes real driving data to model the human behaviors. Depending on the structure of the data, learning-based models can predict both rational and irrational behaviors. But the balance between them cannot be customized, which creates challenges in generalizing the prediction results. Planning-based model, on the other hand, usually assumes human as a rational agent, i.e., it anticipates only rational behavior of human drivers. In this chapter, a generic prediction architecture is proposed to address various rationalities in human behavior. We leverage the advantages from both learning-based and planning-based prediction models. The proposed approach is able to predict continuous trajectories that well-reflect possible future situations of other drivers. Moreover, the prediction performance remains stable under various unseen driving scenarios. A case study under a real-world roundabout scenario is provided to demonstrate the performance and capability of the proposed prediction architecture.

1.2.2 Part II Multi-agent Interaction

For multi-agent interaction, we are still able to use what we have learned from two-agent interaction settings for reference, but there are additional challenges we need to consider. Firstly, when multiple vehicles interact with each other, transitions of their states are highly coupled and interaction patterns between any two vehicles are different, which require more sophisticated prediction and planning models. Secondly, each driver has his/her own personality that leads to certain driving style. Hence, while interacting with multiple agents, the autonomous vehicle is expected to infer such unobservable information and react accordingly. Finally, the interaction patterns among agents also relate to environment information. For example, different topological structures or traffic regulations will influence the behavior of each agent and thus influence how a driving scene will be developed into the future. Therefore, it is necessary to find suitable representations of the environment and incorporate them into the model. In this part, we will address the aforementioned challenges under multi-agent settings and explore a total of seven model properties when designing prediction and planning models (see Fig. 1.2 for details).

Chapter 4

When predicting possible behaviors of traffic participants, the majority of current researches fix the number of driving intentions by considering only a specific scenario. However, distinct driving environments usually contain various possible driving maneuvers. Therefore, an intention prediction method that can adapt to different traffic scenarios is needed. To further improve the overall vehicle prediction performance, motion information is usually incorporated with classified intentions. As suggested in some literature, the methods that directly predict possible goal locations can achieve better performance for long-term motion prediction than other approaches due to their automatic incorporation of environment constraints. Moreover, by obtaining the temporal information of the predicted destinations, the optimal trajectories for predicted vehicles as well as the desirable path for ego autonomous vehicle could be easily generated. In this chapter, we propose a Semantic-based Intention and Motion Prediction (SIMP) method, which can be adapted to any driving scenarios by using semantic-defined vehicle behaviors. It utilizes a probabilistic framework based on deep neural network to estimate the intentions, final locations, and the corresponding time information for surrounding vehicles. An exemplar real-world scenario was used to implement and examine the proposed method.

Chapter 5

In this chapter, we present a probabilistic framework that is able to jointly predict continuous motions for multiple interacting road participants under any driving scenarios and is capable of forecasting the duration of each interaction, which can enhance the prediction performance and efficiency. The proposed traffic scene prediction framework contains two hierarchical modules: the upper module and the lower module. The upper module forecasts the intention of the predicted vehicle, while the lower module predicts motions for interacting scene entities. An exemplar real-world scenario is used to implement and examine the proposed framework.

Chapter 6

Since autonomous vehicles need to navigate in dynamically changing environments, they are expected to make accurate predictions regardless of where they are and what driving circumstances they encountered. A number of methodologies have been proposed to solve prediction problems under different traffic situations. However, these works either focus on one particular driving scenario (e.g. highway, intersection, or roundabout) or do not take sufficient environment information (e.g. road topology, traffic rules, and surrounding agents) into account. In fact, the limitation to certain scenario is mainly due to the lackness of generic representations of the environment. The insufficiency of environment information further limits the flexibility and transferability of the predictor. In this chapter, we propose a scenario-transferable and interaction-aware probabilistic prediction algorithm based on semantic graph reasoning. We first introduce generic representations for both static and dynamic elements in driving environments. Then these representations are utilized to describe semantic goals for selected agents and incorporate them into spatial-temporal structures. Finally, we reason internal relations among these structured semantic representations using learning-based method and obtain prediction results. The proposed algorithm is thoroughly examined under several complicated real-world driving scenarios to demonstrate its flexibility and transferability, where the predictor can be directly used under unforeseen driving circumstances with different static and dynamic information.

Chapter 7

In order to drive safely and efficiently under merging scenarios, autonomous vehicles should be aware of their surroundings and make decisions by interacting with other road participants. Moreover, different strategies should be made when the autonomous vehicle is interacting with drivers having different level of cooperativeness. Whether the vehicle is on the merge-lane or main-lane will also influence the driving maneuvers since drivers will behave differently when they have the right-of-way than otherwise. Many traditional methods have been proposed to solve decision making problems under merging scenarios. However, these works either are incapable of modeling complicated interactions or require implementing hand-designed rules which cannot properly handle the uncertainties in real-world scenarios.

In this chapter, we proposed an interaction-aware decision making with adaptive strategies (IDAS) approach that can let the autonomous vehicle negotiate the road with other drivers by leveraging their cooperativeness under merging scenarios. A single policy is learned under the multi-agent reinforcement learning (MARL) setting via the curriculum learning strategy, which enables the agent to automatically infer other drivers' various behaviors and make decisions strategically. A masking mechanism is also proposed to prevent the agent from exploring states that violate common sense of human judgment and increase the learning efficiency. An exemplar merging scenario was used to implement and examine the proposed method

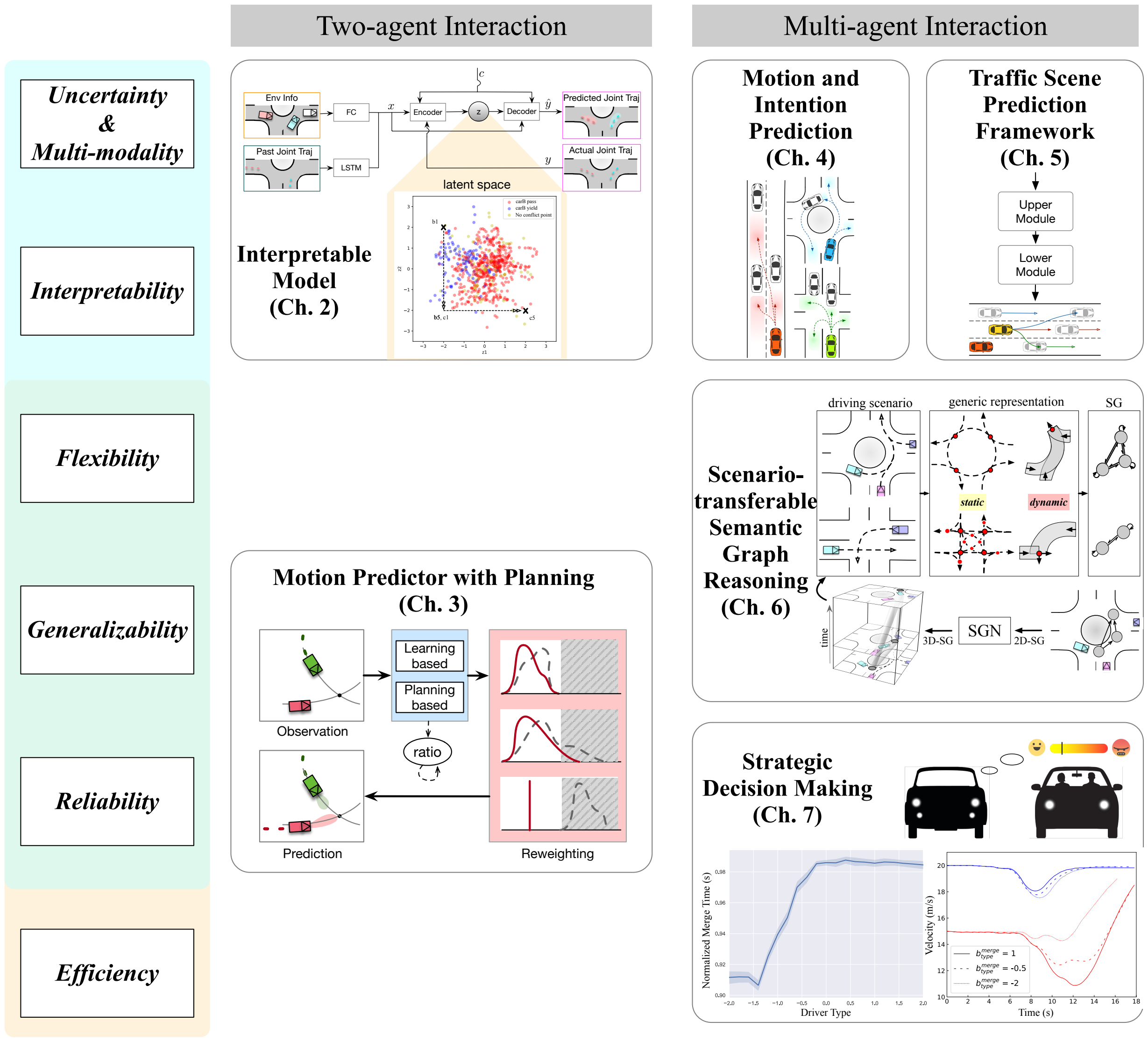


Figure 1.2: Dissertation outline.

Part I

Two-agent Interaction

Chapter 2

Multi-modal and Interpretable Predictor

2.1 Introduction

The idea of predicting the future behavior of statistical time series has a wide range of application in economics, weather forecast, intelligent agent systems, etc. The autonomous vehicle is one of the well-known intelligent agents and it is expected to predict behaviors of other road entities. Accurate and reasonable prediction is a prerequisite of performing reliable tasks involving motion planning, decision making, and control.

There have been numerous researchers working on behavior prediction problems and many approaches have been explored to solve such problem in the autonomous driving area. Some of these approaches only performed point estimate by assuming that the environment is deterministic. However, they failed to taking into account the uncertainty of future outcomes caused by partial observation or stochastic dynamics, which will induce the lost of information that capture the real physical interactions. Therefore, in this chapter, we take into account the uncertainty of drivers as well as the evolution of the traffic situations and try to predict possible behaviors of multiple traffic participants several steps into the future.

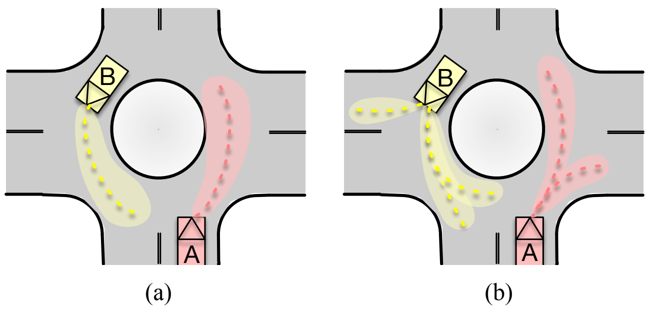


Figure 2.1: A demonstration of (a) single-modal and (b) multi-modal predicted distributions.

When dealing with uncertainties for sequential prediction problems, two aspects need to be further discussed: the estimation of *multi-modal* output distribution and the *interpretability* of the output samples. The multi-modal property of a method can be regarded as having different motion patterns in the outputs, which is illustrated in Fig. 2.1. As was reviewed and summarized in [1], motion patterns can be categorized hierarchically into route, pass-yield and subtle patterns in various kinds of scenarios. For routing information, motion patterns can be regarded as discrete. In contrast, predicted future trajectories are expected to have continuous motion patterns, where the agent's speed could increase, decrease or change randomly for a sequence of future motions.

The model interpretability is also a crucial aspect that needs to be considered. Since the output uncertainty is usually achieved by sampling data points from some learned distributions, it is necessary to reason about what causes the motion pattern to vary among samples. However, most of the commonly used approaches cannot provide much insight on the structure of the function that is being approximated, especially for learning-based methods.

The contributions of this chapter are in four folds: First, we proposed a multi-modal probabilistic prediction structure for autonomous vehicles using only a single learning-based model. Second, we considered the sequential motion prediction of each pair of interacting vehicles. Third, the proposed model is interpretable as we are able to explain each sampled data point and relate it to the underlying motion pattern. Last but not least, we trained and tested our method under a complicated roundabout scenario, which adds more difficulties to the behavior prediction problem.

The remainder of the chapter is organized as follows: Section 2.2 provides a brief overview of works related to interpretable models, trajectory prediction, and multi-modality. Section 2.3 provides the detailed explanation of the proposed approach; Section 2.4 discusses an exemplar scenario to apply our method; evaluations and results are provided in Section 2.5; and Section 2.6 concludes the chapter.

2.2 Related Works

2.2.1 Interpretable Models

To solve prediction problems for autonomous vehicles, many researchers utilized traditional methods such as constant velocity (CV), constant acceleration (CA), Intelligent Driver Model (IDM), and Kalman Filter (KF) [3]. However, these methods work well only under simple driving scenarios and their performances degrade for long-term prediction as they ignores surrounding context. As these models can easily fail when scaled to complicated traffic scenes, classical machine learning models are usually used instead, such as Hidden Markov Model (HMM) [4], Bayesian Networks (BN) [5], Gaussian Process (GP) [6], and Inverse Reinforcement Learning (IRL) [7][8]. The aforementioned methods either utilize some known transitions models or incorporate hand-designed domain knowledges to enhance the prediction performance. Although these works consist of interpretable methodology, their performances are usually worse than most learning-base methods that lack interpretations.

The success of deep learning in many real-life applications motivates research on its use for motion prediction and related methods include Mixture Density Networks (MDN) [9], Recurrent Neural Networks (RNN) [10], and Convolutional Neural Networks (CNN) [11]. Deep learning models can achieve high accuracy but at the expense of high abstraction which cannot be trusted.

Recently, numerous researchers tried to reason about learning-based methods by utilizing the idea of variational autoencoder (VAE) [12] which is a latent variable model that is able to learn a factored, low-dimensional representation of data. [13] developed a framework for incorporating structured graphical models in the encoders of VAE that allows them to induce interpretable representations through approximate variational inference. [14] proposed a novel factorized hierarchical VAE to learn disentangled and interpretable latent representations from sequential data. Our goal is to develop an interpretable architecture for behavior prediction based on the latent variable model such that features involved in modeling can be described through latent codes and explainable future motions can be generated.

2.2.2 Trajectory Prediction

There are variety of works dealing with trajectory prediction problems for road entities such as vehicles and pedestrians. [15] proposed a Long Short-Term Memory (LSTM) encoder-decoder model to predict a multi-modal predictive distribution over future trajectories based on maneuver classes. [16] applied the Hidden Markov Model (HMM) to predict the trajectories for individual driver. [17] combined CNN and LSTM to predict multi-modal trajectories for an agent on a bird-view image. The main limitation of these works, however, is that they only predict motions for one selected agent without considering the influence of other agents with potential interactions.

Since the motion of an agent can be largely influenced by other surrounding agents, some researchers began to tackle the scene prediction problem. Modeling future distributions over the entire traffic scene is a challenging task, given the high dimensional feature space and complex dynamics of the environment. [18] and [19] brought forward a LSTM-based structure to predict the most possible K trajectory candidates for every vehicles over occupancy grid map. [20] utilized the Dynamic Bayesian network (DBN) for behavior and trajectory prediction. In [21], the authors proposed a hierarchical scene prediction framework, where the Conditional Variational Autoencoder (CVAE) was used in their lower module to predict continuous motions for multiple interacting road participants.

2.2.3 Multi-modality

There exists a number of studies addressing the problem of modeling multi-modality. Feedforward network with Gaussian Mixture [9][10] is usually applied to solve multi-modal regression tasks but it is often difficult to train in practice due to numerical instabilities when operating in high-dimensional spaces such as predicting future sequences.

Other works solved this problem by utilizing different regression models for different possible discrete intentions of road entities [6][21]. However, when the intention space is large or data is insufficient, such method becomes inefficient and the model will even suffer from over-fitting problems. Alternatively, [11][22] treated the discrete intention as one of the state input to the proposed structure to generate different types of output.

2.3 Approach

In this section, we first introduce the main algorithm of the proposed behavior prediction approach. Then the details of the intention prediction method are illustrated.

2.3.1 Interactive Behavior Prediction

Our proposed method is based on the structure of CVAE which has a similar encoder-decoder structure as the typical VAE. Two types of conditional input are considered in our model structure: historical scene information and inferred driving intention.

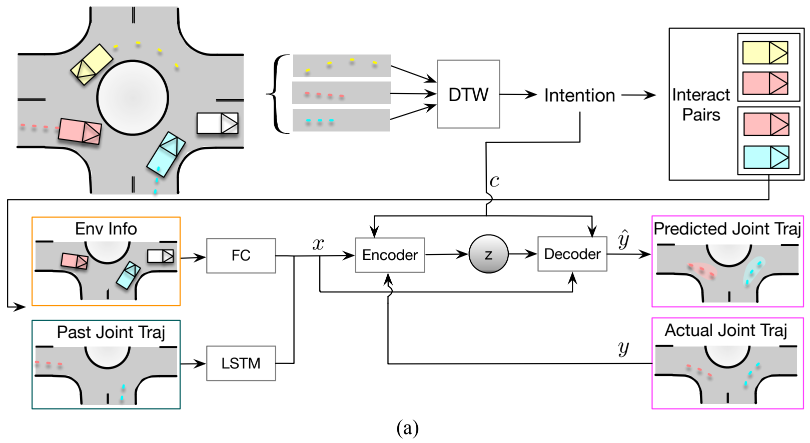
We focus on predicting human drivers' interactive behaviors between two vehicles: vehicle A, (denoted by $(\cdot)^A$), and vehicle B, (denoted by $(\cdot)^B$). Both vehicles are regarded as the predicted vehicle and we are interested in jointly predicting their behaviors, while taking into account any internal correlations. Note that it is trivial to convert the output joint distribution to a conditional distribution by treating one of the predicted vehicles as the ego vehicle and obtain the behavior prediction of the other. However, we will not address further on such problem setting in this chapter.

For a given vehicle, we use ξ to represent its actual trajectory and $\hat{\xi}$ as the trajectory we predict. At the current time step t, we denote the vehicle's historical trajectory as $\xi_{(t-T_1):t}$ and its future trajectory as $\xi_{t:(t+T_2)}$, where T_1 and T_2 represent the number of time steps into the past and future, respectively. Moreover, we denote $\mathcal{I}$ as the discrete intention of a vehicle and $\mathcal{E}$ as the environment information that contains states of surrounding vehicles.

Given the historical trajectory and driving intention of two interactive vehicles, along with the environment information, the objective of estimating probabilistic joint trajectories can be expressed as:

$$P(\boldsymbol{\xi}^{\{A,B\}}_{t:(t+T_2)} \mid \boldsymbol{\xi}^{\{A,B\}}_{(t-T_1):t}, \boldsymbol{\mathcal{I}}^{\{A,B\}}, \mathcal{E}). \tag{2.1}$$

To formulate the problem in the CVAE structure, we let the encoder, q_θ, take the input x as a learned embedded space of historical trajectory and environment information, c as the intention vector $\boldsymbol{\mathcal{I}}^{\{A,B\}}$, and y as the actual trajectory $\boldsymbol{\xi}^{\{A,B\}}_{t:(t+T_2)}$, to "encode" them into a latent z-space. Then the decoder, p_ϕ, takes x and c as input, to "decode" the sampled values from z-space back to the output $\hat{y}$, which corresponds to the predicted future trajectories $\hat{\boldsymbol{\xi}}^{\{A,B\}}_{t:(t+T_2)}$. To enable backpropagation, the network is trained using the reparameterization



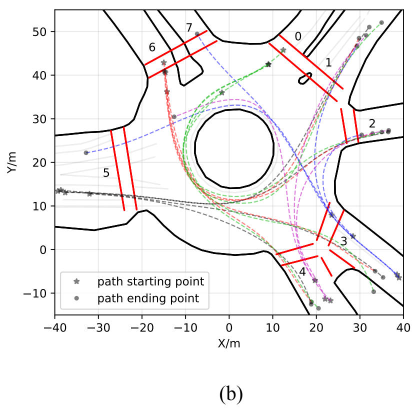


Figure 2.2: (a) The overall structure of the proposed method. (b) Roundabout map and all reference path.

trick [12] such that the latent variables can be expressed as:

$$z = \mu_\theta(x, c, y) + \sigma_\theta(x, c, y) \times \epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \tag{2.2}$$

Here, θ and ϕ are the parameters of the encoder and decoder network, respectively. To learn these parameters, we can optimize the variational lower bound:

$$\mathcal{L} = -\mathbb{E}_{q_\theta(z|x,c,y)}\big[\log p_\phi(y|x, c, z)\big] + \beta D_{KL}(q_\theta(z|x, c, y)||p(z)), \tag{2.3}$$

where the model is forced to perform a trade-off between a good estimation of data log-likelihood and the KL divergence of the approximated posterior q_θ from prior $p(z)$ which, in our case, is assumed as unit Gaussian $\mathcal{N}(0, I)$. We also utilize the hyperparameter β to control the training balance between the two losses for better performance.

At test time, we can directly sample from $\mathcal{N}(0, I)$ as the latent variable input and only use the decoder to obtain the predicted joint distribution. Notice that among the three input of the decoder network, only x is fixed at a given time step while both c and z are non-deterministic factors. Therefore, in the following section, we will analyze the effects of these factors to the output trajectories, demonstrate how the proposed model can estimate multi-modal distributions over future sequences, and explore the interpretability underneath the model.

2.3.2 Intention Prediction

Bayesian Approach

During the scene evolution, we use a Bayesian approach to predict each vehicle's intention i, based on history observation h. In this problem, we consider the vehicle's past trajectory as the observation h since an agent's potential intent can largely influence its trajectory. Therefore, the Bayesian equation can be written as:

$$f(i|h) = \frac{f(h|i)f(i)}{\sum_i f(h|i)f(i)}, \tag{2.4}$$

where $f(\cdot)$ represents the probability density function. The term $f(i)$ is the prior belief of the intention and is initialized with a known distribution according to initial observation; $f(h|i)$ is the likelihood of observing h for a given intention i; and the denominator is a normalization term.

Dynamic Time Wraping (DTW)

The dynamic time warping (DTW) distance as proposed in [23] is a trajectory measure that can be used on general time series. DTW does not require both trajectories to have the same length. Instead, DTW measures the temporal changes that are necessary in order to warp one trajectory into another.

If we consider driving intention as pursuing some goal location such as one of the exit branch in roundabout scenario or left/right turn in intersections, we are able to obtain a reference driving path for each intention. Therefore, we can use the DTW algorithm to determine the likelihood of an observed trajectory h given a reference path r_i assuming the agent has intention i:

$$f(h|i) = \frac{e^{-D(r_i,h)}}{\sum_i e^{-D(r_i,h)}}, \tag{2.5}$$

where $D(r_i, h)$ is the cost calculated by the DTW algorithm. The smaller the cost is, the closer the observed trajectory is to the reference path, and thus the higher the probability is for intention i. In fact, we are interested in the DTW value between the observed trajectory and a segment of the reference path closer to the trajectory instead of the full reference path.

Select Interacting Pairs

After obtaining the intention probabilities of each vehicle in the scene at a given time step, we can determine whether any of the two vehicles have potential interaction according to their corresponding reference path. If all potential reference path of two vehicles have no crossing point, the vehicles' future trajectories will be independent from each other and thus no attempt is needed to further predict their joint motions. In this work, we make an assumption that the interaction happens only between two agents but there can be multiple interacting pairs in the scene concurrently.

Avoid Constantly-Changed Prediction Results

To avoid rapid fluctuation of the likelihood distribution, we perform the aforementioned intention prediction algorithm for at least every other 0.4s. Here we assume that a certain driving intention will not change within a short period of time especially for the roundabout scenario where the driver already know his/her intended road branch to exit.

2.4 Experiments

In this section, we use an exemplar roundabout scenario to apply our proposed behavior prediction method. The data source and details of the problem formulation are presented.

2.4.1 Real Driving Scenario

Dataset

The driving data we used was collected by our Mechanical Systems Control Lab at a single-lane roundabout in Berkeley, CA. The roundabout, shown in Fig. 2.2(b), is connected with 8 branches and each of them has one entry lane and one exit lane.

The data was collected by a drone from bird's eye view. To smoothen the noisy data, we performed a downsampling to decrease the sampling rate from 10Hz to 5Hz and applied an Extended Kalman Filter (EKF). We manually picked 1534 driving segments from the collected data, where 80% were randomly selected for training and 20% for testing.

Reference Path

According to the dataset, a total of 19 reference paths were considered, which can be seen in Fig. 2.2(b). Each reference path corresponds to a routing information from one lane to another and is generated by finding the best fitted path among all vehicle trajectories in the data that have the same entry and exit lane.

2.4.2 Problem Description and Feature Selection

Problem Description

For roundabout scenario, one of the most typical interaction scenarios happens when one vehicle (car A) is waiting behind the stop/yield sign and trying to enter the circular roadway, while another vehicle (car B) is driving on the circular roadway towards the direction of car A. Under such circumstance, the potential exit lane for car B will largely influence the driving behavior of both vehicles. For example, if car B exits the circular roadway before reaching the current lane of car A, the driving trajectories of two vehicles will not be influenced by each other; contrarily, if car B keep driving on the circular roadway, two cars will begin negotiating with each other to decide who should go first, which will affect their future trajectories.

As most of the selected cases in our dataset belong to the aforementioned situation, we only consider the driving intent of car B as the intention input $\mathcal{I}$ in our proposed predic-

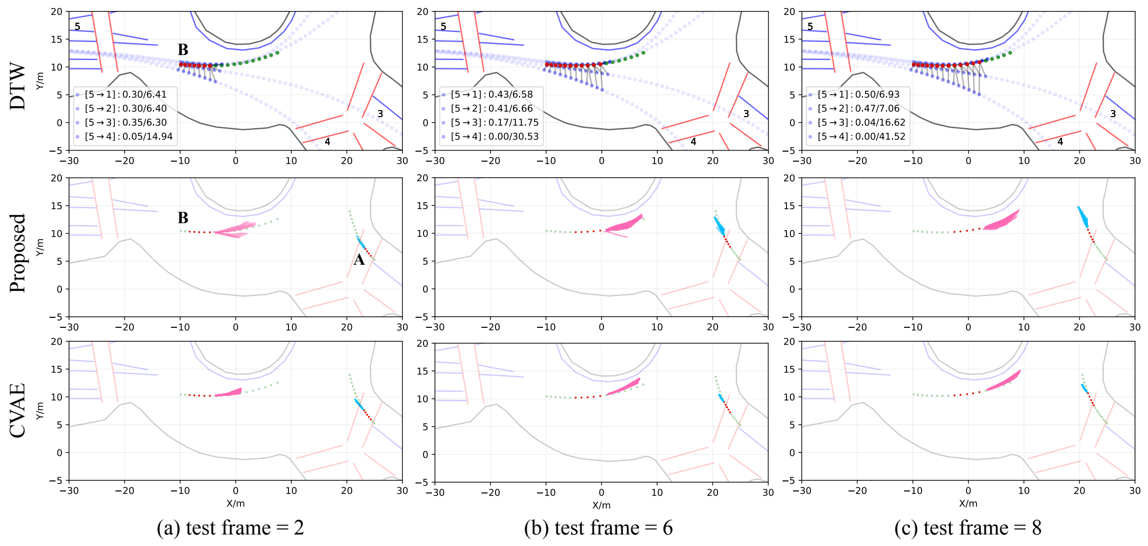


Figure 2.3: Visualization of the prediction results under a selected test scenario. Each dotted point represents the position of the vehicle. The green dots represent the ground truth trajectory of the vehicle, where the time step between each two dots is 0.2s; the red dots represent the input historical trajectory. Ten future trajectories for car A and car B are sampled from the trained model and are shown in cyan and pink color, respectively. The legend in the plots of the first row should be interpreted as: *[entry lane → exit lane]: posterior intention probability / current DTW value.*

tion model. Moreover, the front vehicles of car A and car B are regarded as environment information in each driving case, which are essential influence factors of vehicle behaviors.

Feature Selection

The input of past joint trajectories contains four features at each time step: $\boldsymbol{\xi}^{A,B}_{(t-T1):t} = [\boldsymbol{x}^A_{(t-T1):t}, \boldsymbol{y}^A_{(t-T1):t}, \boldsymbol{x}^B_{(t-T1):t}, \boldsymbol{y}^B_{(t-T1):t}]$. The environment input contains the surrounding vehicles' information at the current time step t, which is the state of each interactive car's front vehicle: $\mathcal{E} = [\boldsymbol{x}^{front}_t, \boldsymbol{y}^{front}_t, \boldsymbol{v}^{front}_t]$. Here, x and y represent the vehicle's location in Euclidean coordinates, while v denotes the vehicle speed. The driving intention is converted to an one-hot vector, which denotes the intended exit branch for the selected vehicle out of all 8 possible branches.

2.4.3 Implementation Details

As shown in Fig. 2.2, the environment information passes through a fully connected network with 16 neurons and the sequence of past joint trajectories are fed into one LSTM cell with 16 neurons. Both the encoder and decoder contain three fully connected layers of

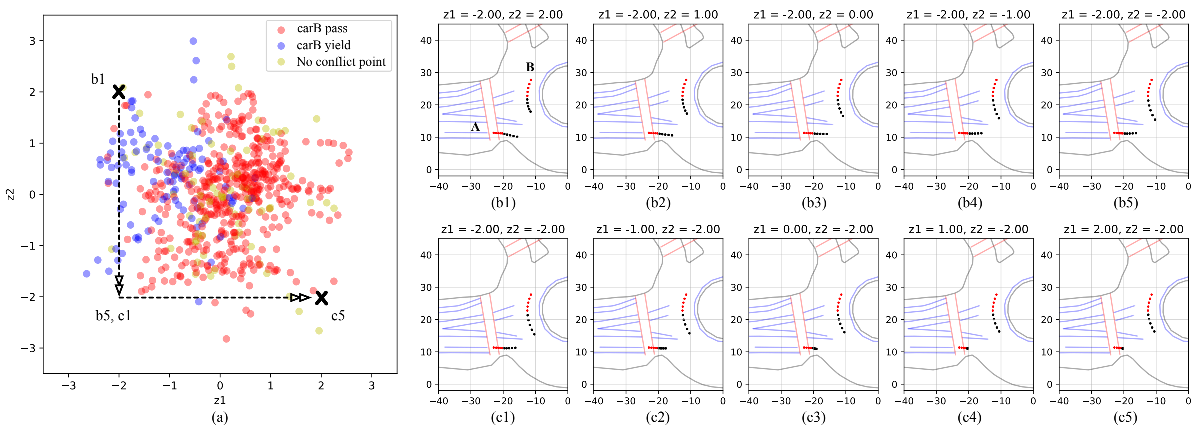


Figure 2.4: Latent space visualization and demonstration of the interpretable model. In (b1-b5) and (c1-c5), red points represent historical trajectories and black points represent predicted future trajectories for both vehicles.

64 neurons with *tanh* as non-linear activation function. The latent space dimension is set to 2 and a randomly sampled z-vector from a unit Gaussian is used as one of the input of the decoder. In this problem, $T1$ and $T2$ are both set to 5, where we want to predict 1s into the future using the past 1s information. According to the experiment, the method has the best performance when β is set to 0.005.

2.5 Results and Evaluations

In this section, we first visually illustrate the result of the proposed model through several selected cases. Then we introduce the quantitative evaluation metric and present the comparison result with other baseline methods.

2.5.1 Visual Illustration

Intention Prediction

We selected a representative case to demonstrate how intention is predicted using Bayesian update and DTW, shown in the first row of Fig. 2.3. The top four reference paths that have the highest intention probability were plotted in light blue and we further selected a path segment (dark blue) from each reference path to calculate the DTW value with the observed historical trajectory (red).

At the beginning, the vehicle's intention is ambiguous and it has similar probability of exiting at branch 1, 2, and 3. However, the reference path of exiting at the 4th branch has the largest DTW value compared to the vehicle's observed trajectory and thus its corresponding intention probability is the smallest among the listed four reference path. As the vehicle continues to move, it drives closer to the roundabout center and has lesser intent to exit. Such behavior is well captured by the intention prediction module as shown in the first and second column of the first row in Fig 2.3, where the probabilities of the vehicle exiting the 1st and 2nd branch increase while the likelihood of exiting at the 3rd branch decreases.

Trajectory Prediction

We tested and compared the prediction results of our proposed method with the original CVAE approach that does not contain intention inference module. The testing results on a selected scene are shown in the bottom two rows of Fig. 2.3. To make a fair comparison, we fixed the 10 randomly sampled latent z-values in both methods and used them to generate 10 future joint trajectories of two interacting vehicles.

According to the result, our proposed method successfully generates different motion patterns which are consistent with the intention prediction result. In contrast, the traditional CVAE method only predicts one motion pattern and it fails to consider the possibility that car B might exit the roundabout at the 3rd branch.

We argue that although two vehicles have interaction and it may influence their future trajectories, the intention of which road to exit for car B is not influenced by the other vehicle, A. Therefore, if we are about to predict the joint trajectories of two cars using learning-based methods like CVAE, each vehicle's trajectory will be largely influenced by the data distribution and will not reveal multi-modal property if the amount of data we used are not large enough to cover every possible cases. Even if we have sufficient data, the CVAE network will most likely learn how to closely relate the future motions of two vehicles instead of learning to infer the future joint trajectories based on the historical trajectory of each individual vehicle. In other word, we don't want the network to only learn the motion correlations between two vehicles without treating each of them individually. Hence, the intention factor we added in the proposed method will mitigate such problem by encouraging the network to relate each vehicle's intention to its own past trajectory and then generating its future motions while taking other vehicle's historical motions into consideration.

Interpretability

The learned latent space is plotted in Fig. 2.4(a) where we assigned different colors to different interact cases. Although the pass/yield information of two interacting vehicles is not used during training, our proposed method successfully distinguished such motion patterns in the latent space. To illustrate the influence of the sampled z-vector to the predicted trajectories, we fixed the intention input c and only changed the z value along its two dimensions.

As we fix $z1$ and decrease $z2$ (figure b1 to b5), car B increases its speed and shifts from yielding car A to passing car A while the speed of A does not change much. As we fix $z2$ and increase $z1$ (figure c1 to c5), car B always passes car A but the speed of car A gradually decreases. Therefore we can conclude that if we move z from the 2nd to the 4th quadrant of the 2D unit Gaussian, there will be an obvious change of interaction patterns between two cars. Hence, the proposed method is interpretable as the sampled output can be well-explained by the location of z. Moreover, being able to generate various motion patterns from different sampled z values can be also regarded as having the multi-modal property.

2.5.2 Metric

Mean Squared Error (MSE)

MSE is commonly used to evaluate the prediction performance and the equation can be written as:

$$MSE = \frac{1}{N_s} \sum_{s=1}^{N_s} (y - \hat{y}^s)^2, \tag{2.6}$$

where $\hat{y}^s$ is the s-th sampled prediction result out of N_s output samples and y is the ground-truth.

MSE is skewed in favor of models that average different output modes. In particular, this average may not represent a good prediction when more than one mode exists.

Negative Log Likelihood (NLL)

While MSE provides a tangible measure for the predictive accuracy of models, it has limitations while evaluating multi-modal predictions. NLL, instead, is a proper metric for evaluating predictive uncertainty [24] and it can be calculated as:

$$NLL = \frac{\log \sigma^2(\hat{\boldsymbol{y}})}{2} + \frac{(y - \mu(\hat{\boldsymbol{y}}))^2}{2\sigma^2(\hat{\boldsymbol{y}}))}, \tag{2.7}$$

where $\mu(\hat{\boldsymbol{y}})$ and $\sigma^2(\hat{\boldsymbol{y}})$ represent the mean and variance of N_s output samples respectively.

2.5.3 Quantitative Performance Evaluation

We compared our method with the following three approaches, where all of them take historical trajectories as input and generate a sequence of future trajectories as output.

- *Conditional Variational Autoencoder (CVAE w/o $\mathcal{I}$)*: We compared the proposed method with the traditional CVAE approach without using intention prediction module.

- *Multi-layer Perceptron Ensemble (MLP-ensemble)*: The MLP is designed to have the same network structure as the decoder in our proposed model. To incorporate uncertainty, we applied the *bagging* strategy to combine predictions of models built on subsets created by bootstrapping.

- *Monte Carlo dropout (MC-dropout)*: We also implemented the MC-dropout method [25] to estimate the prediction uncertainty by using *Dropout* during training and test time. The mean and variance can be obtained by performing stochastic forward passes and averaging over the outputs.

The MSE and NLL values are calculated for all four methods and the results are shown in Table 2.1. It is apparent from the table that our method has the largest value in terms of the MSE but has the smallest NLL. Such results indicate that the proposed method is able to generate output trajectories with the largest uncertainties due to its multi-modal prediction results at the expense of slightly larger MSE value. The reason that other methods have smaller MSE value is because they can only approximate single motion model and all the corresponding output samples are distributed around the groudtruth. However, small MSE is not favorable when the output is supposed to have multiple motion models.

Moreover, we notice that CVAE has comparable results against ML dropout and MLP ensemble in terms of the two evaluation metrics. Therefore, the proposed method, based on the CVAE methodology, is not only able to generate desirable performances, but also capable of estimating explainable multi-modal distributions.

Table 2.1: Performance Comparison

Method	Proposed	CVAE Without $\mathcal{I}$	MC Dropout	MLP Ensemble
MSE	0.45 ± 0.11	0.16 ± 0.14	**0.15 ± 0.10**	0.20 ± 0.11
NLL	**0.83 ± 0.26**	2.57 ± 1.02	2.10 ± 0.63	3.05 ± 0.97

2.6 Conclusions

In this chapter, a multi-modal probabilistic prediction method is proposed, which can predict interactive behavior for traffic participants and acquires interpretability. An exemplar roundabout scenarios with real-world data collected by ourselves was used to demonstrate the performance of our method. First, the prediction results for intention and motion of selected vehicles are visually illustrated through a representative driving case. Then, we plotted the learned latent space to demonstrate the interpretability. Finally, we quantitatively compared the proposed method with three different models: CVAE, MLP ensemble and MC dropout. Our method outperforms these methods in terms of the negative log likelihood metric, which shows its advantages for learning conditional models on multi-modal distributions. In future works, we will focus on making the prediction algorithm not only interpretable but also safe to use under various circumstances.

