Towards Understanding Human Behavior for Autonomous Driving and Robotics

by

Hengbo Ma

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Spring 2023

Towards Understanding Human Behavior for Autonomous Driving and Robotics

Abstract

Towards Understanding Human Behavior for Autonomous Driving and Robotics

by

Hengbo Ma

Doctor of Philosophy in Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

Understanding human behavior plays a significant role in many industrial applications, such as autonomous driving and robotics, that entail human participation. Having an efficacious human behavior model facilitates intelligent agents in arriving at high-quality decisions while aligning with human values. In the past decades, while we have witnessed the remarkable progress of human behavior modeling with modern deep learning approaches, it remains a challenging task due to the inherent stochasticity of human behavior and the intricate structural design of deep learning models.

This dissertation focuses on *learning-based human behavior prediction and generation for autonomous driving and robotics from both the model and learning algorithm design perspective.* In particular, the dissertation is concerned with two manifestations of human behaviors: human driving behavior and 3D human motion. The dissertation is divided into two parts. Part I focuses on improving the generalization of human driving behavior prediction problems in different problem settings. Chapter 2 introduces a transferable human driving behavior prediction model leveraging domain-knowledge-based representation. In Chapter 3, a neural memory system based on generative models is proposed to enable the capability of continual learning for multi-agent vehicle trajectory prediction. In Chapter 4, an uncertainty analysis of trajectory prediction in terms of static and dynamic information is provided, and a static-information-only-based approach for difficult instance discovery is proposed to improve the adaptability of the prediction model. Part II focuses on two properties of human behavior generation: diversity and feasibility. Chapter 5 introduces a novel generative model exploring diverse 3D human motion behavior and enables testing-time adjustment of prediction and generation. Chapter 6 introduces a differentiable safety-critical control framework to ensure the feasibility and generalizability of generated motions, and we demonstrate the efficiency of the proposed approach in the collision avoidance application.

To my family and friends

Contents

List of Figures

List of Tables

Acknowledgments

Over the past years, I have embarked on an incredible and unforgettable adventure that has led me to where I am today. I would not have been able to achieve my milestones today without the tremendous and generous support and guidance from many people.

First and foremost, I extend my deepest and most sincere gratitude to my Ph.D. advisor, Professor Masayoshi Tomizuka, who has been a truly exceptional mentor during my Ph.D. study. His profound knowledge, insightful perspectives, enthusiasm for work, and positive attitude toward life, have been invaluable to me. Without his guidance and persistent help, this work would not have been possible. Besides academic guidance, the beautiful memories of the countless happy moments spent with Professor Tomizuka have imprinted themselves deeply within me, leaving me with an indescribable feeling of gratitude and appreciation.

I am deeply grateful to Professor Mark Mueller and Professor Trevor Darrell for serving on my dissertation committee. I am also thankful to Professor Kameshwar Poolla, Professor Anil Aswani, and Professor Somayeh Sojoudi for serving on my qualifying exam committee. They provided excellent support for my significant milestones in my Ph.D. study.

My time as a Mechanical System Control (MSC) laboratory member has been a wonderful experience. I am sincerely thankful for my major collaborators at MSC Lab: Dr. Wei Zhan, Dr. Liting Sun, Dr. Jiachen Li, Dr. Chen Tang, Dr. Yeping Hu, and Lingfeng Sun. I would also like to express my thanks to my current and former MSC Lab colleagues: Prof. Changliu Liu, Prof. Jianyu Chen, Dr. Cheng Peng, Dr. Yongxiang Fan, Dr. Daisuke Kaneishi, Dr. Shuyang Li, Dr. Kiwoo Shin, Dr. Zining Wang, Dr. Zhuo Xu, Dr. Yujiao Cheng, Dr. Shiyu Jin, Dr. Jessica Leu, Changhao Wang, Xinghao Zhu, Yiyang Zhou, Huidong Gao, Zheng Wu, Jinning Li, Catherine Weaver, Wu-Te Yang, Xiang Zhang, Chengfeng Xu, Chenran Li, Akio Kodaira, Ran Tian, Ce Hao, Wei-Jer Chang, Jen-Wei Wang, and Yichen Xie, for their help during my graduate study and their efforts to maintain a good research atmosphere in the group. Meanwhile, I would like to thank other professors and students I have worked with: Prof. Koushil Sreenath, Bike Zhang, Mingyu Ding, Yaofeng Sun, Zhaoheng Yin, Zhihao Zhang, and Jiaqi Han. I enjoyed working with all these people very much.

Great thanks also go to all my sponsors and external collaborators in the industry during my Ph.D. study. In particular, I would like to thank Dr. Chiho Choi and all the members of Honda Research Insitute, USA. Without their valuable discussion and mentorship, I will not go this far. I am also thankful to Prof. Chuang Gan and all the members associated with MIT-IBM Watson AI Lab for their help, inspiration, and encouragement for my research.

Last but not least, I give my deepest love to my family. I want to thank my parents for their endless support. I also would like to my deep love and appreciation to Pengbo Zhu and her family for their understanding and unconditional support during my Ph.D. study.

Chapter 1

Introduction

1.1 Understanding Human Behavior: An Overview

The present era is witnessing a rapid development of intelligent agents that interact with humans and there is no doubt that such intelligent agents will become an integral part of human life. For instance, several companies, such as Waymo, Cruise, etc., have already deployed autonomous vehicles in local regions; Mobile robots are deployed in hotels and restaurants; Also, people have recently become enthusiastic about developing humanoid robots to support human well-being with daily jobs like housework and healthcare. In all of the aforementioned scenarios, The keyword is "human". Therefore, understanding human behaviors is crucial for enabling intelligent agents to provide effective, reliable, and valuable support. For instance, an autonomous vehicle should be able to perceive and predict how other traffic participants will behave in order to keep passengers safe and maintain the efficiency of the autonomous driving system simultaneously; Another example is that household robots should anticipate the motion and intention of their owners to decide how they can provide more accurate services accordingly.

In general, human behavior is defined as the potential and expressed capacity to react to internal or external stimuli in their environment [172]. Human behavior can be observed and analyzed through various lenses, including verbal communication, body language, facial expressions, and physical movements. In the context of autonomous driving, human behavior can also be expressed through the trajectories of vehicles, which is referred to as driving behavior. In this dissertation, our primary focus is modeling human behavior expressed through physical movements at the human body level and vehicle maneuvers since they are the major manifestations of human behavior in autonomous driving and robotics.

Technically, understanding human behavior is an extremely challenging task. In contrast to mechanical systems, which are typically deterministic and have known models, the mechanism underlying human cognitive processes - which drives human behavior - remains largely a mystery and exhibits a high degree of stochasticity. Hence, one of the suitable formulations is to represent human behavior as a probability distribution. With this formulation,

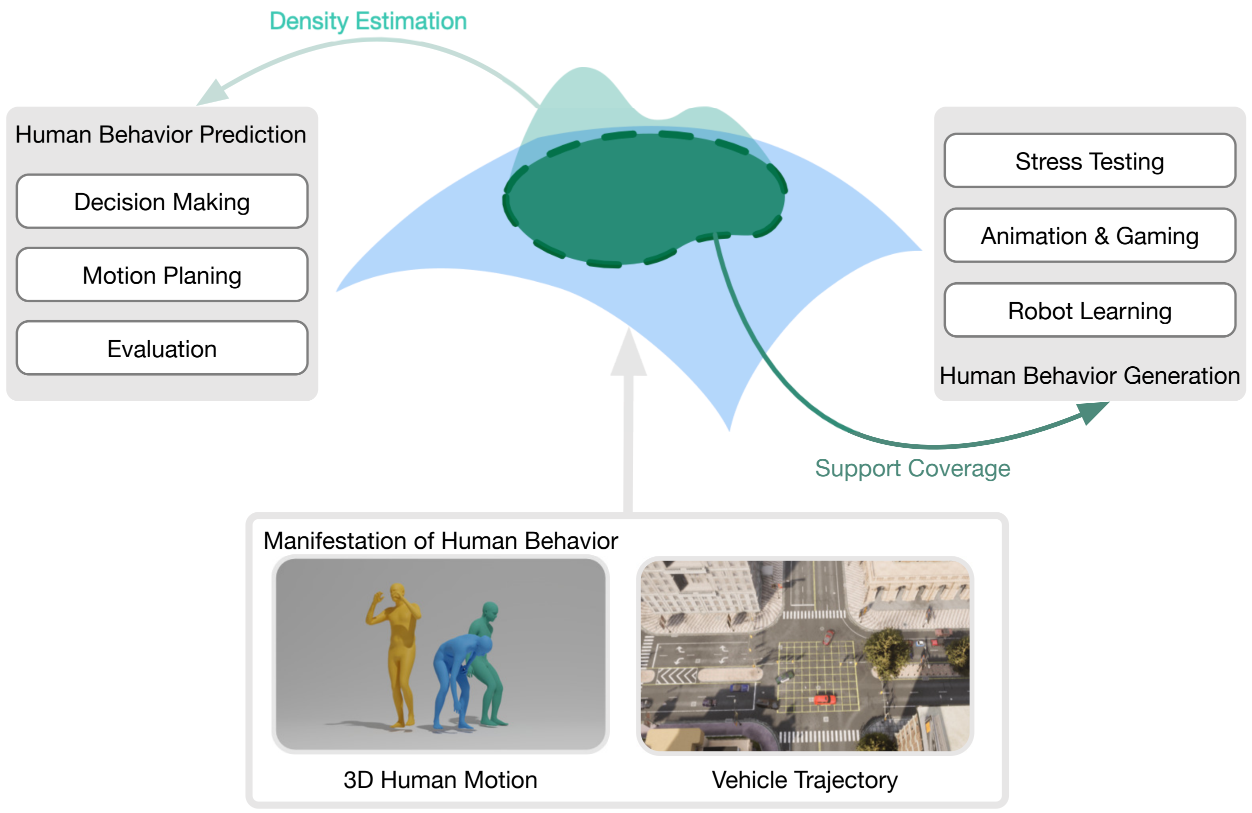


Figure 1.1: Human behavior prediction and generation. The relation between these two tasks is illustrated from a learning perspective. Human behavior prediction focuses more on the accuracy of the probability density estimation. Human behavior generation focuses more on the coverage of the probability distribution support. The typical applications for each task are enumerated in the grey box. The images in the block "manifestations of human behavior" are generated using SMPL-X [124] and CARLA [48].

this dissertation concerns two major tasks, human behavior prediction and human behavior generation, for autonomous driving and robotics. Although these two tasks are highly correlated and sometimes not clearly distinguished, we would like to differentiate these two tasks in the following sections to highlight the desirable properties that we focus on in this dissertation for each task. The relation between human behavior prediction and generation is illustrated in Figure 1.1.

1.1.1 Human Behavior Prediction

There are several applications of human behavior prediction in autonomous driving and robotics. For instance, a traffic participant prediction model can provide the future possible dangerous spatial-temporal region to one autonomous vehicle in order to enable it to make correct decisions. Meanwhile, human behavior prediction models can also serve as surrogate behavior models in a simulation. Such simulation can evaluate the performance of intelligent agents. A good prediction model should have several important properties: accuracy, generalizability, robustness, trustworthiness, etc. In this dissertation, we will mainly focus on the first two properties:

- **Accuracy** The human behavior model's accuracy significantly impacts downstream tasks such as decision making or planning, especially in safety-critical scenarios. One of the challenges of obtaining accurate human behavior distribution estimation is multi-modality. In particular, given limited observation at an intersection, one vehicle may have different routing options, such as going straight, turning left, or stopping. Moreover, even under the same route, the vehicle's speed profile may also vary significantly. In this situation, having a probabilistic model which can capture the accurate "shape" of the distribution of human behavior will make the planning of autonomous vehicles more robust and efficient.
- **Generalizability** Generalization refers to the ability of the prediction model to have a good performance on novel and unseen scenarios which are not provided in the training data. There are serval problem settings to test the generalizability.
  - The target domain (evaluation data) and source domain (training data) are from the same distribution. It is the most common learning setting. In this setting, the prediction model should not overfit the training data.
  - **Transfer Learning** Transfer learning refers to the learning settings where the target domain and source domain are significantly different. There are several types of discrepancies between the target domain and source domain [131]. In the context of human behavior prediction, we usually focus on one type of transfer learning problem, which is called domain adaption. Domain adaptation has two assumptions: i) The relation between input (historical observation) and output (predicted future human behavior) is determined and invariant to the marginal distribution of input and output. ii) The input distribution of the target domain and source domain are different.
  - **Continual Learning** In contrast to the aforementioned learning settings that deal with static datasets, continual learning investigates the generalization performance in dynamic changing environments. In this setting, datasets come sequentially. For instance, human driving datasets can be collected continually across different locations. Although the power of computation has increased rapidly in recent years, and the current computation infrastructure can afford large datasets and models, it is still not the desirable and optimal way to retrain the model with incremental datasets from the beginning. One of the challenges of continual

learning is how to mitigate catastrophic forgetting, which means that the performance of models degrades on previous tasks when learning a new task. Such a phenomenon is usually observed in deep-neural-nework-based approaches.

