Designing Explainable Autonomous Driving System for Trustworthy Interaction

by

Chen Tang

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

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Professor Anil Aswani
Professor Francesco Borrelli
Professor Mark Mueller

Spring 2022

Designing Explainable Autonomous Driving System for Trustworthy Interaction

Abstract

Designing Explainable Autonomous Driving System for Trustworthy Interaction

by

Chen Tang

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

The past decade has witnessed significant breakthroughs in autonomous driving technologies. We are heading toward an intelligent and efficient transportation system where human errors are eliminated. While excited about the emergence of autonomous vehicles with increasing intelligence, the public has also raised concerns about their reliability. Modern autonomous driving systems usually adopt black-box deep-learning models for multiple function modules (e.g., perception, behavior prediction, behavior generation). The opaque nature of neural networks and their complex system architecture make it extremely difficult to understand the behavior of the overall system, which prevents humans from confidingly sharing the road and interacting with autonomous vehicles. This motivates the design of a more transparent system to build a foundation for trustworthy interaction between humans and autonomous vehicles.

This dissertation is concerned with the design of an *explainable autonomous driving system*, leveraging the strengths of explainable artificial intelligence, control, and causality. In particular, we focus on the behavior system of an autonomous vehicle, which plays a crucial role in its interaction with human road participants. The work consists of two parts. In Part I, we explore methods to improve model interpretability. The goal is to ensure that the model is more intelligible for humans in the design stage, which is achieved by introducing hard or soft constraints formulated from domain knowledge. We demonstrate how to formulate domain knowledge of social interaction into structured reward functions (Chapter 2) and pseudo labels (Chapter 3) as well as how to utilize them to induce interpretable driving behavior models. We also introduce an interpretable and transferable hierarchical driving policy that combines deep learning with robust model-based control (Chapter 4). In Part II, we explore the usage of post hoc explanation techniques in diagnosing model behavior. We introduce two case studies, in which we utilize sparse graph attention to diagnose interaction modeling in behavior prediction (Chapter 5) and develop a Shapley-value-based method to study the inherent causality issue in conditional behavior prediction (Chapter 6).

To my family and friends

Contents

List of Figures

List of Tables

Acknowledgments

The past six years have been an incredible journey in my life. I would not have come this far without the tremendous help from my family, friends, mentors, and colleagues. Their support has been even more valuable considering the difficult times during the pandemic.

First, I would like to sincerely thank my advisor Professor Masayoshi Tomizuka for his generous support and guidance over the past six years. He has been a great mentor and role model, and I am grateful for his profound knowledge, enthusiasm, and open-minded attitude toward research. Without his guidance as well as the excellent research atmosphere he created in the MSC lab, I would never have learned how to become an independent researcher or found my passion for an academic career. I hope I can follow his example and become an extraordinary scholar and mentor.

I would like to thank Professor Anil Aswani, Professor Francesco Borrelli, and Professor Mark Mueller for serving as my dissertation committee members. I am also thankful to Professor Kameshwar Poolla and Professor Jonathan Shewchuk for serving on my qualifying exam committee. Without their knowledge, I could not have finished the work on this dissertation. I want to especially thank Wei-Bin Zhang for his tremendous support and help throughout my time at Berkeley. In addition, Professor Jianjun Shi has given me valuable advice and guidance since I was an undergraduate student.

During my PhD study, I have collaborated with many excellent minds in the MSC lab. Some of these works are part of this dissertation. While the rest are not included in this dissertation, I was greatly inspired by all of these collaborations. I am sincerely grateful to Professor Changliu Liu, Professor Jianyu Chen, Dr. Wei Zhan, Dr. Zhuo Xu, Jinning Li, Lingfeng Sun, and Chenfeng Xu. I would also like to express my thanks to my current and past colleagues in the MSC lab: Minghui Zheng, Shiying Zhou, Kevin Haninger, 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, Kiwoo Shin, Zining Wang, Jiachen Li, Yujiao Cheng, Shiyu Jin, Hengbo Ma, Jessica Leu, Changhao Wang, Xinghao Zhu, Yiyang Zhou, Ge Zhang, Huidong Gao, Zheng Wu, Catherine Faulkner, Wu-Te Yang, Xiang Zhang, Ting Xu, Chengfeng Xu, Chenran Li, Akio Kodaira, Ran Tian, Ce Hao, Wei-Jer Chang, Jen-Wei Wang, and Yichen Xie. I am grateful for their assistance during my PhD study and their efforts in maintaining a great research atmosphere.

I was also fortunate to collaborate with many outstanding researchers from different institutes. I would like to thank the visiting scholars Professor Angel Cuenca, Dr. Long Xin, Julian M. Salt Ducaju, Haonan Chang, and Yaru Niu. In addition, I want to thank Dr. Sujitha Martin, Dr. Teruhisa Misu, Dr. Chiho Choi, Nishan Srishankar, and Enna Sachdeva from the Honda Research Institute for the research collaborations that contributed to this dissertation. I am also thankful to my mentors from my internship at Waymo, Dr. Stéphane Ross and Qiaojing Yan, who inspired me greatly in my research.

Last, but not least, I give my deepest love to my family. I want to thank my parents and grandparents for their endless support, encouragement, and love. I am especially grateful to my wife, Mengxin Wang, who I met and married during my time at Berkeley. Thank you for your accompaniment and support and for all the wonderful memories we created and shared over the past five years.

Chapter 1

Introduction

1.1 Explainable Autonomous Driving: An Overview

The past decade has witnessed the rapid development of autonomous vehicles (AVs). According to the reports from California Department of Motor Vehicles (DMV) [6], the total annual mileage driven by the licensed test AVs in California increased from $450,597$ miles to $4,051,850$ miles between 2015 and 2021. Meanwhile, the longest mileage per disengagement for a single company drastically increased from $1,244$ miles (achieved by Waymo) to $50,108$ miles (achieved by AutoX) between 2015 and 2021. The statistics seem to suggest that there has been a substantial leap toward the ideal driverless transportation system that motivated the development of autonomous driving, where human-related errors are eliminated, resulting in reduced traffic congestion and improved safety [4]. Deep learning has played an essential role in pushing forward autonomous driving technologies in recent years. Deep-learning models have been widely adopted for different AV function modules. They have even become standard practice for some modules, such as perception and behavior prediction.

While those black-box neural network models significantly boosted the overall performance, they also led to severe societal concerns about the reliability of the systems. Unlike previous application domains where deep learning has been proven successful, such as computer vision (CV) and natural language processing (NLP), AVs operate in a safety-critical environment, where they need to interact with other human road participants (e.g., pedestrians, bicycles, and human-driven cars) while carrying human passengers. Humans need to be convinced of their reliability to engage in trusting cooperation with AVs. However, it is still not possible to fully understand or anticipate the behavior of these black-box models. Furthermore, modern autonomous driving systems usually stack multiple deep-learning modules over their pipelines, which makes it even more prohibitive to inspect and analyze the behavior of the overall system. The public concern has also been exaggerated by recent traffic accidents associated with AVs, including some fatal accidents in which people were killed. While AVs are not necessarily responsible for all the accidents, the opacity of the autonomous driving systems makes it challenging to assert responsibility in accidents involving AVs. Taken together, these factors highlight a need to improve the transparency of autonomous driving systems.

In the broader community of machine learning (ML) and artificial intelligence (AI), explainable AI (XAI) has drawn increasing research attention in recent years. XAI aims to develop techniques allowing humans to better understand the behavior of an AI system by providing explanations [45], which improves transparency and builds the foundation for trustworthy human–AI interaction. In this dissertation, we are interested in exploring how we may utilize the design principles and techniques of XAI to develop an *explainable autonomous driving system*. XAI research has explored techniques for systems with different levels of requirements for explainability. We follow the literature to categorize XAI methods in order of increasing explainability as follows:

- **Neural Network Post Hoc Explanation**: Techniques to explain the behavior of a deep-learning model with another neural network. The neural network for explanation can be either integrated into the backbone model (e.g., visual attention network [62]) or trained as a separate module (e.g., textual explanation via NLP models [63]). The explanation networks explain the model behavior in a post hoc manner and do not interfere with the training process of the backbone model. Therefore, such kinds of techniques have minimal impact on performance. However, these explanations generated by *black-box* modules could be ambiguous and misleading. Therefore, they should be interpreted with extreme caution. Otherwise, they could instead cause significant damage when applied to high-stakes decision-making systems [104].

- **Model-Agnostic Post Hoc Explanation**: Techniques to create model-agnostic explanations, for instance, feature attribution methods (e.g., Shapley values [84]). Unlike the last category, model-agnostic methods do not rely on specific explanation networks. They quantify certain model characteristics (e.g., sensitivity to a specific input feature) with a sequence of explicit operations. Compared to neural network explanations, the generated explanations are well defined and have explicit semantic meaning. Nevertheless, we should still treat them cautiously because neural networks are complex nonlinear systems. It could be risky to over-interpret the results (e.g., generalizing a local quantitative characteristic globally without care).

- **Interpretable ML**: Techniques to improve *model interpretability* in the *design* stage. Unlike post hoc explanations, these techniques focus on directly designing ML models with improved interpretability. Formally, soft and hard interpretability constraints are incorporated into the original training objective of an ML model to make it more intelligible for humans [106]. The constraints are domain-specific, reflecting the definition of interpretability in the specific domain (e.g., sparsity, disentanglement). For example, a common technique for improving the interpretability of a neural network model is to design a low-dimensional intermediate embedding space. The interpretability of the low-dimensional space can be further improved by enforcing a disentanglement constraint. For neural network models, we usually focus on interpretability constraints targeting some sub-modules, such as the low-dimensional embedding space mentioned above. This results in *partially interpretable* ML models. In some applications, it is also possible to enforce a constraint on the entire model. For instance, we may restrict the model structure to inherently interpretable models (e.g., decision trees). In this case, the model becomes a *fully interpretable* one. While a fully interpretable model is always preferred if applicable, it could be challenging to optimize in practice, making it prohibitive to obtain fully interpretable ML models with performance comparable to their black-box counterparts in many applications.

When designing an explainable autonomous driving system, the core decision to make is the required level of explainability. Ideally, we want fully interpretable ML models that perform equally as well as the currently used deep-learning models. However, this could be extremely difficult for a complex problem domain like autonomous driving. In this dissertation, we will discuss what we think is the best practice for designing an explainable autonomous driving system.

1.2 A Designer's Perspective on Explainable Autonomous Driving

Fig. 1.1 illustrates a typical paradigm of an explainable autonomous driving system. By equipping the AVs with explanations, an obvious benefit is that end users (e.g., passengers) can better understand the decisions made by the AVs. This allows the users to monitor the decision-making procedure of the AVs in real time. If the AVs falsely understand the environment and make inappropriate decisions, the users can provide feedback or promptly override the AVs. In this dissertation, we consider this paradigm from a designer's perspective. In particular, we focus on the behavior system of the overall pipeline. As illustrated in Fig. 1.2, a behavior system mainly consists of three key elements: 1) *behavior prediction*: forecasting the future behavior of the surrounding agents; 2) *motion planning*: planning the future behavior of the ego vehicle; 3) *vehicle control*: controlling the ego vehicle to fulfill the anticipated behavior. Altogether, these three modules control the behavior of the AVs, given the information collected from the upstream perception module. The behavior system involves reasoning about the sophisticated interaction among all the road participants. Moreover, it directly governs how the AV interacts with the environment. Therefore, it is crucial to develop a transparent behavior system to support trusting interaction between humans and AVs.

We divide our discussion in this dissertation into two parts. Part I focuses on developing methodologies to improve model *interpretability*, which we argue should be the primary design objective for high-stakes decision-making systems like autonomous driving. The goal is to ensure that the model is more intelligible for humans in the design stage, which is achieved by introducing hard or soft constraints formulated from domain knowledge. Part II explores the role of *post hoc explanations* in the design loop of autonomous driving systems.

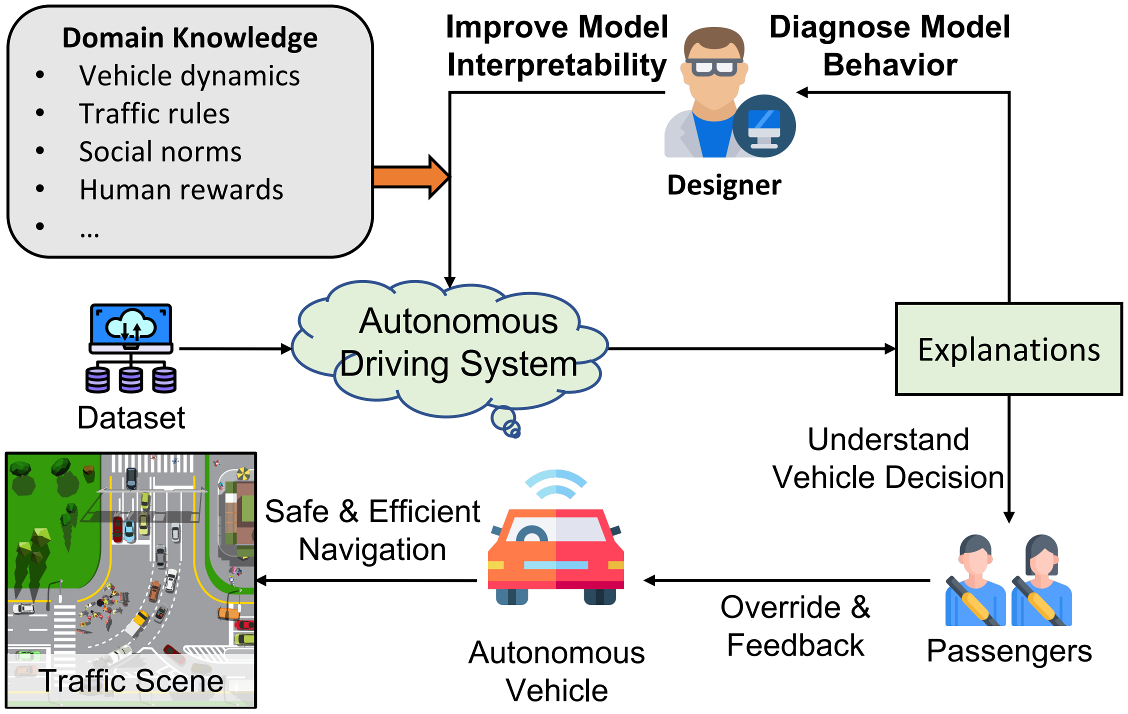


Figure 1.1: The paradigm of an explainable autonomous driving system. End users (e.g., passengers) can better understand the decision-making process of the autonomous driving system via explanations. In return, they could provide feedback or override AVs promptly if the AVs falsely understand the environment and make inappropriate decisions. In this dissertation, we focus on the perspective of model designers in the paradigm. Part I focuses on developing methodologies to improve model *interpretability*, which we argue should be the primary design objective for high-stakes decision-making systems like autonomous driving. The goal is to ensure that the model is more intelligible for humans in the design stage, which is achieved by introducing hard or soft constraints formulated from domain knowledge. Part II explores the role of *post hoc explanations* in the design loop of autonomous driving systems. We demonstrate how we may diagnose the systems and further improve the model design by analyzing the model behavior via explanations.

We demonstrate how we may diagnose the systems and further improve the model design by analyzing the model behavior via explanations. An overview of the approaches and emphasis of the two parts is provided in Fig. 1.1.

1.2.1 Model Design

As mentioned in Sec. 1.1, when designing an explainable autonomous driving system, the core decision to make is our requirement for the level of explainability. We want to empha-

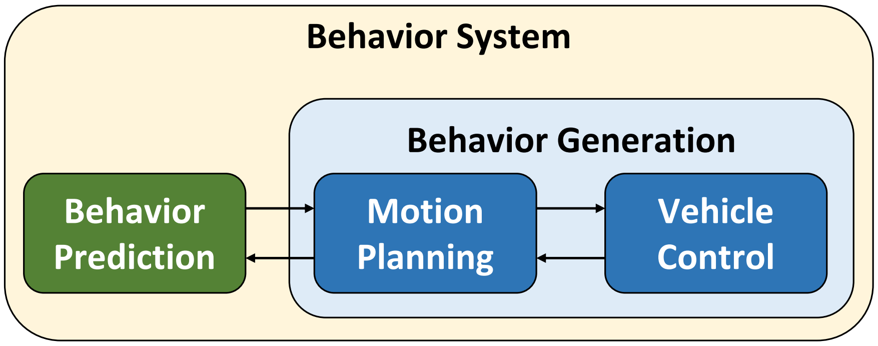


Figure 1.2: The three key elements of the behavior system: 1) *behavior prediction*: forecasting the future behavior of the surrounding agents; 2) *motion planning*: planning the future behavior of the ego vehicle; 3) *vehicle control*: controlling the ego vehicle to fulfill the anticipated behavior. In a typical workflow, the modules are executed sequentially in the order of behavior prediction, motion planning, and vehicle control. We connect them with bi-directional arrows to unify the advanced system architectures that involve mutual information exchange between modules.

size that our primary objective should be to improve the models' inherent interpretability instead of designing post hoc explanation modules. As stated in Sec. 1.1, post hoc explanations, especially those generated by black-box modules, could be ambiguous and misleading. Further, the underlying mechanism of a black-box neural network model is too intricate to be fully understood through post hoc explanations. Therefore, it is risky to merely rely on post hoc explanations to understand a complex and high-stakes decision-making system like autonomous driving. Instead, we should design a more interpretable model so that the decision-making procedure is aligned with human reasoning. In the first part of the dissertation, we explore various methodologies for improving the interpretability of different modules of a behavior system. Specifically, we focus on incorporating domain knowledge into a deep-learning model so that the model operates consistently with the domain knowledge. When designing each method in Part I, we essentially address the task by answering the following two key questions:

- What domain knowledge do we want to incorporate into the deep-learning models? Some examples of domain knowledge we think are potentially beneficial are vehicle dynamics, traffic rules, and social norms. For instance, if constraints on vehicle dynamics are enforced, we can ensure that the behavior system outputs dynamically feasible actions. If we regularize the behavior prediction model with traffic rules and social norms, the AVs can reason the behavior of the surrounding agents in a more human-like way.

- How can we formulate the domain knowledge into interpretability constraints, either hard or soft, and optimize the training objective? While some formats of domain knowledge (e.g., vehicle dynamics) are well-defined, knowledge about, for example, traffic rules and social norms is rather abstract. It is not trivial to formulate such abstract knowledge into constraints in a principled manner. In addition, we need to ensure that a practical training algorithm exists to optimize the model for the formulated constraints.

1.2.2 Model Diagnosis

While interpretability should be prioritized when designing explainable autonomous driving systems, post hoc explanations can still play an essential role in the design procedure, especially when the model is *partially interpretable* and contains black-box modules. If carefully interpreted, explanations can help the designers monitor the model behavior and iterate the design if any failure mode is detected. In the second part of this dissertation, we investigate how we may diagnose an autonomous driving system with post hoc explanations. In particular, we focus on studying the interaction modeling problem in behavior prediction. In highly interactive urban traffic scenarios, it is crucial to model the interaction among road participants for accurate forecasting. Various model structures have been proposed to encode social information from aggregated observations in the literature. However, most existing works have focused on overall prediction accuracy without verifying whether social interaction is appropriately modeled and utilized. Only recently have researchers started to formally investigate this problem, including works presented in this dissertation [127, 128] as well as [85]. It was found that existing methods did not necessarily encode and utilize social information as desired. In the second part of this dissertation, we present two case studies on diagnosing and improving interaction modeling for behavior prediction with the help of post hoc explanations.

1.3 Contributions and Outline

In summary, this dissertation aims to develop methodologies to develop an explainable behavior system, an essential building block of an explainable autonomous driving system. The outline of the dissertation is summarized in Fig. 1.3. In Part I, we focus on improving model interpretability by incorporating human domain knowledge. Chapters 2–3 introduce two different approaches to designing interpretable driving behavior models. In particular, knowledge of interaction patterns is utilized to regularize the models. The interaction knowledge is formulated into either reward functions (Chapter 2) or pseudo labels (Chapter 3). Chapter 4 demonstrates how we can utilize prior knowledge of vehicle dynamics to develop an interpretable and transferable deep-learning-based planning and control framework. In Part II, we focus on developing post hoc explanation toolkits for diagnosing interaction modeling in behavior prediction. Chapter 5 introduces a sparse graph attention mechanism to diagnose the social posterior collapse issue of variational autoencoders for interaction modeling. Chapter 6 introduces a Shapley-value-based method to diagnose the inherent causality issue of using conditional behavior prediction for interactive planning. A detailed summary of the contribution of each chapter is provided in the following subsections.

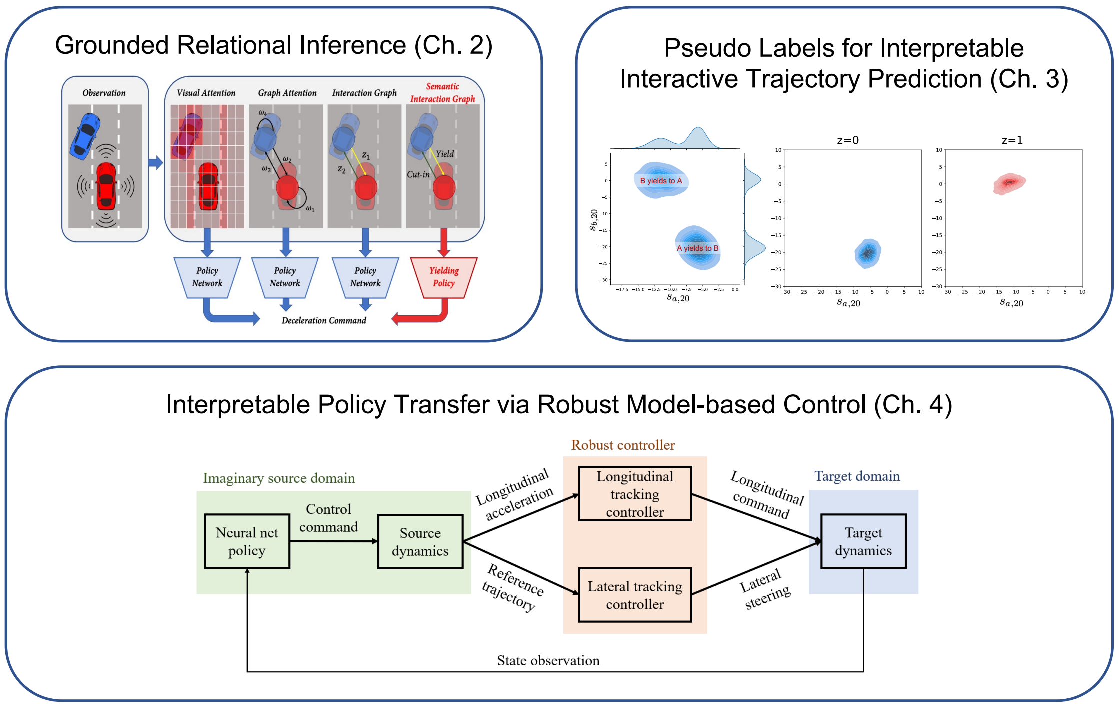
Chapter 2: Grounded Relational Inference

Multi-agent interaction modeling is crucial for accurate trajectory prediction in highly interactive scenarios. We investigate how to design an interpretable deep-learning model for interaction modeling. We propose equipping the model with explanations revealing the underlying mechanism of the model. However, unlike post hoc explanations, which could be ambiguous and falsely interpreted by humans, we want to ensure that the explanations are consistent with both *human domain knowledge* and the model's *inherent causal relation* in the *design* stage, thereby binding the model's underlying mechanism with human understanding to ensure interpretability. We focus on the relational inference problem studied in [67], where an interactive system is modeled by explicitly inferring the inherent relations between interacting objects. Kipf et al. proposed the neural relational inference (NRI) model for this problem. In NRI, the inferred relations are formatted as a latent interaction graph, whose edges are aligned with discrete latent variables corresponding to a cluster of pairwise interactive behaviors between the objects. The inferred interaction graph could potentially serve as the explanations. However, since NRI learns the latent space in an unsupervised manner, it cannot be ensured that humans will be able to precisely interpret the semantic meaning behind the interaction graph without ambiguity. To address this issue, we propose grounding the latent space in a set of interactive behaviors defined with domain knowledge.

We reframe relational inference as an inverse reinforcement learning (IRL) problem and introduce structured reward functions to ground the latent space. Concretely, we model the system as a multi-agent Markov decision process (MDP), where the agents share a reward function that depends on the relational latent space. We design structured reward functions based on human domain knowledge to explicitly define the interactive behaviors corresponding to the latent space. To solve the formulated IRL problem, we propose grounded relational inference (GRI) [129]. It has the variational-autoencoder-like (VAE) graph neural network (GNN) in NRI [67] as the backbone model. We incorporate the structured reward functions into the model as an additional reward decoder. A variational extension of the adversarial inverse reinforcement learning (AIRL) [35] algorithm is derived to train the model in an end-to-end fashion. We demonstrate that the proposed GRI framework can model interactive traffic scenarios in both simulated and real-world settings and generate semantic interaction graphs grounded in domain knowledge to explain vehicles' behavior based on their interactions. Hence, GRI could serve as an essential building block of an interpretable trajectory prediction model.

Chapter 3: Pseudo Labels for Interpretable Interactive Trajectory Prediction

Chapter 2 introduces GRI as an interpretable model for multi-agent interaction modeling, which could be further extended into an interpretable trajectory prediction model. In this



Part I: Domain Knowledge Driven Interpretable Model Design

Part II: Model Diagnosis with Post hoc Explanation

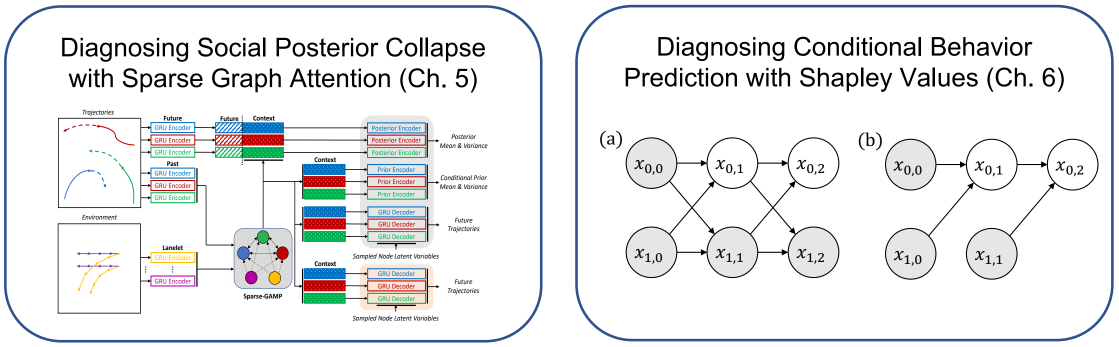

Diagnosing Social Posterior Collapse with Sparse Graph Attention (Ch. 5)
Trajectories
Environment
GRU Encoder
Posterior Encoder
Prior Encoder
GRU Decoder
Sparse-GAMP
Future Trajectories
Sampled Node Latent Variables
Diagnosing Conditional Behavior Prediction with Shapley Values (Ch. 6)
(a)
(b)

Figure 1.3: Dissertation outline. Chapters 2-4 belong to Part I. Chapters 2-3 present two approaches for regularizing interaction modeling with domain knowledge of interaction patterns, which is formulated into either reward functions (Ch.2) or pseudo labels (Ch.3). Chapter 4 demonstrates how we can utilize prior knowledge of vehicle dynamics to develop an interpretable and transferable deep-learning-based planning and control framework. Chapters 5-6 belong to Part II. Chapter 5 introduces a sparse graph attention mechanism for diagnosing the social posterior collapse issue of variational autoencoders for interaction modeling, while Chapter 6 introduces a Shapley-value-based method for diagnosing the inherent causality issue of using conditional behavior prediction for interactive planning.

chapter, we instead study how to directly improve the interpretability of state-of-the-art trajectory prediction models while maintaining consistent prediction accuracy. We focus on the interaction prediction problem, where the trajectories of two interacting agents are predicted and evaluated jointly. This prediction task is formulated by Waymo [25], targeting highly interactive scenarios. In particular, we are interested in the interaction prediction problem under the goal-conditioned framework. We extend the state-of-the-art single-agent goal-conditioned framework [42] to jointly predict the goal distribution of two interacting agents. Additionally, we leverage conditional variational autoencoder (CVAE) [117] and introduce a discrete latent space to capture the interaction modes explicitly, which should improve the model's interpretability and sampling efficiency.

However, it cannot be guaranteed that the model can learn an informative latent space distinguishing semantically meaningful interaction modes that are useful for downstream modules. We find that the vanilla model is prone to posterior collapse, resulting in an entirely uninformative latent space. To tackle this problem, we propose the use of *pseudo labels* designed based on domain knowledge to guide the training [121]. For each ground-truth goal pair, we assign positive target values to goal pair candidates similar to it. The model learns to encode similar goal pairs into the same latent variable by minimizing the corresponding auxiliary loss function, defined as the distance between the decoded distribution and the pseudo labels. In particular, we introduce three types of pseudo labels incorporating three different types of domain knowledge. We apply the proposed method to train the prediction model on the Waymo Open Dataset and show that the pseudo labels can effectively induce an interpretable latent space and further improve prediction performance.

Chapter 4: Interpretable Policy Transfer via Robust Model-based Control

In Chapter 4, we turn to the downstream behavior generation module and develop an interpretable learning-based planning and control framework. In particular, we focus on the policy transfer problem, where the driving policy network is trained in a source domain and deployed in a target domain with modeling discrepancy. This is an important problem because robustness has been the main drawback preventing the application of neural networks policies in autonomous driving. Prior learning-based policy transfer efforts focus on transfer learning and meta learning. However, their applications in autonomous driving are limited due to safety concerns. Overall, such learning-based policy transfer methods embed transferable representations into *black-box* neural networks and hope for the best in the target domain, and thus they are not transparent and reliable. We seek to solve the policy transfer problem using an alternative tool, robust control, and propose a generic PN-RC transfer framework [140, 126]. The PN-RC framework aims to solve the policy transfer problem between domains with different vehicle dynamics models. In this framework, the policy network is applied to an imaginary setting in the source domain to generate a kinematic-level reference trajectory for the target vehicle. In the target domain where we assume prior knowledge of the dynamics of the target vehicle, a robust controller is designed to track the reference trajectory tolerating the modeling gap. The proposed framework has two fundamental advantages: 1) transferring the *interpretable* kinematic features makes the framework transparent and reliable; 2) the stability and response of the low-level controller can be analyzed to design the optimal controller parameters. We present one realization of the PN-RC framework. For the policy network, we implement a hierarchical parallel attribute network (PAN) [141], which can flexibly compose a set of elementary policy networks to tackle various driving tasks. For the robust controller, we design an adaptive disturbance-observer-based (DOB) robust tracking controller. The DOB controller is integrated with a novel reference smoothing algorithm, which improves the tracking performance when a dynamical re-planning scheme is involved. The simulation and preliminary experimental results validate the ability of the proposed PN-RC architecture to zero-shot transfer the policy under a certain level of parameter variation and external disturbances.

Chapter 5: Diagnosing Social Posterior Collapse with Sparse Graph Attention

Variational Autoencoder (VAE) [66] has been widely used in multi-agent behavior modeling and trajectory prediction due to its ability to learn a low-dimensional representation of the original high-dimensional data. However, VAEs do not necessarily learn a good representation of the data [18]. This leaves us to wonder whether a VAE-based model can always learn a good representation of a multi-agent interacting system. In particular, we are interested in the following question: *Does the latent space always properly model interaction?* Formally, given a latent variable model of an interacting system, where a latent variable governs each agent's behavior, we wonder if the VAE learns to encode social context into the latent variables. This is a crucial question that is under-explored in the literature. Without properly encoding the social context, the model may suffer from over-estimated variance and large prediction error. More importantly, since the joint behavior of the agents is a consequence of their interactions, ignoring the causes may lead to poor generalization ability [129, 47].

In this chapter, we initiate the study on this important issue with the help of a novel sparse graph attention message-passing (sparse-GAMP) layer. We incorporate the sparse-GAMP layer into the prediction model as the encoder for social context aggregation. Sparse-GAMP generates a sparse attention map, from which we can directly identify those surrounding agents ignored by the model. With the help of sparse-GAMP, we find that a typical formulation of VAE for multi-agent interaction is indeed prone to ignoring the historical social context. We refer to this phenomenon as *social posterior collapse*. We analyze social posterior collapse under this formulation and propose several measures to alleviate the issue. Our experiments show that social posterior collapse indeed occurs in real-world prediction tasks and that the proposed measures can effectively alleviate the issue. The results suggest that the model without social posterior collapse can attain better generalization performance if the historical social context is informative for prediction.

Chapter 6: Diagnosing Conditional Behavior Prediction with Shapley Values

In Chapter 5, we study the interaction modeling problem in behavior prediction. Like most existing works [149, 43], the prediction model we focus on follows a *passive* prediction scheme: The target agents' future trajectories are predicted given their historical trajectories and those of other surrounding agents. When using such a prediction model, downstream decision-making modules determine the autonomous agent's action according to the predicted trajectories passively. To ensure safety under various predicted trajectories of others, the ego agent has to be overly conservative with inefficient maneuvers, especially in highly interactive scenarios. To this end, researchers started to investigate a more coherent interactive prediction and planning framework that relies on predicting the surrounding agents' future trajectories conditioned on the ego agent's future actions. Under such frameworks, the autonomous agents can reason over potential actions while considering their influence on surrounding agents. This can then result in more efficient and less conservative maneuvers in interactive scenes. In particular, we are interested in the line of research where a different prediction task is formulated to develop and evaluate the prediction sub-modules in a self-contained manner, which we refer to as the *conditional behavior prediction* (CBP) task. In the CBP task, the future trajectories of the target agents are predicted conditioned on the ground-truth future trajectory of an ego agent. Standard prediction metrics are adopted to quantify the performance. This allows us to leverage large-scale naturalistic traffic datasets to develop and validate a conditional prediction model prior to closed-loop testing.

However, we argue that it is risky to train and evaluate the model for *conditional inference*. In the current CBP task, the prediction model learns the posterior distribution of future trajectories conditioned on the future trajectory of the ego agent, where the future trajectory of the ego agent is falsely treated as an *observation*. We argue that this results in a discrepancy between 1) what information an autonomous agent receives by querying a CBP model with a potential plan and 2) how the others will actually react if the agent executes the plan. This discrepancy may lead to overly confident anticipation of the ego agent's influence on the surroundings, resulting in potential safety hazards during online usage. Instead, we argue that we should treat the query plan as an *intervention* [93] and build the prediction model to approximate the future trajectory distribution under the intervention of enforcing the ego agent's future trajectory. We refer to this new task as the *interventional behavior prediction* (IBP) task [128]. In IBP, we still train and evaluate the model with an offline dataset. The setting is essentially the same as CBP, except for learning an interventional distribution instead of a conditional one.

Without knowing the ground-truth distribution in the intervention, we can only compare the model's output against the ground-truth labels for evaluation. However, such evaluation metrics are naturally biased toward a CBP model. Accordingly, we propose verifying the inherent temporal independence of a prediction model before comparing the prediction performance to ensure proper evaluation of the IBP task. Under the interventional distribution, the predicted states of the target agents in earlier time steps should be independent from the ego agent's states in later time steps. We propose a *Shapley-value-based metric* to verify whether the model obeys this temporal independence. We show that the proposed metric can effectively identify a CBP model violating the temporal independence. More importantly, we show that a state-of-the-art CBP model of a popular prediction benchmark indeed violates the temporal independence, and its prediction accuracy benefits from this violation. The results support the necessity of establishing a benchmark for our newly formulated IBP task to replace the commonly adopted CBP benchmarks, in which the proposed Shapley-value-based metric will play an important role.

Part I

Domain Knowledge Driven Interpretable Model Design

Chapter 2

Grounded Relational Inference

2.1 Introduction

In this chapter, we explore how to design an interpretable model for an essential building block of behavior system—multi-agent interaction modeling. It is crucial to properly model the interaction among road participants for accurate behavior prediction in highly interactive scenarios. However, modeling interaction between intelligent agents is challenging due to the intricate nature of human minds. To gain the trust of human users, it is crucial that the behavior system reasons the interaction transparently so that the human users can monitor whether the system correctly understands an interactive scenario. As mentioned in Chapter 1, one popular approach to give people a better understanding of the underlying mechanisms of deep learning models is through post hoc explanations [3]. Vision-based approaches, such as visual attention [62] and deconvolution [12], illustrate which segments of the input image are important to the output. Interaction-aware models, such as social Long-Short Term Memory network (LSTM) with social attention [1, 136] and GNN with graph attention [53, 135, 120, 67], identify the agents that are critical to the decision-making procedure. However, post hoc explanations could be ambiguous and falsely interpreted by humans. For instance, a visual attention map only illustrates which regions of the input image the model's output depends on. The semantic meaning behind the causal relation is left for human users to interpret. Kim et al. [63] attempted to resolve this ambiguity by aligning textual explanations with visual attention. However, the underlying mechanism of the model is not necessarily consistent with the textual explanations.

To develop an interaction model that humans can truly trust, we argue that the model should be equipped with explanations that are consistent with both *human domain knowledge* and the model's *inherent causal relation*, thereby binding the model's underlying mechanism with human understanding to ensure interpretability. To explore how to approach the desired model, we study the relational inference problem formulated in [67], where an interactive system is modeled by explicitly inferring the inherent relations between interacting objects. They proposed the NRI model to solve this problem. Formally, the NRI model aims to solve a reconstruction task. Given the observed trajectories of all the objects, an encoder first infers the interaction between objects represented by an interaction graph, whose edges are aligned with discrete latent variables corresponding to a cluster of pairwise interaction behaviors between the objects. Afterward, a decoder, which learns the dynamical model conditioned on the inferred interaction graph, reconstructs the trajectories given the initial states. If the decoder can accurately reconstruct the trajectories, the latent space can effectively capture the interaction between interacting objects.

We find this discrete latent space particularly interesting because the inferred interaction graph could potentially serve as the desired explanation: it explains the reconstructed trajectories as a sequence of interaction behaviors among agents. Moreover, the reconstructed trajectories are governed by the same interaction graph. Therefore, the NRI model seems promising to fulfill our goal to make the explanation consistent with the model's underlying mechanism. However, since the NRI model learns the latent space in an unsupervised manner, we cannot ensure that humans will be able to precisely interpret the semantic meaning behind the inferred interaction graph without ambiguity. To address this issue, we propose grounding the latent space in a set of interactive behaviors defined with human domain knowledge.

As a running example, consider the scenario depicted in Fig. 2.1, where we ask different models to control the red vehicle. Attention mechanisms can indicate the critical pixels or agents, but they cannot recognize different effects—the two cars are mutually important but affect each other in distinct ways. The NRI model can distinguish between different interactive behaviors. Still, the latent space does not have explicit semantic meaning. In contrast, our model should determine the interaction graph with a latent space grounded in yielding and cutting-in behaviors. It learns control policies that generate behaviors consistent with their definitions in domain knowledge (e.g., traffic rules) and executes the corresponding policies according to the inferred edge types. This semantic interaction graph illustrates the model's understanding of the scenario and explains the action it takes.

If we merely want to make the interaction graph consistent with humans' labeling of the scenes, a straightforward approach is training the encoder directly via supervised learning. Interaction labels can be obtained either from human experts [123] or rule-based labeling functions [69]. However, labels for the interaction graph are insufficient to induce the decoder to synthesize the interactive behaviors suggested by the labels, as the model cannot capture the semantic meaning behind those interaction labels. Instead, we reframe relational inference as an IRL problem and introduce structured reward functions to ground the latent space. Concretely, we model the system as a multi-agent MDP, where the agents share a reward function that depends on the relational latent space. We design structured reward functions based on expert domain knowledge to explicitly define the interactive behaviors corresponding to the latent space. To solve the formulated IRL problem, we propose GRI, for which a VAE-like GNN in NRI [67] serves as the backbone model. Additionally, we incorporate the structured reward functions into the model as an additional reward decoder. A variational extension of the AIRL algorithm is derived to train all the modules simultaneously.

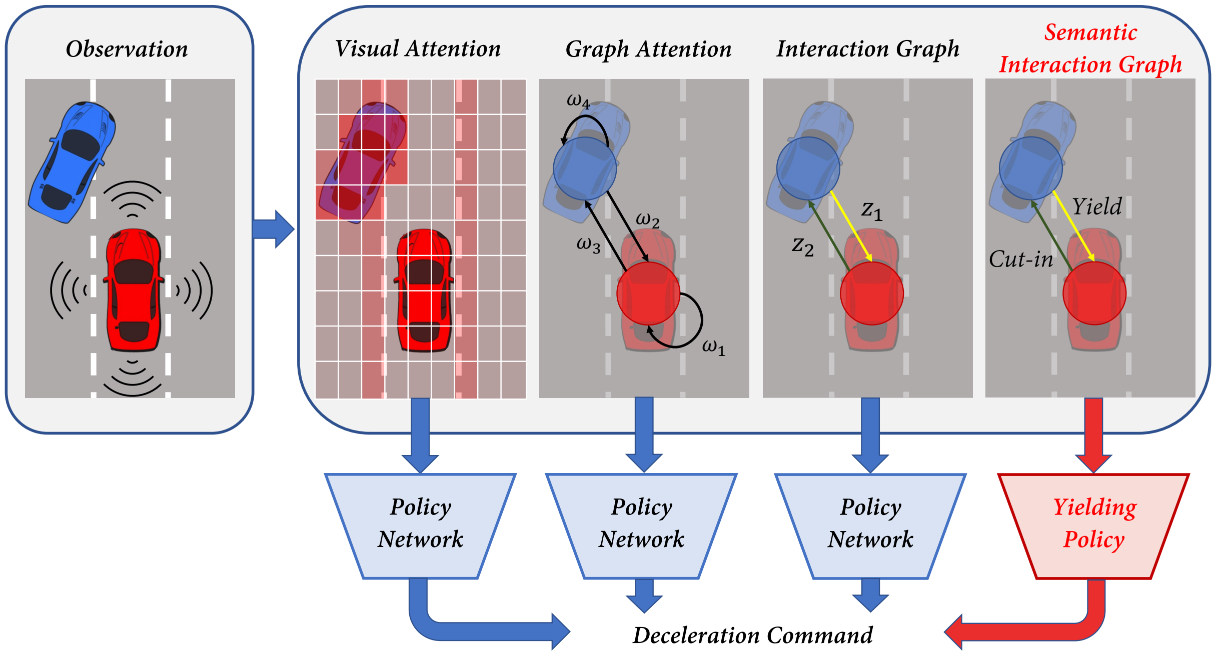


Figure 2.1: A motivating lane-changing scenario where we ask different models to control the red vehicle. All the models generate deceleration commands but have different intermediate outputs. With the aid of visual attention, we generate a heat map indicating the critical pixels of the input image. The graph attention network assigns edge weights ω_i to specify the importance of surrounding vehicles to the controlled vehicle. However, the attention mechanisms cannot recognize different effects—the two cars are mutually important but affect each other in distinct ways. The NRI model can distinguish between different interactive behaviors by assigning different values to the latent variables z_i in the interaction graph. Still, the latent space does not have explicit semantic meaning. In contrast, our model ensures a semantic interaction graph, which illustrates the model's understanding of the scenario and explains the action it takes. It determines the interaction graph with a latent space grounded in yielding and cutting-in behaviors. Further, it learns control policies that generate behaviors consistent with their definitions in domain knowledge (e.g., traffic rules) and executes the corresponding policies according to the inferred edge types.

Compared to direct supervision via interaction labels, we provide implicit supervision to GRI in terms of the structures of the reward functions. Since each reward function defines a type of interactive behavior, we confine the latent space to a cluster of interactive behaviors. This has two main advantages for supervision through labeling: 1) First, since the policy decoder learns to maximize the cumulative reward given the inferred interaction graph, the structured reward functions guide the policy to synthesize the corresponding semantic behaviors rather than simply mimicking the demonstrated trajectories; 2) Second, the end-to-end training scheme leaves the model to identify the underlying interaction graph of the observed trajectories and learn the characteristics of different behaviors (i.e., parameters of reward functions) from the data. This avoids the undesired bias introduced during the labeling procedure. Labels generated by human experts are subjective, and different people may interpret an interacting scenario in different ways. In contrast, there are systematic and principled ways to investigate what reward functions human behavior is subject to from data [89].

The rest of the chapter is organized as follows. In Sec. 2.2, we present a concise review of existing works that are closely related to ours in terms of methodology or motivation. In Sec. 2.3, we briefly summarize NRI and AIRL to prepare the readers for the core technical content. In Sec. 2.4, we introduce how we reformulate relational inference into a multi-agent IRL problem with relational latent space. In Sec. 2.5, we present the GRI model in a general context. In Sec. 2.6, we demonstrate how we can apply the proposed framework to model interactive traffic scenarios in both simulated and real-world settings. The experimental results show that the GRI can model interactive traffic scenarios and generate semantic interaction graphs that are consistent with both human domain knowledge and the modeled interactive behaviors.

2.2 Related Work

Our model combines GNNs and AIRL for interactive system modeling. This section provides a concise review of these two topics and summarizes the existing works that are closely related to ours. We also discuss some additional works on explainable driving models as a complement to the discussion in Sec. 2.1.

Interaction modeling using GNN

GNNs have been widely applied for interactive system modeling in recent years [120, 133, 8]. One category of models we find interesting includes those with a graph attention mechanism. One seminal model is the graph attention network (GAT) [135], which performs well on large-scale inductive classification problems. Vertex attention interaction network (VAIN) [53] applies attention in multi-agent modeling. The attention map unravels the interior interaction structure to some extent, thus improving the explainability of VAIN. An approach closely related to ours is NRI [67], which models the interaction structure explicitly with a discrete relational latent space compared to continuous graph attention. We explain the difference between NRI and our proposed method in Sec. 2.1 and Sec. 2.5. A related work in the autonomous driving domain is [69], which also modeled interactive driving behavior with semantically meaningful interactions but learned in a supervised manner.

Another type of model that is worth noting is the spatio-temporal graph (st-graph). An st-graph decomposes a complex problem into components and their spatio-temporal interactions, which are represented by nodes and edges of a factor graph. This makes st-graph a ubiquitous representation for interacting systems, e.g., human motion [58], human–robot interaction [83], and traffic flow [143]. Jain et al. [58] proposed a general method for transforming any st-graph into a mixture of recurrent neural networks (RNNs) called structural-RNN (S-RNN). When using a recurrent decoder, our GNN policy is similar to S-RNN, as it captures the same spatio-temporal dependency. In particular, Liu et al. [83] combined S-RNN with model-free RL to obtain a structured policy for robot crowd navigation. In terms of the underlying MDP, our GRI model is developed based on a multi-agent MDP, whereas theirs uses a single robot as the agent and regards the surrounding humans as parts of the environment. In addition, we adopt a structured reward function for each agent based on the graph and introduce a relational latent space into the MDP.

Adversarial IRL and Imitation Learning

Next, we present a brief review of related works on adversarial IRL. We also include works related to generative adversarial imitation learning (GAIL) [51], as it is closely connected to AIRL [31]. Both methods have generative adversarial networks (GANs) as backbone models and learn the discriminator through maximum entropy IRL. The difference is that GAIL uses an unstructured discriminator and does not use the generator's density.

Our work is mainly related to two types of methods: multi-agent and latent AIRL/GAIL algorithms. Yu et al. [145] proposed a multi-agent AIRL framework for Markov games under correlated equilibrium. It is capable of modeling general heterogeneous multi-agent interactions. The PS-GAIL algorithm [10] considers a multi-agent environment in the driving domain that is similar to ours—homogeneous agents with a shared policy under centralized control—and extended GAIL [51] to model interactive behaviors. In [11], the authors augmented the reward in PS-GAIL as a principle strategy for specifying prior knowledge, which is similar to the structured reward functions in GRI.

Latent AIRL models integrate a VAE into either the discriminator or the generator for different purposes. Wang et al. [138] conditioned the discriminator on the embeddings generated by a VAE trained separately using behavior cloning. The VAE encodes trajectories into low-dimensional space, enabling the generator to produce diverse behaviors from limited demonstration. VDB [97] constrains information contained in the discriminator's internal representation to balance the training procedure for adversarial learning algorithms. The PEMIRL framework [146] achieves meta-IRL by encoding demonstration into a contextual latent space. Though studied in different context, PEMIRL is conceptually similar to our framework as both its generator and discriminator depend on the inferred context variables.

2.3 Background

In this section, we briefly summarize two algorithms that are closely related to our approach in order to prepare the readers for the core technical content.

2.3.1 Neural Relational Inference

Kipf et al. [67] represent an interactive system with N interacting objects as a complete bi-directed graph:

$$\mathcal{G}_{\text{scene}} = (\mathcal{V}, \mathcal{E}),$$

where the vertices and edges are defined as:

$$\mathcal{V} = \{v_i\}_{i=1}^{N}, \ \mathcal{E} = \{e_{i,j} = (v_i, v_j) \mid i \neq j\}.$$

The edge $e_{i,j}$ refers to the one pointing from the vertex v_i to v_j. Each vertex corresponds to an object in the system. The NRI model is formalized as a VAE with a GNN encoder inferring the underlying interactions and a GNN decoder synthesizing the system dynamics given the interactions.

Formally, the model aims to reconstruct a given state trajectory, denoted by:

$$\mathbf{x} = \left(\mathbf{x}^0, \ldots, \mathbf{x}^{T-1}\right),$$

where T is the number of time steps and $\mathbf{x}^t = \{\mathbf{x}_1^t, \ldots, \mathbf{x}_N^t\}$. The vector $\mathbf{x}_i^t \in \mathbb{R}^n$ denotes the state vector of object v_i at time t. Alternatively, the trajectory can be decomposed into $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$, where $\mathbf{x}_i = \left\{\mathbf{x}_i^0, \ldots, \mathbf{x}_i^{T-1}\right\}$. The encoder operates over $\mathcal{G}_{\text{scene}}$, with $\mathbf{x}_i$ as the node feature of v_i. It infers the posterior distribution of the edge type $z_{i,j}$ for all the edges, collected into a single vector $\mathbf{z}$. The decoder operates over an interaction graph $\mathcal{G}_{\text{interact}}$ and reconstructs $\mathbf{x}$. The graph $\mathcal{G}_{\text{interact}}$ is constructed by assigning sampled $\mathbf{z}$ to the edges of $\mathcal{G}_{\text{scene}}$ and assigning the initial state to the nodes of $\mathcal{G}_{\text{scene}}$. If $\mathcal{G}_{\text{interact}}$ represents the interactions sufficiently, the decoder should be able to reconstruct the trajectory accurately.

The model is trained by maximizing the evidence lower bound (ELBO):

$$\mathcal{L} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\gamma(\mathbf{x}|\mathbf{z})\right] - D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) || p(\mathbf{z})\right],$$

where $q_\phi(\mathbf{z}|\mathbf{x})$ is the encoder output, which can be factorized as:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^{N} \prod_{j=1, j\neq i}^{N} q_\phi(z_{i,j}|\mathbf{x}), \tag{2.1}$$

where ϕ refers to the parameters of the encoder. The decoder output $p_\gamma(\mathbf{x}|\mathbf{z})$ can be written as:

$$p_\gamma(\mathbf{x}|\mathbf{z}) = \prod_{t=0}^{T-1} p_\gamma(\mathbf{x}^{t+1}|\mathbf{x}^t, \ldots, \mathbf{x}^0, \mathbf{z}),$$

where γ refers to the parameters of the decoder.