Chapter 3

Rational and Irrational Motion Predictor with Planning

3.1 Introduction

3.1.1 Motivation

While interacting with human drivers, autonomous vehicles need to be aware of the multi-modal interaction outcomes in order to have reasonable prediction results. Such multi-modality comes from the fact that human can have different levels of rationality. Rational behaviors usually result in safe and feasible driving motions, while irrational behaviors are typically reflected by dangerous and unusual driving motions that may lead to car accidents.

Even though most people would consider themselves to be logical and can make rational decisions, drivers are not necessarily absolute rational. They do not always drive in optimal and safe trajectories since sometimes they are willing to take risks for their own benefits. For example, a driver may perform a dangerous cut-in maneuver to change into a desired lane quickly; a novice driver might be overly cautious and tends to make improper braking. Besides, drivers can easily have irrational driving behaviors due to wrong predictions of other road entities or unawareness of the surroundings. Therefore, autonomous cars are expected to consider not only rational but also irrational behaviors of other drivers during the prediction process, which will assure preparation for potential emergencies as well as safe and comfortable driving experiences.

3.1.2 Related Works

Learning-based Approach

Learning-based methods have been wildly used for prediction problems for autonomous vehicles, which utilize real data to produce the future outcomes of human drivers. In [26], the authors modeled the driver behavior by hidden Markov models (HMM) and Gaussian

Process (GP) to generate a group of future trajectories of the predicted vehicle. The long short-term memory (LSTM) method is utilized in [15] and [19] to analyze past trajectory data and predict the future locations of the surrounding vehicles. [21] proposed to combine a modified mixture density network (MDN) [9] and a conditional variational autoencoder (CVAE) to predict both discrete intention and continuous motions for multiple interacting vehicles. Based on generative models, an interpretable multi-modal prediction method is proposed in [27] , which can predict interactive behavior for traffic participants.

The main advantage of learning-based methods is that complicated models can be learned to represent real-life situations for prediction. However, two drawbacks need to be concerned for the learning-based method. First is the data insufficiency. Although researchers have tried to use more training data to learn driving models, it is nearly impossible to have a dataset that is large enough to cover every possible driving situations and thus the learned model can easily fail under any unseen or corner cases. Second is the inherent biases of the collected data. In fact, if the training data has some inherent biases, the driving model will not only learn those biases but will end up amplifying them. For example, if under a certain driving scenario, the data contain mostly irrational behaviors, the learned model will be inclined to predict more irrational than rational behaviors.

Planning-based Approach

Planning-based approaches [28, 29, 30, 31, 32, 33] assume that human drivers are approximately optimal planners with respect to some reward functions, i.e., their future trajectories are maximizing their rewards. Hence, the prediction of their trajectories can be obtained by solving optimization problems with the correct reward functions.

To acquire such reward functions, inverse reinforcement learning (IRL) has been widely adopted. It aims at finding a reward function which can match best in terms of key features with the observed demonstrations. Initially proposed by Kalman [34], [35] and [36] formulated it as apprenticeship learning by maximizing the margin in terms of feature matching. Later, Ziebart *et al.* [37] further extended IRL to deal with the probabilistic characteristic of reward functions via the maximum entropy principle. Levine *et al.* [38] proposed to solve the maximum-entropy IRL directly in continuous domain and applied it to predict human driving behaviors. Based on these works, many IRL based prediction or human behavior modeling approaches have been proposed. For instance, [39] learned a model for cooperative agents to generate human navigation behavior, and in [8], the authors used IRL to model the social impact between interactive agents. In [7], the authors formulated a hierarchical IRL to model human driver's decision making and trajectory planning, so that interactive driving behaviors can be predicted.

One main advantage of the aforementioned planning-based methods is their inherent implication of causality. Hence, they can easily guarantee the feasibility of the predicted trajectories and have better generalization ability to unseen circumstances. For instance, [40] used planning-based methods to infer about uncertainties for better prediction. However, the planning-based approach also suffer from several issues. First, most of the planning-based approaches assume that all drivers are approximately rational, i.e., they are optimizing some reward/cost function while driving. This assumption cannot be strictly hold in practice since “irrational” behaviors are inevitable. Consequently, the predicted trajectories might be too conservative to reflect potential dangers in some situations. Although recent works such as [41] and [42] have introduced some “irrational” behaviors into planning-based approaches, it is still not sufficient to cover all human driving behaviors. Moreover, to obtain a solvable and interpretable planning problem, the learned reward/cost functions are typically linear combinations of features. Such representation might not be complicated enough to capture different driving behavior.

3.1.3 Contribution

As planning is a lower module of prediction, it is common to solve planning problems by utilizing prediction results to incorporate uncertainties. However, very few works explicitly apply the planning approach in the prediction problem. In this work, we propose to leverage the advantages of both the planning-based and learning-based methods to mutually compensate for their drawbacks, which can enhance the overall prediction performance. The main contributions of this work are as follows:

- We propose a generic environmental representation methodology for vehicle behavior predictions under highly-interactive scenarios.
- Both the irrational and rational future trajectories of human drivers can be predicted, which can provide better environment information to the autonomous vehicle.
- The performance of the proposed architecture remains stable and can be safely used under rare events and corner cases.
- A challenging real-world roundabout scenario is considered in this work to demonstrate the capability of our method.

3.2 Problem Statement

3.2.1 A mathematical representation

In this problem, we consider the interactive behavior between two vehicles: the ego autonomous vehicle (denoted by $(\cdot)^{ego}$) and the predicted human-driven vehicle (denoted by $(\cdot)^{pred}$). Other vehicles in the scene will be regarded as surrounding vehicles which is denoted by $(\cdot)^{surr}$. For a given vehicle, we use ξ to represent historical trajectories, $\hat{\xi}$ for future trajectories, and $\hat{\xi}_{gt}$ for ground truth future trajectories. We store all historical trajectories of the related vehicles in ξ, where $\xi = [\xi^{pred}, \xi^{ego}, \xi^{surr}]$. Note that a trajectory is represented by a sequence of states of the vehicle, i.e. $\xi = [x_1, x_2, \ldots, x_T]$ where T is the trajectory horizon and x_t denotes the vehicle state at t-th time step.

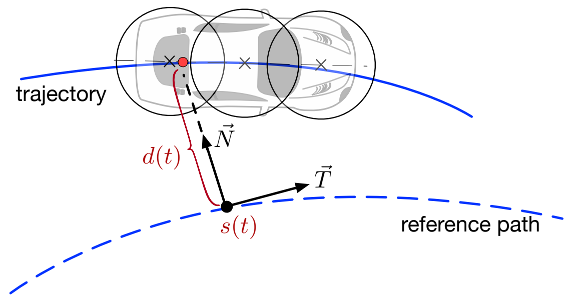


Figure 3.1: Illustration of the Frenét frame and vehicle boundaries. The red point represents the vehicle's center of mass. The shape of the vehicle is approximated by three circles.

We aim at predicting the behavior of the selected vehicle while considering the potential influence of its future behavior from our own vehicle. We use the following conditional probability density function (PDF) to represent the correlated future trajectories of two interactive vehicles:

$$P(\hat{\xi}^{pred}, \hat{\xi}^{ego} | \xi), \tag{3.1}$$

where the possible joint trajectories of the two vehicles depend on the historical trajectories of their own as well as all surrounding vehicles in the scene. However, after the ego vehicle planned a possible future trajectory for its own, it is necessary to marginalize out $\hat{\xi}^{ego}$ to obtain the future trajectories of the predicted vehicle conditioned on the ego vehicle's planned trajectory. Mathematically, such dependencies can be expressed as:

$$P(\hat{\xi}^{pred} | \hat{\xi}^{ego}, \xi). \tag{3.2}$$

In general, we are trying to infer possible future behaviors of the predicted vehicle given several potential future trajectories planned for the ego vehicle.

3.2.2 Generic Environmental Representation

Representation in Frenét Frame

Instead of Cartesian coordinate, we utilized the Frenét Frame to represent vehicle state. As illustrated in Fig. 3.1, the vehicle motion in the Frenét Frame can be represented with the longitudinal position along the path $s(t)$, and lateral deviation to the path $d(t)$. Therefore the vehicle state at time step t can be defined as $x_t = (s(t), d(t))$. Note that the reference path of a vehicle will change according to the road it is current driving on. The origin of the reference path can be defined differently according to different objectives and each reference path will have its own Frenét Frame. Since we are dealing with interaction between two vehicles, we define the origin as the cross point of their reference path. Note that each vehicle can have several possible reference paths, thus there could be multiple combinations of reference path pair corresponding to different cross points.

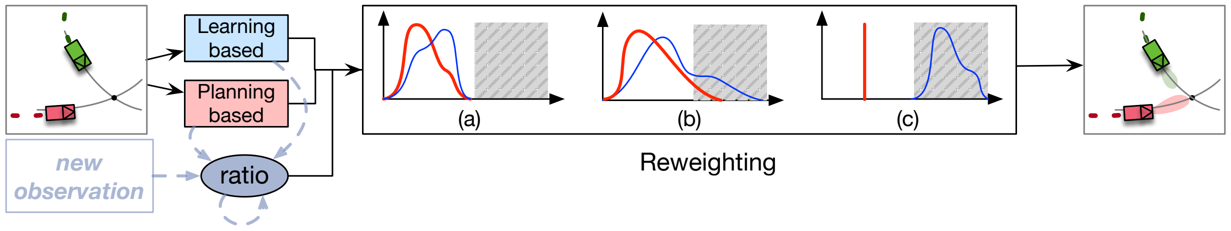


Figure 3.2: This plot shows the process of proposed architecture. Note that trajectories directly sampled from the learning-based method can have three different degrees of rationality: (a) fully rational; (b) partially rational; (c) fully irrational. Here, we demonstrate intuitive illustration of the reweighting process in each of these cases, where the original probability density function (PDF) of all sampled trajectories is shown in blue, the PDF after the reweighting process is shown in red, and the shaded area contains infeasible sample points.

Conversion between Cartesian coordinate and Frenét Frame

For both ego and predicted vehicle, the following steps need to be performed throughout the testing process of the proposed architecture:

(i) Determine the possibility of every reference trajectory according to historical path using dynamic time wrapping (DTW) [23][27].

(ii) Map the current global position (in Cartesian coordinate) of the vehicle on to each possible reference path (in Frenét Frame).

(iii) Perform the proposed prediction algorithm under the Frenét Frame.

(iv) Convert the predicted results back to Cartesian coordinate to check for collision and visualize the result.

(v) Receive a new observation in Cartesian coordinate and repeat step (i)-(iv).

3.3 Prediction Architecture

In this section, we first introduce the detailed formulation and structure of the overall prediction framework. We then overview the approaches used for the learning-based and planning-based model. The graph illustration of the proposed architecture as well as the pseudo-code of the algorithm are shown in Fig. 3.2 and Algorithm 1 respectively.

3.3.1 Overall Framework

The proposed generic prediction framework contains the following six steps:

Sample Joint Trajectories

Given the observed historical trajectories of the two interacting vehicles and their surrounding vehicles, the future joint trajectories can be sampled from the predicted joint distribution generated by a learning-based method. The N sampled trajectory pairs can be expressed as $\{\hat{\xi}^{pred}, \hat{\xi}^{ego}\}_{1:N}$.

Convert to Conditional Distribution

Since we are interested in predicting the possible future trajectories of the predicted vehicle given the most likely future trajectory of the ego vehicle, we need to convert the predicted joint distribution into the a conditional distribution. Namely, we want to get $P(\hat{\xi}^{pred}|\hat{\xi}^{ego}_{gt}, \xi)$ from $P(\hat{\xi}^{pred}, \hat{\xi}^{ego}|\xi)$. According to Bayes rule, we have:

$$P(\hat{\xi}^{pred}|\hat{\xi}^{ego}_{gt}, \xi) = \frac{P(\hat{\xi}^{pred}, \hat{\xi}^{ego}_{gt}|\xi)}{\sum_{\tilde{\xi}^{pred}} P(\tilde{\xi}^{pred}, \hat{\xi}^{ego}_{gt}|\xi)}, \tag{3.3}$$

where the numerator is an unknown distribution since it is nearly impossible to obtain a sampled future trajectory of the ego vehicle that exactly equals to the ground-truth. To resolve this problem, we first rewrite $\hat{\xi}^{ego}$ as $\hat{\xi}^{ego}_{gt} + \Delta\xi^{ego}$, where $\hat{\xi}^{ego}_{gt}$ denotes the ground truth future trajectory of the ego vehicle and $\Delta\xi^{ego}$ represents the discrepancy between the ego vehicle's ground truth and each of its possible future trajectory. Then among all the sampled joint trajectories, if any $\Delta\xi^{ego}$ is smaller than a certain threshold, we denote the predicted ego vehicle's trajectory as a satisfied trajectory $\hat{\xi}^{ego}_{s}$ that can well approximate $\hat{\xi}^{ego}_{gt}$ and we store the corresponding trajectory pair $\{\hat{\xi}^{pred}_{s}, \hat{\xi}^{ego}_{s}\}$. Finally, for all saved samples, we can rewrite (3.3) as:

$$\begin{aligned} P(\hat{\xi}^{pred}_{s}|\hat{\xi}^{ego}_{gt}, \xi) &\approx P(\hat{\xi}^{pred}_{s}|\hat{\xi}^{ego}_{s}, \xi) \\ &= \frac{P(\hat{\xi}^{pred}_{s}, \hat{\xi}^{ego}_{s}|\xi)}{\sum_{\tilde{\xi}^{pred}_{s}} P(\tilde{\xi}^{pred}_{s}, \hat{\xi}^{ego}_{s}|\xi)} \\ &\approx P(\hat{\xi}^{pred}_{s}, \hat{\xi}^{ego}_{s}|\xi). \end{aligned} \tag{3.4}$$

Note that the ground-truth trajectory of the ego vehicle is unknown if the algorithm is running online. Therefore, the ego vehicle needs to first plan for itself in real-time and use each of the planned trajectories as its ground-truth trajectory.

Optimal Trajectory Generation

Apart from sampling trajectories that may contain both rational and irrational behaviors, we can further generate the most probable trajectory by solving a finite horizon Model Predictive Control (MPC) problem using the learned continuous cost function via a planning-based method, which guarantees the resulting behavior be rational. The equation of obtaining the optimal trajectory can be expressed as:

$$\hat{\xi}^{pred}_{opt} = \arg\min_{\tilde{\xi}^{pred}} C(\boldsymbol{\theta}, \tilde{\xi}^{pred}, \hat{\xi}^{ego}_{gt}, \xi), \tag{3.5}$$

where $\boldsymbol{\theta}$ is the weight parameter vector for cost function C to determine the importance of each of the features.

Weight Ratio Update

The next step is to determine how many optimal trajectory pairs $\hat{\xi}^{pred}_{opt}$ should we add to the trajectory set so that the resampling result are reasonable. In fact, we want to keep updating the probability of both the optimal trajectory P_{opt} and the average of the satisfied trajectories obtained from the learning-based prediction approach P_s. Note that $P_{opt}+P_s = 1$ and we denote the ratio of the two probabilities as r where $r = \frac{P_s}{P_{opt}}$. The probabilities are calculated by comparing the similarities between the actual observed trajectory and the predicted trajectory for the predicted vehicle, which will be updated after each new observation at the next time step using Bayes' theorem. The number of added optimal trajectory pairs N_{opt} is then equal to $r \times N_s$, where N_s denotes the total number of satisfied trajectories $\hat{\xi}^{pred}_{s}$ from step 2). Therefore, the ratio r can be also regarded as the weight ratio between the optimal and the satisfied trajectory pairs.

Distribution Reweighting

After having all the satisfied trajectories sampled from the learning-based method along with the added optimal trajectories from the planning-based method, we are then able to re-evaluate each trajectory's probabilities using our learned cost function. According to the principle of maximum entropy, the distribution of agents' behaviors can be approximated by an exponential distribution family and thus we can write

$$P(\hat{\xi}^{pred} | \hat{\xi}^{ego}_{gt}, \xi) \propto \exp^{-C(\boldsymbol{\theta}, \hat{\xi}^{pred}, \hat{\xi}^{ego}_{gt}, \xi)}, \tag{3.6}$$

where we will minimize the probability of irrational behaviors that might generate infeasible trajectories while increasing the likelihood of rational behaviors that result in safe trajectories.

Resampling

Finally, we need to resample trajectories from the sample set according to the updated conditional distribution obtained from the previous step. These sampled trajectories are then regarded as our final prediction results at the current time step.

Algorithm 1: Proposed Generic Prediction Architecture

```
Input            : $\xi$ , map information
Output           : $\hat{\xi}^{pred}$ at each prediction time step
Trained Models   : $\mathcal{M}_L$ - learning-based model
                   $\mathcal{M}_P$ - planning-based model
while algorithm is running do
    1) $\{\hat{\xi}^{pred}, \hat{\xi}^{ego}\}_{1:N} \leftarrow$sample joint trajectories ($\mathcal{M}_L$)
    2) $\hat{\xi}^{ego}_{gt} \leftarrow$ path planning for ego vehicle
       $\{\hat{\xi}^{pred}, \hat{\xi}^{ego}_s\}_{1:N_s} \leftarrow$filter trajectories and update sample set
    3) $\hat{\xi}^{pred}_{opt} \leftarrow$generate optimal trajectory ($\mathcal{M}_P$ + MPC)
    4) $r \leftarrow$ update weight ratio
       $N_{opt} = r \times N_s \leftarrow$ add $N_{opt}$ optimal trajectories to the current sample set
    5) $P(\hat{\xi}^{pred} | \hat{\xi}^{ego}_{gt}, \xi, z) \leftarrow$ reweight samples ($\mathcal{M}_P$)
    6) $\hat{\xi}^{pred} \leftarrow$ resample from the updated conditional distribution
    Return Prediction: $\hat{\xi}^{pred}$
    Obtain new state observation from the sensor.
    $\xi$, map information $\leftarrow$ update
end
```

3.3.2 Learning-based Trajectory Prediction

The learning-based trajectory prediction method we applies is called the conditional variational autoencoder (CVAE) [27][43], which is a latent variable model that is rooted in Bayesian inference. The goal is to model the underlying probability distribution of the data using a factored, low-dimensional representation. In this problem, our objective is to estimate the probability distribution of joint trajectories in (1) by utilizing the encoder-decoder structure of CVAE.

The encoder $\mathcal{Q}_\varphi$, parameterized by φ, takes the input X as a learned embedded space of historical trajectories of all vehicles (ξ) and Y as the actual future trajectories of the two interacting vehicles ($\hat{\xi}^{ego}_{gt}, \hat{\xi}^{pred}_{gt}$) to "encode" them into a latent z-space. Then the decoder $\mathcal{P}_\psi$, parameterized by ψ, takes X and sampled z values from the latent space to "decode" them back to the future trajectories $\hat{Y}$ as the prediction result ($\hat{\xi}^{ego}, \hat{\xi}^{pred}$). The network is trained using the reparameterization trick [12] to enable back-propagation, where the network tries to minimize the evidence lower bound (ELBO) and it is formulated as:

$$\mathcal{L} = -\mathbb{E}_{\mathcal{Q}_\varphi}\big[\log \mathcal{P}_\psi(Y|X,z)\big] + \beta D_{KL}(\mathcal{Q}_\varphi(z|X,Y)||p(z)), \tag{3.7}$$

where $p(z)$ denotes the prior distribution of the latent z space and it is usually defined as a unit Gaussian. The overall idea of the loss function is to have a good estimation of data log-likelihood as well as a small Kullback-Leibler (KL) divergence denoted by D_{KL} between the approximated posterior and the prior $p(z)$ at the same time. The hyperparameter β is used to control the training balance between the two losses for better performance.

Note that during the test time, only the decoder will be used. Each predicted joint trajectories will be generated when we randomly sample one z value and feed it into the decoder network along with the historical input X.

3.3.3 Planning-based Trajectory Prediction

The planning-based trajectory prediction stems from Theory of Mind [44] which describes the prediction process of human. We let the ego vehicle simulate what the other vehicle will do assuming that it is approximately optimal planners with respect to some reward or cost functions, i.e., it is a noisily rational driver. Under this assumption, the target vehicle's driving behavior can be described via its cost function which can be learned based on demonstrations. In this chapter, we adopt the continuous domain maximum-entropy inverse reinforcement learning (IRL) [37, 38]. A brief review of the algorithm is given below.

Assume that the cumulative cost C of the target vehicle is a linear combination of a set of selected features over a defined horizon N. Then given a demonstration set U_D which contains M interactive trajectories of both the ego and target vehicles:

$$U_D = \{(\xi_{gt,i}^{pred}, \xi_{gt,i}^{ego}, \xi), i = 1, 2, \cdots, M\}, \tag{3.8}$$

we can write the cumulative cost function as

$$C(\xi_{gt}^{pred},\xi_{gt}^{ego},\xi;\theta) = \theta^T \sum_{t=0}^{N-1} \phi(x_t^{pred},x_t^{ego},\xi), \tag{3.9}$$

where ϕ is the feature vector which includes the distance between two vehicles, the speed gap with respect to the speed limit, the acceleration, and the lateral deviation from the target lane. x_t^{pred} and x_t^{ego} are, respectively, the states of the target vehicle and the ego vehicle at time instant t within the planning horizon.

Building on the principle of maximum entropy, we assume that trajectories are exponentially more likely when they have lower cost:

$$P(\xi_{gt}^{pred}|\xi_{gt}^{ego}, \xi) \propto \exp\left(-C(\xi_{gt}^{pred},\xi_{gt}^{ego},\xi;\theta)\right). \tag{3.10}$$

Then, our goal is to find the weight θ which maximizes the likelihood of the demonstration set U_D:

$$\begin{aligned}\theta^* &= \arg\max_{\theta} P(U_D|\theta) \\ &= \arg\max_{\theta} \Pi_{i=1}^{M} \frac{P(\xi_{gt,i}^{pred}|\xi_{gt,i}^{ego}, \xi_i, \theta)}{P(\theta)} \\ &= \arg\max_{\theta} \Pi_{i=1}^{M} \frac{P(\xi_{gt,i}^{pred}|\xi_{gt,i}^{ego}, \xi_i, \theta)}{\int P(\tilde{\xi}_{gt}^{pred}|\xi_{gt,i}^{ego}, \xi_i, \theta) d\tilde{\xi}_{gt}^{pred}}\end{aligned} \tag{3.11}$$

To tackle the partition term $\int P(\tilde{\xi}_{gt}^{pred}|\xi_{gt,i}^{ego}, \xi_i, \theta) d\tilde{\xi}_{gt}^{pred}$ in (3.11), we approximate C with its Laplace approximation as proposed in [38]:

$$\begin{aligned}&C(\tilde{\xi}_{gt}^{pred}, \xi_{gt,i}^{ego}, \xi_i; \theta) \\ \approx\ &C(\xi_{gt,i}^{pred}, \xi_{gt,i}^{ego}, \xi_i; \theta) + \left(\tilde{\xi}_{gt}^{pred} - \xi_{gt,i}^{pred}\right)^T \frac{\partial C}{\partial \xi_{gt}^{pred}} + \frac{1}{2}\left(\tilde{\xi}_{gt}^{pred} - \xi_{gt,i}^{pred}\right)^T \frac{\partial^2 C}{\partial \xi_{gt}^{pred}} \times \left(\tilde{\xi}_{gt}^{pred} - \xi_{gt,i}^{pred}\right).\end{aligned} \tag{3.12}$$

With the assumption of locally optimal demonstrations, we have $\frac{\partial C}{\partial \xi_{gt}^{pred}}|_{\xi_{gt,i}^{pred}} \approx 0$ in (3.12). This simplifies the partition term $\int P(\tilde{\xi}_{gt}^{pred}|\xi_{gt,i}^{ego}, \xi_i, \theta) d\tilde{\xi}_{gt}^{pred}$ as a Gaussian Integral where a closed-form solution exists (see [38] for details). Substituting (3.12) into (3.11) yields the optimal parameter θ^* as the maximizer.

Once we have learned the cost function, we can use it to either evaluate the probabilities of given trajectory samples, or generate the most probable trajectory by solving a MPC problem, as explained in Section 3.3.1.

3.4 Experiments and Results

In this section, we first introduce the scenario that we used in the experiment. Afterwards, we evaluate the prediction results with only the learning-based part of the proposed architecture. Finally, we assess the quality of the proposed model architecture in different aspects.

3.4.1 Real-world Scenario

We conduct experiments on a roundabout scenario included in the INTERACTION dataset [45, 46]. It is a 8-way roundabout and each of the branch has one entry lane and one exit lane. The bird-view image of the roundabout as well as the reference path information can be found in [45, 27, 46]. We manually selected 1120 highly interactive driving segments

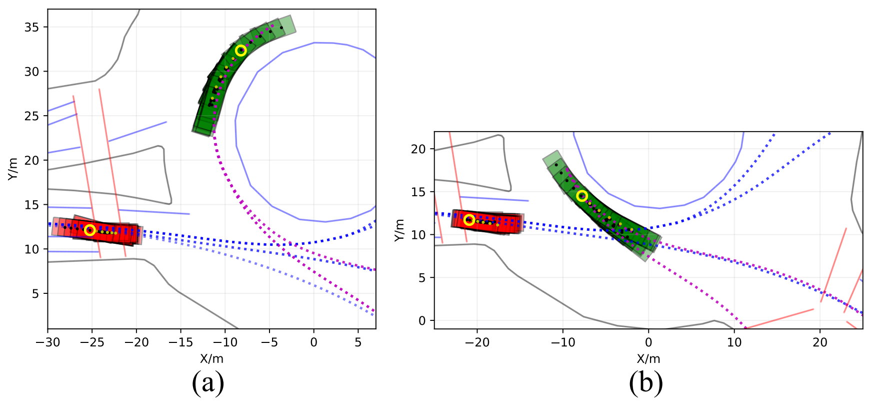


Figure 3.3: Visualized prediction results of a selected scene at different time steps. The red car is the ego vehicle and the green car is the predicted vehicle. We plotted both the historical and the predicted trajectories of each vehicle, where the yellow circles represent the current state of two vehicles. The small yellow dots denote ground-truth states and the dashed lines are the possible reference paths for both vehicles.

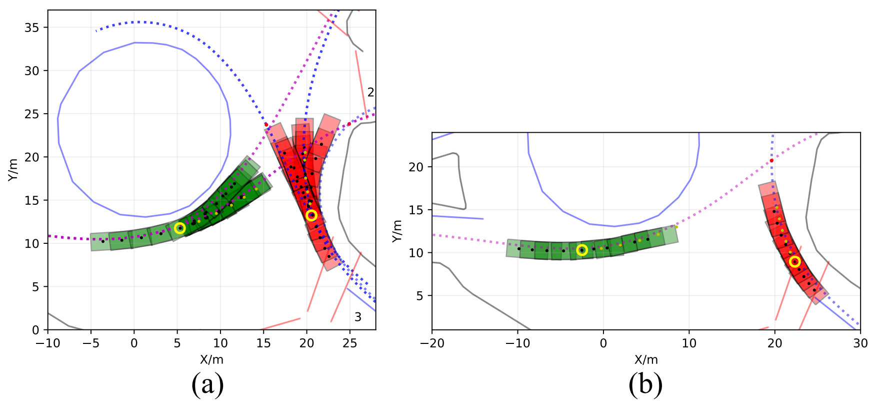


Figure 3.4: Generalization results at an unseen roundabout entrance.

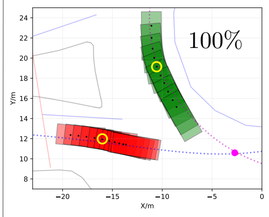
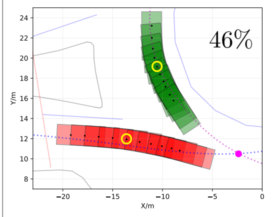
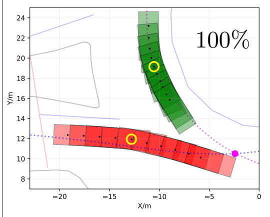
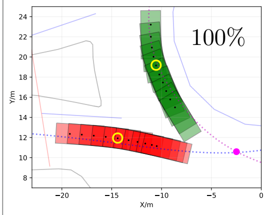
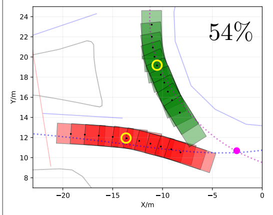
	(a) $4m/s$	(b) $6m/s$	(c) $7m/s$	(d) $8m/s$
Rational				
Irrational				

Figure 3.5: Selected artificial test scenarios and the corresponding results. The pink dot represents the cross point of two vehicles' ground-truth reference paths. The number on the top right corner denotes the percentage rate of rational or irrational behaviors.

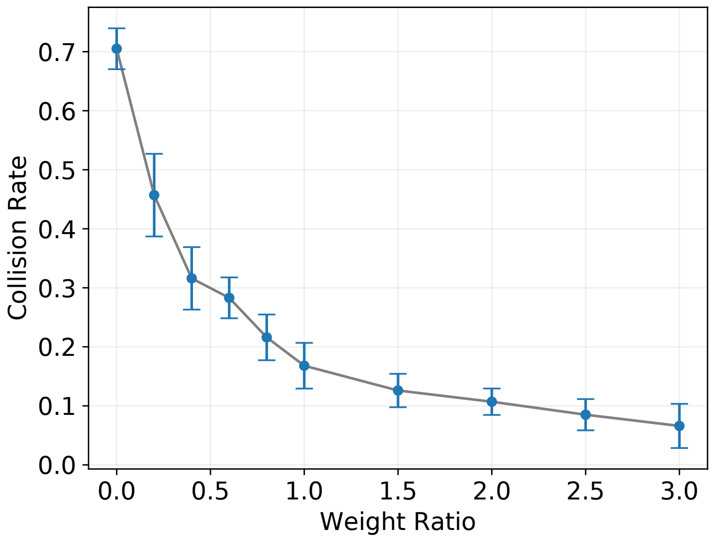


Figure 3.6: Weight ratio versus collision rate.

in the dataset and used 80 % for training and the remaining for testing. We define the car that is about to enter the circular roadway as the ego vehicle and the car that is already driving on the circular roadway as the predicted vehicle. In this problem setting, our goal is to predict 1s into the future using the past 1s information with a sampling frequency of 5Hz.

3.4.2 Learning-based Trajectory Prediction

Prediction Accuracy

We calculated the root mean squared error (RMSE) between the predicted and ground-truth state for both interacting vehicles at each future time step. The results are shown in the table below. According to the table, the mean RMSE error and the standard deviation continuously increase as the prediction horizon extends. However, prediction errors for both vehicles are all within an acceptable range even when the prediction horizon reaches one second.

Table 3.1: Evaluation Results (m)

	0.2s	0.4s	0.6s	0.8s	1.0s
ego	0.07±0.01	0.16±0.05	0.29±0.12	0.40±0.20	0.51±0.29
pred	0.08±0.02	0.18±0.04	0.33±0.10	0.50±0.18	0.70±0.26

Generalization Ability

To illustrate that the proposed prediction algorithm is able to generalize well under unseen scenarios, we select a test data from another entrance of the roundabout that is not considered in the training data. As shown in Fig. 3.4, although the location of the interaction changes, our architecture can still have reasonable prediction results and can generate multi-modal trajectories.

Failures under Unseen/Corner Cases

Even if the learning-based prediction method has been proved to have good testing performances in the previous section, the model can still fail under some corner cases that have certain discrepancies from the collected data as discussed in Section 3.1.2. To demonstrate potential failure testing cases, we formulated four different artificial test scenarios that are unseen from the training set. We fixed ξ^{pred} and the initial state of the ego vehicle and assumed that the ego vehicle would drive at constant speed during the historical time steps. We then assigned four different velocities to the ego vehicle: $\{4m/s, 6m/s, 7m/s, 8m/s\}$.

From the prediction results in Fig. 3.5, the sampled trajectories in case (a) and (d) are all feasible and collision free. When ego vehicle's historical motion changes slightly as in (b) and (c), infeasible trajectories are predicted, where a collision occurs. However, we cannot simply conclude that the prediction method failed in every cases that a collision is predicted since under some dangerous circumstances, the predictor is expected to generate results that can reflect such situation.

Therefore, to further analyze if (b) and (c) are indeed failure test cases, we first checked whether the two vehicles have any chances to avoid collision given their initial states for each testing case. We applied a constant deceleration model on one vehicle while letting the other vehicle drive with constant speed. Our result shows that if any of the two vehicles brakes in time, collision can be avoided. Therefore, the predicted result in all four scenarios should be expected to be either collision free or have small collision rate. However, as observed in Fig. 3.5, all the sampled joint trajectories in case (b) and more than half percent of the samples in case (c) are infeasible. Therefore, we concluded that that these two scenarios are indeed failure testing cases as they violate the common consensus of human behaviors, which verifies the drawback of using pure learning-based method for trajectory prediction.

3.4.3 Overall Framework Evaluation

Convert to Conditional Distribution

In order to convert the predicted joint distribution to conditional distribution, we need to filter out the predicted trajectories for the ego vehicle $\hat{\xi}^{ego}$ that are far from its planned ground-truth trajectory $\hat{\xi}^{ego}_{gt}$.

We obtained the trajectory discrepancies by calculating the RMSE error between the last state of each predicted trajectory and that of the actual trajectory. We set the discrepancy threshold to 0.2 meter and thus the sampled joint trajectory will be retained only when $|\hat{\xi}^{ego}_{gt}(t_{final}) - \hat{\xi}^{ego}(t_{final})| \leq 0.2m$, where t_{final} is the the final prediction time step.

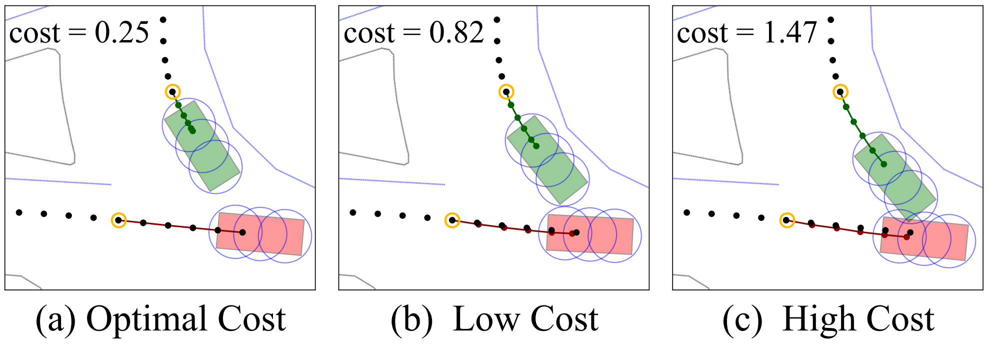


Figure 3.7: Different sampled trajectories and their corresponding costs at a selected scenario.

Trajectory Cost Evaluation

We defined three categories of driving behaviors for the predicted vehicle: optimal, rational, and irrational. In order to demonstrate the cost differences among these categories, we plotted one sampled trajectory from each category and calculated the corresponding cost C after acquiring the parameters $\boldsymbol{\theta}$ of the cost function from the IRL algorithm (Fig. 3.7). According to the corresponding cost of each sample, we can conclude that the learned IRL cost parameters are able to assign high cost when collision occurs due to irrational behavior (Fig. 3.7(c)) and low cost when the predicted trajectory results in rational behavior (Fig. 3.7(b)). In this testing scenario, the optimal cost is achieved when the predicted vehicle applies maximum deceleration ($4m/s^2$) at every future time steps as shown in Fig. 3.7(a).

Distribution Re-weighting

One of the most important aspects needs to be evaluated is whether we are indeed able to reasonably change the original sample distribution. Besides, we need to further examine the correlations between the weight ratio factor r and the final outcomes. First of all, under a selected testing scenario, we let $N_{opt} = 1$ during the 4^{th} step in Algorithm 1, where we added only one optimal trajectory to the current sample set and resampled N final trajectories for evaluation. We then gradually increased N_{opt} while keep sampling the same number of final samples. Lastly, we calculated the collision rate of the sampled trajectory set corresponds to each r and plotted the result in Fig. 3.6.

From the plot, we notice that when $r = 0$, where no optimal trajectory is added, the collision rate is 0.7 from the pure learning-based prediction results. As the planning-based model is introduced to the algorithm and the weight ratio gradually increases, the collision rate decreases dramatically at the beginning and slowly converges to 0 as r passes 1. Such result is reasonable since when more rational and feasible samples with lower costs are added to the sample set, the original conditional distribution will not only gradually shifts away from the infeasible sample area horizontally but also gets lower probability at those irrational sample points as illustrated in the reweighting part of Fig. 3.2. Therefore, we concluded that the proposed architecture is capable of reshaping the original sample distribution to desired outcomes when it is used online with a constantly updating r value.

Final Results Visualization

We selected the same scenario as in 3) to compare the prediction result of our framework with the pure learning-based prediction method. Note that the online path planning for ego vehicle is not the focus of our current work and thus we will not visualize the prediction result of applying online update to the weight ratio. Instead, we assigned r to a fixed ratio and visualized the prediction result at a single time step (Fig. 3.8).