Given that current research on human behavior prediction are still underdeveloped within both transfer learning and continual learning settings, this dissertation primarily focuses on these two problems.

1.1.2 Human Behavior Generation

Human behavior generation can be used in stress testing of autonomous vehicles, animation, and gaming. It can also be transformed into demonstrations for robot (e.g., humanoid robot) skill learning. Compared with human behavior prediction, which focuses more on density estimation, human behavior generation primarily focuses on the support coverage of probability distribution. Besides the aforementioned properties in human behavior prediction, we particularly focus on two properties that are specifically considered in human behavior generation:

- **Diversity** Unlike human behavior prediction, human behavior generation focuses on whether it can generate diverse behavior as many as possible. There needs to be more than just estimating the probability density estimation to satisfy this requirement. Meanwhile, even assuming that we have obtained a perfect estimated probability density of human behavior, generating the minor modes of human behavior within a limited number of samples remains demanding.
- **Feasibility** Another favored property of human behavior generation is feasibility. For instance, only learning human driving behavior directly from data cannot guarantee that the generated trajectories are feasible in satisfying vehicle kinematics, collision avoidance, etc. Therefore, how to incorporate such feasibility into a learning-based system is also an essential question for human behavior generation.

1.2 Dissertation Contributions and Outline

The goal of this dissertation is to *"develop learning-based human behavior modeling approaches for prediction and generation from model and learning algorithm design perspectives"*. As illustrated in Figure 1.2, a typical learning agent has two components, namely, model and learning algorithm. Model design mainly focuses on choosing the proper structure and inductive bias to improve performance; learning algorithm design pays attention to selecting learning objectives and optimization strategies. We will interweave these two design perspectives within several problem settings in this dissertation. In particular, the dissertation is divided into two parts. Part I focuses on improving the generalizability of human driving behavior prediction in transfer learning and continual learning settings. In

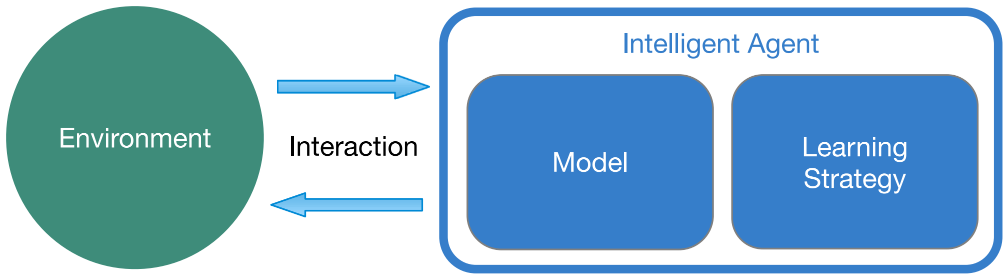


Figure 1.2: Typical components of a learning system.

Part II, we shift the focus from behavior prediction to generation. The following subsections provide a detailed summary of the contribution made by each chapter. In addition, the outline of this dissertation is summarized in Figure 1.3.

1.2.1 Part I: Improving Generalization for Human Behavior Prediction

Chapter 2

How to make precise multi-agent trajectory prediction is a crucial problem in the context of autonomous driving. It is significant to have the ability to predict surrounding road participants' behaviors in many different, seen or unseen scenarios for enhancing autonomous driving safety and efficiency. Extensive research has been conducted to improve the overall prediction performance based on one enormous dataset or pay attention to some specified scenarios. However, how to generalize the prediction to different scenarios is less investigated. In this chapter, we introduce a graph-neural-network-based framework for multi-agent interaction-aware trajectory prediction. In contrast to recent works which use the Cartesian coordinate system and global context images directly as input, we propose to leverage human prior knowledge, such as the comprehension of pairwise relations between agents and pairwise context information extracted by self-supervised learning approaches to attain an effective Frenét-based representation. We evaluate our method across different traffic scenarios with diverse layouts and compare it with state-of-the-art methods. We demonstrate that our approach achieves superior performance in terms of overall performance, zero-shot, and few-shot transferability.

Chapter 3

The current mainstream research of multi-agent trajectory prediction in autonomous driving focuses on how to achieve accurate prediction on one large dataset. However, whether the multi-agent trajectory prediction model can be trained with a sequence of datasets, i.e., continual learning settings, remains a question. Can the current prediction methods avoid catastrophic forgetting? Can we utilize the continual learning strategy in the multi-agent trajectory prediction application? Motivated by the generative replay methods in continual learning literature, we propose a multi-agent interaction behavior prediction framework with a graph-neural-network-based conditional generative memory system to mitigate catastrophic forgetting. To the best of our knowledge, this chapter is the first attempt in the literature to study the continual learning problem in multi-agent interaction behavior prediction problems. We empirically show that several approaches in the literature indeed suffer from catastrophic forgetting, and our approach succeeds in maintaining a low prediction error when datasets come in a sequential way. We also conduct an ablation analysis to show the effectiveness of our proposed approach.

Chapter 4

Besides improving the quality of prediction models, uncertainty estimation is essential to evaluate the reliability of the models. Having a proper uncertain estimation model can also be used to determine if the data collected in new scenarios are valuable, and hence it could be used for improving the data efficiency of transfer learning. However, the current uncertain estimation approaches for vehicle motion prediction are tightly incorporated with predictors, thereby it is not desirable for a third-party organization, such as an insurance company or department of transportation, who wants to evaluate an autonomous driving company's predictor's performance on uncertainty estimation. In this chapter, We first demonstrate that the proposed predictor-agnostic uncertainty estimation approach for trajectory prediction can achieve comparable performance with predictor-dependent approaches. Then we provide an analysis of how different regimes of features, i.e., static information and agent information, contribute to uncertainty estimation for vehicle trajectory prediction. Finally, we show how to utilize the conclusion of the analysis to improve the predictor's training efficiency when transferring to new scenarios based on the proposed framework.

1.2.2 Part II: From Human Behavior Prediction to Generation

Chapter 5

Obtaining accurate and diverse human motion prediction and generation is essential to many industrial applications, especially robotics and autonomous driving. Recent research has explored several techniques to enhance diversity and maintain the accuracy of human motion prediction at the same time. However, most of them need to define a combined loss, such as the weighted sum of accuracy loss and diversity loss, and then decide their weights as hyperparameters before training. In this chapter, we aim to design a prediction framework that can balance the accuracy sampling and diversity sampling during the testing phase. In order to achieve this target, we propose a multi-objective conditional variational inference prediction model. We also propose a short-term oracle to encourage the prediction framework to explore more diverse future motions. We evaluate the performance of our proposed approach on two standard 3D human motion datasets. The experiment results show that our approach is effective and on par with state-of-the-art performance in terms of accuracy and diversity.

Chapter 6

In this chapter, we investigate how to ensure the feasibility of the generated human behavior. In particular, we focus on the trajectory generation problem in collision avoidance scenarios. Recently one of the optimization-based approaches called control barrier functions (CBF) has become a popular tool to enforce the safety of a control system. CBFs are commonly utilized in a quadratic program formulation as safety-critical constraints. A class $\mathcal{K}$ function in CBFs usually needs to be tuned manually to balance the trade-off between performance and safety for each environment. However, this process is often heuristic and can become intractable for high relative-degree systems. Moreover, it prevents the CBF-QP from generalizing to different environments in the real world. By embedding the optimization procedure of the exponential control barrier function based quadratic program (ECBF-QP) as a differentiable layer within a deep learning architecture, we propose a differentiable safety-critical control framework that enables generalization to new environments for high relative-degree systems with forward invariance guarantees. Finally, we validate the proposed control design with 2D double and quadruple integrator systems in various environments. The proposed method can be integrated with the approaches proposed in the previous chapters.

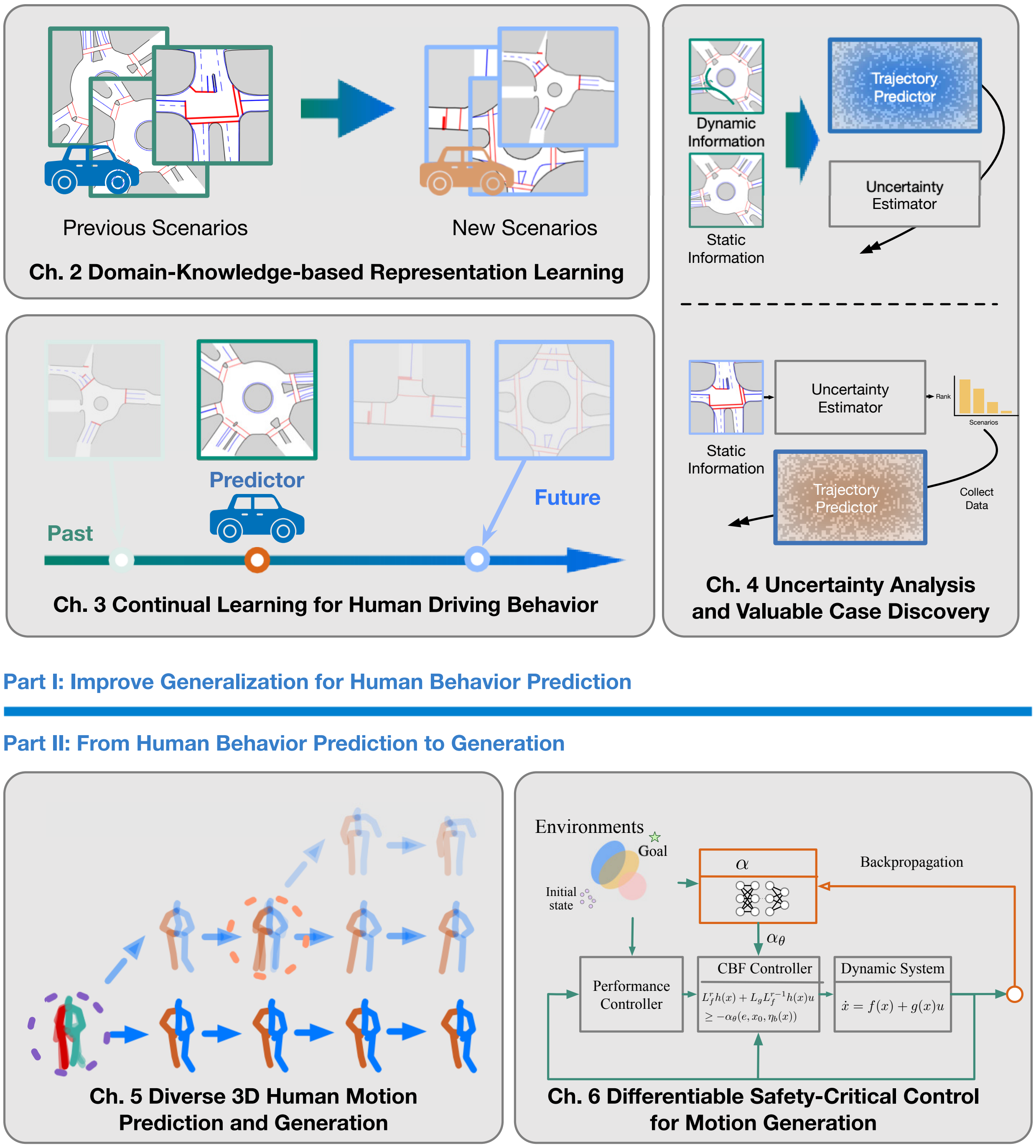

Previous Scenarios
New Scenarios
Ch. 2 Domain-Knowledge-based Representation Learning
Dynamic Information
Static Information
Trajectory Predictor
Uncertainty Estimator
Static Information
Uncertainty Estimator
Rank
Scenarios
Trajectory Predictor
Collect Data
Ch. 4 Uncertainty Analysis and Valuable Case Discovery
Predictor
Past
Future
Ch. 3 Continual Learning for Human Driving Behavior
Part I: Improve Generalization for Human Behavior Prediction
Part II: From Human Behavior Prediction to Generation
Ch. 5 Diverse 3D Human Motion Prediction and Generation
Environments
Goal
Initial state
α
Backpropagation
α_θ
Performance Controller
CBF Controller
$L_f^r h(x) + L_g L_f^{r-1} h(x) u \geq -\alpha_\theta(e, x_0, \eta_b(x))$
Dynamic System
$\dot{x} = f(x) + g(x)u$
Ch. 6 Differentiable Safety-Critical Control for Motion Generation

Figure 1.3: Dissertation outline.