2.3.2 Adversarial Inverse Reinforcement Learning (AIRL)

The AIRL algorithm follows the principle of maximum entropy IRL [152]. Consider an MDP defined by $(\mathcal{X}, \mathcal{A}, \mathcal{T}, r)$, where $\mathcal{X}, \mathcal{A}$ are the state space and action space respectively. In the remainder of the chapter, we use $\mathbf{x}$ and $\mathbf{a}$ with any superscript or subscript to represent a state and action in $\mathcal{X}$ and $\mathcal{A}$. $\mathcal{T}$ is the transition operator given by $\mathbf{x}_{t+1} = f(\mathbf{a}_t, \mathbf{x}_t)$[1], and $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the reward function. The maximum entropy IRL framework assumes a suboptimal expert policy $\pi^{\mathrm{E}}(\mathbf{a}|\mathbf{x})$. The demonstration trajectories generated with the expert policy, $\mathcal{D}^{\mathrm{E}} = \left\{\boldsymbol{\tau}_1^{\mathrm{E}}, \dots \boldsymbol{\tau}_M^{\mathrm{E}}\right\}$, where $\boldsymbol{\tau}_i^{\mathrm{E}} = \left(\mathbf{x}_i^{\mathrm{E},0}, \mathbf{a}_i^{\mathrm{E},0}, \dots, \mathbf{x}_i^{\mathrm{E},T-1}, \mathbf{a}_i^{\mathrm{E},T-1}\right)$, have probabilities that increase exponentially with the cumulative reward. Concretely, they follow a Boltzmann distribution:

$$\boldsymbol{\tau}_i^{\mathrm{E}} \sim \pi^{\mathrm{E}}(\boldsymbol{\tau}) = \frac{1}{Z} \exp\left(\sum_{t=0}^{T-1} r_\lambda(\mathbf{x}_t, \mathbf{a}_t)\right),$$

where r_λ is the reward function with parameters denoted by λ. Maximum entropy IRL aims to infer the underlying reward function parameters of the expert policy. It is formalized as a maximum likelihood problem:

$$\lambda^* = \arg\max_\lambda \mathbb{E}_{\boldsymbol{\tau}^{\mathrm{E}} \sim \pi^{\mathrm{E}}(\boldsymbol{\tau})} \left[\sum_{t=0}^{T-1} r_\lambda(\mathbf{x}_t^{\mathrm{E}}, \mathbf{a}_t^{\mathrm{E}})\right] - \log Z.$$

To derive a feasible algorithm to solve the problem, we need to estimate the partition function Z. One practical solution is co-training a policy model with the current estimated reward function through reinforcement learning (RL) [30]. Finn et al. [31] found equivalency between it and a special form of the GAN. The policy model is the generator, whereas a structured discriminator is defined with the reward function to distinguish a generated trajectory $\boldsymbol{\tau}^{\mathrm{G}}$ from a demonstrated one $\boldsymbol{\tau}^{\mathrm{E}}$. Fu et al. [35] proposed the AIRL algorithm based on it, using a discriminator that identifies generated samples based on state–action pairs instead of the entire trajectory to reduce variance:

$$\mathcal{D}_{\lambda,\eta}(\mathbf{x}, \mathbf{a}) = \frac{\exp\left\{r_\lambda(\mathbf{x}, \mathbf{a})\right\}}{\exp\left\{r_\lambda(\mathbf{x}, \mathbf{a})\right\} + \pi_\eta(\mathbf{a}|\mathbf{x})}, \tag{2.2}$$

where $\pi_\eta(\mathbf{a}|\mathbf{x})$ is the policy model with parameters denoted by η. The models $\mathcal{D}_{\lambda,\eta}$ and π_η are trained adversarially by solving the following min-max optimization problem:

$$\min_\eta \max_\lambda \mathbb{E}_{\mathbf{x}^{\mathrm{E}}, \mathbf{a}^{\mathrm{E}} \sim \pi^{\mathrm{E}}(\mathbf{x}, \mathbf{a})} \left[\log\left(\mathcal{D}_{\lambda,\eta}(\mathbf{x}^{\mathrm{E}}, \mathbf{a}^{\mathrm{E}})\right)\right] + \mathbb{E}_{\mathbf{x}^{\mathrm{G}}, \mathbf{a}^{\mathrm{G}} \sim \pi_\eta(\mathbf{x}, \mathbf{a})} \left[\log\left(1 - \mathcal{D}_{\lambda,\eta}(\mathbf{x}^{\mathrm{G}}, \mathbf{a}^{\mathrm{G}})\right)\right], \tag{2.3}$$

where $\pi^{\mathrm{E}}(\mathbf{x}, \mathbf{a})$ denotes the distribution of state and action induced by the expert policy, and $\pi_\eta(\mathbf{x}, \mathbf{a})$ is the distribution induced by the learned policy.

[1]The transition is assumed to be deterministic to simplify the notation. A more general form of the algorithm can be derived for stochastic systems, which is essentially the same as the deterministic case.

2.4 Problem Formulation

Our GRI model grounds the relational latent space in a clustering of semantically meaningful interactions by reformulating the relational inference problem into a multi-agent IRL problem. Since the framework has the potential to be generalized to interactive systems in other domains besides autonomous driving, we will introduce our approach in a general tone. However, it should be noted that we limit our discussion to autonomous driving problems, without claiming that our approach can be directly applied to other domains. GRI relies on expert domain knowledge to identify all possible semantic behaviors and design the corresponding reward functions. There is a broad range of literature on interactive driving behavior modeling [123, 60], which we can refer to when designing the rewards. We can extend the proposed framework to other fields if proper domain knowledge is available, which is left for future investigation.

We start with modeling the interactive system as a multi-agent MDP with a graph representation. As in NRI, the system has an underlying interaction graph $\mathcal{G}_{\text{interact}}$. The discrete latent variable $z_{i,j}$ takes a value from $0, 1, \ldots, K-1$, where K is the number of interactions. It indicates the type of relation between v_i and v_j with respect to its effect on v_j. Additionally, we assume that the objects of the system are homogeneous intelligent agents who make decisions based on their interactions with others. Concretely, each agent is modeled with an identical state space $\mathcal{X}$, action space $\mathcal{A}$, transition operator $\mathcal{T}$, and reward function $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$. At time step t, the reward of agent v_j depends on the states and actions of itself and the pairwise interactions between itself and all its neighbors:

$$r_{\xi,\psi}(v_j^t, \mathbf{z}_j) = r_\xi^n(\mathbf{x}_j^t, \mathbf{a}_j^t) + \sum_{i \in \mathcal{N}_j} \sum_{k=1}^{K} \mathbf{1}(z_{i,j} = k) r_{\psi_k}^{e,k}(\mathbf{x}_i^t, \mathbf{a}_i^t, \mathbf{x}_j^t, \mathbf{a}_j^t), \tag{2.4}$$

where $\mathbf{z}_j$ is the collection of $\{z_{i,j}\}_{i \in \mathcal{N}_j}$, r_ξ^n is the node reward function parameterized by ξ, $\mathcal{N}_j$ is the set of v_j's neighboring nodes, $\mathbf{1}$ is the indicator function, and $r_{\psi_k}^{e,k}$ is the edge reward function parameterized by ψ_k for the k^{th} type of interaction. We utilize expert domain knowledge to design $r_{\psi_k}^{e,k}$, so the corresponding interactive behavior emerges by maximizing the rewards. In particular, the edge reward equals zero for $k = 0$, indicating that the action taken by v_j does not depend on its interaction with v_i. We assume the agents act cooperatively to maximize the cumulative reward of the system:

$$\mathcal{R}_{\xi,\psi}(\boldsymbol{\tau}, \mathbf{z}) = \sum_{t=0}^{T-1} \mathbf{r}_{\xi,\psi}\left(\mathbf{x}^t, \mathbf{a}^t, \mathbf{z}\right) = \sum_{t=0}^{T-1} \sum_{j=1}^{N} r_{\xi,\psi}\left(v_j^t, \mathbf{z}_j\right),$$

with a joint policy denoted by $\boldsymbol{\pi}_\eta\left(\mathbf{a}^t | \mathbf{x}^t, \mathbf{z}\right)$. The cooperative assumption is not necessarily valid for generic traffic scenarios [145], but it simplifies the training procedure significantly. We will leave the extension of the proposed method to non-cooperative interactive traffic scenarios for future work. Given a demonstration dataset, we aim to infer the underlying reward function and policy. Different from a typical IRL problem, both $r_{\xi,\psi}$ and π_η depend on $\mathbf{z}$. Therefore, we need to infer the distribution $p(\mathbf{z}|\boldsymbol{\tau})$ to solve the IRL problem.

2.5 Grounded Relational Inference Framework

We now present the GRI model to solve the IRL problem specified in Sec. 2.4. The model consists of three modules modeled by message-passing GNNs [37]: an encoder inferring the posterior distribution of edge types, a policy decoder generating control actions conditioned on the edge variables sampled from the posterior distribution, and a reward decoder modeling the rewards conditioned on the inferred edge types.

2.5.1 Architecture

The overall model structure is illustrated in Fig. 2.2. Given a demonstration trajectory $\boldsymbol{\tau}^{\mathrm{E}} \in \mathcal{D}^{\mathrm{E}}$, the encoder operates over $\mathcal{G}_{\text{scene}}$ and approximates the posterior distribution $p(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}})$ with $q_\phi(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}})$. The policy decoder operates over a $\mathcal{G}_{\text{interact}}$ sampled from the inferred $q_\phi(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}})$ and models the policy $\boldsymbol{\pi}_\eta\left(\mathbf{a}^t|\mathbf{x}^t,\mathbf{z}\right)$. Given an initial state, we can generate a trajectory by sequentially sampling $\mathbf{a}^t$ from $\boldsymbol{\pi}_\eta\left(\mathbf{a}^t|\mathbf{x}^t,\mathbf{z}\right)$ and propagating the state. The state is propagated with either the transition operator $\mathcal{T}$, if given, or a simulating environment if $\mathcal{T}$ is not accessible. We denote a generated trajectory given the initial state of τ^{E} as τ^{G}.

In terms of model structure, both the encoder and the policy decoder are built based on node-to-node message passing [37], consisting of node-to-edge message passing and edge-to-node message passing:

$$v \to e: \quad \mathbf{h}_{i,j}^l = f_e^l(\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{i,j}), \tag{2.5}$$

$$e \to v: \quad \mathbf{h}_j^{l+1} = f_v^l\left(\sum\nolimits_{i\in\mathcal{N}_j} \mathbf{h}_{i,j}^l, \mathbf{x}_j\right), \tag{2.6}$$

where $\mathbf{h}_i^l$ is the embedded hidden state of node v_i in the l^{th} layer, and $\mathbf{h}_{i,j}^l$ is the embedded hidden state of the edge $e_{i,j}$. The features $\mathbf{x}_i$ and $\mathbf{x}_{i,j}$ are assigned to the node v_i and the edge $e_{i,j}$, respectively, as inputs. $\mathcal{N}_j$ denotes the set of the indices of v_i's neighboring nodes connected by an incoming edge. The functions f_e^l and f_v^l are neural networks for edges and nodes, respectively, shared across the graph within the l^{th} layer of node-to-node message passing.

GNN Encoder

The GNN encoder is essentially the same as in NRI. It models the posterior distribution as $q_\phi(\mathbf{z}|\boldsymbol{\tau})$ with the following operations:

$$\begin{aligned}
\mathbf{h}_j^1 &= f_{\text{emb}}(\mathbf{x}_j),\\
v \to e: \quad \mathbf{h}_{i,j}^1 &= f_e^1(\mathbf{h}_i^1, \mathbf{h}_j^1),\\
e \to v: \quad \mathbf{h}_j^2 &= f_v^1\left(\sum\nolimits_{i\neq j} \mathbf{h}_{i,j}^1\right),\\
v \to e: \quad \mathbf{h}_{i,j}^2 &= f_e^2(\mathbf{h}_i^2, \mathbf{h}_j^2),\\
q_\phi(\mathbf{z}_{i,j}|\boldsymbol{\tau}) &= \operatorname{softmax}\left(\mathbf{h}_{i,j}^2\right),
\end{aligned}$$

where f_e^1, f_v^1 and f_e^2 are multilayer perceptrons (MLPs) and f_{emb} is a 1D convolutional network (CNN) with attentive pooling.

GNN Policy Decoder

The policy operates over $\mathcal{G}_{\text{interact}}$ and models the distribution $\boldsymbol{\pi}_\eta\left(\mathbf{a}^t|\mathbf{x}^t, \mathbf{z}\right)$, which can be factorized with $\pi_\eta\left(\mathbf{a}_j^t|\mathbf{x}^t, \mathbf{z}\right)$ as in Eqn. (2.1). We model π_η as a Gaussian distribution, with the mean value parameterized by the following GNN:

$$v \to e: \quad \tilde{\mathbf{h}}_{i,j}^t = \sum_{k=0}^{K} \mathbf{1}(z_{i,j} = k) \tilde{f}_e^k(\mathbf{x}_i^t, \mathbf{x}_j^t), \tag{2.7}$$

$$e \to v: \quad \mu_j^t = \tilde{f}_v\left(\sum\nolimits_{i \neq j} \tilde{\mathbf{h}}_{i,j}^t\right), \tag{2.8}$$

$$\pi_\eta\left(\mathbf{a}_j^t|\mathbf{x}^t, \mathbf{z}\right) = \mathcal{N}(\boldsymbol{\mu}_j^t, \sigma^2\mathbf{I}). \tag{2.9}$$

Alternatively, we can improve model capacity by using a recurrent policy denoted by $\pi_\eta\left(\mathbf{a}_j^t|\mathbf{x}^t, \ldots, \mathbf{x}^1, \mathbf{z}\right)$; Namely, the agents take actions according to the historical trajectories of the system. We follow the practice in [67] and add a gated recurrent unit (GRU) to obtain the following recurrent model:

$$v \to e: \quad \tilde{\mathbf{h}}_{i,j}^t = \sum_{k=0}^{K} \mathbf{1}(z_{i,j} = k) \tilde{f}_e^k\left(\tilde{\mathbf{h}}_i^t, \tilde{\mathbf{h}}_j^t\right), \tag{2.10}$$

$$e \to v: \quad \tilde{\mathbf{h}}_j^{t+1} = \text{GRU}\left(\sum\nolimits_{i \neq j} \tilde{\mathbf{h}}_{i,j}^t, \mathbf{x}_j^t, \tilde{\mathbf{h}}_j^t\right), \tag{2.11}$$

$$\mu_j^t = f_{\text{out}}\left(\tilde{\mathbf{h}}_j^{t+1}\right), \tag{2.12}$$

$$\pi_\eta\left(\mathbf{a}_j^t|\mathbf{x}^t, \ldots, \mathbf{x}^1, \mathbf{z}\right) = \mathcal{N}(\boldsymbol{\mu}_j^t, \sigma^2\mathbf{I}), \tag{2.13}$$

where $\tilde{\mathbf{h}}_i^t$ is the recurrent hidden state encoding the historical information up to time step $t-1$.

The reward decoder computes the reward of a state–action pair given the sampled edge variables. We use it to compute the cumulative rewards of $\boldsymbol{\tau}^{\text{G}}$ and $\boldsymbol{\tau}^{\text{E}}$ conditioned on the sampled $\mathcal{G}_{\text{interact}}$. The reward decoder is in the form of Eqn. (2.4). Additionally, we augment the functions r_ξ^n and $r_{\psi_k}^{e,k}$ with MLP shaping terms to mitigate the reward-shaping effect [35], resulting in:

$$f_{\xi,\omega}^n(\mathbf{x}_j^t, \mathbf{a}_j^t, \mathbf{x}_j^{t+1}) = r_\xi^n(\mathbf{x}_j^t, \mathbf{a}_j^t) + h_\omega^n(\mathbf{x}_j^{t+1}) - h_\omega^n(\mathbf{x}_j^t), \tag{2.14}$$

and

$$f_{\psi_k,\chi_k}^{e,k}(\mathbf{x}_i^t, \mathbf{a}_i^t, \mathbf{x}_i^{t+1}, \mathbf{x}_j^t, \mathbf{a}_j^t, \mathbf{x}_j^{t+1}) = r_{\psi_k}^{e,k}(\mathbf{x}_i^t, \mathbf{a}_i^t, \mathbf{x}_j^t, \mathbf{a}_j^t) + h_{\chi_k}^{e,k}(\mathbf{x}_i^{t+1}, \mathbf{x}_j^{t+1}) - h_{\chi_k}^{e,k}(\mathbf{x}_i^t, \mathbf{x}_j^t), \tag{2.15}$$

where h_ω^n and $h_{\chi_k}^{e,k}$ are MLPs with parameters denoted by ω and χ, respectively. We denote the shaped reward function of agent v_j by $\mathbf{f}_{\xi,\omega,\psi,\chi}\left(\mathbf{x}^t, \mathbf{a}^t, \mathbf{x}^{t+1}, \mathbf{z}\right)$, which is equal to the left-hand side of Eqn. (2.4) but with r_ξ^n and $r_{\psi_k}^{e,k}$ substituted by the augmented rewards. The

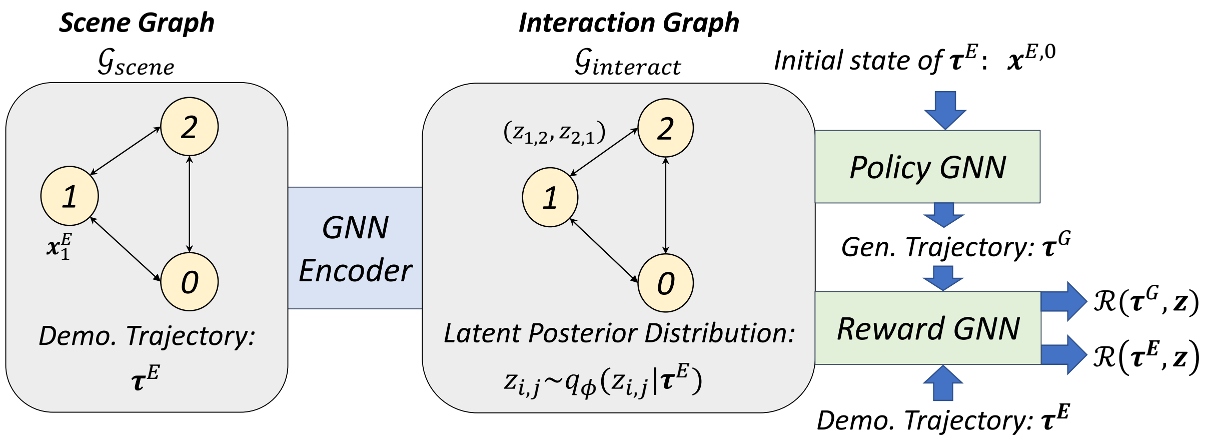


Figure 2.2: The Architecture of GRI. Given a demonstration trajectory $\boldsymbol{\tau}^{\mathrm{E}} \in \mathcal{D}^{\mathrm{E}}$, the encoder operates over $\mathcal{G}_{\text{scene}}$ and approximates the distribution $p(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}})$ with $q_\phi(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}})$. The policy decoder operates over a $\mathcal{G}_{\text{interact}}$ sampled from the inferred $q_\phi(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}})$ and models the policy $\boldsymbol{\pi}_\eta\left(\mathbf{a}^t|\mathbf{x}^t, \mathbf{z}\right)$. Given the initial state of $\boldsymbol{\tau}^{\mathrm{E}}$, we sample a trajectory $\boldsymbol{\tau}^{\mathrm{G}}$ by sequentially sampling $\mathbf{a}^t$ from $\boldsymbol{\pi}_\eta\left(\mathbf{a}^t|\mathbf{x}^t, \mathbf{z}\right)$ and propagating the state. Finally, we use the reward GNN to compute the cumulative rewards of $\boldsymbol{\tau}^{\mathrm{G}}$ and $\boldsymbol{\tau}^{\mathrm{E}}$ conditioned on the sampled $\mathcal{G}_{\text{interact}}$.

$$\begin{aligned}\max_{\eta} \min_{\xi,\omega,\psi,\chi,\phi} \mathbb{E}_{\boldsymbol{\tau}^{\mathrm{E}}\sim\boldsymbol{\pi}^{\mathrm{E}}(\boldsymbol{\tau})}\Bigg\{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}})}\Bigg[&-\sum_{t=0}^{T-1}\log\mathcal{D}_{\xi,\omega,\psi,\chi,\eta}(\mathbf{x}^{\mathrm{E},t}, \mathbf{a}^{\mathrm{E},t}, \mathbf{x}^{\mathrm{E},t+1}, \mathbf{z}) \\ &- \mathbb{E}_{\boldsymbol{\tau}^{\mathrm{G}}\sim\boldsymbol{\pi}_\eta(\boldsymbol{\tau}|\mathbf{z})}\sum_{t=0}^{T-1}\log\left(1-\mathcal{D}_{\xi,\omega,\psi,\chi,\eta}(\mathbf{x}^{\mathrm{G},t}, \mathbf{a}^{\mathrm{G},t}, \mathbf{x}^{\mathrm{G},t+1}, \mathbf{z})\right)\Bigg]\Bigg\}, \\ \text{s.t.} \quad \mathbb{E}_{\boldsymbol{\tau}^{\mathrm{E}}\sim\boldsymbol{\pi}^{\mathrm{E}}(\boldsymbol{\tau})}\left\{D_{KL}\left[q_\phi\left(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}}\right)\right) || p(\mathbf{z})\right]\right\} \leqslant I_c, &\end{aligned} \tag{2.16}$$

shaped reward function together with the policy model defines the discriminator, which distinguishes $\boldsymbol{\tau}^{\mathrm{G}}$ from $\boldsymbol{\tau}^{\mathrm{E}}$:

$$\mathcal{D}_{\xi,\omega,\psi,\chi,\eta}(\mathbf{x}^t, \mathbf{a}^t, \mathbf{x}^{t+1}, \mathbf{z}) = \frac{\exp\left\{\mathbf{f}_{\xi,\omega,\psi,\chi}\left(\mathbf{x}^t, \mathbf{a}^t, \mathbf{x}^{t+1}, \mathbf{z}\right)\right\}}{\exp\left\{\mathbf{f}_{\xi,\omega,\psi,\chi}\left(\mathbf{x}^t, \mathbf{a}^t, \mathbf{x}^{t+1}, \mathbf{z}\right)\right\} + \boldsymbol{\pi}_\eta\left(\mathbf{a}^t|\mathbf{x}^t, \mathbf{z}\right)}.$$

2.5.2 Training

We aim to train the three modules simultaneously. Consequently, we incorporate the encoder model $q_\phi\left(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}}\right)$ into the objective function of AIRL, resulting in the optimization problem (2.16). The encoder is integrated into the minimization problem because the reward function has a direct dependence on the latent space. The model is then trained by solving Problem (2.16) in an adversarial scheme: We alternate between training the encoder and reward for the minimization problem and training the policy for the maximization problem. Specifically, the objective for the encoder and reward is the following minimization problem given fixed η:

$$\begin{aligned} \min_{\xi,\omega,\psi,\chi,\phi} \quad & \mathcal{J}(\xi,\omega,\psi,\chi,\phi,\eta) \\ \text{s.t.} \quad & \mathbb{E}\left\{D_{KL}\left[q_\phi\left(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}}\right))||p(\mathbf{z})\right]\right\} \leqslant I_c, \end{aligned} \tag{2.17}$$

where $\mathcal{J}(\xi,\omega,\psi,\chi,\phi,\eta)$ is the objective function of Problem (2.16). The objective for the policy is maximizing $\mathcal{J}(\xi,\omega,\psi,\chi,\phi,\eta)$ with fixed ξ,ω,ψ,χ and ϕ.

The objective function in Problem (2.16) is essentially the expectation of the objective function in Problem (2.3) over the inferred posterior distribution $q_\phi\left(\boldsymbol{z}|\boldsymbol{\tau}^{\mathrm{E}}\right)$ and the demonstration distribution $\boldsymbol{\pi}^{\mathrm{E}}\left(\boldsymbol{\tau}\right)$. The constraint enforces an upper bound I_c on the Kullback–Leibler (KL) divergence between $q_\phi\left(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}}\right)$ and the prior distribution $p(\mathbf{z})$. A sparse prior is chosen to encourage sparsity in $\mathcal{G}_{\text{interact}}$. It has a similar regularization effect as the D_{KL} term in ELBO. We borrow its format from variational discriminator bottleneck (VDB) [97]. VDB improves adversarial training by constraining the information flow from the input to the discriminator. The KL divergence constraint is derived as a variational approximation of the information bottleneck [2]. Although having a different motivation, we adopt it for two reasons. First, the proposed model is not generative because our goal is not to synthesize trajectories from the prior $p(\mathbf{z})$ but to infer the posterior $p\left(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}}\right)$. Therefore, regularization derived from the information bottleneck is more sensible compared to ELBO. Second, the constrained problem (2.17) can be relaxed by introducing a Lagrange multiplier β. During training, β is updated through dual gradient descent as follows:

$$\beta \leftarrow \max\left(0, \alpha_\beta\left(\mathbb{E}\left\{D_{KL}\left[q_\phi\left(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}}\right))||p(\mathbf{z})\right]\right\} - I_c\right)\right) \tag{2.18}$$

We find the adaptation scheme particularly advantageous. The model can focus on inferring $\mathbf{z}$ for reward learning after satisfying the sparsity constraint because the magnitude of β decreases toward zero once the constraint is satisfied. However, it is worth noting that our framework does not rely on the bottleneck constraint to induce a semantically meaningful latent space as in [50]. In contrast, GRI relies on the structured reward functions to ground the latent space in semantic interactive behaviors. The bottleneck serves as a regularization to determine the minimal interaction graph to represent the interactions. In fact, we trained the baseline NRI models with the same constraints and weight update scheme. The experimental results show that the constraint itself is not sufficient to induce a sparse interaction graph.

In general, when the dynamics $\mathcal{T}$ are unknown or non-differentiable, maximum entropy RL algorithms [72] are adopted to optimize the policy. We assume known and differentiable dynamics, which is a reasonable assumption for the investigated scenarios. This allows us to directly back-propagate through the trajectory for gradient estimation, which simplifies the training procedure.

2.6 Experiments

We evaluate the proposed GRI model on a synthetic dataset as well as a naturalistic traffic dataset. The synthetic data are generated using policy models trained given the ground-truth reward function and interaction graph. We intend to verify whether GRI can induce a semantically meaningful relational latent space and infer the underlying relations precisely. The naturalistic traffic data are extracted from the Next Generation Simulation (NGSIM) dataset. We aim to validate whether GRI can model real-world traffic scenarios effectively with the grounded latent space. Unlike synthetic agents, we cannot access the ground-truth graphs governing human drivers' interactions. Instead, we construct hypothetical graphs after analyzing the segmented data. The hypotheses reflect humans' understanding of the traffic scenarios. Moreover, the hypothetical graphs are built upon a set of interactive behaviors whose characteristics are described by the designed reward functions. We would like to see if the reward functions can incorporate the semantic information into the latent space and allow GRI to model real-world interactive systems in the same way as humans. In each setting, we consider two traffic scenarios: car-following and lane-changing scenarios.

2.6.1 Baselines

The main question of interest is whether GRI can induce semantically meaningful interaction graphs. To answer this question, the most important baseline model for comparison is NRI, as GRI shares the same prior distribution of latent variables with NRI. Comparing the posterior distributions provides insights on whether the structured reward functions can ground the latent space in semantic interactive behaviors. In each experiment, the baseline NRI model has the same encoder and policy decoder as the GRI model. Additionally, as stated in Sec. 2.5, the same bottleneck constraint and the weight update scheme in Eqn. (2.18) were applied as regularization for minimal representation.

Another model for comparison is a supervised policy decoder. We assume that the ground-truth graphs or human hypotheses are available. Therefore, we can directly train a policy decoder in a supervised way. The ground-truth graph is fed to the policy decoder as a substitute for the interaction graph sampled from the encoder output $q_\phi(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}})$. The training of the decoder becomes a simple regression problem. We used mean square error as the loss function to train it. As additional information is provided, it is unfair to directly compare the performance of GRI with the supervised policy model. Since the supervised model is trained with the ground-truth interaction graphs governing the systems, it is expected to have a smaller reconstruction error. However, the supervised baseline provides some useful insights. In the naturalistic traffic scenarios, the supervised model offers more information regarding whether the human hypotheses are reasonable. If the supervised model can reconstruct the trajectories precisely, it will justify our practice to adopt graph accuracy as one of the evaluation metrics.

More importantly, in Sec. 2.6.5, we demonstrate that GRI's latent space maintains its semantic meaning under some perturbations to the initial states, whereas the decoders of the

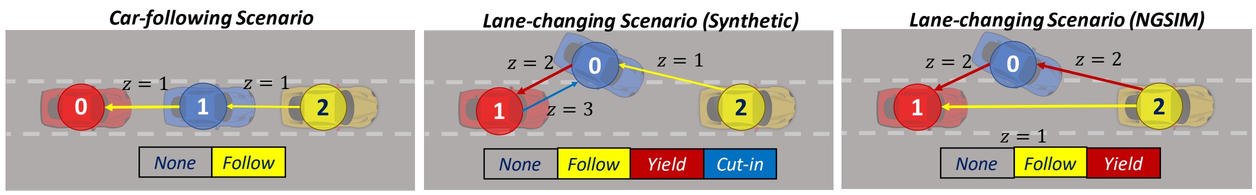


Figure 2.3: Test scenarios with the underlying interaction graphs. In the synthetic scenarios, the graphs are the ground-truth ones governing the synthetic experts. In the naturalistic traffic scenarios, the graphs are human hypotheses reflecting human understanding of the traffic scenarios.

baseline models fail to synthesize those behaviors under the same perturbations, including the supervised policy decoder, which is trained with the ground-truth interaction graphs. This supports our argument that direct supervision via interaction labels is not sufficient to guide the policy to synthesize behaviors with correct semantic meaning.

There are alternative methods for trajectory reconstruction. However, it is not our goal to find an expressive model for accurate reconstruction. Therefore, we do not consider other baselines from this perspective. For the task of grounding the latent space in semantic interactive driving behaviors, we did not find any exact alternatives in the literature. For the specific scenarios studied, we may design some rule-based approaches to directly infer the interaction graph. However, it is difficult to determine the parameters that best describe the interactive behaviors, as there is a spectrum in how people follow the rules [71]. We are interested in a data-driven module that can be incorporated into an end-to-end learning model and has the potential to be generalized to complicated driving scenarios and systems in other domains. Apart from GRI, a potential alternative solution could be adopting a differentiable logic module. For instance, Leung et al. [71] proposed a differentiable parametric signal temporal logic formula (pSTL) that could be learned from data. We will investigate this in our future works.

2.6.2 Evaluation Metrics

To evaluate a trained model, we sample a $\boldsymbol{\tau}^{\mathrm{E}}$ from the test dataset and extract the maximum posterior probability (MAP) estimate of edge variables, $\hat{\mathbf{z}}$, from $q_\phi(\mathbf{z}|\boldsymbol{\tau}^{\mathrm{E}})$. Subsequently, we obtain a single sample of trajectories $\hat{\boldsymbol{\tau}}$ by executing the mean value of the policy output. The root mean square errors (RMSE) of the states and the accuracy of $\mathcal{G}_{\text{interact}}$ are selected as the evaluation metrics, which are computed based on $\hat{\mathbf{z}}$, $\hat{\boldsymbol{\tau}}$, $\boldsymbol{\tau}^{\mathrm{E}}$, and the ground truth or hypothetical latent variables denoted by $\mathbf{z}^{\mathrm{E}}$:

$$\mathrm{RMSE}_{\epsilon} = \sqrt{\frac{1}{(N-1)T}\sum_{j=1}^{N}\sum_{t=0}^{T-1}(\epsilon_j^{\mathrm{E},t} - \hat{\epsilon}_j^t)^2},$$

$$\mathrm{Accuracy} = \frac{\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\mathbf{1}(z_{i,j}^{\mathrm{E}} = \hat{z}_{i,j})}{N(N-1)}.$$

If multiple edge types exist, we test all the possible permutations of edge types and report the one with the highest graph accuracy for NRI.

It is worth noting that the graph accuracy of the naturalistic traffic dataset merely quantifies the divergence between the inferred graphs and the hypotheses we construct. We anticipate that GRI can achieve a higher accuracy than NRI. This would imply that we can incorporate human domain knowledge into GRI and induce a semantic relational latent space consistent with the hypotheses built upon the same domain knowledge. However, a low graph accuracy does not necessarily mean that humans cannot interpret the inferred graphs well. The hypothetical graphs represent one perspective for interpreting the interactive scenes. It is possible that NRI may find another sensible way to categorize and interpret the interactions, which can also be understood by humans.

To further study the interpretability of the learned latent spaces, we consider the inferred graphs and make a qualitative comparison between the latent spaces learned by the two models. For each setting, we compute the distribution of estimated edge variables $\hat{\mathbf{z}}$ over the test dataset. As in [67], we visualize the results in multiple adjacency matrices corresponding to different edge types. In the adjacency matrix corresponding to the k^{th} type of interaction, the element $A_{i,j}$ indicates the relative frequency of $\hat{z}_{j,i} = k$, where $\hat{z}_{j,i}$ is the latent variable for the edge from node j to node i. In other words, $A_{i,j}$ equals the ratio of test samples in which the model infers $\hat{z}_{j,i} = k$. By inspecting the edge type distributions, we can obtain insight into the interpretability of the two models beyond the quantitative metrics.

2.6.3 Synthetic Scenes

As mentioned above, we designed two synthetic scenarios, car-following and lane-changing scenarios. The two scenes and their underlying interaction graphs are illustrated in Fig. 2.3. In both scenarios, there is a leading vehicle whose behavior does not depend on the others. Its trajectory is given without the need for reconstruction. We assume it runs at constant velocity. The other vehicles interact with each other and the leader in different ways. In the car-following scene, we model the system with two types of edges: $z_{i,j} = 1$ means that Vehicle j follows Vehicle i; $z_{i,j} = 0$ means that Vehicle j does not interact with Vehicle i. In the lane-changing scene, two additional edge types are introduced: $z_{i,j} = 2$ means that Vehicle j yields to Vehicle i; $z_{i,j} = 3$ means that Vehicle j cuts in front of Vehicle i.

The MDPs for the tested scenarios are specified as follows. In the car-following scene, since the vehicles mainly interact in a longitudinal direction, we only model their longitudinal dynamics to simplify the problem. For all $j \in \{1, 2, 3\}$, the state vector of Vehicle j consists of three states: $\mathbf{x}_j^t = \left[x_j^t \; v_j^t \; a_j^t\right]^\intercal$, where x_j^t is the longitudinal coordinate, v_j^t is the velocity, and a_j^t is the acceleration. There is only one control input, which is the jerk. We denote it as δa_j^t. The dynamics are governed by a one-dimensional (1D) point-mass model:

$$\begin{aligned} x_j^{t+1} &= x_j^t + v_j^t \Delta t + \frac{1}{2} a_j^t \Delta t^2, \\ v_j^{t+1} &= v_j^t + a_j^t \Delta t, \\ a_j^{t+1} &= a_j^t + \delta a_j^t \Delta t, \end{aligned}$$

where Δt is the sampling time. In the lane-changing scene, we consider both longitudinal and lateral motions. The state vector consists of six states instead: $\mathbf{x}_j^t = \left[x_j^t \; y_j^t \; v_j^t \; \theta_j^t \; a_j^t \; \omega_j^t\right]^\intercal$. The three additional states are the lateral coordinate y_j^t, the yaw angle θ_j^t, and the yaw rate ω_j^t. There is one additional action, which is the yaw acceleration, denoted by $\delta \omega_j^t$. We model the vehicle as a Dubins' car:

$$\begin{aligned} x_j^{t+1} &= x_j^t + v_j^t \cos(\theta_j^t) \Delta t, \\ y_j^{t+1} &= y_j^t + v_j^t \sin(\theta_j^t) \Delta t, \\ v_j^{t+1} &= v_j^t + a_j^t \Delta t, \\ \theta_j^{t+1} &= \theta_j^t + \omega_j^t \Delta t, \\ a_j^{t+1} &= a_j^t + \delta a_j^t \Delta t, \\ \omega_j^{t+1} &= \omega_j^t + \delta \omega_j^t \Delta t. \end{aligned}$$

The structured reward functions were designed based on expert domain knowledge (e.g., transportation studies [60, 131]). We mainly referred to [123, 89] in this work. The reward

Table 2.1: Performance Comparison on Synthetic Dataset

Model	Car Following ($\Delta t = 0.2s$, $T = 20$)			
	RMSE$_x$(m)	RMSE$_y$(m)	RMSE$_v$(m/s)	Accuracy(%)
GRI	0.241 ± 0.125	-	0.174 ± 0.068	$\mathbf{100.00 \pm 0.00}$
NRI	0.047 ± 0.024	-	0.056 ± 0.015	66.70 ± 0.00
Supervised	$\mathbf{0.039 \pm 0.016}$	-	$\mathbf{0.050 \pm 0.009}$	-
Model	**Lane Changing ($\Delta t = 0.2s$, $T = 30$)**			
	RMSE$_x$(m)	RMSE$_y$(m)	RMSE$_v$(m/s)	Accuracy(%)
GRI	0.529 ± 0.230	0.207 ± 0.046	0.303 ± 0.128	$\mathbf{99.95 \pm 0.01}$
NRI	0.109 ± 0.045	0.155 ± 0.038	0.061 ± 0.016	55.9 ± 7.98
Supervised	$\mathbf{0.062 \pm 0.027}$	$\mathbf{0.145 \pm 0.035}$	$\mathbf{0.048 \pm 0.011}$	-

[1] The data are presented in the form of mean ± std.

function of the car-following behavior is defined as follows:

$$\begin{aligned} r_{\psi_1}^{e,1}\left(\mathbf{x}_i^t, \mathbf{x}_j^t\right) =& -\left(1+\exp(\psi_{1,0})\right) g_{\text{IDM}}(\mathbf{x}_i^t, \mathbf{x}_j^t) \\ & -\left(1+\exp(\psi_{1,1})\right) g_{\text{dist}}(\mathbf{x}_i^t, \mathbf{x}_j^t) \\ & -\left(1+\exp(\psi_{1,2})\right) g_{\text{lat}}(\mathbf{x}_i^t, \mathbf{x}_j^t), \end{aligned}$$

where the features are defined as:

$$g_{\text{IDM}}(\mathbf{x}_i^t, \mathbf{x}_j^t) = \left(\max\left(x_i^t - x_j^t, 0\right) - \Delta x_{i,j}^{\text{IDM},t}\right)^2, \tag{2.19}$$

$$g_{\text{dist}}(\mathbf{x}_i^t, \mathbf{x}_j^t) = \exp\left(-\frac{\left(\max\left(x_i^t - x_j^t, 0\right)\right)^2}{\zeta^2}\right), \tag{2.20}$$

$$g_{\text{lat}}(\mathbf{x}_i^t, \mathbf{x}_j^t) = \left(y_j^t - g_{\text{center}}(y_i^t)\right)^2.$$

The feature g_{IDM} suggests a spatial headway $\Delta x_{i,j}^{\text{IDM},t}$ derived from the intelligent driver model (IDM) [60]. The feature f_{dist} ensures a minimum collision-free distance. We penalize the following vehicle for surpassing the preceding one with the help of $x_{i,j}^{\text{IDM},t}$ in Eqn. (2.19) and Eqn. (2.20). The last feature g_{lat} exists only in the lane-changing scenario. It regulates the following vehicle to stay in the same lane as the preceding one with the help of g_{center}, which determines the lateral coordinate of the corresponding centerline based on the position of the preceding vehicle. Altogether, the features define the following behavior as staying in the same lane as the preceding vehicle while maintaining a safe longitudinal headway.

The reward function for yielding is defined as:

$$\begin{aligned} r_{\psi_2}^{e,2}\left(\mathbf{x}_i^t, \mathbf{x}_j^t\right) =& -\left(1+\exp(\psi_{2,0})\right) g_{\text{yield}}(\mathbf{x}_i^t, \mathbf{x}_j^t) \\ & -\left(1+\exp(\psi_{2,1})\right) g_{\text{dist}}(\mathbf{x}_i^t, \mathbf{x}_j^t). \end{aligned}$$

The feature g_{dist} is defined in Eqn. (2.20). The other feature g_{yield} suggests an appropriate spatial headway for yielding:

$$\begin{aligned} g_{\text{yield}}(\mathbf{x}_i^t, \mathbf{x}_j^t) =& \mathbf{1}\left(g_{\text{center}}(y_j^t) = g_{\text{center}}(y_i^t)\right) g_{\text{IDM}}(\mathbf{x}_i^t, \mathbf{x}_j^t) \\ & + \mathbf{1}\left(g_{\text{center}}(y_j^t) \neq g_{\text{center}}(y_i^t)\right) g_{\text{goal}}(\mathbf{x}_i^t, \mathbf{x}_j^t), \\ g_{\text{goal}}(\mathbf{x}_i^t, \mathbf{x}_j^t) =& \left(\max\left(x_i^t - x_j^t - \Delta x^{\text{yield}}, 0\right)\right)^2. \end{aligned} \tag{2.21}$$

The suggested headway is set to a constant value, Δx^{yield}, when the other vehicle is merging, and it switches to $\Delta x_{i,j}^{\text{IDM},t}$ once the merging vehicle enters into the same lane, where its behavior becomes consistent with car that is following. We follow [123] and adopt different reward functions depending on the lanes in which the vehicles are located. Merging occurs during a short period of time. Therefore, we assume the driver sets a fixed short-term goal distance as in [123] and then transits to following behavior afterwards.

The reward function for cutting-in is similar:

$$\begin{aligned} r^{e,3}_{\psi_3}\left(\mathbf{x}^t_i, \mathbf{x}^t_j\right) = &-\left(1+\exp(\psi_{3,0})\right) g_{\text{goal}}(\mathbf{x}^t_j, \mathbf{x}^t_i) \\ &-\left(1+\exp(\psi_{3,1})\right) g_{\text{dist}}(\mathbf{x}^t_j, \mathbf{x}^t_i), \end{aligned}$$

where the features are defined as in Eqn. (2.20) and Eqn. (2.21) but with the input arguments switched, as the merging vehicle should stay in front of the yielding one.

Apart from the edge rewards, all the agents share the same node reward function. The following one is adopted for lane changing:

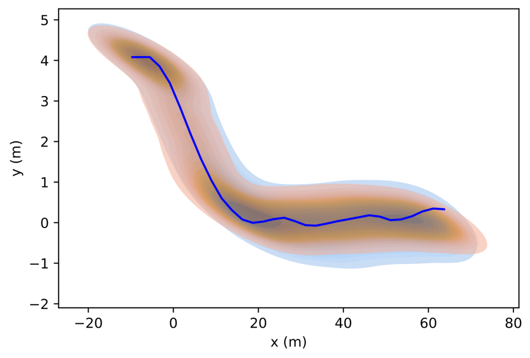
$$\begin{aligned} r^n_\xi(\mathbf{x}^t_j, \mathbf{a}^t_j) = &-\left(1+\exp(\xi_0)\right) f_v(\mathbf{x}^t_j) \\ &-\left(1+\exp(\xi_{1:3})\right)^\intercal f_{\text{state}}(\mathbf{x}^t_j) \\ &-\left(1+\exp(\xi_{4:5})\right)^\intercal f_{\text{action}}(\mathbf{a}^t_j) \\ &-\left(1+\exp(\xi_6)\right) f_{\text{lane}}(\mathbf{x}^t_j), \end{aligned}$$

where f_{state} and f_{action} take the element-wise square of $\left[a^t_j\ \theta^t_j\ \omega^t_j\right]$ and $\left[\delta a^t_j\ \delta\omega^t_j\right]$, respectively. It penalizes large control inputs as well as drastic longitudinal and angular motions to induce smooth and comfortable maneuvers. The feature f_v is the squared error between v^t_j and the speed limit v_{lim}. It regulates the vehicles to obey the speed limit. The last term f_{lane} penalizes the vehicle for staying close to the lane boundaries. For car following, we simply remove the terms that are irrelevant in 1D motion. In all the reward functions, the parameters collected in ψ and ξ are unknown during training and inferred by GRI. We take their exponents and add one to the results. This requires the model to use the features when modeling the corresponding interactions.

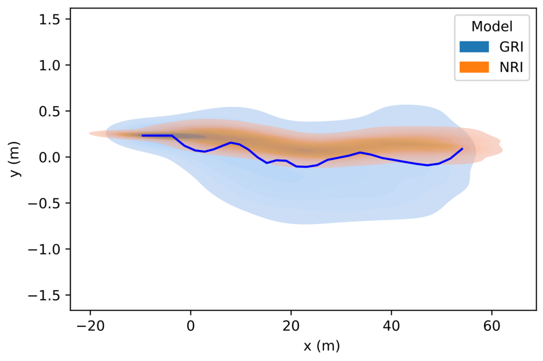
For the scenarios defined above, we aimed to generate one dataset for each scenario. For each scenario, we randomly sampled the initial states of the vehicles and trained an expert policy given the ground-truth reward functions and the interaction graph. Then, we used the trained policy to generate the dataset. The same sampling scheme was used to sample the initial states.

Results

We trained a GRI model with the policy decoder (2.7)-(2.9) on each dataset. The results are summarized in Table 2.1. The NRI model can reconstruct the trajectories with errors close to the supervised policy. However, it learns a relational latent space that is different from the one governing the demonstration. Therefore, the edge variables cannot be interpreted as those semantic interactive behaviors. In contrast, our GRI model interprets the interactions consistently with the domain knowledge inherited in the demonstration and recovers the interaction graph with high accuracy. However, it has larger reconstruction errors compared to the baseline approaches. To better understand this performance gap in reconstruction, we examined the reconstructed trajectories of both models. Instead of executing the mean value of the policy output, we sampled the actions from the policy distribution to



(a) Car 0



(b) Car 1

Figure 2.4: Visualization of the reconstructed trajectories in a lane-changing scene. (a) and (b) correspond to the trajectories of Car 1 and Car 0, respectively. We visualize the distributions of the reconstructed trajectories estimated using the kernel density estimate. The ground-truth trajectories are denoted by the blue curves.

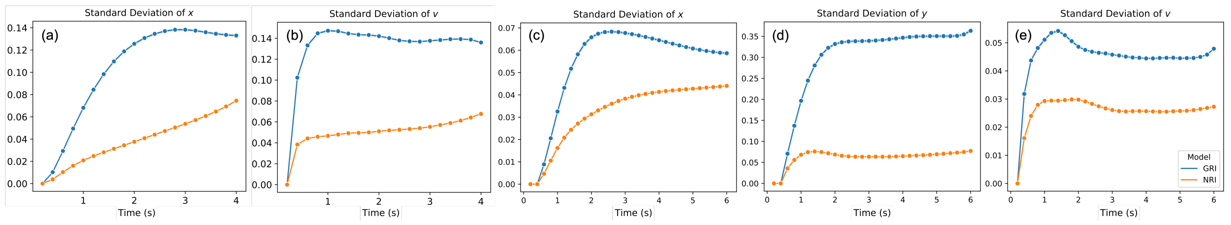


Figure 2.5: Average standard deviation of states along the time horizon. (a) and (b) show the standard deviation of x and v in the synthetic car-following scenario. (c)–(e) show the standard deviation of x, y, and v in the synthetic lane-changing scenario.

estimate the variance of reconstructed trajectories. In Fig. 2.5, we plot the average standard deviation of reconstructed states along the time horizon. We observed that the GRI policy decoder tends to have a larger variance. This partially explains the large RMSE values reported in Table 2.1: the metrics were computed with a single reconstructed trajectory. The policy distribution of GRI still has larger bias than that of NRI. We visualize the reconstructed trajectories of a lane-changing case in Fig. 2.4. While the GRI policy induces larger variance, the distribution of the reconstructed trajectories is sensible, which means that GRI can still sufficiently recover the interactive behaviors.

We computed the empirical distribution of the estimated edge variables $\hat{z}$ over the test dataset. The results are shown in Fig. 2.6. The distribution concentrates into a single interaction graph for both models in both scenarios—as opposed to the case on the naturalistic traffic dataset introduced in the next section—because the synthetic agents have consistent interaction patterns over all the samples. We observe that NRI learns symmetric relations: In both scenarios, the NRI model assigns the same edge types to the edges $e_{0,1}$ and $e_{1,0}$. It is difficult to interpret their semantic meaning because those pairwise interactions are asymmetric in our synthetic scenes. In contrast, the reward functions in our GRI model enforce an asymmetric relational latent space.

2.6.4 Naturalistic Traffic Scenes

To evaluate the proposed method in real-world traffic scenarios, we investigated the same scenarios as in the synthetic case, car-following and lane-changing scenarios. We segmented data from the Highway-101 and I-80 NGSIM datasets. Subsequently, we further screened the data to select the interactive samples and ensure that no erratic swerving or multiple lane changes occur. Unlike the synthetic agents, human agents do not have a ground-truth interaction graph that governs their interactions. Instead, we constructed hypothetical $\mathcal{G}_{\text{interact}}$ after analyzing the segmented data. The hypotheses for the two scenarios are depicted in Fig. 2.3. The one for car following is identical to the ground-truth interaction graph we designed for the synthetic agents. However, we proposed a different hypothesis for lane changing. We excluded the cutting-in relation to reduce the number of edge types and therefore simplify the training procedure. Moreover, we differentiated distinct interactions according to the vehicles' lateral position. We say that a vehicle yields to its preceding vehicle if it drives in neighboring lanes, whereas it follows the preceding one if it drives in the same lane.

As in the synthetic scenes, the trajectory of the leading vehicle is given without the need for reconstruction. We feed the ground-truth state of the leading vehicle sequentially to the policy decoder when decoding the trajectories of the other vehicles. This practice enables us to heuristically isolate a small interacting group from the large number of vehicles on the highway. While the leading vehicle's behavior depends on the other vehicles, it is fairly reasonable to assume that the behavior of the modeled following vehicles is independent of that of other surrounding vehicles on the road after conditioning on the trajectory of the leading vehicle. Even though there may still be other surrounding vehicles interacting with them, their influence should be subtle. The models should be able to capture the interactions among the modeled subset effectively while marginalizing out those subtle effects.

The node dynamics are the same as in the synthetic car-following scene. For the lane-changing scenario, since we did not have accurate heading information, we adopted a two-dimensional (2D) point-mass model instead. As the behavior of human drivers is much more complicated than that of the synthetic agents, we designed reward functions with larger model capacity using neural networks. In the car-following scenario, the reward functions

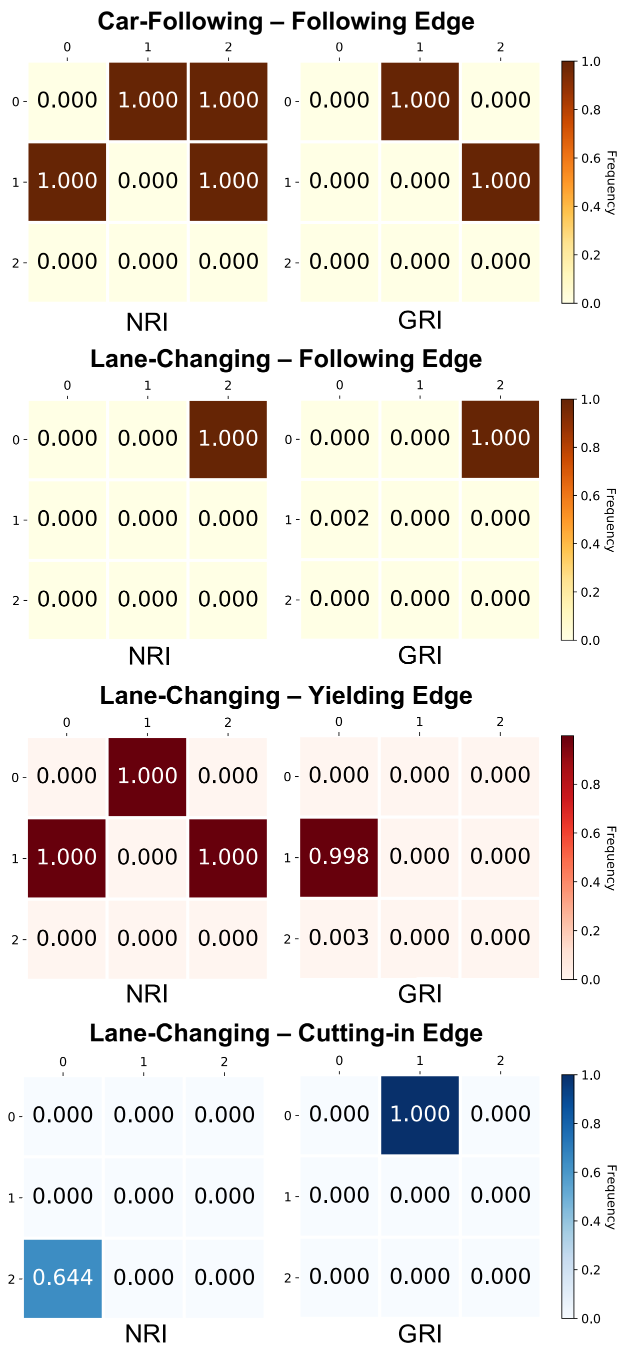


Figure 2.6: The empirical distribution of estimated edge variables $\hat{z}$ over the test dataset in the synthetic scenarios. We summarize the results in multiple adjacency matrices corresponding to different edge types. In the adjacency matrix corresponding to the k^{th} type of interaction, the element $A_{i,j}$ indicates the relative frequency of $\hat{z}_{j,i} = k$, where $\hat{z}_{j,i}$ is the latent variable for the edge from node j to node i.

are defined as follows:

$$\begin{aligned} r^{e,1}_{\psi_1}\left(\mathbf{x}^t_i, \mathbf{x}^t_j\right) &= -\left(1+\exp(\psi_{1,0})\right) g^{\mathrm{NN}}_{\mathrm{v}}(\mathbf{x}^t_i, \mathbf{x}^t_j) \\ &\quad -\left(1+\exp(\psi_{1,1})\right) g^{\mathrm{NN}}_{\mathrm{s}}(\mathbf{x}^t_i, \mathbf{x}^t_j), \\ r^{n}_{\xi}\left(\mathbf{x}^t_j, \mathbf{a}^t_j\right) &= -\left(1+\exp(\xi_0)\right) f^{\mathrm{NN}}_{\mathrm{v}}(\mathbf{x}^t_j) \\ &\quad -\left(1+\exp(\xi_1)\right) f_{\mathrm{acc}}(\mathbf{x}^t_j) \\ &\quad -\left(1+\exp(\xi_2)\right) f_{\mathrm{jerk}}(\mathbf{x}^t_j, \mathbf{a}^t_j), \end{aligned}$$

where the features are defined as:

$$\begin{aligned} f^{\mathrm{NN}}_{\mathrm{v}}(\mathbf{x}^t_j) &= \left(v^t_j - h_1(\mathbf{x}^t_j)\right)^2, \\ g^{\mathrm{NN}}_{\mathrm{v}}(\mathbf{x}^t_i, \mathbf{x}^t_j) &= \left(v^t_j - h_2(\mathbf{x}^t_i, \mathbf{x}^t_j)\right)^2, \\ g^{\mathrm{NN}}_{\mathrm{s}}(\mathbf{x}^t_i, \mathbf{x}^t_j) &= \mathrm{ReLU}\left(h_3\left(\mathbf{x}^t_i, \mathbf{x}^t_j\right) - x^t_i + x^t_j\right)^2. \end{aligned}$$

The features f_{acc} and f_{jerk} penalize the squared magnitude of acceleration and jerk to induce smooth and comfortable maneuvers. The functions h_1, h_2, and h_3 are neural networks with rectified linear unit (ReLU) output activation. The feature $g^{\mathrm{NN}}_{\mathrm{s}}$ is the critical component that shapes the car-following behavior. It learns a non-negative reference headway and penalizes the following vehicle for violating it. The features $g^{\mathrm{NN}}_{\mathrm{v}}$ and $f^{\mathrm{NN}}_{\mathrm{v}}$ suggest reference velocities considering interaction and themselves, respectively. The edge reward function has a large modeling capacity because we allow it to learn the adaptive reference headway and velocity from data. Nevertheless, it still defines the fundamental characteristic of the following behavior, which is always staying behind the preceding vehicle.