The result of directly sampling from the learned learning-based prediction models is shown in Fig. 3.8(a), which corresponds to step 1) in Algorithm 1. After converting the joint distribution to conditional distribution in step 2), the remaining satisfied trajectories

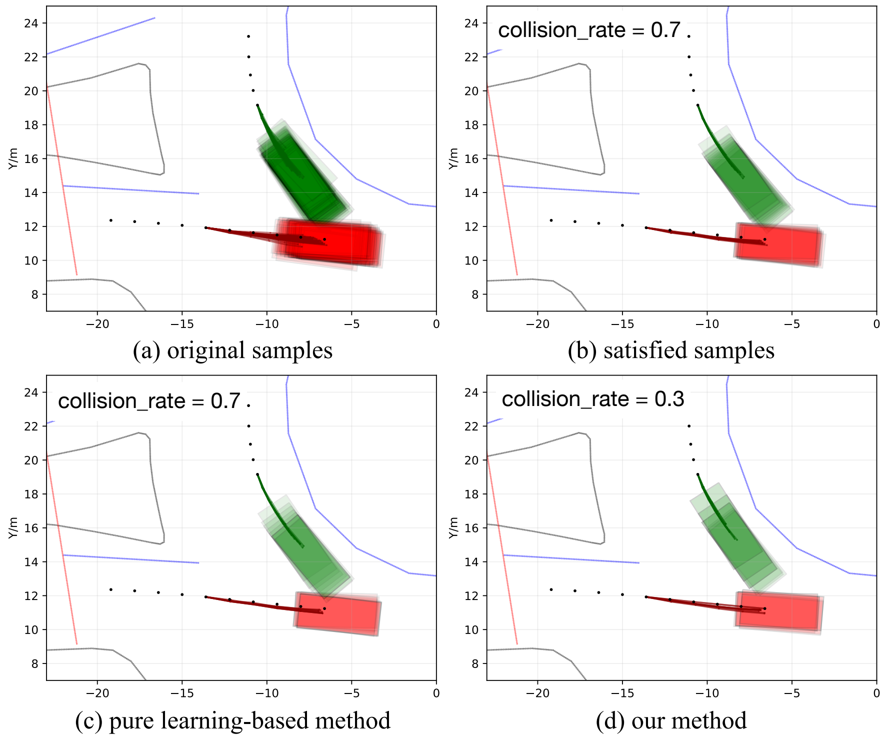


Figure 3.8: Visualization of the overall framework at different stages and a result comparison with the pure learning-based prediction method.

are shown in Fig. 3.8(b), where we notice that all the ego vehicle's predicted trajectories are close to the ground-truth as desired. The collision rate in (b) is 0.7, which implies that more irrational than rational behaviors are predicted. We then directly sampled from the current sample set and obtained the prediction results by using only the learning-based method as in Fig. 3.8(c). Comparing to the predicted results in Fig. 3.8(c) , the generated trajectories of our approach in Fig. 3.8(d) tend to have more rational behaviors while still having a collision rate of 0.3 as a warning to the ego vehicle. In this way, the ego vehicle is able to have both rational and irrational information about the predicted vehicle's future behavior, where it believes that the predicted vehicle will be more likely to behave rationally in the future while preparing for possible irrational behaviors.

3.5 Conclusions

In this chapter, a generic prediction architecture is proposed, which can predict continuous trajectories of other vehicles by considering both rational and irrational driving behaviors. An exemplar roundabout scenario with real-world data was used to demonstrate the performance of our method. We first evaluated the prediction accuracy of the learning-based method on the test dataset. Then, we demonstrated the generalizability of the method by using our generic environmental representation. By testing on some unseen and corner driving scenarios, we revealed the limitations of using pure learning-based prediction method. Finally, by thoroughly examining the proposed architecture, we concluded that the approach of combining both learning-based and planning-based method can enhance the overall prediction performance by providing sufficient possible outcomes to the ego vehicle. For future work, we will perform human-in-the-loop experiments online using the proposed prediction architecture to evaluate its capabilities.

Part II

Multi-agent Interaction

Chapter 4

Semantic Intention and Motion Prediction

4.1 Introduction

Safety is the most fundamental aspect to consider for both human drivers and autonomous vehicles. Human drivers are capable of using past experience and intuitions to avoid potential accidents by predicting the behaviors of other drivers. However, some drivers have poor driving habits such as changing lanes without using turn signals, which adds difficulties for prediction. Moreover, human drivers might easily overlook dangerous situations due to limited concentration. Therefore, the Advanced Driver Assistance Systems (ADAS) should have the ability to simultaneously and accurately anticipate future behaviors of multiple traffic participants under various driving scenarios, which may then assure a safe, comfortable and cooperative driving experience.

There have been numerous works focused on predicting vehicle behavior which can be divided into two categories: **intention/maneuver prediction** and **motion prediction**. Many intention estimation problems have been solved by using classification strategies, such as Support Vector Machine (SVM) [47], Bayesian classifier [48], Multilayer Perceptron (MLP) [49], and Hidden Markov Models (HMMs) [50]. Most of these approaches were only designed for one particular scenario associated with limited intentions. For example, [47, 48, 49, 50] dealt with non-junction segment such as highway, which involves lane keeping (LK), lane change left (LCL) and lane change right (LCR) maneuvers. Whereas [5, 49, 50, 51] concentrated on junction segment such as intersection, which includes left turn, right turn, and go straight maneuvers. However, in order for autonomous vehicles to drive through dynamically changing traffic scenes in real life, an intention prediction module that can adapt to different scenarios with various possible driving maneuvers is necessary. [52] proposed a maneuver estimation approach for generic traffic scenarios, but the classified driving maneuvers are too specific, which will not only require multiple manually-selected classification thresholds, but also raise problems when unclassified maneuvers occur.

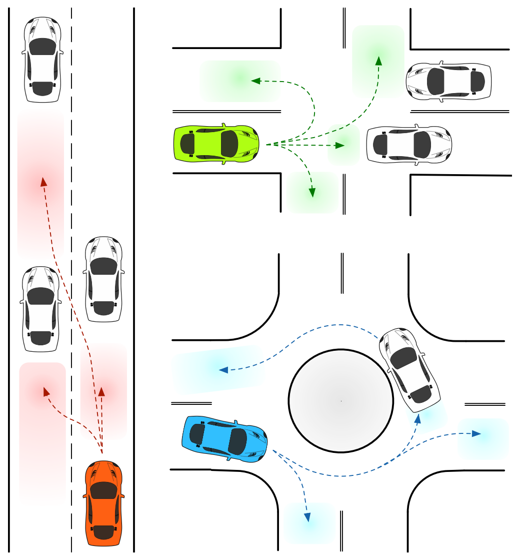


Figure 4.1: Insertion areas (colored regions) under different driving scenarios for the predicted vehicle.

As a result, we proposed to use semantics to represent the driver intention, which is defined as the intent to enter each *insertion area*. These areas can be the available gaps between any two vehicles on the road or can be the lane entrances/exits. Fig. 4.1 visualizes the insertion areas under distinct environments. An advantage of using semantic approach is that situations can be modeled in a unified way [53] such that varying driving scenarios will have no effect on our semantics defined problem. Even for a scenario that has a combination of all the road structures in Fig. 4.1, the proposed semantic definition still holds.

Motion prediction is mostly treated as a regression problem, where it tries to forecast the short-term movements and long-term trajectories of vehicles. By incorporating motion prediction with intention estimation, not only the high-level behavioral information, but also the future state of the predicted vehicle can be obtained. For short-term motion prediction, various approaches such as constant acceleration (CA), Intelligent Driver Model (IDM) [5], and Particle Filter (PF) [3] have been suggested. The main limitation of these works, however, is that they either considered simple cases such as car following or did not take environment information into account.

For future trajectory estimation, Dynamic Bayesian Networks (DBN) [20] and other regression models [54] have been used in several studies. Methods based on artificial neural network (ANN) are also widely applied. In [55], the authors used the LSTM to predict the vehicle trajectory in highway situation. [56] brought forward a Deep Neural Networks (DNN) to obtain the lateral acceleration and longitudinal velocity. However, these approaches only predicted the most likely trajectory for the vehicle without considering uncertainties in the environment. To counter this issue, a Variational Gaussian Mixture Model (VGMM) was proposed for probabilistic long-term motion prediction [57]. Nevertheless, the method was only tested in a simulation environment and the input contains history information over a long period of time, which is usually unaccessible in reality. There are also researches that project the prediction step of a tracking filter forward over time, but the growing uncertainties often cause future positions to end up at some physically impossible locations.

In contrast, works such as [58] and [59] highlighted that by predicting goal locations and assuming that agents navigate toward those locations by following some optimal paths, the accuracy of long-term prediction can be improved. The main advantage of postulating destinations instead of trajectories is that it allows one to represent various dynamics and to automatically incorporate environment constraints for unreachable regions.

Apart from obtaining the possible goals of predicted vehicles, the required time to reach those locations is also an essential information especially for the subsequent trajectory planning of the ego vehicle. Therefore, many attempts have been made in order to directly predict temporal information. [60] used LSTM to forecase time-to-lane-change (TTLC) of vehicles under highway scenarios. A recent work [61] utilized the Linear Quantile Regression (LQR) and Quantile Regression Forests (QRF) methods for the probabilistic regression task of TTLC. The authors also concluded that QRF has better performance than LQR.

In this chapter, Semantic-based Intention and Motion Prediction (SIMP) method is proposed. It utilizes deep neural network to formulate a probabilistic framework which can predict the possible semantic intention and motion of the selected vehicle under various driving scenarios. The introduced semantics for this prediction problem is defined as answering the question of *"Which area will the predicted vehicle most likely insert into? Where and when?"*, which incorporates both the goal position and the time information into each insertion area. Moreover, the adoption of probability can take into account the uncertainty of drivers as well as the evolution of the traffic situations.

The remainder of the chapter is organized as follows: Section 4.2 provides the concept of the proposed SIMP method; Section 4.3 discusses an exemplar scenario to apply SIMP; evaluations and results are provided in Section 4.4; and Section 4.5 concludes the chapter.

4.2 Concept of Semantic-based Intention and Motion Prediction (SIMP)

In this section, we first provide a brief overview of Mixture Density Network (MDN), which is an idea we utilize for our proposed method. Then, the detailed formulation and structure of the SIMP method are illustrated.

4.2.1 Mixture Density Network (MDN)

Mixture Density Network is a combination of ANN and mixture density model, which was first introduced by Bishop [62]. The mixture density model can be used to estimate the underlying distribution of data, typically by assuming that each data point has some probability under a certain type of distribution. By using a mixture model, more flexibility can be given to completely model the general conditional density function $p(\boldsymbol{y}|\boldsymbol{x})$, where $\boldsymbol{x}$ is a set of input features and $\boldsymbol{y}$ is a set of output. The probability density of the target data is then represented as a linear combination of kernel functions in the form

$$p(\boldsymbol{y}|\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m(\boldsymbol{x})\phi_m(\boldsymbol{y}|\boldsymbol{x}), \tag{4.1}$$

where M denotes the total number of mixture components and the parameter $\alpha_m(\boldsymbol{x})$ denotes the m-th mixing coefficient of the corresponding kernel function $\phi_m(\boldsymbol{y}|\boldsymbol{x})$. Although various choices for the kernel function was possible, for this work, we utilize the Gaussian kernel of the form

$$\phi_m(\boldsymbol{y}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{y}|\mu_m(\boldsymbol{x}), \sigma_m^2(\boldsymbol{x})). \tag{4.2}$$

Such formulation is called the Gaussian Mixture Model (GMM)-based MDN, where a MDN maps input $\boldsymbol{x}$ to the parameters of the GMM (mixing coefficient α_m, mean μ_m, and variance σ_m^2), which in turn gives a full probability density function of the output $\boldsymbol{y}$. It is important to note that the parameters of the GMM need to satisfy specific conditions in order to be valid: the mixing coefficients α_m should be positive and sum to 1; the standard deviation σ_m should be positive. The use of softmax function and exponential operator in (4.3) fulfills the aforementioned constraints. In addition, no extra condition is needed for the mean μ_m.

$$\alpha_m = \frac{\exp(z_m^{\alpha})}{\sum_{i=1}^{M}\exp(z_i^{\alpha})}, \quad \sigma_m = \exp(z_m^{\sigma}), \quad \mu_m = z_m^{\mu} \tag{4.3}$$

The parameters z_m^{α}, z_m^{σ}, z_m^{μ} are the direct outputs of the MDN corresponding to the mixture weight, variance and mean for the m-th Gaussian component in the GMM.

The objective of training the MDN is to minimize the negative log-likelihood as loss function

$$Loss = -\sum_{n} log\left\{\sum_{m=1}^{M} \alpha_m^n(\boldsymbol{x}^n)\phi_m(\boldsymbol{y}^n|\boldsymbol{x}^n)\right\}, \tag{4.4}$$

where n denotes the number of training data. The detailed derivations on closed-form gradient formulation can be found in [62], which demonstrated the capability of training the MDN using back propagation.

4.2.2 Proposed SIMP Method

Our task is to generate probability distributions of the designed semantic description given some representation of the current state. We assign a Gaussian Mixture Model (GMM) to each insertion area and multiple GMMs will be involved in one driving scenario. Each Gaussian mixture models the probability distribution of a certain type of motion for the predicted vehicle. Since obtaining the insertion location and the arriving time are the focus of our interests, a 2D Gaussian mixture is used and the predicted variables are constructed as a two dimensional vector: $\boldsymbol{y} = [y_s, y_t]^T$. The variable y_s describing the vehicle locations and the variable y_t describing the time information, can be specifically defined according to the driving environment.

Given the current state features $\boldsymbol{x}$, the probability distribution $\boldsymbol{y}_a$ over a single area a for the predicted vehicle is of the form

$$f(\boldsymbol{y}_a|\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m \mathcal{N}(\boldsymbol{y}_a|\boldsymbol{\mu}_m, \Sigma_m) \tag{4.5}$$

with mean and covariance constructed as

$$\boldsymbol{\mu}_m = \begin{bmatrix} \mu_{s,m} \\ \mu_{t,m} \end{bmatrix}, \quad \Sigma_m = \begin{bmatrix} \sigma_{s,m}^2 & \rho_m \sigma_{s,m} \sigma_{t,m} \\ \rho_m \sigma_{s,m} \sigma_{t,m} & \sigma_{t,m}^2 \end{bmatrix}, \tag{4.6}$$

where $\rho_m \in [-1, 1]$ is the correlation coefficient.

In addition to formulate a regression model for each insertion area, we also require the probability of entering each area for the predicted vehicle. Therefore, Deep Neural Network (DNN) was used as the basis for our Semantic-based Intention and Motion Prediction (SIMP) structure. The output of the network contains both necessary parameters for every 2D Gaussian Mixture Model (GMM) and the weight w_a for each insertion area a.

For the desired outputs, we expect not only the largest weight to be associated to the actual inserted area, but also the highest probability at the correct location and time for the output distributions of that area. Consequently, we define our loss function as

$$\begin{aligned} L = W_1 \Big(- \sum_n log \Big\{ \sum_{a=1}^{N_a} \hat{w}_a^n f(\boldsymbol{y}_a^n|\boldsymbol{x}) \Big\} \Big) \\ + W_2 \Big(- \sum_n \sum_{a=1}^{N_a} \hat{w}_a^n log(w_a^n) \Big), \end{aligned} \tag{4.7}$$

where N_a denotes the total number of insertion areas and $\hat{w}_a$ denotes the ground truth, which is the one-hot-encoding of the final area that the predicted vehicle entered. The last term denotes the cross-entropy loss of the area weights. Parameters $W1$ and $W2$ need to be manually tuned such that the two loss components will have the same order of magnitude during training.

The overall architecture of our SIMP method is shown in Fig. 4.2. Due to the first-order Markov assumption, the input features depend only on the current time step. The network consists of an input layer, several fully connected layers, and a dropout layer which ensures better generalization and prevents overfitting of the training data. After passing

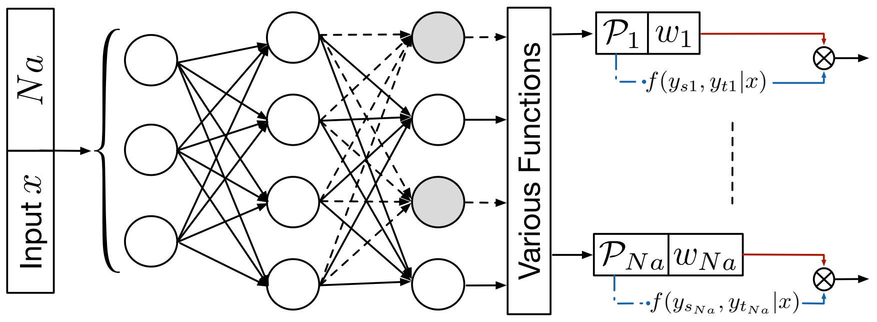


Figure 4.2: Structure of the SIMP Method

different types of parameters through corresponding functions, the output will satisfy the aforementioned constraints. For N_a insertion areas, the total number of output parameters can be calculated as: $N_a * (M * 6 + 1)$. The interpretation is: there is a weight parameter w_a associated to each area $a \in N_a$, and for every $m \in M$ within an area, six parameters are needed, $\mathcal{P}_a^m = \{\alpha_m, \mu_{s,m}, \mu_{t,m}, \sigma_{s,m}, \sigma_{t,m}, \rho_m\}$, to formulate the 2D GMM.

4.3 An Exemplar Highway Scenario

In this section, we use an exemplar highway scenario to apply the proposed Semantic-based Intention and Motion Prediction (SIMP) method. The data source and detailed problem formulation are presented.

4.3.1 Dataset

All the data we used was taken from the NGSIM US 101 dataset which is publicly available online at [63]. It contains detailed vehicle trajectory data collected on the highway with 10 Hz sampling frequency. The measurement area is approximately 640 meters (2100 feet) in length and there are five freeway lanes plus an auxiliary lane for the on/off-ramp. For each vehicle that performs a lane change maneuver, we picked up to 40 frames (4s) before the vehicle's center intersects the lane mark; for vehicles that keep driving on the same lane for a long period, we considered these frames as input for the lane keeping maneuver. A total of 17,179 frames were selected from the dataset and split into 80% for training and 20% for testing.

4.3.2 Scenario and Problem Description

A representation of the exemplar highway driving scenario is shown in Fig. 4.3. The yellow car is the vehicle we decide to predict; the three blue cars (*car2, car4*, and *car6*) are the reference vehicles, which are selected as having the closest Euclidean distance to the

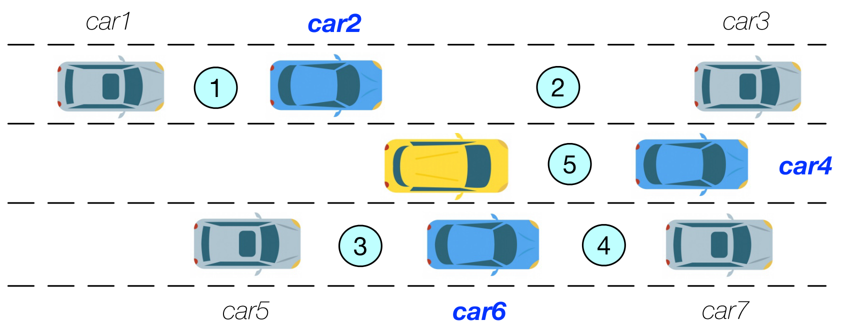


Figure 4.3: An exemplar driving scenario.

predicted vehicle on each of the three lanes (we consider only the front vehicle on predicted vehicle's lane); the four gray cars (*car1, car3, car5*, and *car7*) are named as 'other vehicles', which are vehicles in front and behind each of the two reference cars: *car2* and *car6*. If any of these surrounding vehicles is too far from the predicted vehicle, we consider it as nonexistence within the range of the current scenario. Therefore, for each input frame, a maximum of three driving lanes and seven vehicles are considered.

In Fig. 4.3, there are five circled areas that our predicted vehicle could end up going into and we name them as Dynamic Insertion Area (DIA). If the predicted vehicle (yellow car) inserts into area 1-4, a lane change behavior is indicated; however, if it inserts into area 5, a lane keeping behavior is implied. These areas are dynamic because both their locations and sizes will vary at each time step.

In this particular highway scenario setting, the output y_s represents the absolute distance between the final insertion point and the corresponding reference vehicle for that inserted area; y_t represents the time-to-lane-change (TTLC) of the predicted vehicle. When the center of the vehicle intersects the lane mark, TTLC $= 0$. For the lane keeping situation, TTLC is set to a large number (4s) to represent that the vehicle has not yet decided to change the lane.

4.3.3 Features and Structure Details

For each input frame, a total of 25 input features are selected which are listed in Table 4.1. Each input frame corresponds to 3 types of labels extracted from data: area weight, final goal location, and remaining insertion time. According to the data, the longitudinal direction is the driving direction. The current lane center (CLC) denotes the midpoint of the current lane. Because of the small angle difference between the front and the predicted vehicle, only the relative angle information for the left and right reference vehicles are considered. Time-to-collision (TTC) is calculated by dividing the speed difference by the relative distance of two vehicles. We compute the inverse of time-to-collision (iTTC) instead due to the existence of infinity TTC value as the speed difference gets close to zero.

As mentioned previously, there will be a maximum of seven cars within each input frame. If, however, a vehicle does not exist, we assign its longitudinal distance to a large number and its velocity to be the same as that of the predicted vehicle. If there is no available lane on one side of the predicted vehicle, we set the three vehicles in that nonexistent lane to be close to each other and the reference vehicle to be directly above/below the predicted vehicle. Similarly, all these three vehicles are set to have the same speed as the predicted vehicle. Such setting can guarantee the feasibility of the predicted results.

As for the network structure, we use three fully connected layers of 400 neurons each, with *tanh* non-linear activation function. After that, a dropout layer of rate 0.5 is appended. The parameter N_a is five for this particular scenario.

Table 4.1: Features for One Input Frame

	Feature	*Description*
Predicted Vehicle	v^y_{pred}	Absolute velocity in longitudinal direction.
	$d^x_{CLC_{pred}}$	Lateral distance to the current lane center.
Reference Vehicles	v^y_{ref}	Absolute velocity in longitudinal direction for every reference vehicle.
	$d^y_{ref,pred}$	Relative longitudinal position between predicted vehicle and every reference vehicle.
	$d^x_{(l,r),pred}$	Relative lateral position between left/right reference vehicle and predicted vehicle.
	$\theta_{(l,r),pred}$	Relative angle between left/right reference vehicle and predicted vehicle.
	$iTTC_{f,pred}$	Inverse time-to-collision between front reference vehicle and predicted vehicle.
Other Vehicles	v^y_o	Absolute velocity in longitudinal direction.
	$d^y_{o,pred}$	Position in longitudinal direction, relative to predicted vehicle.
	$iTTC_{o,ref}$	Inverse time-to-collision relative to corresponding reference vehicle.

4.4 Evaluation and Results

In this section, different evaluation techniques are presented to assess the model quality and the final results are discussed.

4.4.1 Evaluation Setup

Baseline Model

To evaluate our SIMP method, we trained a Support Vector Machine (SVM) and a Quantile Regression Forests (QRF) [64] separately. Since SVM is wildly used for classification problems, we compared it with the intention prediction part of our framework. The QRF is a combination of Quantile Regression and Random Forests [65], which extends the concept of tree ensemble learning to probabilistic prediction. Instead of point estimating the conditional mean for the selected variables like other regression methods, the objective is to estimate an arbitrary conditional quantile. The quantiles can provide detailed information of the minimum and maximum values for the dependent variable and encompass the uncertainty estimation. Hence, we compared our motion prediction part of the probabilistic framework with the QRF method for evaluation. The details of the baseline models are presented below

- SVM: kernel = (Gaussian) radial basis function (RBF)
- QRF: ntree = 1000, mtry = 5, nodesize = 10

where *ntree* is the number of trees in the forest, *mtry* is the number of random features in each tree, and *nodesize* is the minimal size of terminal nodes. All these parameters were selected using five-fold cross validation.

Evaluation for Intention Estimation

For training and testing, each sample from our data was assigned to a semantic intention class, which is expressed as $I \in \{area1, area2, area3, area4, area5\}$. However, since these dynamic insertion areas (DIA) change constantly during the driving period, it is hard to detect the final insertion area at the early stage. Therefore, for better evaluation at the beginning of the input driving segments, we merged the original five semantic intentions into three: $\{LCL, LCR, LK\}$, where $\{area1, area2\} \in LCL$, $\{area3, area4\} \in LCR$, and $\{area5\} \in LK$. During training, the input features for SVM were the same as our method, and the labels were the corresponding final DIA numbers. The evaluation contains three steps:

i. For all testing data, create the Receiver Operating Characteristic (ROC) curve to compare our method with SVM. (Use the simplified 3 intention classes.)

ii. Find the best threshold from the ROC curve and use it to calculate the *recall, precision, F1 score* as well as the *average prediction time* for both methods.

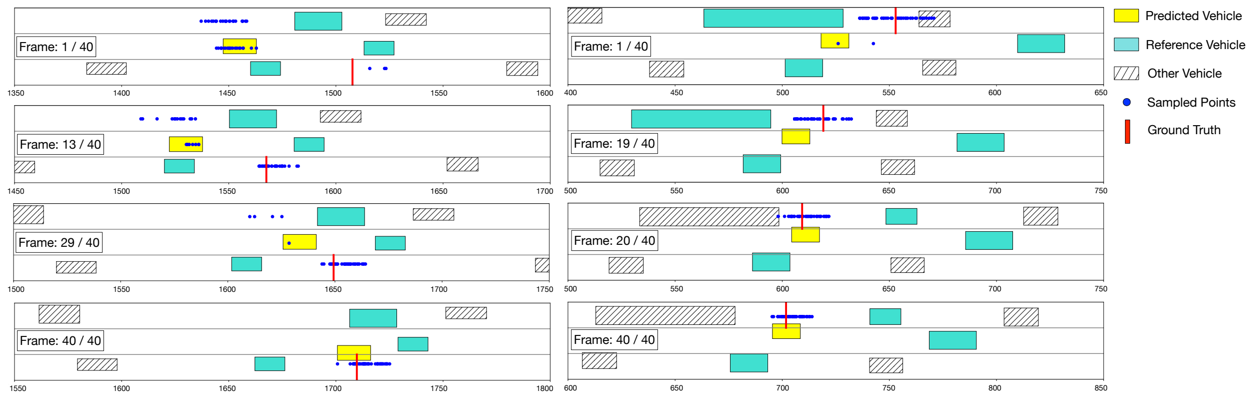
iii. For testing data that has a TTLC smaller than the obtained *average prediction time*, analyze the performance of each DIA. (Use the original 5 semantic intention classes.)

Evaluation for Motion Prediction

In our problem setting, two semantic described motions are predicted: final locations in each insertion area (destination) and the remaining time to reach those locations (TTLC). For the conditional distribution of each motion, we expect not only small difference between the predicted mean and the actual value, but also centralized distribution around the predicted mean. Hence, we evaluated the *root mean squared error* (RMSE) of the output mean as well as the confidence interval for both the QRF and the SIMP method. The number of mixture components M for each DIA was set to one for analysis purpose. For the training process of QRF, we trained two separate random forest quantile regressors, where the input features remains the same and the label is either the location or the time information.

Two different intervals were chosen to assess the testing results for each method:

- SIMP-1σ: one standard deviation interval
- SIMP-2σ: two standard deviation interval
- QRF-68%: 16% to 84% quantile interval
- QRF-95%: 2.5% to 97.5% quantile interval.



(a) Typical Lane Change (b) Sudden Change of Reference Vehicle

Figure 4.4: Two example cases to visualize the performance. Fifty points were assigned to each testing frame and were sampled by following two steps: 1) multiply the total number of dots by each DIA weight. 2) for every dot assigned to each DIA, sample it according to the corresponding distribution of that area. *(The unit of the horizontal axis is in feet.)*

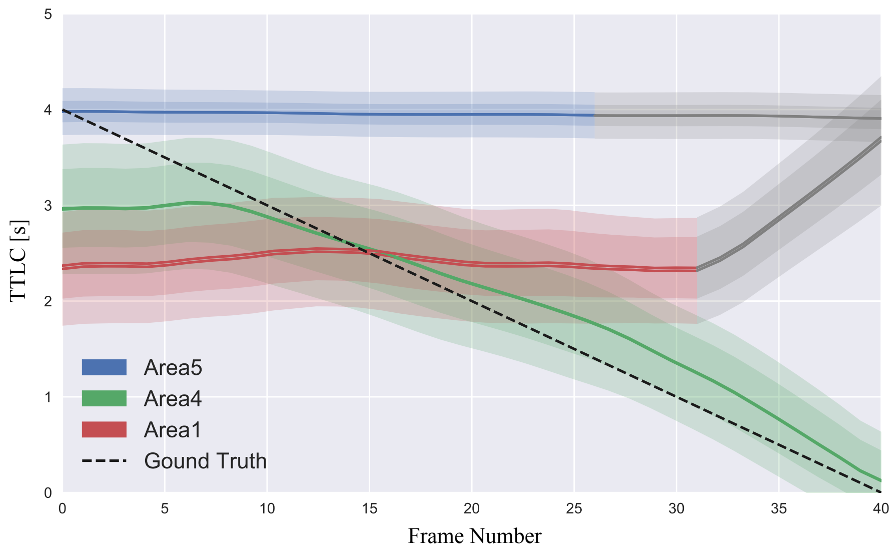


Figure 4.5: TTLC illustration for the case in Fig. 4.4(a). We sampled 100 points from the mixture distribution of each related DIA and plotted the mean as well as the 3σ and 1σ prediction intervals for these samples. When area weight is too small to be associated with sampled points, the TTLC result of that area at the corresponding frame will be colored in gray.

4.4.2 Results and Discussion

Visualization of Selected Cases

We selected two distinct traffic situations to visualize our results. Each situation had 40 frames (4s) and we chose four representative frames from each case to illustrate the overall performance. [1]

A typical lane change situation is illustrated in Fig. 4.4(a) where the sampled points are all in the proper DIA for each frame. It is reasonable to have several possible areas at the early stage since there are multiple choices for the driver and no specific one has been chosen yet. It should be noted that it is difficult to numerically justify the correctness of these circumstances without using the human-labeled ground truth. However, as soon as the driver decides where to go, our result could be compared with the label extracted from data. We further used this case to illustrate the TTLC prediction result in Fig. 4.5. The differences between our resulted mean and the ground truth are all smaller than 0.3s within three seconds before lane change; besides, the predicted TTLC values for other insertion areas remain in reasonable ranges.

Since the reference vehicle will switch from one to another while the predicted vehicle is driving, we need to guarantee the capability of our method to handle such cases without

[1]The full video can be found on `https://www.youtube.com/watch?v=6A3Hl-mRhbI`.

large discontinuity on the prediction result. Therefore, we examined one of such cases shown in Fig. 4.4(b) and it can be observed that such sudden change occurs between frame 19 and 20. During this period, our sampled points are able to keep in the correct DIA and tightly distributed around the red target line.

Intention Estimation

The ROC curves of the SIMP and the SVM methods are visualized in Fig. 4.6. The curves were created by plotting the *true positive rate* (TPR) against the *false positive rate* (FPR) at various threshold settings. Similar to [60], we defined two positive classes (lane change left and right) and one negative class (lane keeping). The area under the ROC curve (AUC) can be used as an aggregated measure of the classifier performance. The true positive (TP) represents correct prediction of either lane change left or right, the false positive (FP) indicates mispredicting the lane change direction, and the false negative (FN) means incorrectly predicting a lane change into lane keeping.

From Fig. 4.6 and AUC values, we observe that our method outperforms SVM for lane change maneuvers. A threshold of 0.3 for classification was chosen for making the best trade-off between a high TPR and a low FPR. Given the selected threshold, we can further calculate the *precision* and *recall* as

$$precision = \frac{TP}{TP+FP}, \quad recall = \frac{TP}{TP+FN} \tag{4.8}$$

and the F1 score can be obtained by the formula

$$F1 = \frac{2 * precision * recall}{precision + recall}, \tag{4.9}$$

which denotes how good the classification abilities are. Moreover, how early the lane change can be recognized is also in the focus of our interests. Thus, we calculated the average prediction time from the testing data that were classified as true positive. The overall performance of the two methods are compared in Table 4.2. It is apparent from table that the proposed method has better performance than SVM in terms of both prediction accuracy and the average prediction time which describes how early the ground truth can be predicted.

Since our method can correctly forecast the predicted vehicle's intention approximately 2s in advance to the actual lane change according to Table 4.2, we further plotted the ROC curve and calculated the AUC for each dynamic insertions area (DIA) to examine how well can SIMP predict the final insertion region. The obtained AUC values for *Area1, Area2,* and *Area3* are all equal to 1, and *Area4* has a 0.994 AUC value. The result implies that the proposed method can not only detect the lane change direction but also the specific dynamic insertion area (DIA) with high accuracy for the selected time window.

Motion Prediction

The comparison results between QRF and the proposed method for two motion prediction tasks are shown in Fig. 4.7 and Fig. 4.8. The mean for QRF was obtained by calculating

Table 4.2: Performance Comparison

Method	Precision	Recall	F1-Score	Average Predict Time (s)
SVM	0.859	0.919	0.888	1.911
SIMPF	0.936	0.925	0.931	1.957

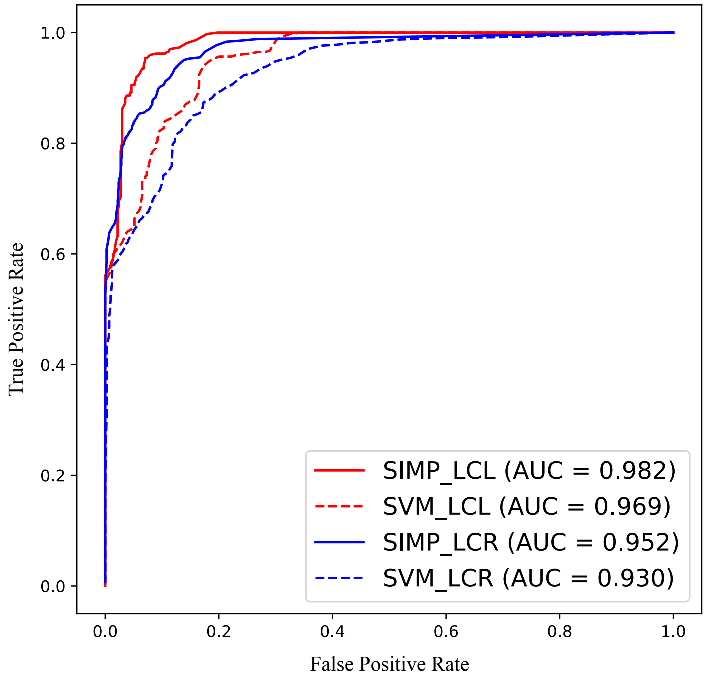


Figure 4.6: ROC curve comparison.

the 50% quantile (or median) assuming symmetric distribution. We utilized the testing data that has a TTLC smaller than the average prediction time derived in the previous section. The mean and confidence interval were calculated from the obtained output distribution of the correct insertion area. As can be seen in the plots, the RMSE of our approach for both motion predictions are smaller compared with the QRF method. The RMSE error of the TTLC tends towards zero for the lane change cases by using the SIMP method. One thing need to mention is that the error for the destination prediction is not close to zero even at $t = 0$. However, this is not unexpected given the fact that the predicted distance is relative to the reference car. Thus, the results might deviate due to the consideration of any velocity variance of the reference vehicle.

For the confidence interval comparison, it is obvious to see that the performance of our

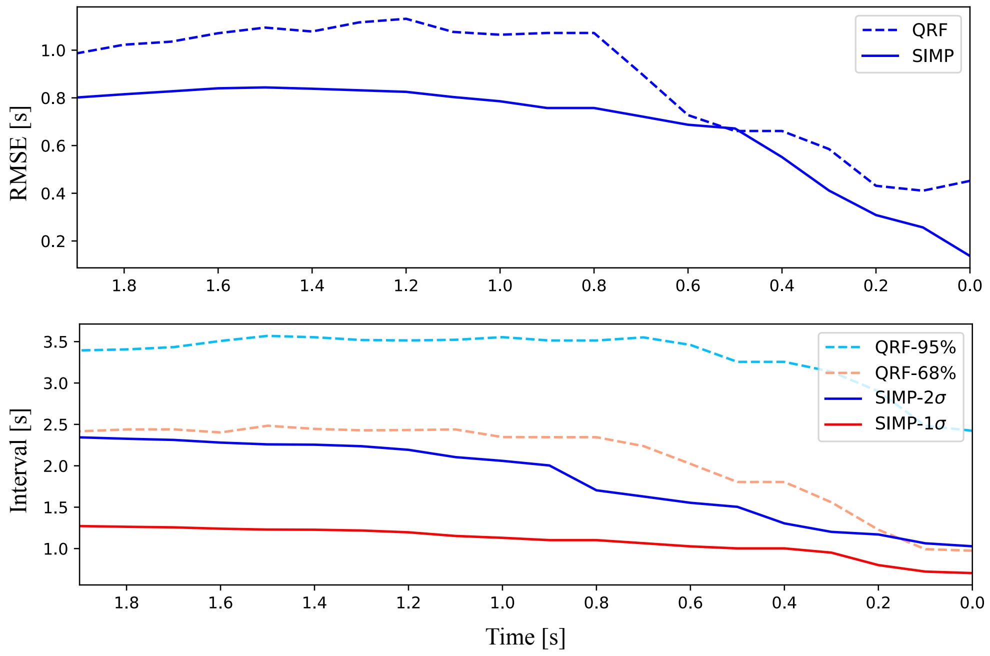


Figure 4.7: Comparison of Time-to-Lane-Change (TTLC) prediction.