Part I

Improving Generalization for Human Behavior Prediction

Chapter 2

Domain-Knowledge-Based Transferable Behavior Prediction

2.1 Introduction

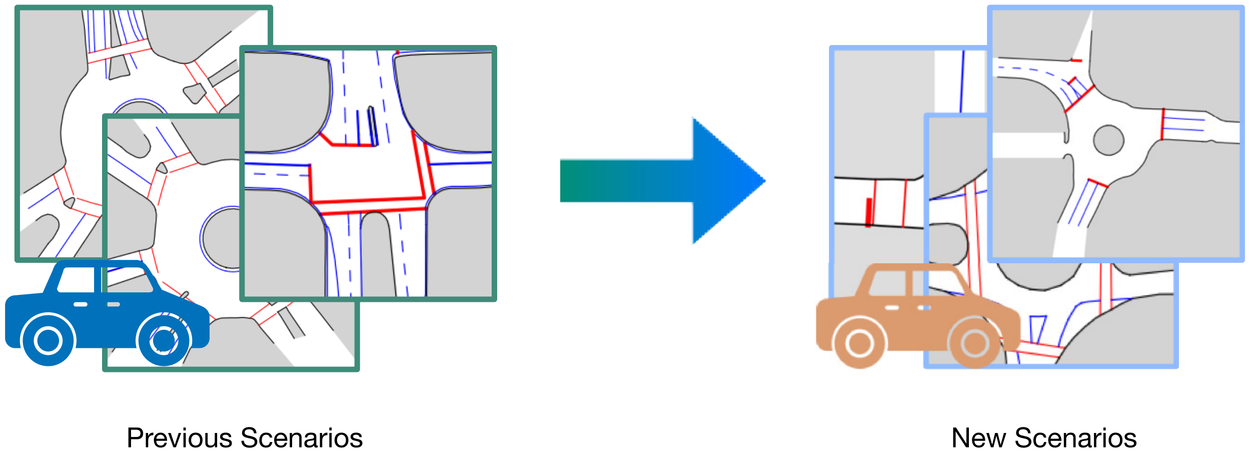


Figure 2.1: The illustration of generalization of multi-agent trajectory prediction across different scenarios. The predictor could be trained on a dataset of several scenarios, then tested on several new scenarios without new data (zero-shot) or with a small batch of data (few-shot) for training.

Multi-agent behavior prediction has a pivotal role in many real-world applications, such as autonomous driving and mobile robot navigation. Making a precise prediction in different situations gives a promise of safety with proper planning algorithms. Human drivers can transfer their prediction and driving ability from previous scenarios to new ones only after driving in the new scenarios a few times. Imagine that a driver Alice has driven in San Francisco (SF) for many years. She goes to New York (NY) for business and rents a car. Although she has never been to NY, where driving behaviors and road layouts are different from those in SF, she can be familiar with the driving patterns in NY very quickly. Much of the current literature on behavior prediction pays particular attention to improving overall prediction performance. However, in this chapter, we focus more on the generalization problems of multi-agent trajectory prediction in autonomous driving applications as shown in Figure 2.1.

Recently, several works in machine learning and computer vision indicate that introducing inductive bias is necessary to improve the generalization of the deep learning framework. The inductive biases could be specified deep learning model structures, constraints, and context information, etc. Such inductive bias could be used to extract general representation of data for future usage. For instance, there exist many works proposing different model structures such as graph neural network [83, 96], transformers [162] to capture the multi-agent interaction mechanism. These specified deep learning model structures indeed improve the generalization, while they still lack the subtle domain knowledge of autonomous driving to improve the transferability in advance. Although several works are incorporating more context information such as high-definition maps [54, 189, 98] to improve the prediction accuracy, most of the methods are trained in an end-to-end style. It is not clear that if such context representation and training strategy are efficient for generalization.

In this chapter, we propose an approach to more effectively utilize the context information, such as the references of agents, via leveraging human's prior knowledge. First, we argue that instead of providing the road layout information, i.e., the reference of each vehicle implicitly, such as given the rasterized images of the high-definition map directly as input, we can explicitly incorporate the references information to future trajectories predictions by Frenét transformation. The Frenét transformation can constrain the predicted trajectories around the references, which improves the zero-shot and few-shot transferability. Then we design a set of features based on the human understanding of interaction behaviors in the Frenét coordinate system serving as the inductive bias. Lastly, in contrast to using end-to-end supervised learning, we apply the self-supervised learning technique, which can reduce invariant factors to get a more general representation for the intrinsic relative geometry information of references of each interaction pair of agents. After obtaining the feature representation, we use a message-passing graph neural network to capture the interaction behaviors. We argue that such representations not only improve the overall prediction performance but also improve the generalization and transferability significantly.

The main contributions are summarized as follows:

- We demonstrate an effective approach to leverage Frenét-based trajectory prediction and rule-based interaction-level semantic classification to extract a good feature representation, which enhances the transferability across different scenarios.

- We adopt a self-supervised learning technique to extract the context information of interaction pairs and demonstrate that it can achieve better prediction performance.

- We evaluate the overall prediction accuracy and transferability of the proposed approach on a benchmark dataset including various interactive driving scenarios. The framework achieves significant enhancement compared with state-of-the-art methods.

2.2 Related Works

2.2.1 Behavior and Trajectory Prediction

In recent years, there has been an increasing amount of literature on trajectory prediction due to the rising of topics including autonomous driving and human-robot interaction. Early examples of research focus on using model-based or traditional machine learning methods such as intelligent driving model [161], hidden Markov model [170] to predict the future trajectories. With the increase of the computational power, deep-learning-based methods become more available and achieve superior performances compared with the traditional methods. Furthermore, more particular issues, including how to deal with the different number of agents, probabilistic prediction and how to incorporate map information, are investigated. One line of the methods such as [107, 92, 95, 37, 94], etc. propose generative learning frameworks to obtain the complex, multi-modal future trajectory prediction. The development of graph neural networks [64, 83, 96, 164, 31] and attention mechanism provided powerful tools to solve the multi-agent prediction problems. Several works including [100, 59, 137, 44, 108] successfully adopt such ideas into the multi-agent trajectories problems. Other approaches focus on how to incorporate more information, such as the high-definition (HD) map and point clouds. Convnet [23] proposes to use the rasterized image of maps directly as inputs of a convolutional neural network. Vectornet [54] proposes to encode the vectors of lanes into a graph as the context information. [22] designs a method to utilize both maps and LiDAR information.

In contrast to these methods extracting the future road information of one agent implicitly from the contextual inputs such as the image or vectors of roads, we explicitly constrain the future predicted trajectories by mapping it into the Frenét coordinate system according to the reference of one agent. The Frenét representation is well-investigated, especially in motion planning literature [171] while there is little work about how to incorporate it into trajectory prediction framework. We show that this approach is more effective and enhances the performance of generalizations to new scenarios.

2.2.2 Self-Supervised Learning

Since the increasing expense of labeling massive data, researchers have shown an increased interest in learning representations from unlabeled data. Sometimes it is impossible to label data before knowing the downstream tasks. However, since many data have their own particular information structures, e.g., the local relation in the image, it becomes possible to exploit such information to obtain the intrinsic representation for future usage. Self-supervised learning is a technique to extract efficient representation before the task (e.g, classification) is known. Since there is no task information, the auxiliary (pretext) tasks should be defined in order to discover the similarity between different features. These pretext tasks provide pseudo labels as supervision. For instance, color transformation and geometric transformation are usually used in the computer vision area. Some works propose to use the contrastive loss as the self-supervision [26, 61, 62]. One intuitive explanation of contrastive learning is to make similar samples closer and make dissimilar samples repulse each other [74]. Self-supervised learning has been empirically demonstrated to be able to extract better representation when the labels are limited in many applications such as image classification [26], natural language processing [41], and reinforcement learning [149, 186]. Recent work [150] shows that self-supervised learning can also improve the few-shot learning performance in image classification. We adopt the contrastive learning concepts to extract the relative geometry information of references.

2.3 Problem Formulation

Without loss of generality, we assume that there are N agents in a case. In different cases, there may be a varying number of agents. We have the historical observations $O^i_{t-H+1:t}$ of each agent i, which includes its trajectory $X^i_{t-H+1:t}$ and the reference information R^i. The reference R^i represents the vehicle's routing information. It can be the middle of the lane which the vehicle is following. Such information can be extracted from the HD map. We denote the rasterized image of R^i as I^i, and the rasterized image of each pair of references R^i, R^j as I^{ij}. Given such information, we aim to predict the conditional distribution $P(X_{t+1:t+F}|O_{t-H+1:t})$. We denote H as the length of the historical horizon and F as the length of the prediction horizon. The variables without agent index i are denoted as the collections of different agents' corresponding variables (e.g. $X = \{X^i\}_{i=1:N}$). For the zero-shot or few-shot learning setting, we train our model on one dataset mixing several scenarios and test it on several new scenarios.

2.4 Methodology

2.4.1 Framework Overview

The whole framework is introduced in three parts: feature representation with human's prior knowledge, graph neural network design, and the training loss. The section of feature representation is divided into two parts: Frenét-based trajectory representation and self-supervised context representation. Section 2.4.3 introduces the graph neural network, which is divided into the attribute encoding layer, the message passing procedure and multi-modal decoder module. Section 2.4.4 introduces the training loss we used. A high-level summary is shown in Figure 2.2.

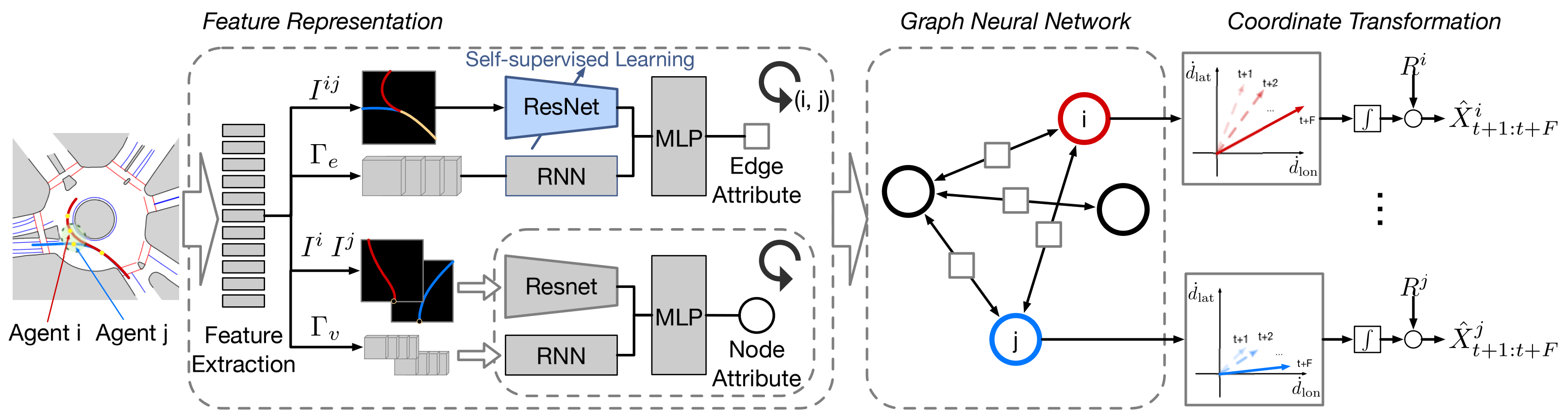


Figure 2.2: Overview of the proposed framework. There are three procedures in our approaches: **I** Feature Representation (Section 2.4.2). During the feature representation phase, the original multi-agent trajectory data is processed pairwise. Here we show one pair of agents (Agent i and Agent j) as an example. The blue ResNet block is pre-trained with self-supervised learning (Section 2.4.2). **II** Graph Neural Network (Section 2.4.3). Once we obtain the node attributes and edge attributes, we use a graph neural network to capture the interaction mechanism between agents. **III** Frenét-Cartesian Transformation (Section 2.4.2). The graph neural network predicts the future velocities of agents in the Frenét coordinate system. We integrate the predicted velocity of each agent and transform the trajectories into the Cartesian coordinate system.