In lane changing, the node reward function and the edge reward function for the following behavior are similar to those in the car-following scenario. The node reward function has an additional term for lateral position, which encourages the vehicles to drive in the target lane, i.e., the lane in which the leading vehicle is driving. It also has additional terms to penalize the magnitude of lateral velocity and acceleration to induce comfortable maneuvers. To design the yielding reward, we define a collision point of two vehicles based on their states. We approximate the vehicles' trajectories as piecewise-linear between sequential time steps and compute the collision point as the intersection between their trajectories (Fig. 2.7). We threshold the point if it exceeds a hard-coded range of interest (e.g., if it is behind the vehicles or greater than a certain distance). Next, we define the distance to collision (d_{poc}) as the longitudinal distance from the vehicle to the collision point and the time to collision (T_{col}) as the time to reach the collision point, which is calculated by dividing d_{poc} by the velocity of the vehicle. Then, the yielding reward function is defined as follows:

$$\begin{aligned} r^{e,2}_{\psi_2}\left(\mathbf{x}^t_i, \mathbf{x}^t_j\right) &= -\left(1+\exp(\psi_{2,0})\right) g^{\mathrm{NN}}_{\mathrm{spatial}}(\mathbf{x}^t_i, \mathbf{x}^t_j) \\ &\quad -\left(1+\exp(\psi_{2,1})\right) g^{\mathrm{NN}}_{\mathrm{time}}(\mathbf{x}^t_i, \mathbf{x}^t_j), \end{aligned}$$

where

$$\begin{aligned} g^{\mathrm{NN}}_{\mathrm{spatial}}(\mathbf{x}^t_i, \mathbf{x}^t_j) &= \mathrm{ReLU}\left((x_j - x_{poc}) - h_{\mathrm{d_{poc}}}(\mathbf{x}^t_i, \mathbf{x}^t_j)\right)^2, \\ g^{\mathrm{NN}}_{\mathrm{time}}(\mathbf{x}^t_i, \mathbf{x}^t_j) &= \mathrm{ReLU}\left(h_{\mathrm{T_{col}}}(\mathbf{x}^t_i, \mathbf{x}^t_j) - (T_{col_i} - T_{col_j})\right)^2. \end{aligned}$$

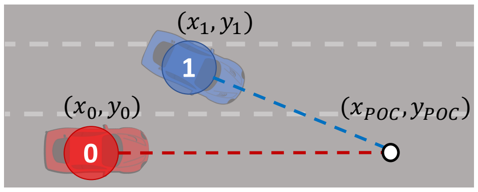


Figure 2.7: Collision point diagram. At every time step, the heading vector of the agents can be calculated by approximating the motion as linear. The intersection between these vectors is taken to be the collision point where the agents would collide if a yield action is not taken.

The functions $h_{\mathrm{d_{poc}}}$ and $h_{\mathrm{T_{col}}}$ are neural networks with ReLU output activation. The g_{spatial} term learns a spatial aspect of the yield behavior and compares the agent's distance from the estimated collision point with the NN-learned *safe* reference within which the lane-changing maneuver can be done. The second term g_{time} adds a temporal aspect by enforcing the vehicle to ensure a minimum *safe* time headway. We adopt g_{time} because time to collision is an important measure in traffic safety assessment [88]. The intuition behind this is to ensure that the vehicles do not occupy the same position at the same time.

Results

For each scenario, we trained a GRI model with the recurrent policy decoder (2.10)-(2.13). As shown in Table 2.2, in car following, the NRI model still performs better on trajectory reconstruction, but the GRI model achieves a comparable RMSE on the NGSIM dataset. In lane changing, the comparison is consistent: The NRI model slightly outperforms our model in trajectory reconstruction, while our model dominates the NRI model in graph accuracy.

We visualize the interaction graphs in Fig. 2.8. One interesting observation is that the graphs inferred by NRI have more edges in general. We want to emphasize that both models are trained under the same sparsity constraint. The results imply that we could guide the model to explore a clean and sparse representation of interactions by incorporating relevant domain knowledge, whereas the sparsity regularization itself is not sufficient to serve this purpose. Moreover, the NRI model assigns the same edge type to both edges between a pair of agents. This makes the graphs less interpretable because the vehicles should affect each other in different ways. On the other hand, while differing from the hypotheses, our GRI model tends to infer sparse graphs with directional edges.

The supervised policy has the lowest reconstruction error in lane changing. This implies that the human hypothesis is reasonable because it is capable of modeling the interactions among human drivers. In the car-following case, the reconstruction error is slightly higher than NRI. Since we cannot ensure that our hypothesis is the ground-truth interaction graph underlying the interacting system—in fact, as we mentioned before, we never meant to treat it as the ground-truth—it is possible that the NRI model can find a latent space that can effectively model the interactions in an unsupervised manner. However, as shown in Fig. 2.8, it is difficult to interpret the graphs inferred by NRI. Considering the sparse and semantic nature of the hypothesis as well as the fact that the supervised policy's reconstruction error is on par with the NRI model, we think the chosen hypothesis is a valid one.

2.6.5 Semantic Meaning of Latent Space

The experimental results described above show that our GRI model can recover the ground-truth interaction graphs in the synthetic scenarios with high accuracy and infer interaction graphs that are consistent with human hypotheses regarding the NGSIM dataset. However, as we argue in Sec. 2.1, accurate interaction inference alone is not sufficient to show that the model can learn a semantically meaningful latent space that is consistent with human domain knowledge. Given an edge, the policy decoder should also synthesize the corresponding semantic interactive behavior indicated by its edge type. It is difficult to verify whether the policy decoder is able to synthesize semantically meaningful interaction simply by monitoring the reconstruction error. A small reconstruction error on in-distribution data could be achieved by imitating demonstration without modeling the correct interaction [47, 127]. To study the semantic meaning of the latent space, we design a set of out-of-distribution tests [2] by adding increasing perturbation to the initial states. We then enforce the same edge types as in the in-distribution case and run those different policy decoders to generate the trajectories. We are curious whether the policy decoders can consistently synthesize the correct semantic interactive behavior under a distribution shift. If so, we argue that the

[2] For clarification, the models used in this section are the same as those introduced in Sec. 2.6.3. We merely designed additional out-of-distribution cases for testing.

Table 2.2: Performance Comparison on a Naturalistic Traffic Dataset

Model	Car Following ($\Delta t = 0.2s$, $T = 20$)			
	RMSE$_x$(m)	RMSE$_y$(m)	RMSE$_v$(m/s)	Accuracy(%)
GRI	1.700 ± 1.005	-	0.721 ± 0.363	$\mathbf{100.00 \pm 0.00}$
NRI	$\mathbf{1.436 \pm 0.880}$	-	$\mathbf{0.650 \pm 0.328}$	64.09 ± 0.08
Supervised	1.482 ± 0.938	-	0.665 ± 0.344	-
Model	**Lane Changing ($\Delta t = 0.2s$, $T = 30$)**			
	RMSE$_x$(m)	RMSE$_y$(m)	RMSE$_v$(m/s)	Accuracy(%)
GRI	7.118 ± 3.647	0.764 ± 0.336	4.320 ± 2.392	$\mathbf{98.55 \pm 0.06}$
NRI	6.532 ± 3.822	0.330 ± 0.181	$\mathbf{4.291 \pm 2.544}$	28.98 ± 0.08
Supervised	$\mathbf{5.897 \pm 3.651}$	$\mathbf{0.323 \pm 0.223}$	4.307 ± 2.435	-

[1] The data are presented in the form of mean $\pm$ std.

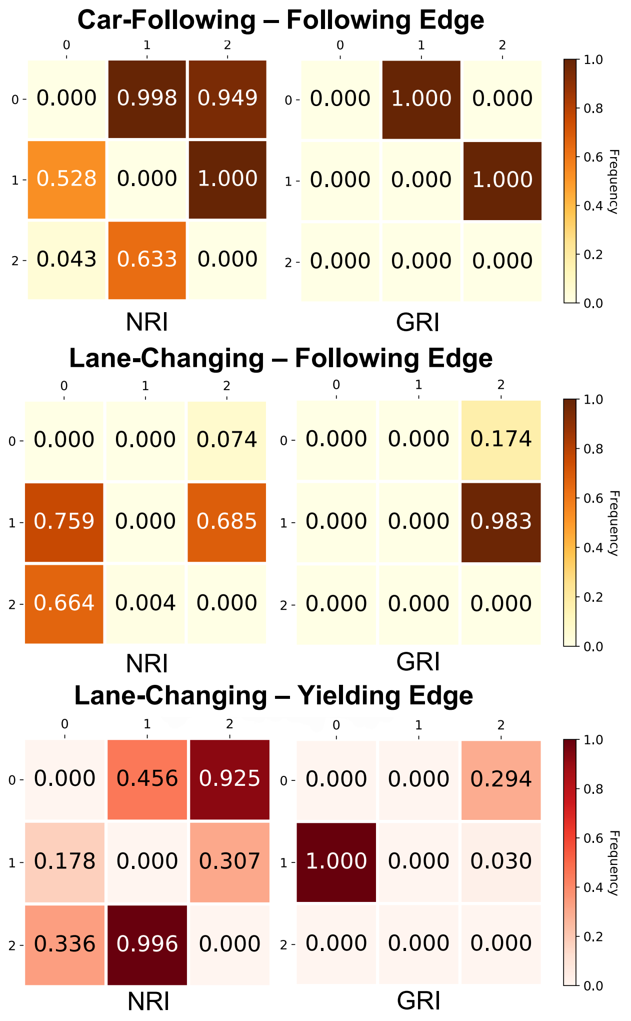


Figure 2.8: The empirical distribution of estimated edge variables $\hat{z}$ in the test dataset for the naturalistic traffic scenarios. We summarize the results in multiple adjacency matrices corresponding to different edge types. In the adjacency matrix corresponding to the k^{th} type of interaction, the element $A_{i,j}$ indicates the relative frequency of $z_{j,i} = k$, where $z_{j,i}$ is the latent variable for the edge from node j to node i.

latent space indeed possesses the semantic meaning that is consistent with human domain knowledge.

In the synthetic scenarios, we focus on the following relation. For both car-following and lane-changing scenes, we maintain the following relation for the two vehicles, resulting in interaction graphs merely consisting of the following edges (Fig. 2.9). We introduce perturbation by decreasing the initial longitudinal headway to values unseen during the training stage. The initial longitudinal headway is defined as $\Delta x = x_1^0 - x_0^0$, namely, the longitudinal distance from Vehicle 1 to Vehicle 0 at the first time step. During the training stage, we sampled Δx from uniform distributions: in car following, $\Delta x \sim \mathrm{unif}(4, 8)$; in lane changing, $\Delta x \sim \mathrm{unif}(8, 12)$. In the out-of-distribution experiments, we gradually decreased Δx from the lower bound to some negative value, which means Vehicle 0 is placed in front of Vehicle 1. We are curious whether the models can generate trajectories meeting the characteristics of the car-following behavior in these unseen scenarios—scenarios with a different number of vehicles and a distorted state distribution. To quantitatively evaluate if the synthesized behavior satisfies the requirement of car following, we consider three metrics for evaluation:

- Success Rate:

$$\begin{aligned} \mathrm{SuccessRate} &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(\Delta x_i^f \geqslant \delta_f), \\ \text{where } \Delta x_i^f &= x_{1,i}^T - x_{0,i}^T, \end{aligned} \tag{2.22}$$

- Collision Rate:

$$\begin{aligned} \mathrm{CollisionRate} &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(d_{\min,i} \leqslant \delta_c), \\ \text{where } d_{\min,i} &= \min_t \sqrt{\left|x_{1,i}^t - x_{0,i}^t\right|^2 + \left|y_{1,i}^t - y_{0,i}^t\right|^2}, \end{aligned} \tag{2.23}$$

- Lateral Distance:

$$\Delta y = \left|y_1^T - y_0^T\right| - \left|y_1^0 - y_0^0\right|. \tag{2.24}$$

We intend to quantify three typical characteristics of following behavior with the metrics defined above: 1) staying behind the leading vehicle; 2) maintaining a substantial safe distance from the leading vehicle; 3) staying in the same lane as the leading vehicle. We consider the following vehicle's maneuver successful if the vehicle manages to keep a substantial positive final headway, and we consider two vehicles as colliding if the minimum distance between them is smaller than the safety threshold. Finally, we expect the following behavior to attain a negative Δy, which means the following vehicle attempts to approach the leading vehicle's lane.

All metrics were applied in the lane-changing scenario, but we only adopted SuccessRate in the car-following scenario. Since we only model the longitudinal dynamics, Δy is not applicable. For the same reason, if their initial positions are too close or the following vehicle is located ahead of the leading one initially, the following vehicle will inevitably crash into the leading vehicle, resulting in $d_{\min} = 0$. Therefore, we only care about the first characteristic. The results are summarized in Fig. 2.10 and Fig. 2.11, where we plot the mean values of the evaluated metrics versus Δx. In the car-following scenario, the NRI policy fails to slow

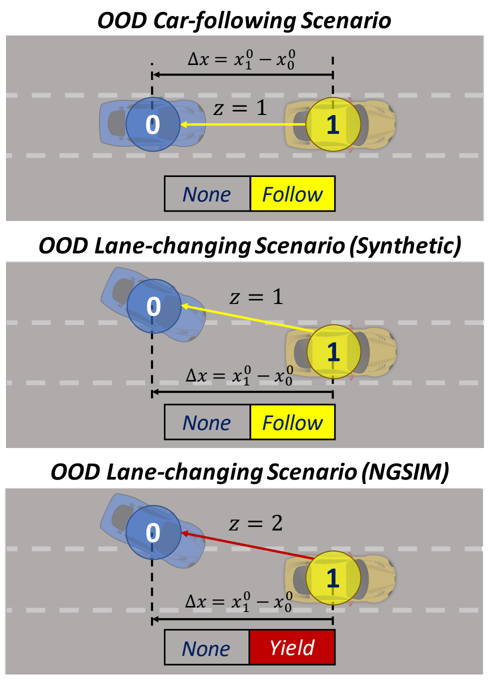

OOD Car-following Scenario
$\Delta x = x_1^0 - x_0^0$
$z = 1$
0
1
None
Follow
OOD Lane-changing Scenario (Synthetic)
$z = 1$
$\Delta x = x_1^0 - x_0^0$
None
Follow
OOD Lane-changing Scenario (NGSIM)
$z = 2$
$\Delta x = x_1^0 - x_0^0$
None
Yield

Figure 2.9: Out-of-distribution scenarios. We removed one vehicle from the nominal scenes and shifted the initial longitudinal headway Δx to unseen values.

down Vehicle 0 to follow Vehicle 1 when Δx becomes negative. In contrast, the supervised policy and GRI policy maintain high success rates with negative Δx. However, the number of failure cases starts to increase for the supervised policy when Δx becomes substantially negative, whereas the GRI policy maintains a perfect success rate over the tested range of perturbation. We visualize a marginal example in Fig.2.14, where both the NRI policy and the supervised policy fail to maintain a positive final headway.

In the lane-changing scenario, the GRI policy maintains a consistent perfect success rate over all tested values of Δx. For the other two models, the success rates drastically decrease with decreasing Δx. In terms of Δy, all models tend to reduce the lateral distance between the vehicles, which is consistent with the second characteristic of the following behavior. However, the GRI policy attains an average Δy with a smaller magnitude, and the magnitude decreases with decreasing Δx. This implies that the GRI policy changes its strategy when the initial position of Vehicle 0 is ahead of Vehicle 1. In order to maintain a proper safe distance, Vehicle 0 does not change its lane until Vehicle 1 surpasses it. Meanwhile, the lateral behavior is unchanged for the other two models. However, the vehicle cannot maintain a substantial safe distance if it changes lanes too early which is verified by the plot of collision rate versus Δx. The difference in their strategies is further illustrated by the example visualized in Fig. 2.14.

We repeat the experiment on the NGSIM datasets. Similar to the case of the synthetic

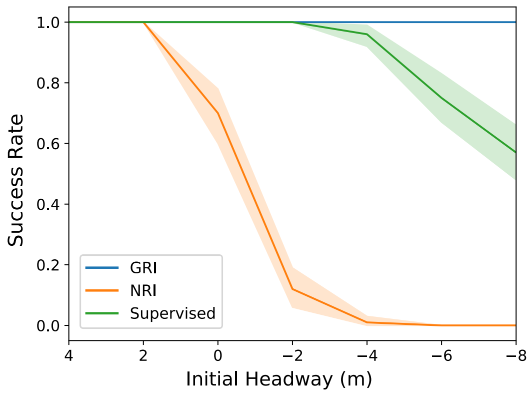


Figure 2.10: Results of the out-of-distribution synthetic car-following scenario. We plot SuccessRate versus Δx with the error band denoting the 95% confidence interval of the indicator, $\mathbf{1}(\Delta x_i^f \geqslant \delta_f)$. We set $\delta_f = 2\text{m}$.

dataset, we remove one vehicle from each scene, resulting in an interaction graph consisting of a single edge (Fig. 2.9). It is worth noting that removing a vehicle from a scene alters the dynamic of the interacting system. It is not fair to expect the models to synthesize the same trajectories in the dataset. Therefore, we do not aim to compare the generated trajectories with the ones in the dataset in this out-of-distribution test. Instead, we check whether the generated trajectories satisfy the desired characteristics of the corresponding interactive behaviors.

In the lane-changing case, the remaining edge has the type of yielding. According to our definition of the yielding relation, we consider the same characteristics and adopt the same metrics defined in Eqn. (2.22)-(2.24) for evaluation. Since we do not have control over the data generation procedure, we generate out-of-distribution test samples with different levels of discrepancy by controlling the ratio of longitudinal headway change. Given a sample from the original test dataset, we generate its corresponding out-of-distribution sample by shifting its initial longitudinal headway Δx by a certain ratio, denoted by δ, resulting in a new longitudinal headway $\Delta x'$:

$$\Delta x' = (1 - \delta)\Delta x.$$

We evaluate the models on datasets generated with different values of δ. We are particularly interested in the cases when $\delta \geqslant 1$, which leads to a negative initial headway. We present the results in Fig. 2.12 and 2.13. The comparison is consistent with the synthetic scenarios. Compared to the other baselines, our GRI policy can synthesize trajectories that satisfy the desired semantic properties in a larger range of distribution shifts.

The results suggest that even though the NRI model can accurately reconstruct the trajectories, the unsupervised latent space and the corresponding policies do not capture the semantic meanings behind the interactions. In contrast, the GRI model learns a semantically meaningful latent space, which is consistent with human domain knowledge. Another useful

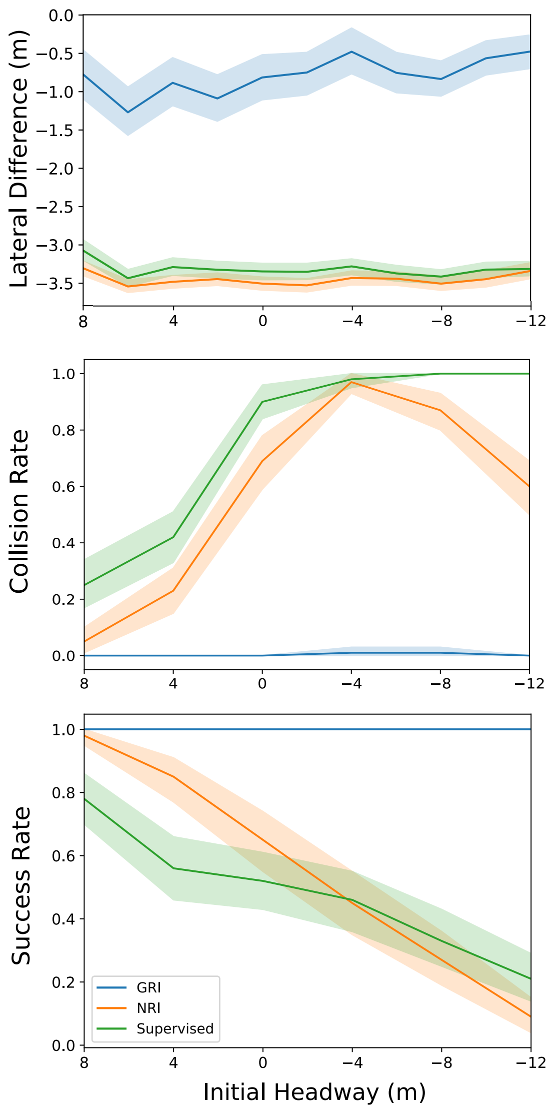


Figure 2.11: Results of the out-of-distribution synthetic lane-changing scenario. We plot SuccessRate, CollisionRate, and the mean value of Δy versus Δx. The error bands denote the 95% confidence interval. For SuccessRate and CollisionRate, the error bands are of the indicator functions. We set $\delta_f = \delta_c = 2\text{m}$.

insight we draw from the experiment is that interaction labels are not sufficient to induce an explainable model with semantic latent space. Even though the supervised policy utilizes additional information on the ground-truth interaction graph, it fails to synthesize the following behavior in novel scenarios. Although the GRI model still exhibits a considerable gap in reconstruction performance compared to the supervised baseline, it represents a promising and principled way to incorporate domain knowledge into a learning-based autonomous

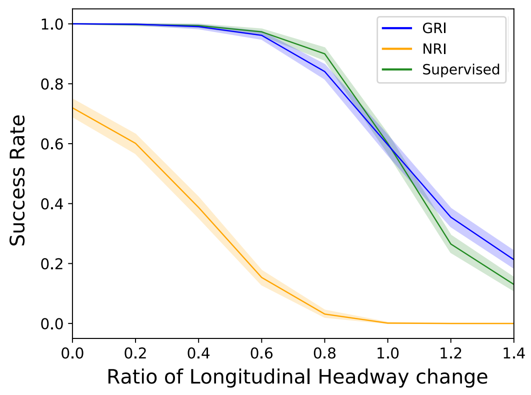


Figure 2.12: Results of the out-of-distribution naturalistic traffic car-following scenario. We plot SuccessRate versus Δx, with the error bands denoting the 95% confidence interval of the indicator, $\mathbf{1}(\Delta x_i^f \geqslant \delta_f)$. We set $\delta_f = 2\text{m}$.

driving system and induce an explainable model.

2.6.6 AIRL Ablation Study

With the motivation of incorporating semantic meaning into the relational latent space, we developed GRI by introducing AIRL into relational inference and studied how the semantic reward functions may guide relational latent space learning. However, it would be interesting to take a different perspective and study the effects of introducing relational inference and semantic reward functions into AIRL. In this section, we take the synthetic scenarios as examples and conduct an ablation study comparing GRI against two variants.

The first one is an AIRL variant, denoted as GRI-AIRL, which is obtained by removing relational inference and semantic reward functions from GRI. Concretely, both the policy and reward decoders operate on a fully connected interaction graph with a homogeneous edge type. We simply use MLPs to model the reward functions in Eqn. (2.14) and (2.15) instead of those semantic reward functions. The objective function then becomes Eqn. (2.16), but without the expectation of $\mathbf{z}$ or the information bottleneck constraint. The second one is a variational AIRL variant, denoted as GRI-VAIRL, in which we introduce relational inference but do not use the semantic reward functions. In this case, the objective function is identical to the one in GRI, that is, Eqn. (2.16).

The results are summarized in Table 2.3. For the car-following scenario, the reconstruction performance is improved after introducing relational inference into AIRL. It is interesting that the GRI-VAIRL variant is able to recover the ground-truth interaction graph, even without the semantic reward functions. It makes sense because the car-following scenario only consists of a single non-trivial edge type. It is plausible for the model to distinguish non-interaction edges from the others, because null reward is enforced for non-interaction edges. In some senses, we may still consider the reward function semantic—it incorporates

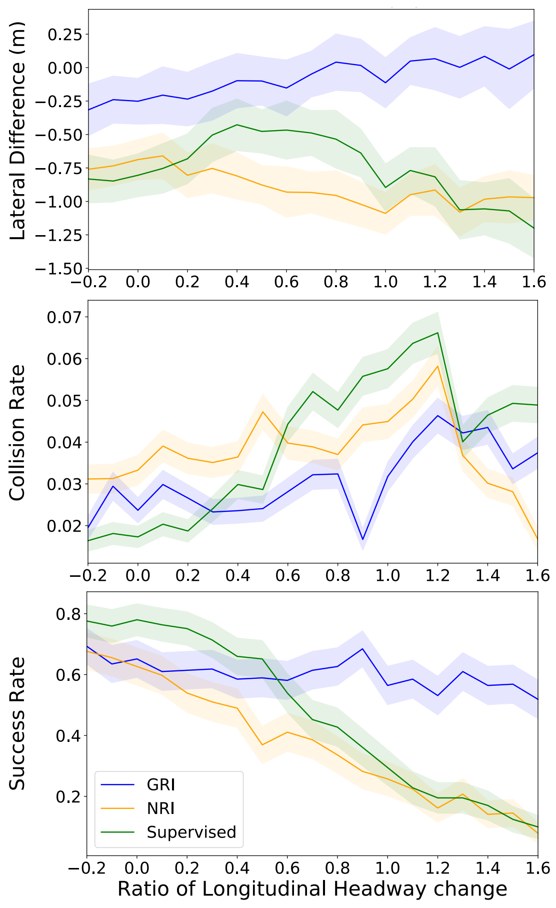


Figure 2.13: Results of the out-of-distribution naturalistic traffic lane-changing scenario. We plot SuccessRate, CollisionRate, and the mean value of Δy versus Δx. The error bands denote the 95% confidence interval. For SuccessRate and CollisionRate, the error bands are of the indicator functions. We set $\delta_f = \delta_c = 2$m.

the semantic meaning of non-interaction into the latent space. However, we cannot guarantee that GRI-VAIRL can distinguish between different non-trivial interactive behaviors, which is verified by the lane-changing case. Fig. 2.15 shows the inferred interaction graph. The model only adopts a single non-trivial edge type to describe all the interactive behaviors. Compared to the ground-truth graph, the inferred graph has an additional edge $z_{2,1}$ but ignores the edge $z_{1,0}$. Ignoring this edge limits the modeling capacity of the policy decoder, which could possibly explain why GRI-VAIRL has larger RMSE$_x$ and RMSE$_v$ than GRI-AIRL in the lane-changing case.

In summary, we could improve reconstruction performance by introducing relational inference into AIRL. While GRI-VAIRL has a larger reconstruction error in the lane-changing case due to the biased inferred graph, we still observe that GRI-VAIRL converges faster. The

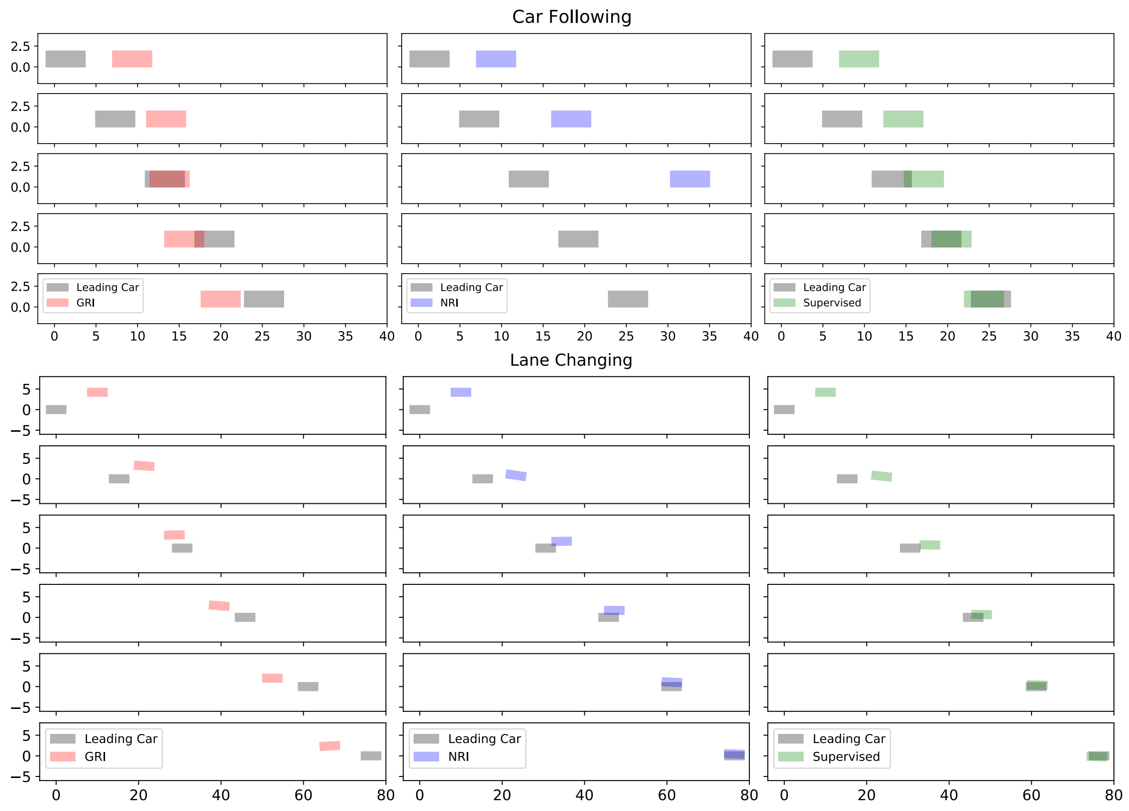


Figure 2.14: Examples where the leading car is placed behind the following car at the initial time step. The trajectories are visualized as sequences of rectangles. Each rectangle represents a vehicle at a specific time step. The vehicles are driving along the positive direction of the x-axis. The GRI policy still prompts the car-following behavior: It slows the vehicle until the leading one surpasses it. Meanwhile, the NRI policy and the supervised policy do not behave as $\mathcal{G}_{\text{interact}}$ suggests.

learning process becomes more stable and less sensitive to different hyperparameters. We think this is because the model may identify agents that are not interacting with each other, preventing the reward decoder from fitting a reward function unifying both interactive and non-interactive behaviors. It is still necessary to incorporate semantic reward functions to differentiate different interactive behaviors and induce a semantically meaningful interaction graph. However, semantic latent space comes at a cost of reconstruction performance. The structured reward functions limit the modeling capacity of the reward decoder. Additionally, although the structured reward functions are differentiable, there is no guarantee that they can be well optimized through gradient descent. As a result, they may interfere with the stability of the learning procedure.

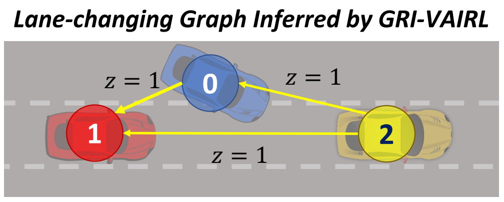


Figure 2.15: The graph inferred by GRI-VAIRL in the synthetic lane-changing scenario.

2.7 Discussion and Limitation

2.7.1 Application of the Semantic Latent Space

Designing an explainable model is a crucial step toward trustworthy human-AI interaction. However, it is still unclear how humans may benefit from improved explainability. We would like to briefly discuss the potential application of the semantic latent space introduced in GRI. When the autonomous vehicle encounters an unfamiliar situation (e.g., the out-of-distribution scenarios studied in Sec. 2.6.5), a semantic latent space gives the safety drivers or passengers the ability to review and override the inferred interaction graph if the model misunderstands the scenario. In contrast, if the learned interactive behaviors do not have explicit semantic meaning, humans can neither understand an interaction graph nor identify the correct edge types. Such safety assurance could help in facilitating safe and trustworthy cooperation between humans and autonomous vehicles.

However, it is impractical to require users to monitor the model output in real-time.

Table 2.3: Ablation Study on Synthetic Dataset

Model	Car Following ($\Delta t = 0.2s$, $T = 20$)			
	RMSE$_x$(m)	RMSE$_y$(m)	RMSE$_v$(m/s)	Accuracy(%)
GRI	0.241 ± 0.125	-	0.174 ± 0.068	$\mathbf{100.00 \pm 0.00}$
GRI-VAIRL	$\mathbf{0.120 \pm 0.054}$	-	$\mathbf{0.116 \pm 0.039}$	$\mathbf{100.00 \pm 0.00}$
GRI-AIRL	0.138 ± 0.068	-	0.150 ± 0.043	-
Model	**Lane Changing ($\Delta t = 0.2s$, $T = 30$)**			
	RMSE$_x$(m)	RMSE$_y$(m)	RMSE$_v$(m/s)	Accuracy(%)
GRI	0.529 ± 0.230	0.207 ± 0.046	0.303 ± 0.128	$\mathbf{99.95 \pm 0.01}$
GRI-VAIRL	0.377 ± 0.201	$\mathbf{0.160 \pm 0.038}$	0.190 ± 0.058	50.0 ± 0.00
GRI-AIRL	$\mathbf{0.304 \pm 0.321}$	0.198 ± 0.065	$\mathbf{0.173 \pm 0.101}$	-

[1] The data are presented in the form of mean $\pm$ std.

Instead, we can introduce an additional module to detect out-of-distribution scenes [29, 122] and use the estimated epistemic uncertainty to decide when to query the end users. In [29], the authors proposed an adaptive variant of their robust imitative planning algorithm, which incorporates such a unit. It is also a common practice for current autonomous driving companies to have human assistants for vehicles to query when encountering abnormal situation.

2.7.2 Limitation of the Learning Algorithm

In our experiments, GRI always has higher reconstruction error than NRI, especially on the synthetic dataset. One of the reasons for this is that reconstruction error is not directly optimized under the AIRL formulation. The objective function of NRI consists of a reconstruction loss, which essentially minimizes the Euclidean distance between the reconstructed trajectory and the ground-truth trajectory. In other words, it directly minimizes the RMSE metrics used in our evaluation. In contrast, GRI adopts the objective function of AIRL, which also minimizes the distance between the trajectory pair. However, the distance is defined by the learned discriminator and is not necessarily equivalent to the Euclidean distance. In Sec. 2.6.6, we study two AIRL baseline models on the synthetic dataset. The results suggest that none of these AIRL-based approaches achieve the same reconstruction performance as NRI.

Another reason is that the current learning algorithm is not entirely stable because of the adversarial training scheme we introduce when incorporating AIRL into the original NRI model. In typical AIRL settings, we can mitigate this problem by warm-starting the training with a policy network pretrained through imitation learning or behavior cloning [30, 146]. However, since we aim to learn a semantic latent space, warm-starting the training with a model with unsupervised latent space is not helpful. Alternatively, we may initialize the policy decoder with the supervised one. One issue with this is that it will change our current setting where human labels are not required. We will investigate this new setting in our future work and develop a more stable training scheme to further optimize the performance of GRI. A stable training scheme is also a prerequisite before applying GRI to more sophisticated real-world scenarios.

The structured reward functions also interfere with the stability of the learning procedure. Compared to the GRI variant studied in Sec. 2.6.6 with the semantic reward functions removed, GRI is more sensitive to hyperparameters and prone to diverging if not carefully tuned. This is because although the structured reward functions are differentiable, there is no guarantee that the reward functions can be stably optimized through gradient descent. In our future work, we will explore a more stable and robust learning scheme with those structured reward functions.

2.8 Chapter Summary

In this chapter, we propose GRI, which models an interactive system's underlying dynamics by inferring the agents' semantic relations. By incorporating structured reward functions, we ground the relational latent space into semantically meaningful behaviors defined with expert domain knowledge. We demonstrate that GRI can model interactive traffic scenarios under both simulated and real-world settings and generate semantic interaction graphs explaining the vehicle's behavior by their interactions.

There are several technical gaps we need to bridge before extending the current framework to more complicated traffic scenarios. One technical gap is graph dynamics. We currently assume a static interaction graph over the time horizon. We will investigate how to incorporate dynamic graph modeling into the current framework. Another gap is the cooperative assumption, which we would like to remove in the future so that the framework can be generalized to non-cooperative scenarios. Further, as we mentioned earlier, there is still a considerable gap in the reconstruction performance of the GRI model compared to the other baselines. In future work, we will improve the model architecture and training algorithm to fill this performance gap while maintaining the advantages of GRI as an interpretable model.

Nevertheless, we are encouraged to see that the structured reward functions can indeed ground the relational latent space into semantically meaningful interactive behaviors. This allows us to incorporate knowledge of human interaction in a principled manner to induce an interpretable interaction model. Moreover, although we limit our experiments to the autonomous driving domain, the model itself is formulated without specifying the context. As long as proper domain knowledge is available, the proposed method can be extended naturally to other fields (e.g., human–robot interaction). We believe that the GRI framework proposed in this chapter could be an important building block to improve model interpretability for autonomous driving as well as other intelligent systems interacting with humans.

Chapter 3

Pseudo Labels for Interpretable Interactive Trajectory Prediction

3.1 Introduction

The last chapter introduced GRI as an interpretable model for multi-agent interaction modeling. Although we can extend GRI for trajectory prediction, the current performance limits of GRI make it impractical to directly adapt it to trajectory prediction tasks in complicated interactive traffic scenarios. As mentioned at the end of the last chapter, we need to fill the performance gap before extending GRI to more complicated scenarios, which will be left for future investigation. In this chapter, we look into this problem from the opposite direction. Instead of developing an interpretable model and extending it to trajectory prediction, we study how to improve the interpretability of state-of-the-art trajectory prediction models while maintaining consistent prediction accuracy.

We focus on the interaction prediction problem formulated by Waymo on their Waymo Open Motion Dataset [25]. Previous prediction benchmarks mainly focus on single-agent settings [150]. When multiple agents exist, the predicted trajectories of the agents are evaluated separately. While this evaluation scheme is sufficient for ordinary cases, it cannot precisely assess a prediction model in highly interactive scenarios (e.g., intersections and roundabouts). For instance, a vehicle may enter the intersection at a two-way stop or yield before the stop line. In such a scenario, a prediction model capturing the multimodality of a single vehicle can achieve good performance under the single-agent evaluation scheme. However, two vehicles from different directions should never enter the intersection simultaneously. In this case, it is necessary to accurately predict the *joint* behavior of interacting agents to ensure safe and efficient operations. However, a single-agent evaluation scheme cannot effectively evaluate a model for joint prediction. In the interaction prediction benchmark provided by Waymo, the trajectories of two interacting agents are predicted and evaluated jointly. This motivates us to study the interaction prediction problem to derive a suitable trajectory prediction model for highly interactive scenarios.

In particular, we are interested in the interaction prediction problem under the goal-conditioned framework, as goal-conditioned methods can effectively capture the multimodality in trajectory distribution [86, 151, 42]. However, previous methods mainly focus on single-agent prediction. For multiple agents, these methods predict the trajectories independently for each agent. To model the joint distribution of interacting agents' goals, we extend the goal set to a goal-pair set which allows joint prediction of two agents' endpoints. Under this framework, we first explicitly predict the distribution of an agent's endpoint over a discretized goal set and then complete the trajectories conditioned on the selected goal points. By choosing a dense set as in [42], this categorical distribution of goal pairs can reasonably approximate the joint distribution in any interactive scenarios.

In practice, downstream modules require a small set of representative predictions [24]. The limited onboard computational resources also restrict the number of sampled trajectories. For the downstream module to understand the interactive scenario precisely, it is critical to ensure that different interaction modes can be efficiently captured with a limited number of sampled trajectories. To this end, we leverage the CVAE framework [117] and introduce a discrete latent space to capture the interaction modes explicitly. Compared to continuous latent variables, a discrete latent variable enables better interpretability of the results [56, 109]. However, it does not guarantee that the model can learn an informative latent space distinguishing semantically meaningful interaction modes that are useful for downstream modules.

In our goal-conditioned CVAE framework, the goal pair follows a categorical distribution, changing the reconstruction task into a multi-label classification problem. Without knowing the distance between the goal pairs, it is difficult for the model to distinguish between them. Therefore, it is difficult to determine which encoding goal pairs should correspond to the same latent variable, which leads to the problem of posterior collapse in CVAE, resulting in an uninformative latent space. To tackle this problem, we propose guiding the training with *pseudo labels*[1] designed based on domain knowledge. For each ground-truth goal pair, we assign positive target values to goal pair candidates similar to it. The model learns to encode similar goal pairs into the same latent variable by minimizing the distance between the decoded distribution and the pseudo labels. Since the goal pair distribution is defined over a fixed finite set, the pseudo labels can be pre-computed for each goal pair candidate. We do not require the computation of pseudo labels to be differentiable, which allows us to flexibly incorporate domain knowledge and specify any interaction modes for the latent space to capture. In particular, we introduce three types of pseudo labels corresponding to different domain knowledge on interaction. We show that the model learns to capture the designated interaction modes in its latent space with the proposed pseudo labels.

The rest of the chapter is organized as follows. In Sec. 3.2, we briefly introduce the goal-conditioned prediction framework under the single-agent setting, which is commonly adopted in the literature. In Sec. 3.3, we formulate the goal-conditioned prediction problem

[1]For clarification, we refer to any generated labels other than the ground-truth ones as pseudo labels. They are not necessarily generated for semi-supervised or self-supervised learning.

for joint trajectory prediction of interacting pairs. In Sec. 3.4, we illustrate the motivation of interpretable interactive prediction and pseudo labels with a toy example and derive three types of pseudo labels incorporating different domain knowledge. In Sec. 3.5, we present the architecture of the model we design based on state-of-the-art goal-conditioned trajectory prediction models. In Sec. 3.6, we report the experiments on the Waymo Open Dataset, showing that the proposed pseudo labels can effectively induce an interpretable latent space and further improve prediction performance.

3.2 Background: Goal-Conditioned Prediction

In general, a trajectory prediction model learns to model the distribution $p(\boldsymbol{y}|\boldsymbol{T})$, where $\boldsymbol{y}$ denotes the future trajectory of the target agent, and $\boldsymbol{T}$ denotes the embedding of the agent's history and context information. In a goal-conditioned trajectory prediction framework, the prediction task consists of two stages, goal prediction and trajectory completion, resulting in the decomposition of $p(\boldsymbol{y}|\boldsymbol{T})$:

$$p(\boldsymbol{y}|\boldsymbol{T}) = \int_{g\in\mathcal{G}} p(\boldsymbol{y}|g,\boldsymbol{T}) \cdot p(g|\boldsymbol{T})dg,$$

where $\mathcal{G}$ is the goal space. The goal-prediction model $p(\boldsymbol{g}|\boldsymbol{T})$ can capture the multimodality in driver intention, while the goal-conditioned trajectory completion module models the driving behavior to reach the goals.

The overall framework has three stages. The first stage is *goal distribution prediction*. Depending on the goal space, $p(\boldsymbol{g}|\boldsymbol{T})$ can be modeled as either a continuous or discrete distribution. We are particularly interested in the formulation of [42], in which $\mathcal{G}$ is defined as a dense and discretized goal set covering the drivable area. Here, $p(\boldsymbol{g}|\boldsymbol{T})$ directly models the distribution of goal points instead of anchor points, as in [151]. The second stage is *goal-conditioned trajectory prediction*, where the conditional distribution of future motions is modeled as a simple unimodal distribution (e.g., Gaussian distribution). The third stage is *sampling and selecting*, where a final small number of predicted trajectories are selected to fulfill the requirement of downstream applications. Heuristic-based algorithms, such as non-maximum suppression (NMS), are commonly used for this purpose [151].

3.3 Problem Formulation

The framework described in Sec. 3.2 is primarily designed for single-agent prediction. It is not straightforward to extend this two-stage prediction scheme to multi-agent settings. In multi-agent trajectory prediction, we need to model the joint distribution of all agents' future trajectories, i.e., $p\left(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_N|\boldsymbol{T}\right)$. We can decompose the interacting agents in the trajectory completion stage by adopting the assumption that the trajectories are independent after conditioning on the goals. However, we still need to model the joint distribution of their goals, i.e., $p(\boldsymbol{g}_1, \boldsymbol{g}_2, \cdots, \boldsymbol{g}_N|\boldsymbol{T})$. We cannot simply assume that the trajectories of interacting agents are independent and decompose the joint distribution into $\prod_{i=1}^{N} p(\boldsymbol{g}_i|\boldsymbol{T})$. The simplified distribution cannot model the interactive behavior between agents, for instance, the fundamental interacting rule—collision avoidance. Meanwhile, if we directly model the joint distribution, we need to select a discrete goal set $\mathcal{G}_i$ for each modeled agent i. The overall dimension of the joint distribution becomes $\prod_{i=1}^{N} |\mathcal{G}_i|$, which grows exponentially with the number of agents.

To tackle this problem, one heuristic method is to first separately predict the marginal distributions of the goals $g_1, g_2, \cdots, g_N$ and then prune the unrealistic combinations using designed rules. For example, collision is normally ruled out in the predicted trajectories [130, 124] to reduce prediction errors. However, it is difficult to prune the distribution with heuristically selected rules without introducing bias into the model. After heuristic pruning, the model cannot capture rule-violating behaviors (e.g., traffic accidents).

Instead, we propose an alternative scheme to mitigate the curse of dimensionality. We first predict the marginal distributions of the goals. Then, we use the marginal distributions to prune the goal sets $\{\mathcal{G}_i\}_{i=1}^{N}$. Concretely, we select K goal candidates with the highest marginal probability for each agent. We find that we can reasonably approximate the marginal distribution with $K << |\mathcal{G}_i|$. It is then sufficient to model the distribution of $|K|^N$ goal combinations, which is applicable for the prediction task of the interacting pairs we study. Further, we do not regularize joint prediction with interaction rules to avoid bias or over-regularization. Instead, we propose a novel approach for incorporating interaction domain knowledge under the VAE formulation with pseudo labels, which will be introduced in the next section.

3.4 Interpretable Interactive Prediction

This section presents our study on a motivating toy example of an interactive traffic scenario. The purpose is twofold: 1) to demonstrate the necessity of modeling the joint distribution in interactive prediction; 2) to explore how we may incorporate domain knowledge on interaction rules to induce an interpretable model.

The scenario we consider is illustrated in Fig. 3.1. Vehicle A and B are driving toward a collision point. The states of the vehicles are s_a, s_b and v_a, v_b, where $s_{a,b}$ are the displacements of the vehicles A, B relative to the collision point and $v_{a,b}$ are the absolute velocities. Each vehicle is assigned a target position to follow at each step, depending on which vehicle has the "right of way." If a vehicle has the right of way, it follows a target substantially far away. Otherwise, the target is set as the collision point if the other vehicle has the right of way and has not passed the collision point. We assume that the right of way is affected by the difference in the time headway at the initial time since the car with a shorter headway time to the collision point is more likely to have the right of way in the interaction. The time headway at time step t is defined as $T_{\text{head},t} = \max\left(\frac{s_t}{v_t}, 0\right)$. The probability of Vehicle

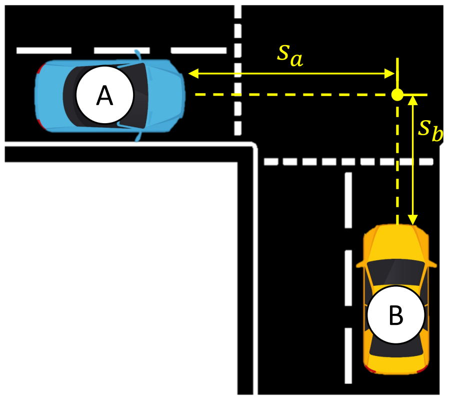


Figure 3.1: A motivating toy example, where two cars are driving toward a collision point. The states of the vehicles are s_a, s_b and v_a, v_b, where $s_{a,b}$ are the displacements of the vehicles A, B relative to the collision point, and $v_{a,b}$ are the absolute velocities. Each vehicle is assigned a target position to follow at each step, depending on which vehicle has the "right of way." If a vehicle has the right of way, it follows a target substantially far away. Otherwise, the target is set as the collision point if the other vehicle has the right of way and has not passed the collision point.

A getting the right of way is calculated as follows:

$$p_A = 0.5\left(\tanh\frac{T_{a,\text{head},0} - T_{b,\text{head},0}}{\eta} + 1\right),$$

where η controls the rate of transition between entering into the intersection and yielding. The dynamics of the vehicles are governed by the intelligent driver model (IDM) [132]:

$$\begin{aligned} s_{t+1} &= s_t - \Delta t \cdot v_t, \\ v_{t+1} &= v_t + \Delta t \cdot \left\{ a\left[1 - \left(\frac{v_t}{v_0}\right)^{\delta} - \left(\frac{s^*(v_t, \Delta v_t)}{s_t - d_t}\right)^2\right] + \omega_t \right\}, \end{aligned}$$

where

$$\begin{aligned} \Delta v_t &= v_t - v_0, \\ s^*(v_t, \Delta v_t) &= s_0 + \max\left(0, v_t T + \frac{v_t \Delta v_t}{2\sqrt{ab}}\right), \\ \omega_t &\sim \mathcal{N}\left(0, \sigma^2\right). \end{aligned}$$

The term d_t denotes the target position. It is zero if the vehicle yields and the other vehicle has not passed the collision point. Otherwise, a large negative value is assigned to d_t. The Gaussian noise ω_t is added to inject stochasticity. The remaining parameters are defined as in the standard IDM. Readers may refer to [132] for detailed definitions. In our experiments, we set $v_0 = 10\text{m/s}$, $T = 2\text{s}$, $s_0 = 4\text{m}$, $\delta = 4$, $a = 1\text{m/s}^2$, $b = 1.5\text{m/s}^2$, $\Delta t = 0.2\text{s}$, and $\sigma = 4\text{m/s}^2$.

Given the same initial conditions, there are two possible interaction modes, i.e., Vehicle A yields to Vehicle B and vice versa. These two interaction modes result in a multimodal future trajectory of both vehicles. The task here is to jointly predict the endpoints $\boldsymbol{g} = (\boldsymbol{g}_a, \boldsymbol{g}_b) = (s_{a,20}, s_{b,20})$ of both vehicles after 20 time steps. We abuse the notation here and denote the initial condition as $\boldsymbol{T}$, i.e., $\boldsymbol{T} = (\boldsymbol{T}_a, \boldsymbol{T}_b) = ([s_{a,0}, v_{a,0}], [s_{b,0}, v_{b,0}])$, as the initial condition is the context information in this toy example. This is analogous to the goal prediction stage in the goal-conditioned trajectory prediction framework. To train the model, We construct a dataset with randomly sampled initial conditions.

3.4.1 Marginal vs. Joint Prediction

To demonstrate the necessity of joint prediction in this interactive scenario, we compare the joint and marginal distributions of $(\boldsymbol{g}_a, \boldsymbol{g}_b)$. We approximate the distributions with samples and visualize the distributions in Fig. 3.2. For each vehicle, the marginal distribution of the endpoint has two peaks, corresponding to entering the intersection and yielding. We can observe that the peaks are paired, corresponding to two different interaction modes from the joint distribution. However, we cannot identify the correspondence between the peaks from the marginal distributions. As a result, we may obtain unrealistic predictions (e.g., both vehicles yield) by querying a marginal prediction model. In highly interactive cases, this may cause the downstream planner to misunderstand the scenarios and generate dangerous or inefficient maneuvers.

3.4.2 Joint Prediction with Interactive Latent Variable

Now we study joint prediction using this toy example. We discretize the spaces of $s_{a,20}$ and $s_{b,20}$ and obtain a discrete set of goal pairs, $\mathcal{G}_{a,b}$. We formulate the joint prediction problem as a classification problem to predict the joint goal distribution from the initial conditions. We leverage the CVAE framework [117] and introduce a discrete latent variable $\boldsymbol{z}$. The objective of the prediction model is twofold:

- Accurately model the ground-truth joint distribution of two agents' goal pairs given the initial conditions.
- Encode different interaction modes into the discrete latent space to enhance interpretability and sampling efficiency.