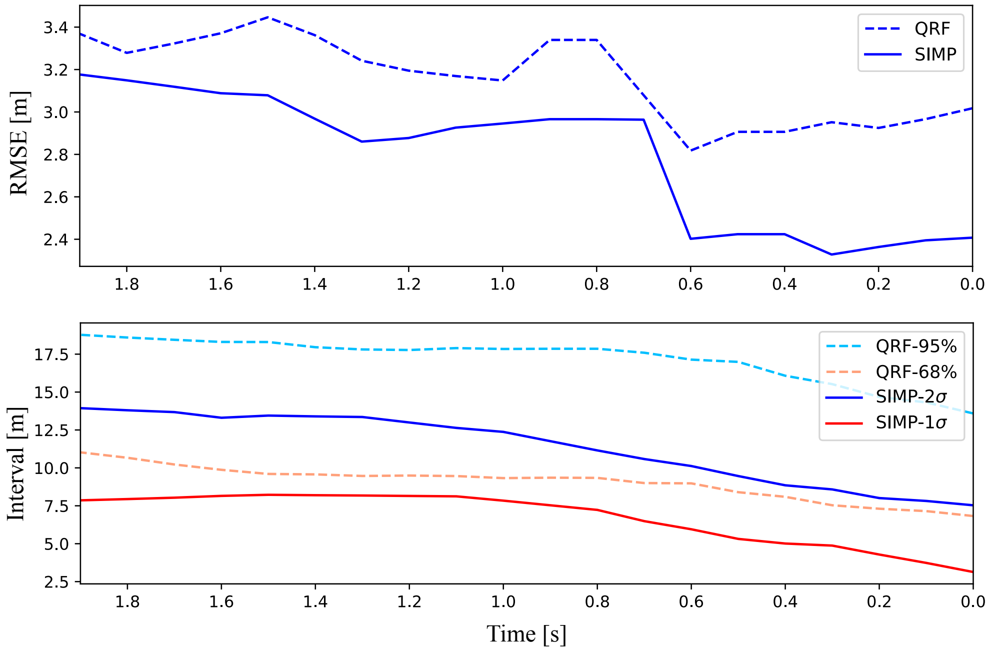


Figure 4.8: Comparison of destination prediction.

proposed method surpasses QRF especially for the TTLC prediction, where the 2-σ interval of the SIMP method is even smaller than the 68% interval of the QRF. The gradually decreasing difference between the one and two standard deviation interval as well as the declining interval values imply that our predicted Gaussian distribution is becoming more centralized around the ground truth as the TTLC approaching zero.

4.5 Conclusions

In this chapter, a Semantic-based Intention and Motion Prediction (SIMP) method was proposed, which can generate various designated conditional distributions for predicted vehicles under any circumstances. An exemplar highway scenario with real-world data was used to apply the idea of SIMP. First, two representative driving cases were utilized to visualize the testing result. Then the intention prediction and the motion prediction part were separately compared with two different baseline models: SVM and QRF. Our approach outperforms these methods in terms of both the prediction error and the confidence intervals. The key conclusion is that by combining different prediction tasks using semantics in a single framework, we can easily generalize the idea into any traffic scenarios and obtain competitive performance compared to traditional methods at the same time. The output goal position and time information can be further used to generate optimal trajectories for predicted vehicles and eventually obtain a desirable path for our own autonomous vehicle. For future work, we will examine the SIMP method on more complex scenarios as well as take into account the occurrence of vehicle occlusion.

Chapter 5

Traffic Scene Prediction Framework

5.1 Introduction

As the autonomous vehicle is becoming a big trend nowadays, safety is the most essential aspect to consider. Being able to predict future motions of the surrounding entities in the scene can greatly enhance the safety level of autonomous vehicles since potentially dangerous situations could be avoided in advance. Therefore, the Advanced Driver Assistance Systems (ADAS) is expected to have a full interpretation of the scene and be able to accurately predict possible behaviors of multiple traffic participants under various driving scenarios, which will then assure a safe, comfortable and cooperative driving experience. An illustration of the scene prediction is shown in Fig. 5.1.

There have been numerous works focusing on predicting the traffic participants in different driving scenarios. The predicted outcomes can be divided into two general categories : discrete and continuous. Intention estimation can be regarded as a discrete prediction problem, which is usually solved by classification strategies, such as Support Vector Machine (SVM) [47], Bayesian classifier [48], Hidden Markov Models (HMMs) [4], and Multilayer Perceptron (MLP) [49]. Motion prediction, on the other hand, is mostly treated as a re-

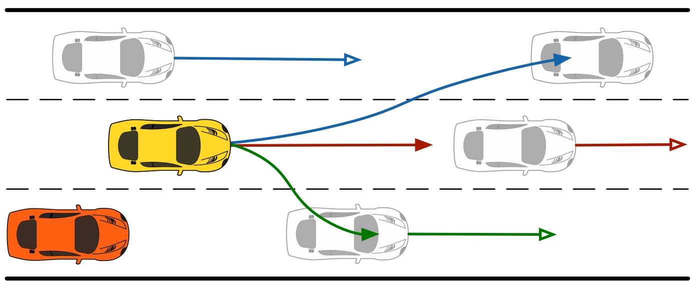


Figure 5.1: An illustration of the scene prediction results. The red car is the autonomous vehicle which is predicting the yellow car and its possible interaction with other scene entities.

gression problem, where it forecasts the short-term movements and long-term trajectories of vehicles. Various motion prediction methods use vehicle kinematic models at the prediction step and estimate the state recursively. For example, methods such as constant velocity (CV), constant acceleration (CA) and Intelligent Driver Model (IDM) [5] have been wildly used. However, these methods are generally considered in simple traffic scenarios such as car following.

There are also many approaches that deal with motion predictions in more complicated situations such as lane changing and ramp merging. [66] utilized Artificial Neural Network (ANN) structure to predict vehicle lateral motions and used a SVM to further determine if a lane change will happen. A Variational Gaussian Mixture Model (VGMM) is proposed in [57] to classify and predict the long-term trajectories of vehicles in a simulated environment. The main limitation of these works, however, is that they failed to take surrounding vehicles into account, which is unreasonable since the trajectory of the predicted vehicle will be largely influenced by the environment.

To handle such problems, several works incorporated potentially affecting vehicles by making use of relational features. In [55], the authors used the long short-term memory (LSTM) to predict the most likely trajectories for vehicles in highway situation while considering their nine surrounding vehicles. [56] brought forward a Deep Neural Networks (DNN) to obtain the lateral acceleration and longitudinal velocity while taking into account five vehicles around the predicted car. Nevertheless, these methods only predict trajectories for a single scene entity without estimating the possible motions of other vehicles. Works such as [19] and [18] utilized a LSTM-based structure for each vehicle in the scene and predicted the probabilistic information on future locations over a occupancy grid map. Although possible trajectories for every surrounding vehicles were predicted, the authors treated each vehicle independently during the prediction process, which cannot provide sufficient and accurate predictions especially in highly interactive scenarios.

Very few studies have been done for simultaneously predicting multiple interacting traffic participants. The approaches of [67] and [52] do incorporate such interdependencies by jointly predicting behavior patterns of all on-road vehicles. However, acquiring only the discrete behavior pattern is not enough for autonomous vehicles to fully predict the traffic scene or to directly perform risk assessment.

All the aforementioned motion prediction works did not have an estimation for the prediction horizon. They either fixed the prediction length beforehand or recursively estimated the vehicle state until a designated location is reached. However, for a given scene, multiple types of interaction between vehicles are possible and each of them is expected to have different time span. Therefore, it is not only irrational but also computationally expensive to predict every vehicle trajectories for the same time horizon.

In this chapter, a probabilistic framework that is able to predict various types of dynamic scenes is proposed. It contains an upper module and a lower module, where the Semantic-based Intention and Motion Prediction (SIMP) method is used in the upper module and the Conditional Variational Autoencoder (CVAE) is used in the lower module. The upper module is capable of predicting possible semantic intention and motion of the selected vehicle, while the lower module can further predict joint probability distributions of motions for interacting traffic participants. Possible future trajectories will be sampled from each joint distribution and the prediction horizon for different interacting entities can be received from the upper module. Also, our framework can guarantee feasibilities for every sampled trajectory. These trajectories could then be easily used by the decision-making and motion planning process for autonomous vehicles.

The remainder of the chapter is organized as follows: Section 5.2 provides the detailed explanation of the proposed scene prediction framework; Section 5.3 discusses an exemplar scenario to apply our framework; evaluations and results are provided in Section 5.4; and Section 5.5 concludes the chapter.

5.2 Scene Prediction Framework

In this section, we first introduce the method applied to each of the two modules in the scene prediction framework. Then, the overall prediction process of the framework is illustrated.

5.2.1 Upper Module

For the upper module, we implemented the Semantic based Intention and Motion Prediction (SIMP) approach [9] due to its great adaptability for various scenarios and competitive prediction performance compared to other methods. It utilizes deep neural network to formulate a probabilistic framework which can predict the possible semantic intention and motion of the selected vehicle under various traffic scenes. The introduced semantics is defined as answering the question of *"Which area will the predicted vehicle most likely insert into? Where and when?"*. The inserted area is called Dynamic Insertion Area (DIA), which can be a available gap between any two vehicles on the road or can be a lane entrance/exit area. Each DIA is assigned to a 2D Gaussian Mixture Model (GMM) to predict a two dimensional vector: $\boldsymbol{y} = [y_s, y_t]^T$, where y_s describes vehicle's location and y_t represents the time information. Note that both variables can be explicitly defined according to the problem formulation.

Therefore, given a set of input features $\boldsymbol{x}$, the probability distribution $\boldsymbol{y}_a$ over a single area a for the predicted vehicle can be expressed as:

$$p(\boldsymbol{y}_a|\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m \frac{1}{2\pi\sqrt{|\Sigma_m|}} exp\left(-\frac{D_m^T \Sigma_m^{-1} D_m}{2}\right), \tag{5.1}$$

where $D_m = \boldsymbol{y}_a - \mu_m$ and M denotes the total number of mixture components. For each mixture component m, the mixing coefficient α_m, mean μ_m, and covariance Σ_m formulate a probability density function of the output $\boldsymbol{y}_a$.

The output of the SIMP structure contains the required parameters for every 2D GMM and the weight w_a for each insertion area a. As for the desired outputs, not only the largest weight is expected to be associated to the actual inserted area, but also the highest probability is supposed to be at the proper location and time for the output distributions of that area. The loss function is then defined as

$$Loss = W_1\Big(-\sum_n log\Big\{\sum_{a=1}^{N_a} \hat{w}_a^n p(\boldsymbol{y}_a^n|\boldsymbol{x})\Big\}\Big) + W_2\Big(-\sum_n \sum_{a=1}^{N_a} \hat{w}_a^n log(w_a^n)\Big), \tag{5.2}$$

where N_a denotes the total number of DIA in the scene and $\hat{w}_a$ denotes the ground truth of the area weight, which is the one-hot-encoding of the final insertion area for the predicted vehicle. Parameters W1 and W2 are manually tuned such that the two loss components will have the same order of magnitude during training.

5.2.2 Lower Module

The lower module contains different motion models which will be assigned to each insertion area according to some criteria that will be discussed in Section 5.3. Here, we applied the Conditional Variational Autoencoder (CVAE) [43][68] method to each motion model.

CVAE has a similar structure to the typical variational autoencoder which contains an encoder and a decoder. It is rooted in bayesian inference, where the goal is to model the underlying probability distribution of the data so that new data could be sampled from that distribution. The overall structure of the CVAE we used is shown in Fig. 5.2.

In order to obtain the distribution of the output Y given the input X (i.e. $P(Y|X)$), a latent variable $z \sim \mathcal{N}(0, I)$ is introduced such that

$$P(Y|X) = \mathcal{N}(f(z, X), \sigma^2 \cdot I), \tag{5.3}$$

where it has mean f which is a function that can be directly learned from the data and

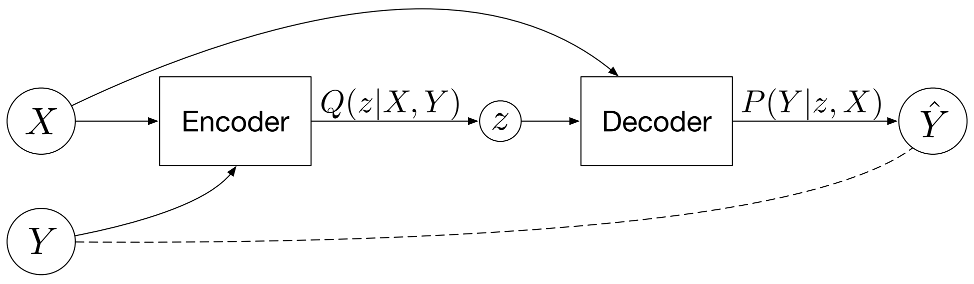


Figure 5.2: CVAE structure.

covariance equal to the identity matrix I times a scalar σ.

Given the training data (X, Y), the framework first samples z from some arbitrary distribution Q different from $\mathcal{N}(0, I)$. According to Bayes rule, we have:

$$E_{z\sim Q}\big[\log P(Y|z,X)\big] = E_{z\sim Q}\big[\log P(z|Y,X) + \log P(Y|X) - \log P(z|X)\big] \tag{5.4}$$

and by subtracting $E_{z\sim Q}(\log Q(z))$ from both sides:

$$\begin{aligned}\log P(Y|X) - E_{z\sim Q}\big[\log Q(z) - \log P(z|X,Y)\big] &= \\ E_{z\sim Q}\big[\log P(Y|z,X) + \log P(z|X) - \log Q(z)\big]&.\end{aligned} \tag{5.5}$$

To make the right hand side closely approximates $\log P(Y|X)$, Q is constructed to depend on both X and Y, which will make $E_{z\sim Q}\big[\log Q(z) - \log P(z|X,Y)\big]$ small.

By writing the above expectation as Kullback-Leibler (KL) divergences, our variational objective is in the form:

$$\begin{aligned}\log P(Y|X) - D_{KL}\big[Q(z|X,Y)||P(z|X,Y)\big] &= \\ E_{z\sim Q}\big[\log P(Y|z,X) - D_{KL}\big[Q(z|X,Y)||P(z|X)\big]\big]&,\end{aligned} \tag{5.6}$$

where $P(z|X) = \mathcal{N}(0, I)$ since the model assumes that z is independent of X if Y is unknown. The right hand side can be optimized via stochastic gradient descent by using the reparameterization trick to enable the encoder to generate a vector of means and a vector of standard deviation. In general, Q is trained to "encode" Y into the latent z space such that the values of z can be "decoded" back to the output.

The loss function then becomes a summation of the generative loss, which is the mean square error between the network output and the ground truth, and a latent loss, which is the KL divergence term that forces the latent variables match a unit Gaussian:

$$Loss = ||Y - \hat{Y}||^2 + D_{KL}\big[Q(z|X,Y)||\mathcal{N}(0,I)\big], \tag{5.7}$$

where Y is the ground truth and $\hat{Y}$ is the output estimation. At test time, we can sample from the distribution $P(Y|X)$ by sampling z directly from $\mathcal{N}(0, I)$.

5.2.3 The Overall Framework

For a selected vehicle, its input feature vector $\boldsymbol{x}$ contains all the information of itself and its surrounding environment. The SIMP method takes $\boldsymbol{x}$ as the input and generates three types of outputs: $\boldsymbol{w}$, $p(y_s|\boldsymbol{x})$, and $p(y_t|\boldsymbol{x})$. The vector $\boldsymbol{w}$ represents the weight for each of the N_a areas; $p(y_s|\boldsymbol{x})$ denotes the probabilistic distribution of the final destination within each DIA; and $p(y_t|\boldsymbol{x})$ denotes the probabilistic distribution of the time remained to enter each DIA for the predicted vehicle, which can be also interpreted as the time-to-lane-change (TTLC) distribution. These outputs along with the filtered input features will then feed into different motion models inside the lower module. Finally, the joint probability distribution of motions over multiple interacting entities at the next time step can be obtained. At

Algorithm 2: Scene Prediction Framework

```
1  $N_s$ : total number of output samples
2  $f_s$ : data sampling rate
3  $\boldsymbol{x}$  : input vectors at current time
4  $\boldsymbol{w}, p(y_s|\boldsymbol{x}), p(y_t|\boldsymbol{x}) \leftarrow SIMP(\boldsymbol{x})$
5  for $a = 1 : N_a$ do
6      $M \leftarrow$ current motion model
7      $n_a \leftarrow N_s \cdot w_a$
8      for $i = 1 : n_a$ do
9          $T \leftarrow sample(p(y_{t,a}|\boldsymbol{x}))$
10         $\boldsymbol{s}_1, \boldsymbol{o}_1 \leftarrow \boldsymbol{x}$
11         for $j = 1 : (T \cdot f_s)$ do
12             $T_j \leftarrow T - (j-1) \cdot f_s$
13             $\boldsymbol{a}_j \leftarrow sample(M(\boldsymbol{s}_j, \boldsymbol{o}_j, T_j))$
14             if $\boldsymbol{a}_j$ not feasible then
15                 redo iteration
16             else
17                 $\boldsymbol{s}_{j+1} \leftarrow f(\boldsymbol{a}_j, \boldsymbol{s}_j)$
18             end
19         end
20         if $p(y_{s,a} = \boldsymbol{s}_{final}|\boldsymbol{x}) < \epsilon$ then
21             redo iteration
22         end
23     end
24 end
```

the current time step t, we denote states of the interacting vehicles as $\boldsymbol{s}_t$, the other sensor observations of the surroundings as $\boldsymbol{o}_t$, and the predicted actions as $\boldsymbol{a}_t$. The next state of the interacting vehicles can be directly calculated using some mapping function f such that $\boldsymbol{s}_{t+1} = f(\boldsymbol{a}_t, \boldsymbol{s}_t)$.

Therefore, given the current state information and according to the first-order Markov assumption, the joint probability distribution over the prediction horizon T can be expressed as:

$$p(\boldsymbol{s}_{t+1}, \boldsymbol{s}_{t+2}, ..., \boldsymbol{s}_{t+T}|\boldsymbol{s}_t) = p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t)p(\boldsymbol{s}_{t+2}|\boldsymbol{s}_{t+1}) \cdot \dots \cdot p(\boldsymbol{s}_{t+T}|\boldsymbol{s}_{t+T-1}), \tag{5.8}$$

where T has the distribution of $p(y_t|\boldsymbol{x})$. Since $\boldsymbol{s}_t$ is independent of $\boldsymbol{o}_t$ and T, we have:

$$p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t) = \sum_T \sum_{\boldsymbol{o}_t} p(\boldsymbol{s}_{t+1}, \boldsymbol{o}_t, T|\boldsymbol{s}_t) = \sum_T \sum_{\boldsymbol{o}_t} p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{o}_t, T)p(\boldsymbol{s}_t), \tag{5.9}$$

where $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{o}_t, T)$ is obtained from the output distribution of the motion prediction model and $\boldsymbol{o_t}$ is assumed to have a gaussian measurement noise. Here, we use sample-based

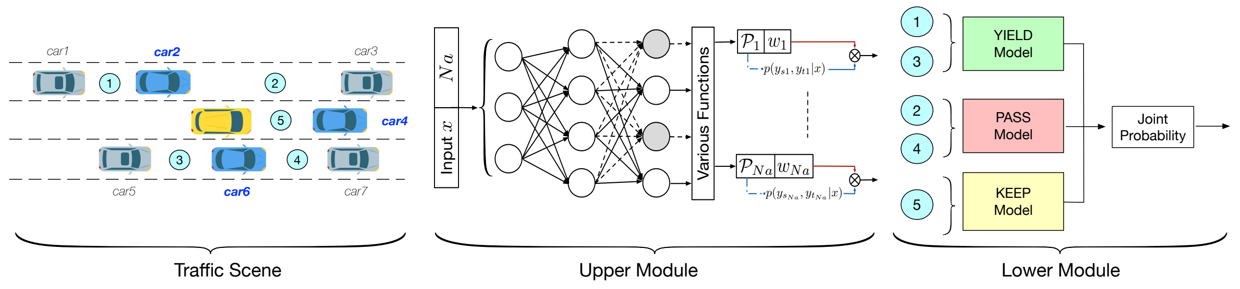


Figure 5.3: The scene prediction framework for the selected exemplar scenario.

method to infer the desired joint probability distribution and the overall process of the proposed scene prediction framework is illustrated in Algorithms 2.

Each motion model has its corresponding dynamic insertion area and the model will be triggered once its insertion weight w_a is greater than a threshold (i.e. n_a is not zero). Within each trajectory sampling process, we first sample the time information T, from $p(y_t|\boldsymbol{x})$. Then, within the given prediction horizon $T \cdot f_s$, we iteratively apply our lower motion model to obtain the sampled output action, which will go through a feasibility check to make sure the predicted next state is reachable by the vehicle. Finally, we use the destination distribution $p(y_s|\boldsymbol{x})$ from the SIMP method to ensure our predicted final state is within the desired range.

5.3 An Exemplar Scenario

In this section, we use an exemplar highway driving scenario to apply our proposed scene prediction framework. The data source and detailed problem formulation are presented.

5.3.1 Dataset

The data we used was taken from the NGSIM dataset which is publicly available online at [63]. We used the US highway 101 dataset which contains detailed vehicle trajectory data collected with sampling rate $f_s = 10$ Hz. Since the original data, especially for the velocity and acceleration, are very noisy, we used an Extended Kalman Filter (EKF) for better estimations. For lane-change situation, we picked up to 40 frames (4s) before the vehicle's center intersects the lane mark; for lane-keep situation, 40 frames were selected for each vehicle. A total of 15,240 frames were chosen from the dataset and randomly split into 80% for training and 20% for testing.

5.3.2 Scenarios and Problem Description

The representation of the exemplar scenario and the corresponding framework structure are shown in Fig. 5.3. The yellow car is the predicted vehicle; the blue cars are the reference vehicles, which are vehicles that the predicted vehicle will most likely interact with; the grey cars are other surrounding vehicles that are assumed to be fully observable.

In the typical driving scenario, there are five circled areas (DIA) that our predicted vehicle could eventually enter. More specifically, if the predicted vehicle (yellow car) inserts into $area1$ or $area3$, it will yield to car2 or car6 respectively; if the predicted vehicle enters $area2$ or $area4$, it will pass car2 or car6 respectively; if, however, $area5$ is inserted, the predicted vehicle will keep its lane and follow car4. Therefore, we considered three motion models inside the lower module: **yield**, **pass** and **keep**. Each model is trained separately on data that satisfy each particular model type. Note that for other traffic scenes such as ramp merging, the same type of motion models can be directly applied.

5.3.3 Features and Structure Details

The input features for the upper module are selected as same as in [9]. For the lower module, input states of the two interactive vehicles can be written as $\boldsymbol{s}_t = \{v^t_{pred}, v^t_{ref}, x^t_{ref} - x^t_{pred}, y^t_{ref} - y^t_{pred}\}$, where the subscript $pred$ denotes the predicted vehicle and ref denotes the reference vehicle, and the observed input $\boldsymbol{o}_t$ is determined by the type of the motion model. For example, $\boldsymbol{o}_t$ for the *pass* model contains states of the vehicle in front of the predicted and the reference car, whereas no $\boldsymbol{o}_t$ is needed for the *keep* model since only the front reference car need to be considered which information is already included in the input state. The predicted actions $\boldsymbol{a}_t$ for each motion model are expressed as $\boldsymbol{a}_t = \{\Delta x^t_{pred}, \Delta v^t_{pred}, \Delta x^t_{ref}, \Delta v^t_{ref}\}$, representing lateral displacements and the longitudinal velocity differences for both the predicted vehicle and the reference vehicle. Here, the aforementioned function f will map $\boldsymbol{a}_t$ and $\boldsymbol{s}_t$ linearly to $\boldsymbol{s}_{t+1}$.

For the SIMP structure, we use three fully connected layers of 400 neurons each, with $tanh$ non-linear activation functions. A dropout layer is appended to the end to enhance the network's generalization ability and prevent overfitting. For the CAVE structure, we utilize two fully connect layers of 128 neurons each for both the encoder and the decoder, and use three latent variables. The two modules are trained hierarchically and concatenated during testing.

5.4 Evaluation and Results

In this section, different evaluation methods are presented to assess the model quality and the final results are discussed.

5.4.1 Performance Evaluation

The output of the scene prediction framework are trajectories sampled from joint motion distributions of the predicted vehicle and each of its interacting entities (i.e. reference vehicles). We used the Root-Mean-Square Error (RMSE) as the validation metric to evaluate the

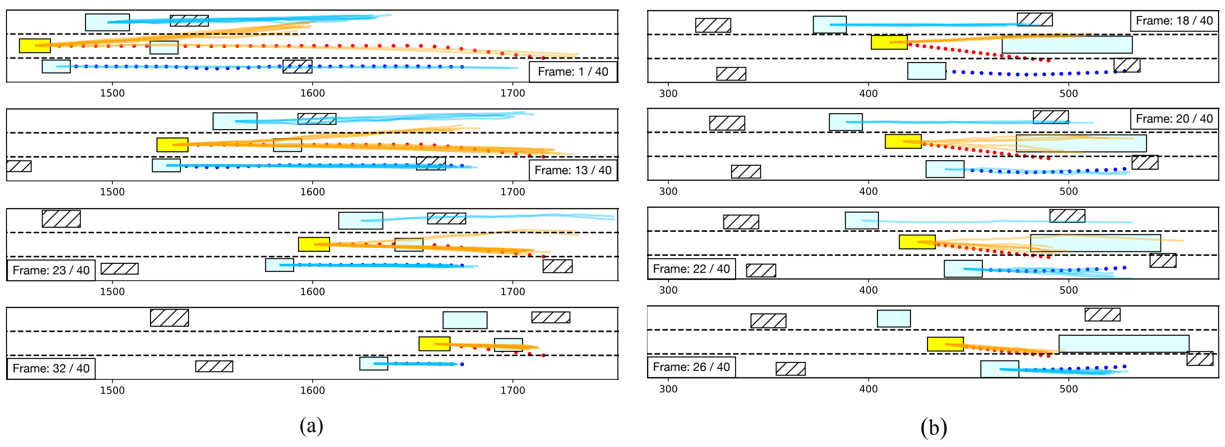


Figure 5.4: Test results of two example cases. The yellow car represents the predicted vehicle; the blue cars are reference vehicles that might interact with the predicted vehicle; and the striped cars denote other surrounding vehicles which will not have direct interactions with the predict vehicle. The red and blue dotted lines are the ground truth trajectories (with 0.1s step-time) for the predicted vehicle and its interacting reference vehicle respectively. The solid lines are sampled trajectories from the motion models.

output trajectories. However, multiple possible trajectories can be sampled from different motion models at the early stage of a lane changing scenario and more trajectories will be sampled from the ground-truth motion model as the time-to-lane-change value gets smaller. Therefore, in order to calculate the RMSE for a longer horizon, we analyzed our result by sampling trajectories directly from the actual motion model during the whole prediction period and compared them with the real trajectory.

For a selected test case, the ground-truth motion model is determined by the final insertion area and the RMSE can be calculates as:

$$RMSE = \sqrt{\frac{1}{N_s \cdot T \cdot f_s} \sum_{i=1}^{N_s} \sum_{j=1}^{T \cdot f_s} (\boldsymbol{s}_j - \hat{\boldsymbol{s}}_j^i)}, \tag{5.10}$$

where $\boldsymbol{s}_j$ is the true state value at time j and $\hat{\boldsymbol{s}}_j^i$ is the predicted value from the i-th sampled trajectory at the same time step j. We evaluated the RMSE of the lateral position and the longitudinal velocity separately for each motion model. Moreover, we selected the *pass* model to evaluate the performance of using and not using the time-to-lane-change information obtained from the upper module. The RMSE results across different motion models under various prediction horizons are presented in Table 5.1

As shown in Table 5.1, our method has lateral position errors within 0.5m for each model and a 1.3m/s error in velocity when prediction horizon is 4s. For the lateral position error, it can be clearly seen that the performance of the reference vehicle is better than that of the predicted vehicle. This is because during the lane changing period, the predicted car has larger lateral displacement than the reference car which usually remains its lateral position. When the *pass* model does not use the time information given by the SIMP method, the RMSE values increase especially for velocity errors. This happened because without the time constraint, the model may assume both vehicles will remain their speed and none of them will yield to the other at the beginning stage, which could cause cumulated speed deviations for both vehicles as well as a delay in the lateral displacement for the predict vehicle (yellow). Note that only a 4s prediction horizon is examined for the *keep* model since there won't have much motion changes within that period for car following cases.

5.4.2 Visualization of Selected Cases

We selected two distinct traffic situations to visualize our results as shown in Fig. 5.4. Each situation has a maximum prediction horizon of 4s (40 frames) and four representative frames are selected from each scenario to illustrate the performance of our framework. In order to better visualize the results, we did not add the trajectories generated by the *keep* motion model on the plot.

At the early stage of the situation in Fig.5.4(a), the predicted vehicle is likely to interact with its left reference car since the vehicle is closer to the area behind its left car than the area in front of its right car. Therefore, in the first frame, most trajectories are sampled from the *yield* motion model and the yielding pattern can be clearly recognized from the plot. As time goes, the predicted vehicle starts to increase its speed and finally passes its right reference vehicle. For case in Fig.5.4(b), the predicted vehicle changes its mind by first having a large intention to pass its left vehicle and then choosing to yield to its right reference vehicle. Note that both cases have some intermediate stages where multiple possible interaction exist and thus it is necessary to predict all these potential joint trajectories to have a full understanding of the scene.

5.4.3 Framework Robustness Test

Although our framework shows great performances during the evaluation, it is possible that some corner cases are not included in the test data. Therefore, to examine whether our method can generate reasonable results under various situations, we manually changed some parameter settings to have an comprehensive analysis. We used two methods to evaluate the framework robustness:

- Fix the current input scenario and change motion models in the lower module.
- Fix the motion model in the lower module and change the time-to-lane-change (TTLC) distribution obtained from the upper module.

For the first evaluation, we chose the 10th frame of the test scenario in Fig. 5.4 (a), where the predicted car has a similar longitudinal position as its right reference car and the actual

Table 5.1: RMSE Evaluation Results

		Prediction horizon					
Model	Vehicle	4s	3s	2s	1.5s	1s	0.5s
Keep	*pred*	0.30	-	-	-	-	-
	ref	0.28	-	-	-	-	-
Yield	*pred*	0.51	0.48	0.39	0.21	0.14	0.08
	ref	0.42	0.36	0.25	0.18	0.11	0.07
Pass	*pred*	0.43	0.41	0.37	0.21	0.19	0.07
	ref	0.41	0.33	0.22	0.14	0.04	0.02
Pass (w/o time)	*pred*	1.10	0.77	0.67	0.51	0.28	0.12
	ref	0.34	0.30	0.29	0.18	0.07	0.03

(a) RMSE for Lateral Position (m)

		Prediction horizon					
Model	Vehicle	4s	3s	2s	1.5s	1s	0.5s
Keep	*pred*	1.11	-	-	-	-	-
	ref	1.07	-	-	-	-	-
Yield	*pred*	1.24	1.19	0.91	0.79	0.70	0.31
	ref	1.25	1.05	0.72	0.67	0.33	0.22
Pass	*pred*	1.23	1.08	0.89	0.74	0.50	0.25
	ref	1.34	1.01	0.74	0.58	0.49	0.20
Pass (w/o time)	*pred*	2.59	2.25	1.97	1.41	0.90	0.79
	ref	2.17	2.06	1.32	1.08	0.64	0.43

(b) RMSE for Longitudinal Velocity (m/s)

motion model is *pass*. We then changed the motion model to *yield* and the comparison results of using different models are shown in Fig. 5.5. As can be seen in the plot, the *pass* model predicts that the predicted vehicle will first increase the speed to surpass the reference vehicle and then perform large lateral deviations for lane change. The *yield* model, however, indicates that the predicted vehicle will decrease its speed at the beginning and yield the reference car while changing its lane. Also, as the model is switched from *pass* to *yield*, the longitudinal velocity of the reference vehicle increases since the predicted car is

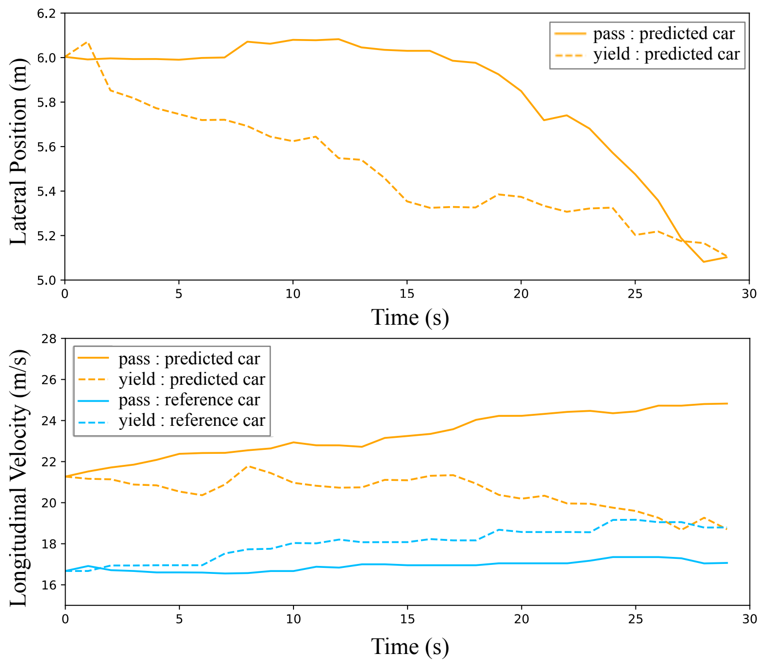


Figure 5.5: Performance comparison using different motion models.

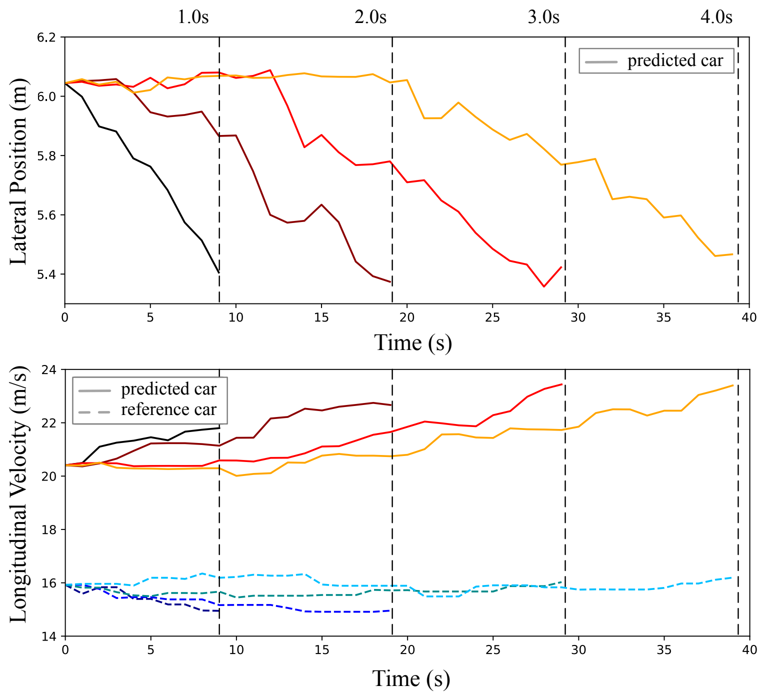


Figure 5.6: Performance comparison using different temporal information.

assumed not to insert in front of it due to the decreasing speed of the predicted car.

We used the same test scenario for the second evaluation but evaluated from the starting frame. By raising the predicted TTLC value from 1s to 4s, the predicted vehicle will have more time to change its lane. According to Fig. 5.6, when TTLC is 1s, the predicted car has a 0.6m lateral deviation within one second; when TTLC increases, the vehicle gradually changes its lane, which takes up to 2s to have the same amount of lateral deviation. The speed comparison also reasonably indicates that the vehicle will have a larger acceleration when it decides to change the lane faster.

5.5 Conclusions

In this chapter, a framework for probabilistic traffic scene prediction was proposed. It can simultaneously predict possible motions for multiple interacting traffic participants under various circumstances. An exemplar highway scenario with real-world data was used to demonstrate the performance of the framework. The proposed framework is not only able to generate accurate trajectories sampled from the predicted joint distributions of scene entities, but also has the adaptability to different driving situations. Note that we used the CVAE method to analyze the performance of the framework and it does not necessarily conclude that this is the best suited to the lower module. Other approaches that can generate conditional joint distributions are also able to use and we will compare different methods as well as examine the framework on other scenarios in our future works.

Chapter 6

Scenario-transferable Semantic Graph Reasoning for Prediction

6.1 Introduction

Prediction plays important roles in many fields such as economics [69], weather forecast [70], and human-robot interactions [71]. For intelligent robots such as autonomous vehicles, accurate behavioral prediction of their surrounding entities could help them evaluate their situations in advance and drive safely.

6.1.1 Challenges

One challenge of developing prediction algorithms is to find comprehensive and generic ***representations*** for common scenarios that can be encountered in the real world. In fact, finding suitable representations of the environment has been an open problem not only for prediction but also for decision making [72] and planning [73] tasks. If such generic representations can be found, another challenging problem arises as how to utilize these representations for predicting and imitating human behaviors while preserving explicit structures within these representations. Since prediction is a highly data-driven problem, deep learning methods are usually used for its strong capacity of learning and modeling complex relationships among predictor variables [74], [75]. However, they may not directly encode tractable or interpretable structure. Therefore, it is desired to take the advantage of deep learning models while retaining structural information of extracted generic representations using existing prior knowledge.