2.4.2 Feature Representation

Frenét-based Trajectory Representation

The Frenét coordinate system is used since it can represent arbitrary reference paths efficiently. In the Frenét representation, the Cartesian coordinate (x, y) is transformed into longitudinal distance d_{lon} and lateral displacement d_{lat} given the reference R. Since the Frenét representation has already included the geometry information about each vehicle's reference, the final prediction results will incorporate the reference naturally. [185] argues that the topological relationship between any of two references can be decomposed into different types, and we adopt these ideas as prior knowledge to define four types of features: node features and three types of edge features.

For node features, we use the longitudinal speeds $\dot{d}^i_{\text{lon}}$, lateral speeds $\dot{d}^i_{\text{lat}}$, and the rasterized image I^i of the vehicle i's reference R^i as the features. The image I^i use the vehicle i's coordinate system, where the Y-axis direction of image is the velocity's direction. We denote the node feature selection and extraction as a mapping $\Gamma_v : \mathcal{X} \rightarrow \mathcal{S}_v$, where $\mathcal{X}$ is the trajectory space and $\mathcal{S}$ is the feature space.

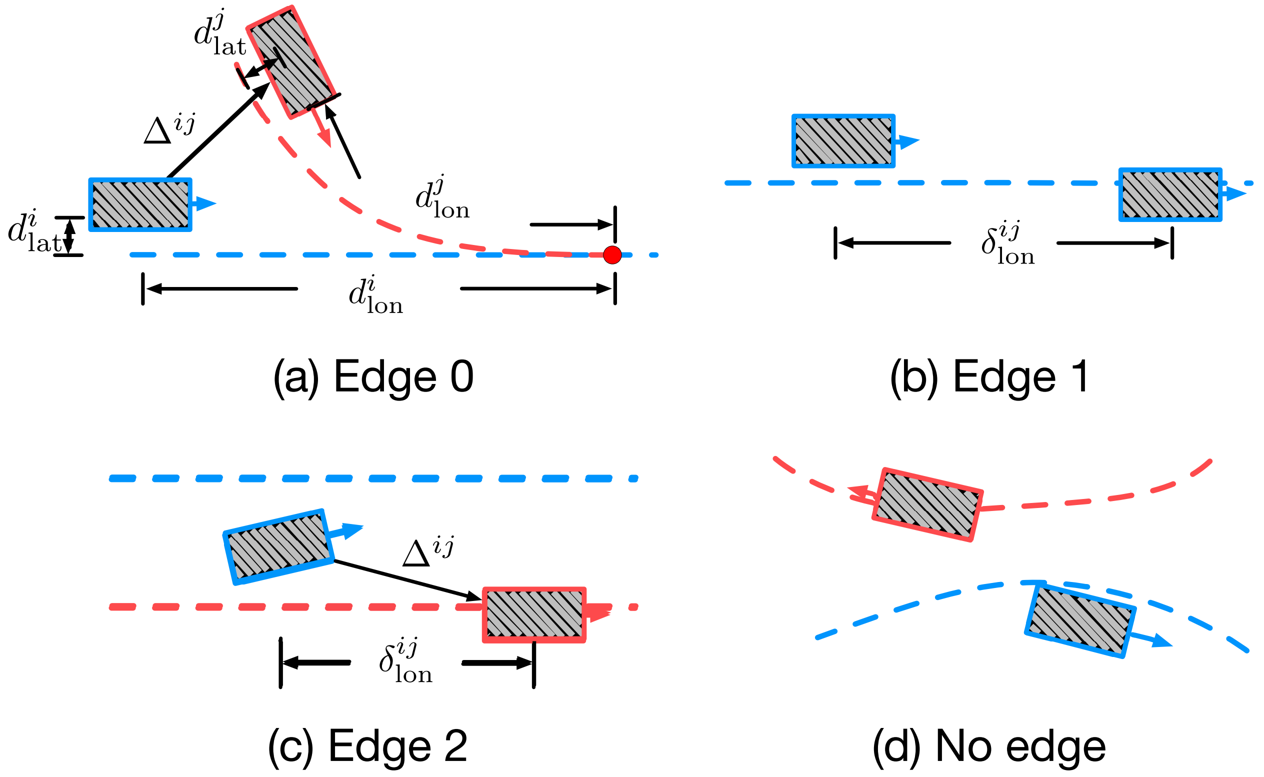


Figure 2.3: The illustration of different types of edges. Different colors represent agents with different references. The dashed lines are the references for agents.

Table 2.1: The different types of features.

Features	Node	Edge0	Edge1	Edge2
$\dot{d}^i_{\text{lon}}$	✓			
$\dot{d}^i_{\text{lat}}$	✓			
R^i	✓			
$d^{\text{i, j}}_{\text{lon}}$		✓		
$d^{\text{i, j}}_{\text{lat}}$		✓		
Δ^{ij}		✓		✓
C^{ij}		✓		
δ^{ij}_{lon}			✓	✓

For edge features, we illustrate different conditions of interaction pairs in Figure 2.3. In Figure 2.3 (a), when the references of two vehicles intersect, we set the intersection point as the origin of the Frenét coordinate system. We denote $d^{\text{i, j}}_{\text{lon}}$ as the collection of vehicle i's and j's longitudinal distance to the origin point. $d^{\text{i, j}}_{\text{lat}}$ has the similar definition for the lateral displacement. We denote Δ^{ij} as the relative position of the vehicle i and j in the Cartesian coordinate system, where the origin point is the location of the vehicle i and the Y-axis direction is its velocity direction. We also employ the context information of the relation between the references of a pair of agents as one of the features C^{ij}. The details of the feature C^{ij} will be introduced in Section 2.4.2. In Figure 2.3 (b), if two vehicles share the same reference, we define the relative longitudinal distance δ^{ij}_{lon} between them. In Figure 2.3 (c), if two vehicles' references do not intersect while lane changing is feasible, we use δ^{ij}_{lon} and Δ^{ij} as features. In Figure 2.3 (d), if two references are mutually exclusive, there will be no edge. Table 2.1 summaries the feature selections for different types of features. We denote the edge feature selection and extraction as a mapping $\Gamma_e : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{S}_e$, where $\mathcal{X}$ is the trajectory space and $\mathcal{S}$ is the feature space. We also denote the edge type of one pair of agents i and j as α_{ij}.

In order to generate such trajectory representation, there are two submodules here: reference path extraction and coordinate transformation.

Reference Path Extraction This module aims to extract each vehicle's reference. If a high-definition map is provided and the road layout is very simple, we can directly use the reference defined in the HD map. However, if the road layout is complicated, such as roundabouts, using the hand-crafted references is not accurate. One better solution is to use a few trajectory data with the same starting area and ending area to get the approximate references. We first collect the trajectories according to the starting and ending areas. Each starting and ending area is defined as a quadrilateral area indicating an agent coming from or heading to this path. Then we can select a few data with this reference. For a set of

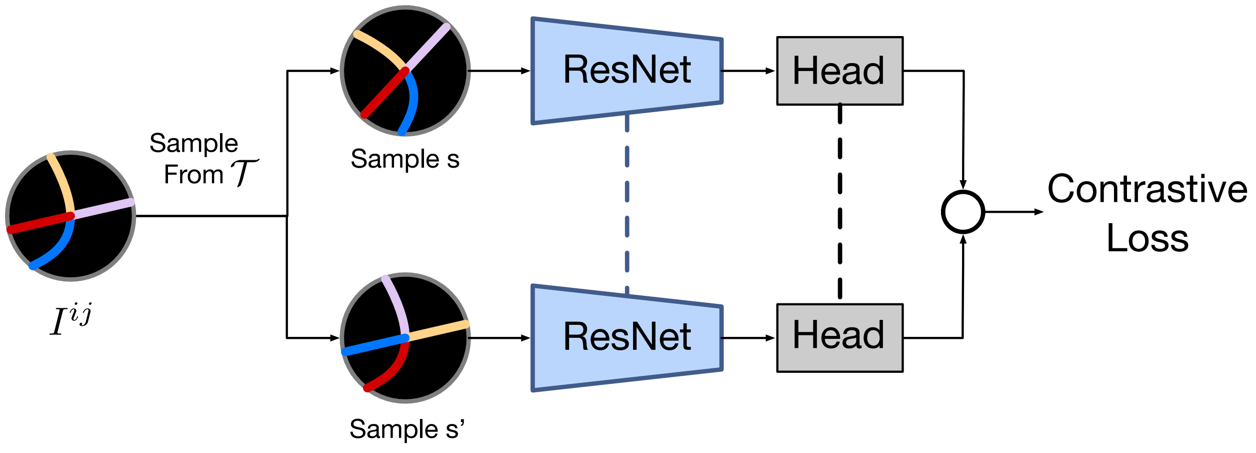


Figure 2.4: The contrastive learning framework. Sample s indicates one sample from the rotation transformation, and s' indicates one sample from the semantic exchanging transformation. I^{ij} is one example of the rasterized image of two intersected references. The blue-yellow curve represents one reference and the direction is from blue to yellow. The red-purple curve represents the other reference and the direction is from red to purple.

trajectories with the same starting and ending area, we use a polynomial function to fit them. Then we sample points from the fitted curve every 0.05 meter and use them as the reference path.

Coordinate Transformation To transform coordinates from Cartesian to Frenét, we can find the nearest point on the references, and then calculate the lateral displacement d_{lat} to the projection point and the longitudinal distance d_{lon}. Through the inverse process, we can transform the trajectories in the Frenét coordinate system back to the Cartesian coordinate system.

Context Representation with Self-Supervised Learning

Despite the Frenét coordinate system is used in the feature representation, it only contains the geometric information of the reference path of each vehicle. When two vehicles are interacting with each other, their behaviors also depend on the relations between the two reference paths. The relations between two references have different levels of abstractions such as topology and geometry [185]. Under the circumstance that the references of two vehicles i and j intersect, we can rasterize those two references in Image I^{ij} with the intersection point as the center point of the image. We use different pairs of colors to indicate the vehicle's moving direction in order to distinguish two references. One example is shown in Figure 2.4.

We suggest that the relation between two references is invariant to different positions (views) of vehicles. For instance, the drivers of two different vehicles will have the same understanding of the relation between the two references. Besides, the orientation information of reference for each agent has already been provided in the node feature, i.e., the image I^i. We assume that there is an effective abstraction that can represent the relation between two references and use the self-supervised technique to extract such representation. We utilize a similar approach in SimCLR [26] to discover the similarities between different objects by designing some pretext tasks. These pretext tasks serve as data augmentation for generating positive samples. Considering the property of the context images as we suggest above, we define the family of tasks $\mathcal{T}$ as:

- **Rotation**: The original image is centered at the intersection point, which is defined as the first point at which two references intersect. Then the original image is rotated randomly from 0 to 2π. In Figure 2.4, s shows one sample by rotation.
- **Semantic Exchanging**: We need to use distinct colors to indicate different references to avoid vagueness (i.e. which segment belongs to which reference, what is the direction of one reference). Since the topology and geometry are not related to the semantic order, we exchange the colors of two different references. In Figure 2.4, s' shows one sample by semantic exchanging.