The CVAE model consists of three modules: 1) An encoder $q_\theta(\boldsymbol{z}|\boldsymbol{T}, \boldsymbol{g})$ approximating the posterior distribution of $\boldsymbol{z}$; 2) A conditional prior $p_\phi(\boldsymbol{z}|\boldsymbol{T})$; 3) A decoder $p_\psi(\boldsymbol{g}|\boldsymbol{T}, \boldsymbol{z})$ modeling

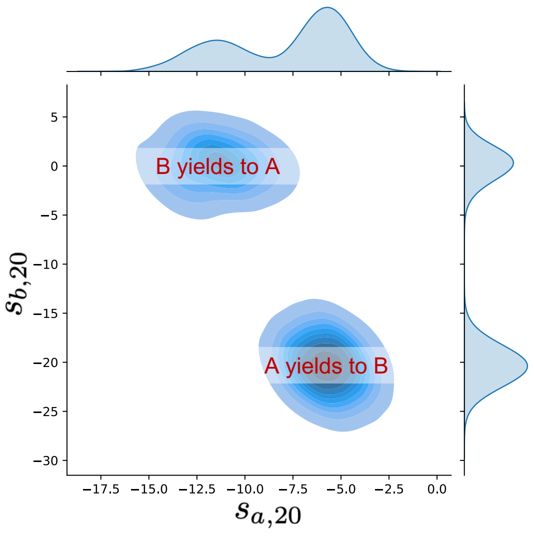


Figure 3.2: The ground-truth joint distribution and marginal distributions of $s_{a,20}$ and $s_{b,20}$. We use kernel density estimation (KDE) to smooth the empirical distributions.

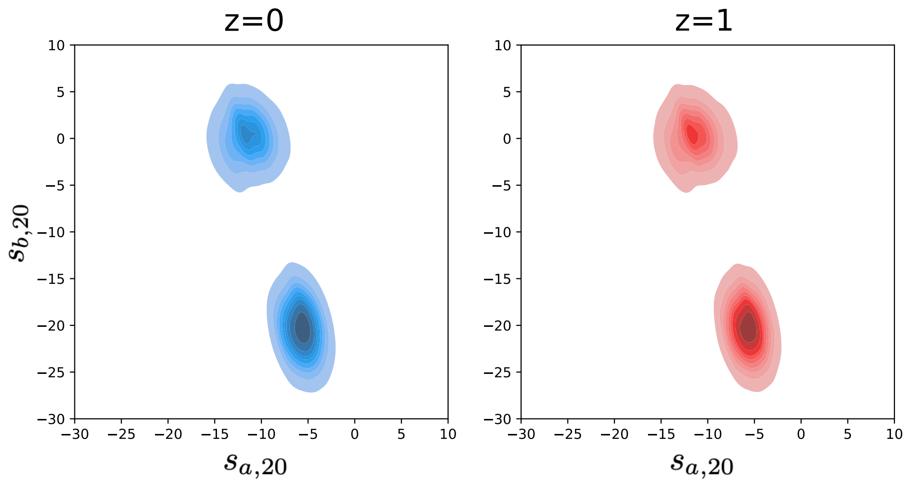
the conditional joint goal distribution. We use simple MLPs for all the modules. The model is trained by maximizing the ELBO:

$$\mathcal{L}(\theta,\phi,\psi)=\mathbb{E}_{\boldsymbol{T},\boldsymbol{g},\boldsymbol{y}\sim\mathcal{D}}\left\{\mathbb{E}_{\boldsymbol{z}\sim q_\theta(\boldsymbol{z}|\boldsymbol{T},\boldsymbol{g})}\left[f\left(\boldsymbol{y},p_\psi\left(\cdot|\boldsymbol{T},\boldsymbol{z}\right)\right)\right]-\beta D_{KL}\left[q_\theta(\boldsymbol{z}|\boldsymbol{T},\boldsymbol{g})\|p_\phi(\boldsymbol{z}|\boldsymbol{T})\right]\right\}, \quad (3.1)$$

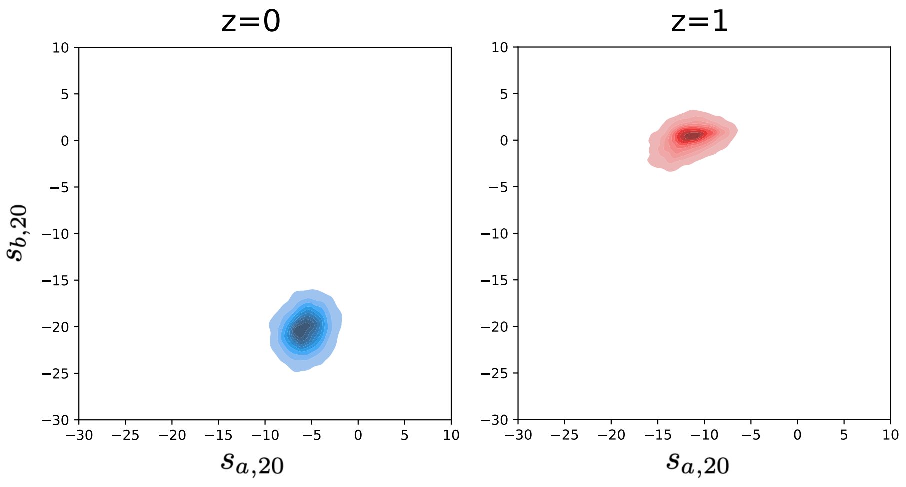
where $\mathcal{D}$ is the dataset consisting of initial states $\boldsymbol{T}$, goal pairs $\boldsymbol{g}$, and ground-truth labels for goal distribution $\boldsymbol{y}$. The vector $\boldsymbol{y}\in\{0,1\}^{|\mathcal{G}_{a,b}|}$ collects ground-truth scores of the goal pairs in $\mathcal{G}_{a,b}$. We assign one to the ground-truth goal pair and zero to the others. In the objective function, the first term is the *joint reconstruction* loss. We implement a binary cross-entropy (BCE) loss for the categorical joint goal distribution. The second term is the *Kullback–Leibler (KL) divergence* between posterior and prior encoder.

3.4.3 Avoiding KL Vanishing with Pseudo Labels

Our experiments with the CVAE model formulated above show that the KL divergence term tends to vanish, and the conditional prior distribution always concentrates into a single value. As shown in Fig. 3.3a, the latent space is completely uninformative. While the decoder can still model the joint distribution, the model does not fulfill our objective to explicitly capture interaction modes with the latent space. This phenomenon is similar to the posterior collapse problem that occurs when an autoregressive decoder is used in sequence modeling [33]. The MLP decoder we use can model the joint distribution without the latent space. With such a powerful decoder, the model is prone to ignoring the latent space to minimize the KL divergence.



(a) Vanilla CVAE Model



(b) CVAE Model with Pseudo Distance Labels

Figure 3.3: Joint goal distributions decoded from different latent variables with different models. For the vanilla CVAE model, the decoded distributions are invariant because of posterior collapse. With the pseudo distance labels, the CVAE model is able to capture the two interaction modes in its latent space. The results are the same when using other pseudo labels.

We find it challenging to force the model to escape from posterior collapse in our case. Given the same initial conditions, the joint prediction model essentially solves the following maximization problem when the discrete latent variables are not involved:

$$\max_{\boldsymbol{p}\in\Delta^d} \sum_{i=1}^{|\mathcal{D}|} \sum_{j=1}^{d} \mathbf{1}\left(y_j^i = 1\right) \log p_j + \mathbf{1}\left(y_j^i \neq 1\right) \log\left(1 - p_j\right),$$

where $d = |\mathcal{G}_{a,b}|$. The optimal solution equals the empirical distribution of goal pairs in the dataset:

$$p_j^* = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \mathbf{1}\left(y_j^i = 1\right), \text{ for } j = 1, 2, \cdots, d. \tag{3.2}$$

By introducing a discrete latent variable $z \in \{1, 2, \cdots, d_z\}$, the VAE model essentially clusters $\mathcal{G}_{a,b}$ into d_z subgroups and solves the optimization problem separately for each subgroup. The optimal clustering scheme maximizes the sum of objectives over the subgroups. Given a fixed clustering scheme with subgroups $\{S_k\}_{k=1}^{d_z}$, the optimal objective value equals to:

$$\mathcal{L}(\{S_k\}) = \sum_{k=1}^{d_z} \sum_{j\in S_k} n_j \log(\frac{n_j}{n_{S_k}}) + (n_{S_k} - n_j) \log(1 - \frac{n_j}{n_{S_k}}),$$

where we define n_j and n_{S_i} as:

$$n_j = \sum_{i=1}^{|\mathcal{D}|} \mathbf{1}\left(y_j^i = 1\right), \quad n_{S_k} = \sum_{i=1}^{|\mathcal{D}|} \sum_{j\in S_k} \mathbf{1}\left(y_j^i = 1\right).$$

The optimal solution is then the subgroups that maximize $\mathcal{L}(\{S_k\})$. With posterior collapse, the clustering scheme corresponds to having all the elements in a single subgroup while leaving the remaining subgroups empty. In our toy example, it is easy to check that better solutions do exist, for example, the one shown in Fig. 3.3b. However, it is difficult for the model to escape from the suboptimal solution shown in Fig. 3.3a. From the equation above, we can see that the optimal clustering scheme relies solely on the frequencies of different classes in the dataset. We can interchange goal pairs that occur with similar frequencies without affecting the objective value. It is then difficult for the VAE model to learn whether two goal pairs should correspond to the same latent variable. Even if the model manages to avoid KL vanishing, a latent space that clusters goal pairs based purely on frequencies is not informative.

If we introduce an additional loss function incorporating our domain knowledge about the proximity between goal pairs, it will break the tie and regularize the model to cluster the goal pairs. We propose incorporating such auxiliary loss functions with generated pseudo labels. For each goal pair $\boldsymbol{g}_j$, we generate pseudo labels over $\mathcal{G}_{a,b}$ denoted by $\hat{\boldsymbol{y}}_j \in [0,1]^d$. We then add the following auxiliary loss function to the original ELBO objective:

$$\alpha \mathbb{E}_{\boldsymbol{T},\boldsymbol{g},\boldsymbol{y}\sim\mathcal{D},\boldsymbol{z}\sim q_\theta(\boldsymbol{z}|\boldsymbol{T},\boldsymbol{g})} \sum_{j=1}^{d} \mathbf{1}(y_j = 1) f\left(\hat{\boldsymbol{y}}_j, p_\phi(\cdot|\boldsymbol{T},\boldsymbol{z})\right).$$

The function f quantifies the distance between the pseudo labels and the conditional joint goal distribution. Intuitively, we assign positive target values to goal pairs that we think are "close" to $\boldsymbol{g}_j$. By optimizing the auxiliary loss, the model learns to encode "close" goal pairs into the same latent variable. We do not require the pseudo labels to be differentiable functions of goal pairs, which allows us to incorporate arbitrary formats of pseudo labels. In the following section, we will introduce three types of pseudo labels.

3.4.4 Pseudo Labels

Pseudo Distance Labels

Since the agents move continuously, their behaviors should be consistent if the targeted goal pairs are close to each other in Euclidean distance. Such goal pairs should therefore be clustered into the same group. Consequently, we introduce the pseudo distance labels defined as:

$$\hat{\boldsymbol{y}}_{j,i}^{\text{distance}} = \exp\left(-\frac{\|\boldsymbol{g}_j - \boldsymbol{g}_i\|^2}{2\sigma^2}\right),\ i = 1, 2, \cdots, d.$$

This essentially smooths the original singular positive label with the radial basis (RBF) kernel. We choose f as the BCE loss function.

With the auxiliary loss defined with the pseudo distance labels, the CVAE model learns to separate the two interaction modes in the latent space (Fig. 3.3b). Further, the prior probabilities of the two latent variables are consistent with the ground-truth probabilities of the corresponding interaction modes. The interaction modes can be effectively separated because the Euclidean distance between goal pairs from different clusters is far large.

Pseudo Marginal Labels

The joint goal distribution is the consequence of the interaction between agents. Suppose Agent A behaves consistently while Agent B changes its behavior. In that case, we may characterize the interaction by the goal point of Agent A. Therefore, we consider goal pairs that share the same goal of one agent closer than those that are totally different. We then define two sets of pseudo marginal labels:

$$\hat{\boldsymbol{y}}_{j,i}^{\text{marginal},a} = \mathbf{1}\left(\boldsymbol{g}_{j,a} = \boldsymbol{g}_{i,a}\right), \quad \hat{\boldsymbol{y}}_{j,i}^{\text{marginal},b} = \mathbf{1}\left(\boldsymbol{g}_{j,b} = \boldsymbol{g}_{i,b}\right).$$

And we define the corresponding loss function:

$$\begin{aligned} & f^{\text{marginal}}\left(\hat{\boldsymbol{y}}_j^{\text{marginal},a}, \hat{\boldsymbol{y}}_j^{\text{marginal},b}, p_\phi(\cdot|\boldsymbol{T}, \boldsymbol{z})\right) \\ =& \log\left(\sum_{i=1}^{d} \mathbf{1}\left(\hat{\boldsymbol{y}}_{j,i}^{\text{marginal},a} = 1\right) p_\phi(\boldsymbol{g}_i|\boldsymbol{T}, \boldsymbol{z})\right) \\ +& \log\left(\sum_{i=1}^{d} \mathbf{1}\left(\hat{\boldsymbol{y}}_{j,i}^{\text{marginal},b} = 1\right) p_\phi(\boldsymbol{g}_i|\boldsymbol{T}, \boldsymbol{z})\right). \end{aligned}$$

We essentially maximize the log likelihood of the ground-truth goal pairs under the marginal goal distributions.

In the toy example, we can guide the CVAE model to perfectly separate the goal pairs into two interaction modes. The result is the same as shown in Fig. 3.3b. The interaction modes can be perfectly identified because the goal pairs from different clusters happen to have distinct coordinates in both dimensions in our toy example. If only one of the agents changes his behavior in different modes, the pseudo marginal labels alone will not be helpful.

Pseudo Interaction Labels

The last type of pseudo labels we introduce allows us to incorporate domain knowledge on interaction in a flexible way, which we refer to as pseudo interaction labels. From the perspective of a downstream planner, we want the latent space to distinguish specific interaction modes for efficient planning and risk evaluation (e.g., collision vs. no collision, yielding vs. passing). If we know that these interaction modes can be identified with certain features, we can design the corresponding pseudo interaction labels as follows:

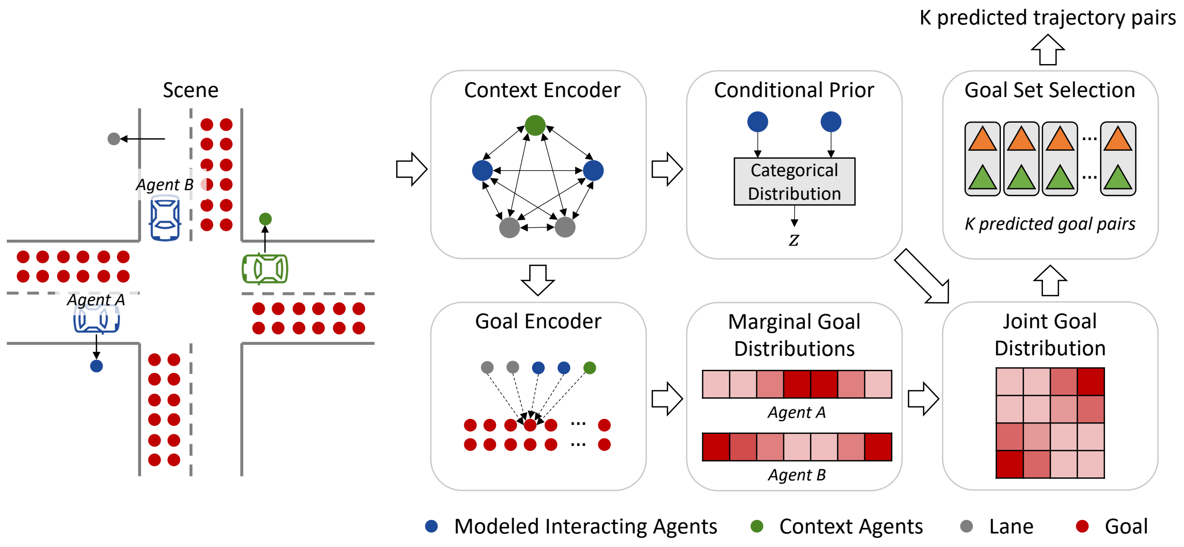
$$\hat{\boldsymbol{y}}_{j,i}^{\text{interact}}(\boldsymbol{T}) = \mathbf{1}\left(h(\boldsymbol{T}, \boldsymbol{g_i}) = h(\boldsymbol{T}, \boldsymbol{g}_j)\right),$$

where the function h maps the goal pair and initial states to a vector of discrete variables characterizing the interaction. We assign positive labels to the goal pairs that have the same features as the ground-truth goal pair. This indicates that they are under the same interaction mode as the ground-truth one. We note that pseudo interaction labels unify the interaction rules that have been applied in prior work as regularization (e.g., collision penalty [130, 124]). However, pseudo interaction labels are only applied to the distribution decoded from the latent variable to which the ground-truth goal pair belongs. In other words, we only require that there be an interaction mode in the latent space that is consistent with the ground truth rather than requiring all the predicted goal pairs to satisfy the constraints. As a result, we can avoid over-regularization and unnecessary bias.

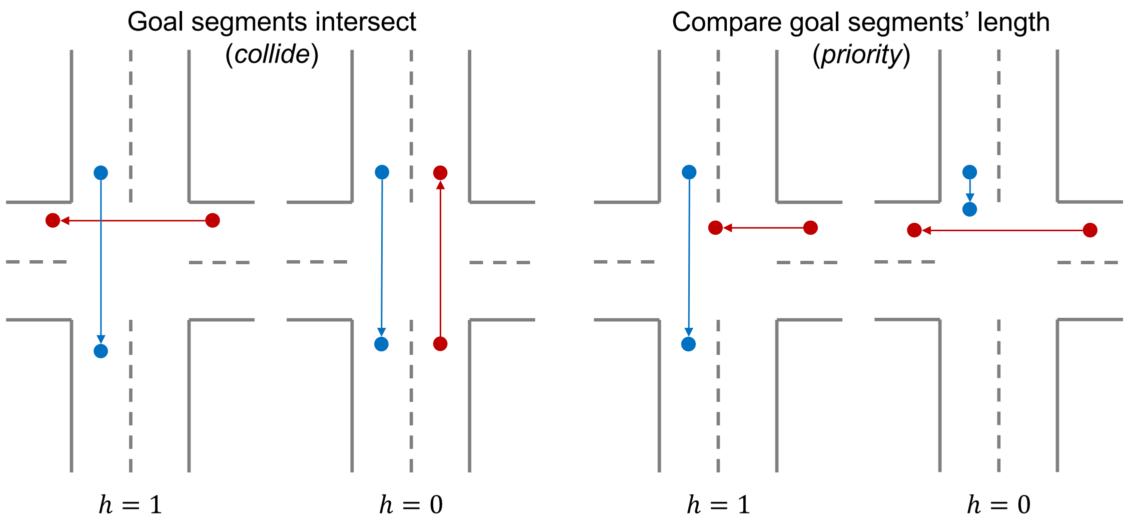
Regarding the loss function, maximizing the log likelihood of positive goal pairs could be misleading. There could be a large ratio of goal pair candidates under the same interaction mode. Inspired by [64], we adopt a loss function to minimize the probabilities of negative labels:

$$\begin{aligned} & f^{\text{interact}}\left(\hat{\boldsymbol{y}}_j^{\text{interact}}, p_\phi(\cdot|\boldsymbol{T}, \boldsymbol{z})\right) \\ =& \sum_{i=1}^{d} \mathbf{1}(\hat{\boldsymbol{y}}_{j,i}^{\text{interact}} = 0) \log\left(1 - p_\phi(\boldsymbol{g}_i|\boldsymbol{T}, \boldsymbol{z})\right). \end{aligned}$$

In this toy example, we adopt an interaction feature indicating which agent has longer displacement in 20 steps, i.e., $\mathbf{1}\left(s_{a,0} - s_{a,20} > s_{b,0} - s_{b,20}\right)$. With this feature, we can identify which agent decides to yield. By incorporating the pseudo interaction labels, we are able to separate the interaction modes and obtain a model similar to the one shown in Fig. 3.3b.



(a) Overall Model Architecture



(b) Pseudo Interaction Labels

Figure 3.4: Overall Model Architecture and Pseudo Interaction Labels.

3.5 Framework Architecture

In this section, we introduce the architecture of the model we propose for interactive trajectory prediction. As illustrated in Fig. 3.4a, the model consists of three modules: 1) A marginal goal prediction module that predicts the goal distribution of each interacting agent separately; 2) A joint goal prediction module that explicitly models the joint distribution of goal pairs based on the predicted marginal distributions; 3) A trajectory completion module that predicts the trajectory of each agent conditioned on sampled goal points. We also introduce the architecture of each module and the training scheme. Finally, we describe the goal selection method we use to query the model during online inference.

3.5.1 Modules

Marginal Goal Prediction

We choose DenseTNT [42] as the backbone model when designing the marginal goal prediction module. Concretely, we extract features of high-definition (HD) maps and agents using the vectorized encoding method proposed in [36]. Subsequently, we use the obtained scene context embeddings to generate goal embeddings for a dense goal set $\mathcal{G}$. The goal set is sampled from the HD maps to cover the drivable area of the modeled interacting agents. We follow DenseTNT and use the attention mechanism in [134] to extract local information between the goals and the scene. We denote the embeddings obtained at this stage for the dense goals and interacting agents as $\boldsymbol{F} \in \mathbb{R}^{|\mathcal{G}| \times d_g}$ and $\boldsymbol{L} \in \mathbb{R}^{2 \times d_v}$, respectively, where d_g and d_v are the dimensions of goal and agent embeddings.

The interaction prediction task of the Waymo Open Dataset has a prediction horizon of 8s. It is difficult to capture the multimodality in long-term trajectory distribution with a single goal point. We follow [43] to model the goal distributions in an autoregressive manner at 3s, 5s and 8s, respectively. To encourage the use of interaction information in goal prediction, we add an MLP to update the interacting agent embeddings at each time step as follows:

$$\hat{\boldsymbol{L}}_{t,i} = \text{MLP}\left(\boldsymbol{L}_i, \boldsymbol{L}_{-i}, \boldsymbol{F}_{k^i_{1:t-1}}, \boldsymbol{F}_{k^{-i}_{1:t-1}}\right),$$

where $\boldsymbol{F}_{k^i_{1:t-1}}$ collects the embeddings of the i^{th} agent's goals in previous time steps. Then, we predict the score of the k^{th} goal for the i^{th} agent at each time step as follows:

$$\phi^i_{t,k} = \frac{\exp\left(\text{MLP}(\boldsymbol{F}_k, \hat{\boldsymbol{L}}_{t,i})\right)}{\sum_{j=1}^{|\mathcal{G}|} \exp\left(\text{MLP}(\boldsymbol{F}_j, \hat{\boldsymbol{L}}_{t,i})\right)}. \tag{3.3}$$

In the training stage, we follow the well-known practice in autoregressive model training by feeding the ground-truth goals of the previous time steps into the model.

Joint Goal Prediction

With the marginal goal distributions at time step t, we first select the top-M goal candidates for each agent based on their scores and then model the joint distribution over the M^2 goal pair candidates. As mentioned in Sec. 3.4, we model the joint distribution with a CVAE and utilize the pseudo labels to induce an interpretable interactive latent space. The conditional prior encoder models the distribution of $\boldsymbol{z}$ conditioned on $\boldsymbol{L}$. The posterior encoder further conditions $\boldsymbol{z}$ on $\boldsymbol{F}_{k^1_{1:T}}$ and $\boldsymbol{F}_{k^2_{1:T}}$, i.e., the embeddings of the two agents' ground-truth goals. Both the conditional prior and posterior encoders are modeled with simple MLPs. To decode the joint goal distribution from a sampled $\boldsymbol{z}$, we first obtain a joint agent embedding $\tilde{\boldsymbol{L}}_t \in \mathbb{R}^{1\times d_h}$ as follows:

$$\tilde{\boldsymbol{L}}_t = \mathrm{MLP}\left(\boldsymbol{L}, \boldsymbol{F}_{k^1_{1:t-1}}, \boldsymbol{F}_{k^2_{1:t-1}}, \boldsymbol{z}\right).$$

We obtain the features of goal pairs by concatenating the corresponding goals' embeddings and their marginal probabilities, denoted by $\tilde{\boldsymbol{F}}_t \in \mathbb{R}^{M^2\times d_h}$. Subsequently, we use an attention mechanism to gather the local information of goal pairs:

$$\begin{aligned}
\boldsymbol{Q}_t &= \tilde{\boldsymbol{F}}_t \boldsymbol{W}^Q, \\
\boldsymbol{K}_t &= \left[\tilde{\boldsymbol{F}}_t \boldsymbol{W}^K_m ; \tilde{\boldsymbol{L}}_t \boldsymbol{W}^K_v\right], \\
\boldsymbol{V}_t &= \left[\tilde{\boldsymbol{F}}_t \boldsymbol{W}^V_m ; \tilde{\boldsymbol{L}}_t \boldsymbol{W}^V_v\right], \\
\bar{\boldsymbol{F}}_t &= \mathrm{softmax}\left(\frac{\boldsymbol{Q}_t \boldsymbol{K}_t^\intercal}{\sqrt{d_k}}\right)\boldsymbol{V}_t,
\end{aligned}$$

where $\boldsymbol{W}^Q, \boldsymbol{W}^K_m, \boldsymbol{W}^K_v, \boldsymbol{W}^V_m, \boldsymbol{W}^V_v \in \mathbb{R}^{d_h\times d_k}$ are matrices for linear projection, d_k is the dimension of query/key/value vectors. We predict the score of the k^{th} goal pair at the given time step in a similar way as in Eqn. (3.3).

Trajectory Completion

Our trajectory completion module is similar to the ones in [151] and [42]. Given a sequence of goals, we pass their embeddings to a simple MLP to decode the whole trajectory. The trajectories for the two agents are decoded separately. In the training stage, the teacher forcing technique is applied by feeding the ground-truth goal sequences into the model when training the trajectory completion module.

3.5.2 Training Scheme

To train the overall model, we first train the marginal goal prediction module together with the trajectory completion module. The loss function is the same as in [42]. Afterwards, we freeze the parameters of these modules and train the joint prediction module. The objective function is essentially ELBO, in addition to the auxiliary loss corresponding to the three types of pseudo labels introduced in Sec. 3.4. In particular, the pseudo interaction labels are defined for each pair of segments connecting goal points at neighboring time steps (e.g., 0s–3s, 3s–5s, 5s–8s). As illustrated in Fig. 3.4b, for each pair of segments, the pseudo interaction labels are two indicators showing: 1) if the goal segments of the two vehicles intersect; 2) if the goal segment of the first vehicle is longer than the one of the second vehicle. The first feature gives us a hint regarding whether the two vehicles have a conflict zone along their driving directions. The second feature provides a necessary condition on their right of way. If a vehicle has the right of way, it should have a larger average speed than the vehicle yielding to it.

3.5.3 Goal Selection

At test time, we need to select a final small number of goal pairs for prediction. The most widely used algorithm is NMS. However, such a heuristic approach is difficult to tune and is not guaranteed to find the optimal solution. To address this issue, an optimization-based approach is proposed in [42] to select a goal set from a predicted distribution. While we may adopt it to select goal pairs at a single time step, it remains heuristic when sampling from the latent space as well as the autoregressive model. To ensure a fair comparison among the different model variants studied in Sec. 3.6, inspired by [24], we instead first randomly sample N sequences of goal pairs and then fit them to a Gaussian mixture model (GMM) to obtain the final K goal pair sequences.

3.6 Experiments

We evaluate the proposed prediction framework on the Waymo Open Motion Dataset. In particular, we focus on the interaction prediction task, where the future trajectories of an interacting pair for the next 8s are predicted, given the historical observation of the past 1s. With the experiments, we would like to answer the following questions: 1) Do the proposed pseudo labels help induce a meaningful latent space? 2) If learning a meaningful latent space, does the CVAE structure improve performance and sampling efficiency?

Dataset and Data Pre-processing

The Waymo Open Motion Dataset is a large-scale motion forecasting dataset for autonomous driving containing data mined on interactive behaviors across a diverse set of road geometries. It provides more than 570 hours of unique data of over 1750 km of roadways, with over 100000 scenes. In selected interactive subsets, i.e., interaction pairs of vehicles, pedestrians and cyclists are labeled for the interaction prediction task. In our experiments, we used the subset of the dataset with labeled interaction pairs of *vehicles* for training and evaluation. We followed the data processing routine in [42] to calibrate the coordinates and segment the road polylines. One issue we encountered was how to select the region of interest on the HD map. The most commonly adopted method is to define the region of interest as the union of circles or rectangles centered at the target agents. However, since the prediction horizon in the Waymo dataset is extensive, the remaining road segments could be numerous. Since we adopted a dense goal candidate set covering the road segments, there was a large number of goal candidates. Because of the attention mechanism, the computational complexity of goal encoding scales exponentially with the number of goal candidates. To tackle this problem and achieve efficient encoding process, we further filtered the road segments based on traveling distance using a graph search. The algorithm is similar to the one used in [127], the details of which will be presented in Chapter 6.

Model Variants

Our experiment mainly focuses on ablation studies, comparing our model against its variants. We compare the performance of three models: 1) The *Joint-Vanilla* model, which is our joint prediction model without the pseudo labels; 2) The *Joint-NonInteract* model, uses pseudo distance and marginal labels in addition to the vanilla version; 3) The *Joint-Full* model, which is the one we propose, i.e., the joint prediction model with the auxiliary losses corresponding to all the proposed pseudo labels (i.e., distance, marginal, interaction). We do not experiment with other methods from the literature for the following reasons: 1) The Waymo Open Motion Dataset is a newly released dataset with few reproducible prior works, especially for the interaction prediction task; 2) Our core contribution lies in utilizing the novel pseudo labels to induce a non-trivial and interpretable latent space. Achieving state-of-the-art performance on the benchmark is not our objective.

Training Settings

To train the overall model, we first train the marginal goal prediction module together with the trajectory completion module following most of the hyper-parameters introduced in [42]. Then, we select $K = 65$ goal candidates based on the marginal probability for each agent and train the joint goal prediction module. We add annealing on the KL divergence weight to mitigate KL vanishing as a complement to the pseudo labels.

Evaluation Metrics

We use these metrics—minADE, minFDE, and mAP—introduced in [25], to evaluate the interactive prediction performance. The metrics for joint prediction involve the predicted trajectories of two interacting vehicles at the same time. The definitions of minADE and minFDE are similar to the single-agent case. However, the displacement errors are computed between the trajectory pairs and their ground-truth labels jointly. The mAP metric is a newly proposed metric for Waymo Open Challenge[2]. It computes the mean average precision over eight different ground-truth trajectory primitives defined based on the dataset.

3.6.1 Empirical Prediction Results

In Table 3.1, we compare the prediction performance of the model variants on the validation dataset. We evaluate the prediction over 20000 validation samples in 3s, 5s, and 8s time horizons with the metrics introduced earlier. The results for the three time horizons are averaged and reported. We show the evaluation results based on different numbers of

[2]Please refer to `https://waymo.com/intl/en_us/open/challenges/` for more details.

samples before GMM fitting, with $N = 8$ and $N = 120$. From Table 3.1, we can see that the prediction performance is sensitive to the sample number N. As N increases, the sampled trajectories are more likely to cover the multimodality in joint distribution, which leads to more diverse and accurate prediction after GMM fitting. From the table, we can conclude that both the *Joint-Full* and *Joint-Vanilla* models make better predictions when N is large, and the *Joint-Full* model has better performance with the same sample number N. Note that in online prediction, the maximum allowable N is directly determined by the required computational time. Our purpose is to obtain accurate and diverse predictions with a small sample number N to enable efficient online inference. To this end, we observe a larger improvement with the use of pseudo labels when $N = 8$ compared to $N = 120$.

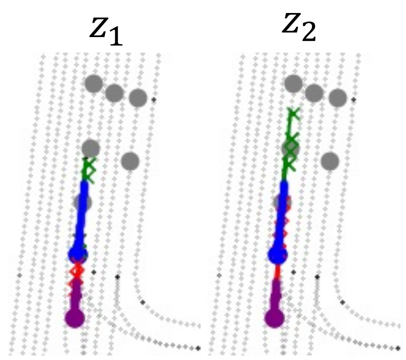
To evaluate our proposed joint prediction model in highly interactive scenarios, we select a set of strong-interactive cases from the validation dataset. These highly interactive scenarios essentially require joint modeling and sampling from the goal distribution of interacting agents and effectively demonstrate the strength of our proposed framework. We select the data samples where goal segments of two vehicles intersect. This is done using pseudo interaction labels, introduced in Sec. 3.5.2. The prediction results of models using different pseudo labels are shown in Table 3.2. Since mAP is extremely sensitive to the hyper-parameters when N is low, we do not consider the mAP comparison for quantitative analysis. As the number of selected samples is low compared to the complete validation set (351 of 20000), we evaluate each model three times and report the mean and standard deviation. We observe a significant improvement in prediction performance and stability by adding interaction pseudo labels (*Joint-Full* model). With a well-trained latent space, we are more likely to cover more interaction patterns in the finite $N = 8$ samples. This is essential for strong-interactive cases in which multiple interaction modes exist and the reason behind the observed improvement.

3.6.2 Latent Space learned by CVAE with Pseudo Labels

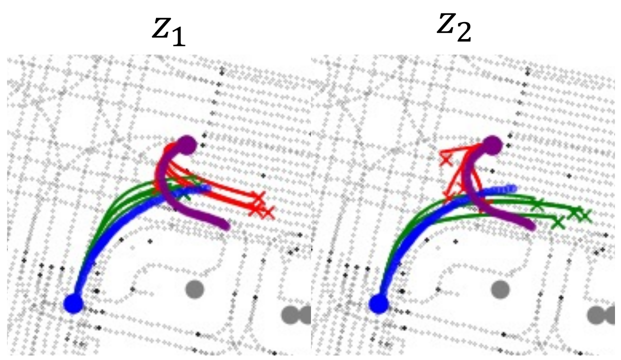
During training, we indeed observed that pseudo labels, especially marginal pseudo labels, help avoid KL vanishing in most cases. To demonstrate the interactive pattern encoded by latent space, we visualize predicted trajectories for selected interactive scenarios from the

Table 3.1: Validation Results on All Samples

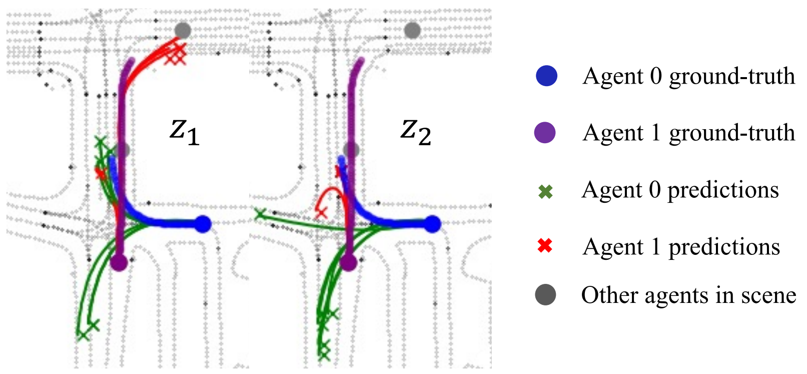
Method	minADE	minFDE	mAP
Joint-Vanilla, $N = 120$	1.58	3.44	0.078
Joint-Full, $N = 120$ (Ours)	1.55	3.33	0.084
Joint-Vanilla $N = 8$	1.98	4.28	0.020
Joint-Full $N = 8$ (Ours)	**1.89**	**4.09**	**0.027**



(a) Different speed



(b) Different right of way



(c) Different route selection

Figure 3.5: Comparison of six sampled first-step goal predictions conditioned on two different selected latent z value using the *Joint-Full* model. Different interaction modes can be found in different latent values, meaning we have learned a meaningful latent space.

dataset, as shown in Fig. 3.5. We use the *Joint-Full* model with different latent variables in the same scenario to make these predictions. Given the historical information, we sample six different goal pairs from the joint goal distribution prediction model conditioning on two different discrete latent variable z with the largest probabilities. In Fig. 3.5, we can clearly see two different interaction modes with different z. The agents either change their speed, route, right of way, or combinations of previous features when switching the latent variables. Meanwhile, the *Joint-Vanilla* model fails to give a separated latent space (e.g., predictions sampled from different latent variables are similar) in the same scenarios because of vanishing KL. This shows that our proposed model indeed learns an interpretable latent space capturing the interaction modes inherited from the pseudo labels.

Table 3.2: Ablation Study on High-Interactive Samples

Method	minADE	minFDE
Joint-Vanilla, $N = 8$	1.89 (0.06)	4.11 (0.17)
Joint-NonInteract, $N = 8$	1.88 (0.04)	4.02 (0.07)
Joint-Full, $N = 8$ (Ours)	**1.76** (**0.02**)	**3.78** (**0.04**)

3.7 Chapter Summary

In this chapter, we study the interaction prediction problem with a goal-conditioned prediction model. To develop an interpretable and sampling-efficient prediction model, we leverage the CVAE framework to explicitly capture diverse interaction modes in joint goal distribution. While the discrete latent space we adopt is generally more interpretable than a continuous latent space, we find that the vanilla model is prone to posterior collapse, resulting in a totally uninformative latent space. To mitigate this issue, we propose the introduction of auxiliary loss functions defined with pseudo labels incorporating domain knowledge on interaction. We show that the proposed pseudo labels can effectively induce an interpretable and meaningful interactive latent space and further improve prediction performance.

We find the proposed pseudo labels particularly interesting as a principled and flexible way to incorporate domain knowledge to shape the latent space. Since we do not require the generation process of the pseudo labels to be differentiable, we can design pseudo labels reflecting arbitrary knowledge about the proximity between goal pairs. In contrast to GRI, where a structured reward function is enforced, the pseudo labels only introduce a soft regularization, making it easier to balance between interpretability and performance. While a fully interpretable model should always be the ultimate goal, methods like pseudo labels provide a practical way to improve the interpretability of current state-of-the-art models without compromising performance.

Chapter 4

Interpretable Policy Transfer via Robust Model-based Control

4.1 Introduction

In the last two chapters, we focused on improving the interpretability of driving behavior modeling approaches. In this chapter, we turn to the downstream planning and control task, developing an interpretable learning-based driving policy. Learning intelligent and reliable driving policy networks (PNs) has been an ongoing challenge in deep learning and control. Although conventional planning-control approaches [73] are shown to have the capability to control autonomous vehicles, deep learning-based methods are promising for tackling more complicated and interactive scenarios, especially those rare corner cases that are hard to program manually. However, robustness has been the main drawback that prevents the application of neural network policies in autonomous driving. The policy networks are over-specified for the source domain in which the policy networks are trained and thus often fail when the dynamics of the target vehicle deviate from the training setting or when the target vehicle is affected by external disturbances due to strong winds or body incline. The difference between the source and target settings is called the modeling gap. As the modeling gap is an inevitable barrier to the deployment of deep learning driving policies in autonomous driving, we aim to bridge the modeling gap by achieving a fast and safe transfer of driving policy.

Prior learning-based policy transfer efforts attempted to achieve their objective using transfer learning and meta learning. Various contributions in these areas have indeed succeeded in getting source policies to work in the target settings, but their applications in autonomous driving are limited due to safety concerns. Overall, such learning-based policy transfer methods embed transferable representations into *black-box* neural networks and hope for the best in the target domain and thus are not transparent and reliable. We seek to solve the policy transfer problem using an alternative tool: robust control (RC), and propose a generic PN-RC transfer framework. The PN-RC framework aims to solve the policy transfer problem between domains with different vehicle dynamics models. In this framework, the policy network is applied to an imaginary setting in the source domain to generate a kinematic-level reference trajectory for the target vehicle. In the target domain, where we assume prior knowledge of the dynamics of the target vehicle, a robust controller is designed to track the reference trajectory tolerating the modeling gap. This framework is generic because it can be generalized to other control tasks, and different kinds of robust controllers can be used for tracking. Compared to other transfer learning methods, the proposed framework has two fundamental advantages: 1) transferring the *interpretable* kinematic features makes the framework transparent and reliable; 2) the stability and response of the low-level controller can be analyzed to design the optimal controller parameters.

In this chapter, we investigate one realization of the PN-RC framework. For the policy network, we implement a hierarchical PAN [141], which can flexibly compose a set of elementary policy networks to tackle a variety of driving tasks. For the controller, we design an adaptive DOB robust tracking controller. The DOB controller is integrated with a novel reference smoothing algorithm, which improves the tracking performance when a dynamical re-planning scheme is involved. The simulation and preliminary experimental results validate that the proposed PN-RC architecture can zero-shot transfer the policy under a certain level of parameter variation and external disturbances.

The rest of this chapter is organized as follows. In Sec. 4.2, we summarize the related works on policy transfer and vehicle control. In Sec. 4.3, we present the PN-RC architecture for policy transfer. Then, we describe the high-level hierarchical PANs for motion planning and the low-level adaptive DOB-based lateral tracking controller. In Sec. 4.4, we present the simulation results, showing the effectiveness of the proposed method in actively rejecting the modeling gap and the external disturbances in sim-to-sim policy transfer tasks and comparing the proposed method with a wide range of baselines. In Sec. 4.5, the preliminary experimental results are presented, showing the capability of the proposed method to handle sim-to-real autonomous driving policy transfers.

4.2 Related Work

Neural Network Policy Transfer

Regarding pure learning-based policy transfer, one stream of literature examines the use of source domain randomization to embed robustness. The ensemble policy optimization (EPOpt) algorithm [101] randomly samples dynamic parameters from a prior distribution and optimizes the policy for cases with worst-performing dynamic parameters. [96] uses an RNN to take in historical paths, expecting the policy to be adaptive by implicitly identifying the parameters. A similar idea has been adopted in [147], but an online system identification module is explicitly introduced to identify the parameters. The estimated parameters are fed into a parameter-universal policy. All of these works are limited in that they only address the discrepancy in dynamic parameters, while robustness against external disturbances is excluded from consideration.

Another line of work involves model adaptation. [34] trains a neural network for a dynamics model and adapts its online local linear model for model-based control methods. In [22], a deep inverse dynamics model of the target system is trained, and transfer is achieved by executing inverse model outputs to reach the nominal state generated by performing source policy in the source environment. One problem with such approaches is that a feasible and accurate inverse model for feed-forward tracking can hardly be guaranteed in the target domain, unlike tracking based on feedback control. In addition, a neural network approximating system dynamics is non-trivial to find in practice. Furthermore, fine-tuning in the target domain is not desirable due to safety.

A method developed based on a similar idea to the one in this paper is reported in [49], in which a model predictive control (MPC) controller is designed to stabilize the target system around the nominal trajectory generated by consecutively applying the policy in the source system. Theorems on tube-based MPC ensure that the states are bounded under modeling errors. However, the bound cannot be explicitly found, and no asymptotic stability can be guaranteed. Moreover, solving online optimization problems is computationally expensive, while robust controllers are usually easier and faster for dealing with the trajectory tracking problem of automated vehicles.

Vehicle Lateral Control

Lateral control of an autonomous vehicle has been well established with many existing methods [91]. Two main categories of controllers have been investigated in the literature. For the first type of controller, tracking errors are defined in relation to a reference point on the reference path. Control laws are derived to stabilize the vehicle around zero tracking errors. The alternative method considers a section of the reference path or trajectory. A widely adopted approach is formulating the problem as an MPC problem. Compared to tracking a single reference point, MPC can handle a wider range of scenarios, especially extreme scenarios where an accurate and sophisticated vehicle model is required. However, the problem of tracking a dynamically re-planned reference path is nontrivial for MPC. If an MPC-based motion planner is used, the problem can be formulated as a hierarchical MPC problem [26][110]. Otherwise, designing an MPC controller for dynamically re-planning motion planners is not intuitive. Therefore, we adopt the first stream of control techniques and develop an intuitive and straightforward method to improve tracking performance with a re-planned path. In this chapter, we apply DOB together with the nominal feedback controller to enhance the robustness of the system. DOB is a robust control technique for rejecting disturbances with guaranteed stability for both linear [17] and nonlinear systems [16][19]. For a linear system, given a nominal stabilizing controller, the Q-filter in DOB can be an arbitrary stable filter to shape the sensitivity function as desired [20]. A linear DOB has been applied for the robust path tracking of automated vehicles, and its effectiveness has been verified by experiments [102].

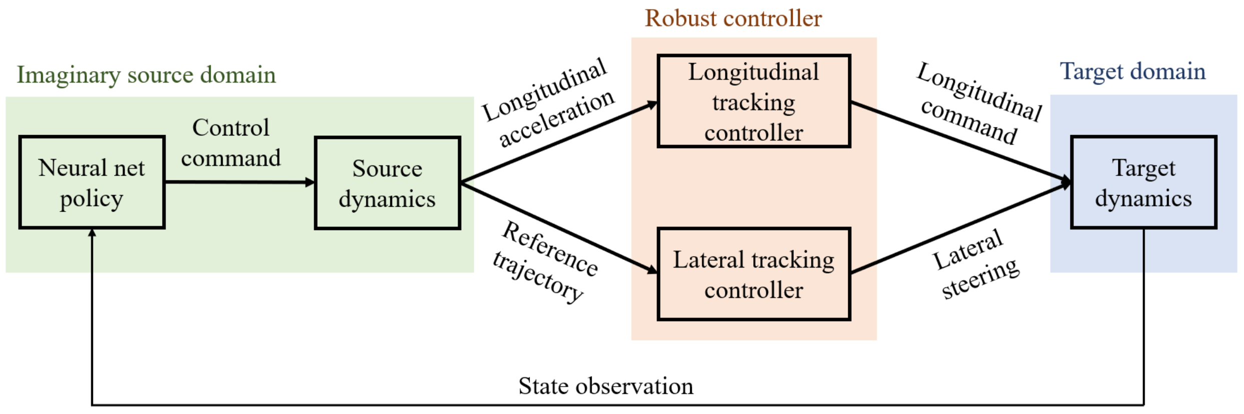


Figure 4.1: Target domain on-line implementation of the PN-RC architecture

4.3 Methodology

4.3.1 PN-RC Framework

The proposed architecture is a two-layer planning-control architecture consisting of a learning-based policy network as the high-level planner and a robust low-level tracking controller. Therefore, it is called the PN-RC architecture. The use of the PN-RC architecture consists of two streams: the offline policy network training and the online policy transfer. The offline source domain, as presented in Sec. 4.3.2, is a standard training process in which the policy network maps the state observation to the control commands corresponding to the source domain vehicle dynamics. In the source domain, we construct an autonomous driving simulation environment in which the autonomous car interacts with the road, the other cars, and the traffic rules. The online policy transfer methodology is shown in Fig. 4.1 and elaborated in Sec. 4.3.3. The PN-RC system performs closed-loop tracking of the driving behavior generated by the pretrained policy in the imaginary source domain. Concretely, the system constructs an imaginary source domain based on the target domain state observation at each time step. The imaginary source domain is constructed with the same set of parameters as the source domain where the policy network is trained. We then apply the pretrained policy network to control the ego vehicle in the imaginary source domain to perform the driving task. The resulting longitudinal and lateral kinematic behavior of the ego vehicle is directly transferred to the target domain, serving as the reference for the ego vehicle in the target domain. Given the intermediate reference trajectory, we apply longitudinal and lateral closed-loop robust tracking controllers in the target domain to produce the actual control commands for the target ego vehicle. As the target environment makes one step forward, the new state observation is collected, and the system iterates. This architecture transfers the kinematic features without change, while robust tracking controllers compensate for the modeling gap and other disturbances.

There are two key underlying assumptions of the PN-RC framework: 1) The kinematic reference planned in the imaginary source domain is comparably satisfying for the target task; 2) It is feasible for the target vehicle to track the trajectories produced by the source vehicle. These assumptions are reasonable when the source and target vehicles and settings are similar. Our proposed PN-RC framework is generic in many aspects: 1) This architecture can transfer policy networks for various kinds of tasks in autonomous driving. It can also be applied in other robotics and control scenarios if the assumptions are satisfied, and a robust controller can be designed; 2) The proposed method places no restrictions on the structure of the policy network. One can use reinforcement learning (RL), imitation learning (IL), or other approaches to train the policy network; 3) Any RC method is compatible with the PN-RC architecture.

Some may argue that a more straightforward choice to generate the reference trajectory is to train a neural network mapping the state observation directly into a sequence of waypoints. We think this is less desirable than our proposed PN-RC framework because it has several shortcomings. First, the generated trajectory is not guaranteed to be feasible or smooth. In contrast, the reference trajectory in our framework is produced by simulation in the imaginary source domain. The vehicle model encoded in the source domain can guarantee its feasibility and smoothness. Second, it could be challenging to train the network, either by supervised learning or reinforcement learning. The state-space of the trajectory is high dimensional, especially for a long planning horizon. Therefore, it is difficult for RL to explore the trajectory space efficiently for an optimal policy. This also makes it challenging to design an appropriate objective function for regression. Moreover, the size of the neural network needs to be extremely large for high-dimensional output.

4.3.2 Neural Network Driving Policy

We adopt the PAN proposed in [141] as the policy network in our experiments. To construct PAN, we decompose driving tasks into a set of attributes. The attributes are classified into two types: base and add-on attributes. The base attribute defines the canonical driving task, whereas the add-on attributes define additional requirements altering the canonical task. In our experiments, we consider a driving task involving one base attribute, lane tracking (LT), and two types of add-on attributes, obstacle avoidance (OB) and speed limit (SL). The lane tracking attribute defines the fundamental driving task, where we control the vehicle to track the centerline of the closest lane. The OB attribute is defined for each obstacle, including surrounding vehicles and static road obstacles. We create an OB attribute defining a safety region for each obstacle. The SL attribute specifies the SL subject to traffic rule regularization. We denote a driving task with n obstacles and l SL constraints as:

$$LT \oplus \sum_{i=1}^{n} OB_i \oplus \sum_{k=1}^{l} SL_k. \tag{4.1}$$

A PAN policy consists of a series of attribute networks, with each network trained for one specific attribute of the driving task. A base attribute network outputs a canonical reference control command to solve the elementary task. Each add-on attribute network defines a feasible set in the action space consisting of actions satisfying the specification defined by the add-on attribute. As illustrated in Fig. 4.2, the overall control command is computed by projecting the reference control command into the intersection of all the add-on feasible sets. We essentially solve a constrained optimization problem by projection, subject to the add-on feasibility constraints. To ensure fast online computation, we allow the output of each add-on attribute network to define a half-space. The resulting optimization problem is then a quadratic program. Since the resulting control command satisfies the specifications of all the attributes, the PAN policy solves the overall driving task defined by all these attributes. By combining elementary attribute networks in this way, we can conveniently solve various driving tasks that share the same attributes without training a separate policy network for each task from scratch.

In our experiments, we develop a PAN policy for the driving task defined in Eqn. (4.1) and in a source domain where the vehicle dynamics is governed by the point-mass model. Each vehicle has a state vector consisting of the planar coordinates, speed, and yaw angle, i.e., $s_i = [x_i, y_i, v_i, \theta_i]^T$. The control inputs are the longitudinal acceleration a_i and angular velocity $\dot{\theta}_i$, i.e., $u_i = [a_i, \dot{\theta}_i]^T$. The discretized point-mass model is defined as follows:

$$s_i(k+1) = s_i(k) + \begin{bmatrix} v_i \cos(\theta_i) \\ v_i \sin(\theta_i) \\ u_i \end{bmatrix} \cdot dt, \tag{4.2}$$

where k is the time index and dt is the sampling time. More details on PAN policy design can be found in [141, 126].

4.3.3 Robust Trajectory Tracking

The reference trajectory generated with the neural network policy is a time series of state vectors, $(x_{0,k}, y_{0,k}, v_{0,k}, \theta_{0,k})$ for $k = 1, 2, ..., h$, where h is the number of time steps within the planning time horizon. A robust tracking controller is required to track the trajectory in the presence of model uncertainty and external disturbances. A common practice to simplify the control problem is decomposing it into longitudinal speed control and lateral steering control problems. These two problems are then considered separately. Accurate modeling of the longitudinal dynamics, especially the powertrain, is difficult. As a result, designing a longitudinal controller with guaranteed robust stability for the actual vehicular system is hard. Nonlinear control techniques, such as the direct Lyapunov approach, can be applied to handle longitudinal model uncertainties [5]. However, in practice, a well-tuned PID controller should be sufficient as long as a smooth speed profile is generated. In this section, we focus on designing a robust lateral controller while assuming the speed profile can be perfectly tracked. The task of the lateral controller is to track the reference path consisting of $(x_{0,k}, y_{0,k}, \theta_{0,k})$. In particular, we adopt DOB to design a robust lateral controller rejecting modeling error and external disturbance. We start with a basic DOB-based controller with constant parameters and then design an adaptive extension of it.