Moreover, human drivers are able to forecast the evolvement of driving environments regardless of whether they have encountered the same situations before, such as the identical road structure or traffic density. However, it is difficult for autonomous vehicles to possess such capability as human drivers when unforeseen circumstances are encountered. Therefore, for the autonomous vehicle, it is challenging to have a prediction algorithm that is both ***flexible*** and ***transferable***. Specifically, flexibility refers to the predictor's ability to

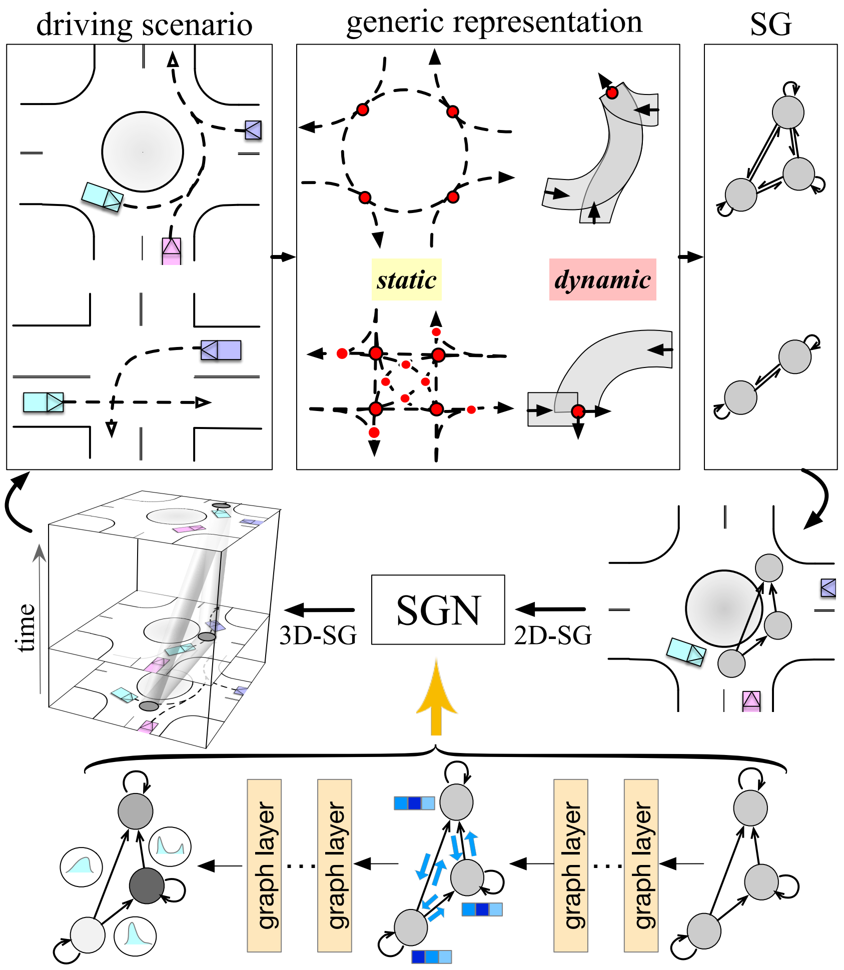


Figure 6.1: Illustration of the overall concept of this chapter. Given any driving scenario, we are able to extract its generic static and dynamic representations. We then introduce semantic graphs (SG) to support spatial-temporal structural relations within generic representations related to semantic goals. Finally, we utilize semantic graph network (SGN) to operate on semantic graphs and make predictions by reasoning internal structural relations of these graphs.

handle a time-varying number of homogeneous or even heterogeneous agents in a scenario. Transferability, on the other hand, refers to the degree to which the predictor can be generalized or transferred to various situations. It is expected that a predictor is transferable across unseen driving scenarios or domains and such transferability is extremely important for enabling autonomous vehicles to navigate in dynamically changing environment in real life.

6.1.2 Insights

Research efforts have been devoted to address the aforementioned challenges separately. However, these challenging factors are, indeed, highly correlated with each other and cannot be solved independently. For instance, if generic representations of the road entities can be found, it will become easier for the predictor to achieve flexibility. In addition, the flexibility of an algorithm in one scene should be maintained while it is transferred to another scene, which reflects the necessity of flexibility for a transferable predictor. Furthermore, transferability of an algorithm largely depends on whether its input and output can be generically represented under difference scenes. To the best of our knowledge, this is the first work that manages to tackle all these challenges simultaneously and merge them into a single behavioral predictor for autonomous vehicles.

6.1.3 Contributions

In this chapter, a scenario-transferable probabilistic prediction algorithm based on semantic graph reasoning is introduced. Several concepts are proposed and defined in this chapter such as dynamic insertion area (DIA), semantic goals, and semantic graphs (SG), which are building blocks for the semantic graph network (SGN) we designed to predict agents' behaviors. The key contributions of this work are as follows:

- Introducing generic representations for both static and dynamic elements in driving scenarios, which take into account Frenét frame coordinates, road topological elements, traffic regulations, as well as dynamic insertion areas (DIA) defined in this chapter.
- Utilizing generic representations to define semantic goals and incorporating them into the proposed semantic graphs (SG).
- Proposing the semantic graph network (SGN) to reason internal spatial-temporal structural relations of semantic graphs and make predictions.
- Examine the predictor's accuracy, flexibility, and transferability via real-world driving data from two highly interactive and complex scenarios: an eight-way roundabout and an unsignalized T-intersection with completely different road structures.

6.2 Related Works

In this section, we provide an overview of related works through four aspects and briefly discuss how each of them is addressed in this work.

6.2.1 Generic Representations of Driving Environments

Generic representations of driving environments can be regarded as invariant features across different driving scenarios or domains. Very few works have tried to find generic representations of driving environments. [76] applied affine transformation of pedestrians' trajectories into a uniform curbside coordinate frame. [77] utilized the Frenét coordinate frame along road reference paths to represent feature vectors of two interacting agents. In [78], self-centered image-based features were used as input to the network, where traffic regulations were encoded through images. [79] brought forward bird's eye representation of the scene surrounding the object, fusing various types of information on the scene which include satellite images and bounding boxes of other traffic participants.

These works either focus on extracting representations for a specific type of driving scenario (e.g. intersection [76], roundabout [77]) or applying end-to-end learning approaches to implicitly learn generic representations across different scenarios [78], [79]. A recent work from Waymo [80] proposed a vectorized representation to encode HD maps and agent dynamics. Although such vectorized representations are applicable to various driving scenarios, even some simple or obvious relations between road or agent vectors have to be learned by the network and there is no guarantee that those known relations can be learned correctly.

In fact, representations obtained from end-to-end deep learning models are with high abstraction level, which cannot be fully trusted and may fail under scenarios that are not well covered by the training data. Instead, in this work, we will take the advantage of our domain knowledge while constructing desired generic representations for various driving environments.

6.2.2 Behavior Prediction for Autonomous Vehicles

Many researchers have been focusing on probabilistic behavior prediction of autonomous vehicles and one of the most common objectives is to predict trajectories. Methods such as deep neural networks (DNN) [81], long short-term memory (LSTM) [82, 83], convolutional neural networks (CNN) [11, 84], generative adversarial network (GAN) [85], conditional variational autoencoder (CVAE) [86], and gaussian process (GP) [6] are typically utilized for trajectory prediction of intelligent agents. Moreover, inverse optimal control (or inverse reinforcement learning (IRL)) method has also been use for probabilistic reaction prediction under social interactions [7], [87].

Alternatively, there are also works focusing on predicting agent's goal state information directly, which contain both intention (e.g. left/right turn) and motion information (e.g. goal location and arrival time). For example, [88] obtained a probability distribution over all

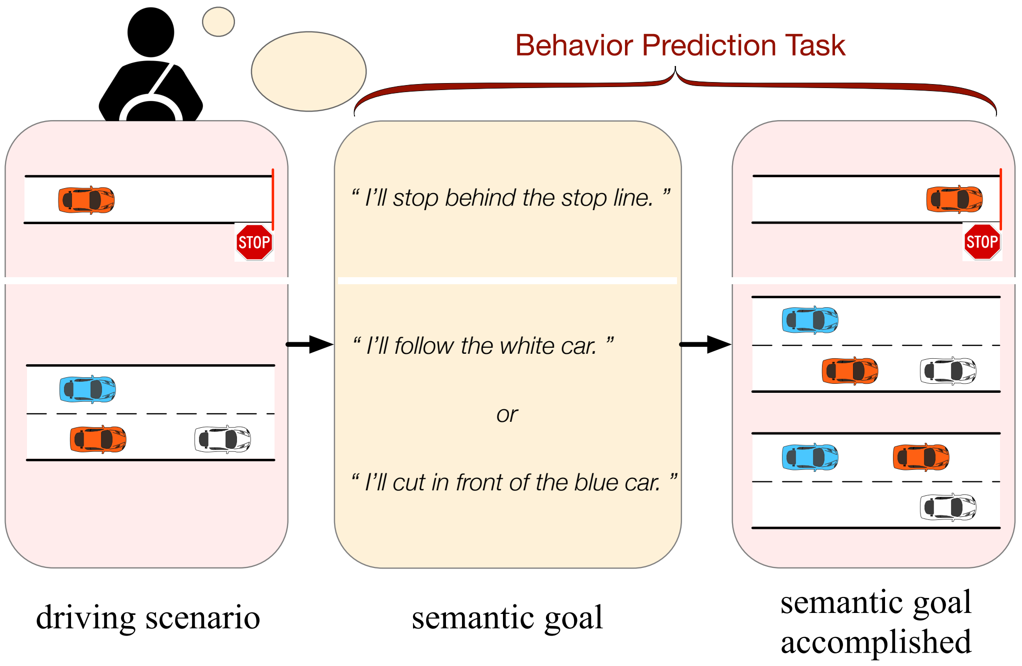

Behavior Prediction Task
STOP
" I'll stop behind the stop line. "
" I'll follow the white car. "
or
" I'll cut in front of the blue car. "
STOP
driving scenario
semantic goal
semantic goal accomplished

Figure 6.2: Illustration of the semantic goal concept as well as our behavior prediction task. The target vehicle is shown in red and our objective is to not only predict which semantic goal the driver is going to choose but also forecast corresponding vehicle's end state of achieving each semantic goal.

possible exit branches for a vehicle driving in a roundabout using recurrent neural network (RNN); [60] used LSTM to forecast time-to-lane-change (TTLC) of vehicles under highway scenarios; [89] proposed to use A*-based inverse planning to recognize the goals of vehicles on the road. In fact, by predicting goal states and assuming that agents navigate toward those goals by following some optimal or learned trajectories, the accuracy of prediction can be improved [58, 90]. For most goal-related prediction methods, the approach to selecting goals is either task-dependent such as left turn and lane change [91] or by sampling-based method along lane centerlines [92]. However, these potential goals are either too coerce to capture intra-category multimodality or based only on static map information without considering their dependencies with dynamically moving agents.

In this chapter, we introduce the concept of semantic goals based on the proposed generic representations of driving environments. Our objective is to predict the probability of each possible semantic goal and agent's behaviors when each goal is accomplished. Different from previous works, the proposed semantic goals can cover all possible driving situations by modeling various goals in the environment uniformly using semantics. As shown in Fig. 6.2, these semantic goals can not only describe static road information (e.g. stop line), but also represent dynamic relations among on-road agents (e.g. car following). It is worth mentioning that the predicted outcomes can be further used for goal-based trajectory estimation or planning tasks but it will not be the focus of this work.

6.2.3 Flexibility of Prediction Algorithms

In order to handle different input sizes (due to frequently changed number of surrounding agents) and achieve invariance to input ordering, Graph Neural Network (GNN) has been widely used recently as it processes strong relational inductive biases [93]. In general, graphs are a representation that supports pairwise relational structure and graph networks are neural networks that operate on graphs to structure their computations accordingly. In [94], the authors utilized graph to represent the interaction among all close objects around the autonomous vehicle and employed an encoder-decoder LSTM model to make predictions. Instead of treating every surrounding agent equally, the attention mechanism was applied to GNN in [95], where the proposed graph attention network (GAT) can implicitly focus on the most relevant parts of the input (i.e. specify different weights to neighboring agents) to make decisions. Works such as [96], [97], and [98] applied such a method to predict future states of multiple agents while considering their mutual relations.

Inspired by these works, we design a semantic graph network (SGN) which takes the advantage of the inductive biases in the graph network structure and operates on semantic graphs (SG). The proposed semantic graph incorporates generic representations of the environment related to semantic goals and its internal spatial-temporal structural relations will be reasoned by the network.

6.2.4 Transferability of Prediction Algorithms

Researchers have developed various machine learning algorithms to enhance the accuracy of behavior prediction tasks for autonomous vehicles under one or more driving scenarios such as highway (e.g. [82], [9], and [99]), intersection (e.g. [81], [6], and [100]), and roundabout (e.g. [101] and [102]). Although these machine learning algorithms provide excellent prediction performance, a key assumption underlying the remarkable success is that the training and test data usually follow similar statistics. Otherwise, when test domains are unseen (e.g. different road structure from training domains) or Out-of-Distribution (OOD) [103], the resulting train-test domain shift will lead to significant degradation in prediction performance. Incorporating data from multiple training domains somehow alleviates this issue [104], however, this may not always be applicable as it can be overwhelming to collect data from all the domains, especially for the autonomous driving industry.

Therefore, it is important for the predictor to have zero-shot transferability or domain generalizability, where it can be robust to domain-shift without requiring access to any data from testing scenarios during training. In this work, we will demonstrate the zero-shot transferability of our prediction algorithm when limited training domains are available. This is mainly achieved by combining the proposed generic representations of driving environments with the SGN framework.

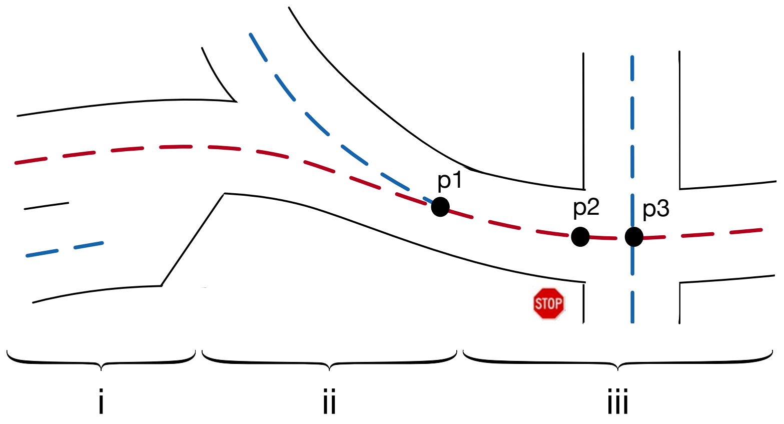


Figure 6.3: Illustration of reference paths (shown in dashed curves) and reference points (i.e. p_1, p_2, p_3) on one of the paths (red). The lower-case roman numerals split the scenario into three different sections which represent various road topological relations.

6.3 Generic Representation of the Static Environment

In order to design a prediction algorithm that can be used under different driving scenarios (e.g. highway, intersection, roundabout, etc.), we need a simple and generic representation of the static environment. The extracted expressions of the static environment should be able to describe road geometries and their interconnection as well as traffic regulations. We combine all these static environment information into road reference paths and the detailed methodology is described in this section.

6.3.1 Reference Path

A traffic-free reference path can be obtained either from road's centerline for constructed roads or by averaging human driving paths from collected data for unconstructed areas. The red and blue dashed lines in Fig. 6.3 denote different reference paths in the given scenario.

Reference point

In order to incorporate map information into the reference path, we introduce the concept of *reference points* which are selected points on the reference path. Reference points can either be **topological elements** that represent topological relations between two paths or **regulatory elements** that represent traffic regulations.

According to [105], the topological relationship between any of two reference paths can be decomposed into three basic topological elements: *point-overlap*, *line-overlap*, and *undecided-overlap*. In Fig. 6.3, segment (i) has two parallel reference paths and can be categorized as undecided-overlap case corresponding to lane change or overtaking scenarios, which has no fixed reference point; segment (ii) is a merging scenario that belongs to the line-overlap case with reference point $p1$; segment (iii) is an intersection and can be regarded as point-overlap scenario with $p3$ as the reference point.

Moreover, a traveling path on public road is normally guided by regulatory elements like *traffic lights* and *traffic signs*. Therefore, it is reasonable to incorporate these regulatory elements into each reference path, where we utilize the reference point to denote the location of each regulatory element. As an example shown in Fig. 6.3, the point $p2$ denotes the location of the stop line, which is one of the reference points on the red reference path.

Mathematical definition

We utilize the notation $\mathcal{X}_{ref}$ to represent the property of a reference path and each reference path is fitted by several way points through a polynomial curve and consists of various reference points. Therefore, we can mathematically define each reference path as $\mathcal{X}_{ref} = \{(x_k, y_k), (x_p, y_p)\}$, where $x_{(\cdot)}$ and $y_{(\cdot)}$ are global locations of each point on the reference path, k denotes the k-th way point, and p denotes the p-th reference point.

6.3.2 Representation in Frenét Frame

In this work, we utilize the Frenét Frame instead of Cartesian coordinate to represent the environment. The advantage of the Frenét Frame is that it can utilize any selected reference path as the reference coordinate, where road geometrical information can be implicitly incorporated into the data without increasing feature dimensions. Specifically, given a vehicle moving on a reference path, we are able to convert its state from Cartesian coordinate $(x(t), y(t))$ into the longitudinal position $s(t)$ along the path, and lateral deviation $d(t)$ to the path. Note that the origin of the reference path is defined differently according to different objectives and each reference path will have its own Frenét Frame.

6.4 Generic Representation of the Dynamic Environment

Based on the generic representation of the static environment defined in Section 6.3, we further design a uniform representation of the dynamic environment that can cover all types of driving situations on the road. In this section, we first redefine the Dynamic Insertion Area (DIA) concept, originally introduced in [9], by providing comprehensive and mathematical definitions. We then thoroughly illustrate how the dynamic environment can be generically described by utilizing DIAs.

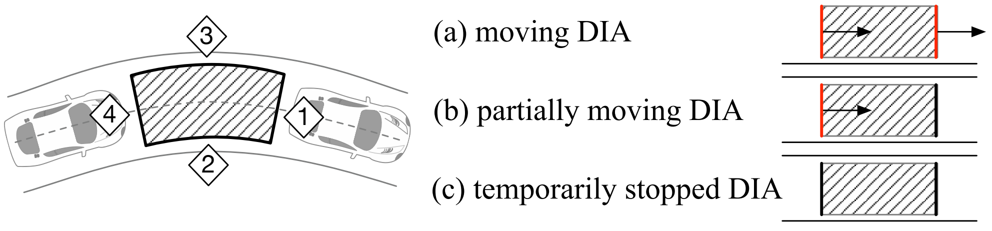


Figure 6.4: Basic properties for dynamic insertion area.

6.4.1 Definition of DIA

General descriptions

A dynamic insertion area (DIA) is semantically defined as: *a dynamic area that can be inserted or entered by agents on the road.* An area is called dynamic when both its shape and location can change with time. As can be seen in Fig. 6.4, each dynamic insertion area contains four boundaries: a front and a rear boundary (i.e. 1 & 4), as well as two side boundaries (i.e. 2 & 3). The front and rear boundaries of a DIA are usually formulated by road entities[1], but the two boundaries can also be any obstacles or predefined bounds based on traffic rules and road geometry, which details will be discussed later. Since the two side boundaries for each DIA are formulated by connecting the front and rear boundary along road markings or curbs (as seen in Fig. 6.4), the shape of the area highly depends on the geometry of the reference path it is currently on.

Table 6.1: Features for the Dynamic Insertion Area

	Feature	*Description*
Area Spec	l	Length of the area along reference path.
	θ	Orientation of the area.
Front/Rear Boundary	$v_{f/r}$	Boundary's velocity in moving direction.
	$a_{f/r}$	Boundary's acceleration in moving direction.
	$d_{f/r}^{lon}$	Boundary's longitudinal distance to the active reference point.
	$d_{f/r}^{lat}$	Boundary's lateral deviation from the reference path.

Each DIA has three different states as listed on the right side of Fig. 6.4 and these states are categorized mainly by the motion of DIA's front and rear boundaries. For example, if both boundaries have non-zero speed, the corresponding DIA is called a moving DIA

[1]The DIA boundaries can be formulated by any types of road entities including vehicles, cyclists and pedestrians. However, in this work, we will focus on vehicles only.

(i.e. Fig. 6.4(a)). If only one of the boundaries has zero speed, the DIA is regarded as partially moving (i.e. Fig. 6.4(b)). If, instead, both boundaries have zero speed, the DIA is temporarily stopped (i.e. Fig. 6.4(c)). Note that, in this work, we do not consider the case where the DIA is permanently stationary such as parking areas, which violates the dynamic property of DIA.

Mathematical definition

We define each dynamic insertion area as $\mathcal{A} = (\mathcal{X}_f, \mathcal{X}_r, \mathcal{X}_{ref})$, where $\mathcal{X}_f$ and $\mathcal{X}_r$ represent the properties of the front and rear bound of the DIA respectively; $\mathcal{X}_{ref}$, defined in the previous section, denotes the information of $\mathcal{A}$'s reference path which is the path that the area is currently moving on. Specifically, $\mathcal{X}_f = (x_f, y_f, v_f, a_f)$ and $\mathcal{X}_r = (x_r, y_r, v_r, a_r)$, where x, y are the global locations of each boundary's center point, v denotes the velocity, and a denotes the acceleration. As the geometric properties of DIA's side boundaries can be described by $\mathcal{X}_{ref}$ and the states of each DIA are mainly depend on its front and rear boundaries, we do not consider the states of side boundaries in the definition of $\mathcal{A}$.

Selected features

As discussed in the previous section, we are able to utilize reference path $\mathcal{X}_{ref}$ as the reference coordinate under the Frenét Frame and thus the property of each dynamic insertion area $\mathcal{A}$ can also be converted to the Frenét Frame. We extract six higher-level features to represent each insertion area $\mathcal{A}$ from $(\mathcal{X}_f, \mathcal{X}_r, \mathcal{X}_{ref})$ under the Frenét Frame, which are listed in Table 6.1.

The length l of each DIA is measured along its corresponding reference path, which can be expressed as: $l = d_f^{lon} - d_r^{lon}$. Here, $d_{(\cdot)}^{lon}$ denotes boundary's distance to the active reference point rpt_{act} which is the point we select as the origin of the environment and might change with time. Note that for all DIAs in the scene that the predicted vehicle might reach, they will always share the same rpt_{act} at a given time step. The criteria of choosing the active reference point will be discussed in the next subsection. The orientation θ of each area $\mathcal{A}$ is defined as the angle of the tangential vector to the reference path $\mathcal{X}_{ref}$ at the area's center point on the reference path, where the vector is pointed towards $\mathcal{A}$'s moving direction. Here, θ is measured relative to the global Cartesian coordinate instead of the local Frenét Frame.

6.4.2 DIAs in Dynamic Environment

After introducing the basic concept of dynamic insertion area, we will first describe how to systematically extract DIAs in any given environments. We then illustrate how DIA can be combined with static environment information to generically represent different dynamic environments. Besides, we will examine the relationships amongst different DIAs when more than one DIA exists. As mentioned in Section 6.3, we are able to utilize two different aspects (i.e. topological and regulatory elements) to design a generic representation for the static

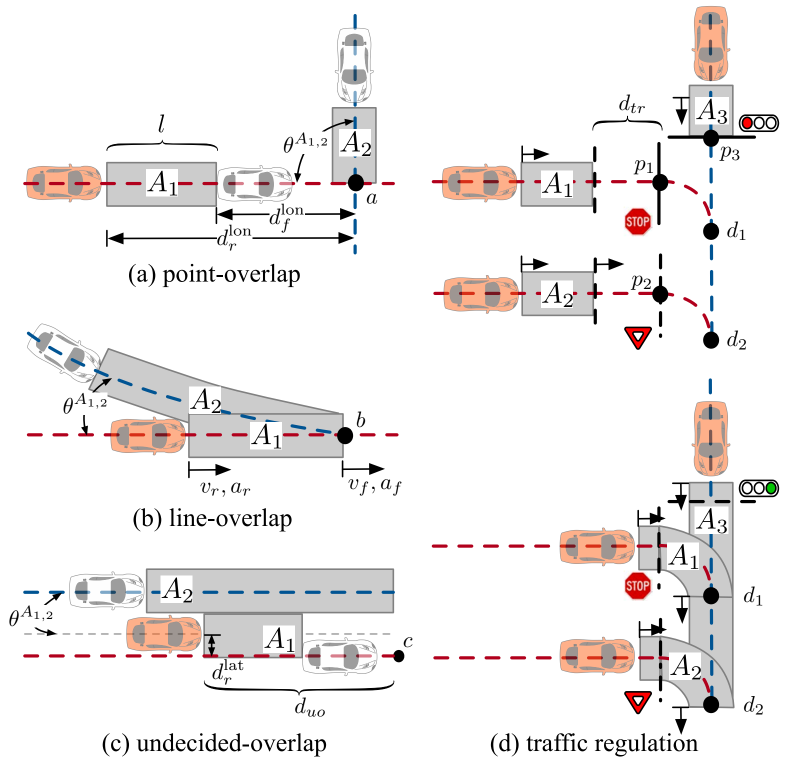


Figure 6.5: Demonstration of how DIA can be used to represent various driving environment.

environment. Therefore, we demonstrate the adaptability of DIA across several dynamical environments by varying the two aforementioned perspectives in the map (details shown in Fig. 6.5).

Algorithm of extracting DIAs

The entire algorithm of extracting DIAs in a scene, at any given time step, is described in Algorithm 3. The first step is to select the proper active reference point, the procedures of which are shown in Algorithm 3-(1). After obtaining the active reference point, we are able to extract all related DIAs in the environment by following the steps illustrated in Algorithm 3-(2). Since we are interested in the DIAs that the predicted vehicle might be inserted into, we only need to extract the DIAs within the observation range of the vehicle and we assume the predicted vehicle has full observation of its surroundings.

It is worth mentioning that it is possible for the predicted vehicle to have several possible reference paths when its high-level routing intention is ambiguous (e.g. the vehicle can either go straight or turn left/right at an intersection). In that case, Algorithm 3 needs to be operated under each potential reference path of the predicted vehicle. However, since predicting high-level routing intention is not the focus of our interests in this work, without loss of generality, we assume that the predicted vehicle's ground-truth reference path is known throughout the rest of this chapter.

Algorithm 3: Process of selecting the active reference point and extracting DIAs in a driving environment

```
For each predicted vehicle and the reference path $\mathcal{X}_{ref}$ it is moving on, do the following steps:
(1) Find the active reference point $rpt_{act}$ in the scene:
  $flag$ = False                                              ▷ $rpt_{act}$ is not found yet
  for $\forall rpt$ (in front of the predicted vehicle) $\in \mathcal{X}_{ref}$ do
      if $rpt \in$ regulatory elements then
          if $rpt \in$ traffic signs or $rpt \in$ red traffic light then
              $flag$ = True                                   ▷ $rpt$ is $rpt_{act}$
              break the loop
          end
      end
      if $rpt \in$ topological elements then
          if $\exists \mathcal{X}'_{ref}$ in the environment s.t. $\big((\mathcal{X}'_{ref} \cap \mathcal{X}_{ref} = rpt) \wedge (\exists car \text{ on } \mathcal{X}'_{ref})\big)$ then
              $flag$ = True                                   ▷ $rpt$ is $rpt_{act}$
              break the loop
          end
      end
  end
  if $flag$ == False then                                     ▷ no $rpt_{act}$ is found
      define $rpt_{act}$ as a point in front of the predicted vehicle along $\mathcal{X}_{ref}$, with distance $d_{uo}$
  end
(2) Find all DIAs in the scene up until $rpt_{act}$:
  for $\forall \mathcal{X}'_{ref}$ in the environment s.t. $\big((rpt_{act} \in \mathcal{X}'_{ref}) \vee (\mathcal{X}'_{ref}$ is parallel to $\mathcal{X}_{ref})\big)$ do
      if $\mathcal{X}'_{ref} == \mathcal{X}_{ref}$ then
          extract only the DIA in front of the predicted vehicle
      else
          extract all DIAs along $\mathcal{X}'_{ref}$
      end
  end
```

DIAs under different topological relations

We can use topological relations to represent the relationship between any two DIAs in a dynamic environment when they are moving on different reference paths, namely different $\mathcal{X}_{ref}$. The incorporations of DIAs under three basic topological elements are demonstrated as follows.

- **Point-overlap**: This corresponds to scenarios with crossing traffic such as intersections and an exemplar driving scenario is shown in Fig. 6.5(a). When we consider the red car as the predicted vehicle, point a is the active reference point in the scene according to the selection procedure in Algorithm 3. The two extracted DIAs are shaded in gray and their corresponding reference paths are represented in red (for $\mathcal{A}_1$) and blue (for $\mathcal{A}_2$) dashed lines. Hence, we denote the distances from the front bound and rear bound of $\mathcal{A}_1$ to a as d_f^{lon} and d_r^{lon}, respectively. The variable $\theta^{\mathcal{A}_{1,2}}$ denotes the relative angle between $\mathcal{A}_1$ and $\mathcal{A}_2$, where $\theta^{\mathcal{A}_{1,2}} = \theta^{\mathcal{A}_2} - \theta^{\mathcal{A}_1}$. In this example, both $\mathcal{A}_1$ and $\mathcal{A}_2$ are regarded as moving DIA. Here, we assign $\mathcal{A}_2$'s front boundary velocity $v_f^{\mathcal{A}_2}$ as the speed limit of its corresponding reference path (i.e. blue dashed line) and assume zero acceleration ($a_f^{\mathcal{A}_2} = 0$).

- **Line-overlap**: This corresponds to scenarios of merging and car following, where an exemplar case is shown in Fig. 6.5(b). In this situation, the front boundaries for $\mathcal{A}_1$ and $\mathcal{A}_2$ have some shared properties including a_f, d_f^{lon}, and d_f^{lat}. However, there front boundaries' velocities are not the same if their corresponding reference paths have different speed limits.

- **Undecided-overlap**: This corresponds to scenarios that do not have a fixed merging or demerging point such as lane change. As can be seen in Fig. 6.5(c), the two reference paths do not have a shared topological reference point that is fixed. For such situation, the active reference point in the scene is chosen to be the point, c, that has a pre-defined distance d_{uo} to the predicted vehicle. Here, the dynamic insertion area $\mathcal{A}_1$ is moving towards $\mathcal{A}_2$ with a moving direction vertical to $\mathcal{A}_1$'s reference path. Therefore, the lateral deviation, d_r^{lat}, of $\mathcal{A}_1$'s rear bound is no longer closer to zero as in the previous two scenarios. Note that in order to clearly represent each DIA on the map, the front boundary of $\mathcal{A}_1$ shown in Fig. 6.5(c) has the same lateral deviation as that of its rear boundary, however, the value of d_f^{lat} should be consistent with the actual lateral deviation of $\mathcal{A}_1$'s front boundary. Also, in this driving situation, the relative angle between two areas almost equals to zero (i.e. $\theta^{\mathcal{A}_{1,2}} \approx 0$).

DIAs under different traffic regulations

As there can be several different traffic regulations that guide objects to move on each reference path, we categorize them into two groups and illustrate the incorporation of DIAs under each of these regulations.

- **Traffic lights**: Traffic lights are usually positioned at road intersections, pedestrian crossings, and other locations to control traffic flows, which alternate the right of way accorded to road entities. If a reference path is guided by a traffic light, the reference point that represents such regulatory element is placed on the corresponding stop line. The signal color will affect the extraction of the dynamic insertion area. For example, when we select the vehicle moving in the vertical direction as the predicted vehicle and the light in front of it is red (see the top scenario in Fig. 6.5(d)), the active reference point is p_3. In such case, the stop line that p_3 is on is treated as the front boundary of $\mathcal{A}_3$, where $v_f^{\mathcal{A}_3} = 0$ and $\mathcal{A}_3$ is a partially moving DIA. When the light is green (see the bottom scenario in Fig. 6.5(d)) or yellow, the active reference point for $\mathcal{A}_3$ switches to d_1. Under such situation, $v_f^{\mathcal{A}_3}$ equals to the speed limit on the blue reference path and thus $\mathcal{A}_3$ is regarded as a moving DIA.

- **Traffic signs**: Traffic signs can be grouped into several types such as priority signs, prohibitory signs, and mandatory signs. In fact, sign groups that contain prohibitory and mandatory signs can be directly incorporated into the static environment representation by defining different reference paths. In this section, we only consider the sign groups that have influences on the dynamic environment. The sign group that is most commonly seen on the road is the groups of priority signs. Priority traffic signs include the stop and yield sign, which indicate the order in which vehicles should pass intersection points.

  When a vehicle is moving towards a stop sign, it will first decrease its speed before reaching the stop line and then slowly inching forward while paying attention to other lanes. In order to represent the differences between these two stages through DIA, we create a virtual stop line at a distance d_{tr} before the actual stop line. Fig. 6.5(d) illustrates this two-stage process, where the active reference point for the vehicle behind the stop sign changes from p_1 to d_1 and the front boundary of $\mathcal{A}_1$ moves from the virtual stop line to the line across d_1 (see the transition from the top to the bottom scenario in Fig. 6.5(d)). During the whole process, $\mathcal{A}_1$ transforms from a partially moving DIA into a moving DIA. Alternatively, if a yield sign is encountered, the vehicle will not necessarily decrease its speed unless it has to yield other vehicles on the main path. However, if the speed limit on the yield path is lower than that of on the main path, the two-stage process is also necessary. Such situation is illustrated in Fig. 6.5(d) where $\mathcal{A}_2$ remains as a moving DIA throughout the process. It is noteworthy that in Fig. 6.5(d), when we separately predict the two horizontally moving vehicles, the front boundary for $\mathcal{A}_3$ will vary due to different selection of the active reference point (i.e. when the vehicle behind the stop sign is predicted, the front boundary of $\mathcal{A}_3$ is at d_1; when the vehicle behind the yield sign is predicted, $\mathcal{A}_3$'s front boundary changes to d_2).

6.5 Semantic Graph Network (SGN)

In this section, we first state the prediction problem we aim to solve in this work. Then the concept of semantic graph (SG) is explained. Finally, we introduce the proposed semantic graph network (SGN) which can predict behaviors of interacting agents by reasoning their internal relations.

6.5.1 Problem Statement

As discussed in Section 6.2.2, we aim at directly predicting vehicle's behaviors related to its semantic goals. Specifically, we decide to use our proposed dynamic insertion area (DIA) as a uniform description of semantic goals and integrate it into the spatial-temporal semantic graph (SG) to construct a structural representation of the environment. In fact, as each DIA represents a semantic goal, we will use these two terminologies interchangeably in the rest of this chapter. It is worth to address that the reason we regard DIA as semantics is twofold: (1) DIA is defined by semantic description; (2) navigation-relevant semantic map information can be explicitly or implicitly included in DIAs.

Therefore, in this work, we would like to predict or answer the following questions: ***"Which DIA will the vehicle most likely insert into eventually? Where is the insertion location? When will the insertion take place?"***.

6.5.2 Semantic Graphs

The proposed semantic graph network (SGN) operates on semantic graphs (SG), where both its input and output are represented by graphs. There are two types of semantic graph in SGN: two-dimensional semantic graph (2D-SG) and three-dimensional semantic graph (3D-SG).

The 2D-SG is defined similar to the traditional graph [93] $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with node $n \in \mathcal{N}$ and edge $e = (n, n') \in \mathcal{E}$ which represents a directed edge from n to n'. For undirected edge , it can be modeled by explicitly assigning two directed edges in opposite directions between two nodes. The feature vector associated with node n_i at time step t is denoted as $\mathbf{x}_i^t$. The feature vector associated with edge $e_{ij} = (n_i, n_j)$ at time step t is denoted as $\mathbf{x}_{ij}^t$. Note that within a 2D-SG, only spatial relations are described since different nodes are connected using edges at the same time-step.