Under those pretext tasks $\mathcal{T}$, the loss function for a positive pair of samples I_s^{ij} and $I_{s'}^{ij}$ is:

$$l(I_s^{ij}, I_{s'}^{ij}) = -\log \frac{e^{\beta \cos(z_s, z_{s'})}}{\sum_{k \neq s} e^{\beta \cos(z_s, z_k)}}, \tag{2.1}$$

where $z_s = \mathrm{h}(\mathrm{enc}(I_s^{ij}))$, and $I_s^{ij}, I_{s'}^{ij} \sim \tau(I^{ij})$ are the random samples. h is a projection head. $\tau(\cdot) : \mathcal{I} \to \mathcal{I}$ is a random function sampled from the pretext task family $\mathcal{T}$. After training with the contrastive loss, we set $C^{ij} = \mathrm{enc}(I^{ij})$ as the representation of context information.

2.4.3 Graph Neural Network Design

We can represent the agents in the traffic as a graph and use a graph neural network to capture the interaction behaviors. The graph can be defined as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_i\}, i \in \{1, \ldots, N\}$ and $\mathcal{E} = \{e_{ij}\}, i, j \in \{1, \ldots, N\}$. v_i, e_{ij} denote the node attribute and the edge attribute, respectively. Specifically, e_{ij} denotes the edge attribute from v_j to v_i. Given a set of trajectory observations, we can apply the feature selection and extraction process mentioned in Section 2.4.2 to compute the initial attributes for each node and edge. We name this layer as the attribute encoding layer:

$$\begin{aligned} v_i^0 &= f(\mathrm{RNN}(\Gamma_v(X^i_{t-H+1:t})), \mathrm{ResNet}(I^i_t)), \\ e_{ij}^0 &= h(\mathrm{RNN}(\Gamma_e(X^i_{t-H+1:t}, X^j_{t-H+1:t})), C^{ij}_t), \end{aligned} \tag{2.2}$$

where f and h are the node and edge embedding functions. Note that C_t^{ij} is only used with Edge 0.

Then we use the message passing mechanism to reason the multi-agent interaction mechanisms. For each node and edge, we have:

$$\begin{aligned} e_{ij}^m &= f_{e,\alpha_{ij}}^m([v_i^{m-1}, v_j^{m-1}, e_{ij}^{m-1}]), \\ v_i^m &= f_v^m(\Phi[j \in \mathbf{N}(v_i)](e_{ij}^m)), m = 1, \ldots, n. \end{aligned} \tag{2.3}$$

where $f_{e,\alpha_{ij}}$ denotes different edge embedding functions for different edge types α_{ij}. f_v denotes the embedding function for nodes. The superscripts of v_i^m, e_{ij}^m, f_v^m, $f_{e,\alpha_{ij}}^m$ denote the m-th message passing. $\Phi[j \in \mathbf{N}(v_i)](\cdot)$ aggregates the information of all the edges e_{ij} between v_i and its neighbors $\mathbf{N}(v_i)$.

In order to output multi-modal trajectory prediction, a Gaussian mixture model is used as the multi-modal decoder to generate the future velocities in the Frenét coordinate system:

$$\begin{aligned} & w_j = \mathrm{softmax}(f_w^j(v_i^n)), \mu_j = f_\mu^j(v_i^n), \Sigma_j = f_\Sigma^j(v_i^n), \\ & \{\dot{d}_{\mathrm{lon},t+1:t+F}, \dot{d}_{\mathrm{lat},t+1:t+F}\} \sim \sum_j w_j \mathcal{N}(\mu_j, \Sigma_j), \end{aligned} \tag{2.4}$$

where w_j, μ_j, and Σ_j describe the weight, mean, and variance of the j-th Gaussian function.

After the predicted velocities of each agent are obtained, a first-order integrator is applied to get the predicted future positions in the Frenét coordinate system. Then the predicted trajectories would be transformed to the Cartesian coordinate system to evaluate the performance. We illustrate the Frenét-Cartesian transformation in Figure 2.2.

2.4.4 Training Loss

We use the negative log-likelihood $\mathcal{L}(\theta, \mathcal{D})$ as the objective function during the training phase:

$$\mathbb{E}_{(O_{t-H+1:t}, X_{t+1:t+F}) \sim \mathcal{D}}[-\log P_\theta(X_{t+1:t+F} | O_{t-H+1:t})], \tag{2.5}$$

where θ represents all the parameters of our model. Since the direct outputs of our model are the predicted velocities $(\dot{d}_{\mathrm{lon}}, \dot{d}_{\mathrm{lat}})$ based on the Frenét coordinate system, we can also directly optimize the empirical loss based on the Frenét coordinate system during training.

2.5 Experiment Results

This section introduces the dataset, evaluation metrics and baselines in Section 2.5.1 and Section 2.5.2 firstly. Then the comparisons between our method and other baseline approaches of the overall prediction performance and transferability are demonstrated in Section 2.5.3 and Section 2.5.4.

2.5.1 Dataset

The experiments were conducted on the INTERACTION dataset [184], which contains naturalistic motions of various traffic participants in a variety of highly interactive driving scenarios, including roundabouts, unsignalized intersection, and lane merging. In each scenario, the data is sampled from different locations. We chose this dataset for the following reasons. First, the geometry of road layouts is complicated. Most of the cases in other datasets are collected with simple road layouts like straight multi-lane and cross-style intersections. In contrast, the INTERACTION dataset contains more curved, challenging road layouts such as roundabouts. Second, it has a higher detection accuracy than other datasets and more highly interactive cases. Therefore, it is suitable for testing transferability across different scenarios and multi-agent interactive behavior prediction. We selected five urban representative scenarios (MA, FT, SR, EP-T, and EP-R), which have various road layouts in our experiments. We predicted the future 10 time steps (5.0s) based on the historical 4 time steps (2.0s) in all experiments.

2.5.2 Metrics and Baselines

We adopt two widely used probabilistic prediction metrics. One is the minimum average displacement error (mADE), which computes the Euclidean distance between the ground truth positions and the closest trajectory from K candidates, which are sampled from the predicted probability distribution. The other is the minimum final displacement error (mFDE) that evaluates only the displacement error at the last time step. Both metrics are suitable to measure probabilistic prediction. We compare our method with five baseline approaches about the overall performance and transferability. We provide the same input information for all the methods. The following are the algorithms we compare:

- LSTM. Long-short term memory is a kind of recurrent neural networks used widely to learn the time-series pattern. We use it as a prediction model which does not consider the interaction explicitly.
- Social LSTM (S-LSTM) [4]. The model designs a social pooling mechanism based on LSTM.
- Social GAN (S-GAN) [59]. The model employs generative adversarial learning into S-LSTM.
- Trajectron++ [138]. One of the state-of-the-art approaches employs spatial-temporal graph with dynamic constraints.
- Graph Neural Network (GNN). The network structure of this method is similar to our proposed method. The difference is that GNN uses the historical trajectories in the Cartesian coordinate system as the node features and does not use the routing-related edge features such as the edge types and the relative positions in the Frenét

Table 2.2: Comparison of mADE / mFDE (Meters) in All Scenarios

	LSTM	S-LSTM	S-GAN
@3.0s	0.344 / 0.597	0.438 / 0.692	0.400 / 0.652
@4.0s	0.598 / 1.128	0.611 / 1.071	0.599 / 1.042
@5.0s	0.918 / 1.888	0.879 / 1.721	0.845 / 1.528
	GNN	**Trajectron++**	**Ours**
@3.0s	0.271 / 0.534	0.185 / 0.336	**0.139** / **0.284**
@4.0s	0.469 / 0.969	0.360 / 0.677	**0.268** / **0.562**
@5.0s	0.695 / 1.457	0.608 / 1.167	**0.432** / **0.913**

Coordinate system. Also, GNN does not employ our contrastive learning method to extract context features. Hence, GNN can also serve as an ablation method.

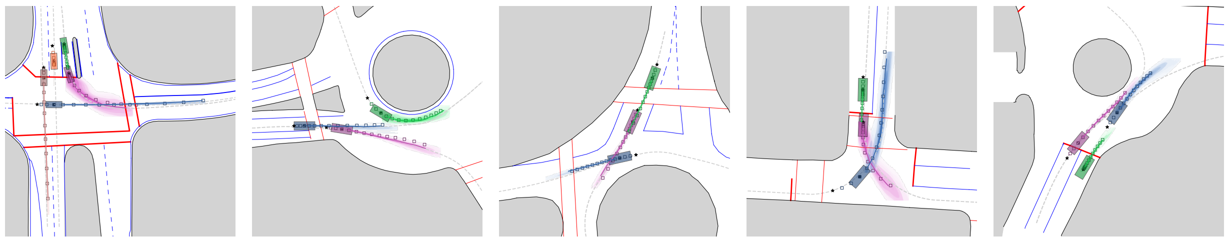
2.5.3 Prediction Performance in All Scenarios

Quantitative Results

First, we show the general prediction performance of our method compared with the baselines. We test all the models with all the data in different scenarios. The prediction results on all scenarios are shown in Table 2.2. The units of reported metrics are meters in the Cartesian coordinate system. We observe that all the methods which model the interaction explicitly, such as S-LSTM, S-GAN, GNN, Trajectron++, and Ours, are better than LSTM, which predicts each vehicle independently. Our approach improves about 24.9% in mADE with 3 seconds prediction horizon and 28.9% in mADE with 5 seconds prediction horizon compared with the best baseline (Trajectron++). It is also about 15.5% and 21.8% improvement in mFDE with 3 seconds and 5 seconds prediction horizon, respectively. It shows that our method has significant improvement compared with baselines.

Qualitative Results

We visualize five typical cases where there are more than two vehicles in Figure 2.5. The ground truth and predicted trajectories are shown in the same color for each vehicle. We find that the prediction results in all the scenarios are accurate. Besides, Our method is capable of capturing subtle behaviors such as yielding or not yielding in complicated scenarios.



(a) MA (b) FT (c) SR (d) EP-T (e) EP-R

Figure 2.5: The visualization of prediction results. The box markers are the ground truth trajectories. The solid lines are the sample trajectories with the smallest mADE. The density maps around the solid lines are generated by using kernel density estimation (KDE) to fit the sampled trajectories. The grey dash lines are the references (R) of each vehicle. The star markers are the starting positions of the historical trajectories.

2.5.4 Transferability to Other Scenarios

In this section, we compare the transferability of different methods. All the methods are trained on the mixed data of two scenarios: a roundabout (FT) and an intersection (MA). We choose these two scenarios since they can cover most of the different types of road layouts, including roundabout and intersection so that we can train a good basic predictor at the beginning. Then we evaluate the zero-shot and few-shot performance of transferring to another three different scenarios (two of them are different roundabouts, and the other is a different intersection). The results with 5 seconds prediction horizon are shown in Table 2.3. The mADE of our approach is 32.4%, 11.0%, and 38.4% better than the baseline methods in zero-shot transfer to SR, EP-T, and EP-R, respectively. For few-shot learning, we randomly sample 100 trajectories for each scenario as the training data. We demonstrate that our method performs better than the others with 40.6%, 30.8%, and 37.4% improvements in mADE for SR, EP-T, and EP-R. We observe that the improvement of zero-shot / few-shot learning on EP-T is relatively small compared with the other two scenarios. We suggest that the reason is that the road layout of EP-T is very similar to the one in MA, since they both include 90-degree intersections. It also implies that the more similar the scenarios are, the easier generalization will be. The observation also adheres to our intuition.

2.6 Ablative Analysis

We intend to answer the following questions with the ablation models in Table 2.4.:

- **Does the self-supervised learning technique improve the performance, and is the context image I^{ij} useful?** We compare Ours, Ours-E2E, and Ours-no_image in Table 2.4. The difference between them is the way they process the context images.