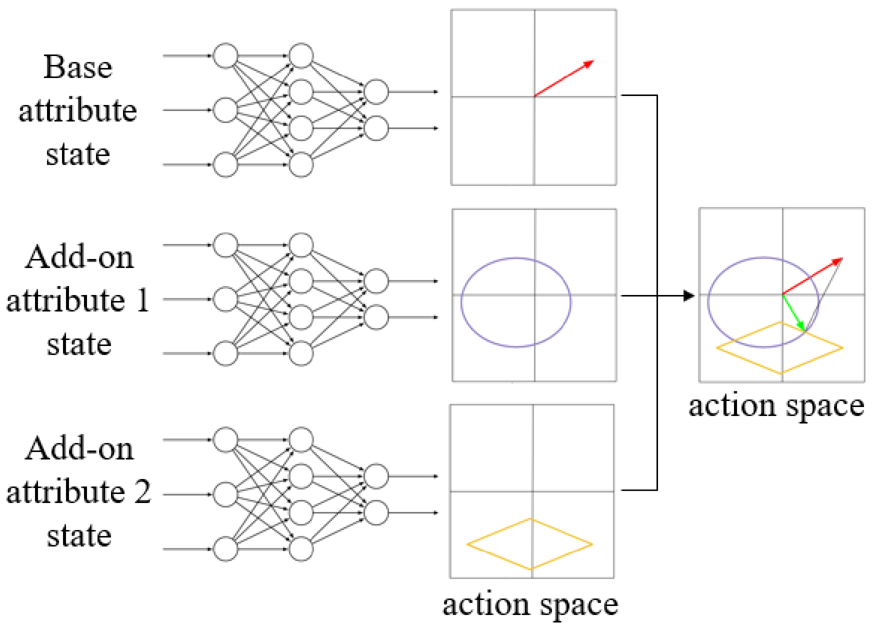


Figure 4.2: The architecture of the parallel attribute network (PAN). The output of the base attribute network is a reference action in the autonomous vehicle action space (the red vector), while the output of the add-on attribute networks is the satisfactory sets in the action space. The overall output is the projection of the reference action into the intersection of all the satisfactory sets (the green vector).

DOB-based Tracking Controller with Constant Parameter

To design the controller, we need to obtain an approximate linear model for the tracking problem. We adopt the constant speed linear bicycle model for lateral dynamics [100]:

$$\dot{x} = A \cdot x + B \cdot \delta, \tag{4.3}$$

where the state variable is $x = [v_y \ \theta \ \dot{\theta}]^T$, with v_y as the lateral velocity, θ as the yaw angle, and $\dot{\theta}$ as the yaw rate. The control input δ is the steering angle. Apart from the vehicle dynamics, we need to define a tracking model given a reference as well. Following [77], the nonlinear path-following model in the Serret-Frenet frame with an orthogonal projection is illustrated in Fig. 4.3 and described by the following equations:

$$\Delta\dot{\theta}_s = \dot{\theta} - \frac{\kappa_s v_s \cos(\Delta\theta_s + \beta_s)}{1 - \kappa_s \Delta y_s} \tag{4.4}$$

$$\Delta\dot{y}_s = v_s \sin(\Delta\theta_s + \beta_s)\,. \tag{4.5}$$

The tracking errors are defined with regard to a look-ahead point S. The point S is defined as the point along the longitudinal axis a distance of d_s from the center of gravity,

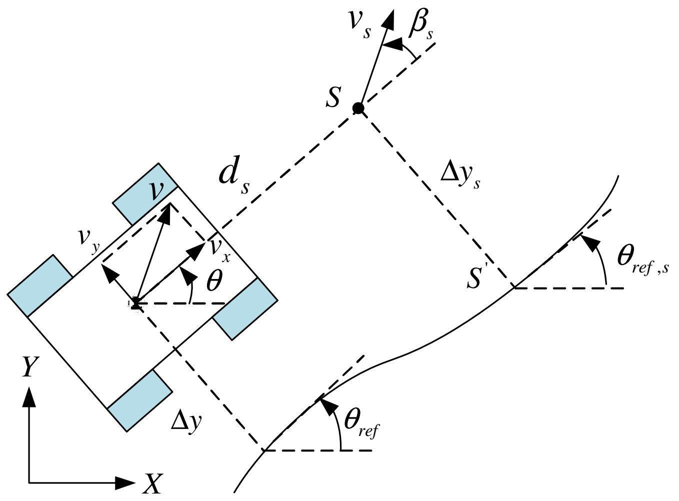


Figure 4.3: The nonlinear path-following model in the Serret-Frenet frame with an orthogonal projection.

and its orthogonal projection onto the reference path S'. The angular error $\Delta\theta_s$ is defined as the angle between the heading direction of the vehicle θ and the tangent direction of the reference path at the projected point $\theta_{s,ref}$, i.e., $\Delta\theta_s = \theta - \theta_{s,ref}$. The lateral distance error Δy_s is the signed distance between S and S'. In the equations, v_s is the magnitude of the velocity at S. The variable κ_s is the signed curvature of the reference path at S'. Under the small angle assumption, $\beta \approx v_y/v_x$, we can thereby approximate the slip angle at S, β_s, as follows:

$$\beta_s \approx \frac{v_x\beta + d_s\dot{\theta}}{v_x} = \frac{1}{v_x}v_y + \frac{d_s}{v_x}\dot{\theta},$$

where v_y is the lateral velocity at the center of gravity and v_x is the longitudinal velocity at the center of gravity. Thus:

$$\theta_{v_s} = \theta + \beta_s = \begin{bmatrix} \frac{1}{v_x} & 1 & \frac{d_s}{v_x} \end{bmatrix} \begin{bmatrix} v_y \\ \theta \\ \dot{\theta} \end{bmatrix} = C \cdot x. \tag{4.6}$$

By linearizing Eqn. (4.4) and (4.5) at zero tracking errors, i.e., $\Delta\dot{\theta}_s = 0$ and $\Delta\dot{y}_s = 0$, as well as zero lateral velocity, we can obtain the following linearized tracking model:

$$\begin{aligned} \Delta\dot{\theta}_s &= \dot{\theta}_{v_s} - v_x\kappa_{s,ref}, \\ \Delta\dot{y}_s &= v_x\Delta\theta_s, \end{aligned}$$

where $\kappa_{s,ref}$ refers to the curvature at the reference point. Equivalently, $\dot{\theta}_{s,ref} \approx v_x\kappa_{s,ref}$. Using the equations above and applying forward Euler discretization, the overall tracking model is obtained as the block diagram shown in Fig. 4.4, where T_s is the sampling time, and $\frac{T_s z^{-1}}{1-z^{-1}}$ is the transfer function of the discretized integrator. $G_v(z^{-1})$ refers to the vehicle dynamics that maps δ to θ_{v_s}, from which we can derive the nominal transfer function $G_{nv}(z^{-1})$ using Eqn. (4.3) and (4.6). As shown in the diagram, $\theta_{s,ref}$ can be considered a disturbance to the system.

The robust controller is designed based on the nominal model of the vehicle in the target domain. First, we design a proportional feedback controller as follows: $u_c = -k_1\Delta\theta_s - k_2\Delta y_s$. For analysis, we can further write the control law as follows:

$$u_c = -k_2(\frac{k_1}{k_2} + \frac{v_x T_s z^{-1}}{1-z^{-1}})\Delta\theta_s = -k_2 C_1(z^{-1})\Delta\theta_s, \tag{4.7}$$

where $C_1(z^{-1}) = \frac{k_1}{k_2} + \frac{v_x T_s z^{-1}}{1-z^{-1}}$. In the implementation, we hold $\frac{k_1}{k_2}$ constant and tune k_2 to achieve stability of the closed-loop system using the root locus. We inspect the stability for various v_x such that the resulting controller can stabilize the vehicle for the range of velocities in the given driving tasks.

Next, a DOB is added to the nominal feedback controller. The block diagram for the overall closed-loop system is shown in Fig. 4.5. DOB is inserted between $C_1(z^{-1})$ and k_2 so that the disturbance due to modeling error between Δy_s and $\Delta\theta_s$ can be rejected. $P_n(z^{-1})$ and $\hat{P}_n(z^{-1})$ are the nominal plant and nominal plant without delay, defined as follows:

$$P_n(z^{-1}) = z^{-2}\hat{P}_n(z^{-1}) = G_{nv}(z^{-1})C_1(z^{-1}) \tag{4.8}$$

In our case, the nominal plant $P_n(z^{-1})$ has a delay of two time steps. We make $\hat{P}_n(z^{-1})$ free of delay so that $\hat{P}_n^{-1}(z^{-1})$ is realizable. The closed-loop sensitivity function with DOB is:

$$S = \frac{1}{1 + k_2 G_v C_1 + z^{-2}(\frac{G_v}{G_{nv}} - 1)Q}(1 - z^{-2}Q). \tag{4.9}$$

We design Q as a second-order low-pass filter to reject low-frequency disturbances, as $\kappa_{s,ref}$ should have relatively low-frequency components for smooth reference trajectory. When modeling errors exist, we can use the robust control theorem to ensure robust stability [20]. However, modeling uncertainty between the linear and nonlinear models is involved in our problem, which complicates the analysis. Future efforts will be made to approximate the uncertainty bound for robustness specification. One last note is that P is time-varying since v_x is not constant in general. In our implementation, we design the DOB based on the final steady speed of the vehicle in simulations for simplification. In practice, an adaptive DOB can be designed, which will be introduced in the next subsection.

Adaptive Disturbance Observer

The performance of the time-invariant DOB-based controller has been validated with simulation results in [140]. However, the effect of DOB is limited because the cut-off frequency of the Q-filter has to be relatively low due to stability considerations. Meanwhile,

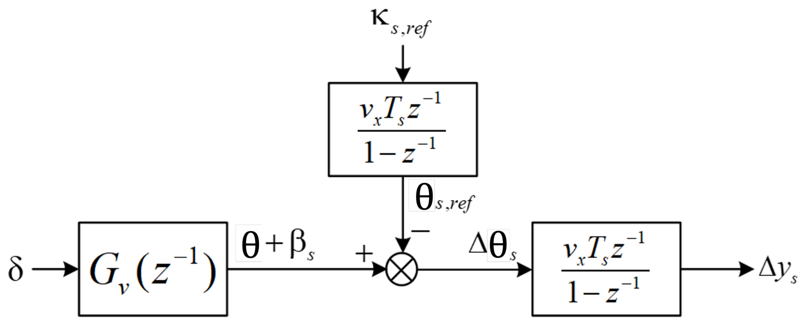


Figure 4.4: Block diagram of the linear tracking model.

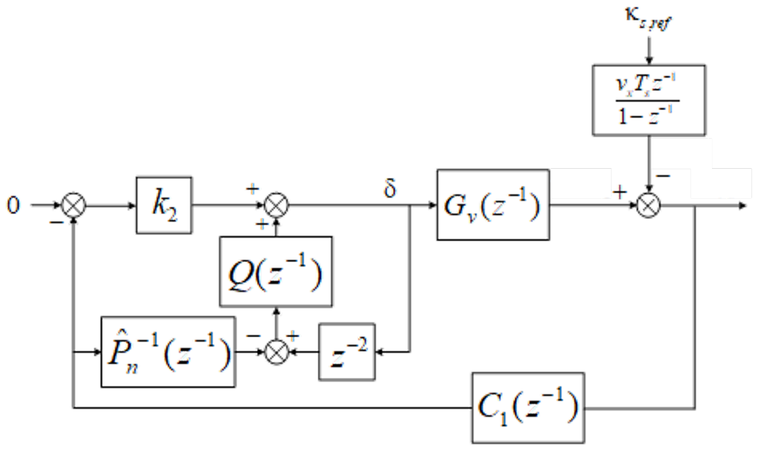


Figure 4.5: Block diagram of the closed-loop system.

high-frequency components are observed in the estimated disturbance signal. Consequently, the compensating input signal generated by DOB has a small magnitude compared to the estimated disturbance, meaning that only a limited range of disturbance is rejected by applying DOB. In this section, we extend the DOB to be adaptive to the varying states.

To tackle this problem, we first need a more accurate nominal model so that the cut-off frequency of the Q-filter can be increased. This can also reduce the high-frequency disturbance introduced by model linearization. The linear tracking model adopted previously is obtained by linearizing the model at zero tracking errors. Consequently, the inverse nominal model used in the DOB becomes inaccurate with increasing tracking errors. Moreover, the lateral behavior of the vehicle varies drastically with longitudinal speed. The constant speed linear bicycle model is not accurate when the speed of the vehicle deviates significantly from the designed operating speed. Therefore, if the tracking model and the bicycle model are linearized online in terms of current tracking errors and the longitudinal speed of the vehicle, the resulting adaptive inverse nominal model should be more accurate and perform better.

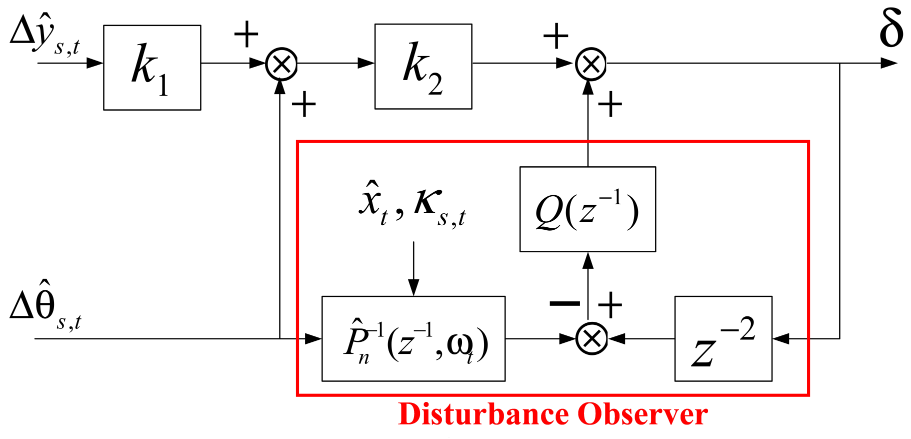

$\Delta\hat{y}_{s,t}$
k_1
+
+
k_2
+
+
δ
$\hat{x}_t, \kappa_{s,t}$
$Q(z^{-1})$
$\Delta\hat{\theta}_{s,t}$
$\hat{P}_n^{-1}(z^{-1},\omega_t)$
−
+
z^{-2}
Disturbance Observer

Figure 4.6: Block diagram of the feedback controller with adaptive DOB. The DOB is inserted into the loop of $\Delta\theta_s$. The inverse nominal model has parameters varying with the current states of the system, including $\Delta\theta_{s,t}$, $\Delta y_{s,t}$, v_s, and the curvature of the reference path κ_s.

Regarding the tracking model, if we linearize the Eqn. (4.4) and (4.5) for the current states of the system, a parameter-varying linear model that is more accurate than the time-invariant linear tracking model is obtained. Specifically, the model is linearized for the current estimated value of Δy_s, $\Delta\theta_s$, v_s and zero β_s, resulting in a system of linear equations in Eqn. (4.10) and Eqn. (4.11):

$$\Delta\dot{y}_s = \hat{v}_{s,t}\sin\Delta\hat{\theta}_{s,t} - \hat{v}_{s,t}\cos\Delta\hat{\theta}_{s,t} - \hat{v}_{s,t}\cos\Delta\hat{\theta}_{s,t}\Delta\theta_s + \cos\Delta\hat{\theta}_{s,t}v_y + d_s\cos\Delta\hat{\theta}_{s,t}\dot{\theta} \tag{4.10}$$

$$\begin{aligned}\Delta\dot{\theta}_s = {} & \frac{\kappa_s^2\hat{v}_{s,t}\cos\Delta\hat{\theta}_{s,t}\Delta\hat{y}_{s,t}}{(1-\kappa_s\Delta\hat{y}_{s,t})^2} - \frac{\kappa_s\hat{v}_{s,t}\sin\Delta\hat{\theta}_{s,t}\Delta\hat{y}_{s,t}}{1-\kappa_s\Delta\hat{y}_{s,t}} + \left(\frac{\kappa_s d_s\sin\Delta\hat{\theta}_{s,t}\Delta\hat{y}_{s,t}}{1-\kappa_s\Delta\hat{y}_{s,t}} + 1\right)\dot{\theta} \\ & - \frac{\kappa_s^2\hat{v}_{s,t}\cos\Delta\hat{\theta}_{s,t}}{(1-\kappa_s\Delta\hat{y}_{s,t})^2}\Delta y_s + \frac{\kappa_s\sin\Delta\hat{\theta}_{s,t}}{1-\kappa_s\Delta\hat{y}_{s,t}}v_y + \frac{\kappa_s\hat{v}_{s,t}\sin\Delta\hat{\theta}_{s,t}}{1-\kappa_s\Delta\hat{y}_{s,t}}\Delta\theta_s,\end{aligned} \tag{4.11}$$

where $\Delta\hat{y}_{s,t}$, $\Delta\hat{\theta}_{s,t}$, and $\hat{v}_{s,t}$ are the estimated states at t.

Since the model is not linearized for the equilibrium point, additional disturbances are introduced. Since the curvature κ_s should be quite small for normal driving scenarios, the disturbance in Eqn. (4.11) is negligible compared to the term in Eqn. (4.10). Moreover, the magnitude of longitudinal speed v_x should be much larger than that of the lateral speed v_y when the vehicle is moving. Consequently, v_x dominates v_y in Eqn. (4.10), and the lateral dynamics mainly affect Δy_s through $\Delta\theta_s$ and $\dot{\theta}$. Therefore, it is sufficient for the DOB to merely reject the disturbance from $\Delta\theta_s$. By doing so, disturbances entering into the lateral dynamics system are rejected, while the disturbances introduced through linearization will barely affect the performance of the DOB.

The overall structure of the controller is shown in Fig. 4.6. The control law of the feedback controller is the same as in the time-invariant case, i.e., $u = k_2(\Delta\hat{\theta}_{s,t} + k_1\Delta\hat{y}_{s,t})$. However, the DOB is inserted into the loop of $\Delta\theta_s$ instead of the loop of the weighted sum of $\Delta\theta_s$ and Δy_s. The DOB consists of two components: $Q(z^{-1})$ and $\hat{P}_n^{-1}(z^{-1}, \boldsymbol{\omega_t})$. The Q-filter is designed as a discrete-time low-pass filter with the following transfer function:

$$Q(z^{-1}) = \frac{a}{1 - z^{-1} + az^{-2}}. \tag{4.12}$$

The transfer function $\hat{P}_n^{-1}(z^{-1}, \boldsymbol{\omega_t})$ parameterized by $\boldsymbol{\omega_t}$ is the inverse of the nominal plant transfer function without delay from δ to $\Delta\theta_s$, computed based on Eqn. (4.10) and (4.11), along with the nominal linear bicycle model of lateral vehicle dynamics. The parameter vector $\boldsymbol{\omega_t}$ is a function of $\Delta\theta_{s,t}$, $\Delta y_{s,t}$, v_s and κ_s. The analytical format of the function can be computed offline. The estimated values are substituted into the function online, which can be completed in constant time. Assuming that the system varies slowly over time, the robust control theorem can be applied to determine the cut-off frequency of the Q-filter while ensuring robust stability [19]. Specifically, provided that the feedback controller stabilizes the system, if all the zeros of $\hat{P}_n(z^{-1}, \boldsymbol{\omega_t})$ are inside the unit circle, we have

$$\|\Delta\left(e^{j\omega}, \boldsymbol{\gamma_t}\right) Q(e^{j\omega})\| < 1, \forall\omega \tag{4.13}$$

with $\Delta\left(e^{j\omega}, \boldsymbol{\gamma_t}\right)$ being the unmodeled dynamics with respect to the nominal model $\hat{P}_n(z^{-1}, \boldsymbol{\omega_t})$ and the actual dynamics. However, modeling uncertainty between the linear and nonlinear models is involved in our problem, which complicates the analysis. Future efforts will be made to approximate the uncertainty bound for robustness specification.

Closed-loop Reference Path Smoothing

Apart from the linearization error, the motion planning procedure also introduces additional high-frequency disturbance into the system. The entire policy transfer framework is a hierarchical control system with trajectory as its intermediate output. It is prohibitive to analyze or optimize the entire system's behavior, especially when a dynamical re-planning mechanism is involved. The policy network generates a new reference trajectory based on the most updated observations for every few steps. Therefore, the reference path for the tracking controller varies rapidly compared to the length of the planning horizon. This results in a very different tracking behavior than in the case where the reference path is consistent over time because the closed-loop tracked path is not continuous. Although the dynamical re-planning mechanism is extensively adopted in practice, this issue has seldom been addressed in the literature. The controller is usually designed and analyzed merely in the context of a fixed and smooth reference path. Fortunately, the path-following model adopted here and the use of DOB allow us to analyze the problem and improve the tracking performance in the presence of a dynamically re-planned reference path.

At each time step, the tracking errors are computed with regard to a point S_t' on the current reference path. Therefore, the controller is equivalently tracking a curve from which all the reference points $\{S'_t\}_{t=1}^{t=\infty}$ are sampled. We refer to this curve as the closed-loop reference path. Although there is always a smooth curve interpolating $\{S'_t\}_{t=1}^{t=\infty}$, the curvature and its derivative are inevitably larger within the section interpolating reference points coming from reference paths generated at two adjacent time steps. Since the control system is discrete, step signals are introduced into Δy_s and $\Delta\theta_s$ when the trajectory is re-planned. Such jumps in the reference signal are undesirable because they lead to inappropriate transient response, e.g., overshoot. In motion control systems, a practical solution is designing a transient profile that the plant can follow reasonably [48]. Inspired by this idea, we construct a transient curve that smoothly joins the closed-loop reference path and the newly received planned trajectory.

This procedure can be formulated as a trajectory smoothing problem solved online by a numerical optimization method. We denote the newly received planned trajectory as P_t. The current closed-loop reference path is referred to as P_t^c. The points S'_{t-1} and S'_t are the reference points found at current and previous time steps. Their coordinates in Cartesian space are denoted as $x_{S'_{t-1}}$ and $x_{S'_t}$. The point $x_{ref,1}$ is computed based on the reference velocity at S'_{t-1}. Concretely, $x_{ref,1} = x_{S'_{t-1}} + \Delta t \cdot v_{S'_{t-1}}$, where $v_{S'_{t-1}}$ is the velocity vector at S'_{t-1} and Δt is the sampling time. The remaining reference points are the next $h-1$ points after $X_{S'_t}$ on P_t. The smooth transient trajectory is represented by the vector $x = [x_0^\intercal, x_1^\intercal, \cdots, x_h^\intercal]^\intercal \in \mathbb{R}^{2(h+1)}$. The reference trajectory is denoted as $x_{ref} = [x_{S'_{t-1}}^\intercal, x_{ref,1}^\intercal, \cdots, x_{ref,h}^\intercal]^\intercal \in \mathbb{R}^{2(h+1)}$. The optimization problem is formulated as a quadratic program as follows:

$$\min_x \; J(x; x_{ref}) = \|x - x_{ref}\|_Q^2 + \omega_2 \|x\|_S^2 \tag{4.14}$$

$$\text{s.t. } x_0 = x_{S'_{t-1}}, x_1 = x_{ref,1} \tag{4.15}$$

$$x_{h-1} = x_{ref,h-1}, x_h = x_{ref,h} \tag{4.16}$$

where $Q = I_{2(h+1)} + \omega_1 V^\intercal V$ and $S = A^\intercal A$. The matrix $V : \mathbb{R}^{2h \times 2(h+1)}$ and $A : \mathbb{R}^{2(h-1) \times 2(h+1)}$ are finite difference operators, as defined in [81], such that Vx is the velocity vector and Ax is the acceleration vector of the trajectory x. The weights $\omega_1, \omega_2 \in \mathbb{R}^+$ can be tuned to adjust the cost function. The first term $\|x - x_{ref}\|_Q^2$ penalizes the distance from the transient trajectory to the reference trajectory. The second term $\|x\|_S^2$ penalizes the magnitude of acceleration. The constraints (4.15) and (4.16) ensure continuity in coordinates and velocity. Safety constraints can also be added to enable a collision-free transient trajectory. If the constraints are non-convex, algorithms such as convex feasible set [81] can be applied to solve the problem in real time.

4.4 Sim-to-sim Policy Transfer

We first tested the proposed policy transfer framework with three examples of sim-to-sim policy transfer. The training scheme and results of the PAN are presented in Sec. 4.4.1, and the tested driving scenarios are described in Sec. 4.4.2. The vehicle is simulated with the point-mass model in the source domain. The simplicity in vehicle modeling enables

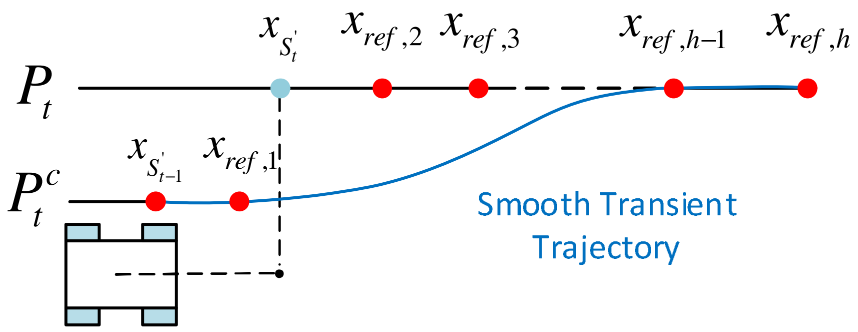


Figure 4.7: Closed-loop reference path smoothing. The received planned trajectory is denoted as P_t, and the current closed-loop reference path is denoted as P_t^c. The points S'_{t-1} and S'_t are the reference points found at current and previous time steps. Their coordinates are denoted as $x_{S'_{t-1}}$ and $x_{S'_t}$. The reference point $x_{ref,1}$ is computed based on the reference velocity at S'_{t-1}. The remaining reference points are the next $h-1$ points after $X_{S'_t}$ on P_t. The smooth transient trajectory is an optimized curve passing through $x_{S'_{t-1}}$, $x_{ref,1}$, $x_{ref,h-1}$ and $x_{ref,h}$.

faster convergence in policy training. Meanwhile, the performance of the policy in the target domain is less sensitive to the selection of the vehicle dynamics model in the training domain because of our policy transfer framework. In the target domains, the lateral dynamics are modeled with the nonlinear bicycle model [27] and Pacejka's tire model [90]. The longitudinal model is the same as in the source domain, where the acceleration command is perfectly executed by the vehicle. For each example, the policy was first transferred to the target domain with nominal model parameters. The parameters were consistent with the nominal linear bicycle model used for controller design. Subsequently, we further transferred the policy to target domains with uniformly distributed parameter variation or constant side force. We compared the performance of the PN-RC framework and the original policy networks in the target domains. In the target domains, the yaw rate command generated by the policy network is converted into a steering angle command based on the kinematic bicycle model:

$$\delta = \arctan\left(\frac{l_f + l_r}{l_r}\tan\left(\arcsin\left(\dot{\theta}\frac{l_r}{v}\right)\right)\right),$$

where l_f and l_r are the length from the front axis and the rear axis, respectively, to the center of gravity. The testing procedure is summarized in Fig. 4.8.

4.4.1 Preparation of the Policy Networks

As described in Sec 4.3.2, we apply PAN to simultaneously train policy networks for all the described driving tasks. In the source domain, the autonomous vehicle is assumed to start in the rightmost lane with a desired longitudinal speed of $v_{tar} = 10m/s \approx 22mph$. The

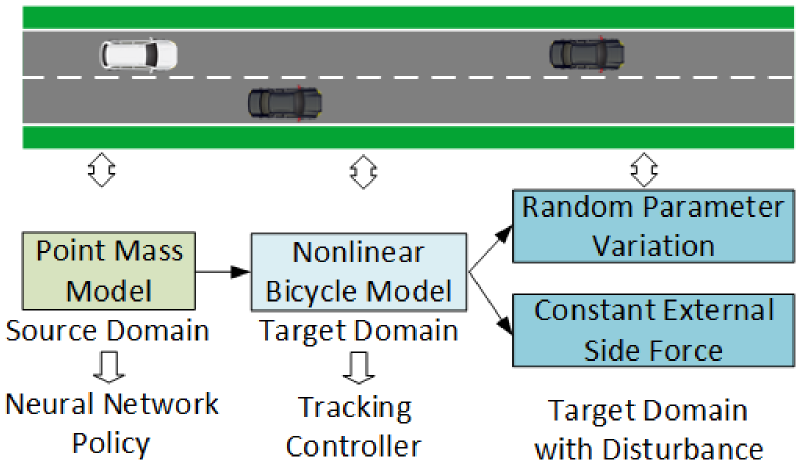


Figure 4.8: Diagram summarizing the sim-to-sim policy transfer procedure for a given test driving scenario. The policy is trained in the source domain with the point-mass model. It is transferred to the nominal target domain simulated with the nonlinear bicycle model. The tracking controller is developed based on the nominal model parameters. Subsequently, the policy and the PN-RC framework are tested in the target domains with disturbances.

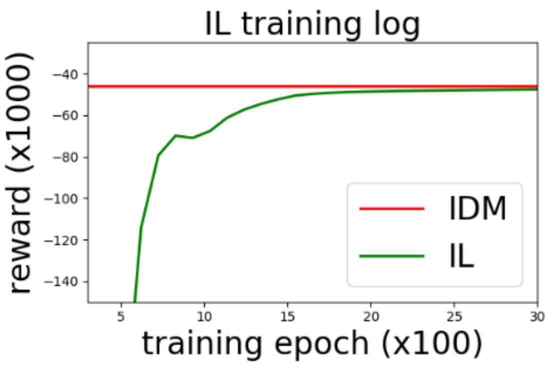
sampling time is $0.02s$. In our experiments, the base LT attribute module is trained using the proximal policy optimization (PPO) algorithm [113] with a reward function of

$$r(t) = 2 \cdot \Delta y_0^2 + (v_0 - v_{tar})^2. \tag{4.17}$$

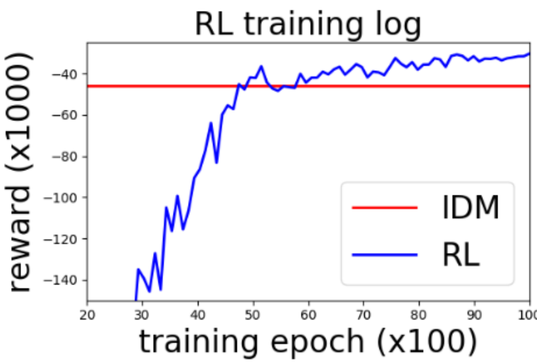
The add-on attribute modules are trained using IL, with analytical models serving as expert demonstrations. The RL training process normally takes three hours to converge, while the IL process typically takes only minutes. For IL of an add-on attribute network, the training data are collected from the cases with only the base LT attribute and the attribute to be trained to guarantee its independence from the other add-on attributes. For the OB modules, the collision avoidance constraints are approximated using the safety set algorithm (SSA) [79, 80]; for the SL modules, the speed constraints are derived using the IDM [60]. We then reform the constraints into the form of half planes and use them as the IL training data. After the IL converges, the trained attribute modules are fixed and assembled into PAN policies as needed. The projection of the reference action is a quadratic program problem, which is solved using the CVX solver [40, 41].

We also compared the performance of RL and IL in training the SL attribute module, as one can conveniently define a proper reward function for the $LT \oplus SL$ task. Making a minor modification to the reward function defined in Eqn. (4.17), we can design the reward function as follows:

$$r(t) = \begin{cases} -2 \cdot \Delta y_0^2 - (v_0 - v_{tar})^2, & \text{out of speed limit} \\ -2 \cdot \Delta y_0^2 - min(v_0 - v_{sl}, 0)^2, & \text{inside speed limit} \end{cases}$$



(a) The training log using IL



(b) The training log using RL

Figure 4.9: The training log using IL and RL.

Based on the new reward function, we fix the base LT module and train the add-on SL attribute network using RL and IL and then record and compare their performances (training logs shown in Fig. 4.9). The SL module trained using RL can achieve better performance than the analytical solution of IDM, and the SL module trained using IL can achieve comparable performance to the IDM.

In the PAN framework, one can use either RL or IL to train an attribute network, and further, for IL, the training data can come from either human labels or analytical solutions. In practice, RL training is the most difficult but has the best performance. While IL with analytically calculated labels is more convenient, its performance is not as good as RL. Therefore, the training approach should be selected based on the needs.

4.4.2 Driving Scenarios

Three different driving scenarios were implemented for the evaluation of the proposed policy transfer framework, in which various combinations of attribute modules were encoded into different PANs. More details of the attributes modules can be found in [141].

Following Curved Lane (CL)

The basic scenario is lane keeping, where the autonomous vehicle tracks a fixed reference path. Only the LT module is involved in the PAN policy. To evaluate the robustness against variation in lateral dynamics, the reference path is designed as a sine curve.

Collision Avoidance (CA)

The CA scenario involves two obstacle vehicles driving at a constant speed of $5m/s$ on a two-lane straight road. The autonomous vehicle is expected to make double-lane changes to overtake the two vehicles. LT modules and four OB modules are used. Two of the OB networks correspond to the obstacle vehicles, respectively. The other two correspond to the road edges.

Collision Avoidance under Speed Limit (CASL)

The last scenario is similar to the second one, but it has an additional SL constraint. An SL model is added to the PAN policy for the CA scenario to handle. The autonomous vehicle behaves differently due to the SL.

4.4.3 Software Implementation

The policy networks were implemented in Python with Tensorflow. The controllers were developed in MATLAB Simulink and converted into C++ code by code generation. The Robot Operating System (ROS) was adopted to manage the entire system and mimic the actual hardware system on the test vehicle. The driving scenario is simulated at a frequency of 50Hz. When the neural network policy directly controls the vehicle, it is operated at 50Hz as well. When the PN-RC framework is running, the planning module generates a reference trajectory with a planning horizon of 4.5s for every 0.6s. The tracking controller is operated at the frequency of simulation. Meanwhile, the policy networks output the longitudinal acceleration command at the same frequency. The tracking controller is configured such that $k_1 = 0.2$ and $k_2 = 0.5$. The Q-filter has a cut-off frequency of 10Hz. For the closed-loop path smoothing algorithm, $h = 80$ and $\Delta t = 0.02s$.

4.4.4 Tracking Performance

Before evaluating the policy transfer performance, we check whether the proposed controller can improve the tracking performance, especially when the reference trajectory is dynamically re-planned. We choose the scenario of CL following to compare the tracking performance. The tested controllers include the basic proportional controller without DOB, the controller with time-invariant DOB, and the controller with adaptive DOB. We also compare the cases with and without the proposed closed-loop trajectory smoothing algorithm. We investigate the effect of modeling error on tracking performance by shifting the nominal model parameters with uniformly distributed errors bounded by 45%. The effect of modeling errors on the transfer function is illustrated in [140]. A more comprehensive analysis of this subject can be found in [44].

The results are summarized in Table 4.1. If the smoothing algorithm is not applied, introducing DOB into the controller does not significantly improve tracking performance. The tracking errors Δy_s and $\Delta \theta_s$ can only be slightly decreased by adding DOB. Meanwhile, the average yaw rate increases, indicating worse transient behavior with larger oscillation. In contrast, the adaptive DOB achieves the smallest tracking errors and a comparable yaw rate after applying the trajectory smoothing algorithm. Moreover, it maintains consistent

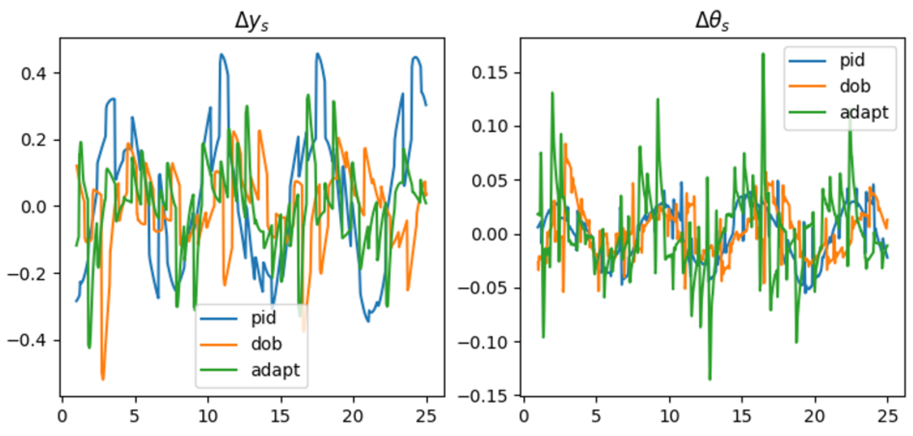


Figure 4.10: Tracking errors without trajectory smoothing. High oscillation occurs for both Δy_s and $\Delta\theta_s$. Neither DOB nor adaptive DOB can enhance the tracking performance to a significant degree due to the high-frequency disturbance introduced by re-planning.

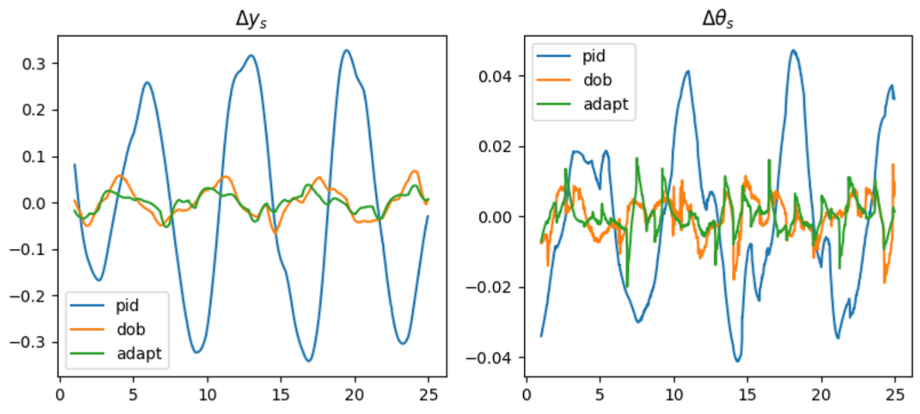


Figure 4.11: Tracking errors with trajectory smoothing. The frequency reduces for both Δy_s and $\Delta\theta_s$. DOB is able to compress the tracking errors after the smoothing procedure. This is because the high-frequency disturbance introduced by re-planning is filtered out.

performance with modeling error. Indeed, smoothing the closed-loop reference path is a key procedure for improving the tracking performance of the DOB-based controller, and its effectiveness is shown in Fig. 4.10 and Fig. 4.11.

4.4.5 Policy Transfer Evaluation

In this subsection, we evaluate the policy transfer performance using the proposed PN-RC architecture and its variants, including:

Table 4.1: Tracking performance comparison

Modeling Error	Variables	Without Smoothing		
		PID	DOB	Adaptive DOB
No	$\Delta y_s\ (m)$	0.212 ± 0.050	0.128 ± 0.034	$\mathbf{0.127 \pm 0.027}$
	$\Delta\theta_s\ (rad)$	0.025 ± 0.001	$\mathbf{0.023 \pm 0.001}$	0.038 ± 0.003
	$\dot{\theta}\ (rad/s)$	$\mathbf{0.073 \pm 0.004}$	0.076 ± 0.006	0.109 ± 0.013
Yes	$\Delta y_s\ (m)$	0.230 ± 0.059	$\mathbf{0.139 \pm 0.036}$	0.151 ± 0.035
	$\Delta\theta_s\ (rad)$	0.029 ± 0.001	$\mathbf{0.025 \pm 0.001}$	0.041 ± 0.002
	$\dot{\theta}\ (rad/s)$	$\mathbf{0.081 \pm 0.007}$	0.086 ± 0.008	0.119 ± 0.015
Modeling Error	Variables	With Smoothing		
		PID	DOB	Adaptive DOB
No	$\Delta y_s\ (m)$	0.211 ± 0.036	0.033 ± 0.001	$\mathbf{0.020 \pm 0.001}$
	$\Delta\theta_s\ (rad)$	0.023 ± 0.001	0.006 ± 0.000	$\mathbf{0.005 \pm 0.000}$
	$\dot{\theta}\ (rad/s)$	$\mathbf{0.068 \pm 0.004}$	0.073 ± 0.004	0.074 ± 0.005
Yes	$\Delta y_s\ (m)$	0.199 ± 0.035	0.034 ± 0.001	$\mathbf{0.020 \pm 0.001}$
	$\Delta\theta_s\ (rad)$	0.035 ± 0.022	0.006 ± 0.000	$\mathbf{0.004 \pm 0.000}$
	$\dot{\theta}\ (rad/s)$	0.073 ± 0.005	0.080 ± 0.005	$\mathbf{0.071 \pm 0.005}$

[1] Data are presented in the form of m$ean \pm$ std.

1. Use the baseline PN to directly control the vehicle.
2. Use the PN-RC architecture but with the basic proportional controller without the adaptive DOB, denoted as PN-PID, to control the vehicle.
3. Use the PN-RC architecture with the controller with the adaptive DOB, denoted as PN-DOB, to control the vehicle.

We simulated each test case ten times and recorded the total reward of each episode. The episode length was fixed to 1200 steps, corresponding to 24s. The accumulated reward was divided by the episode length to compute the average step reward. The mean and standard deviation were computed over the ten episodes. Subsequently, the mean step reward obtained by the baseline PN in the source domain was subtracted from the mean step reward for each test case in the corresponding driving scenario. This makes the policy transfer performance more apparent in the presented results, which are summarized in Table 4.2. For the test cases with model parameter variation, uniformly distributed errors bounded by 45% are added to the nominal model parameters. For the cases with constant side force, the magnitude of the side force is $3000N$. Both PN-PID and PN-DOB can complete the tasks

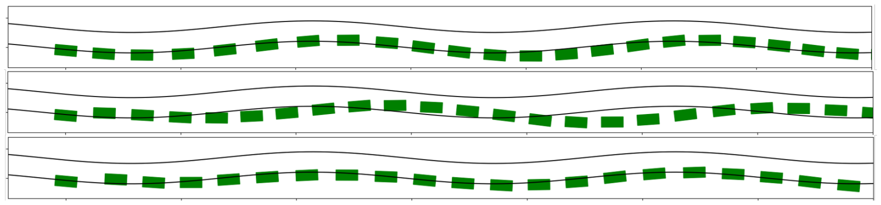
Table 4.2: Comparison of the performances of the baseline PN and PN-RC architectures

Task	Nominal Bicycle Model		
	PN	PN-PID	PN-DOB
CL	-14.51 ± 39.98	-13.31 ± 37.13	$\mathbf{10.69 \pm 21.73}$
CA	-153.60 ± 175.34	116.09 ± 36.06	$\mathbf{146.40 \pm 82.61}$
CASL	3.23 ± 12.39	6.69 ± 8.51	$\mathbf{10.89 \pm 8.60}$
Task	**Model Parameter Variation**		
	PN	PN-PID	PN-DOB
CL	-4484.18 ± 7736.17	-39.67 ± 79.19	$\mathbf{-2.40 \pm 38.40}$
CA	-202645.74 ± 571517.22	60.33 ± 68.19	$\mathbf{162.49 \pm 80.33}$
CASL	-48673.10 ± 78547.46	$\mathbf{11.53 \pm 8.00}$	9.03 ± 5.91
Task	**Constant Side Force**		
	PN	PN-PID	PN-DOB
CL	-29.66 ± 62.99	-42.50 ± 52.31	$\mathbf{9.60 \pm 29.38}$
CA	-13783.18 ± 41876.28	54.98 ± 82.63	$\mathbf{117.82 \pm 88.12}$
CASL	-6.24 ± 13.38	8.32 ± 8.92	$\mathbf{10.65 \pm 7.32}$.

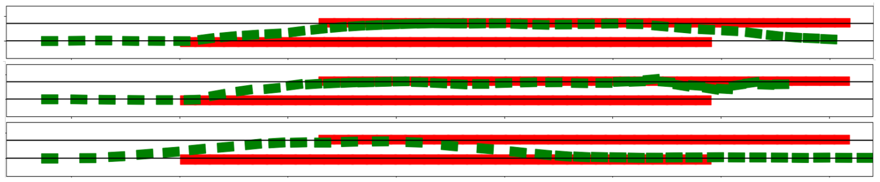
[1] Data are presented in the form of $mean \pm std$.
[2] The reward is divided by the episode length to compute the average step reward. The mean value of the average step reward in the source domain is then subtracted from the computed reward
.

with accumulative rewards comparable to PN in the source domain. Moreover, PN-DOB achieves the highest reward in most of the test cases. In contrast, the performance of the baseline PN is deteriorated by the modeling gap between the source and target domains. Failure cases occur when additional modeling errors or external disturbances exist. The vehicle is driven toward the road boundary and eventually leaves the drivable area.

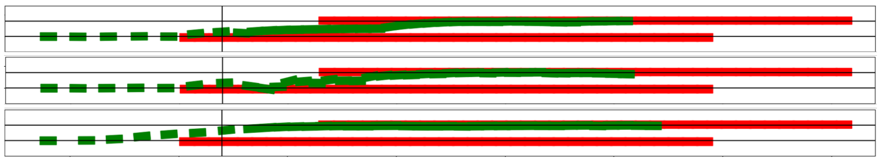
The robustness of PN-DOB can be further verified by visualizing the trajectories as in Fig. 4.12a to 4.12c. The green rectangles represent the ego vehicle. Moreover, the red rectangles represent the surrounding vehicles. Although the trajectories obtained with PN-DOB are slightly different from those in the source domain, all of the trajectories have smooth lateral behavior and keep the same high-level behaviors as in the source domain. However, when directly applying the baseline policy in the target domains, the difference is quite apparent. Large tracking error can be observed in the lane-following scenario. In the CA scenarios, significant lateral oscillation occurs. Furthermore, the oscillation in Fig. 4.12b even affects the high-level behavior. The vehicle fails to overtake the second vehicle due to its



(a) Curved lane following



(b) Collision avoidance



(c) Collision avoidance with speed limit

Figure 4.12: Trajectories of PN in the source domain, PN in the target domain with parameter variation, and PN-DOB in the target domain with parameter variation. We plot the results for three scenarios: CL following, CA, and CASL. In the last plot, the black vertical line indicates the location where the SL takes effect.

unstable lateral motion. Based on the table and the visualized trajectories, we conclude that the proposed PN-RC framework is more robust against the modeling gap than the baseline policy. Moreover, the designed adaptive DOB-based controller improves its performance compared to the baseline proportional controller.

4.5 Preliminary Sim-to-Real Policy Transfer

4.5.1 Experimental Setup

We validated the capability of the proposed architecture to generate onboard driving commands for real autonomous vehicles in a preliminary experiment carried out in the Richmond Field Station. The tested case is an obstacle avoidance task ($LT \oplus OB_1 \oplus OB_{re1} \oplus OB_{re2}$), with a static obstacle vehicle parked in front of the autonomous vehicle. In the task decomposition, OB_1 refers to the OB attribute corresponding to the parked vehicle, and OB_{re1} and OB_{re2} refer to the OB attribute corresponding to the left and right road boundaries. The autonomous vehicle starts with a speed of $10m/s \approx 22mph$. The autonomous vehicle

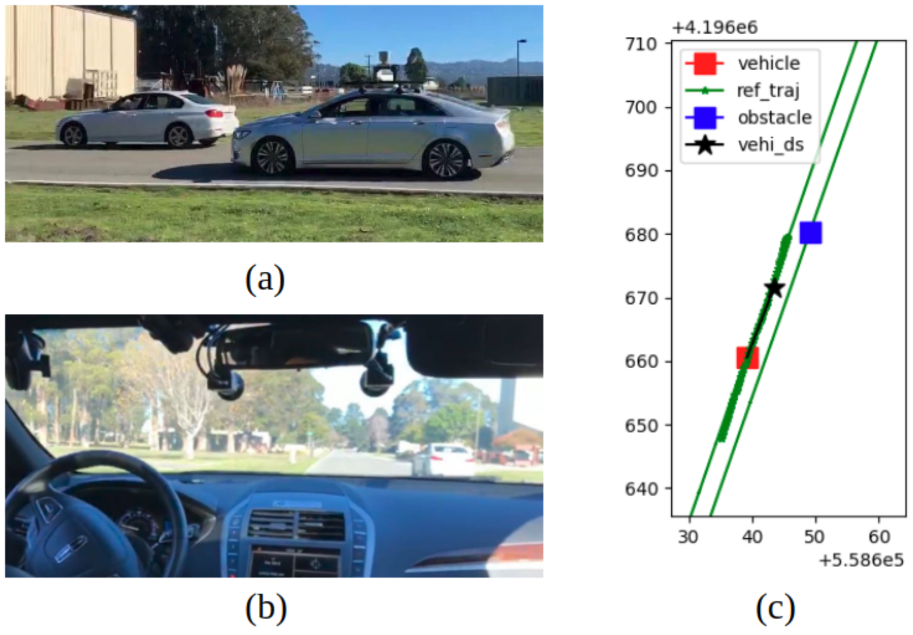


Figure 4.13: (a)(b) The experiment setting and the obstacle avoidance task; (c) The onboard tracking visualization screenshot, where the red dot is the ego vehicle, the blue dot is the obstacle vehicle, and the thick green line is the reference trajectory generated using the source domain [141].

incorporates GPS and inertial measurement units (IMUs) that can measure its states, while the obstacle vehicle states are assumed to be known. Screenshots of the experiment setting and the onboard tracking visualization are presented in Fig. 4.13.

For the motion planning part, a PAN policy with three pretrained OB modules is applied to produce the longitudinal acceleration a_0 and the lateral yaw rate $\dot{\theta}_0$. During online usage, we first employ the PAN policy to generate a reference trajectory for 80 time steps (1.6 seconds) in the simulator. For the longitudinal control, we use a PID controller to track the generated longitudinal speed profile. For the lateral control, we apply the DOB-based tracking controller to produce the steering commands to track the reference trajectory.

4.5.2 Lane Tracking and Obstacle Avoiding Performance

The online performance of the proposed architecture is stable and fast in the task, even though the task has not been trained previously. The PAN policy node in the ROS publishes the commands at 50Hz, the imaginary trajectory used to produce the steering command publishes at a frequency of 5Hz, and the DOB-based robust tracking controller produces the steering commands at 50Hz. We performed the experiment 10 times with different starting positions and achieved a 10/10 success rate. Fig. 4.14 shows a typical experimental

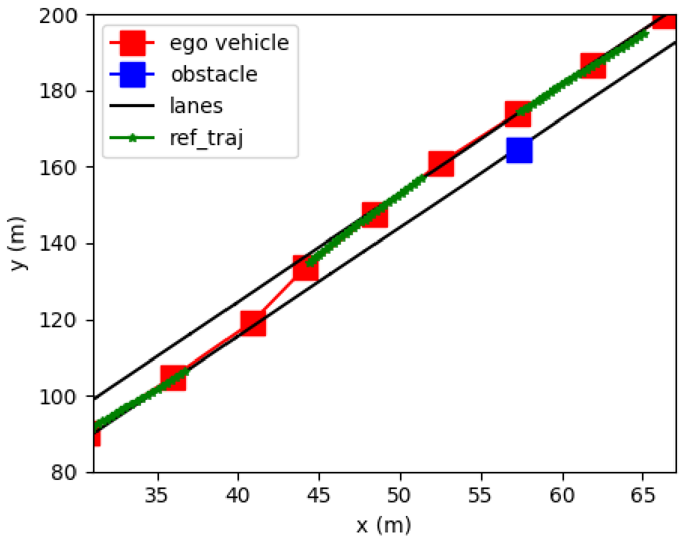


Figure 4.14: Behavior of the autonomous vehicle in the $LT \oplus OB_1 \oplus OB_{re1} \oplus OB_{re2}$ task in one of the real vehicle experiments. The red squares indicate the real trajectory of the autonomous vehicle, the blue block is the parked obstacle vehicle, and the green lines are a few reference trajectories generated by the PAN policy in the imaginary simulation [141].

trajectory. The experiment shows that the proposed PN-RC framework is promising in zero-shot transferring policy networks to solve driving tasks in real-world environments.

4.6 Chapter Summary

In this chapter, we focus on the driving policy transfer problem between domains where discrepancies in vehicle dynamics exist. We propose the PN-RC framework, which treats state trajectories as invariant representations that are transferable across different domains. Concretely, the policy network trained in the source domain is used to generate a reference trajectory through simulation in the source domain. Subsequently, a robust tracking controller is used to track the reference trajectory in the target domain. The robust controller is designed based on our knowledge of the vehicle dynamics in the target domain so that disturbances caused by the modeling gap can be rejected. In particular, we present an implementation of the generic framework consisting of a PAN policy network and an adaptive DOB-based robust tracking controller. Our experiments in simulated environments and on a real testing vehicle show that the proposed PN-RC framework can achieve consistent performance in target domains with modeling discrepancies and external disturbances. In contrast, the baseline end-to-end policy network is sensitive to dynamic variations and fails to complete the task when a modeling gap exists. Future efforts will extend the current method for more complicated and general driving tasks.