Alternatively, we define a 3D-SG as $\mathcal{G}^{t\to t'} = (\mathcal{N}^{t\to t'}, \mathcal{E}^{t\to t'})$, where $t \to t'$ denotes the time span from time step t to a future time step t' with $t' > t$. The graph $\mathcal{G}^{t\to t'}$ contains information that spans the entire period of scene evolution. The spatial and temporal relationship are jointly described by edges in 3D-SG, where the temporal relation between any of the two nodes in a 3D-SG can differ. We define node $n^\tau \in \mathcal{N}^{t\to t'}$ with $\{\tau \in \mathbb{R} | t \leq \tau \leq t'\}$ and edge $e^{t\to t'} = (n^t, (n')^{t'}) \in \mathcal{E}^{t\to t'}$. The feature vector associated with node n_i at time step t_i is denoted as $\mathbf{x}_i^{t_i}$. The feature vector associated with edge $e_{ij}^{t_i\to t_j} = (n_i^{t_i}, n_j^{t_j})$ that spans from

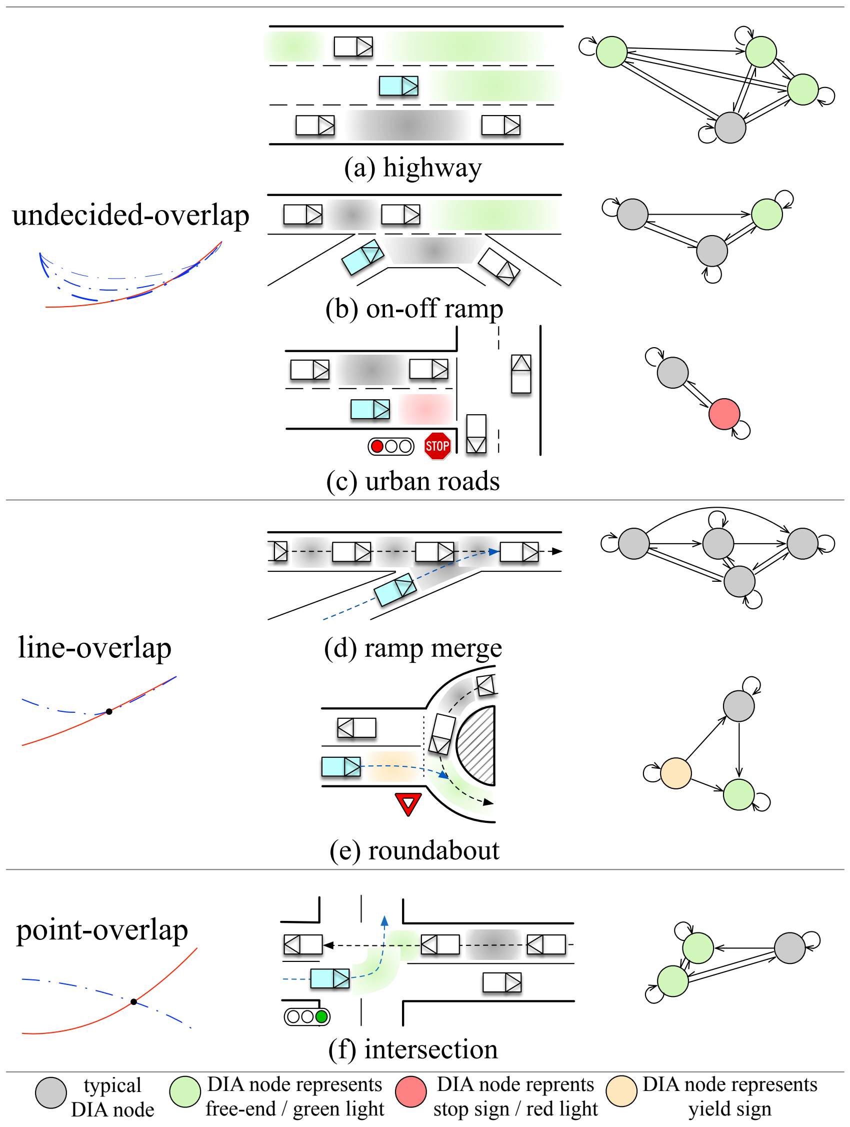


Figure 6.6: Illustration of various driving scenarios with extracted DIAs and corresponding 2D semantic graphs. The predicted vehicle is colored in cyan. Notice that all DIA nodes in the scene are defined uniformly and we color the DIAs for better interpretation of different driving situations.

t_i to t_j is denoted as $\mathbf{x}_{ij}^{t_i \to t_j}$. Note that when $t_i = t_j$, the spatial-temporal edge is the same as the spatial edge in 2D-SG (i.e. $e_{ij}^{t_i \to t_j} = e_{ij}$).

For driving scenarios, rather than assigning node attribute as individual entities (e.g. vehicles), we utilize DIA instead. Since DIA can not only describe dynamic environments but also inherently incorporate static map information, each node in the graph is able to represent semantic information in the environment. By defining node attributes as semantic objects like DIA, we are able to implicitly encode both static and dynamic information into the graph. Moreover, the egde attribute describes the relationship between any of two DIAs. For a 2D-SG, each edge may describe the strength of correlation between its corresponding two DIAs at the same time step; whereas for a 3D-SG, each edge may represent some future information of the two DIAs. For example, in the 3D-SG of scene in Fig. 6.5(c), the edge between $\mathcal{A}_1$ and $\mathcal{A}_2$ might encode the information of when and how two areas will merge together, which can be interpreted as when the red vehicle will cut in front of its left vehicle and at what location. Details on how various driving scenarios can be represented by semantic graphs are illustrated in Fig 6.6.

6.5.3 Network Architecture

The entire architecture of SGN is shown in Fig. 6.7 and each module is explained in details.

Input and output

For the proposed framework, the input can either be a 2D-SG at the current time step or a sequence of historical 2D-SGs. The output is a set of 3D-SGs that encompass the information of how the current scene will progress in the future, which could provide answers to our questions in Section 6.5.2.

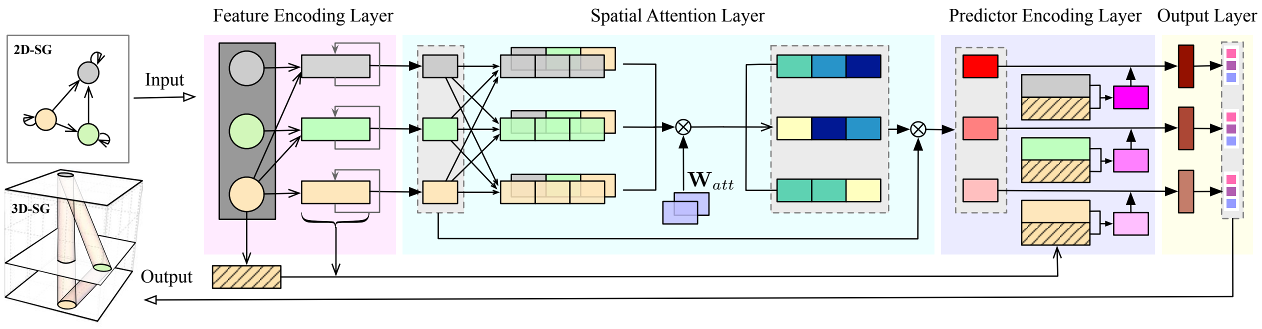


Figure 6.7: Semantic graph network (SGN).

Feature encoding layer

Since the state information of the predicted vehicle is implicitly contained in the DIA directly in front of it (i.e. the predicted vehicle forms the rear boundary of its front DIA), we can regard its front DIA as the reference DIA in the scene. Therefore, in a 2D-SG, if node i is selected as the reference DIA at the current time step t, we can then define the feature vector of some node n_j relative to node n_i as $\mathbf{x}^t_{j\to i}$, denoted as the relative node feature, and can be obtained by the following equation:

$$\mathbf{x}^t_{j\to i} = f_{j\to i}(\mathbf{x}^t_i, \mathbf{x}^t_j), \tag{6.1}$$

where $\mathbf{x}^t_i$ and $\mathbf{x}^t_j$ are absolute node features described in Section 6.4.1. We utilize a linear function $f_{j\to i}$ to map from absolute to relative node features. If we also have historical information of the node attribute, we can add a recurrent layer to further encode these sequential features:

$$\hat{\mathbf{h}}^t_{j\to i} = f^1_{rec}(\hat{\mathbf{h}}^{t-1}_{j\to i}, \mathbf{x}^t_{j\to i}), \tag{6.2}$$

where $\hat{\mathbf{h}}^t_{j\to i}$ is the hidden state also the output of the recurrent function f^1_{rec}. We choose the graph recurrent unit (GRU) [106] as our recurrent function, where for each node n_j the input to the GRU update is the previous hidden state $\hat{\mathbf{h}}^{t-1}_{j\to i}$ and the current input $\mathbf{x}^t_{j\to i}$.

Similarly, we encode feature sequences of the reference DIA by applying another recurrent function f^2_{rec}:

$$\hat{\mathbf{h}}^t_i = f^2_{rec}(\hat{\mathbf{h}}^{t-1}_i, \mathbf{x}_i). \tag{6.3}$$

Spatial attention layer

The task of the attention layer is to help with modeling the locality of interactions among DIAs and improve performance by determining which DIAs will share information. We first encode the embedded features from the previous layer to yield a fixed-length vector $\mathbf{h}^t_{j\to i}$:

$$\mathbf{h}^t_{j\to i} = f^1_{enc}(\hat{\mathbf{h}}^t_{j\to i}), \tag{6.4}$$

where f^1_{enc} denotes the encoder. We then compute attention coefficients $a^t_{(j\to i)(k\to i)}$ that indicate the importance of relative node feature $\mathbf{h}^t_{j\to i}$ to node feature $\mathbf{h}^t_{k\to i}$ as follows:

$$a^t_{jk} = f_{att}(concat(\mathbf{h}^t_{j\to i}, \mathbf{h}^t_{k\to i}); \mathbf{W}_{att}), \tag{6.5}$$

where we have simplified $a^t_{(j\to i)(k\to i)}$ as a^t_{jk} for readability. We denote $\mathbf{W}_{att}$ as the attention weight and f_{att} as a function that maps each concatenated two node intention features into a scalar. To make coefficients easily comparable across different node relations, we normalize them across all choices of j using the *softmax* function:

$$\alpha^t_{jk} = \frac{\exp(a^t_{jk})}{\sum_{k'\in\mathcal{N}^t_i} \exp(a^t_{jk'})}, \tag{6.6}$$

where α^t_{jk} denotes the normalized attention coefficient and $\mathcal{N}^t_i$ is a set of nodes that surrounds n_i in the graph at time step t. Finally, we derive the attention-weighted relative node feature $\bar{\mathbf{h}}^t_{j\to i}$, which is an encoded vector weighted by attention as:

$$\bar{\mathbf{h}}^t_{j\to i} = \sum_{k'\in\mathcal{N}^t_i} \alpha^t_{jk'} \odot \mathbf{h}^t_{k'\to i}, \tag{6.7}$$

where $\odot$ is the element-wise multiplication.

Predictor Encoding layer

For a given predicted vehicle, there will always be a DIA that is right in front of it and we regard this DIA as the reference DIA as stated previously. Therefore, if we want to infer the relations between the predicted vehicle and each of the DIAs on the road, we can alternatively infer the relations between the reference DIA and each of the other DIAs. Note that the predicted vehicle can also insert into the reference DIA (i.e. its front DIA), which might correspond to car-following in highway scenarios or yielding other cars in merging scenarios.

Therefore, we need to encode the relationship between any of the two nodes and make a prediction on their relations in the future. Such predicted relations will be reflected through the edges in the output 3D-SG. For any pair of nodes (i, j) that has connected edges in the input 2D-SG, we first concatenate their features to formulate the edge feature as either

$$\hat{\mathbf{h}}^t_{ij} = concat(\hat{\mathbf{h}}^t_{j\to i}, \hat{\mathbf{h}}^t_i) \quad \text{or} \quad e^t_{ij} = concat(\mathbf{x}^t_{j\to i}, \mathbf{x}^t_i), \tag{6.8}$$

depending on whether we have embedded historical node features or not. $\hat{\mathbf{h}}^t_{ij}$ denotes the hidden edge relation between node i and j over certain past horizon. We can then generate an embedded vector $\mathbf{h}^t_{ij}$ as follows:

$$\mathbf{h}^t_{ij} = f^2_{enc}(\hat{\mathbf{h}}^t_{ij}), \tag{6.9}$$

where the subscript $(\cdot)_{ij}$ denotes that i is the index of the reference node and j is the index of the node that connects to it. Different from the previous encoding function f^1_{enc} that outputs encoded information for each node, the function f^2_{enc} aims at encoding the edge information. After the edge encoding, we concatenate the result with the aggregated feature of the reference node and perform a decoding process f_{pred} to generate predicted edge information:

$$\mathbf{o}_{ij} = f_{pred}(concat(\bar{\mathbf{h}}^t_{j\to i}, \mathbf{h}^t_{ij})), \tag{6.10}$$

where $\mathbf{o}_{ij}$ denotes the encoded feature vector that will be later used to generate features for the 3D-SG.

6.5.4 Output Layer

In order to determine what elements should be generated by $\mathbf{o}_{ij}$, we first need to know what behavior we want to predict by revisiting our problem. Our task is to generate probabilistic distributions of the states for every possible insertion area in the input 2D-SG. In other word, we want to have a probabilistic distribution of states for every edge in the output 3D-SG. Without loss of generality, we assume the distribution can be described by a mixture of Gaussian. Therefore, we assign a Gaussian Mixture Model (GMM) to each 3D edge, where each Gaussian mixture models the probability distribution of a certain type of edge relation between its two connected nodes.

To infer the final insertion location of the predicted vehicle, we need to have at least two predicted variables: location of the inserted DIA and location of the vehicle in that DIA. Besides, since the time of insertion is also the focus of our interests, a 3D Gaussian mixture is used and the predicted variables are constructed as a three dimensional vector: $\mathbf{y} = [y_{s_1}, y_{s_2}, y_t]^T$. The variable y_{s_1} denotes the location of the inserted DIA, y_{s_2} denotes the location of the predicted vehicle relative to the DIA it enters, and y_t denotes the time left for the predicted vehicle to finish the insertion.

Predicting when and where the predicted vehicle will be inserted into a particular DIA associated with node j, is equivalent to predict edge relations between the predicted vehicle's front DIA (assuming it is associated with node i) and n_j. Hence, given the encoded edge feature vector $\mathbf{o}_{ij}^t$, the probability distribution of the output $\mathbf{y}_{ij}^{t_i \to t_j}$ over the edge $e_{ij}^{t_i \to t_j}$ is of the form $f(\mathbf{y}_{ij}^{t_i \to t_j}|\mathbf{o}_{ij})$. For brevity, we will eliminate the superscript of $\mathbf{y}_{ij}^{t_i \to t_j}$ for the rest of the chapter. Since we utilize the Gaussian kernel function to represent the probability density, we can rewrite $f(\mathbf{y}_{ij}|\mathbf{o}_{ij})$ as:

$$f(\mathbf{y}_{ij}|\mathbf{o}_{ij}) = f\big(\mathbf{y}_{ij}|f_{out}^1(\mathbf{o}_{ij})\big) = \sum_{m=1}^{M} \alpha_{ij}^m \mathcal{N}\big(\mathbf{y}_{ij}|\boldsymbol{\mu}_{ij}^m, \Sigma_{ij}^m\big), \tag{6.11}$$

where $\mathcal{N}\big(\mathbf{y}_{ij}|\boldsymbol{\mu}_{ij}^m, \Sigma_{ij}^m\big)$ can be expanded as:

$$\mathcal{N}\big(\mathbf{y}_{ij}|\boldsymbol{\mu}_{ij}^m, \Sigma_{ij}^m\big) = \frac{\exp(-\frac{1}{2}(\mathbf{y}_{ij} - \mu_{ij}^m)^T \Sigma^{-1} (\mathbf{y}_{ij} - \mu_{ij}^m))}{\sqrt{(2\pi)^d |\Sigma|}}, \tag{6.12}$$

where d denotes the output dimension which is three in this problem. In Eq. 6.11, M denotes the total number of mixture components and the parameter α_{ij}^m denotes the m-th mixing coefficient of the corresponding kernel function. The function f_{out}^1 maps input $\mathbf{o}_{ij}$ to the parameters of the GMM (i.e. mixing coefficient α, mean μ, and covariance Σ), which in turn gives a full probability density function of the output $\mathbf{y}_{ij}$. Specifically, the mean and covariance are constructed as:

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_{s_1} \\ \mu_{s_2} \\ \mu_t \end{bmatrix}, \Sigma = \begin{bmatrix} \sigma_{s_1}^2 & \sigma_{(s_1,s_2)} & \sigma_{(s_1,t)} \\ \sigma_{(s_2,s_1)} & \sigma_{s_2}^2 & \sigma_{(s_2,t)} \\ \sigma_{(t,s_1)} & \sigma_{(t,s_2)} & \sigma_t^2 \end{bmatrix}. \tag{6.13}$$

Besides predicting the state of final insertion in each DIA for the predicted vehicle, we also want to know the probability of inserting into each DIA observed in the scene. Therefore, given the encoded edge feature vector $\mathbf{o}_{ij}^t$, we further derive the insertion probability of node j's associated DIA as:

$$w_{ij} = \frac{1}{1 + exp(f_{out}^2(\mathbf{o}_{ij}^t))}, \tag{6.14}$$

which is the *logistic* function of the scalar output from function f_{out}^2. We also normalize the insertion probability such that $\sum_{k \in \mathcal{N}_i} w_{ik} = 1$.

Finally, we obtain the feature vector associated with each edge in a 3D-SG as: $\mathbf{x}_{ij}^{t_i \to t_j} = [\mathbf{y}_{ij}, w_{ij}]$. In the case where i is the reference node, t_i represents the current time of prediction and t_j is sampled from the distribution of the predicted insertion time variable y_t. By sampling from the predicted distribution of each edge in 3D-SG, we are able to formulate several 3D-SGs as possible outcomes of the scene evolution.

6.5.5 Loss Function

For the desired outputs, we expect not only the largest weight to be associated to the actual inserted area ($\mathcal{L}_{class}$), but also the highest probability at the correct location and time for the output distributions of that area ($\mathcal{L}_{regress}$). Consequently, we define our loss function as

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{regress} + \beta \mathcal{L}_{class} \\ &= -\sum_{\mathcal{G}_s} \sum_{i \in \mathcal{N}^s} \left(\log \left\{ \sum_{k \in \mathcal{N}_i^s} \hat{w}_{ik} f(\mathbf{y}_{ik} | \mathbf{o}_{ik}) \right\} - \beta \sum_{k \in \mathcal{N}_i^s} \hat{w}_{ik} \log(w_{ik}) \right),\end{aligned} \tag{6.15}$$

where $\mathcal{G}_s$ denotes the s-th 2D-SG, $\mathcal{N}^s$ denotes all the nodes in the current semantic graph, and $\mathcal{N}_i^s$ denotes the set of nodes surround n_i. Note that the number of nodes in set $\mathcal{N}^s$ and $\mathcal{N}_i^s$ is not fixed and will vary with time. The one-hot encoded ground-truth final inserting area is denoted by $\hat{w}_{ik}$. The hyperparameter β is used to control the balance between the two losses for better performance.

6.5.6 Design Details

In this work, we utilize feed-forward neural networks for all related functions described in Secton 6.5.3 (i.e. $f_{enc}^{1,2}$, f_{att}, f_{pred}, $f_{out}^{1,2}$), due to neural network's strong capacity of learning and modeling complex relationships between input and output variables. The function f_{out}^1 can thus be regarded as a GMM-based mixture density network (MDN) [62]. It is important to note that the parameters of the GMM need to satisfy specific conditions in order to be valid. For example, the mixing coefficients α^m should be positive and sum to 1 for all M, which can be satisfied by applying a *softmax* function. Also, the standard deviation for each output variable should be positive, which can be fulfilled by applying an exponential operator.

Moreover, we should notice that in Eq. 6.12, Σ is invertible only when it is a positive definite matrix. However, there is no guarantee that our formulated covariance matrix is non-singular. One solution to fix a singular covariance matrix is to create a new matrix $\hat{\Sigma} = \Sigma + kI$, where we want all the eigenvalues of the new covariance matrix be positive such that the matrix is invertible. Ideally, we prefer k to be a very small number so not to bias our original covariance matrix. At the meantime, since the eigenvalues of $\hat{\Sigma}^{-1}$ are the reciprocals of the originals, we want k to be large enough so that the eigenvalues of $\hat{\Sigma}^{-1}$ won't explode. Therefore, the best way of selecting k is hyperparameter tuning.

6.5.7 Inference for Semantic Prediction

At test time, we fit the trained model to observed historical 2D-SGs up until the current time step t and sample from the probabilistic density function $f(\mathbf{y}|\mathbf{o})$ to obtain a set of possible scene evolution outcomes. Although the network only output edge features of the 3D-SG, the node features are implicitly predicted as we know the spatial-temporal relations between any of two nodes. Hence, each sampled testing results can be formulated as a 3D-SG and we will thus obtain a set of 3D-SGs.

It should be pointed out that for a given 2D-SG, if the reference DIA is changed, we might end up obtaining different output 3D-SGs. This is reasonable since as we modify the reference node, we potentially alter the vehicle we want to predict. Therefore, under the perspective of distinct drivers, the scene will evolve into the future differently. On the other hand, if we assume vehicle-to-vehicle (V2V) communication, it is possible for all drivers on the road to reach a consensus on the future states of the scene.

6.6 Experiments on Real-World Scenarios

In this section, we evaluate the capability of the proposed algorithm through different aspects, where its overall performance, flexibility, and transferability are examined.

6.6.1 Dataset

The experiment is conducted on the INTERACTION dataset [46], [45], where two different scenarios are utilized: a 8-way roundabout and an unsignalized T-intersection. All data were collected by a drone from bird's-eye view with 10 Hz sampling frequency. Information such as reference paths and traffic regulations can be directly extracted from high definition maps that are in lanelet [107] format. We further utilize piece-wise polynomial to fit each of the reference paths in order to improve smoothness. Figure 6.8 shows the two scenarios we used in this work as well as their reference paths.

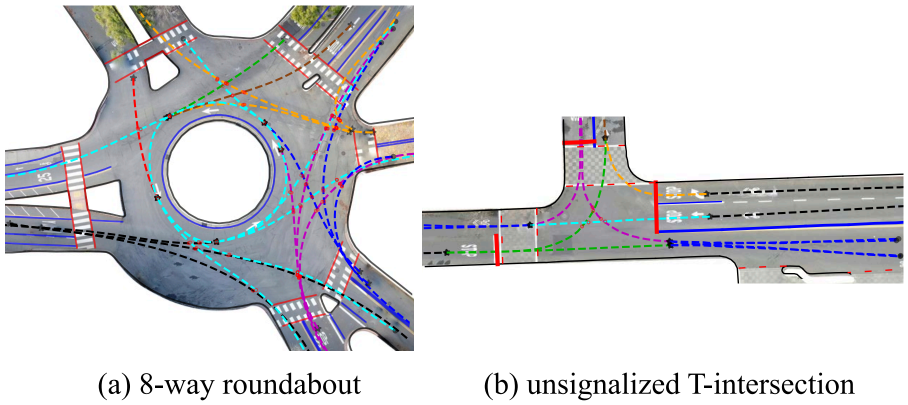


Figure 6.8: Scenarios that are utilized in this chapter as well as their corresponding reference paths.

The roundabout scenario is used to evaluate the flexibility and prediction accuracy of the proposed semantic-based algorithm. The intersection scenario, on the other hand, is used to examine the transferability of the algorithm. For roundabout scenario, a total of 21,868 data points are extracted and randomly split into approximately 80% for training and 20% for testing. For intersection scenario, there are 13,653 data points in total and we randomly select 80% of the data to train a new SGN model specifically for intersection scenario. The rest of the data collected at the intersection are used to evaluate the transferability of the SGN model learned under the roundabout scenario.

6.6.2 Implementation Details

In our experiment, up to two historical time steps of the semantic graph are utilized as input to the network although more historical time steps can be considered to improve the prediction performance. The reason is because, in this work, we focus more on how the proposed generic representations can be integrated into the graph-based network structure to enable flexibility and transferability of the prediction algorithm. Moreover, we want to show that even with limited historical information, the proposed algorithm can still generate desirable results.

The dimension for GRU-based recurrent functions, $f_{rec}^{1,2}$, is set to 128. All the layers in the network embed the input into a 64-dimensional vector with *tanh* non-linear activation function. A dropout layer is appended to the end of each layer to enhance the network's generalization ability and prevent overfitting. The size of the attention weight, $\mathbf{W}_{att}$, is set to 128. A batch size of 512 is used at each training iteration with learning rate of 0.001.

6.6.3 Visualization Results

We selected two distinct traffic situations under the roundabout scenario to visualize our test results, where the number of road agents in each case is time varying. The semantic intention prediction results and the corresponding normalized attention coefficients (represented through heatmap) at each tested frame for the two driving cases are shown in Fig. 6.9. It should be stressed that the attention coefficients are implicitly learned in the spatial attention layer of SGN during training without any supervision.

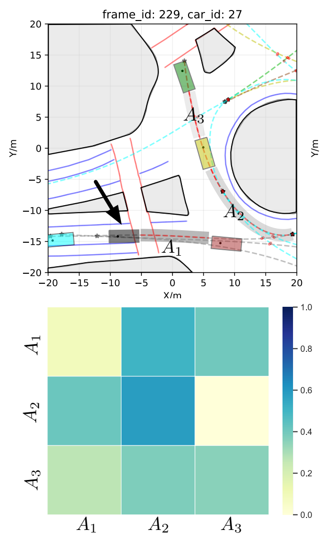
Case 1

In Fig. 6.9(a)-(d), the predicted vehicle (colored in black) manages to enter the roundabout and, at the meantime, it needs to interact with the other two vehicles that it may have conflict with. At the time frame in Fig. 6.9(a), the predicted vehicle begins to enter the roundabout and it has three options: (1) insert into $\mathcal{A}_1$, which can be interpreted as keep following its front car while expecting the other two cars (i.e. the yellow and green vehicle) to pass first; (2) insert into $\mathcal{A}_2$, which is equivalent to cut in front of the yellow vehicle; (3) insert into $\mathcal{A}_3$, which can be regarded as cut in between the green and yellow vehicle. Our result reveals that at such situation, the predicted vehicle has roughly equal probability of inserting into $\mathcal{A}_1$ and $\mathcal{A}_2$, with slightly lower probability of entering $\mathcal{A}_3$. As the predicted vehicle keeps moving forward (Fig. 6.9(b) - (d)), its probability of inserting into $\mathcal{A}_2$ decreases and goes to zero while the probability of inserting into $\mathcal{A}_3$ increases.

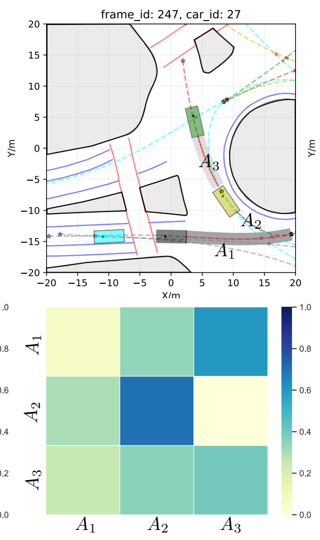
We also visualize the learned attention coefficients to examine whether the applied attention mechanism learned to associate different weights to different DIAs with reasonable interpretations. According to the attention heatmaps, $\mathcal{A}_1$'s attention vacillates between $\mathcal{A}_2$ and $\mathcal{A}_3$ to decide which area the predicted vehicle will insert into. After the decision is made, $\mathcal{A}_1$ does not need to care about other areas besides itself and thus its own attention coefficient gets higher in Fig. 6.9(d). On the other hand, $\mathcal{A}_2$ initially pays some attention to $\mathcal{A}_1$ but it gradually diverse its attention from $\mathcal{A}_1$ after realizing $\mathcal{A}_1$ does not have much interaction with itself. Note that $\mathcal{A}_2$ pays no attention to $\mathcal{A}_3$ throughout the entire period as its future states will not be influenced by its rearward DIAs. The insertion area $\mathcal{A}_3$ uniformly assign its attention to all DIAs until it is about to be inserted by the predicted vehicle where $\mathcal{A}_3$ starts to pay more attention to $\mathcal{A}_1$.

Case 2

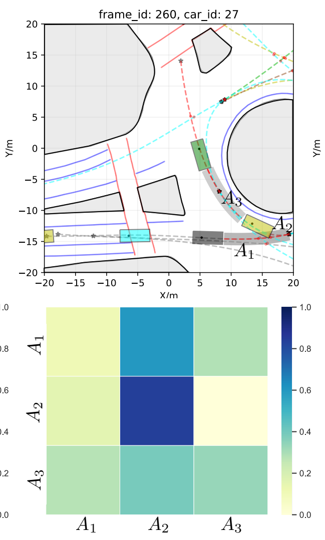
Different from case 1, this driving case is a situation where the predicted vehicle has to interact with vehicles driving on different reference paths while entering the roundabout (Fig. 6.9(e)-(g)). Initially, the predicted vehicle can choose to insert into either one of the four areas (i.e. $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$, $\mathcal{A}_4$). In Fig. 6.9(e), inserting into $\mathcal{A}_3$ has zero probability for the predicted vehicle since the area size is small and it is hard to be reached. Inserting into $\mathcal{A}_4$ also has low probability since the predicted vehicle has large geometric distance to $\mathcal{A}_4$ at the current time step. As the predicted vehicle keeps moving forward (Fig. 6.9(f), (g)), the



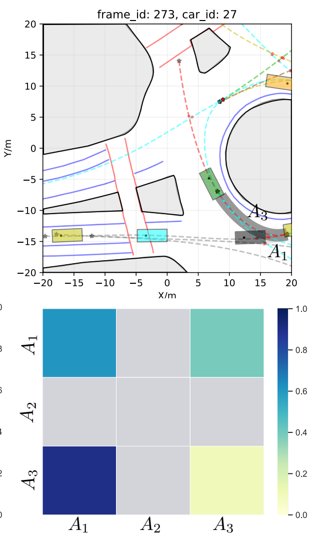
(a)



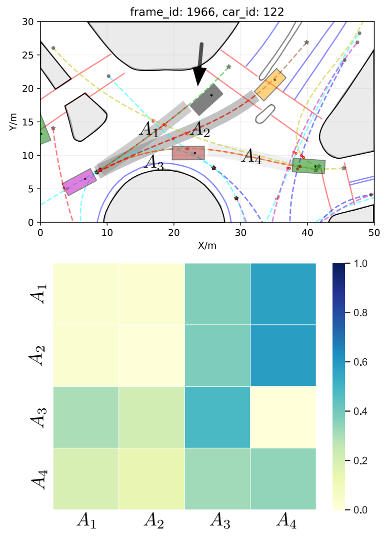
(b)



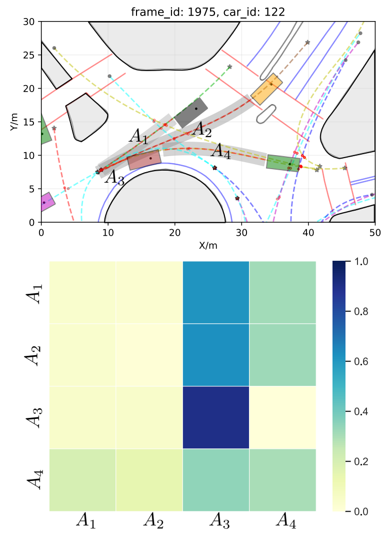
(c)



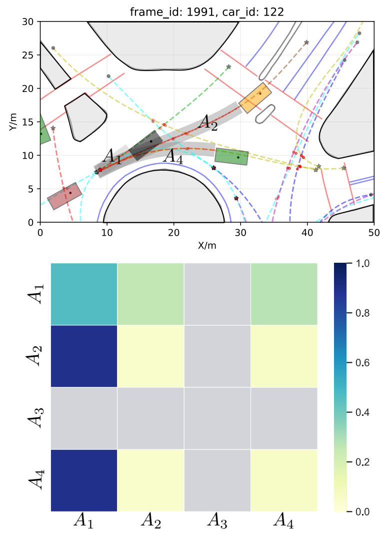
(d)



(e)



(f)



(g)

Figure 6.9: Visualization results of semantic intention and attention heatmap for case 1 (a)-(d) and case 2 (e)-(g). The predicted vehicle is colored in black. The darker the color of the dynamic insertion area, the higher probability for it to be inserted by the predicted vehicle. For each DIA that might be inserted by the predicted vehicle, the corresponding horizontal grids in the heatmap reflect how much its states will be influenced by other DIAs respectively.

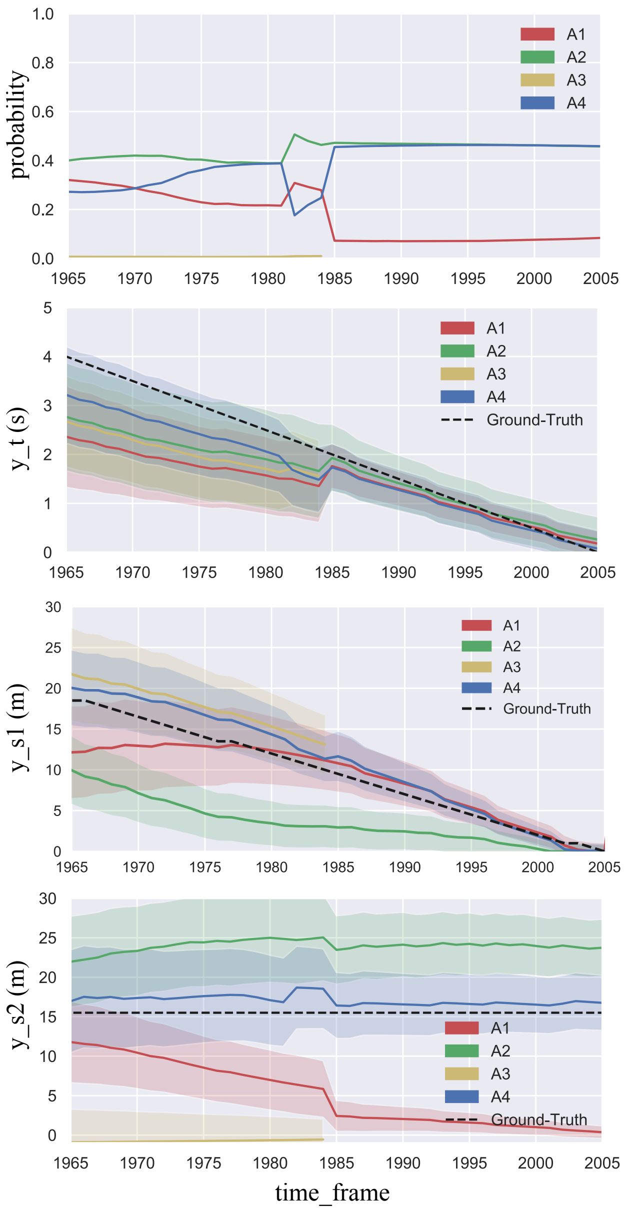


Figure 6.10: Illustration of the behavior prediction results for test case 2. For each DIAs that might be inserted by the predicted vehicle and for each goal-state variable, we plotted the predicted mean value and confidence interval based on fifty samples.

probabilities of inserting into $\mathcal{A}_2$ and $\mathcal{A}_4$ increase and almost equal to each other. Eventually, the predicted vehicle inserted into both $\mathcal{A}_2$ and $\mathcal{A}_4$. The corresponding attention heatmaps also provide some reasonable interpretations of this driving case. For example, $\mathcal{A}_1$ and $\mathcal{A}_2$ gradually shift their attention from $\mathcal{A}_4$ to $\mathcal{A}_3$ as they are being inserted by the red car (which formulates the rear bound of $\mathcal{A}_3$). Also, $\mathcal{A}_3$ stops concentrating on $\mathcal{A}_1$ and $\mathcal{A}_2$ as soon as it reveals higher chances to pass the reference point first.

We further illustrate numerical results of semantic intention and semantic goal state prediction for all possible insertion areas at each time step of this driving case, which are shown in Fig. 6.10. It is worth to note that both $\mathcal{A}_2$ and $\mathcal{A}_4$ can be regarded as the final insertion area for this case, but $\mathcal{A}_4$ is chosen since its rear bound is closer to the predicted vehicle than that of $\mathcal{A}_2$. The first plot shows the insertion probability of each DIA at each time step, the value of which coincide with the visualization results in Fig. 6.9(e)-(g). As can be seen from the last three plots in Fig. 6.10, each predicted state for $\mathcal{A}_4$ does not have large deviation from the ground truth in terms of the mean value. Also, the variance of each predicted state gradually decreases as the predicted vehicle gets closer to finish insertion. Moreover, even for those dynamic insertion areas that are not eventually inserted by the predicted vehicle, our proposed algorithm is still able to make reasonable predictions.

6.6.4 Qualitative Result Evaluation

We compared the performance of our model with that of the following five alternative approaches[2], where three of them are selected baseline methods for probabilistic prediction and the rest are variations of the proposed SGN method for ablation study. For a fair comparison, hyper-parameters such as the number of neurons, batch size, dropout rate, and training iterations in all these methods are kept the same.

- *Monte Carlo dropout* (**MC-dropout**): The MC-dropout method [25] is implemented to estimate the prediction uncertainty by using dropout during training and test time. The network we use is a four-layer multilayer perceptron (MLP) with *tanh* non-linearity. The predicted mean and variance can be obtained by performing stochastic forward passes and averaging over the output.

- *Semantic-based Intention and Motion Prediction* (**SIMP**): This is the method used in [9], where a fixed number of surrounding DIAs are considered. The entire framework is based on the standard mixture density network, where the output mean and variance are directly obtained. Notice that the way that DIAs are defined in [9] is only applicable to highway scenarios and thus we utilize the generic DIA representation proposed in this chapter to apply SIMP in other environments.

[2]Note that since our task is to predict vehicle's semantic goal information instead of trajectory, our method is not comparable to most of the state-of-the-art trajectory prediction algorithms. However, our predicted outcomes can be used for downstream tasks such as trajectory prediction or planning but will not be the focus of this work.

- *Encoder-Decoder Network* (**Enc-Dec**): The network structure of this method includes an encoder and a decoder, the implementation of which is similar to [77]. During inference, points sampled from the encoded latent space will be fed into the decoder to obtain a set of possible outcomes.

- *No-Concatenation SGN* (**NC-SGN**): This is the proposed method with modification on the predictor encoding layer, where for Eq.(9), we directly use $\hat{\mathbf{h}}_i^t$ as input for f_{enc}^2, instead of concatenating it with $\hat{\mathbf{h}}_{j\to i}^t$. In this way, the input of f_{pred} becomes the hidden edge relation between node i and j.

- *Uniform-Attention SGN* (**UA-SGN**): This is the proposed method with modification on the spatial attention layer, where we manually assign uniform attention coefficient to each node.

The intention prediction results are evaluated by calculating the multi-class classification accuracy and the semantic goal state prediction results are evaluated using root-mean-squared-error (RMSE) as well as standard deviation. Note that the input dimension has to be fixed for baseline models due to the limitation of their network structures. Therefore, only a fixed number of surrounding DIAs can be considered. As most scenarios have three surrounding DIAs, we select three DIAs that are closest to the predicted vehicle to extract input features for baseline methods. If less than three surrounding DIAs exist at a certain time frame, we assign features of those non-existent DIAs to zero.

According to the results shown in Table 6.2, MC-dropout has the lowest intention prediction accuracy and the smallest prediction variance amongst all baseline methods. This is because the dropout method is incapable of bringing enough uncertainties to the model and thus it is more likely to have over-fitting problems. Moreover, Enc-Dec has slightly worse performances than SIMP, which might due to the additional loss term in Enc-Dec. In fact, the loss function for Enc-Dec method has a trade-off between a good estimation of data and the Kullback–Leibler (KL) divergence for latent space distribution, which two terms need to be carefully fine-tuned for desirable results.