Table 2.3: Comparison of mADE / mFDE (meters) in the Different Scenarios

Scenario	Task	LSTM	S-LSTM	S-GAN
SR	zero-shot	1.849 / 4.470	1.927 / 3.625	1.801 / 3.539
	few-shot	1.184 / 2.452	1.458 / 3.010	1.267 / 2.385
EP-T	zero-shot	1.278 / 2.824	1.594 / 3.207	1.655 / 3.465
	few-shot	0.978 / 1.873	1.037 / 2.071	1.098 / 2.239
EP-R	zero-shot	2.268 / 5.306	2.824 / 5.531	2.590 / 5.130
	few-shot	1.453 / 2.970	1.520 / 3.281	1.483 / 2.739
Scenario	**Task**	**GNN**	**Trajectron++**	**Ours**
SR	zero-shot	1.977 / 4.714	1.395 / 2.333	**0.943** / **2.266**
	few-shot	1.180 / 2.430	1.043 / 1.816	**0.620** / **1.214**
EP-T	zero-shot	1.587 / 3.607	1.092 / **1.703**	**0.972** / 2.279
	few-shot	1.028 / 2.064	0.838 / 1.528	**0.580** / **1.003**
EP-R	zero-shot	2.388 / 5.514	2.074 / 3.478	**1.277** / **2.742**
	few-shot	1.393 / 2.743	1.328 / 2.376	**0.831** / **1.462**

Table 2.4: The Models used for Ablation

	Frenét-based Representation	Contrastive Learning	Image of References
Ours	✓	✓	✓
Ours-E2E	✓		✓
Ours-no_image	✓		
GNN			✓

Ours-no_image does not utilize the context image I^{ij}. Ours-E2E uses the context image but does not use contrastive learning.

- **Is the performance of the Frenét-coordinate-based trajectory prediction better than the Cartesian-coordinate-based one?** Here we compare the model GNN and Ours-E2E in Table 2.4, since the only difference between these two models is whether the Frenét-based features are used.

2.6.1 Self-Supervised Learning Ablation

Quantitative Analysis

To prove the effectiveness of the self-supervised learning method, we conduct experiments on the whole dataset, including all the scenarios for the three methods: Ours, Ours-E2E, and Ours-no_image. Ours-E2E trains the whole model in an end-to-end fashion, and Ours-no_image does not utilize the context image I^{ij} as one of the edge features. In Table 2.5, we find that Ours is about 10.2% better than Ours-E2E and approximately 16.4% better than Ours-no_image with 5.0s prediction horizon. Thus, it illustrates that employing self-supervised auxiliary tasks to pre-train the feature embedding layers does help to improve the prediction performance in general. Also, Ours-E2E improves about 7.0% more than Ours-no_image, which shows that the context information is indeed useful.

Qualitative Analysis

In Figure 2.6, we show the feature extraction results of the self-supervised learning method. We use t-SNE to illustrate the relations between each extracted feature. It shows that similar pairs of road references are gathered into the same group, and the different ones are separated. We also find that the self-supervised procedure could also discover the similarities between the pictures from new coming scenarios, which means our self-supervised method has a good transferability.

Table 2.5: Ablation of the Frenét Coordinate System and Self-Supervised Learning (mADE / mFDE)

methods	@3.0s	@4.0s	@5.0s
Ours	**0.139** / **0.284**	**0.268** / **0.562**	**0.432** / **0.913**
Ours-E2E	0.151 / 0.318	0.295 / 0.646	0.481 / 1.052
Ours-no_image	0.162 / 0.337	0.315 / 0.685	0.517 / 1.124
GNN	0.271 / 0.534	0.469 / 0.969	0.695 / 1.457

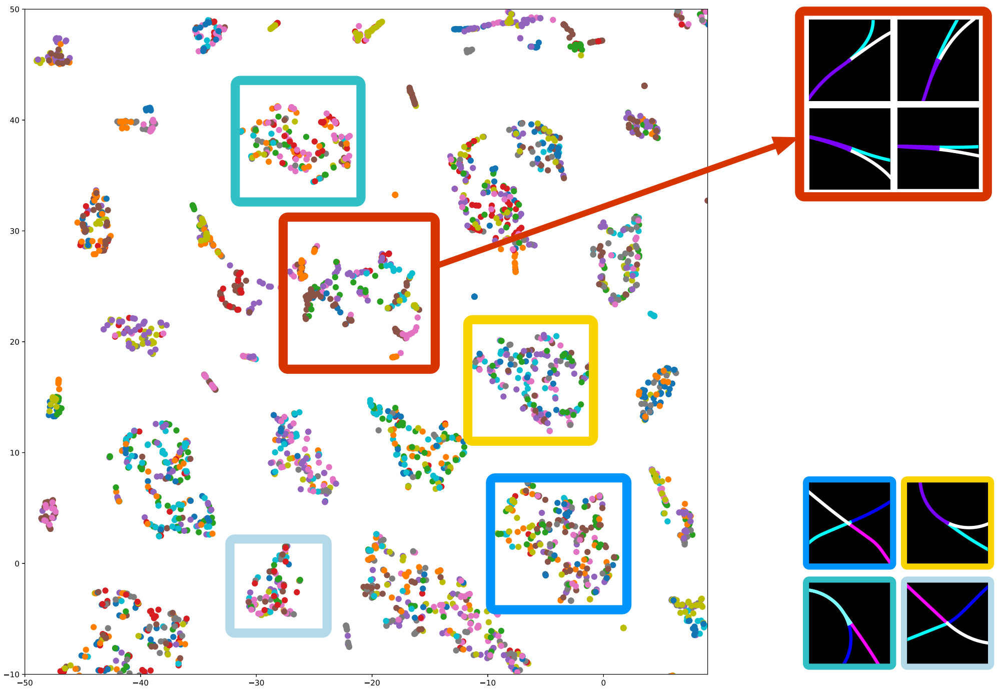


Figure 2.6: The t-SNE illustration of context information. Different colors of scatters represent different pairs of references in all scenarios. We illustrate several groups of extracted features here and show their corresponding rasterized images. The red rectangle shows that the new pairs of references are clustered into the same group. The other rectangles show the differences between different groups of extracted features.

2.6.2 Frenét-based Trajectory Prediction Ablation

In this analysis, we compare GNN and Ours-E2E. Notice that GNN does not use the Frenét-based features and the self-supervised learning, so the only difference between GNN and Ours-E2E is whether the Frenét-based features are used. In Table 2.5, we find that Ours-E2E improves about 30.8% mADE in the future 5s, which implies that our proposed Frenét-based trajectory representation improves the prediction performance remarkably.

2.7 Implementation Details

In the attribute encoding layer, we use GRU as the cell of RNN to extract the historical trajectory information for nodes and edges. We use ResNet18 [60] to encode all the rasterized images. We concatenate those two features and use a MLP as function f, h in Section 2.4.3. The ResNet for context images I^{ij} is pretrained by contrastive learning with a batch size of 1024. The dimensions of the image representation and the latent variables are 32 and

16. For the message passing procedure, f_v^m and $f_{e,\alpha}^m$ are MLPs. We use different MLPs for different types of edges. The message passing number is 3. For the multi-modal decoder module, the number of Gaussian kernels is 4, and we sample 20 trajectories to calculate the mADE and mFDE. f_w, f_μ, and f_Σ are also MLPs. All the MLP modules in our model are two-layer fully connected networks with an activation function of ReLU and a hidden size of 256. For the experiments of all scenarios, we mix all the data from five scenarios and divide it into 5:2:3 for training, validation, and testing. We train the model with a batch size of 64 for 100 epochs using Adam optimizer with an initial learning rate of 0.001. For the transferability experiments, the datasets of different scenarios are also divided into 5:2:3 for training, validation, and testing. The initial model is trained on MA+FT using the same hyperparameters as the all-scenarios experiments. We sample 100 cases from the training set of the new scenarios for the few-shot adaptation and fine-tune the model with a batch size of 20 and an initial learning rate of 0.0005. For all experiments, we show the average results of three random initializations.

2.8 Chapter Summary

In this chapter, a graph-neural-network-based framework with self-supervised domain knowledge is proposed to solve the multi-agent human driving behavior prediction problem. We incorporate human's prior knowledge and self-supervised learning techniques to enhance the generalizability and transferability across different traffic scenarios. Experiments demonstrate that our approach achieves significant improvement compared with other state-of-the-art methods. The ablative analysis demonstrates the effectiveness of the proposed feature representation technique. There are several future directions to investigate, such as how to incorporate more domain knowledge into this framework, including traffic rules and related knowledge about other types of traffic participants. Another interesting topic is how to design better pretext tasks for self-supervised learning.

Chapter 3

Continual Multi-agent Driving Behavior Prediction

3.1 Introduction

Predicting the possible future trajectories of surrounding traffic participants in different scenarios is essential to achieve the efficiency and safety of an autonomous driving system. The complicated interaction behaviors are attributed to many aspects, such as various complex road geometries [63, 23, 54] and multiple traffic agents [109, 138]. There are many existing works devoted to providing practical approaches by considering as many factors as possible given an enormous dataset [22]. However, there is little work investigating whether there is a multi-agent prediction approach working on the datasets continuously coming in a sequential way, and to our best knowledge, there is also no detailed investigation about the performance of existing multi-agent behavior prediction models in the continual learning settings.

With the fast development of hardware and autonomous driving infrastructure, the amount of data increases rapidly every day. It becomes not efficient to retrain a prediction model on the increasing datasets. The ideal way is to update the trained model on the new datasets without access to the old datasets. Also, those new datasets can be collected in new scenarios which are unseen before [183, 19, 85]. The current literature studies how to achieve a good prediction performance based on the datasets with all the scenarios. However, current methods may not work when the datasets are collected at different scenarios which are not available simultaneously. Intuitively, since the interaction behaviors at the new location may differ from the old ones remarkably due to the very different road layouts, the model may prefer to "fit" more on the current one rather than the old locations if we continually train our model on the new location without access to the old ones. Then the model will "forget" what it has learned before and perform worse on the previous locations. This phenomenon is well known as the "catastrophic forgetting issue" in the continual learning area [93]. Figure 3.1 is a demonstration of continual learning settings in the multi-agent

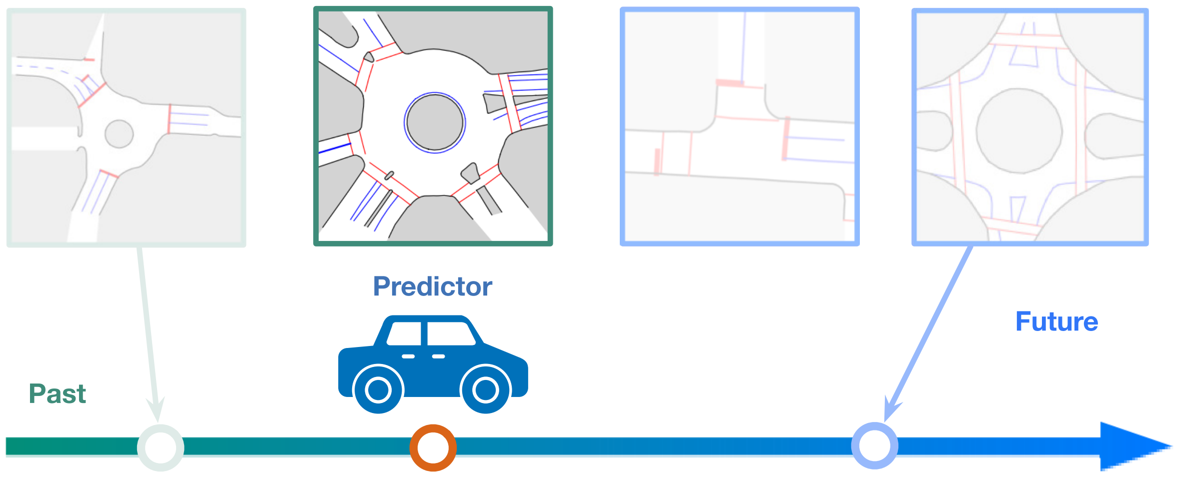


Figure 3.1: Multi-agent interaction behavior prediction in the continual learning setting. The faded scenarios demonstrate the datasets with which the prediction system has been trained before. The datasets of the faded scenarios are no longer fully accessible. The prediction system takes the references information and observations as input to predict the distribution of future trajectories.

interaction prediction. Although several works have investigated the adaptation in behavior prediction [144], it has significant differences with continual learning. Adaptive or transfer learning merely cares about whether the model can adapt to new datasets. In contrast, continual learning concerns the performance on both the new and old datasets [93].