We close this chapter with a brief discussion of the importance of reference trajectories as the interpretable intermediate representations of a policy network, even outside the context of policy transfer. While it is sufficient to output control commands to control the vehicle directly, it is difficult to understand the intended behavior of the policy network from the control commands at a single time step. In contrast, a trajectory consisting of target states over the preview horizon contains richer semantic information, making it easier to interpret the underlying driving intention. Therefore, we should adopt this hierarchical planning and control architecture, even just from the perspective of interpretability.

Part II

Model Diagnosis with Post hoc Explanation

Chapter 5

Diagnosing Social Posterior Collapse with Sparse Graph Attention

5.1 Introduction

Accurate modeling of the social interaction among road participants is a prerequisite for accurate and robust trajectory prediction in interactive traffic scenarios. Generative latent variable models are popular modeling options due to their ability to generate diverse and naturalistic behavior [124, 46, 68, 109]. We focus on one category of generative models, VAE [66], which has been widely used in multi-agent behavior modeling and trajectory prediction [124, 109, 56, 86, 74, 148]. It is desirable because it learns a low-dimensional representation of the original high-dimensional data. However, VAEs do not necessarily learn a good representation of the data [18]. For instance, prior works on sequence modeling have found that the model tends to ignore the latent variables if the decoder is overly powerful [13, 32, 114] (e.g., an autoregressive decoder). This makes us wonder whether a VAE-based model can always learn a good representation of a multi-agent interacting system. However, this is a general question, as researchers may look for different properties of the latent space, for example, interpretability [54, 129] and multi-modality [109, 56]. In this chapter, we focus on a fundamental aspect of this general question: *Does the latent space always properly model interaction?* Formally, given a latent variable model of an interacting system, where a latent variable governs each agent's behavior, we seek to understand whether the VAE learns to encode the social context into the latent variables.

This is a concern because VAE handles two distinct tasks in training and testing. In the training stage, it learns to reconstruct a datum instead of generating one. For multi-agent interaction modeling, the sample for reconstruction is a set of trajectories of all the agents. Ideally, the model should learn to model the interaction among agents and jointly reconstruct the trajectories. However, there is no mechanism to prevent the model from separately reconstructing the trajectories. In fact, it could even be more efficient in the early stage of training when the model has not learned an informative embedding of the social context. We then end up with a model that models each agent's behavior separately without social context, which might suffer from over-estimated variance and large prediction error. More importantly, since the joint behavior of the agents is a consequence of their interactions, ignoring the causes may lead to poor generalization ability [129, 47]. We find that a typical formulation of VAE for multi-agent interaction is indeed prone to ignoring historical social context (i.e., interactions over the observed time horizon). We refer to this phenomenon as *social posterior collapse*. This issue has never been discussed in the literature. Considering the consequences of ignoring social context, we believe it is necessary to study such a crucial and fundamental issue.

In this chapter, we first abstract the VAE formulation from a wide range of existing studies [124, 109, 56, 74]. Then, we analyze social posterior collapse under this formulation and propose several measures to alleviate the issue. Next, we study the issue in real-world settings with a realization of the abstract formulation we design. To monitor and analyze social posterior collapse, we develop an *explainable* model with intermediate output indicating social attention. In particular, we propose a novel sparse-GAMP layer and develop the social-CVAE model, which incorporates a sparse-GAMP encoder for social context aggregation. With the help of sparse-GAMP, we can directly identify the surrounding agents ignored by the model. Consequently, we can easily detect social posterior collapse if the attention map indicates that the model ignores all its surrounding agents. Our experiments show that social posterior collapse occurs in real-world prediction tasks and that the proposed measures can effectively alleviate this issue. We also evaluate how social posterior collapse affects the model performance on these tasks. The results suggest that the model without social posterior collapse can attain better generalization performance if the historical social context is informative for prediction.

The rest of the chapter is organized as follows. In Sec. 5.2, we first formulate the VAE interaction model studied. Subsequently, we define and analyze the social posterior collapse phenomenon, which leads to several potential measures to alleviate this issue. In Sec. 5.3, we present the proposed sparse-GAMP layer, which is powered by the α-entmax [98] function. In Sec. 5.4, we introduce the architecture of social-CVAE, which incorporates sparse-GAMP for social posterior collapse analysis and proposed modules to mitigate social posterior collapse. In Sec. 5.5, we report our experiments on real-world trajectory prediction benchmarks. In Sec. 5.6, we further analyze our findings with the help of detailed ablation studies. In Sec. 5.7, we provide more details about the experimental settings. In Sec. 5.8, we conclude the chapter by discussing the connections of our work with the existing literature as well as the current limitations of our work.

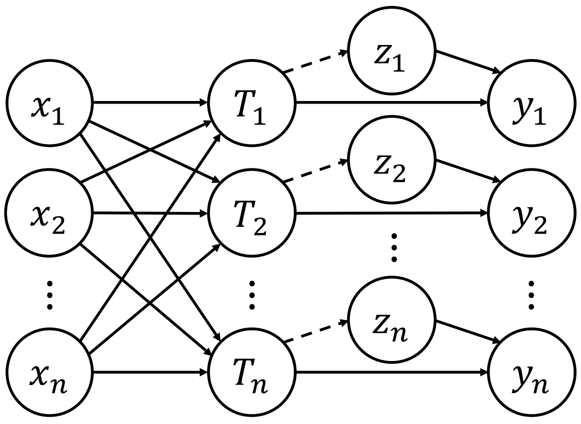


Figure 5.1: A latent variable model for interaction modeling. The dash edges from T_i to z_i apply only in the CVAE setting.

5.2 Social Posterior Collapse in Variational Autoencoder Interaction Models

5.2.1 A Latent Variable Model for Interaction Modeling

Given an interacting system with n agents, an interaction-aware trajectory prediction model takes all the agents' observed historical trajectories, denoted by $\{\mathbf{x}_i\}_{i=1}^n$, as input and predicts the future trajectories of all the agents or a subset of enquired agents. We denote the collection of future trajectories as $\{\mathbf{y}_i\}_{i=1}^n$. In this chapter, we focus on the latent variable model illustrated in Fig. 5.1, which is abstracted from existing literature on VAEs for multi-agent behavior modeling [124, 109, 56, 74]. We model interaction by introducing a set of variables $\{\mathbf{T}_i\}_{i=1}^n$, which aggregates each agent's state and its observation of other agents. Formally, each $\mathbf{T}_i$ is modeled as a deterministic function of $\{\mathbf{x}_i\}_{i=1}^n$, i.e., $\mathbf{T}_i = f_i\left(\{\mathbf{x}_i\}_{i=1}^n\right)$. Afterward, the agents make decisions based on the aggregated information over the predicted horizon. Latent variables $\{\mathbf{z}_i\}_{i=1}^n$ are introduced to model the inherent uncertainty in each agent's behavior. It should be noted that interaction over the predicted horizon is not modeled explicitly in this formulation. Although it can be achieved by exchanging information between agents recurrently (e.g., social pooling in [1]), it is a common practice to avoid explicit modeling of future interaction due to computational and memory costs [68, 74, 61].

We train the model as a VAE, where an encoder $q\left(\mathbf{z}|\mathbf{x},\mathbf{y}\right)$ is introduced to approximate the posterior distribution $p\left(\mathbf{z}|\mathbf{x},\mathbf{y}\right)$ for efficient sampling in the training stage[1]. The performance of variational inference is optimized if the KL divergence between the posteriors, i.e., $D_{KL}\left[q\left(\mathbf{z}|\mathbf{x},\mathbf{y}\right)\|p\left(\mathbf{z}|\mathbf{x},\mathbf{y}\right)\right]$, is small [18]. To derive a better approximation, we can in-

[1]The vectors $\mathbf{x}, \mathbf{y}$, and $\mathbf{z}$ collect the corresponding variables for all n agents.

corporate inductive bias based on the characteristics of the true posterior into the encoder function. We introduce the following proposition, which guides our model design:

Proposition 1. *For any $i = 1, 2, ..., n$ and $j = 1, 2, ..., n$, 1) If $j \neq i$, $\mathbf{z}_i$ and $\mathbf{y}_j$ are conditionally independent given $\mathbf{x}$ and $\mathbf{y}_i$; 2) If $j \neq i$, $\mathbf{z}_i$ and $\mathbf{z}_j$ are conditionally independent given $\mathbf{x}$; 3) $\mathbf{z}_i$ and $\mathbf{x}_j$ are conditionally independent given $\mathbf{T}_i$.*

Proof. Because the graph in Fig. 5.1 is a directed acyclic graph (DAG), we can apply the d-separation criterion [94] to analyze the conditional independence. For each pair of i and j with $i \neq j$, the node $\mathbf{z}_i$ is connected to $\mathbf{y}_j$ through $\mathbf{T}_i$ and $\mathbf{T}_j$. Further, any path between $\mathbf{T}_i$ and $\mathbf{T}_j$ has a triple $\mathbf{T}_i \leftarrow \mathbf{x}_k \rightarrow \mathbf{T}_j$ for some $k = 1, 2, ..., n$, which is inactive after conditioning on $\mathbf{x}$. Therefore, we can apply the d-separation criterion to conclude that $(\mathbf{z}_i \perp\!\!\!\perp \mathbf{y}_j) \mid \mathbf{x}, \mathbf{y}_i$. The same inactive triples also imply that $(\mathbf{z}_i \perp\!\!\!\perp \mathbf{z}_j) \mid \mathbf{x}$. For the third statement, any path between $\mathbf{z}_i$ and $\mathbf{x}_j$ has a inactive path $\mathbf{x}_j \rightarrow \mathbf{T}_i \rightarrow \mathbf{y}_i$. Therefore, the d-separation criterion implies that $(\mathbf{z}_i \perp\!\!\!\perp \mathbf{x}_j) \mid \mathbf{T}_i$. □

Following the proposition, we decompose the posterior distribution as $\prod_{i=1}^{n} p(\mathbf{z}_i|\mathbf{T}_i, \mathbf{y}_i)$. The decomposition suggests two insights. First, the encoder does not need to aggregate future context information when inferring the posterior distribution of $\mathbf{z}_i$ for each agent. Second, the historical context variable $\mathbf{T}_i$ is all that is required in terms of the historical information of the agent i. The encoder and decoder can share the same function to encode historical information.

5.2.2 Social Posterior Collapse

We then design a VAE model reflecting the characteristics of the true posterior discovered in Sec. 5.2.1. To simplify the problem, we further make the assumption of homogeneous agents. The model has three basic building blocks: 1) A function modeling the historical context, i.e., $\mathbf{T}_i = f_\theta(\mathbf{x}_i, \mathbf{x})$; 2) A function decoding the distribution of the future trajectory $\mathbf{y}_i$ given $\mathbf{T}_i$ and $\mathbf{z}_i$, i.e., $p_\phi(\mathbf{y}_i|\mathbf{T}_i, \mathbf{z}_i)$; 3) A function approximating the posterior of $\mathbf{z}_i$ conditioned on $\mathbf{T}_i$ and $\mathbf{y}_i$, i.e., $q_\psi(\mathbf{z}_i|\mathbf{T}_i, \mathbf{y}_i)$. They build up the encoder and decoder of the VAE model as follows:

$$q_{\theta,\psi}(\mathbf{z}|\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{n} q_\psi(\mathbf{z}_i|f_\theta(\mathbf{x}_i, \mathbf{x}_{-i}), \mathbf{y}_i), \qquad p_{\theta,\phi}(\mathbf{y}|\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{n} p_\phi(\mathbf{y}_i|f_\theta(\mathbf{x}_i, \mathbf{x}_{-i}), \mathbf{z}_i). \tag{5.1}$$

We consider a continuous latent space and model q_ψ and p_ϕ as diagonal Gaussian distributions. The model is trained by maximizing the ELBO:

$$\max_{\theta,\psi,\phi} \mathbb{E}_{\mathbf{x},\mathbf{y}\sim D}\left[\mathbb{E}_{\mathbf{z}\sim q_{\theta,\psi}(\mathbf{z}|\mathbf{x},\mathbf{y})}\left[\log p_{\theta,\phi}(\mathbf{y}|\mathbf{x}, \mathbf{z})\right] - \beta D_{KL}\left[q_{\theta,\psi}(\mathbf{z}|\mathbf{x}, \mathbf{y})\,\|p(\mathbf{z})\right]\right]. \tag{5.2}$$

However, we find a critical issue in this naive formulation during experiments, which is the social posterior collapse phenomenon mentioned previously. With the help of the sparse graph attention mechanism introduced in Sec. 5.3, we find that the model is prone to ignoring historical social context when reconstructing the future trajectory of one agent. Equivalently, the latent variable model collapses to one with all the variables of other agents marginalized out.

We believe the reason for this is similar to, but different from, a well-known phenomenon in VAE training—posterior collapse [18]. Due to the KL regularization term in ELBO, the variational posterior distribution is likely to collapse toward the prior. This is particularly likely to occur in the early stage of training, when the latent space is still uninformative [33]. In our case, the posterior of $\mathbf{z}_i$ collapses into $q_{\theta,\psi}(\mathbf{z}_i|\mathbf{x}_i, \mathbf{y}_i)$. It does minimize the KL divergence: if both $q_{\theta,\psi}(\mathbf{z}_i|\mathbf{x}_i, \mathbf{y}_i)$ and $q_{\theta,\psi}(\mathbf{z}_i|\mathbf{x}, \mathbf{y}_i)$ exactly approximate the true posteriors, then conditioning on more context information increases the expected value of KL regularization:

$$\mathbb{E}_D \left[D_{KL} \left[p\left(\mathbf{z}_i|\mathbf{x}, \mathbf{y}_i\right) \| p(\mathbf{z}_i)\right]\right] = I(\mathbf{z}_i; \mathbf{x}, \mathbf{y}_i) \geqslant I(\mathbf{z}_i; \mathbf{x}_i, \mathbf{y}_i) = \mathbb{E}_D \left[D_{KL} \left[p\left(\mathbf{z}_i|\mathbf{x}_i, \mathbf{y}_i\right) \| p(\mathbf{z}_i)\right]\right].$$

However, we argue that an additional factor contributes to the social posterior collapse problem, which makes it a unique phenomenon for interaction modeling. In the training stage, the goal is to reconstruct the future trajectories $\mathbf{y}$. The trajectory itself contains all the information needed for reconstruction. The history of the agent i provides complementary information, such as the current state and static environmental information. There is no explicit regulation in the current framework to prevent the model from extracting information solely from $\mathbf{x}_i$ and $\mathbf{y}_i$. In fact, the coding scheme is more efficient when the model has not learned an informative representation of interaction. Techniques such as KL annealing [114, 33] rely on various β scheduling schemes to prevent KL vanishing, which has been shown to be effective in mitigating the posterior collapse problem. However, reducing β could merely privilege the model to gather more information from $\mathbf{y}_i$, which is consistent with the reconstruction objective. Therefore, we need to explore alternative solutions to tackle the social posterior collapse problem deliberately.

We start by changing the model into a conditional generative one [117, 57]. Additional edges $\{\mathbf{T}_i \to \mathbf{z}_i\}_{i=1}^n$ are added to the original graph, which are annotated with dashed lines in Fig. 5.1. We follow the practice in [117] and formulate the model as a CVAE. It is straightforward to verify that the conclusion of Proposition 5.2.1 still applies. We model the encoder and decoder as in Eqn. (5.1). The CVAE framework introduces an additional module—a function approximating the conditional prior $p_\eta\left(\mathbf{z}_i|\mathbf{T}_i\right)$, which becomes $p_{\theta,\eta}\left(\mathbf{z}_i|\mathbf{x}\right)$ after incorporating $f_\theta(\mathbf{x})$. The objective then becomes:

$$\mathcal{L}(\theta, \psi, \phi, \eta) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim D} \left[\mathbb{E}_{\mathbf{z}\sim q_{\theta,\psi}(\mathbf{z}|\mathbf{x},\mathbf{y})} \left[\log p_{\theta,\phi}\left(\mathbf{y}|\mathbf{x}, \mathbf{z}\right)\right] - \beta D_{KL}\left[q_{\theta,\psi}\left(\mathbf{z}|\mathbf{x}, \mathbf{y}\right) \| p_{\theta,\eta}(\mathbf{z}|\mathbf{x})\right]\right].$$

Compared to Eqn. (5.2), we no longer penalize the encoder for aggregating context information but only restrict the information encoded from future trajectories. Therefore, the encoder does not need to ignore context information to fulfill the information bottleneck. However, the model still lacks a mechanism for encouraging context information encoding deliberately. To achieve the goal, we propose incorporating an auxiliary prediction task into the training scheme. Concretely, we introduce another module $\mathbf{y}_i = g_\zeta\left(\mathbf{T}_i, \mathbf{z}_i\right)$. Composing g_ζ

with f_θ gives us another decoder $\mathbf{y} = h_{\theta,\zeta}(\mathbf{x}, \mathbf{z})$. The difference is that the latent variables are always sampled from $p_\eta(\mathbf{z}_i|\mathbf{T}_i)$. The auxiliary task is training this trajectory decoder, which shares the same context encoder and conditional prior with the CVAE. Because the auxiliary model does not have access to the ground-truth future trajectory, it needs to utilize context information for accurate prediction. Consequently, it encourages the model to encode context information into $\mathbf{T}$ and $\mathbf{z}$. We use mean squared error (MSE) loss as the objective function. The overall objective is a weighted sum of the two objective functions. In practice, we follow the common practice of fixing the variance of $p_{\theta,\phi}(\mathbf{y}|\mathbf{x}, \mathbf{z})$. Consequently, the reconstruction loss is equivalent to an MSE loss, and the overall objective becomes maximizing the following objective function:

$$\begin{aligned}\hat{\mathcal{L}}(\theta, \psi, \phi, \eta, \zeta) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim D}\left[\mathbb{E}_{\mathbf{z}\sim q_{\theta,\psi}(\mathbf{z}|\mathbf{x},\mathbf{y})}\left\|h_{\theta,\phi}(\mathbf{x}, \mathbf{z}) - \mathbf{y}\right\|^2 - \beta D_{KL}\left[q_{\theta,\psi}(\mathbf{z}|\mathbf{x}, \mathbf{y})\,\|p_{\theta,\eta}(\mathbf{z}|\mathbf{x})\right]\right] \\ - \alpha\mathbb{E}_{\mathbf{x},\mathbf{y}\sim D,\mathbf{z}\sim p_{\theta,\eta}(\mathbf{z}|\mathbf{x})}\left\|h_{\theta,\zeta}(\mathbf{x}, \mathbf{z}) - \mathbf{y}\right\|^2.\end{aligned} \tag{5.3}$$

The training scheme looks similar to the one in [117], in which a Gaussian stochastic neural network (GSNN) is trained together with the CVAE model. However, ours is different from theirs in some major aspects. The GSNN model shares the same decoder with the CVAE model. The primary motivation of incorporating another learning task is to optimize the generation procedure during training directly. In contrast, our auxiliary prediction model has a separate trajectory decoder. This is because we do not want the auxiliary task to interfere with the decoding procedure of the CVAE model. We find that sharing the decoder leads to less diversity in trajectory generation, which is not desirable.

5.3 Sparse Graph Attention Message-Passing Layer

Before introducing the specific model we developed for real-world prediction tasks, we present a novel sparse-GAMP layer, which helps us detect and analyze the social posterior collapse phenomenon.

5.3.1 Sparse Graph Attention Mechanism

Our sparse-GAMP layer incorporates α-entmax [98] as the graph attention mechanism. α-entmax is a sparse transformation that unifies softmax and sparsemax [87]. Sparse activation functions have received growing attention recently because they can induce sparse and interpretable outputs. In the context of VAEs, they have been used to sparsify discrete latent space for efficient marginalization [23] and tractable multi-modal sampling [55]. In our case, we mainly use α-entmax to induce a sparse and interpretable attention map within the encoder for diagnosing social posterior collapse.

We are particularly interested in the 1.5-entmax variant, which is smooth and can be exactly computed. It is also easy to implement on GPUs using existing libraries (e.g., PyTorch [92]). Concretely, the 1.5-entmax function maps a d-dimensional input $\mathbf{s} \in \mathbb{R}^d$ into

$\mathbf{p} \in \Delta^d = \left\{\mathbb{R}^d : \mathbf{p} \geqslant 0, \|\mathbf{p}\|_1 = 1\right\}$ as $\mathbf{p} = [\mathbf{s}/2 - \tau\mathbf{1}]_+^2$, where τ is a unique threshold value computed using $\mathbf{s}$. Based on the theoretical results in [98], we derive an insightful proposition that makes 1.5-entmax a desirable option in our framework.

Proposition 2. *Let $s_{[d]} \leqslant \cdots \leqslant s_{[1]}$ denote the sorted coordinates of $\mathbf{s}$. Define the top-ρ mean, unnormalized variance, and induced threshold for $\rho \in \{1, ..., d\}$ as*

$$M_{\mathbf{s}}(\rho) = \frac{1}{\rho}\sum_{j=1}^{\rho} s_{[j]},\ S_{\mathbf{s}}(\rho) = \sum_{j=1}^{\rho}\left(s_{[j]} - M_{\mathbf{s}}(\rho)\right)^2,\ \tau_{\mathbf{s}}(\rho) = \begin{cases} M_{\mathbf{s}}(\rho) - \sqrt{\frac{1-S_{\mathbf{s}}(\rho)}{\rho}}, & S_{\mathbf{s}}(\rho) \leqslant 1, \\ +\infty, & \textit{otherwise.}\end{cases}$$

Let $\mathbf{s}' \in \mathbb{R}^{d+1}$ satisfy $s'_i = s_i$ for $i \leqslant d$, and define $\mathbf{p} = 1.5\text{-}entmax(\mathbf{s})$ and $\mathbf{p}' = 1.5\text{-}entmax(\mathbf{s}')$.

Then we have the following: 1) If $\frac{s'_{d+1}}{2} \leqslant \frac{s_{[d]}}{2} - 1$, then $p'_i = p_i$ for $i = 1, ..., d$ and $p'_{d+1} = 0$; 2) If $p_i > 0$ for $i = 1, .., d$, then $p'_i = p_i$ for $i = 1, 2, ..., d$ and $p'_{d+1} = 0$ iff $s'_{d+1} \leqslant 2\tau_{\mathbf{s}/2}(d)$.

Proof. Proposition 3 in [98] suggests that the threshold value is equal to $\tau_{\mathbf{s}/2}(\rho^*)$ with any ρ^* satisfying $\tau_{\mathbf{s}/2}(\rho^*) \in [\frac{s_{[\rho^*+1]}}{2}, \frac{s_{[\rho^*]}}{2}]$. If d satisfies the condition, then $\tau_{\mathbf{s}/2}(d) \in (-\infty, \frac{s_{[d]}}{2}]$, which is clearly finite. Therefore, $\tau_{\mathbf{s}/2}(d) = M_{\mathbf{s}/2}(d) - \sqrt{\frac{1-S_{\mathbf{s}/2}(d)}{d}} \geqslant \frac{s_{[d]}}{2} - 1$ by definition. Given $\frac{s'_{d+1}}{2} \leqslant \frac{s_{[d]}}{2} - 1$, we have $\tau_{\mathbf{s}'/2}(d) = \tau_{\mathbf{s}/2}(d) \in [\frac{s'_{d+1}}{2}, \frac{s'_{[d]}}{2}]$. Therefore, $\tau_{\mathbf{s}/2}(d)$ is still the threshold value. If d is not a valid ρ^*, then there exists $\rho^* \in \{1, ..., d-1\}$ defining the threshold value. Because $\frac{s'_{d+1}}{2} < \frac{s_{[d]}}{2}$, augmenting the input vector does not affect the threshold value. In both cases, the threshold value is unchanged, which leads to the first statement of the proposition.

Now we prove the second statement of the proposition. Because $p_i > 0$ for $i = 1, .., d$, the threshold value is smaller than $\frac{s_{[d]}}{2}$, which makes d the only possible ρ^*. If $s'_{d+1} \leqslant 2\tau_{\mathbf{s}/2}(d)$, then $\tau_{\mathbf{s}/2}(d)$ defines the threshold value for $\mathbf{s}'$. Therefore, $p'_i = p_i$ for $i = 1, 2, ..., d$ and $p'_{d+1} = 0$. Instead, if we are given $p'_i = p_i$ for $i = 1, 2, ..., d$ and $p'_{d+1} = 0$, the threshold satisfies $\tau_{\mathbf{s}'/2}(\rho^*) \in \left[\frac{s'_{d+1}}{2}, \frac{s_{[d]}}{2}\right]$. Therefore, d is a valid threshold index for both $\mathbf{s}$ and $\mathbf{s}'$, and $s'_{d+1} \leqslant 2\tau_{\mathbf{s}'/2}(d) = 2\tau_{\mathbf{s}/2}(d)$. □

The first statement of the proposition provides a sufficient condition for augmenting an input vector without affecting its original attention values. It is useful when applying 1.5-entmax to graph attention. Unlike typical neural network models, GNNs operate on graphs whose sizes vary over different samples. Meanwhile, nodes within the same graph may have different numbers of incoming edges. Therefore, the 1.5-entmax function has inputs of varying dimensions even within the same batch of training samples, making it inefficient to compute using available primitives. The proposition suggests a simple solution to this problem. Given $\{\mathbf{s}_j\}_{j=1}^m$ with $\mathbf{s}_j \in \mathbb{R}^{d_j}$, we can compute a dummy value as $\min_{j\in\{1,...,m\}, i\in\{1,...,d_j\}} s_{j,i} - 2$. We can augment the input vectors with this dummy value to transform them into a matrix in $\mathbb{R}^{m\times d^*}$, where d^* is the largest value of d_j. The dummy elements will not affect the attention computation, and existing primitives based on matrix computation can be used directly.

The second statement implies that the activated coordinates determine a unique threshold value for augmented coordinates. This property is beneficial when modeling interactions with a large number of agents (e.g., dense traffic scenes). This ensures that the effects of interacting agents will not be diluted by the irrelevant agents, which could potentially improve the robustness of the model [39]. Here, we mainly utilize the interpretability of the sparse graph attention, but we will investigate its application in generalization in future work.

5.3.2 Sparse-GAMP

To obtain the sparse-GAMP layer, we combine the sparse graph attention with a message-passing GNN [37, 67]. Given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertices $v \in \mathcal{V}$ and edges $e = (v, v') \in \mathcal{E}$, we define the sparse-GAMP layer as a composition of a node-to-edge message-passing step $v \to e$ and an edge-to-node message-passing step $e \to v$:

$$\begin{aligned} v \to e: \ & \mathbf{h}_{(i,j)} = f_e\left(\left[\mathbf{h}_i, \mathbf{h}_j, \mathbf{u}_{(i,j)}\right]\right), \\ e \to v: \ & \hat{\mathbf{h}}_j = \sum_{i \in \mathcal{N}_j} w_{(i,j)} \mathbf{h}_{(i,j)}, \quad \text{where } \mathbf{w}_j = \mathcal{G}\text{-entmax}\left(\left\{\mathbf{h}_{(i,j)}\right\}_{i \in \mathcal{N}_j}\right), \end{aligned} \tag{5.4}$$

In our prediction model, we use this sparse-GAMP layer to model the function for historical social context encoding, i.e., $\mathbf{T}_i = f_\theta\left(\mathbf{x}_i, \mathbf{x}\right)$.

5.4 Social-CVAE

In this section, we design a realization of the abstract framework studied in Sec. 5.2 for real-world trajectory prediction tasks. We choose to design the model with GNNs, which enable a flexible graph representation of data. Several GNN-based approaches have achieved state-of-the-art performance in trajectory prediction tasks [151, 75, 149]. The resulting model is depicted in Fig. 5.2, which we refer to as social-CVAE. During training, we first encode the historical and future trajectories of all the agents using a GRU [21] network. For vehicle trajectory prediction tasks, we incorporate the map information in a manner similar to [36]. Different from [36], where road boundaries are modeled as nodes in the scene graph, we adopt the representation in [9], where the map is denoted as a graph consisting of lanelet nodes, i.e., drivable road segments. Each lanelet is represented by its left and right boundaries, consisting of two sequences of points. We use another GRU network to encode them. Although not utilized in this work, the lanelet representation also allows us to combine routing information in a natural manner, as in [149].

To encode the historical context information, we construct a graph using the embeddings of historical trajectories and lanelets if available. Each pair of agent nodes has a bidirectional edge connecting each other. For each lanelet node, we add edges connecting it to all the agent nodes. If the map information is not available, we add self-edges for all the agents to enable direct self-encoding channels. One sparse-GAMP layer is applied to encode historical context for all the agents. If social posterior collapse occurs, the agent-to-agent edges, except

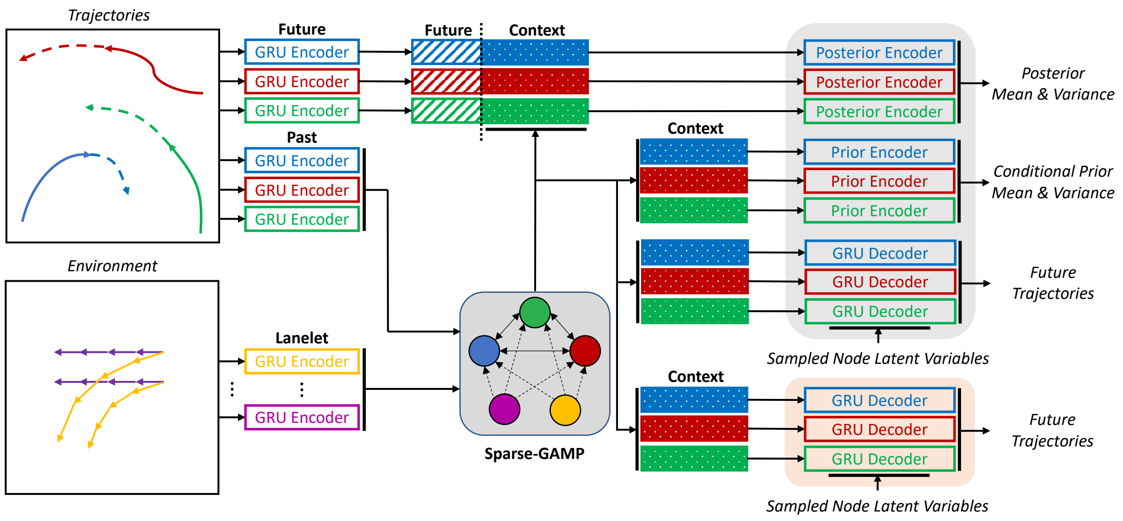


Figure 5.2: Social-CVAE architecture. We adopt a graph representation and use sparse-GAMP to encode the context. The modules within the orange box are the auxiliary decoders.

for self-edges, are more likely to receive zero attention weights due to the sparse attention. Therefore, we can use the percentage of unattended agent-to-agent edges as a metric to detect social posterior collapse. This is a more objective metric than the quantitative magnitude of attention weights. We can assert that an agent node does not contribute to the output if its attention is strictly zero. However, we need to be more careful when comparing the importance of two agents based on non-zero attention weights [59, 139]. For the same reason, we do not consider techniques such as multiple message-passing layers [36, 75, 149] or separating self-edges from aggregation [74]. Although they could potentially boost the performance, it may be inconclusive to analyze the model based on the attention map. After computing the historical context, we use an MLP to model the conditional Gaussian prior $p_\eta(\mathbf{z}_i|\mathbf{T}_i)$. To model the variational posterior, we concatenate the future trajectory embedding with each agent's historical context and use another MLP to output the posterior mean and variance. For the decoder, we use another GRU to decode the future trajectory for each agent separately. The auxiliary decoder shares the same structure but always samples the latent variables from the conditional prior.

5.5 Experiments

In this section, we report the experimental results for two trajectory prediction tasks. The main purpose is not achieving state-of-the-art performance but to study the social posterior collapse problem. We compare social-CVAE with two variants: 1) A model without the auxiliary task; 2) A model without the auxiliary task or the conditional prior. They correspond to the vanilla CVAE and VAE formulations discussed in Sec. 5.2. We are curious about whether the proposed measures can alleviate social posterior collapse. We will briefly introduce the experiment settings and present the main results. More details on experiment settings as well as detailed ablation studies can be found in Sec. 5.6 and Sec. 5.7.

5.5.1 Vehicle Trajectory Prediction

The first task is the vehicle trajectory prediction problem, where we are asked to predict the future trajectory of one target vehicle given its historical trajectories and those of other surrounding vehicles. We train the models on two datasets: the INTERACTION dataset [150] and Argoverse Motion Forecasting dataset [15]. To evaluate the prediction performance, we use the standard minimum average displacement error (*min*ADE) and minimum final displacement error (*min*FDE) over K sampled trajectories as metrics. We follow [15] and define *min*ADE as ADE of the trajectory with minimum FDE. Additionally, we define a unique metric, agent ratio (AR), to study social posterior collapse:

$$\mathrm{AR} = \frac{\sum_{i=1, i\neq j}^{n} \mathbf{1}(\omega_{(i,j)} \neq 0)}{n-1},$$

where the agent j refers to the predicted target vehicle, and $w_{(i,j)}$ is the attention weight assigned to the edge from the agent i to the target vehicle. It is equal to the percentage of surrounding vehicles that receive non-zero attention. A value close to zero implies that the model ignores the majority of the surrounding vehicles, which is a sign of the occurrence of social posterior collapse.

INTERACTION Dataset

The INTERACTION dataset provides three categories of driving scenarios: intersection (IS), roundabout (RA), and highway merging (HM). For each experiment, we ran five trials to account for the randomness in initialization. We then evaluated them on the validation sets and computed the mean and standard deviation of the evaluated metrics over all the trials. The best models were selected for testing on the regular (R) and generalization (G) tracks of the INTERPRET Challenge [2]. The results are summarized in Table 5.1. The CVAE variants have AR values similar to the VAE variants on IS and HM. This is consistent with our argument that merely changing the model to a conditional one is insufficient.

In contrast, our social-CVAE model consistently attains high AR values. However, the CVAE variant achieves similar prediction performance as ours on the validation sets of IS and RA, even if the CVAE variant suffers from the social posterior collapse problem. This is because the driving behavior within ISs and RAs depends highly on location. When the map is less informative (e.g., validation set in HM) or a novel scenario is encountered (e.g., generalization track), our social-CVAE model, which does not have social posterior collapse,

[2]The challenge adopts a different group of evaluation metrics; please see their website for the formal definitions: `http://challenge.interaction-dataset.com/prediction-challenge/intro`

outperforms the other variants. Notably, we compare it with another instance that attains a AR value close to zero even with the auxiliary task. These test performances, especially on the generalization track, are considerably different. This shows that it is mainly the historical social context that improves the prediction performance. While the auxiliary task also contributes, the improvement is not significant, especially in novel scenes, unless it prevents social posterior collapse.

Argoverse Dataset

Similar to the INTERACTION dataset, we trained five models with random initialization for each case and evaluated them on the validation sets. The best social-CVAE instance was selected for submission to the Argoverse Motion Forecasting Challenge. Although achieving state-of-the-art performance is not our objective, we still report the testing result in Sec. 5.5.4 and compare it with the other models on the leaderboard in order to provide the audience with a complete picture of the model. Here, we mainly focus on the validation results in Table 5.2. The conclusion is consistent. The difference is that the social-CVAE model outperforms the others even on the validation set. This is because the validation set of the Argoverse dataset is collected in different areas of the cities, which is more analogous to the generalization track of the INTERPRET Challenge.

5.5.2 Pedestrian Trajectory Prediction

The second task is the pedestrian trajectory prediction problem, where we are asked to forecast the future trajectories of all the pedestrians in each scenario. We trained the models on the well-established ETH/UCY dataset, which combines two datasets, ETH [95] and UCY [70]. Following prior works [46, 108, 109, 86, 148], we adopt the leave-one-out evaluation protocol and do not use any environmental information. All the prediction metrics are computed with $K = 20$ samples. Similar to the vehicle case, we ran five trials for each experiment. The results are summarized in Table 5.3, where we report the testing results for each group and the averages for all the groups. A comparison between our model and other methods can be found in Sec. 5.5.4. In Table 5.3, we see that changing to the CVAE formulation leads to a significant boost in terms of AR, implying that the model gives more attention to surrounding agents. However, neither changing to CVAE nor the auxiliary task improves prediction performance. This is because historical social context is less informative for pedestrian trajectory prediction. The ETH/UCY dataset is collected in unconstrained environments with few objects. In addition, compared to vehicles, pedestrians have fewer physical constraints in 2D motion, resulting in shorter temporal dependency in their movement. This is consistent with the results in [148], where the authors proposed a transformer-based model that achieves the current state-of-the-art performance on ETH/UCY. The visualized attention maps in [148] show that the social context in nearby time steps is more important to their prediction model. In contrast, the historical trajectories of other agents always receive low attention weights. Therefore, even if the auxiliary

Table 5.1: INTERACTION Dataset Validation and Test Results

Scene	Model	AR(%)	K = 1		K = 6	
			*min*FDE	*min*ADE	*min*FDE	*min*ADE
IS	VAE	1.85 ± 4.02	1.32 ± 0.03	0.42 ± 0.01	0.73 ± 0.01	0.26 ± 0.01
	CVAE	0.18 ± 0.36	$\mathbf{1.27 \pm 0.02}$	$\mathbf{0.41 \pm 0.01}$	$\mathbf{0.68 \pm 0.02}$	$\mathbf{0.24 \pm 0.01}$
	Ours	$\mathbf{15.8 \pm 10.4}$	$\mathbf{1.27 \pm 0.02}$	$\mathbf{0.41 \pm 0.01}$	0.70 ± 0.02	0.25 ± 0.01
RA	VAE	0.05 ± 0.04	1.34 ± 0.02	0.42 ± 0.01	0.76 ± 0.01	0.26 ± 0.01
	CVAE	13.4 ± 14.9	1.32 ± 0.01	0.42 ± 0.01	0.73 ± 0.02	$\mathbf{0.25 \pm 0.01}$
	Ours	$\mathbf{19.5 \pm 15.4}$	$\mathbf{1.29 \pm 0.03}$	0.42 ± 0.01	$\mathbf{0.72 \pm 0.01}$	0.26 ± 0.01
HM	VAE	8.68 ± 8.36	0.87 ± 0.08	0.29 ± 0.03	0.42 ± 0.04	0.16 ± 0.01
	CVAE	5.42 ± 10.2	0.83 ± 0.03	0.28 ± 0.03	0.40 ± 0.05	0.16 ± 0.02
	Ours	$\mathbf{15.5 \pm 8.13}$	$\mathbf{0.65 \pm 0.09}$	$\mathbf{0.22 \pm 0.02}$	$\mathbf{0.32 \pm 0.04}$	$\mathbf{0.13 \pm 0.01}$

Track	Model	AR(%)	K = 6			K = 50		
			ADE	FDE	MoN	ADE	FDE	MoN
R	VAE	0.35 ± 5.37	0.5685	1.7573	0.2238	0.5712	1.7709	0.1036
	CVAE	0.07 ± 1.43	0.5323	1.6425	0.2144	0.5410	1.6725	**0.0983**
	Ours*	0.88 ± 2.49	0.5157	1.5823	0.2195	0.5158	1.5823	0.1106
	Ours	$\mathbf{24.8 \pm 20.2}$	**0.4665**	**1.4174**	**0.2011**	**0.4662**	**1.4187**	0.1050
G	VAE	0.02 ± 1.27	1.3428	3.8542	0.8193	1.3436	3.8564	0.5181
	CVAE	0.05 ± 2.20	1.4517	4.2179	0.8743	1.3824	3.9972	0.5676
	Ours*	0.37 ± 2.74	1.1615	3.3927	0.6811	1.1617	3.3939	**0.4218**
	Ours	$\mathbf{15.2 \pm 14.7}$	**0.9205**	**2.7049**	**0.6075**	**0.9186**	**2.6969**	0.4891

Ours* refers to an instance of social-CVAE that still suffers from social posterior collapse. The results are presented in the format of mean $\pm$ std. For the validation set, the mean and standard deviation are computed over multiple trials. For the test set, the mean and standard deviation of AR are computed over all the samples.

prediction task encourages our model to encode historical social context, it does not lead to either a larger AR value or better prediction performance. This suggests that the simplified latent variable model studied in this chapter might not be sufficient to model pedestrian motion. Explicit future interaction should be considered, and an autoregressive decoder, such as the one in [148], should be used to account for it.

Table 5.2: Argoverse Dataset Validation Results

Model	AR(%)	K = 1		K = 6	
		*min*FDE	*min*ADE	*min*FDE	*min*ADE
VAE	1.27 ± 1.00	4.17 ± 0.17	1.86 ± 0.07	2.33 ± 0.06	1.30 ± 0.04
CVAE	0.40 ± 0.31	3.85 ± 0.01	1.73 ± 0.01	2.09 ± 0.02	1.19 ± 0.01
Ours	$\mathbf{28.5 \pm 15.7}$	$\mathbf{3.52 \pm 0.03}$	$\mathbf{1.59 \pm 0.02}$	$\mathbf{1.98 \pm 0.02}$	$\mathbf{1.15 \pm 0.01}$

The results are presented in the format of mean ± std computed over multiple trials.

Table 5.3: ETH/UCY Dataset Leave-One-Out Testing Results

Model	ETH			HOTEL		
	AR(%)	*min*FDE	*min*ADE	AR(%)	*min*FDE	*min*ADE
VAE	74.0 ± 7.33	0.98 ± 0.07	0.63 ± 0.03	29.0 ± 28.1	0.28 ± 0.01	0.19 ± 0.01
CVAE	$\mathbf{90.9 \pm 6.38}$	$\mathbf{0.94 \pm 0.10}$	$\mathbf{0.61 \pm 0.05}$	69.8 ± 41.7	$\mathbf{0.27 \pm 0.01}$	$\mathbf{0.18 \pm 0.01}$
Ours	83.2 ± 12.6	1.08 ± 0.10	0.68 ± 0.04	$\mathbf{72.9 \pm 21.5}$	$\mathbf{0.27 \pm 0.02}$	$\mathbf{0.18 \pm 0.01}$
Model	ZARA1			ZARA2		
	AR(%)	*min*FDE	*min*ADE	AR(%)	*min*FDE	*min*ADE
VAE	40.3 ± 22.8	$\mathbf{0.38 \pm 0.01}$	$\mathbf{0.22 \pm 0.01}$	30.1 ± 17.1	$\mathbf{0.32 \pm 0.01}$	$\mathbf{0.18 \pm 0.01}$
CVAE	$\mathbf{89.4 \pm 7.65}$	0.39 ± 0.01	$\mathbf{0.22 \pm 0.01}$	$\mathbf{64.2 \pm 25.6}$	0.35 ± 0.01	0.19 ± 0.01
Ours	61.9 ± 5.71	$\mathbf{0.38 \pm 0.01}$	$\mathbf{0.22 \pm 0.01}$	58.6 ± 15.7	0.37 ± 0.02	0.20 ± 0.01
Model	UNIV			Average		
	AR(%)	*min*FDE	*min*ADE	AR(%)	*min*FDE	*min*ADE
VAE	0.06 ± 0.09	$\mathbf{0.55 \pm 0.01}$	$\mathbf{0.31 \pm 0.01}$	34.7 ± 29.4	$\mathbf{0.50 \pm 0.27}$	$\mathbf{0.30 \pm 0.17}$
CVAE	$\mathbf{41.2 \pm 25.1}$	0.63 ± 0.05	0.35 ± 0.02	$\mathbf{71.1 \pm 29.5}$	0.51 ± 0.25	0.31 ± 0.17
Ours	36.3 ± 14.4	0.63 ± 0.02	0.35 ± 0.01	62.6 ± 21.0	0.54 ± 0.29	0.33 ± 0.19

The results are presented in the format of mean ± std computed over multiple trials.

5.5.3 Visualization

In this section, we visualize the experimental results for the vehicle prediction task. Fig. 5.3 and Fig. 5.4 show several examples from the INTERACTION and Argoverse datasets, respectively. For each instance, we compare the outputs of the three model variants we study in Sec. 5.5. The target vehicles' historical trajectories and ground-truth future trajectories are denoted by blue and red dots, respectively. We sample six predicted trajectories from each model and plot them using dashed lines of different colors. We also approximate the density function of the prediction output using a kernel density estimator and visualize it with a color map. Since we are particularly interested in monitoring the social posterior collapse phenomenon, we visualize the attention map by highlighting the surrounding vehicles and lanelets that receive non-zero attention weights. In particular, if a vehicle received non-zero attention, we use the same method as with the target vehicle to annotate its historical and future trajectories. Otherwise, its trajectories are annotated with grey dots. If a lanelet node receives non-zero attention, we highlight its boundaries or centerline with orange lines. The VAE and CVAE variants ignore all the surrounding vehicles in all the visualized instances, including those close to the target vehicles. They only pay attention to the lanelet nodes, which results in insensible prediction results, for instance, colliding into preceding vehicles (e.g., the second row in Fig. 5.3 and Fig. 5.4). In contrast, the social-CVAE models assign attention weights to vehicles that might potentially interact with the target vehicles and maintain sparse attention maps in dense traffic scenes (e.g., the last two rows in Fig. 5.3).

5.5.4 Testing Results on Argoverse and ETH/UCY

In this section, we report the testing results for the Argoverse and ETH/UCY datasets and compare the results with other models in the literature. It should be noted that achieving state-of-the-art performance is not our target. The testing results are provided to give the reader a complete picture of the model. For the Argoverse dataset, we collect the results of other models from the leaderboard. Although there are other models on the leaderboard with better performance, we focus on those from published papers to obtain insights on the reasons behind the performance gap. For the ETH/UCY dataset, we collect the results from the corresponding papers.

On the Argoverse dataset, the performance of our social-CVAE model is similar to TNT [151], which adopted a similar map representation to ours, in terms of minFDE and minADE when $\mathrm{K} = 1$. This implies that we could improve the performance of our model by adopting alternative map representations, such as those proposed in LaneGCN [75] and TPCN [142]. Another observation is that, compared to the other approaches, our model has a large performance margin in the case of $\mathrm{K} = 6$. The reason for this is that sampling from a high-dimensional continuous distribution is inefficient, making it less effective for modeling the multi-modality of driving behavior. Using discrete latent space as in [56] and [109] or conditioning the latent space on goal points [86] could be better options under the CVAE framework. On the ETH/UCY dataset, the VAE variant of our model has better or comparable performance to all the other models except for Trajectron++ and AgentFormer. As mentioned in the main text, we are particularly interested in the formulation of AgentFormer. In future studies, we will incorporate a similar autoregressive decoder into our model and investigate social posterior collapse under this setting.

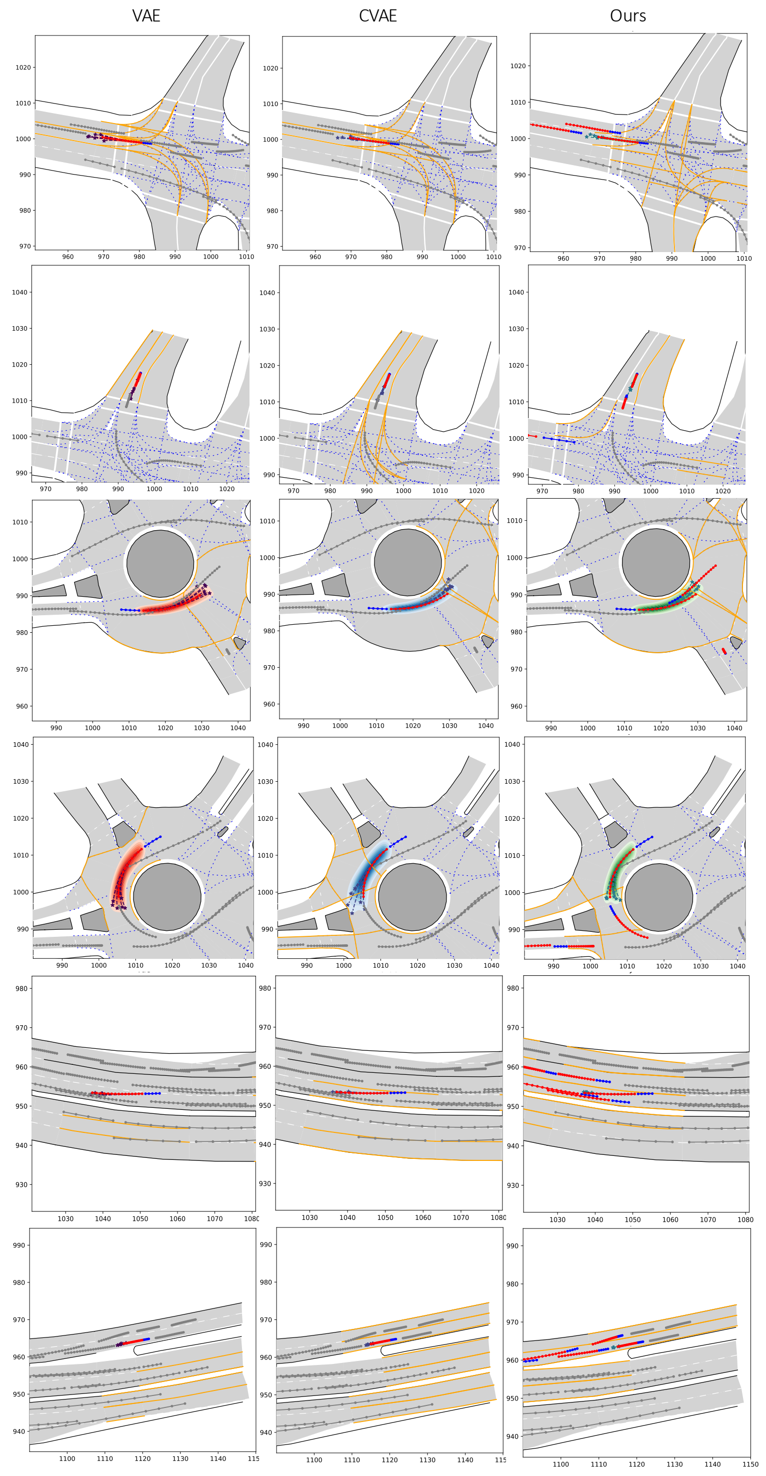


Figure 5.3: Visualizing prediction results for the Interaction dataset.

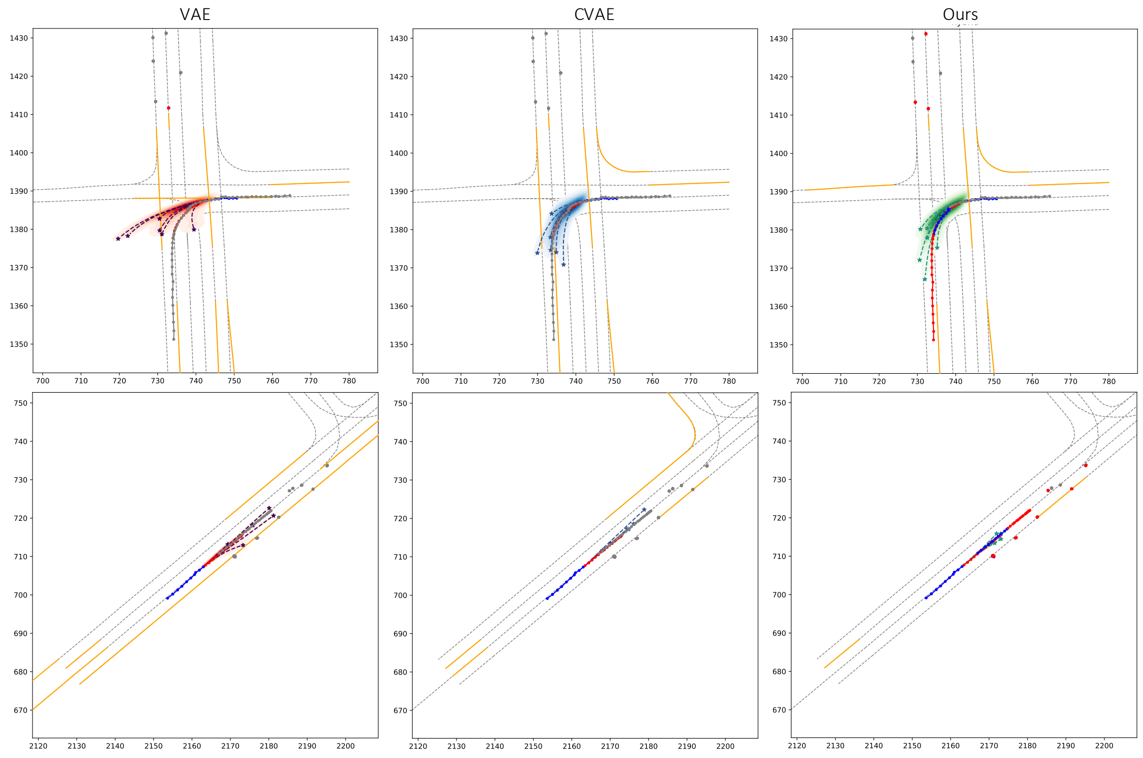


Figure 5.4: Visualizing prediction results for the Argoverse dataset.