Among all the ablation methods, NC-SGN has the worst overall performance, which shows the necessity of emphasizing the relations between features of the reference DIA and other DIAs. The test results of our proposed method surpass those of UA-SGN in terms of the prediction accuracy, which implies the advantages of using attention mechanism to treat surrounding DIAs with different importance. Also, as the prediction results of all three SGN-based methods outperforms those of the three baseline methods, we can conclude that utilizing graph-based networks are, in general, better than traditional learning-based methods that have weak inductive bias. Specifically, the flexibility of dealing with varying number of input features as well as the invariance to feature ordering are essential properties for relational reasoning under prediction tasks.

Table 6.2: Quantitative Evaluation Results. As mentioned in Section 6.5.4, y_t denotes the time remains for the predicted vehicle to finish the insertion; y_{s_1} denotes the location of the inserted DIA; and y_{s_2} denotes the location of the predicted vehicle relative to the DIA it enters.

	Baseline Methods			**Ablation Methods**		
	MC-dropout	SIMP	Enc-Dec	NC-SGN	UA-SGN	SGN **(ours)**
Prob - (%)	83.52	91.29	90.04	93.62	95.05	**95.87**
Time - y_t (s)	2.11 ± 0.02	1.07 ± 1.06	1.75 ± 0.05	1.02 ± 0.78	1.02 ± 0.84	**0.95 ± 0.79**
Loc 1 - y_{s_1} (m)	5.80 ± 0.70	4.51 ± 4.66	6.93 ± 4.19	5.88 ± 6.05	3.87 ± 4.26	**3.45 ± 4.06**
Loc 2 - y_{s_2} (m)	6.35 ± 1.99	5.02 ± 5.16	5.75 ± 4.65	4.69 ± 5.05	3.84 ± 4.53	**3.55 ± 4.25**

6.6.5 Analysis of Scene Transferability

As mentioned in Section 6.2.4, our proposed prediction algorithm is intended to have zero-shot transferability. In this subsection, we explicitly examine how our model, trained under a single domain, is performed when tested under a completely unforeseen domain. To be more specific, we trained our model under an 8-way roundabout and test it under an unsignalized T-intersection.

Overall qualitative evaluation

As mentioned in Section 6.6.1, to evaluate the transferability of the proposed algorithm, the prediction performances of two SGN models are compared using selected test data from the intersection: (1) the first SGN model is learned using only the training data from the roundabout scenario, which is the same model we used for previous evaluations; (2) the second SGN model is learned using only the training data from the intersection scenario. It should be emphasized that the first SGN model is directly tested without additional training on the intersection data. Therefore, we name the first model as the *zero-shot transferred model*. In contrast, the second SGN model is named as the *conventional model*. The testing results are shown in Table 6.3.

From the table, we first notice that the performances of the conventional model using the proposed SGN structure are satisfying in terms of the prediction accuracy. More precisely, the intention prediction accuracy is close to 95% , the average temporal estimation of the goal state is less than 1.5s, the average estimation of the semantic goal location is around 2m, and the variances of these predicted variables are within a reasonable range. When the results of these two models are compared, we observe that the performance of the zero-shot transferred model still maintain desirable performance compared to the conventional model. Specifically, the semantic intention prediction accuracy only decreases 1% , the average temporal prediction error increases 0.5s, and the mean estimation error for semantic goal locations rises 2m.

Table 6.3: Evaluation of Transferability

	Zero-shot Transferred Model	Conventional Model
Prob (%)	93.73	94.68
Time - y_t (s)	2.12 ± 0.79	1.49 ± 1.55
Loc 1 - y_{s_1} (m)	4.24 ± 3.86	2.67 ± 2.13
Loc 2 - y_{s_2} (m)	4.87 ± 4.13	1.41 ± 2.71

Case studies

Two testing cases in the intersection scenario are selected to provide visualization results and detailed analysis. It is worth emphasizing that the testing results shown below are all generated through the zero-shot transferred SGN model learned under the roundabout scenario. The differences between these two domains are mainly related to map information (e.g. road topology) and traffic situation (e.g. number of on-road vehicles).

- Case 3: Figure 6.11(a) is a case where two vehicles reach the stop line simultaneously and they need to negotiate the road with each other. According to the results in Fig. 6.11(a) and (c), the transferred model is able to successfully infer the semantic intention of the predicted vehicle at an early stage (i.e. 7s before it finally inserts into $\mathcal{A}_2$). According to the corresponding heatmaps, the state of $\mathcal{A}_2$ have less effects on the predicted vehicle's decision than the state of $\mathcal{A}_1$. This is because there is no much change on the state of $\mathcal{A}_2$ and thus the predicted vehicle infers that it is unnecessary to pay much attention to $\mathcal{A}_2$. The second plot in Fig. 6.11(c) is the predicted result of y_{s_1} for each DIA. Since the red vehicle keeps waiting behind the stop line, the ground truth of s_1 for $\mathcal{A}_2$ is close to zero during this period. According to the plot, our transferred model successfully predicts such behavior with relatively small variance.

- Case 4: Figure 6.11(b) is a driving case that consists of two different stages, where the predicted vehicle first need to drive towards the stop line and then make a right turn. When the predicted vehicle is approaching the stop line, the only available insertion area is $\mathcal{A}_1$. Hence, during the first stage, the probability of inserting into $\mathcal{A}_1$ remains at one and our transferred predictor successfully infers the state changes of $\mathcal{A}_1$ as shown in Fig. 6.11(d). When the predicted vehicle is close to the stop line and preparing for a right turn, it notices that a yellow car is turning left and has potential conflict with itself. According to the first plot in Fig. 6.11(d), the inserting probability of $\mathcal{A}_1$ gradually increases (i.e. the possibility for the predicted vehicle to yield increases) and about 6s before the final insertion, the predictor is certain that $\mathcal{A}_1$ is the ground-truth DIA. From the second plot of Fig. 6.11(d), although the ground truth of s_1 changes non-linearly with time, our transferred model are still able to make relatively precise predictions. Opposite from case 3, the state of $\mathcal{A}_1$ has less variances than that of $\mathcal{A}_2$ and thus the predicted vehicle's decision depends more on $\mathcal{A}_2$. The intuition is the predicted vehicle needs to keep track of $\mathcal{A}_2$'s state in order to decide when the right turn can be made.

Discussion and further impact

In fact, since both driving scenarios we considered are in urban area with the same speed limit, without the loss of generality, we assume that the overall driving styles under these two domains are similar (i.e. have similar distributions). Therefore, in our case, the train-test domain shift is mainly due to distinct map information and different inter-vehicle relations.

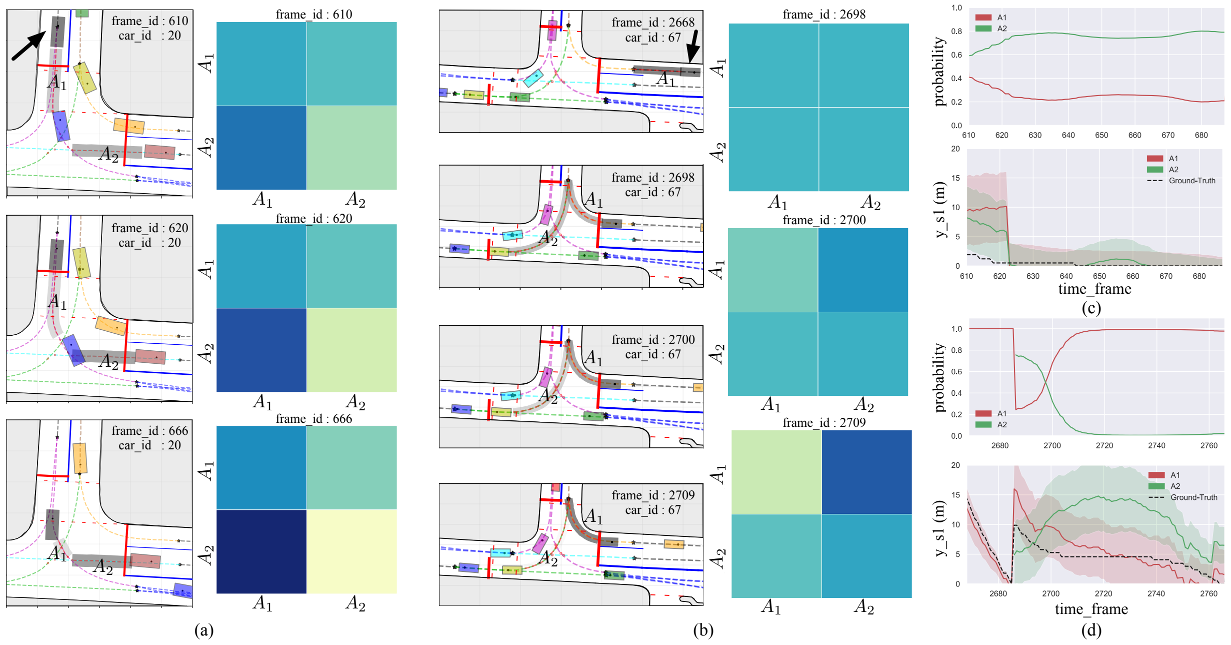


Figure 6.11: Visualization results of semantic intention and attention heatmap for test case 3 (a) and test case 4 (b). Also, selected behavior prediction results for case 3 (c) and case 4 (d) are also illustrated. All these results are generated by the zero-shot transferred SGN model.

However, if the driving style of the test domain is different from that of the training domain (e.g. two domains belong to different countries), zero-shot transfer may not have desirable performance and we may need an extra step to align the two domain distributions, which will be considered in our future works.

In general, after the proposed model is offline trained using data collected from limited driving scenarios, it can be directly utilized online under unforeseen driving environments that have different road structures, traffic regulations, number of on-road agents, and agents' internal relations. Moreover, the proposed method is data-efficient since when a new scenario is encountered, no extra data have to be collected to re-train the model for prediction tasks. Indeed, when an autonomous vehicle is navigating in constantly changing environments, it might be incapable of collecting enough online data to train a new predictor under each upcoming scenario.

6.7 Conclusions

In this chapter, a scenario-transferable semantic graph reasoning approach was proposed for interaction-aware probabilistic prediction. A generic environment representation was proposed, which are utilized to define semantic goals. These semantic goals are then integrated with the concept of semantic graphs to construct structural relations within these representations. Finally, by proposing the semantic graph network framework to operate on semantic graphs, the overall prediction framework is not only flexible to a time-varying number of interacting entities, but also transferable to unforeseen driving scenarios with completely different road structures and traffic regulations. Specifically, in our experiments, we first utilized two representative scenarios to visually illustrate the prediction performance of the algorithm and demonstrate its flexibility under different traffic situations. We then thoroughly evaluated the algorithm under a real-world scenario. According to the results, our method outperformed three baseline methods in terms of both the prediction error and the confidence intervals. We also evaluated the predictor's performance of directly transferring the predictor learned in an 8-way roundabout to an unsignalized T-intersection. The result shows that by directly extracting interpretable domain-invariant representations based on prior knowledge and incorporating these representations into the semantic graph structure as input, the proposed prediction architecture achieves desired transferability or domain generalizability. Moreover, by using graph networks that operate on these graphs and reason internal pairwise structural relations, the proposed algorithm is also invariant to the number and order of input features, which further enables transferability of the predictor.

For future works, we will consider heterogeneous agents (e.g. pedestrians and cyclists) in the environment and other types of domain shift to improve transferability of the predictor. We will also use the predicted semantic goal state information for downstream tasks such as trajectory prediction and goal-based planning.

Chapter 7

Multi-agent Decision Making with Adaptive Strategies

7.1 Introduction

Recent advanced driving assistant systems (ADAS) have had functionalities such as adaptive cruising control (ACC) and automatic parking, however, autonomous vehicles are still unable to exhibit human-like driving behaviors which require interactive decision-making. In fact, to achieve general intelligence, agents must learn how to interact with others in a shared environment.

7.1.1 Problem Description

For this work, our goal is to solve decision making problems under merging scenarios. In general, merging scenarios can be divided into two categories shown in Fig. 7.1, which contain the ones with unknown merge point such as highway entrance and the ones with known merge point that are mostly seen in urban areas. In our problem setting, only the latter category is considered, where all vehicles' motions will be on preplanned paths.

Nowadays, the industry has been taking conservative approaches to deal with merging scenarios for autonomous vehicles. The decision making is mainly based on the road priorities (which is defined in the traffic regulation) and rule-based prediction such as constant velocity. In these methods, the decision making algorithms do not rely on cooperativeness of other drivers on roads which causes autonomous vehicles make suboptimal decisions. If merging roads have similar priority as in Fig.1 (c), it is unreasonable to let the self-driving cars yield at all time and thus it needs to learn how to negotiate with other drivers. Moreover, without considering the cooperativeness of other drivers, autonomous vehicles may never merge on to dense traffic on main roads with higher priority. In this , we would like to leverage the cooperativeness of human drivers to negotiate the road with them. Specifically, we hope to generate maneuvers for self-driving cars which are safe but less conservative.

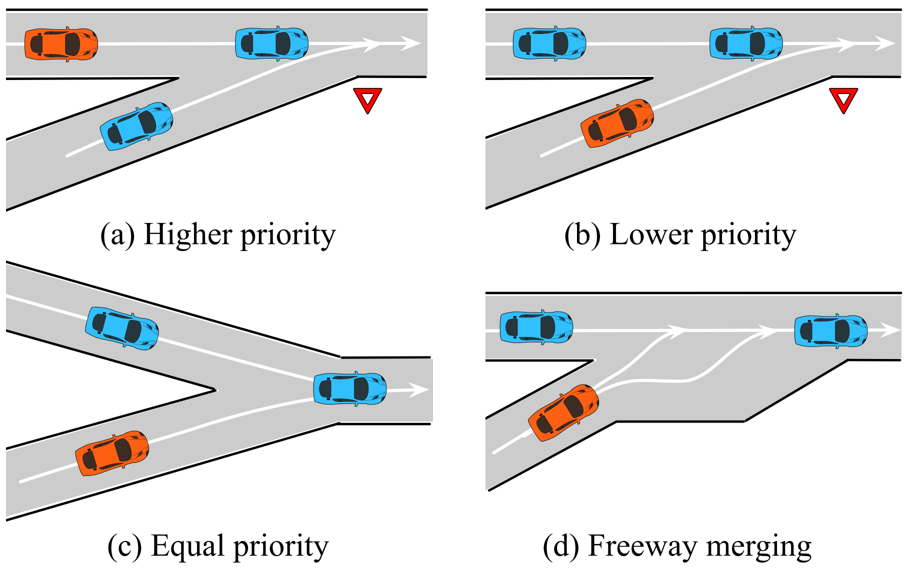


Figure 7.1: Illustrations of various merging scenarios. Red car represents the autonomous vehicle. In (a) and (b), cars with higher priorities have the right of way. However, some level of negotiations is necessary if the traffic gets dense. In (c), the car which is closer to the merge point has the right of way. However, negotiation is required since sometime it is difficult to define who is closer and each driver may have different opinions. Different from other scenarios, the red car in (d) has unknown merge point during merging.

7.1.2 Challenges

Although we focus on the known merge-point scenarios as in Fig. 1(a, b, c) in this work, it is still non-trivial for the autonomous vehicle to interact smoothly with human-driven cars. One of the challenges comes directly from the real-world scenarios: *drivers on the road can behave differently from the cooperativeness point of view*, which makes the decision making difficult for self-driving cars. Other challenges come from the implementation aspect. For example, *the driver model for other cars are usually unknown*, which is problematic since we need other cars' driver model to fine-tune and evaluate our algorithm against it so that we can develop a robust decision-making algorithm. This is in fact a cause and effect dilemma. Moreover, *the current decision-making approaches cannot achieve a desired generalization*. In order for a decision maker to be used on self-driving cars, it needs to deal with various merging scenarios instead of one. The vehicle should be able to drive in a constantly changing environment, where the road priority, traffic density, and driving behaviors of surrounding road entities will vary.

7.1.3 Related Works

1) *Decision Making:* Several rule-based and model-based methods [108, 109, 110, 111, 112] have been utilized to solve the decision making problems under merging scenarios. In DARPA Urban Challenge, the winner utilized a rule-based behavior generation mechanism [111] for predefined traffic rules. [112] modeled the decision process through a mixed logical dynamical system which is solved via model predictive control. Although these methods have reliable performances, they are only designed for specific domains and fail to model interactions between multiple road entities, which is a key component for navigating dynamic traffic environments efficiently and safely in the real-world.

In [113], the authors used Inverse Reinforcement Learning (IRL) to model how agents react to the robot's action. Despite interactions are considered, this approach assumes a full control over other vehicles with known models, which may lead the autonomous vehicle to aggressive and potentially dangerous behavior due to overconfidence in the behavior model [114]. Other work such as [115] used POMDP to solve decision making problem for autonomous driving, however, the interaction model they designed consists only a simple constant braking action that will be activated on emergency.

2) *Multi-agent Reinforcement Learning (MARL):* A variety of MARL algorithms have been proposed to accomplish tasks without pre-programmed agent behaviors. Unlike single agent RL, where each agent observes merely the effect of its own actions, MARL enables agents to interact and learn simultaneously in the same environment. By utilizing MARL methods, agents can automatically learn how to interact with each other without any improper hand-design rules.

A multi-agent deep deterministic policy gradient (MADDPG) method is proposed in [116], which learns a separate centralized critic for each agent, allowing each agent to have different reward functions. However, it does not consider the issue of multi-agent credit assignment which concerns with how the success of an overall system can be attributed to the various contributions of a system's components [117]. Such problem is addressed by Foerster et al. in [118] and they proposed a counterfactual multi-agent (COMA) policy gradients that utilizes an actor and a centralized critic to solve cooperative multi-agent problems. A recent work by Yang et al. [119] proposed a cooperative multi-goal multi-stage multi-agent reinforcement learning (CM3) approach, which learns an actor and a double critic shared by all agents. It not only takes the credit assignment issue into account as in [118], but also adopts the framework of centralized training with decentralized execution that enables the policy to use extra information to ease training.

Although the aforementioned actor-critic methods have good performances in some tested simulation environments under the MARL setting, they consider only a single level of cooperativeness during interaction . Such problem settings cannot be utilized in real driving situations since various degrees of interaction exist between vehicles due to their different cooperativeness level. Also, while driving, the vehicle should not only focus on interacting with other road entities, but also pay attention to the overall traffic flow to not impede others. Therefore, we proposed a modified actor-critic method that takes both the microscopic and macroscopic-level interactions into account.

Moreover, the MARL learning process will be inefficient and slow if each agent explores every possible states including those that are implausible. In [120], a responsibility-sensitive safety (RSS) model for multi-agent safety is developed, which is a formalism of the common sense of human judgment. We integrated such idea into our algorithm by introducing a masking mechanism similar to [121], which prevents autonomous vehicles from learning those common sense rules.

7.1.4 Contributions

In this chapter, an interaction-aware decision making with adaptive strategies (IDAS) approach is proposed for autonomous vehicles to leverage the cooperativeness of other drivers in merging scenarios, which is built upon the multi-agent reinforcement learning (MARL). The contributions of this work are as follows:

1) Utilizing the multi-agent setup to address the lack of driver model for other agents.

2) Introducing the notion of driver type to address the randomness of other drivers' behaviors.

3) Learning a single policy to obtain a generalizable decision maker that can apply adaptive strategies under various merging situations.

4) Incorporating both microscopic and macroscopic perspectives to encourage interactions and cooperativeness.

5) Leveraging curriculum learning and masking mechanism to improve learning efficiency.

With the proposed MARL method, each agent will have opportunities to interact with various types of surrounding agents during training and will be encouraged to learn how to select proper strategies when dealing with changing environments. In general, we want the self-driving cars implicitly reason about the cooperativeness of other drivers during interaction and exploit their willingness of cooperation to handle merging scenarios in a strategic way.

7.2 Preliminaries

7.2.1 Concept Clarification

We introduced two variables throughout the chapter: *road priority* and *driver type*. They will be used as input to the decision maker and the same policy is expected to behave differently by varying these two values.

1) *Road priority*: It is a metric defined by traffic law in each city and the value is discrete. Decision maker does not need to do any assumption about it since it is available in the maps.

2) *Driver type*: It is a metric to define the cooperativeness of the decision maker and the value is continuous. Each agent in the environment will be randomly assigned with a fixed driver type and this information is not shared across agents.

7.2.2 Markov Games

For this problem, we consider a multi-agent Markov game with N agents labeled by $n \in [1, N]$. The Markov game is defined by a set of states $\mathcal{S}$ describing the possible configurations of all agents, a set of partial observations $\mathcal{O}^n$ and a set of actions $\mathcal{A}^n$ for each agent. The road priority and driver-type information $\{b^n_{prio}, b^n_{type}\} \in \mathcal{B}^n$ are two predetermined and fixed parameters that could influence the agent's behavior. Each agent n chooses actions according to a stochastic policy $\pi^n : \mathcal{O}^n \times \mathcal{B}^n \times \mathcal{A}^n \to [0, 1]$, and the joint action of N agents move the environment to the next state according to the transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}^1 \times ... \times \mathcal{A}^N \to \mathcal{S}$. Each agent receives a reward $r^n : \mathcal{S} \times \mathcal{B}^n \times \mathcal{A}^n \to \mathbb{R}$, which is a function of the state, agent's behavior, and agent's action, to maximize its own total expected return $R^n = \sum_{t=0}^{T} \gamma^t r^n_t$, where $\gamma \in [0, 1)$ is a discount factor and T is the time horizon.

7.2.3 Actor-Critic Methods

Actor-critic methods [122] are widely used for variety of RL tasks in both single-agent and multi-agent environments. The actor is a parameterized policy that defines how actions are selected; the critic is an estimated state-value function that criticizes the actions made by the actor. The parameters of the actor are then updated with respect to the critic's evaluation. Different from the REINFORCE [123] algorithm that uses a stationary baseline value to reduce high variance gradient estimate, actor-critic methods use past experiences to update the baseline known as the critic. The learned critic components can shift the value of the critique enabling the gradient to point in the right direction.

Single agent

In a single-agent setting, policy π, parametrized by θ, maximizes the objective $J(\theta) = \mathbb{E}_\pi[R]$ by taking steps in the direction of $\nabla_\theta J(\theta)$, where the expectation $\mathbb{E}_\pi$ is with respect to the state-action distribution induced by π. The gradient of the policy can be written as:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\sum_t \nabla_\theta \log\pi(a_t|s_t)(Q^\pi(s_t, a_t) - b(s_t)) \right], \tag{7.1}$$

where $Q^\pi(s_t, a_t) = \mathbb{E}_\pi[\sum_{t'=t}^{T} \gamma^{t'} r(s_{t'}, a_{t'})|s_t, a_t]$ is the action-value function for policy π, $b(s_t)$ is the introduced baseline, and their difference is known as the advantage function $A^\pi(s_t, a_t)$. By choosing the value function $V^\pi(s_t)$ as the baseline and using the temporal difference (TD) error as an unbiased estimate of the advantage function, we can rewrite the advantage function as $A^\pi(s_t, a_t) \approx r(s_t, a_t) + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$.

Multi agent

For the multi-agent environment, a paradigm of *Centralized Critic, Decentralized Actor* is proposed [116][118], which is an extension of actor-critic methods. The critic is augmented with full state-action information about the policies of other agents, while the actor only has access to local information. Moreover, to address the *credit assignment* problem in multi-agent settings, which could enable each agent to deduce it's own contribution to the team's success, a counterfactual baseline

$$b(s, \boldsymbol{a}^{-n}) = \sum_{a'^n} \pi^n(a'^n | o^n) Q(s, (\boldsymbol{a}^{-n}, a'^n)) \tag{7.2}$$

is suggested in [118], where it marginalizes out the actions a of an agent n and allows the centralized critic to reason about the counterfactuals in which only agent n's actions change.

7.3 Approach

7.3.1 Interaction-aware Decision Making

Our goal is to solve the interaction-aware decision-making problem for autonomous vehicles using MARL strategy. In particular, we proposed to learn a single actor that can generate various driving behaviors and adaptive interaction strategies, as well as a pair of decentralized and centralized critics shared by all agents. We first show two learning objectives, corresponding to the need for agents to drive through merging scenarios while strictly following the rules, and interact with other agents for a more efficient merging while maintaining a good traffic flow. Then the combined objective is illustrated.

1) Since each agent is assigned with different individual rewards in order to learn distinct behaviors, it is difficult to extract various learning signals from a joint reward and thus a decentralized critic for every agent with shared parameters is needed. The **decentralized critic** aimed to provide a policy gradient for agent to learn how to drive under merging scenarios by strictly following the rules while having different behaviors. The agent is not expected to react to other agents and it will only learn how to execute rational actions to finish its own task. By naming this first objective as J_1, the policy gradient is given by:

$$\nabla_\theta J_1(\theta) \approx \mathbb{E}_\pi \Big[\sum_{n=1}^{N} \sum_t \nabla_\theta \log \pi(a_t^n | o_t^n, b^n) \big(r(o_t^n, a_t^n \quad , b^n) + \gamma V_{\phi_1}^\pi (o_{t+1}^n, b^n) - V_{\phi_1}^\pi (o_t^n, b^n) \big) \Big], \tag{7.3}$$

where $V_{\phi_1}^\pi(o_t^n, b^n)$ is the decentralized critic parametrized by ϕ_1 and is updated by minimizing the loss:

$$\mathcal{L}(\phi_1) = \frac{1}{2} \sum_i \left\| r(s_{i,t}, a_{i,t}^n, b_i^n) + \gamma V_{\hat{\phi}_1}^\pi (o_{i,t+1}^n, b_i^n) - V_{\phi_1}^\pi (o_{i,t}^n, b_i^n) \right\|^2, \tag{7.4}$$

where i is the number of sampled batches and $V_{\hat{\phi}_1}^\pi$ is the target network with parameters $\hat{\phi}_1$ that slowly update towards ϕ_1. The target network is used to stabilize the training process.

2) Although strictly following the rules will result in no accident for merging scenes, it is not the optimal strategy if we consider macroscopic-level factors such as traffic flow rate. The **centralized critic** encourages each agent to interact with each other in order to have a joint success and maintain a smooth traffic. We name the second objective as J_2 and derive its policy gradient:

$$\nabla_\theta J_2(\theta) = \mathbb{E}_\pi \Bigg[\sum_{n=1}^{N} \nabla_\theta \log\pi(a^n|o^n, b^n)(Q^\pi_{\phi_2}(s, \boldsymbol{a}, \boldsymbol{b}) - \sum_{a'^n} \pi^n(a'^n|o^n, b^n) Q^\pi_{\phi_2}(s, (\boldsymbol{a}^{-n}, a'^n), \boldsymbol{b})) \Bigg], \tag{7.5}$$

where we utilize the counterfactual baseline discussed in the previous section and define the centralized critic as $Q^\pi_{\phi_2}(s, \boldsymbol{a}, \boldsymbol{b}) = \mathbb{E}_\pi[\sum_{t'=t}^{T} \sum_{n=1}^{N} \gamma^{t'} r(s_{t'}, a^n_{t'}, b^n) | s_t, a^n_t, b^n]$ by considering a joint reward of all agents. Parameterized by ϕ_2, the centralized critic is updated by minimizing the loss:

$$\mathcal{L}(\phi_2) = \frac{1}{2} \sum_i \left\| \sum_{n=1}^{N} r(s_{i,t}, a^n_{i,t}, b^n_i) + \gamma Q^{\hat{\pi}}_{\hat{\phi}_2}(s_{i,t+1}, \hat{\boldsymbol{a}}_{i,t+1}, \boldsymbol{b}) - Q^\pi_{\phi_2}(s_{i,t}, \boldsymbol{a}_{i,t}, \boldsymbol{b}) \right\|^2, \tag{7.6}$$

where $\hat{\pi}$ denotes the target policy network and $\hat{\phi}_2$ represents parameters of the target centralized critic network.

The overall policy gradient can thus be defined as:

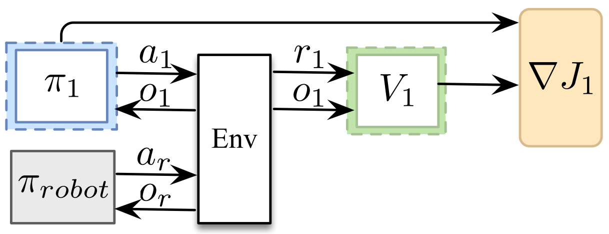
$$\nabla_\theta J(\theta) = \alpha \nabla_\theta J_1(\theta) + (1 - \alpha) \nabla_\theta J_2(\theta), \tag{7.7}$$

where $\alpha \in [0, 1]$ is the weighting factor for the two objectives. By considering two separate objectives, we can easily use curriculum learning strategy during the training process, which will be introduced in the next section.

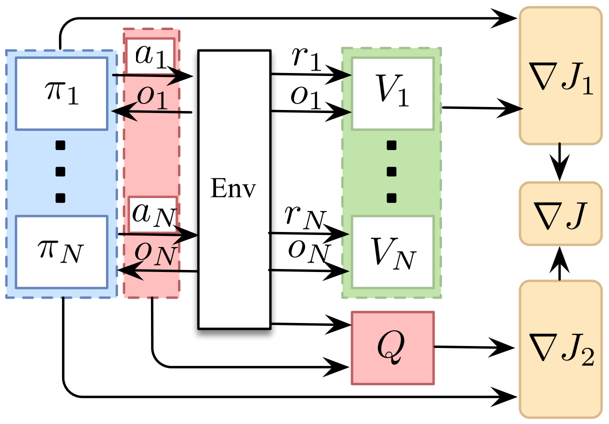
7.3.2 Different Driving Behaviors

We aimed at training a single policy network that can be used on roads with different priorities and is capable of generating different diving behaviors directly according to distinct behavioral parameters. In this way, the multi-agent driving environment will contain agents with various driving styles, which will encourage each agent to select adaptive strategies when interacting with different types of drivers.

To achieve this goal, we defined the reward function as $r(s, a, b)$, such that, besides the state and action pair, the reward also depends on the priority level b_{prio} and the driver type b_{type}. Specifically, the reward function for each agent is composed of two sub-reward functions that are related to the driving behavior : r_{finish} and $r_{collide}$. Each agent will be assigned with a one-time reward if it safely drives through the merging scenario and the reward value depends only on the driver type $r_{finish} = f_1(b_{type})$. For example, a small final reward will



(a) Stage1: Robot World



(b) Stage 2: Intelligent World

Figure 7.2: Two-stage curriculum learning structure. The decentralized critic V has the same input and network structure as the policy network. The centralized critic Q has full information of all agents as input and is connected to two FC layers with 128 units.

encourage the agent to finish the task faster than a large reward due to the discount factor γ, which induces a less cooperative driver type. If two agents collide, a negative reward will be assigned to each agent according to the road priority, namely, $r_{collide} = f_2(b_{prio})$. For instance, if an agent driving on the main lane collides with a merging agent, the latter will take more responsibility since the main-lane car has the right-of-way.

7.3.3 Curriculum Learning

It is difficult and inefficient to let the agent concurrently learn adaptive and interactive driving behaviors especially under multi-agent setting, where the state and action space grow exponentially with the number of agents. Therefore, we introduced a two-stage training process which utilized the curriculum learning [124] strategy. We successively placed the agent in a *Robot World* and a *Intelligent World* during the first and second training stage, with training episodes of 20k and 50k respectively. The discount factor γ is selected as 0.9 and the weighting factor α is set to 0.7.

For the first stage, we randomly placed an agent in a *Robot World* environment, where

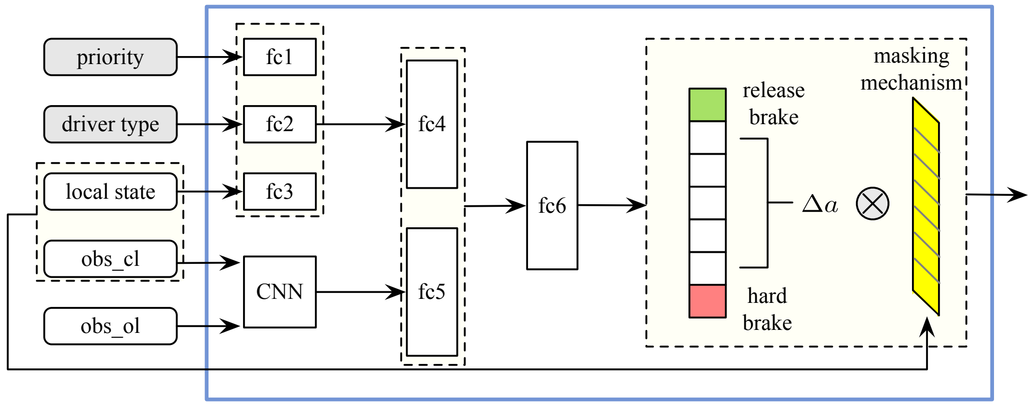


Figure 7.3: Policy network with masking mechanisms. We used 32 units in fc1, fc2, and fc3 layers; the observation input are processed by a convolution layer with 2 filters of size 3x30, stride 1x1; fc4, fc5, and fc6 all have 64 neurons. We used ReLU as the non-linear activation function for all layers.

multiple robot cars are driving on the road with predetermined maneuvers. These robot cars are strictly following the driving rules π_{robot}, where they never yield to the merging car and always give ways to the main-lane car. In order to drive safely in such robot world, the agent has to learn how to strictly follow the rules as well while having different driving behaviors. Under this stage, we trained an actor π_1 and decentralized critic V^{π_1} according to the policy gradient ∇J_1, as shown in Fig. 7.2(a).

For the second stage, we randomly placed eight agents on the road with the same policy trained in Stage1, where no more robot cars are included and agents with different behaviors will learn how to interact with each other intelligently. Based on the pretrained policy and decentralized critic, we added a centralized critic $Q^{\pi}(s, \boldsymbol{a}, \boldsymbol{b})$ and continued training π, V, Q for interactive behaviors using the combined gradient ∇J and the two losses $\mathcal{L}(\phi_1)$, $\mathcal{L}(\phi_2)$. The structure is shown in Fig. 7.2(b).

7.3.4 Masking Mechanism

Instead of exploring actions that will definitely cause an accident or brake the rules, we expect our policy to directly learn how to make decisions on the tactical level. Therefore, a masking mechanism is applied to the policy network, where three kinds of masks are considered based on vehicle kinematics ($\mathcal{M}_k$), traffic rules($\mathcal{M}_r$), and safety ($\mathcal{M}_s$) factors. The direct effect of this is that before choosing the optimal action, we apply the masking mechanism so that only a subset of feasible actions can be selected. The final mask is equal to the union of all three masks: $\mathcal{M} = \mathcal{M}_k \cup \mathcal{M}_r \cup \mathcal{M}_s$.

The kinematic mask keeps agent's acceleration and jerk in feasible ranges; the traffic-rule mask prevents the agent from exceeding the road speed limit; and the safety mask let the agent keep a safe distance from its front car, so that there will be enough reaction time when emergency happens.

7.4 Experiments

7.4.1 Environments

We developed a driving simulator[1] specifically for merging scenarios, where the road geometry, speed limit, and road priority level can be randomly defined. During training, the total number of agents on the road, traffic densities, agents' initial states, and the time each agent enters the scene are randomly assigned, which provide enough stochasticity to this merging problem.

7.4.2 Implementation Details

Basic Information

We constructed two roads with 250 meters each, where each of them can be defined as either the main lane or the merge lane with the speed limit of 20m/s or 15m/s. Similar to the real driving scenario, we allow the merging vehicle to accelerate and exceed the speed limit when its distance to the merging point is below 100m. These road constraints are applied through the rule-based mask $\mathcal{M}_r$.

Moreover, we considered some commonly used kinematic constraints for our agents by adopting the $\mathcal{M}_k$ mask, where the acceleration is within $[-2, 2]m/s^2$ and the jerk range is $[-1, 1]m/s^3$. The Intelligent Driver Model (IDM) is also utilized in order to let the agent keep a safe distance from its front vehicle, which is implemented in the safety-based mask $\mathcal{M}_s$. By using IDM, the maximum acceleration can be calculated as:

$$\begin{aligned} \dot{v_\alpha} &= a\left(1 - \left(\frac{v_\alpha}{v_0}\right)^\delta - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right) \\ s^*(v_\alpha, \Delta v_\alpha) &= s_0 + v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}}, \end{aligned} \tag{7.8}$$

where v_0 denotes the velocity limit, s_0 denotes the minimum distance between two vehicles, which is set to 2m in this problem, T denotes the time headway, and a, b are the maximum and minimum acceleration respectively.

[1]The simulation environment and some testing results can be found on `https://youtu.be/2CTTFHDW1ec`. Detailed descriptions of the simulation environment as well as a more intuitive summarization of the chapter can be found in the video attachment.

	Success (%)	Perfect Success (%)	Avg. Speed (m/s)
IDAS	**100**	**95**	**17.51**
IDAS-V	93	64	13.76
IDAS-direct	77	23	15.44
IDM	80	12	17.01
FSM+IDM	94	57	14.45

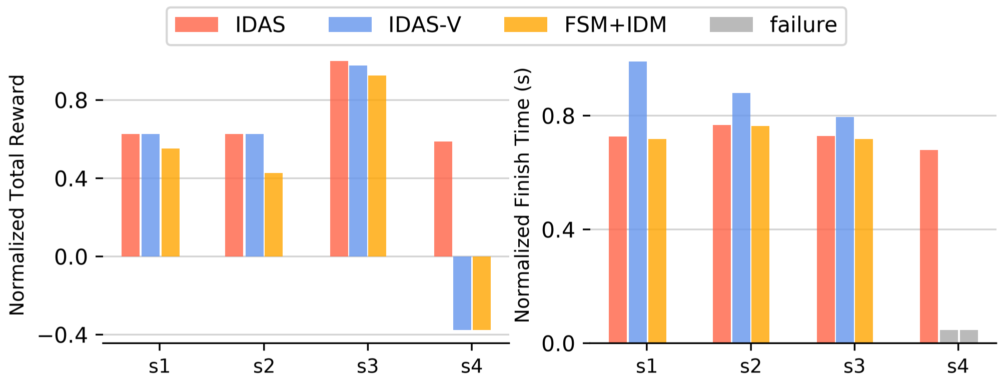


Figure 7.4: The left table shows the overall comparison over all methods. The *success rate* describes the number of success cases in all testing scenarios; the *perfect success rate* denotes the number of times that all agent succeed without any reward dedcution under all testing scearios; the *avg speed* calculates the average speed for all agents in the *perfect success* cases. The right two histograms compare the normalized reward and average finish time among selected methods over four scenarios where detail descriptions can be found in Section 7.5.2. When two vehicles crash into each other, we consider it as the failure case and color in gray.