Continual learning has been studied in many different areas, such as computer vision, machine learning, cognitive science, and neuroscience. Most of the current works focus on image classification and reinforcement learning tasks [69, 121, 1]. Several approaches have been developed from different perspectives. For instance, regularization methods [127] were proposed from the optimization view by constraining the difference between old parameters and new learned parameters; pseudo-rehearsal methods like generative replay [143] inspired by neuroscience were proposed to generate the old data from random noises, etc. Though such works have achieved remarkable results on the image classification task, it does not imply that those methods could work in high-dimensional regression problems such as multi-agent interaction behavior prediction. To our knowledge, there is little related research studying the continual learning problems for trajectory prediction in the autonomous driving domain, especially for multi-agent trajectory prediction.

In our work, we adopt the concepts of pseudo-rehearsal approaches [143] in the continual learning literature. We propose a graph-neural-network-based double memory system that can generate similar interaction behaviors compared with the ground truth data. We update our multi-agent interaction behavior predictor with the generated dataset and the new dataset together to reduce the catastrophic forgetting issue. To summarize, our contributions are three folds:

- To the best of our knowledge, our work is the first to investigate the continual learning problems in multi-agent interaction behavior prediction. We show that several current prediction approaches suffer from forgetting problems when data is coming in sequence.
- We propose a graph-neural-network-based continual multi-agent trajectory prediction framework. In this framework, we propose a conditional generative memory model to mitigate catastrophic forgetting. We also design an episodic memory to store the initial graphs of multi-agent trajectories, which are provided to the conditional generative memory model.
- We validate our approach on two datasets. We show that our method effectively mitigates the catastrophic forgetting problems. An ablation analysis is provided to show the necessity and efficiency of the proposed conditional generative memory and episodic memory buffer, especially compared with the method directly using generative replay.

3.2 Related Works

3.2.1 Social Interaction Modeling and Prediction

Behavior and trajectory prediction is crucial for autonomous driving, especially when considering the multi-agent interaction in different scenarios. Several methods have been developed to predict interactive behaviors among multiple pedestrians [59, 94, 4] and vehicles [31, 23, 97]. Some of these methods use learning-based approaches such as generative models [107, 133, 92, 95], probabilistic graphical models [45] and dynamic Bayesian network [152]. In recent years, more sophisticated models such as graph neural network (GNN) and its variations are proposed [64, 83, 164] and applied to trajectory prediction [109, 100, 106]. In some literature, maps have been widely used to provide context information [63, 23, 54]. [23] proposes to use convolutional neural networks as an encoder. [22] focuses on jointly detecting vehicles and predicting future trajectories from high-definition maps and LiDAR information. However, most of these methods are trained and tested on the datasets in which many vehicles are driving on straight roads or roads with similar layouts [24]. In our work, we propose the graph-neural-network-based predictor and memory systems to capture the interaction behaviors. We also leverage the Frenét coordinate system to represent the map information.

3.2.2 Continual Learning

Continual learning has been receiving more attention in recent years. There are three major categories of methods: rehearsal methods [103, 121], regularization methods and architecture methods [1, 67]. Our proposed approach is inspired by the rehearsal-based method. The core idea of this kind of approaches is using a memory system to "remember" the seen data points, and using this memory to reduce catastrophic forgetting problems when we continually receive new datasets. Gradient episodic memory (GEM) [103] and its variant dubbed-average GEM (A-GEM) [25] use episodic memory as the constraints in the optimization. Instead of storing the data into the episodic memory buffer, [143] used a generative model to generate the seen data, and mixed it with the new dataset to optimize the current model. Although several works [136] further develop the idea of generative memory, they are all tested in the image classification or reinforcement learning tasks. To our knowledge, there are few works to validate whether such ideas can also work in regression or more complex tasks. The most related work is generative replay (GR) [143] that uses generative adversarial network (GAN) to generate previous data as memory. This method can work for image classification. However, we show that it is not enough to generate realistic trajectories in our application due to the complex spatial-temporal data structures. From this intuition, we propose a conditional generative model and combine it with a small episodic memory buffer together to generate better trajectories as memory.

3.3 Problem Formulation

Our goal is to train the multi-agent future trajectories predictor given a stream of datasets collected at different scenarios. We firstly formulate the domain problem (i.e., multi-agent trajectory prediction), and then introduce the formulation of the continual learning problem in our application.

3.3.1 Multi-agent Trajectory Prediction

We assume that there are N agents in each case, and the number of agents for each case may vary. We denote the observations of N agents as $\mathbf{o} = \{\mathbf{o}^1, \mathbf{o}^2, \mathbf{o}^3, \dots, \mathbf{o}^N\}$, where $\mathbf{o}^i = [\mathbf{p}^i_{-t_h+1:0}, \mathbf{c}^i]$. $\mathbf{p}^i_{-t_h+1:0} \in \mathbf{R}^{t_h \times d}$ is the i-th agent's history trajectories with t_h time steps. d is the dimension of state $\mathbf{p}$. $\mathbf{c}^i$ represents the waypoints of the i-th agent's reference in the Cartesian coordinate system. We define I^i as the bird-view rasterized image of $\mathbf{c}^i$. $\mathbf{c}^i$ and I^i come from a provided high-definition (HD) map. Our purpose is to predict the future trajectories of multiple agents $\mathbf{y} = \{\mathbf{y}^1, \mathbf{y}^2, \mathbf{y}^3, \dots, \mathbf{y}^N\}$ given the observations $\mathbf{o}$, i.e., the distribution $P(\mathbf{y}|\mathbf{o})$, where $\mathbf{y}^i$ is defined as $\mathbf{p}^i_{1:t_f}$ and t_f is the time horizon of the future trajectory of each agent.

3.3.2 Continual Learning Problem

Under the problem setting of continual learning, we assume that we cannot store *all* previous data. Each experiment is conducted on several datasets collected at different time and different locations. We denote $\mathcal{D}^i$ as the i-th dataset we received. We evaluate the performance of prediction systems given a sequence of datasets $\{\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^M\}$, M is the number of datasets. Notice that if the current dataset is the k-th dataset, we do not store the past datasets $\{\mathcal{D}^1, \ldots, \mathcal{D}^{k-1}\}$.

3.4 Methodology

Our proposed framework is illustrated in Figure 3.2. We firstly introduce the three modules used in our proposed framework: a graph-neural-network-based predictor, an episodic memory buffer and a conditional-variational-autoencoder-based generative memory module. Then we introduce the continual learning strategy, which includes three steps. Our contribution mainly focuses on the framework design and the conditional generative memory module, and this whole approach has the potential to be incorporated with other suitable predictors.

3.4.1 Predictor

Graph Representation of Multi-agent Trajectories Our predictor computes a multi-modal probabilistic multi-agent trajectory distribution using the observation $\mathbf{o}$ of all agents. Given the reference $\mathbf{c}^i$ of each agent from $\mathbf{o}^i$, we transform the trajectory $\mathbf{p}^i$ of the i-th agent into the Frenét coordinate system and denote it as $X^i = \{d^i_{\text{lon}}, d^i_{\text{lat}}\}$, where d^i_{lon} and d^i_{lat} represent the longitudinal position and lateral position in the Frenét coordinate associated with the reference $\mathbf{c}^i$.

To get a feature representation that is invariant to the origin and the direction of the coordinate system, we use only velocity information $\delta^i = \{\dot{d}^i_{\text{lon}}, \dot{d}^i_{\text{lat}}\}$ for each agent i, where $\dot{d}^i_{\text{lon}}$ and $\dot{d}^i_{\text{lat}}$ represent the longitudinal velocity and lateral velocity w.r.t. the reference $\mathbf{c}^i$. For each edge e_{ij}, we define the edge feature as the relative position Δ^{ij} of agent j in the view of agent i. We use the rasterized image I^i to provide the future lane geometry information. Here we recenter the image at the i-th agent's current position and set the y-axis direction of the image as the velocity direction of the i-th agent. We denote the transformed image as $\tilde{I}^i$. We use a feature embedding function to process the aforementioned information to form initial node attributes v^0_i and edge attributes e^0_{ij}. Given a set of trajectory observations, we have:

$$\begin{aligned} v^0_i &= \text{MLP}((\text{CNN}(\tilde{I}^i)||\text{RNN}(\delta^i))), \\ e^0_{ij} &= \text{RNN}(\Delta^{ij}). \end{aligned} \tag{3.1}$$

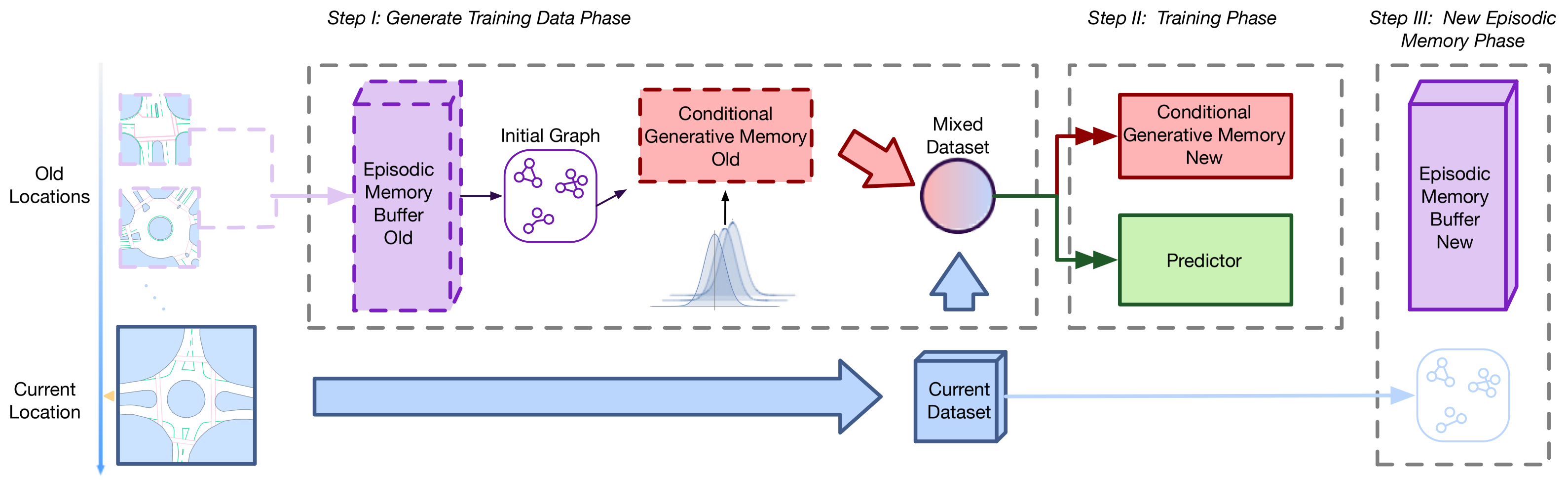

Step I: Generate Training Data Phase
Step II: Training Phase
Step III: New Episodic Memory Phase
Old Locations
Current Location
Episodic Memory Buffer Old
Initial Graph
Conditional Generative Memory Old
Mixed Dataset
Conditional Generative Memory New
Predictor
Episodic Memory Buffer New
Current Dataset

Figure 3.2: The overall framework. Different datasets are collected from several locations. We assume that we cannot store the whole dataset after we optimize our model with it. **Step I**: The purple cube represents the episodic memory, which only stores the initial graph information of the previous scenarios. We sample a batch of initial graphs and use conditional generative memory to generate the trajectories. Then we mix the generated data and the current dataset together. **Step II**: We use the mixed data to train both the conditional generative memory and predictor. **Step III**: We uniformly sample a small portion of the current dataset to store in the episodic memory buffer.