Table 5.4: Argoverse Dataset Testing Results

Model	K = 1		K = 6	
	*min*FDE	*min*ADE	*min*FDE	*min*ADE
TNT [151]	4.9593	2.1740	1.4457	0.9097
LaneGCN [75]	3.7786	1.7060	1.3640	0.8679
LaneRCNN [149]	3.6916	1.6852	1.4526	0.9038
TPCN [142]	**3.6386**	**1.6376**	**1.3535**	**0.8546**
Social-CVAE (Ours)	4.2748	1.9276	2.4881	1.3568

5.6 Ablation Study

5.6.1 Effect of Aggregation Functions

In this section, we present an ablation study on the aggregation functions used in the message-passing network. Because of the sparsity of α-entmax, the usage of sparse graph

Table 5.5: ETH/UCY Testing Results

Model	*min*ADE/*min*FDE, K = 20					
	ETH	HOTEL	UNIV	ZARA1	ZARA2	Average
SGAN[46]	0.81/1.52	0.72/1.61	0.60/1.26	0.34/0.69	0.42/0.84	0.58/1.18
SoPhie[108]	0.70/1.43	0.76/1.67	0.54/1.24	0.30/0.63	0.38/0.78	0.54/1.15
Transformer-TF[38]	0.61/1.12	0.18/0.30	0.35/0.65	0.22/0.38	0.17/0.32	0.31/0.55
STAR[144]	0.36/0.65	0.17/0.36	0.31/0.62	0.26/0.55	0.22/0.46	0.26/0.53
PECNet[86]	0.54/0.87	0.18/0.24	0.35/0.60	0.22/0.39	0.17/0.30	0.29/0.48
Trajectron++[109]	0.39/0.83	0.12/0.21	**0.20**/**0.44**	**0.15**/0.33	**0.11**/0.25	0.19/0.41
AgentFormer[148]	**0.26**/**0.39**	**0.11**/**0.14**	0.26/0.46	**0.15**/**0.23**	0.14/**0.24**	**0.18**/**0.29**
VAE (Ours)	0.59/0.90	0.18/0.26	0.31/0.54	0.21/0.37	0.17/0.31	0.29/0.48
CVAE (Ours)	0.56/0.84	0.18/0.26	0.33/0.58	0.21/0.38	0.18/0.33	0.29/0.48
Social-CVAE (Ours)	0.64/0.99	0.18/0.27	0.35/0.62	0.21/0.37	0.19/0.34	0.32/0.52

attention may induce social posterior collapse. Therefore, we would like to study whether social posterior collapse will still occur if we switch to other aggregation functions. We consider two variants for comparison. The first one is replacing $\mathcal{G}$-entmax with the conventional softmax function, resulting in the following message-passing operations:

$$\begin{aligned} v \rightarrow e:\ & \mathbf{h}_{(i,j)} = f_e\left(\left[\mathbf{h}_i, \mathbf{h}_j, \mathbf{u}_{(i,j)}\right]\right), \\ e \rightarrow v:\ & \hat{\mathbf{h}}_j = \sum_{i \in \mathcal{N}_j} w_{(i,j)} \mathbf{h}_{(i,j)}, \quad \text{where } \mathbf{w}_j = \text{softmax}\left(\left\{\mathbf{h}_{(i,j)}\right\}_{i \in \mathcal{N}_j}\right), \end{aligned}$$

The second variant is using the max aggregation instead of the weighted sum. The message-passing layer becomes the one in Eqn. (5.5) - (5.6). The max aggregation takes the element-wise maximum along the dimension of the hidden unit. Therefore, it allows the number of activated nodes to be at most the dimension of $\mathbf{h}_{(i,j)}$. The social posterior collapse issue should be avoidable if $\mathcal{G}$-entmax is the reason for its occurrence.

$$v \rightarrow e: \mathbf{h}_{(i,j)} = f_e\left(\left[\mathbf{h}_i, \mathbf{h}_j, \mathbf{u}_{(i,j)}\right]\right), \tag{5.5}$$

$$e \rightarrow v: \quad \hat{\mathbf{h}}_j = \text{max-aggregate}\left(\{\mathbf{h}_{(i,j)}\}_{i \in N_j}\right). \tag{5.6}$$

The issue that remains is the evaluation metric. Unlike sparse-GAMP, we do not have the ability to detect social posterior collapse by monitoring the magnitude of AR. We need to find alternative and unified evaluation metrics to compare models with different aggregation functions. We adopt the two feature-importance measures used in [59] as our evaluation metrics: 1) gradient-based measures of feature importance and 2) differences in model output induced by leaving features out. However, instead of studying the importance of single features, we are interested in the contribution of a single agent to model output.

Consequently, we define a customized gradient-based measure as follows:

$$\tau_{g,i} = \frac{1}{2(n-1)T_h} \sum_{j=1,j\neq i}^{n} \left\| \frac{\partial \hat{\mathbf{y}}_{i,T_p}}{\partial \mathbf{x}_j} \right\|_{1,1},$$

where i is the index of the target agent, T_p is the number of predicted frames, $\hat{\mathbf{y}}_{i,T_p}$ is the predicted state of the target agent at the last frame, $\partial \hat{\mathbf{y}}_{i,T_p} / \partial \mathbf{x}_j$ is the partial Jacobian matrix of $\hat{\mathbf{y}}_{i,T_p}$ regarding the observed trajectory of the agent j, and $\|\cdot\|_{1,1}$ defines the entry-wise 1-norm, which sums the absolute values of the matrix's entries. $\tau_{g,i}$ essentially measures the average gradient of the model output regarding the observations of the surrounding agents. If the model ignores the social context, a small $\tau_{g,i}$ is expected. Additionally, we define a customized leave-one-out ADE metric as follows:

$$loo\text{ADE}_i = \frac{1}{n-1} \sum_{j=1,j\neq i}^{n} \text{ADE}\left(\hat{\mathbf{y}}(\mathbf{x}_{-j}), \hat{\mathbf{y}}(\mathbf{x})\right),$$

where $\mathbf{x}_{-j}$ denotes the observed trajectories with the agent j masked out. $loo\text{ADE}_i$ is equal to the average ADE between the normal prediction output and the prediction output with each surrounding agent masked out. Similar to $\tau_{g,i}$, we expect a small $loo\text{ADE}_i$ if social posterior collapse occurs.

We conduct the ablation study on the vehicle prediction task. For both datasets, we repeat the experiments but replace the models with their variants with different aggregation functions. We are mainly interested in comparing the values of τ_g and looADE across the models with the same aggregation functions. Because the other aggregation functions do not encourage a sparse model structure, any surrounding agent could contribute to τ_g and looADE. This makes it less informative to compare them against the model with sparse attention. The results for the validation sets are summarized in Table 5.6 and 5.7. The trends in τ_g and looADE are similar regardless of the aggregation functions. In most circumstances, our social-CVAE model attains the highest scores with large margins, whereas the VAE variant has the lowest τ_g and looADE. Meanwhile, the comparisons of prediction performance are also consistent with the case of sparse-GAMP. Therefore, we can conclude that social posterior collapse is not unique to the models with sparse-GAMPs.

We also observe that changing the formulation from VAE to CVAE always leads to an increase in τ_g and looADE when the alternative aggregation functions are used. In contrast, the values could stay unchanged in the case of sparse-GAMP. We are then curious if merely changing the formulation can alleviate social posterior collapse when sparse-GAMP is not used. We take the softmax variant as an example and investigate its attention maps. However, simply computing the ratio of non-zero attention weights is meaningless because the attention is no longer sparse. To solve this problem, we refine our definition of AR as follows:

$$\text{AR}_\delta = \frac{\sum_{i=1,i\neq j}^{n} \mathbf{1}(\omega_{(i,j)} \geqslant \delta)}{n-1}, \quad \delta \in [0,1].$$

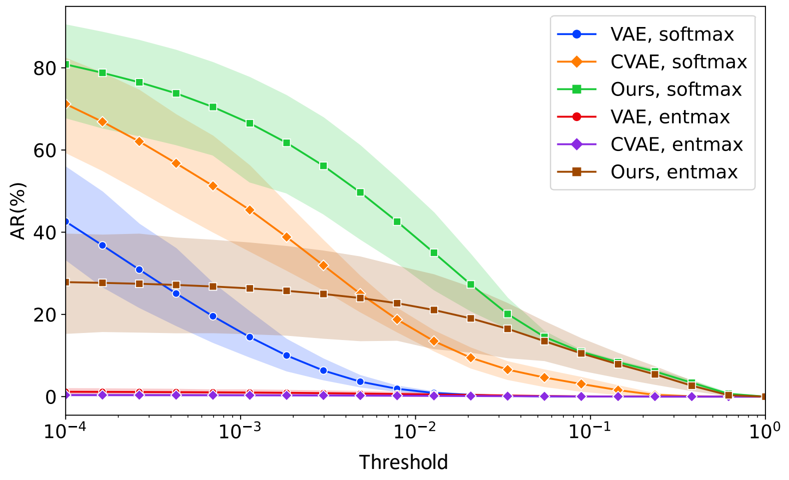


Figure 5.5: $\text{AR}_\delta(\%)$ vs. Threshold on the Argoverse Dataset.

Now AR_δ is a function of a threshold value δ. Instead of looking at a single value at $\delta = 0$, we are interested in seeing how AR_δ changes when increasing δ from 0 to 1. In Fig. 5.5, we plot $\text{AR}_\delta(\%)$ versus the threshold δ for all the models with softmax and entmax functions on the Argoverse dataset. By switching to the conditional model, the softmax variant assigns relatively larger attention weights to surrounding agents. However, the increase in AR_δ mainly occurs at small threshold values. Compared to the social-CVAE models, the ratio of agents receiving large attention weights is lower. This is consistent with our argument that merely changing the formulation is insufficient to alleviate social posterior collapse. Another interesting observation is that the AR_δ curves for the two variants of social-CVAE models coincide when $\delta \geqslant 0.1$. This implies that our sparse graph attention does not prevent the model from identifying interacting agents. It just filters out agents that are recognized as irrelevant to maintain a sparse and interpretable attention map. We can also observe from the evaluated prediction metrics that the sparse graph attention does not interfere with the prediction performance. The social-CVAE models with either attention mechanism achieve similar prediction accuracy. In short, our sparse graph attention function provides a convenient and flexible toolkit that allows us to monitor and analyze social posterior collapse without compromising performance. Although we can still analyze the models without it, these universal metrics, i.e., τ_g and $loo\text{ADE}$, are computationally expensive, especially for evaluation at run time.

5.6.2 Effect of Environmental Information

In this section, we investigate the effect of environmental information on social posterior collapse. In Sec. 5.5, we find that the models behave quite differently on the pedestrian

Table 5.6: INTERACTION Dataset Aggregation Function Comparison

Scene	Aggreg.	Model	τ_g ($\times 10^{-4}$)	*loo*ADE ($\times 10^{-3}$)	K = 6 *min*ADE	K = 6 *min*FDE
IS	max	VAE	0.39 ± 0.10	0.52 ± 0.20	0.28 ± 0.01	0.79 ± 0.01
		CVAE	3.86 ± 1.30	6.38 ± 2.76	$\mathbf{0.24 \pm 0.01}$	$\mathbf{0.69 \pm 0.01}$
		Ours	$\mathbf{14.8 \pm 5.39}$	$\mathbf{20.3 \pm 3.25}$	0.25 ± 0.01	0.70 ± 0.01
	softmax	VAE	0.90 ± 1.37	1.41 ± 2.35	0.26 ± 0.01	0.73 ± 0.02
		CVAE	5.00 ± 1.04	10.6 ± 2.16	$\mathbf{0.24 \pm 0.01}$	$\mathbf{0.67 \pm 0.01}$
		Ours	$\mathbf{33.6 \pm 19.0}$	$\mathbf{17.7 \pm 1.37}$	0.25 ± 0.02	0.70 ± 0.01
	entmax	VAE	0.05 ± 0.09	0.20 ± 0.42	0.26 ± 0.01	0.73 ± 0.02
		CVAE	0.02 ± 0.03	0.02 ± 0.03	$\mathbf{0.24 \pm 0.01}$	$\mathbf{0.68 \pm 0.01}$
		Ours	$\mathbf{29.6 \pm 2.50}$	$\mathbf{8.73 \pm 5.50}$	0.25 ± 0.01	0.70 ± 0.02
RA	max	VAE	0.15 ± 0.05	0.36 ± 0.17	0.30 ± 0.01	0.86 ± 0.02
		CVAE	5.12 ± 2.18	8.42 ± 2.41	$\mathbf{0.26 \pm 0.01}$	$\mathbf{0.74 \pm 0.01}$
		Ours	$\mathbf{36.3 \pm 12.4}$	$\mathbf{28.8 \pm 1.40}$	0.28 ± 0.01	0.76 ± 0.02
	softmax	VAE	1.62 ± 1.57	4.51 ± 3.30	0.27 ± 0.01	0.79 ± 0.02
		CVAE	8.63 ± 2.85	18.4 ± 1.18	$\mathbf{0.26 \pm 0.01}$	$\mathbf{0.73 \pm 0.01}$
		Ours	$\mathbf{40.6 \pm 10.2}$	$\mathbf{22.8 \pm 0.74}$	0.27 ± 0.01	$\mathbf{0.73 \pm 0.01}$
	entmax	VAE	0.00 ± 0.00	0.01 ± 0.01	0.26 ± 0.01	0.75 ± 0.01
		CVAE	11.8 ± 10.9	5.91 ± 5.70	$\mathbf{0.25 \pm 0.01}$	$\mathbf{0.72 \pm 0.01}$
		Ours	$\mathbf{71.0 \pm 71.4}$	$\mathbf{12.9 \pm 8.64}$	0.26 ± 0.01	0.73 ± 0.02
HM	max	VAE	0.45 ± 0.36	0.40 ± 0.18	0.20 ± 0.02	0.52 ± 0.05
		CVAE	37.8 ± 13.3	7.00 ± 0.68	0.15 ± 0.01	0.36 ± 0.01
		Ours	$\mathbf{196 \pm 53.0}$	$\mathbf{8.28 \pm 0.57}$	$\mathbf{0.13 \pm 0.01}$	$\mathbf{0.31 \pm 0.01}$
	softmax	VAE	8.89 ± 0.69	3.92 ± 2.68	0.17 ± 0.01	0.42 ± 0.02
		CVAE	47.0 ± 8.01	$\mathbf{11.5 \pm 0.53}$	0.15 ± 0.01	0.34 ± 0.01
		Ours	$\mathbf{176 \pm 36.0}$	9.42 ± 0.40	$\mathbf{0.14 \pm 0.01}$	$\mathbf{0.32 \pm 0.01}$
	entmax	VAE	10.7 ± 22.4	1.80 ± 3.36	0.16 ± 0.01	0.41 ± 0.04
		CVAE	16.4 ± 36.6	1.93 ± 4.14	0.15 ± 0.02	0.38 ± 0.04
		Ours	$\mathbf{205 \pm 158}$	$\mathbf{6.23 \pm 3.27}$	$\mathbf{0.13 \pm 0.02}$	$\mathbf{0.32 \pm 0.04}$

prediction task and its vehicle counterpart. In our experiments on the ETH/UCY dataset, although switching to CVAE leads to a larger AR value, it does not boost prediction performance. Moreover, the auxiliary task neither improves prediction performance nor further increases the AR value. In the main text, we argue that this is because, unlike vehicles,

Table 5.7: Argoverse Dataset Aggregation Function Comparison

Aggreg.	Model	τ_g ($\times 10^{-3}$)	*loo*ADE ($\times 10^{-2}$)	K = 6 *min*ADE	K = 6 *min*FDE
max	VAE	0.11 ± 0.12	0.24 ± 0.31	1.47 ± 0.09	2.64 ± 0.07
	CVAE	3.91 ± 2.06	3.33 ± 1.82	1.22 ± 0.02	2.15 ± 0.05
	Ours	$\mathbf{13.5 \pm 2.35}$	$\mathbf{13.9 \pm 0.77}$	$\mathbf{1.18 \pm 0.04}$	$\mathbf{2.07 \pm 0.09}$
softmax	VAE	0.12 ± 0.06	0.29 ± 0.12	1.36 ± 0.11	2.40 ± 0.07
	CVAE	5.85 ± 1.64	5.62 ± 1.00	1.16 ± 0.01	1.98 ± 0.01
	Ours	$\mathbf{13.1 \pm 2.17}$	$\mathbf{8.83 \pm 1.27}$	$\mathbf{1.13 \pm 0.01}$	$\mathbf{1.95 \pm 0.03}$
entmax	VAE	0.02 ± 0.02	0.07 ± 0.06	1.29 ± 0.04	2.32 ± 0.05
	CVAE	0.01 ± 0.01	0.02 ± 0.01	1.19 ± 0.01	2.08 ± 0.02
	Ours	$\mathbf{7.30 \pm 2.95}$	$\mathbf{5.77 \pm 2.34}$	$\mathbf{1.15 \pm 0.02}$	$\mathbf{1.97 \pm 0.02}$

Table 5.8: INTERACTION Dataset without Map - HM

Map	Model	AR(%)	K = 1 *min*FDE	K = 1 *min*ADE	K = 6 *min*FDE	K = 6 *min*ADE
Yes	VAE	8.68 ± 8.36	0.87 ± 0.08	0.29 ± 0.03	0.42 ± 0.04	0.16 ± 0.01
	CVAE	5.42 ± 10.2	0.83 ± 0.03	0.28 ± 0.03	0.40 ± 0.05	0.16 ± 0.02
	Ours	$\mathbf{15.5 \pm 8.13}$	$\mathbf{0.65 \pm 0.09}$	$\mathbf{0.22 \pm 0.02}$	$\mathbf{0.32 \pm 0.04}$	$\mathbf{0.13 \pm 0.01}$
No	VAE	24.4 ± 14.4	0.98 ± 0.08	0.34 ± 0.02	0.51 ± 0.05	0.20 ± 0.02
	CVAE	$\mathbf{43.5 \pm 8.72}$	0.72 ± 0.02	0.25 ± 0.01	0.36 ± 0.01	0.15 ± 0.01
	Ours	33.7 ± 5.77	$\mathbf{0.62 \pm 0.01}$	$\mathbf{0.22 \pm 0.01}$	$\mathbf{0.32 \pm 0.01}$	$\mathbf{0.13 \pm 0.01}$

pedestrians do not have a long-term dependency on their interaction. As a result, the model cannot benefit from encoding the historical social context. However, we also adopt different input representations for the two prediction problems. In the vehicle prediction task, the input graphs have additional lanelet nodes, contributing to the difference in model behavior.

To answer this question, we trained models for vehicle prediction with the same representation as pedestrians, which means removing lanelet nodes and adding self-edges instead. We choose to study this problem on the highway merging subset of the INTERACTION dataset because the interaction between vehicles depends less on the map than urban driving. We follow the same practice to run five trials for each model variant and report the means and standard deviations for all the metrics. The results are summarized in Table 5.8. We also copy the results from Table 5.1 for convenient comparison.

We think social posterior collapse still occurs in the baseline VAE model. As social context affects highway driving to a larger degree than road structure, our social-CVAE model that encodes social context can maintain consistent prediction accuracy without map information. On the other hand, removing lanelet nodes leads to a significant drop in the baseline VAE model's prediction accuracy. This indicates that the baseline VAE model does not utilize social context well but relies heavily on map information, which is verified by its lower AR value. In particular, one of the five VAE models has a nearly zero AR value.

The analysis becomes complicated when comparing our model against the CVAE variant. Introducing the auxiliary task still improves prediction accuracy. Further, we note that the CVAE model itself has a lower prediction error after removing lanelet nodes. If the historical social context is the dominant factor determining prediction performance, we may draw two conclusions. First, the CVAE model suffers less from social posterior collapse under the graph representation without lanelet nodes. Second, the auxiliary prediction task can further encourage the model to encode social context. However, we do not have direct evidence showing that the auxiliary task encourages the model to encode social context —the auxiliary task does not lead to a larger AR value.

We think the reason behind this is that removing lanelet nodes makes the attention map less likely to become sparse. In most driving scenarios, the number of lanelet nodes dominates the number of agents. Removing lanelet nodes reduces the number of incoming edges for each agent node. Consequently, the agent-to-agent edges compete with a single self-edge to gain attention instead of numerous lanelet-to-agent edges. This implies that the format of representation affects the impact of social posterior collapse on the model. More importantly, it shows the limitation of using AR as the single metric to analyze social posterior collapse. This is particularly effective when the agent vertices are of a high degree, but the analysis may be inconclusive otherwise. In future studies, we will investigate other metrics and tools that can be applied to a broader range of problems.

5.6.3 Effect of KL Annealing

In Sec. 5.2.2, we argue that social posterior collapse is different from the well-known posterior collapse issue, and typical techniques that address posterior collapse, e.g., KL annealing, may not be effective in mitigating social posterior collapse. In this section, we test if one of the annealing methods —cyclical annealing schedule [33] —can effectively alleviate social posterior collapse. Again, we use the HM scenario from the INTERACTION dataset as an example to study this problem. When training the baseline VAE and CVAE models, we incorporate the cyclical annealing schedule plotted in Fig. 5.6 to adjust the magnitude of β over training epochs. The results are summarized in Table 5.9. The cyclical annealing schedule neither improves prediction performance nor increases AR value consistently. The VAE model with a cyclical schedule does have a larger average AR value. However, two out of the five trials attain zero AR, which causes the large standard deviation. In summary, the cyclical annealing schedule does not alleviate the social posterior collapse issue in the experiments, which is consistent with our previous argument.

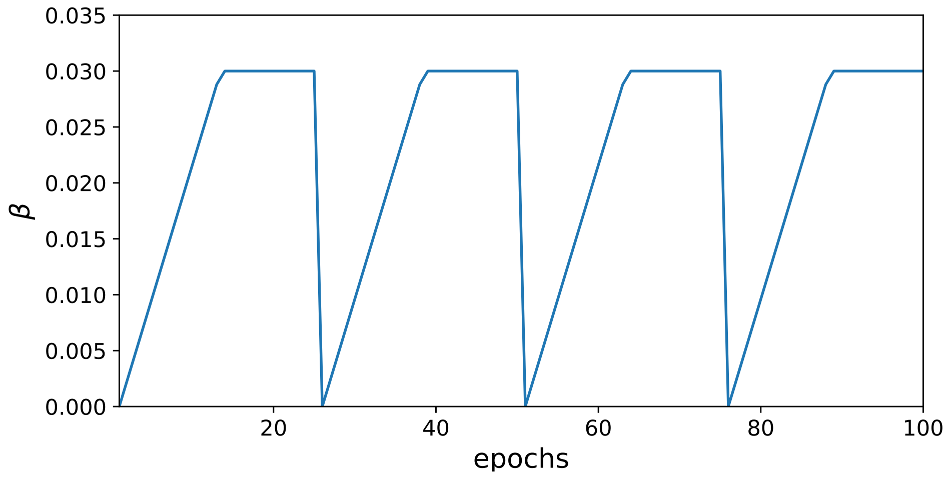


Figure 5.6: The cyclical annealing schedule adopted in our experiments. Each annealing cycle consists of 25 training epochs. The magnitude of β increases linearly from zero to the maximum value in the first half of the cycle, and then it remains unchanged in the remaining epochs. The maximum value is set to be 0.03, which is the value used in previous experiments with constant β.

5.7 Experiment Details

In this section, we report additional details of the experiments, including the data processing scheme, implementation details, and hyper-parameters used.

Table 5.9: INTERACTION Dataset KL Annealing Experiment - HM

Model	Cycling	AR(%)	K = 1		K = 6	
			*min*FDE	*min*ADE	*min*FDE	*min*ADE
VAE	No	8.68 ± 8.36	0.87 ± 0.08	0.29 ± 0.03	0.42 ± 0.04	0.16 ± 0.01
	Yes	17.7 ± 16.5	0.96 ± 0.10	0.33 ± 0.03	0.52 ± 0.04	0.20 ± 0.01
CVAE	No	5.42 ± 10.2	0.83 ± 0.03	0.28 ± 0.03	0.40 ± 0.05	0.16 ± 0.02
	Yes	7.65 ± 10.5	0.84 ± 0.07	0.28 ± 0.02	0.41 ± 0.03	0.16 ± 0.01
Ours	-	$\mathbf{15.5 \pm 8.13}$	$\mathbf{0.65 \pm 0.09}$	$\mathbf{0.22 \pm 0.02}$	$\mathbf{0.32 \pm 0.04}$	$\mathbf{0.13 \pm 0.01}$

5.7.1 Data Processing Scheme

INTERACTION Dataset. The access to the dataset is granted for non-commercial usage through its official website: `https://interaction-dataset.com` (Copyright©All Rights Reserved). First, we use the script from the official code repository to split the training and validation sets and segment the data. Each sample has a length of 4s, with an observation window of 1s and a prediction window of 3s. The sampling frequency is 10Hz. We further augment the dataset by iteratively assigning each vehicle in a sample as the target vehicle. Subsequently, we follow the common practice to translate and rotate the coordinate system, such that the target vehicle is located at the origin with a zero-degree heading angle in the last observed frame.

Regarding map information, the INTERACTION dataset provides maps in a format compatible with the lanelet representation. For each lanelet, we fit its boundaries to two B-splines through spline regression so that we can uniformly sample a fixed number of points on each boundary. Apart from the coordinates, we add additional discrete features (e.g., boundary type, the existence of a stop sign) to each boundary point.

Argoverse Dataset. We download the dataset (v1.1), including the training, validation, and testing subsets, from its official website: `https://www.argoverse.org/data.html` (Copyright©2020 Argo AI, LLC). Each sample in the Argoverse Dataset has a length of 5s, with an observation window of 2s and a prediction window of 3s. The sampling frequency is 10Hz. We follow a similar scheme to process the dataset. However, since each sample contains vehicles located in many city blocks, we first filter out irrelevant vehicles and road segments. We remove surrounding vehicles whose distance to the target vehicle is larger than a certain threshold value in the last observed frame. To identify irrelevant road segments, we use a heuristic-based graph search algorithm similar to the one used in [149] to obtain the road segments of interest (ROI). Instead of setting a threshold Euclidean distance, we decide whether a road segment is relevant by estimating the traveling distance from the target vehicle's location to the segment. The algorithm is summarized in Alg. 1. Given x, the coordinates of the target vehicle in the last observed frame, we set the traveling distance threshold $d_{\max}$ based on the displacement of the target vehicle during the observed time horizon. A larger distance threshold is necessary if the target vehicle is driving at high speed. We initialize the graph search by finding road segments close to x from the map. Then, we expand the search graph by adding adjacent, preceding, and succeeding road segments and finding all segments within the threshold of traveling distances. We use simple heuristics to compute the traveling distance between two segments. If two segments are adjacent, the distance is set to zero. If one segment is a predecessor or successor of the other segments, we set the distance to be the average length of their centerlines. Fig. 5.7 shows an example of the resulting road segments.

Another issue is that the heading angles of the vehicles are not provided. We need to estimate the heading angles of the target vehicles in order to rotate the coordinate system. However, the trajectory data are noisy and include tracking errors. Simply interpolating the coordinates between consecutive time steps results in noisy estimation. Instead, we first

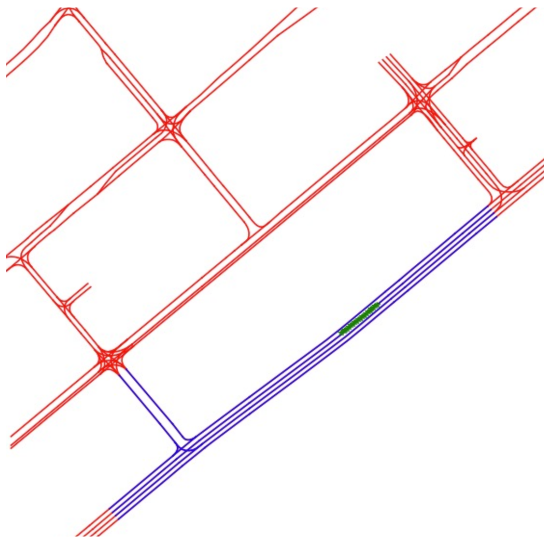


Figure 5.7: Road segments found by the graph search algorithm. We denote the observed trajectory of the target vehicle by the green dots. The road segments returned by the graph search algorithm are highlighted in blue, while the other road segments are drawn in red.

estimate the heading angle in each observed frame by interpolating the coordinates and then obtain a smooth estimation of the heading angle in the last observed frame as follows:

$$\hat{\psi}_{T_h} = \sum_{t=0}^{T_h} \lambda^{T_h - t} \psi_t,$$

where T_h is the number of observed frames, ψ_t is the estimated heading in the t^{th} frame, $\lambda \in (0, 1)$ is the forgetting factor, and $\hat{\psi}_{T_h}$ is the smoothed heading estimation in the last observed frame.

ETH/UCY Datasets. For the ETH/UCY datasets, we adopt the leave-one-out evaluation protocol as in prior works [46, 108, 109, 86, 148] to obtain five groups of datasets: ETH & HOTEL (from ETH) and UNIV, ZARA1 & ZARA2 (from UCY). The data from the corresponding scenario are left out as testing data in each group, and the remaining data are used for training and validation. We use the segmented datasets provided by the code base of social-GAN (MIT License) [46]. Each sample has a length of 8s, with an observation window of 3.2s and a prediction window of 4.8s. The sampling frequency is 2.5Hz. We do not use any visual or semantic information to ensure fair comparisons to prior works. The origin of the coordinate system is translated to the mean position of all agents atin the last observed frame. Random rotation [108, 109] is adopted for data augmentation.

5.7.2 Implementation Details

We implement our social-CVAE model using Pytorch 1.8 (Copyright©Facebook, Inc) [92] and Pytorch Geometric (PyG) 1.7 (MIT License) [28], a geometric deep learning extension library for Pytorch that implements various popular GNN models. The sparse-GAMP network is implemented by adapting the base message-passing class from PyG. The $\mathcal{G}$-entmax function is implemented using the entmax package (MIT License) from [98]. The function f_e in Eqn. (5.4) consists of two MLP networks: one network encodes $\mathbf{h}_i$ and $\mathbf{h}_j$ separately; one network generates $\mathbf{h}_{(i,j)}$ after concatenating the node embeddings with $\mathbf{u}_{(i,j)}$. We select the hyper-parameters through cross-validation with the aid of Tune (Copyright©2021 The Ray Team) [76]. All the MLPs used in the model are one-layer MLPs with layer normalization applied [7]. All of the networks, including MLPs and GRUs, have 64 hidden units. For the experiments on the INTERACTION dataset, we choose a latent space of 16 dimensions. For the experiments on the Argoverse and ETH/UCY datasets, we choose a latent space of 32 dimensions. For the vehicle prediction experiments, we set $\beta = 0.03$. For the pedestrian prediction experiments, we set $\beta = 0.01$. For the weight of the auxiliary loss function, we set $\alpha = 0.3$ on the INTERACTION dataset, $\alpha = 0.5$ on the Argoverse dataset, and $\alpha = 0.2$ on the ETH/UCY dataset. We use Adam [65] to optimize the objective function. To generate trajectories for evaluation, if $K = 1$, we sample a single trajectory by taking the means from the prior or conditional prior as the latent variables. If $K > 1$, we randomly sample the latent variables to generate multiple trajectories. However, the Argoverse Motion Forecasting Challenge evaluates the metrics for $K = 1$ by picking one trajectory out of all the submitted samples. Therefore, for evaluation on the Argoverse dataset, we randomly sample $K - 1$ trajectories and leave the last one as the trajectory corresponding to the mean value of the latent variables. For the vehicle prediction experiments, all the models are trained for 100 epochs with a batch size of 40. For the pedestrian experiments, we choose a batch size of 20. All the models were trained with an Intel Core i9-9920X (12 Cores, 3.50 GHz) and four RTX 2080 Ti GPUs. However, only a quarter of the memory of a single GPU is required to train one model.

5.8 Discussion

Limitations

As the first study on social posterior collapse, we cannot explore every aspect of this subject. Many factors could contribute to this phenomenon. Our experimental results can only speak for the particular model we develop and the tasks we study. The occurrence of social posterior collapse and its effect on overall performance may vary across different problems and different model architectures. For instance, our finding from the pedestrian trajectory prediction task is different from its vehicle counterpart. The format of the graph representation, especially how the environmental information is incorporated, is also important. Moreover, diagnosis based on the AR value could be inconclusive under different graph representations. In Sec. 5.6.2, we show that the AR values of the three models become similar after removing the lanelet nodes for the highway merging subset of the INTERACTION dataset. However, this does not mean that social posterior collapse does not occur. Our social-CVAE model can maintain consistent prediction accuracy without map information, whereas the baseline VAE model has a significant drop in performance. This implies that the baseline VAE model does not properly utilize the historical social context, but we cannot assert that social posterior collapse occurs due to the lack of direct evidence.

Nevertheless, we do demonstrate that social posterior collapse is not unique to our sparse-GAMP module (Sec. 5.6.1). Even if we switch to other aggregation functions, we still find evidence suggesting its occurrence. Our sparse graph attention function, $\mathcal{G}$-entmax, is a convenient and flexible toolkit that allows us to monitor and analyze social posterior collapse without compromising performance. In the future, we will work further in this direction to explore alternative diagnosis toolkits that can be applied to detect social posterior collapse with a broader range of models.

Connections to Related Works

We conclude our discussion with some thoughts about prior works on VAE-based multi-agent behavior modeling. We are curious whether any elements in their models have implicitly tackled this issue. However, we would like to emphasize that it is still necessary to explicitly study social posterior collapse under their settings in the future, as it could provide helpful guidance to avoid social posterior collapse in model design. Due to space limits, we only discuss works using techniques potentially related to social posterior collapse, especially those providing evidence that interacting behaviors have been appropriately modeled (e.g., attention maps or visualized prediction results in interactive scenarios).

Trajectron [56] and Trajectron++ [109], which are formulated as CVAEs, were used to establish the previous state-of-the-art results on ETH/UCY. Unlike our model, they adopted a discrete latent space to account for the multi-modality in human behavior. They did not examine how their models utilize social context. However, we think that a discrete latent space could potentially alleviate the social posterior collapse issue. Compared to a continuous random variable, a discrete one can only encode a limited amount of information. This prevents the model from bypassing the social context since a discrete latent variable is not sufficient to encode all the information of a long trajectory. For the same reason, NRI [67], which adopts a discrete latent space for modeling interactive systems, is also potentially relevant. PECNet [86] is another CVAE-based trajectory prediction model. Instead of conditioning on the entire future trajectory, they proposed an endpoint VAE, in which the posterior latent variables are only conditioned on the endpoints. This could be a solution, as it limits the amount of information the latent space can encode from the future trajectory.

In [124], Suo et al. proposed a multi-agent behavior model for traffic simulation under the CVAE framework. They demonstrated that their method was able to simulate complex and realistic interactive behaviors. In particular, they augmented ELBO with a commonsense objective that regularizes the model from synthesizing undesired interactions (e.g., collisions).

We think it plays a similar role to our auxiliary prediction task. The last work we would like to discuss is the AgentFormer model [148] mentioned in Sec. 5.5.2. Their results suggest that incorporating an autoregressive decoder and a future social context encoder could be more effective in interaction modeling, especially for systems without long-term dependency (e.g., pedestrians). However, as we have mentioned above, the autoregressive decoder introduces another issue if it is overly powerful.

5.9 Chapter Summary

In this chapter, we point out an under-explored issue, which we refer to as social posterior collapse, in the context of VAEs in multi-agent modeling. We argue that one of the commonly adopted formulations of VAEs in multi-agent modeling is prone to ignoring the historical social context when predicting the future trajectory of an agent. We analyze the reason for this and propose several measures to alleviate social posterior collapse. Subsequently, we design a GNN-based realization of the general framework incorporating the proposed measures, which we refer to as social-CVAE. Specifically, social-CVAE is an explainable model incorporating a novel sparse-GAMP layer that helps us detect and analyze social posterior collapse. In our experiments, we show that social posterior collapse occurs in real-world trajectory prediction problems and that the proposed measures effectively alleviate this issue. Further, the experimental results imply that social posterior collapse could cause poor generalization performance in novel scenarios if the future movement of the agents indeed depends on their historical social context. In the future, we will utilize the toolkit developed in this work to explore social posterior collapse in a broader range of model architectures and interacting systems. Additionally, we will explore other XAI techniques to develop more robust and versatile toolkits for interaction modeling analysis.

Algorithm 1 ROI Graph Search

```
1: function ROIGraphSearch($x, d_{\max}, d_{\text{init}}, map$)
2:     $segments \leftarrow map$.segmentsContainXY($x$)    ▷ A set of segments containing $x$
3:     while $segments$ is empty and $d_{\text{init}} < d_{\max}$ do    ▷ Search local region if empty
4:         $segments \leftarrow map$.segmentsInBoundingBox($x, d_{\text{init}}$)
5:         $d_{\text{init}} \leftarrow 2d_{\text{init}}$
6:     end while
7:     $pool \leftarrow$ initialize a FIFO queue
8:     for $segment$ in $segments$ do
9:         $node \leftarrow$ Node($segment, 0, 0$)    ▷ Create node with segment, length and distance
10:        $pool \leftarrow$ Insert($node, pool$)
11:    end for
12:    $ROI \leftarrow$ an empty dictionary of nodes
13:    while $pool$ is not empty do
14:        $node \leftarrow$ Pop($pool$)
15:        if $node.segment$ not in $ROI$ or $ROI[segment].distance \geqslant node.distance$ then
16:            $ROI[segment] \leftarrow node$
17:        end if
18:        $children \leftarrow map$.adjacentSegments($node.segment$)    ▷ Get adjacent road segments
19:        for $child$ in $children$ do
20:            $length \leftarrow map$.segmentCenterline($child$)    ▷ Get centerline length
21:            $node \leftarrow$ Node($child, length, node.distance$)
22:            $pool \leftarrow$ Insert($node, pool$)
23:        end for
24:        $predecessors \leftarrow map$.predecessor($node.segment$)    ▷ Get preceding road segments
25:        $successors \leftarrow map$.successor($node.segment$)    ▷ Get succeeding road segments
26:        $children \leftarrow predecessors + successors$    ▷ Combine predecessors and successors
27:        for $child$ in $children$ do
28:            $length \leftarrow map$.segmentCenterline($child$)
29:            $distance \leftarrow \frac{1}{2}(length + node.length) + node.distance$
30:            if $distance \leqslant d_{\max}$ then
31:                $node \leftarrow$ Node($child, length, distance$)
32:                $pool \leftarrow$ Insert($node, pool$)
33:            end if
34:        end for
35:    end while
36:    return $ROI$
37: end function
```

Chapter 6

Diagnosing Conditional Behavior Prediction with Shapley Values

6.1 Introduction

In the last chapter, we studied the interaction modeling problem in behavior prediction. Like most existing works [149, 43], the prediction model we focus on follows a *passive* prediction scheme, where the target agents' future trajectories are predicted given their historical trajectories and those of other surrounding agents. When using such a prediction model, downstream decision-making modules determine the autonomous agent's action according to the predicted trajectories in a passive manner. To ensure safety under various predicted trajectories of others, the ego agent must be overly conservative with inefficient maneuvers, especially in highly interactive scenarios. This is because passive prediction models ignore the fact that the autonomous agent's future actions can influence other agents' behavior. To this end, researchers have begun to investigate a more coherent interactive prediction and planning framework that relies on predicting the surrounding agents' future trajectories conditioned on the ego agent's future actions [111, 125, 103, 61, 109, 119, 82, 130]. Under such frameworks, the autonomous agents can reason about potential actions while considering their influence on surrounding agents. This can then induce more efficient and less conservative maneuvers in interactive scenes.

Some of these prior works merely demonstrated that their model architecture could support conditional prediction [125, 61]. Another line of work focused on the closed-loop performance, which relies on a simulation environment [111, 103, 82]. More interestingly, some existing works formulated an alternative prediction task to evaluate the prediction module in a self-contained way [119, 109, 130]. We follow [130] and refer to this task as *conditional behavior prediction* (CBP). In the CBP task, the future trajectories of the target agents are predicted conditioned on the ground-truth future trajectory of an assigned ego agent. Standard prediction metrics are adopted to quantify the performance. This allows us to leverage large-scale naturalistic traffic datasets to develop and validate a conditional prediction model

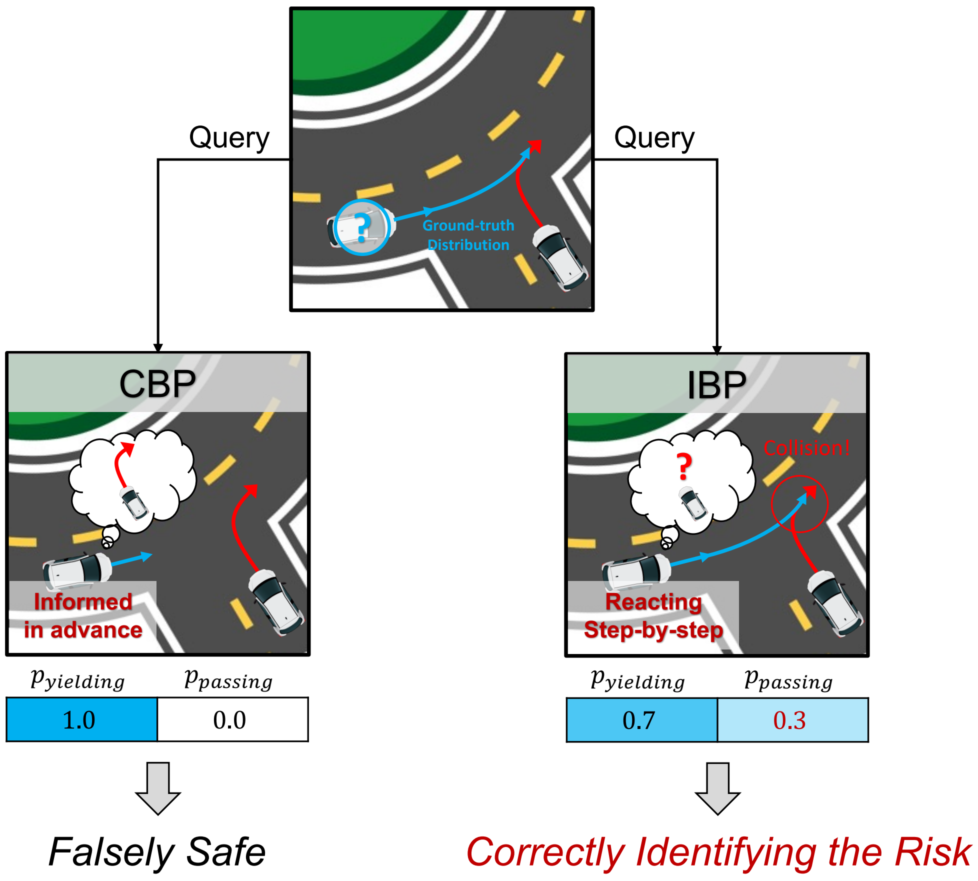


Figure 6.1: An illustration of the difference between CBP and IBP. The robot car plans to enter the roundabout aggressively and force the human car in the roundabout to yield. It queries a prediction model regarding whether the human car will yield or pass. The CBP model predicts the posterior distribution of the human car's behavior conditioned on the plan. Intuitively, it models the human behavior given that the driver is informed about the robot car's plan in advance. Therefore, CDP will always predict that the human car will yield to the robot car. In fact, the human car will only react to the robot car's action at each time step. Consequently, the human car may attempt to pass first, as it has the right of way, which may lead to a collision. The IBP model is able to warn the robot car of the safety risk.

before closed-loop testing. A model that can achieve the smallest prediction error after it is given the additional future information of the ego agent is considered the best. However, we can only evaluate the prediction accuracy given the actual future trajectory of the ego agent with a static offline dataset. It is impossible to quantify the model performance when it is queried by an arbitrary plan of the ego agent. Therefore, we argue that we should be more careful when interpreting the evaluation results. In particular, we argue that it is risky to train and evaluate the model for *conditional inference*. In the current CBP task, the prediction model essentially learns the posterior distribution of future trajectories conditioned on the future trajectory of the ego agent. In this way, the ego agent's future trajectory is treated as an *observation*. Since the actual ego agents in the offline dataset make decisions based on the states of the surrounding agents, the surrounding agents under CBP are implicitly assumed to obtain additional hints about the future behavior of the ego agents. With such an unrealistic assumption, it is natural to consider the CBP model with the lowest prediction error to be the best option for the CBP task. However, the surrounding agents in the real world are not informed of the planned trajectories of the ego agents. Consequently, as illustrated in Fig. 6.1, there will be a discrepancy between 1) what information an autonomous agent receives by querying a CBP model with a potential plan and 2) how the others will actually react if the agent executes the plan. As we will show later in this chapter, this discrepancy may lead to overly confident anticipation of the ego agent's influence on the surroundings, resulting in potential safety hazards during online usage.

This discrepancy is formally captured in the theory of causality [93] by the difference between *observation* and *intervention*. With the intervention of a set of random variables, we enforce the value of a random variable without treating it as the consequence of other random variables. The resulting distribution of the remaining random variables under the intervention is consistent with what will actually happen if we have the ability to manipulate the target random variable as desired. Consequently, we argue that we should build the prediction model to approximate the future trajectory distribution under the intervention of enforcing the ego agent's future trajectory. We refer to this new task as the *interventional behavior prediction* (IBP) task. In IBP, we still want to train and evaluate the model with an offline dataset. The setting is essentially the same as CBP, except for learning an interventional distribution instead of a conditional one.

The remaining issue is how to properly evaluate an IBP model with an offline dataset. Without knowing the ground-truth distribution under intervention, we can only compare the model's output against the ground-truth future trajectories for evaluation. However, such evaluation metrics are naturally biased toward a CBP model. The dataset is collected without intervention. The ego agent in the dataset follows an internal reactive policy. Therefore, the distribution of ground-truth labels given the same input essentially follows a conditional distribution. As a result, a CBP model will consistently outperform an IBP model if prediction accuracy is the only evaluation metric with an offline dataset. Accordingly, we propose verifying the inherent temporal independence of a prediction model before comparing the prediction performance to ensure a proper evaluation of the IBP task. Under the interventional distribution, the predicted states of the target agents at earlier time steps should be independent from the ego agent's states at later time steps. We propose a Shapley-value-based [115, 84, 85] metric to verify whether the model obeys this temporal independence. More importantly, we show that a state-of-the-art CBP model on a popular prediction benchmark indeed violates temporal independence, and its prediction accuracy benefits from this violation. The results support the necessity of establishing a benchmark

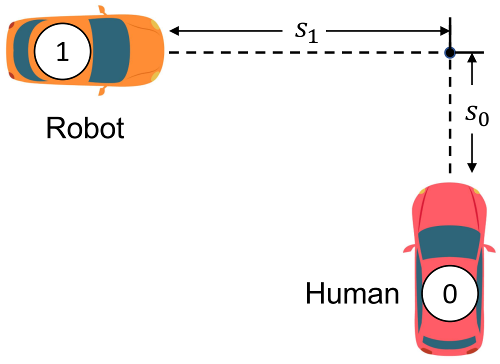


Figure 6.2: A motivating toy example, where a human car and a robot car are driving toward a collision point.

for our newly formulated IBP task to replace the commonly adopted CBP benchmarks, in which the proposed Shapley-value-based metric will play an important role.

The rest of the chapter is organized as follows. In Sec. 6.2, we explain the difference between CBP and IBP and demonstrate the risk of using CBP with a motivating toy example. In Sec. 6.3, we formulate the Shapley-value-based metric we propose for verifying temporal independence and quantify the impact of CBP. In Sec. 6.4, we study a CBP model with the proposed metric to demonstrate that such a model indeed violates temporal independence. Moreover, failure to use the proposed metric results in misleading evaluation results. In Sec. 6.5, we conclude the paper with a discussion on insights for future model design and IBP benchmarks.

6.2 A Motivating Example

We begin our discussion by studying a motivating toy example to demonstrate the issue of using conditional inference for interactive prediction. We consider the example depicted in Fig. 6.2, where two cars are driving toward a collision point. One of them is controlled by a human driver, while the other is an autonomous robot car. As an analogy of the CBP task, the robot car can query the posterior distribution of the human car's trajectory conditioned on the planned trajectory of the robot car. The robot car can then evaluate the risk of multiple planned trajectories and select the optimal one to execute.

We model the human drivers' behavior with IDM [132, 131]. Each car has two states at each time step, $s_{i,t}$ and $v_{i,t}$, where $s_{i,t}$ is the displacement relative to the collision point and $v_{i,t}$ is the absolute velocity. We denote the state vector as $\mathbf{x}_{i,t} = [s_{i,t}\ v_{i,t}]^\intercal$. Each car is assigned a target position to follow at each step, depending on which car has the right of way. If a car has the right of way, it is asked to follow a target substantially far away. Otherwise, if the other car has the right of way and has not passed the collision point, the target is set as the collision point. We determine which car has the right of way based on which car has smaller time headway at the current time step. The time headway is defined as follows:

$$T_{head,i,t} = \max\left(\frac{s_{i,t}}{v_{i,t}}, 0\right).$$

Given a target position, the car dynamics are governed by the intelligent driver model as follows:

$$\begin{aligned} s_{i,t+1} &= s_{i,t} - \Delta t \cdot v_{i,t}, \\ v_{i,t+1} &= v_{i,t} + \Delta t \cdot \left\{ a\left[1 - \left(\frac{v_{i,t}}{v_0}\right)^{\delta} - \left(\frac{s^*(v_{i,t}, \Delta v_{i,t})}{s_{i,t} - d_{i,t}}\right)^2\right] + \omega_{i,t} \right\}, \end{aligned}$$

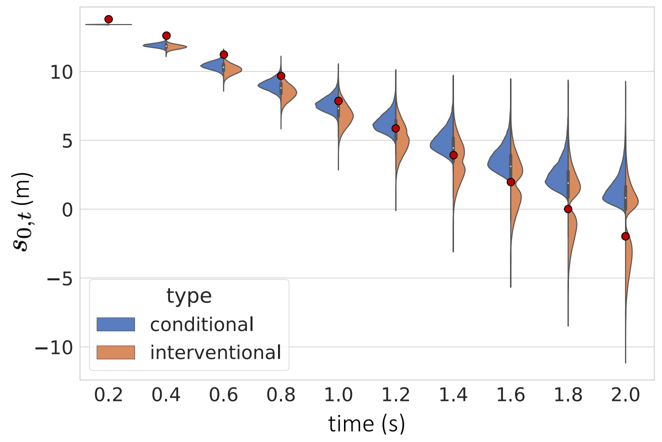
where

$$\begin{aligned} \Delta v_{i,t} &= v_{i,t} - v_0, \\ s^*(v_{i,t}, \Delta v_{i,t}) &= s_0 + \max\left(0, v_{i,t}T + \frac{v_{i,t}\Delta v_{i,t}}{2\sqrt{ab}}\right), \\ \omega_{i,t} &\sim \mathcal{N}\left(0, \sigma^2\right). \end{aligned}$$

The term $d_{i,t}$ denotes the target position. It is equal to zero if the target point collides with the collision point. Otherwise, a large negative value is assigned to $d_{i,t}$. The Gaussian noise $\omega_{i,t}$ is added to inject randomness. The remaining parameters are defined as in the standard IDM model. Readers may refer to [132] for detailed definitions. In our experiments, we set $v_0 = 10\text{m/s}$, $T = 2\text{s}$, $s_0 = 4\text{m}$, $\delta = 4$, $a = 1\text{m/s}^2$, $b = 1.5\text{m/s}^2$, $\Delta t = 0.2\text{s}$, and $\sigma = 4\text{m/s}^2$.

In the CBP task, we aim to approximate the distribution of the human car's future trajectory conditioned on the initial states of the two cars and the future trajectory of the robot car, i.e., $p(\mathbf{x}_{0,1:T_H}|\mathbf{x}_{0,0}, \mathbf{x}_{1,0}, \mathbf{x}_{1,1:T_H})$, where T_H denotes the number of time steps. The robot car can query the conditional distribution with a planned trajectory $\hat{\mathbf{x}}_{1,1:T_H}$. In our experiment, we use likelihood weighting [107] to estimate the conditional distribution given an evidence set $\{\hat{\mathbf{x}}_{0,0}, \hat{\mathbf{x}}_{1,0}, \hat{\mathbf{x}}_{1,1:T_H}\}$. Meanwhile, we can approximate the actual distribution of $\mathbf{x}_{0,1:T_H}$ after executing $\hat{\mathbf{x}}_{1,1:T_H}$ via multiple simulation trials. In Fig. 6.3(a), we compare the two distributions under the same initial conditions and query trajectory. We set $s_{0,0} = s_{1,0} = 15\text{m}$, $v_{0,0} = 8\text{m/s}$, $v_{1,0} = 5\text{m/s}$, and $T_H = 10$. Since the robot car has a smaller initial speed, it is more likely to yield to the human car. However, we let the robot car execute an aggressive maneuver, where it accelerates with an acceleration of 5m/s^2 until reaching the speed of 10m/s.