Policy Network

The policy network structure is shown in Fig. 7.3, which has five types of input and seven actions. Each agent has a priority number, either 0 or 1, according to its current state, where 1 have higher priority than 0. Two agents will have the same road priority when their numbers are both 0 or 1. Note that an agent's priority will change during the entire process if its road priority changes. (e.g. merge into the main road from an on-ramp.)

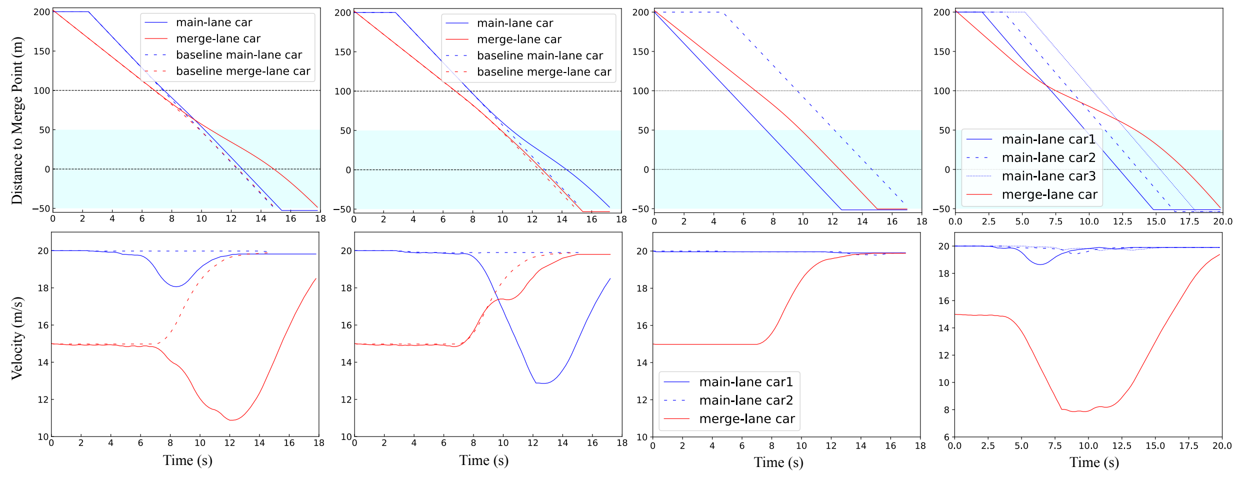
The driver type is a continuous number ranging from -2 (less cooperative) to 2 (really cooperative), which will be randomly selected and fixed for each agent. Every agent is able to receive its own local state information including the current speed, acceleration, and distance to the merging point. According to the sensor range for actual autonomous vehicles, we assume the visibility of each agent is 100m in front and back, with $0.5m/cell$ resolution, for both its current lane (obs_cl) and the other lane (obs_ol). Therefore, the observation input for each lane is a discretized one-dimensional observation grid centered on the agent, with two channels: binary indicator of vehicle occupancy and the relative speed between other vehicles and the agent.

All agents have the same action space, consisting of five discretized values for change of acceleration Δa, one hard-brake action, and one release-brake action. Although the action space is discrete, by assigning small values to Δa, the corresponding acceleration can become roughly continuous. Also, in self driving cars, a motion control module will convert the discretized results of the behavior planner into continuous acceleration by applying algorithms like Model Predictive Control (MPC) at a higher frequency (100Hz). Moreover, by using Δa as our action instead of a, we can have a wide range of actual control input sent to the vehicle while having a low-dimensional action space, which can enhance the training efficiency.

Based on the jerk range and the 5Hz decision-making frequency, we have $\Delta a = \{-0.2, -0.1, 0, 0.1, 0.2\} m/s^2$. Although the acceleration is usually between -2 to 2 m/s^2, to insure safety, the vehicle should be able to brake hard during emergency situations. Therefore, we added a hard-brake action that enables the agent to deceleration at $4m/s^2$ when necessary. A release-brake action, with the acceleration of $0m/s^2$, will be available directly after a hard-brake action in order to let the agent recover from a large negative acceleration and speed up quickly. Notice that we no longer need to include the jerk constraint in the $\mathcal{M}_k$ mask since it is already satisfies in the action space.

Rewards

The designed rewards can be divided into two categories. The first two rewards, r_{finish} and $r_{collide}$, correspond to the *microscopic-level* that influence the dynamics of each agent; the remaining two *macroscopic-level* rewards take all road participants as a whole and focus on the traffic flow characteristics. Specifically, r_{impede} penalizes agents that cause other agents to hard brake; r_{flow} penalizes every sudden brake happens inside the merging zone which is defined as the circular area with 50m radius around the merge point. Since every agent should be responsible for a bad traffic flow, the last reward will be considered only in the



(a) Simultaneous arrival 1 (b) Simultaneous arrival 2 (c) Sparse traffic (d) Dense traffic

Figure 7.5: Selected representative scenarios illustrated in the position-time domain and via the velocity profile. The blue areas are the merging zone as defined in Section 7.4.2. For a uniform comparison, the driver type for the main-lane car and merge-lane car is -1 and 1 respectively. All agents are operating on our learned policy.

joint reward instead of as a penalization for any agent.

- $r_{finish} : f_1 = (15 * b_{type} + 50)/4,\ b_{type} \in [-2, 2]$
- $r_{collide} : f_2 = -5 * b_{prio} - 5,\ b_{prio} \in \{0, 1\}$
- $r_{impede} : -5$
- $r_{flow} : -1$

7.5 Evaluation and Results

In this section, we first compare the overall performance of our method with other methods. Then, we select four representative scenarios to assess each approach through further inspections. Finally, detailed evaluations are provided to illustrate the capabilities of the proposed approach.

7.5.1 Overall Performance

We selected several criteria for validation and compared our method with the following four approaches. The first two are used for ablation study and the last two are commonly-used methods for decision-making problems.

1) *IDAS-V* : It only uses the policy generated from the first training stage.

2) *IDAS-direct*: We directly trained the policy (with 70k training episodes) using our algorithm structure without utilizing curriculum learning.

3) *Intelligent Driver Model (IDM)* : The agent will follow its front car using the IDM model and when no cars in front, it will drive at the road speed limit.

4) *Finite State Machine (FSM) + IDM*: By incorporating FSM, we manually set how each agent should interact with other agents on the other lane. Specifically, the agent will first examine if there is any vehicle on the other lane that has similar distances to the merge point as itself. If there is such vehicle, the agent will select that car as another front vehicle besides its actual front vehicle and then utilize IDM to follow both of them.

We randomly generated 100 scenarios that cover various road situations for testing and the general performances of the above methods are shown in the left table in Fig. 4. Note that when used in the real world, the learned policy will only be applied by the ego car. Therefore, to compare these methods, we first randomly selected a vehicle in the scene to be our agent and then apply each of the selected method to the chosen agent. At the same time, instead of applying a simple driving model to unselected agents, we used the learned IDAS policy on them with different assigned driver types, which provides sufficient randomness to the testing environments. Moreover, to make a fair comparison, all these methods are equipped with the masking mechanism.

As can be seen in the table, our approach outperforms other methods in all evaluated metrics and achieves zero collision. Notice that the perfect success rate for IDAS-V is low even for a high success rate. This is because by strictly following the road priorities, the merging agents always yield to the main-lane traffic to avoid potential collision while fail to consider the overall traffic flow and thus impede other agents behind. The success rate and average speed value for the IDAS-direct are lower than IDAS that uses the curriculum learning strategy, which is due to the fact that the network tries to learn everything at once and fails to acquire the desired behaviors within insufficient learning episodes. The two traditional methods, IDM and FSM+IDM, have reasonable success rates but their perfect success rates are relatively low. The reason is because these methods strictly follow the hand-designed rules and cannot have adaptive strategies when dealing with varying situations. Therefore the ego agent will count on other road entities to react to its maneuvers without considering the discomforting of other participants and any negative influences to the traffic. Such behavior will lead to serious collision if other drivers do not pay much attention to the ego car especially when they have the right-of-way.

7.5.2 Representative Scenes

Based on the evaluation results in the previous section, we selected the best three methods to further examine their performances. Four scenarios that involve high interactions are selected, where two agents are placed in the scene with one agent on each road and the details are shown below:

1) s1: $b^1_{prio} \neq b^2_{prio}$, $b^1_{type} \neq b^2_{type}$, $t^1_{mpt} > t^2_{mpt}$,

2) s2: $b^1_{prio} \neq b^2_{prio}$, $b^1_{type} \neq b^2_{type}$, $t^1_{mpt} < t^2_{mpt}$,
3) s3: $b^1_{prio} \neq b^2_{prio}$, $b^1_{type} = b^2_{type}$, $t^1_{mpt} = t^2_{mpt}$,
4) s4: $b^1_{prio} = b^2_{prio}$, $b^1_{type} \neq b^2_{type}$, $t^1_{mpt} = t^2_{mpt}$,
where t^n_{mpt} denotes the time that car n will arrive at the merge point if no interaction involved. Note that even if two cars do not arrive at the merge point simultaneously as in $s1$ and $s2$, their time difference of arrival is bounded by a small value ϵ, $|t^1_{mpt} - t^2_{mpt}| < \epsilon$, to ensure possible interaction. Different from the previous evaluation, we applied the same decision making approach to both agents. However, since the FSM+IDM method cannot have different driver types and will have the same driving behavior across all situations, we apply the IDAS policy to one of the agents in order to test how this decision maker react in varying environment.

According to the histograms in Fig. 4, IDAS can achieve the highest total reward while maintaining high merging efficiency in all testing scenarios. The other two methods,on the other hand, either have long finishing time or low reward values. Also, both methods fail in the last scene where two roads have same priority. This is because when we set both roads as the main-lane that have equally high priority, IDAS-V lets two agents strictly follow the driving rules and without considering other road's situation, which leads to collision. Note that FSM+IDM can only handle interaction when the agent has a front car on the other road and will not decelerate when no cars in front. Therefore, if any vehicle drives slightly behind the ego car or by its side, it will leave insufficient reaction time for the other agent and thus collision happens.

7.5.3 Detailed Evaluations of IDAS

Adaptive Strategies

To demonstrate that the agent indeed learns to adapt its strategy during different scenarios, we visualized the testing results in the position-time domain as well as through the velocity profile, which are shown in Fig. 7.5. In all four selected scenarios, both agents will apply the IDAS policy and one road is defined as the main-lane while the other road as the merge-lane that has lower priority.

According to the result, although each agent applies the same policy across these scenarios, it exhibit different driving strategies when dealing with different road situations. The merging car yields to the main-lane car when necessary in (d), and accelerates to merge when enough gap is available on the main road in (c). An interesting phenomenon we found in (d) is that although the main-lane agent has the right-of-way, each of the three vehicle tends to slightly press the brake when it has a similar distance to the merge-point as the merging agent. Such behavior resembles actual human drivers, where we usually, for safety purpose, decrease our speed when passing through a road with traffic.

When both agents reach the merging zone simultaneously, Fig. 7.5 (a) and (b), they begin to negotiate to avoid potential conflict. The difference between these two scenarios is that the longitudinal distance between the two cars is larger in (b) than in (a). In both situations, two agents first start to slightly decelerate before reaching the merging zone, which is similar to what human driver will do in order to prepare for emergencies. Then, in case (a), the merging car continues to decelerate due to its higher cooperativeness level and lower priority, which enables the main-lane car to accelerate since it detects the yielding intention from the merging car.

Contrarily, in case (b), although it has low priority, the merging car notices that it is ahead of the main-lane car with an acceptable distance and may have a chance to merge first. Thus the car on the merge lane begins to accelerate, which causes the main-lane car to decelerate. Instead of keep accelerating, the merging car remains its speed for a while to infer the behavior of the main-lane car and then continuous to accelerate as it detects that the other car has the willingness to yield. In contrast, if two vehicles ignore each other and fail to interact in both scenarios, a collision will happen as shown in the dashed lines. More testing results can be seen in the provided video link.

Different Driver Types

To examine whether the learned policy can generate different driving behaviors correspond to different driver types, we fixed all parameters in the scenario considered in Fig. 7.5(a) except for the driver type of the merging car. As shown in Fig. 6 (a), by continuously increasing b_{type}^{merge} from -2 to 2, the average merge time increases. Also, in Fig.6 (b), for different driver types, the merging car's velocity profile has substantial variations, which slightly influence the velocity profile of the main-lane car as well. We noticed that as the merging agent becomes more cooperative, it decelerates for a longer time to ensure a safer merging. Moreover, after a certain threshold in Fig. 6(b), the merging car discontinue to decelerate since it learns to merge efficiently and not impede the traffic flow.

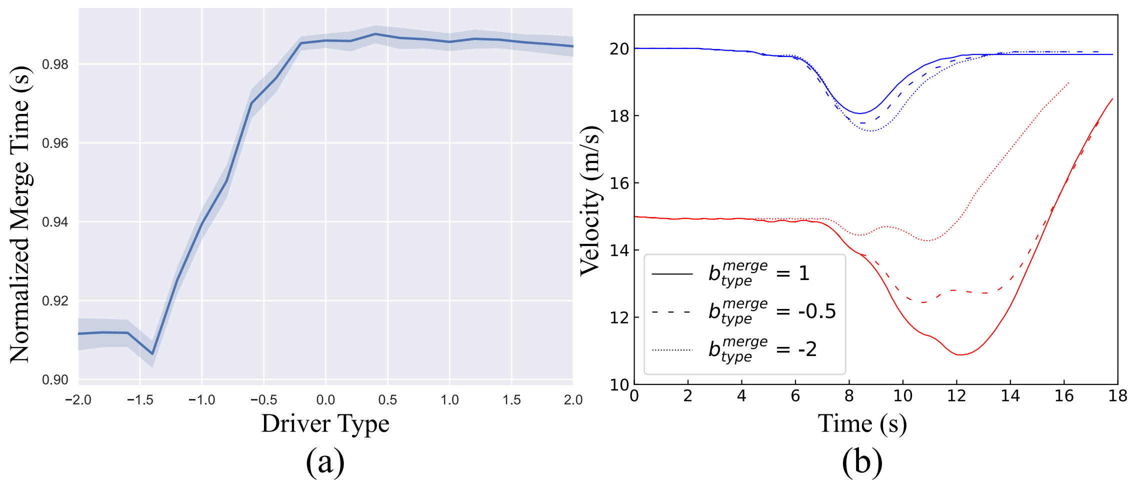


Figure 7.6: Evaluation of different driver types.

7.6 Conclusions

In this chapter, we proposed an approach called IDAS to solve decision-making problems under merging scenarios for autonomous vehicles. By introducing the concept of driver type and road priority, we expect the self-driving cars to automatically infer the behaviors of other drivers during interaction and leverage their cooperativeness to strategically handle different merging situations. We solved this problem under the multi-agent reinforcement learning setting and proposed a double critic containing a decentralized value function as well as a centralized action-value function. The curriculum learning and masking mechanism is also applied to the policy network to increase the learning efficiency. According to the evaluation results, our approach outperforms other selected methods in terms of the success rate and merging efficiency. Moreover, by further assessing IDAS, we concluded that the learned policy can successfully generate distinct driving behaviors and make adaptive strategies during interaction. Although the learned policy can achieve zero collisions in all tested scenarios, one drawback of RL-based methods is that agent's safety cannot be fully guaranteed especially in the real environment with high inherited uncertainty. Therefore, for future work, we will focus on how to transfer the learned policy from the simulation to the real-world with sufficient safety considerations. Based on the current algorithm structure, we will also extend the problem to more complicated scenarios with unknown merge point.

Chapter 8

Final Words

In this dissertation, we explored the methods to design interaction-aware prediction and planning models in various problem settings. A combination of forward and backward design process was proposed and we argued that a model should be designed based on desired model properties. Chapter 2 and Chapter 3 aimed at designing prediction models for two-agent interaction problems. More specifically, in Chapter 2, we discussed joint trajectories prediction of any two interacting agents in the scene, while making sure the model processes uncertainty, multi-modality, and interpretability. In Chapter 3, we discussed predicting trajectories of the other vehicle given possible future motion of the autonomous vehicle itself, while considering model's generalizability and reliability. Based on explorations of two-agent interaction settings, we then discussed the design of prediction and planning models for multi-agent interaction problems in Chapter 4 - 7. The design of motion and intention prediction model under multi-agent scenarios, while ensuring uncertainty, multi-modality, and interpretability, was discussed in Chapter 4 and Chapter 5. To enable model's flexibility and generalizability, the design of scenario-transferable predictor considering various static and dynamic environments was discussed in Chapter 6. The design of strategic decision making model while taking into account reliability and efficiency was discussed in Chapter 7. By utilizing the proposed model design process under different problem settings, we demonstrated that our models are able to achieve desired properties while having great performances in terms of various evaluation criteria.

There are many directions for future work, which are listed below.

- To consider unconstructed driving environment

  This dissertation focused on developing prediction and planning models under constructed driving environment with HD map information. However, it is still difficult for the autonomous vehicle to navigate under unconstructed environments when both HD map and previous driving data are unavailable. How to design models that can be used for unconstructed driving environment is challenging and need to be explored in the future.

- To enhance domain generalizability

  Contemporary machine learning algorithms provide excellent performance when training and testing data are drawn from the same underlying distribution. However, it is often impossible to guarantee prior collection of training data that is representative of the environment in which a model will be deployed, and the resulting train-test domain shift leads to significant degradation in performance. Training a predictor on a large amount of data might ease the problem but the cost of collecting data is expensive and the prediction accuracy may still degrade when unseen driving situations are encountered. In this dissertation, we proposed generic feature representations to increase generalizability, but more theoretical studies are needed to comprehensively analyze the degree of invariant across more diverse driving domains. Moreover, techniques used in the domain generalization (DG) field can be further explored in order to increase model's robustness to domain-shift without requiring access to target domain data.

- To bridge the gap between prediction and planning models

  In this dissertation, we tackled prediction and planning tasks independently, and designed corresponding models. However, since planning is the down-streaming task of prediction, it is necessary to consider how to connect the two modules, which is challenging under three main aspects. First of all, due to multi-modality and uncertainty, the predictor could generate several prediction outputs which will be sent to the planner. Then the question arises as: how can the planner process multiple future trajectories and generate an optimal path? Moreover, if the weight or confidence of each predicted result is different, how to incorporate such information in the planner accordingly? Secondly, when the autonomous vehicle is interacting with multiple vehicles in a dense traffic, there will be sets of joint future trajectories of other vehicles. What should we do if some or all of these sets cause the planner unable to generate feasible paths? Last but not least, under the circumstances where the predicted results might be untrustworthy, how should the planner react? In the future, we will take these issues into consideration while designing prediction and planning models independently or jointly.

With the development of technology, it may no longer be a fantasy to have intelligent and autonomous vehicles that think, behave and interact with the world in the way that human drivers do, so that they can better serve and assist people in daily lives.

Bibliography

[1] Wei Zhan et al. "Towards a fatality-aware benchmark of probabilistic reaction prediction in highly interactive driving scenarios". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 3274–3280.

[2] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information Fusion* 58 (2020), pp. 82–115.

[3] Stefan Hoermann, Daniel Stumper, and Klaus Dietmayer. "Probabilistic long-term prediction for autonomous vehicles". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 237–243.

[4] Jiachen Li et al. "Generic probabilistic interactive situation recognition and prediction: From virtual to real". In: *2018 21st international conference on intelligent transportation systems (ITSC)*. IEEE. 2018, pp. 3218–3224.

[5] Martin Liebner et al. "Driver intent inference at urban intersections using the intelligent driver model". In: *2012 IEEE Intelligent Vehicles Symposium*. IEEE. 2012, pp. 1162–1167.

[6] Quan Tran and Jonas Firl. "Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE. 2014, pp. 918–923.

[7] 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)*. IEEE. 2018, pp. 2111–2117.

[8] Liting Sun et al. "Courteous autonomous cars". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 663–670.

[9] Yeping Hu, Wei Zhan, and Masayoshi Tomizuka. "Probabilistic prediction of vehicle semantic intention and motion". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 307–313.

[10] Jeremy Morton, Tim A Wheeler, and Mykel J Kochenderfer. "Analysis of recurrent neural networks for probabilistic modeling of driver behavior". In: *IEEE Transactions on Intelligent Transportation Systems* 18.5 (2016), pp. 1289–1298.

[11] Henggang Cui et al. “Multimodal trajectory predictions for autonomous driving using deep convolutional networks”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 2090–2096.

[12] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[13] N Siddharth et al. “Inducing interpretable representations with variational autoencoders”. In: *arXiv preprint arXiv:1611.07492* (2016).

[14] Wei-Ning Hsu, Yu Zhang, and James Glass. “Unsupervised learning of disentangled and interpretable representations from sequential data”. In: *arXiv preprint arXiv:1709.07902* (2017).

[15] Nachiket Deo and Mohan M Trivedi. “Convolutional social pooling for vehicle trajectory prediction”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 1468–1476.

[16] Yoshihiro Nishiwaki et al. “Generating lane-change trajectories of individual drivers”. In: *2008 IEEE International Conference on Vehicular Electronics and Safety*. IEEE. 2008, pp. 271–275.

[17] Apratim Bhattacharyya, Bernt Schiele, and Mario Fritz. “Accurate and diverse sampling of sequences based on a “best of many” sample objective”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8485–8493.

[18] Seong Hyeon Park et al. “Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 1672–1678.

[19] ByeoungDo Kim et al. “Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network”. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 399–404.

[20] Tobias Gindele, Sebastian Brechtel, and Rüdiger Dillmann. “A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments”. In: *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE. 2010, pp. 1625–1631.

[21] Yeping Hu, Wei Zhan, and Masayoshi Tomizuka. “A framework for probabilistic generic traffic scene prediction”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 2790–2796.

[22] Mohammad Babaeizadeh et al. “Stochastic variational video prediction”. In: *arXiv preprint arXiv:1710.11252* (2017).

[23] Eamonn J Keogh and Michael J Pazzani. “Scaling up dynamic time warping for datamining applications”. In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2000, pp. 285–289.

[24] Joaquin Quinonero-Candela et al. “Evaluating predictive uncertainty challenge”. In: *Machine Learning Challenges Workshop*. Springer. 2005, pp. 1–27.

[25] Yarin Gal and Zoubin Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059.

[26] Christian Laugier et al. “Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety”. In: *IEEE Intelligent Transportation Systems Magazine* 3.4 (2011), pp. 4–19.

[27] Yeping Hu et al. “Multi-modal probabilistic prediction of interactive behavior via an interpretable model”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 557–563.

[28] Michael Montemerlo et al. “Junior: The stanford entry in the urban challenge”. In: *Journal of field Robotics* 25.9 (2008), pp. 569–597.

[29] Dmitri Dolgov et al. “Path planning for autonomous vehicles in unknown semi-structured environments”. In: *The international journal of robotics research* 29.5 (2010), pp. 485–501.

[30] Aleksandr Kushleyev and Maxim Likhachev. “Time-bounded lattice for efficient planning in dynamic environments”. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 1662–1668.

[31] Yoshiaki Kuwata et al. “Real-time motion planning with applications to autonomous urban driving”. In: *IEEE Transactions on control systems technology* 17.5 (2009), pp. 1105–1118.

[32] Jeong hwan Jeon et al. “Optimal motion planning with the half-car dynamical model for autonomous high-speed driving”. In: *2013 American control conference*. IEEE. 2013, pp. 188–193.

[33] Georges S Aoude et al. “Threat-aware path planning in uncertain urban environments”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 6058–6063.

[34] Rudolf Emil Kalman. “When is a linear control system optimal?” In: (1964).

[35] Andrew Y Ng, Stuart J Russell, et al. “Algorithms for inverse reinforcement learning.” In: *Icml*. Vol. 1. 2000, p. 2.

[36] Pieter Abbeel and Andrew Y Ng. “Apprenticeship learning via inverse reinforcement learning”. In: *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 1.

[37] Brian D Ziebart et al. “Maximum entropy inverse reinforcement learning.” In: *Aaai*. Vol. 8. Chicago, IL, USA. 2008, pp. 1433–1438.

[38] Sergey Levine and Vladlen Koltun. “Continuous inverse optimal control with locally optimal examples”. In: *arXiv preprint arXiv:1206.4617* (2012).

[39] Henrik Kretzschmar et al. "Socially compliant mobile robot navigation via inverse reinforcement learning". In: *The International Journal of Robotics Research* 35.11 (2016), pp. 1289–1307.

[40] Liting Sun et al. "Behavior planning of autonomous cars with social perception". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 207–213.

[41] Anirudha Majumdar et al. "Risk-sensitive Inverse Reinforcement Learning via Coherent Risk Models." In: *Robotics: Science and Systems*. 2017.

[42] Liting Sun et al. "Interpretable modelling of driving behaviors in interactive driving scenarios based on cumulative prospect theory". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 4329–4335.

[43] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. "Learning structured output representation using deep conditional generative models". In: *Advances in neural information processing systems* 28 (2015), pp. 3483–3491.

[44] David Premack and Guy Woodruff. "Does the chimpanzee have a theory of mind?" In: *Behavioral and brain sciences* 1.4 (1978), pp. 515–526.

[45] Wei Zhan et al. "Constructing a highly interactive vehicle motion dataset". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 6415–6420.

[46] Wei Zhan et al. "Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps". In: *arXiv preprint arXiv:1910.03088* (2019).

[47] Hiren M Mandalia and Mandalia Dario D Salvucci. "Using support vector machines for lane-change detection". In: *Proceedings of the human factors and ergonomics society annual meeting*. Vol. 49. 22. SAGE Publications Sage CA: Los Angeles, CA. 2005, pp. 1965–1969.

[48] Joel C McCall et al. "Lane change intent analysis using robust operators and sparse bayesian learning". In: *IEEE Transactions on Intelligent Transportation Systems* 8.3 (2007), pp. 431–440.

[49] Seungje Yoon and Dongsuk Kum. "The multilayer perceptron approach to lateral motion prediction of surrounding vehicles for autonomous vehicles". In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2016, pp. 1307–1312.

[50] Thomas Streubel and Karl Heinz Hoffmann. "Prediction of driver intended path at intersections". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE. 2014, pp. 134–139.

[51] Derek J Phillips, Tim A Wheeler, and Mykel J Kochenderfer. "Generalizable intention prediction of human drivers at intersections". In: *2017 IEEE intelligent vehicles symposium (IV)*. IEEE. 2017, pp. 1665–1670.

[52] Stefan Klingelschmitt, Volker Willert, and Julian Eggert. “Probabilistic, discriminative maneuver estimation in generic traffic scenes using pairwise probability coupling”. In: *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*. IEEE. 2016, pp. 1269–1276.

[53] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. “On a formal model of safe and scalable self-driving cars”. In: *arXiv preprint arXiv:1708.06374* (2017).

[54] Wei Zhan et al. “A non-conservatively defensive strategy for urban autonomous driving”. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2016, pp. 459–464.

[55] Florent Altché and Arnaud de La Fortelle. “An LSTM network for highway trajectory prediction”. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 353–359.

[56] David Lenz et al. “Deep neural networks for Markovian interactive scene prediction in highway scenarios”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 685–692.

[57] Jürgen Wiest et al. “Probabilistic trajectory prediction with Gaussian mixture models”. In: *2012 IEEE Intelligent Vehicles Symposium*. IEEE. 2012, pp. 141–146.

[58] Brian D Ziebart et al. “Planning-based prediction for pedestrians”. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2009, pp. 3931–3936.

[59] Eike Rehder and Horst Kloeden. “Goal-directed pedestrian prediction”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2015, pp. 50–58.

[60] Hien Q Dang et al. “Time-to-lane-change prediction with deep learning”. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 1–7.

[61] Christian Wissing et al. “Probabilistic time-to-lane-change prediction on highways”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 1452–1457.

[62] Christopher M Bishop. “Mixture density networks”. In: (1994).

[63] *U.S. Department of Transportation Intelligent Transportation Systems Joint Program Office (JPO)*. `https://www.its.dot.gov/data/`.

[64] Nicolai Meinshausen and Greg Ridgeway. “Quantile regression forests.” In: *Journal of Machine Learning Research* 7.6 (2006).

[65] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.

[66] Ruben Izquierdo et al. “Vehicle trajectory and lane change prediction using ann and svm classifiers”. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 1–6.

[67] Stefan Klingelschmitt et al. “Probabilistic situation assessment framework for multiple, interacting traffic participants in generic traffic scenes”. In: *2016 IEEE intelligent vehicles symposium (IV)*. IEEE. 2016, pp. 1141–1148.

[68] Jacob Walker et al. “An uncertain future: Forecasting from static images using variational autoencoders”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 835–851.

[69] Graham Elliott and Allan Timmermann. “Economic forecasting”. In: *Journal of Economic Literature* 46.1 (2008), pp. 3–56.

[70] Elizabeth J Kendon et al. “Heavier summer downpours with climate change revealed by weather forecast resolution model”. In: *Nature Climate Change* 4.7 (2014), pp. 570–576.

[71] Akira Kanazawa, Jun Kinugawa, and Kazuhiro Kosuge. “Adaptive Motion Planning for a Collaborative Robot Based on Prediction Uncertainty to Enhance Human Safety and Work Efficiency”. In: 35.4 (2019), pp. 817–832.

[72] Jin Xu, Hai-Bo Shu, and Yi-Ming Shao. “Modeling of driver behavior on trajectory–speed decision making in minor traffic roadways with complex features”. In: 20.1 (2018), pp. 41–53.

[73] Laurène Claussmann et al. “A review of motion planning for highway autonomous driving”. In: 21.5 (2019), pp. 1826–1848.

[74] Nasim Arbabzadeh and Mohsen Jafari. “A data-driven approach for driving safety risk prediction using driver behavior and roadway information data”. In: 19.2 (2017), pp. 446–460.

[75] Li Zhu et al. “Big data analytics in intelligent transportation systems: A survey”. In: 20.1 (2018), pp. 383–398.

[76] Nikita Jaipuria, Golnaz Habibi, and Jonathan P How. “Learning in the Curbside Coordinate Frame for a Transferable Pedestrian Trajectory Prediction Model”. In: 2018.

[77] 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)*. IEEE. 2019, pp. 3539–3546.

[78] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. “Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst”. In: *Proc. of Robotics: Science and Systems*. 2019.

[79] Antonia Breuer et al. “Analysis of the Effect of Various Input Representations for LSTM-Based Trajectory Prediction”. In: 2019, pp. 2728–2735.

[80] Jiyang Gao et al. “Vectornet: Encoding hd maps and agent dynamics from vectorized representation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11525–11533.

[81] Xin Huang et al. "Uncertainty-aware driver trajectory prediction at urban intersections". In: 2019, pp. 9718–9724.

[82] Lian Hou et al. "Interactive Trajectory Prediction of Surrounding Road Users for Autonomous Driving Using Structural-LSTM Network". In: (2019).

[83] Alexandre Alahi et al. "Social lstm: Human trajectory prediction in crowded spaces". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 961–971.

[84] Alessandro Berlati et al. "Ambiguity in Sequential Data: Predicting Uncertain Futures With Recurrent Models". In: 5.2 (2020), pp. 2935–2942.

[85] Agrim Gupta et al. "Social gan: Socially acceptable trajectories with generative adversarial networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2255–2264.

[86] Namhoon Lee et al. "Desire: Distant future prediction in dynamic scenes with interacting agents". In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 336–345.

[87] David Sierra González, Jilles Steeve Dibangoye, and Christian Laugier. "High-speed highway scene prediction based on driver models learned from demonstrations". In: IEEE. 2016, pp. 149–155.

[88] Alex Zyner, Stewart Worrall, and Eduardo Nebot. "Naturalistic driver intention and path prediction using recurrent neural networks". In: (2019).

[89] Stefano V Albrecht et al. "Integrating Planning and Interpretable Goal Recognition for Autonomous Driving". In: *arXiv preprint arXiv:2002.02277* (2020).

[90] Nicholas Rhinehart et al. "Precog: Prediction conditioned on goals in visual multi-agent settings". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2821–2830.

[91] Sergio Casas, Wenjie Luo, and Raquel Urtasun. "Intentnet: Learning to predict intention from raw sensor data". In: *Conference on Robot Learning*. PMLR. 2018, pp. 947–956.

[92] Hang Zhao et al. "Tnt: Target-driven trajectory prediction". In: *Conference on Robot Learning*. PMLR. 2020.

[93] Peter W Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018).

[94] Xin Li, Xiaowen Ying, and Mooi Choo Chuah. "GRIP: Graph-based Interaction-aware Trajectory Prediction". In: 2019, pp. 3960–3966.

[95] Petar Veličković et al. "Graph attention networks". In: *Proc. International Conference on Learning Representations,* 2018.

[96] Yuexin Ma et al. “Trafficpredict: Trajectory prediction for heterogeneous traffic-agents”. In: *Proc. AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 6120–6127.

[97] Yuying Chen et al. “Robot Navigation in Crowds by Graph Convolutional Networks with Attention Learned from Human Gaze”. In: 5.2 (2020), pp. 2754–2761.

[98] Chen Sun et al. “Stochastic prediction of multi-agent interactions from partial observations”. In: *Proc. International Conference on Learning Representations*. 2019.

[99] Mohammad Bahram et al. “A combined model-and learning-based framework for interaction-aware maneuver prediction”. In: 17.6 (2016), pp. 1538–1550.

[100] Fei Ye et al. “Prediction-based eco-approach and departure at signalized intersections with speed forecasting on preceding vehicles”. In: 20.4 (2018), pp. 1378–1389.

[101] Alex Zyner, Stewart Worrall, and Eduardo Nebot. “A recurrent neural network solution for predicting driver intention at unsignalized intersections”. In: 3.3 (2018), pp. 1759–1764.

[102] Naveed Muhammad and Björn Åstrand. “Predicting agent behaviour and state for applications in a roundabout-scenario autonomous driving”. In: *Sensors* 19.19 (2019), p. 4279.

[103] Angelos Filos et al. “Can autonomous vehicles identify, recover from, and adapt to distribution shifts?” In: *International Conference on Machine Learning*. PMLR. 2020, pp. 3145–3153.

[104] Da Li et al. “Deeper, broader and artier domain generalization”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5542–5550.

[105] Wei Zhan et al. “Spatially-partitioned environmental representation and planning architecture for on-road autonomous driving”. In: 2017, pp. 632–639.

[106] Junyoung Chung et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *NIPS Workshop on Deep Learning*. 2014.

[107] Fabian Poggenhans et al. “Lanelet2: A high-definition map framework for the future of automated driving”. In: 2018, pp. 1672–1679.

[108] Michael Montemerlo et al. “Junior: The stanford entry in the urban challenge”. In: *Journal of field Robotics* 25.9 (2008), pp. 569–597.

[109] Chris Urmson et al. “Autonomous driving in urban environments: Boss and the urban challenge”. In: *Journal of Field Robotics* 25.8 (2008), pp. 425–466.

[110] Isaac Miller et al. “Team Cornell’s Skynet: Robust perception and planning in an urban environment”. In: *Journal of Field Robotics* 25.8 (2008), pp. 493–527.

[111] Christopher R Baker and John M Dolan. “Traffic interaction in the urban challenge: Putting boss on its best behavior”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, pp. 1752–1758.

[112] Julia Nilsson and Jonas Sjöberg. “Strategic decision making for automated driving on two-lane, one way roads using model predictive control”. In: *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE. 2013, pp. 1253–1258.

[113] Dorsa Sadigh et al. “Planning for Autonomous Cars that Leverage Effects on Human Actions.” In: *Robotics: Science and Systems*. 2016.

[114] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. “Planning and decision-making for autonomous vehicles”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), pp. 187–210.

[115] Constantin Hubmann et al. “Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles”. In: *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE. 2017, pp. 1671–1678.

[116] Ryan Lowe et al. “Multi-agent actor-critic for mixed cooperative-competitive environments”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6379–6390.

[117] Yu-Han Chang, Tracey Ho, and Leslie P Kaelbling. “All learning is local: Multi-agent learning in global reward games”. In: *Advances in neural information processing systems*. 2004, pp. 807–814.

[118] Jakob N Foerster et al. “Counterfactual multi-agent policy gradients”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[119] Jiachen Yang et al. “CM3: Cooperative Multi-goal Multi-stage Multi-agent Reinforcement Learning”. In: *arXiv preprint arXiv:1809.05188* (2018).

[120] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. “On a formal model of safe and scalable self-driving cars”. In: *arXiv preprint arXiv:1708.06374* (2017).

[121] Mustafa Mukadam et al. “Tactical Decision Making for Lane Changing with Deep Reinforcement Learning”. In: *31th Conference on Neural Information Processing Systems (NIPS)*. 2017.

[122] Vijay R Konda and John N Tsitsiklis. “Actor-critic algorithms”. In: *Advances in neural information processing systems*. 2000, pp. 1008–1014.

[123] Ronald J Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[124] Yoshua Bengio et al. “Curriculum learning”. In: *Proceedings of the 26th annual international conference on machine learning*. ACM. 2009, pp. 41–48.