The conditional distribution implies that the human car always yields to the robot car. However, the human car may actually not yield to the robot car when the robot car executes the planned trajectory. Even if the human car eventually yields to the robot car, it starts decelerating much later than the conditional distribution suggests. If we evaluate the risk based on the conditional distribution, we may falsely conclude that the human car will always yield to the robot car and so the robot car can safely pass the intersection at high speed, which leads to an overly aggressive and unsafe maneuver. This can be further verified by



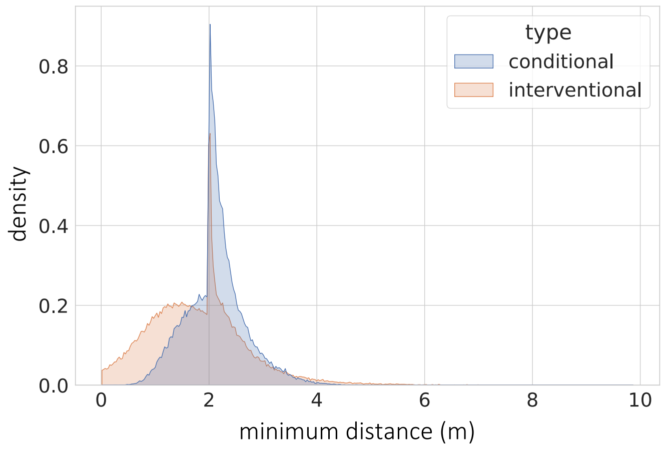


Figure 6.3: **Top**: Histograms of $s_{0,t}$ for $t = 1, \cdots, T_H$ under the conditional distribution and the distribution given the interventional action. The red dots denote the planned trajectory of the robot car; **Bottom**: Normalized histograms of minimum distance between cars under these two distributions. The histograms are based on 10000 simulation trials.

estimating the histograms of the minimum distance between the two cars under these two distributions (Fig. 6.3(b)). The minimum distance is biased under conditional inference. In particular, the conditional distribution falsely implies that the two cars will never collide.

The toy example demonstrates the discrepancy between the reality and the anticipation from conditional inference. Formally, the conditional distribution is governed by the Bayesian network in Fig. 6.4(a), where the initial states and the query trajectory are treated as an *observation*. However, the system is actually governed by the Bayesian network in Fig. 6.4(b) when the robot executes $\hat{\mathbf{x}}_{1,1:T_H}$. The incoming edges of $\mathbf{x}_{1,i}$ are removed because the robot car follows a fixed trajectory regardless of the other car's reaction. If fact, if we treat the Bayesian network governing the system as a causal Bayesian network [93], then Fig. 6.4(b)

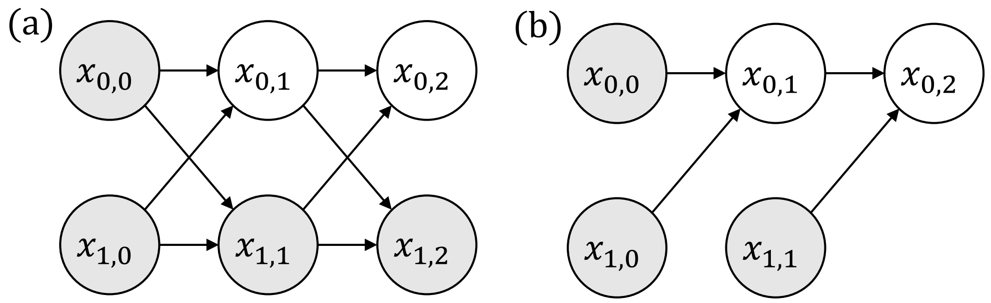


Figure 6.4: (a) The Bayesian network representing the conditional distribution denoted by $p(\mathbf{x}_{0,1:T_H}|\hat{\mathbf{x}}_{0,0}, \hat{\mathbf{x}}_{1,0}, \hat{\mathbf{x}}_{1,1:T_H})$; (b) The Bayesian network representing the distribution resulting from the intervention $do(\mathbf{x}_{1,1:T_H} = \hat{\mathbf{x}}_{1,1:T_H})$, denoted by $p(\mathbf{x}_{0,1:T_H}|\hat{\mathbf{x}}_{0,0}, \hat{\mathbf{x}}_{1,0}, do(\hat{\mathbf{x}}_{1,1:T_H}))$.

represents the distribution resulting from the interventional action $do(\mathbf{x}_{1,1:T_H} = \hat{\mathbf{x}}_{1,1:T_H})$, denoted by $p(\mathbf{x}_{0,1:T_H}|\hat{\mathbf{x}}_{0,0}, \hat{\mathbf{x}}_{1,0}, do(\hat{\mathbf{x}}_{1,1:T_H}))$. The difference between the two distributions, $p(\mathbf{x}_{0,1:T_H}|\hat{\mathbf{x}}_{0,0}, \hat{\mathbf{x}}_{1,0}, \hat{\mathbf{x}}_{1,1:T_H})$ and $p(\mathbf{x}_{0,1:T_H}|\hat{\mathbf{x}}_{0,0}, \hat{\mathbf{x}}_{1,0}, do(\hat{\mathbf{x}}_{1,1:T_H}))$, mirrors the difference between seeing and doing [93]. By conditional inference, we aim to infer the distribution of $\mathbf{x}_{0,1:T_H}$ after *observing* $\hat{\mathbf{x}}_{0,1:T_H}$, intuitively speaking, how the human driver behaves if knowing the robot car will execute $\hat{\mathbf{x}}_{0,1:T_H}$ in advance. However, we should not inform the human driver of the robot car's future motion when evaluating the consequence of the action $do(\mathbf{x}_{1,1:T_H} = \hat{\mathbf{x}}_{1,1:T_H})$. This leads to overly confident anticipation of the human's reaction to aggressive maneuvers, as demonstrated in our toy example. Instead, we should evaluate $\hat{\mathbf{x}}_{0,1:T_H}$ with a model approximating the distribution $p(\mathbf{x}_{0,1:T_H}|\hat{\mathbf{x}}_{0,0}, \hat{\mathbf{x}}_{1,0}, do(\hat{\mathbf{x}}_{1,1:T_H}))$, in other words, a model designed for the IBP task.

6.3 Quantifying the Impact of Conditional Inference on Real-World Datasets

Using the toy example, we have shown that conditional inference leads to biased predictions and a potential safety hazard. We are then curious about 1) how conditional inference may impact interactive prediction in real-world scenarios and 2) how we can identify a CBP model with potential safety risk. Unlike the toy example, we do not have access to the ground-truth dynamics governing the interacting agents. However, it is expensive and dangerous to estimate and compare the conditional and interventional distributions via real-world experiments. Instead, we are interested in an evaluation method purely based on offline datasets. Intuitively, the direct consequence of treating the query trajectory as observation is that the robot may anticipate the interacting agents and react to its future actions in advance. Therefore, we propose detecting and quantifying the impact of conditional inference by examining how much the future segment of the query trajectory contributes to the prediction at prior time steps for a given CBP model. In particular, we adopt Shapley

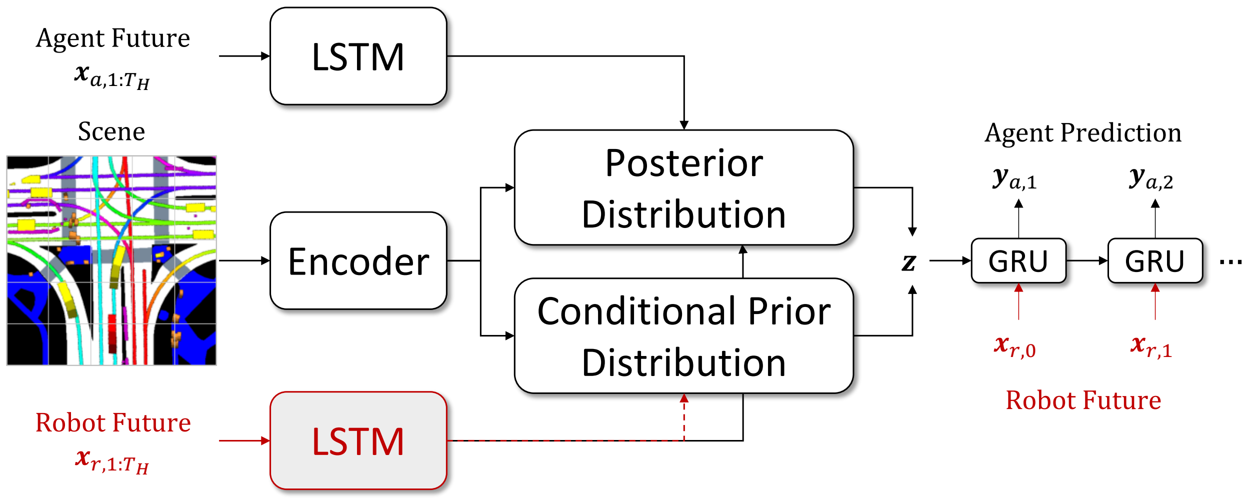


Figure 6.5: The conditional behavior prediction scheme with Trajectron++.

values as the evaluation tool.

6.3.1 Shapley Value in Explainable Deep Learning

Originating from cooperative game theory, Shapley values have been widely used in deep learning to quantify the feature attribution of black-box models [84]. Shapley values quantify the attribution of each dimension of an input $x = (x_1, \cdots, x_n)$ to a function describing the model behavior $f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \to \mathbb{R}$. The output of f could be the direct output of the model. Alternatively, f could also output a numerical value quantifying the performance or uncertainty of the model. Formally, one defines a set function $\nu : \mathbb{S} \to \mathbb{R}$ where $\mathbb{S}$ is the power set of $N := \{1, 2, \cdots, n\}$, i.e., $\mathbb{S} = P(N)$. For a subset $S \in \mathbb{S}$, the output $\nu(S)$ corresponds to running the model on a modified version of the input x for which features not in S are dropped or replaced. For instance, we may replace the dropped features $x_{N\setminus S}$ with samples drawn from their marginal distribution in the dataset [84] and then define the following:

$$\nu(S) = \mathbb{E}\left[f(x_S, X_{N\setminus S})\right]. \tag{6.1}$$

For each feature x_i, its Shapley value $\phi(x_i)$ is defined as:

$$\phi(x_i) = \sum_{S \subseteq N \setminus \{i\}} \frac{1}{n\binom{n-1}{|S|}} \left(\nu\left(S \cup \{i\}\right) - \nu\left(S\right)\right), \tag{6.2}$$

i.e., the difference in ν between including and not including the feature x_i averaged over all subsets S. In the context of trajectory prediction, Shapley values have been adapted to quantify the usage of social cues of prediction models [85].

6.3.2 Shapley Value for Interventional Behavior Prediction

In our case, we define the Shapley values of features based on their attribution to prediction performance. We adopt three evaluation metrics:

- *Average Displacement Error (ADE)*: Mean L_2 distance between the ground-truth and predicted trajectories averaged over K samples.
- *Final Displacement Error (FDE)*: L_2 distance between the ground-truth and predicted final position averaged over K samples.
- *Kernel Density Estimate-based Negative Log Likelihood (KDE NLL)*: Mean NLL of the ground-truth trajectory under a distribution created by fitting a kernel density estimate on K trajectory samples [109].

It is worth noting that we do not follow the common practice of evaluating the minimum distance metrics over sampled trajectories. As argued in [85], computing the minimum leads to biased estimation. To minimize unnecessary bias in the computation of $\nu\left(S \cup \{i\}\right) - \nu\left(S\right)$, we choose a sufficiently large K and only consider the average values of the distance metrics.

The features of interest are essentially the states of the planned trajectory. The model is evaluated with the ground-truth future trajectory of the robot car, denoted by $\mathbf{x}_{r,1:T_H}$. Since additional future information is granted, we expect more accurate prediction on average after conditioning on the ground-truth robot future, which leads to non-negative Shapley values in general. The question that remains is how to segment the robot future trajectory into features. Since the entire trajectory is typically treated as a sequence when encoded [109], perturbing the state at a single time step may have a minimal effect on the model output, as the encoder might manage to smooth out the perturbation. In addition, treating the state at each time step as a single feature leads to large n, which makes the computation of Shapley values expensive, as $|P(N)|$ grows exponentially with n. Instead, we split $\hat{\mathbf{x}}_{r,1:T_H}$ into m segments, with each segment consisting of states from multiple neighboring time steps:

$$\mathbf{x}_{r,1:T_H} = \left[\mathbf{x}_{r,1:t_1},\ \mathbf{x}_{r,t_1:t_2},\ \cdots,\ \mathbf{x}_{r,t_{m-1}:t_m}\right]. \tag{6.3}$$

We are then interested in evaluating the attribution of future segments to the prediction at earlier time steps. To this end, we compute the Shapley values $\phi(\mathbf{x}_{r,t_j:t_{j+1}})$ regarding the prediction at the first t_1 time steps. If the prediction model approximates the interventional distribution, we expect a large value for $\phi(\mathbf{x}_{r,1:t_1})$ but a nearly zero value for the latter segments. If any of the Shapley values for $\mathbf{x}_{r,t_j:t_{j+1}}$ with $j \geqslant 1$ are significant, it indicates that the model learns a distribution with a notable discrepancy related to the interventional distribution, which may cause a safety issue if deployed in on-road autonomous vehicles.

6.4 Experiments

With the proposed toolkit, we now study the impact of conditional inference for a state-of-the-art model on a real-world prediction dataset.

6.4.1 Model and Dataset

We conduct our experiments with a state-of-the art trajectory prediction model, Trajectron++ [109], on the nuScenes dataset [14]. We choose Trajectron++ because it supports conditional trajectory prediction. More importantly, the authors showed that conditioning Trajectron++ on the future trajectory of the ego agent—referred to as *robot future* in [109]—indeed improved the model's prediction performance on the nuScenes dataset.

Trajectron++ leverages the CVAE [117] framework to explicitly model the multi-modality in trajectory distribution. As shown in Fig. 6.5, the robot future trajectory is fed into Trajectron++ through three channels: 1) feeding step-by-step into the corresponding GRU cell [21] of the trajectory decoder; 2) feeding into the encoder modeling the posterior distribution of latent variables after encoded by a LSTM network [52]; 3) feeding into the encoder modeling the conditional prior distribution of latent variables after encoded by the same LSTM network.

The first two channels are not problematic. The computational graph of the first channel is consistent with the causal Bayesian network after intervention, as shown in Fig. 6.4. For the second channel, the posterior distribution is only used during training. However, the last channel is potentially defective. During the inference stage, the latent variables are sampled from the conditional prior distribution, which takes the embedding generated by the LSTM network as input. The embedding fuses information along the entire planning horizon. Consequently, the model has access to the robot future states at later time steps when predicting the target agent's states at former time steps.

We are curious how much performance gain is attributed to this faulty shortcut. Accordingly, we use the Shapley values proposed in Sec. 6.3.2 to analyze the model behavior on the nuScenes dataset. The nuScenes benchmark sets a prediction horizon of 6s. We then split the robot future trajectory into three equal segments. Then, we compute the Shapley values to quantify their attribution to the prediction performance within the first 2s in the future. In addition, we compare Trajectron++ with a variant created by masking out the third channel (i.e., the dashed line in Fig. 6.5). By comparing the prediction performance of these two models, we can see the effect of this shortcut on the model behavior.

To compute the Shapley values, we need to define the set function in Eqn. (6.1), which requires a marginal distribution of dropped features to define the expectation. Since we do not have access to the ground-truth distribution of the dataset, we train an unconditioned Trajectron++ model as an approximation. When computing the Shapley values for a given data sample, we sample trajectories from the trained Trajectron++ model's prediction of the robot car. The resulting Shapley values reveal the characteristics of the prediction model when it is queried by a motion planner imitating human behavior. Alternatively, we may estimate the expectation with the exact motion planner that will be deployed for a customized analysis.

Table 6.1: Shapley Values Comparison

Mask	ϕ_1^{ADE}	ϕ_2^{ADE}	ϕ_3^{ADE}
-	0.0148 ± 0.0839	0.0049 ± 0.0444	0.0044 ± 0.0376
✓	0.0053 ± 0.0197	0.0000 ± 0.0024	0.0000 ± 0.0024
Mask	ϕ_1^{FDE}	ϕ_2^{FDE}	ϕ_3^{FDE}
-	0.0332 ± 0.1569	0.0117 ± 0.0829	0.0109 ± 0.0716
✓	0.0156 ± 0.0568	0.0000 ± 0.0045	0.0000 ± 0.0045
Mask	ϕ_1^{KDE}	ϕ_2^{KDE}	ϕ_3^{KDE}
-	0.0636 ± 0.3365	0.0192 ± 0.1725	0.0179 ± 0.1676
✓	0.0119 ± 0.0857	0.0000 ± 0.0527	0.0001 ± 0.0524

The results are presented in the format of mean $\pm$ std.

Table 6.2: Prediction Performance Comparison

Ablation		$min\text{ADE}_{K=6}$	$min\text{FDE}_{K=6}$	KDE NLL
Robot	Mask			
-	-	1.73 ± 2.32	4.02 ± 5.62	1.86 ± 3.23
✓	-	$\mathbf{1.61 \pm 2.44}$ $(\mathbf{-6.94}\%)$	3.76 ± 5.99 (-6.47%)	$\mathbf{1.61 \pm 3.73}$ $(\mathbf{-13.4}\%)$
✓	✓	$\mathbf{1.61 \pm 2.43}$ $(\mathbf{-6.94}\%)$	$\mathbf{3.72 \pm 5.92}$ $(\mathbf{-7.46}\%)$	1.77 ± 3.43 (-4.84%)

The results are presented in the format of mean $\pm$ std. The number in the parentheses indicates the percentage of improvement over the unconditioned model.

6.4.2 Results

We computed the Shapley values over the test set for the models with and without masking the channel feeding the robot future embedding into the conditional prior encoder. The results are presented in Table 6.1 and Fig. 6.6, where we summarize the statistics of ϕ_j^{ADE}, ϕ_j^{FDE}, and ϕ_j^{KDE} for $j = 1, 2, 3$. The superscript denotes the corresponding performance metric. The subscript denotes the segment of the robot future trajectory. By masking the input channel, the model satisfies the temporal independence inherited in the causal Bayesian network after intervention. As expected, the Shapley values are minimal for $j > 1$. However, we can see from Fig. 6.6 that the values are not strictly zero for all the data samples due to the randomness in model output. In contrast, the model without masking, i.e., the original Trajectron++ model, has significantly larger Shapley values for the latter two segments. More importantly, their magnitude is not negligible compared to the Shapley value of the first segment. This means that the future states of the robot can falsely affect the model's prediction at earlier time steps.

In summary, the Shapley values suggest that the CBP model is indeed biased and could potentially result in a safety hazard after deployment. However, it is difficult to precisely measure these consequences without online testing. Even during online testing, the effect of biased prediction could only be observed in highly interactive scenarios, which only comprise a small proportion of real-world traffic scenarios. Therefore, we argue that a cheaper solution is to prevent the bias in the design stage. Instead of developing models for the CBP task, we should turn to the IBP task. The model should be carefully designed and implemented to follow an interventional distribution. Further, the proposed Shapley values should be used to monitor the model behavior.

A prediction benchmark designed for the IBP task should include such a quantitative metric as a complement to the current prediction metrics. For instance, we may set constraints $\phi_2^{\text{FDE}} \leqslant \epsilon$ and $\phi_3^{\text{FDE}} \leqslant \epsilon$ for some small threshold value ϵ. Only models satisfying the constraints are qualified for performance comparison. Such constraints are crucial for prediction benchmarking because the models are evaluated as black boxes. In general, it is expensive and time-consuming to check whether leakage occurs in the model design and implementation. Without the constraints, the performance comparison could be misleading and unfair. For instance, we compare the performance of the models with and without masking against the unconditioned model in Table 6.2. While the masking only slightly affects the values of *min*ADE and *min*FDE, the model without masking shows significant improvement in terms of KDE NLL. Without the Shapley values, one may consider this model with the defective input channel a better prediction model.

6.5 Discussion

6.5.1 Training Prediction Model for IBP

In our experiments, we demonstrated one practical way to design a prediction model for the IBP task. As with a CBP model, the model takes the ego agent's future trajectory as input during training. However, we should ensure that the model architecture reserves the structure of the causal Bayesian network under intervention (e.g., Fig. 6.4(b)). Alternatively, we may train a prediction model for the joint behavior of all the agents, including the ego agent and its surroundings. For online usage, we can then conduct an intervention on this joint prediction model when given a planned trajectory. In fact, some prior works follow this scheme implicitly for conditional prediction [111, 125, 61]. However, they seek to approximate the conditional distribution by enforcing the ego agent's action sequence, as it is intractable to perform exact conditional inference on the joint prediction model. We are interested in formally comparing these two training schemes in the future.

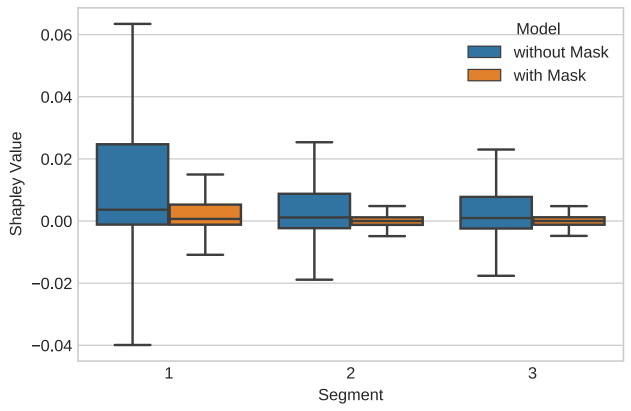
6.5.2 Establishing Prediction Benchmark for IBP

To compute the Shapley values in our experiments, we sample the ego agent's future trajectories from an unconditioned Trajectron++ model. However, the sample distribution is sensitive to the training dataset. Moreoer, if we want to establish a formal IBP benchmark, we cannot ensure a transparent and fair evaluation using a black-box sampling method for Shapley value computation. As a solution, we may evaluate the Shapley values with a set of plausible future trajectories generated by a model-based motion planner [118]. We will investigate this possibility and develop IBP benchmarks on public datasets in our future work.

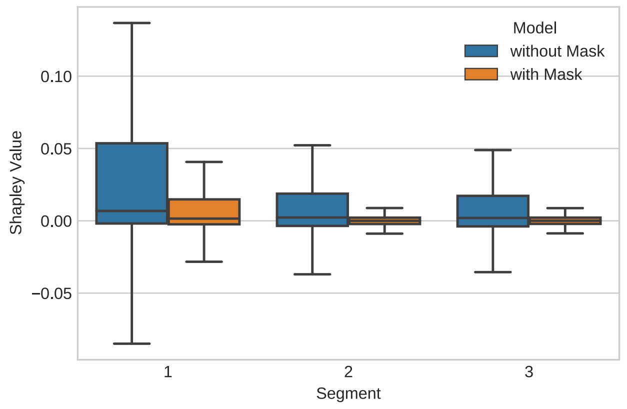
Further, we would like to emphasize that ensuring temporal dependence is only the basic requirement for a good IBP model. Since a planner may query an IBP model with an arbitrary planned trajectory, the IBP model should ideally be accurate over the entire input space of the planned trajectories. However, it is prohibitive in general to train such a perfect model with offline datasets. Instead, a practical solution is to equip the prediction model with a module detecting out-of-distribution (OOD) inputs of planned trajectories [29, 122], which can be utilized to prevent the planning module from exploiting the prediction model with those OOD inputs. Therefore, it is crucial to require such an OOD module for an IBP model and include the evaluation of OOD detection as part of an IBP benchmark.

6.6 Chapter Summary

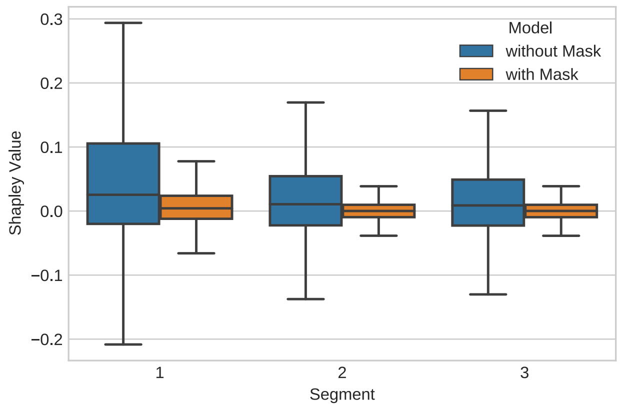
In this chapter, we study the problem of conditional behavior prediction, which builds the foundation for an interactive prediction and planning framework. We argue that it is risky for the planner to query a prediction model trained for a CBP task. Instead, we should treat the planned trajectory as an intervention and let the model learn the trajectory distribution under intervention, which we refer to as an IBP task. To distinguish between a CBP model and an IBP model, we propose a Shapley-value-based metric to verify if the prediction model satisfies the inherent temporal independence of an interventional distribution. With the proposed metric, we show that the CBP model trained on a real-world dataset violates temporal independence, leading to potential safety hazards if queried by planning modules. Additionally, the proposed metric enables a proper evaluation scheme with offline datasets for the IBP task. Thresholds can be set on the Shapley values to ensure a fair evaluation. In the future, we will utilize the proposed metrics to establish IBP benchmarks on public datasets.



(a) ϕ_j^{ADE} for $j = 1, 2, 3$.



(b) ϕ_j^{FDE} for $j = 1, 2, 3$.



(c) ϕ_j^{KDE} for $j = 1, 2, 3$.

Figure 6.6: Box plots of Shapley values for different performance metrics. We compare the Shapley values of different segments of the robot future for the two models.

Chapter 7

Final Words

With the recent developments in autonomous driving, we are heading towards an intelligent driver-less transportation system. However, the behavior of the current system pipeline with black-box modules cannot be well understood, which makes it difficult for the public to embrace and trust in the emerging autonomous driving technologies. Consequently, it is crucial to build explainable autonomous driving systems so that humans can understand and anticipate the decisions made by the AVs with which they share the road. In this dissertation, we investigated explainable autonomous driving techniques from the perspective of a model designer. We explored methods to improve model interpretability in the design stage by incorporating domain knowledge. More specifically, in Chapters 2 and 3, we showed how to formulate domain knowledge of social interaction into structured reward functions and pseudo labels and utilize them to induce interpretable behavior models in a principled manner. In Chapter 4, we demonstrated how to utilize knowledge of vehicle dynamics to develop an interpretable and transferable hierarchical driving policy. Altogether, the methods introduced in these chapters improve the interpretability of the behavior system of an AV, enabling trustworthy interaction between humans and AVs. While we argued that improving model interpretability should be our primary design objective, we also demonstrated the value of post hoc explanations through two case studies in which post hoc explanation techniques were utilized to diagnose behavior prediction models. In Chapter 5, we diagnosed the social posterior collapse issue of VAE-based interaction models with the help of a novel sparse graph attention mechanism. In Chapter 6, we diagnosed the causality issue of conditional behavior prediction with the help of the proposed Shapley-value-based metric.

The work we presented in this dissertation is a step toward developing transparent autonomous driving systems that can be verified by the designers and trusted by the end users. There is still a large gap between a fully transparent and trustful system and the system in its current form. It remains challenging to bridge this gap in model transparency while maintaining the same level of performance. We believe there are several future research directions that could help us reach this ultimate goal.

Fully Interpretable Model

While the techniques proposed in this dissertation can improve interpretability, they mainly rely on introducing interpretable intermediate representations. The backbone model remains a neural network with complex behavior. A fully interpretable replacement could offer greater transparency. While deep learning methods have shown superior performance in different tasks, some recent works [105, 78] have indicated that interpretable models could achieve the same level of performance in some problem domains. In particular, it was shown that RL polices could be synthesized with interpretable models, such as decision trees [116] or symbolic programs [137], while still achieving good performance and transferability. Although current methods are limited to relatively simple tasks, continued investigations are promising and could eventually lead to the construction of a fully interpretable model capable of solving complicated real-world driving tasks.

Causal Inference

ML has largely focused on learning statistical associations between variables. However, recently it has been argued [99, 112] that merely learning statistical correlations is not sufficient to create models that can generalize effectively under distributional shifts or transfer to new problems. This requires learning the underlying *causal model* of a problem. This dissertation touches briefly on the subject of causality. The discrepancy in CBP and IBP elaborated in Chapter 6 is an example from the autonomous driving domain showing that ignoring causal factors may lead to unreliable and risky predictions. Although not formally stated, the social posterior collapse problem introduced in Chapter 5 also occurs because the training process of VAEs encourages learning correlations but not the underlying causal relations of the data. Apart from generalization ability, capturing the causal structure is also essential for building an interpretable and trustworthy model. Humans' thinking process is built upon causal reasoning. Hence, humans are better able to understand an ML model that shares the same reasoning structure. However, it is challenging to discover causal relations from observational data in general. Alternatively, we may utilize the specific characteristics of the problem domains to help the models capture the causal structure in a domain-specific manner. For instance, we may directly impose certain causality constraints [106] based on domain knowledge. It is also possible to learn from interventional data obtained from either online interventions or human expert labeling [47] in some applications. We are interested in exploring these directions to develop autonomous driving systems that can genuinely *think* like humans and thus be trusted by the public.

Comprehensive Sensitivity Analysis and Benchmark

The last two research directions mainly focus on creating models with better interpretability. While we are awaiting fully interpretable models capable of solving sophisticated real-world driving tasks, it is equally important and even more urgent to comprehensively understand the behavior and limitations of the current autonomous driving systems, which are increasing in complexity. Such principled studies are lacking in the literature. For example, prediction accuracy has been the single most important criterion when evaluating a behavior prediction model. However, as suggested by recent works [85] as well as by our work [127, 128] presented in this dissertation, performance metrics alone are not sufficient to evaluate and understand model behavior. Some modules (e.g., social interaction encoders) may not work the same way as claimed [85, 127]. Moreover, there may be defects that cannot be identified based on performance metrics computed using offline data [128]. We have shown that model-agnostic feature attribution methods such as Shapley values are valuable complements to the current performance-orientated metrics for better understanding model behavior. Encouraged by the results, we are interested in conducting a comprehensive sensitivity analysis on state-of-art models for different tasks (e.g., perception, behavior prediction, motion planning). In doing so, we could verify whether those state-of-the-art models properly understand and utilize the semantic information inherited in the input features. We hope that these efforts could eventually lead to more comprehensive and insightful benchmarks, helping us better understand the limitations of current methods and inspiring improvements.

Human-in-the-loop Learning

An explainable autonomous system enables humans to better understand the system's decision-making process, which builds the foundation for trustworthy interaction. Further, a more transparent interface between the system and its end users allows closer collaboration with humans during the learning process. This is particularly beneficial in terms of infusing domain knowledge. While principled methods exist to formulate knowledge into constraints regularizing the learning process, some knowledge is rather vague and subjective. With a human-in-the-loop learning scheme, such knowledge can still be acquired by querying human experts on a case-by-case basis. Human feedback can occur during large-scale batch training or, more interestingly, during online operations. The system then has the capability of lifelong learning, and it can keep improving itself by learning from its users.

* * *

We hope the work presented in this dissertation will provide useful building blocks on the path toward transparent autonomous vehicles or, in a broader sense, autonomous systems that humans can genuinely trust. We are sincerely looking forward to a new era of mixed autonomy in which humans and AI agents share the social space harmoniously.

Bibliography

[1] Alexandre Alahi et al. "Social lstm: Human trajectory prediction in crowded spaces". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 961–971.

[2] Alexander A Alemi et al. "Deep variational information bottleneck". In: *International Conference on Learning Representations (ICLR)* (2017).

[3] 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.

[4] Shahin Atakishiyev et al. "Explainable Artificial Intelligence for Autonomous Driving: A Comprehensive Overview and Field Guide for Future Research Directions". In: *arXiv preprint arXiv:2112.11561* (2021).

[5] Rachid Attia, Rodolfo Orjuela, and Michel Basset. "Combined longitudinal and lateral control for automated vehicle guidance". In: *Vehicle System Dynamics* 52.2 (2014), pp. 261–279.

[6] *Autonomous vehicles*. Jan. 2022. URL: `https://www.dmv.ca.gov/portal/vehicle-industry-services/autonomous-vehicles/`.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[8] Peter Battaglia et al. "Interaction networks for learning about objects, relations and physics". In: *Advances in neural information processing systems (NeurIPS)*. 2016, pp. 4502–4510.

[9] Philipp Bender, Julius Ziegler, and Christoph Stiller. "Lanelets: Efficient map representation for autonomous driving". In: *IEEE Intelligent Vehicles Symposium (IV)*. 2014, pp. 420–425.

[10] Raunak P Bhattacharyya et al. "Multi-agent imitation learning for driving simulation". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1534–1539.

[11] Raunak P Bhattacharyya et al. "Simulating emergent properties of human driving behavior using multi-agent reward augmented imitation learning". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 789–795.

[12] Mariusz Bojarski et al. “Visualbackprop: Efficient visualization of cnns for autonomous driving”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–8.

[13] Samuel Bowman et al. “Generating Sentences from a Continuous Space”. In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. 2016, pp. 10–21.

[14] Holger Caesar et al. “nuScenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.

[15] Ming-Fang Chang et al. “Argoverse: 3d tracking and forecasting with rich maps”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8748–8757.

[16] Wen-Hua Chen. “Disturbance observer based control for nonlinear systems”. In: *IEEE/ASME transactions on mechatronics* 9.4 (2004), pp. 706–710.

[17] Wen-Hua Chen et al. “Disturbance-observer-based control and related methods—An overview”. In: *IEEE Transactions on Industrial Electronics* 63.2 (2016), pp. 1083–1095.

[18] Xi Chen et al. “Variational Lossy Autoencoder”. In: *5th International Conference on Learning Representations (ICLR)*. 2017.

[19] Xinkai Chen, Satoshi Komada, and Toshio Fukuda. “Design of a nonlinear disturbance observer”. In: *IEEE Transactions on Industrial Electronics* 47.2 (2000), pp. 429–437.

[20] Xu Chen and Masayoshi Tomizuka. “Control methodologies for precision positioning systems”. In: *American Control Conference (ACC), 2013*. IEEE. 2013, pp. 3704–3711.

[21] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Oct. 2014, pp. 1724–1734.

[22] Paul Christiano et al. “Transfer from simulation to real world through learning deep inverse dynamics model”. In: *arXiv preprint arXiv:1610.03518* (2016).

[23] Gonçalo Correia et al. “Efficient marginalization of discrete and structured latent variables via sparsity”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11789–11802.

[24] Nachiket Deo and Mohan M Trivedi. “Trajectory forecasts in unknown environments conditioned on grid-based plans”. In: *arXiv preprint arXiv:2001.00735* (2020).

[25] Scott Ettinger et al. “Large Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 9710–9719.

[26] Paolo Falcone et al. “A hierarchical model predictive control framework for autonomous ground vehicles”. In: *2008 American Control Conference*. IEEE. 2008, pp. 3719–3724.

[27] Paolo Falcone et al. “Predictive active steering control for autonomous vehicle systems”. In: *IEEE Transactions on control systems technology* 15.3 (2007), pp. 566–580.

[28] Matthias Fey and Jan E. Lenssen. “Fast graph representation learning with PyTorch Geometric”. In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

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

[30] Chelsea Finn, Sergey Levine, and Pieter Abbeel. “Guided cost learning: Deep inverse optimal control via policy optimization”. In: *International Conference on Machine Learning (ICML)*. 2016, pp. 49–58.

[31] Chelsea Finn et al. “A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models”. In: *arXiv preprint arXiv:1611.03852* (2016).

[32] Marco Fraccaro et al. “Sequential neural models with stochastic layers”. In: *Advances in neural information processing systems (NeurIPS)*. Vol. 29. 2016.

[33] Hao Fu et al. “Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 240–250.

[34] Justin Fu, Sergey Levine, and Pieter Abbeel. “One-shot learning of manipulation skills with online dynamics adaptation and neural network priors”. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 4019–4026.

[35] Justin Fu, Katie Luo, and Sergey Levine. “Learning Robust Rewards with Adverserial Inverse Reinforcement Learning”. In: *International Conference on Learning Representations (ICLR)*. 2018.

[36] Jiyang Gao et al. “VectorNet: Encoding HD maps and agent dynamics from vectorized representation”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 11522–11530.

[37] Justin Gilmer et al. “Neural message passing for quantum chemistry”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)-Volume 70*. JMLR. 2017, pp. 1263–1272.

[38] Francesco Giuliari et al. "Transformer networks for trajectory forecasting". In: *International Conference on Pattern Recognition (ICPR)*. 2021, pp. 10335–10342.

[39] Anirudh Goyal et al. "Recurrent Independent Mechanisms". In: *International Conference on Learning Representations (ICLR)*. 2021.

[40] Michael Grant and Stephen Boyd. *CVX: Matlab Software for Disciplined Convex Programming, version 2.1*. `http://cvxr.com/cvx`. Mar. 2014.

[41] Michael Grant and Stephen Boyd. "Graph implementations for nonsmooth convex programs". In: *Recent Advances in Learning and Control*. Ed. by V. Blondel, S. Boyd, and H. Kimura. Lecture Notes in Control and Information Sciences. Springer-Verlag Limited, 2008, pp. 95–110.

[42] Junru Gu, Chen Sun, and Hang Zhao. "Densetnt: End-to-end trajectory prediction from dense goal sets". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 15303–15312.

[43] Junru Gu, Qiao Sun, and Hang Zhao. "DenseTNT: Waymo Open Dataset Motion Prediction Challenge 1st Place Solution". In: *arXiv preprint arXiv:2106.14160* (2021).

[44] Jürgen Guldner, Han-Shue Tan, and Satyajit Patwardhan. "Analysis of automatic steering control for highway vehicles with look-down lateral reference systems". In: *Vehicle System Dynamics* 26.4 (1996), pp. 243–269.

[45] David Gunning et al. "XAI—Explainable artificial intelligence". In: *Science Robotics* 4.37 (2019), eaay7120.

[46] Agrim Gupta et al. "Social GAN: Socially acceptable trajectories with generative adversarial networks". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[47] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. "Causal confusion in imitation learning". In: *Advances in neural information processing systems (NeurIPS)*. 2019, pp. 11698–11709.

[48] Jingqing Han. "From PID to active disturbance rejection control". In: *IEEE transactions on Industrial Electronics* 56.3 (2009), pp. 900–906.

[49] James Harrison et al. "Adapt: zero-shot adaptive policy transfer for stochastic dynamical systems". In: *Robotics research*. Springer, 2020, pp. 437–453.

[50] Irina Higgins et al. "beta-vae: Learning basic visual concepts with a constrained variational framework". In: *International Conference on Learning Representations (ICLR)* (2017).

[51] Jonathan Ho and Stefano Ermon. "Generative adversarial imitation learning". In: *Advances in neural information processing systems (NeurIPS)*. 2016, pp. 4565–4573.

[52] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[53] Yedid Hoshen. "Vain: Attentional multi-agent predictive modeling". In: *Advances in neural information processing systems (NeurIPS)*. 2017, pp. 2701–2711.

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

[55] Masha Itkina et al. "Evidential sparsification of multimodal latent spaces in conditional variational autoencoders". In: *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), pp. 10235–10246.

[56] Boris Ivanovic and Marco Pavone. "The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatio-temporal graphs". In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 2375–2384.

[57] Boris Ivanovic et al. "Multimodal deep generative models for trajectory prediction: A conditional variational autoencoder approach". In: *IEEE Robotics and Automation Letters (RA-L)* 6.2 (2020), pp. 295–302.

[58] Ashesh Jain et al. "Structural-rnn: Deep learning on spatio-temporal graphs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 5308–5317.

[59] Sarthak Jain and Byron C Wallace. "Attention is not explanation". In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 3543–3556.

[60] Arne Kesting, Martin Treiber, and Dirk Helbing. "Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 368.1928 (2010), pp. 4585–4605.

[61] Siddhesh Khandelwal et al. "What-if motion prediction for autonomous driving". In: *arXiv preprint arXiv:2008.10587* (2020).

[62] Jinkyu Kim and John Canny. "Interpretable learning for self-driving cars by visualizing causal attention". In: *Proceedings of the IEEE international conference on computer vision (ICCV)*. 2017, pp. 2942–2950.

[63] Jinkyu Kim et al. "Textual explanations for self-driving vehicles". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 563–578.

[64] Youngdong Kim et al. "Nlnl: Negative learning for noisy labels". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 101–110.

[65] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[66] Diederik P Kingma and Max Welling. "Auto-encoding variational Bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[67] Thomas Kipf et al. "Neural relational inference for interacting systems". In: *International Conference on Machine Learning (ICML)*. PMLR. 2018, pp. 2688–2697.

[68] Vineet Kosaraju et al. "Social-BiGAT: Multimodal trajectory forecasting using bicycle-GAN and graph attention networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[69] Donsuk Lee et al. "Joint Interaction and Trajectory Prediction for Autonomous Driving using Graph Neural Networks". In: *arXiv preprint arXiv:1912.07882* (2019).

[70] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. "Crowds by example". In: *Computer Graphics Forum*. Vol. 26. Wiley Online Library. 2007, pp. 655–664.

[71] Karen Leung, Nikos Aréchiga, and Marco Pavone. "Backpropagation for Parametric STL". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 185–192.

[72] Sergey Levine. "Reinforcement learning and control as probabilistic inference: Tutorial and review". In: *arXiv preprint arXiv:1805.00909* (2018).

[73] Jesse Levinson et al. "Towards fully autonomous driving: Systems and algorithms". In: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE. 2011, pp. 163–168.

[74] Jiachen Li et al. "Spatio-temporal graph dual-attention network for multi-agent prediction and tracking". In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[75] Ming Liang et al. "Learning lane graph representations for motion forecasting". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 12347 LNCS (2020), pp. 541–556. ISSN: 16113349. arXiv: `2007.13732`.

[76] Richard Liaw et al. "Tune: A research platform for distributed model selection and training". In: *arXiv preprint arXiv:1807.05118* (2018).

[77] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning." In: *International Conference on Learning Representations (ICLR)*. 2016.

[78] Jimmy Lin et al. "Generalized and scalable optimal sparse decision trees". In: *International Conference on Machine Learning (ICML)*. PMLR. 2020, pp. 6150–6160.

[79] Changliu Liu and Masayoshi Tomizuka. "Algorithmic safety measures for intelligent industrial co-robots". In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 3095–3102.

[80] Changliu Liu and Masayoshi Tomizuka. "Enabling safe freeway driving for automated vehicles". In: *American Control Conference (ACC), 2016*. IEEE. 2016, pp. 3461–3467.

[81] Changliu Liu et al. "Convex feasible set algorithm for constrained trajectory smoothing". In: *2017 American Control Conference (ACC)*. IEEE. 2017, pp. 4177–4182.

[82] Jerry Liu et al. "Deep structured reactive planning". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 4897–4904.

[83] Shuijing Liu et al. “Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 3517–3524.

[84] Scott M Lundberg and Su-In Lee. “A unified approach to interpreting model predictions”. In: *Advances in neural information processing systems (NeurIPS)*. Vol. 30. 2017.

[85] Osama Makansi et al. “You Mostly Walk Alone: Analyzing Feature Attribution in Trajectory Prediction”. In: *arXiv preprint arXiv:2110.05304* (2021).

[86] Karttikeya Mangalam et al. “It is not the journey but the destination: Endpoint conditioned trajectory prediction”. In: *European Conference on Computer Vision (ECCV)*. Springer. 2020, pp. 759–776.

[87] Andre Martins and Ramon Astudillo. “From softmax to sparsemax: A sparse model of attention and multi-label classification”. In: *International Conference on Machine Learning (ICML)*. PMLR. 2016, pp. 1614–1623.

[88] Michiel M Minderhoud and Piet HL Bovy. “Extended time-to-collision measures for road traffic safety assessment”. In: *Accident Analysis & Prevention* 33.1 (2001), pp. 89–97.

[89] Maximilian Naumann et al. “Analyzing the suitability of cost functions for explaining and imitating human driving behavior based on inverse reinforcement learning”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 5481–5487.

[90] Hans B Pacejka and Egbert Bakker. “The magic formula tyre model”. In: *Vehicle system dynamics* 21.S1 (1992), pp. 1–18.

[91] Brian Paden et al. “A survey of motion planning and control techniques for self-driving urban vehicles”. In: *IEEE Transactions on Intelligent Vehicles* 1.1 (2016), pp. 33–55.

[92] Adam Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *arXiv preprint arXiv:1912.01703* (2019).

[93] Judea Pearl. *Causality: Models, Reasoning and Inference*. 2nd. Cambridge University Press, 2009.

[94] Judea Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Elsevier, 2014.

[95] S. Pellegrini et al. “You’ll never walk alone: Modeling social behavior for multi-target tracking”. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2009, pp. 261–268.

[96] Xue Bin Peng et al. “Sim-to-real transfer of robotic control with dynamics randomization”. In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 3803–3810.

[97] Xue Bin Peng et al. “Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow”. In: *International Conference on Learning Representations (ICLR)* (2019).

[98] Ben Peters, Vlad Niculae, and André FT Martins. “Sparse Sequence-to-Sequence Models”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 1504–1519.

[99] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[100] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[101] Aravind Rajeswaran et al. “Epopt: Learning robust neural network policies using model ensembles”. In: *arXiv preprint arXiv:1610.01283* (2016).

[102] Christian Rathgeber et al. “Lateral trajectory tracking control for autonomous vehicles”. In: *Control Conference (ECC), 2014 European*. IEEE. 2014, pp. 1024–1029.

[103] 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 (ICCV)*. 2019, pp. 2821–2830.

[104] Cynthia Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215.

[105] Cynthia Rudin and David Carlson. “The secrets of machine learning: ten things you wish you had known earlier to be more effective at data analysis”. In: *Operations Research & Management Science in the Age of Analytics*. Informs, 2019, pp. 44–72.

[106] Cynthia Rudin et al. “Interpretable machine learning: Fundamental principles and 10 grand challenges”. In: *Statistics Surveys* 16 (2022), pp. 1–85.

[107] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd ed. Prentice Hall, 2010.

[108] Amir Sadeghian et al. “Sophie: An attentive GAN for predicting paths compliant to social and physical constraints”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 1349–1358.

[109] Tim Salzmann et al. “Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data”. In: *European Conference on Computer Vision (ECCV)*. Springer. 2020, pp. 683–700.

[110] Riccardo Scattolini and Patrizio Colaneri. “Hierarchical model predictive control”. In: *2007 46th IEEE Conference on Decision and Control (CDC)*. IEEE. 2007, pp. 4803–4808.

[111] Edward Schmerling et al. “Multimodal probabilistic model-based planning for human-robot interaction”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 3399–3406.

[112] Bernhard Schölkopf et al. “Toward causal representation learning”. In: *Proceedings of the IEEE* 109.5 (2021), pp. 612–634.

[113] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[114] Iulian Serban et al. “A hierarchical latent variable encoder-decoder model for generating dialogues”. In: *AAAI Conference on Artificial Intelligence*. 2017.

[115] Lloyd S. Shapley. *Notes on the N-Person Game - II: The Value of an N-Person Game.* Santa Monica, CA: RAND Corporation, 1951.

[116] Andrew Silva et al. “Optimization methods for interpretable differentiable decision trees applied to reinforcement learning”. In: *International conference on artificial intelligence and statistics (AISTATS)*. PMLR. 2020, pp. 1855–1865.

[117] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. “Learning structured output representation using deep conditional generative models”. In: *Advances in neural information processing systems (NeurIPS)*. Vol. 28. 2015.

[118] Haoran Song et al. “Learning to predict vehicle trajectories with model-based planning”. In: *Conference on Robot Learning (CoRL)*. PMLR. 2022, pp. 1035–1045.

[119] Haoran Song et al. “Pip: Planning-informed trajectory prediction for autonomous driving”. In: *European Conference on Computer Vision (ECCV)*. Springer. 2020, pp. 598–614.

[120] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. “Learning multiagent communication with backpropagation”. In: *Advances in neural information processing systems (NeurIPS)*. 2016, pp. 2244–2252.

[121] Lingfeng Sun et al. “Domain Knowledge Driven Pseudo Labels for Interpretable Goal-Conditioned Interactive Trajectory Prediction”. In: *arXiv preprint arXiv:2203.15112* (2022).

[122] Liting Sun, Xiaogang Jia, and Anca Dragan. “On complementing end-to-end human behavior predictors with planning”. In: *Proceedings of Robotics: Science and Systems (RSS)*. July 2021.

[123] 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.

[124] Simon Suo et al. “TrafficSim: Learning to Simulate Realistic Multi-Agent Behaviors”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 10400–10409.

[125] Charlie Tang and Russ R Salakhutdinov. “Multiple futures prediction”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 32. 2019.

[126] Chen Tang, Zhuo Xu, and Masayoshi Tomizuka. “Disturbance-observer-based tracking controller for neural network driving policy transfer”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.9 (2019), pp. 3961–3972.

[127] Chen Tang, Wei Zhan, and Masayoshi Tomizuka. “Exploring Social Posterior Collapse in Variational Autoencoder for Interaction Modeling”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 34. 2021.

[128] Chen Tang, Wei Zhan, and Masayoshi Tomizuka. “Interventional Behavior Prediction: Avoiding Overly Confident Anticipation in Interactive Prediction”. In: *arXiv preprint arXiv:2204.08665* (2022).

[129] Chen Tang et al. “Grounded relational inference: domain knowledge driven explainable autonomous driving”. In: *arXiv preprint arXiv:2102.11905* (2021).

[130] Ekaterina Tolstaya et al. “Identifying driver interactions via conditional behavior prediction”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 3473–3479.

[131] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. “Congested traffic states in empirical observations and microscopic simulations”. In: *Physical review E* 62.2 (2000), p. 1805.

[132] Martin Treiber and Arne Kesting. “Traffic flow dynamics”. In: *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg* (2013).

[133] Sjoerd Van Steenkiste et al. “Relational neural expectation maximization: Unsupervised discovery of objects and their interactions”. In: *International Conference on Learning Representations (ICLR)* (2018).

[134] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems (NeurIPS)*. 2017, pp. 5998–6008.

[135] Petar Veličković et al. “Graph attention networks”. In: *International Conference on Learning Representations (ICLR)* (2018).

[136] Anirudh Vemula, Katharina Muelling, and Jean Oh. “Social attention: Modeling attention in human crowds”. In: *2018 IEEE international Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–7.

[137] Abhinav Verma et al. “Programmatically interpretable reinforcement learning”. In: *International Conference on Machine Learning (ICML)*. PMLR. 2018, pp. 5045–5054.

[138] Ziyu Wang et al. “Robust imitation of diverse behaviors”. In: *Advances in neural information processing systems (NeurIPS)*. 2017, pp. 5320–5329.

[139] Sarah Wiegreffe and Yuval Pinter. “Attention is not not explanation”. In: *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 11–20.

[140] Zhuo Xu, Chen Tang, and Masayoshi Tomizuka. "Zero-shot Deep Reinforcement Learning Driving Policy Transfer for Autonomous Vehicles based on Robust Control". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 2865–2871.

[141] Zhuo Xu et al. "Toward Modularization of Neural Network Autonomous Driving Policy Using Parallel Attribute Networks". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 1400–1407.

[142] Maosheng Ye, Tongyi Cao, and Qifeng Chen. "Tpcn: Temporal point cloud networks for motion forecasting". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 11318–11327.

[143] Bing Yu, Haoteng Yin, and Zhanxing Zhu. "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2018, pp. 3634–3640.

[144] Cunjun Yu et al. "Spatio-temporal graph transformer networks for pedestrian trajectory prediction". In: *European Conference on Computer Vision (ECCV)*. Springer. 2020, pp. 507–523.

[145] Lantao Yu, Jiaming Song, and Stefano Ermon. "Multi-Agent adversarial inverse reinforcement learning". In: *International Conference on Learning Representations (ICLR)* (2019).

[146] Lantao Yu et al. "Meta-inverse reinforcement learning with probabilistic context variables". In: *Advances in neural information processing systems (NeurIPS)*. 2019, pp. 11772–11783.

[147] Wenhao Yu et al. "Preparing for the Unknown: Learning a Universal Policy with Online System Identification". In: *Robotics: Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 12-16, 2017*. Ed. by Nancy M. Amato et al. 2017.

[148] Ye Yuan et al. "AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 9813–9823.

[149] Wenyuan Zeng et al. "Lanercnn: Distributed representations for graph-centric motion forecasting". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 532–539.

[150] Wei Zhan et al. "INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps". In: *arXiv:1910.03088 [cs, eess]* (Sept. 2019).

[151] Hang Zhao et al. "TNT: Target-driven Trajectory Prediction". In: *Conference on Robot Learning (CoRL)*. PMLR. 2021, pp. 895–904.

[152] Brian D Ziebart et al. “Maximum entropy inverse reinforcement learning”. In: *Proceedings of AAAI Conference on Artificial Intelligence*. 2008.