Message Passing Graph Neural Network Following the extracted features $\{v_i^0, e_{ij}^0\}$, a fully-connected graph is constructed to represent the interaction mechanism between different agents. We denote the graph as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_i\}$ denotes the node attributes, and $\mathcal{E} = \{e_{ij}\}$ denotes the edge attributes. At the m-th message passing:

$$\begin{aligned} e_{ij}^m &= f_e^m([v_i^{m-1}, v_j^{m-1}]), \\ v_i^m &= f_v^m(\Phi[j \in \mathbf{N}(v_i)](e_{ij}^m)),\ m = 1, \dots, n. \end{aligned} \tag{3.2}$$

where f_e and f_v are the embedding functions for edges and nodes, respectively. The superscripts of v_i^m, e_{ij}^m, f_v^m, f_e^m denote the m-th message passing. $\Phi[j \in \mathbf{N}(v_i)](\cdot)$ aggregates the information of all the edges e_{ij} between v_i and its neighbors $\mathbf{N}(v_i)$. For the first message passing round, we also aggregate the edge information for edge update. Then we use the attention mechanism similar to Graph Attention Networks (GAT) [164] for the last message passing:

$$\alpha_{ij}^m = \mathrm{softmax}(e_{ij}^m), v_i^m = \sigma(\sum_{j \in \mathbf{N}(v_i)} \alpha_{ij}^m \mathbf{W} v_j^{m-1}). \tag{3.3}$$

Multi-modal Prediction Layer In order to capture the multi-modal interactive behaviors, we use a Gaussian mixture model (GMM) to represent the future predicted actions at different time steps:

$$\begin{aligned} & w_j = \mathrm{softmax}(f_w^j(v_i^n)), \\ & \mu_j = f_\mu^j(v_i^n), \Sigma_j = f_\Sigma^j(v_i^n), \\ & \{\dot{d}_{\mathrm{lon},0:t_f-1}, \dot{d}_{\mathrm{lat},0:t_f-1}\} \sim \sum_j w_j \mathcal{N}(\mu_j, \Sigma_j), \end{aligned} \tag{3.4}$$

where w_j, μ_j, and Σ_j denote the weight, mean, and covariance of the j-th Gaussian function, respectively. Each Gaussian function represents the distribution of the future actions.

After obtaining the action information for each agent i, we use a first-order integrator to get the position X^i in the Frenét coordinate system. Then the predicted trajectories are transformed back to the Cartesian coordinate system according to $\mathbf{c}^i$. This procedure incorporates the road routing information directly. The loss $\mathcal{L}_{\mathrm{P}}(\psi; \mathcal{D})$ is the negative log-likelihood:

$$-\mathbb{E}_{(X,I)\sim\mathcal{D}}[\log P_\psi(X_{1:t_f} | X_{-t_h+1:0}, I)], \tag{3.5}$$

where ψ is the parameter of the predictor.

3.4.2 Double Memory System

The traditional pseudo-rehearsal methods such as the generative replay method [143] generate samples from noises. In our case, it is not enough to generate high-quality full-time-length multi-agent trajectories purely from high-dimensional noises due to the complicated spatial-temporal data structure of multi-agent trajectories. Hence, we propose a graph-neural-network-based conditional generative replay module, which is conditioned on some initial information given by the proposed episodic memory buffer.

Episodic Memory Buffer Episodic memory is defined as a kind of memory of everyday events which could be conjured or explicitly stated. It is the collection of past experiences that occurred at particular times and places [140]. In our application, one "experience" should be one case of multi-agent interaction trajectories. Some works in image classification and reinforcement learning have been proposed to construct the episodic memory by storing a fixed percentage of original data. We intend to reduce the storage of full-time-length trajectories data by only storing some initial information of the trajectories. This reduction is significant, especially when the length of the trajectory is large. We define such initial information as an initial graph $\mathcal{G}_{init} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}^i = \{X_0^i, X_{-t_h+1}^i, X_{t_f}^i, I^i, \mathbf{c}^i\}$ and the edge attribute $E^{ij} \in \mathcal{E}$ denotes whether there is an edge between node i and node j. There is no edge between two agents if their references do not have any intersection or the vehicle on one reference cannot shift to the other reference due to traffic rules. Here X_0^i is the current state defined in previous section, $X_{-t_h+1}^i$ is the state at H time steps before the current state, and $X_{t_f}^i$ is the goal position. We intend to generate the interaction behaviors conditioned on this initial information. It is reasonable that the interaction behaviors could be "conjured" if we know the trajectory tendency (given X_{-t_h+1}) and goal (given X_{t_f}) at the current state X_0. We will show in the experiments that the performance of our methods with this episodic memory buffer is significantly improved compared with directly using the vanilla generative replay methods.

Conditional Generative Memory The objective of the conditional generative memory is to solve $P(X|\mathcal{G}_{init})$, where $X = X_{-t_h+1:t_f}$ means the whole trajectories of all agents. We design a graph-neural-network based conditional variational autoencoder (CVAE) to estimate this distribution. The graph neural networks are used to capture the interaction mechanism between agents. By leveraging the conditional variational inference, we can generate realistic trajectories given the initial graph information. The overall conditional generative memory model structure is illustrated in Figure 3.3.

Encoder The encoder is used to map the trajectories X to the latent variables $\mathbf{z}$, namely the posterior distribution $Q(\mathbf{z}|X, \mathcal{G}_{init})$, where $\mathbf{z} = \{\mathbf{z}^i\}_{i=1:N}$ and $\mathbf{z}^i$ is the Gaussian random variable for the i-th agent. For each agent i, the reference image I^i and the initial state information $\{X_0^i, X_{-t_h+1}^i, X_{t_f}^i\}$ are encoded by a convolutional neural network (CNN) and a multiple-layer perceptron (MLP) respectively. They are served as the conditional inputs. The trajectories $X_{-t_h+1:t_f}^i$ are encoded by a recurrent neural network (RNN). We use an MLP to integrate these three features and construct a graph neural network to process the interaction behaviors. Each node i of the GNN module outputs the mean and the covariance of $\mathbf{z}^i$.

Decoder The decoder is used to map the latent variables $\mathbf{z}$ to the trajectories X, namely $P(X|\mathbf{z}, \mathcal{G}_{init})$. Similar to the encoder, we use a CNN and an MLP to process the reference image I^i and the initial state information for each node. Then we use a GNN to capture

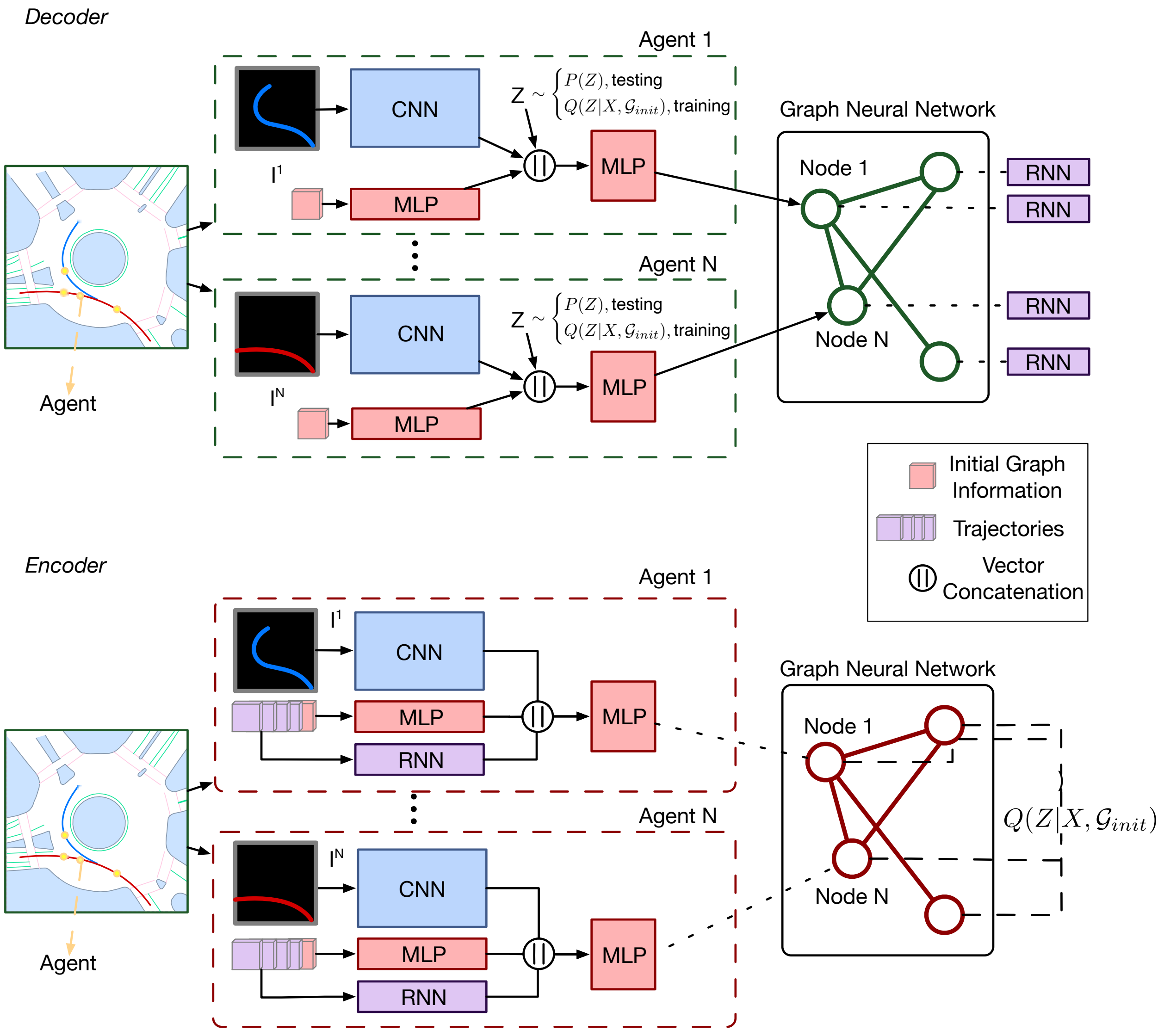


Figure 3.3: The conditional generative memory model. The blue boxes, red boxes and the purple boxes represent CNN, MLP and RNN respectively. I^i demonstrates the rasterized image of the i-th agent's reference.

the interaction pattern and use an RNN to output the trajectories X. The input $\mathbf{z}$ of the decoder is sampled from the output of the encoder $Q(\mathbf{z}|X, \mathcal{G}_{init})$ during training and sampled from the prior distribution $P(\mathbf{z})$ during testing. $P(\mathbf{z})$ is the standard Gaussian distribution. The encoder and decoder are trained jointly. The training loss of the CVAE $\mathcal{L}_{\text{G}}(\theta, \phi; \mathcal{D})$ is:

$$\begin{aligned} &-\mathbb{E}_{(X,\mathcal{G}_{init})\sim\mathcal{D}}\mathbb{E}_{Q_\phi(\mathbf{z}|X,\mathcal{G}_{init})}[\log P_\theta(X|\mathbf{z},\mathcal{G}_{init})] \\ &\quad+\beta\mathbb{E}_{(X,\mathcal{G}_{init})\sim\mathcal{D}}\mathcal{KL}[Q_\phi(\mathbf{z}|X,\mathcal{G}_{init})||P(\mathbf{z})], \end{aligned} \tag{3.6}$$

where ϕ and θ are the parameters of the encoder $Q_\phi(\mathbf{z}|X,\mathcal{G}_{init})$ and the decoder $P_\theta(X|\mathbf{z},\mathcal{G}_{init})$. β is a parameter to adjust the importance of the second regularization.

Once we get the initial graph $\mathcal{G}_{init}$, we can sample r trajectories from the decoder $P_\theta(X|\mathbf{z},\mathcal{G}_{init})$ by sampling r times of different $\mathbf{z}$ from the standard Gaussian distribution. Similar to the predictor, we can transform X^i to $\mathbf{p}^i$ by using the reference $\mathbf{c}^i$.

3.4.3 Training Strategy

There are three procedures in total: generate the training batch, train predictor and generator, and form new episodic memory. The framework is illustrated in Figure 3.2.

Step I: First, we need to construct the mixed training dataset at the k-th scenario. We uniformly sample $|\mathcal{D}^k|/r$ initial graphs $\mathcal{G}_{init}$ from the episodic memory buffer $\mathcal{B}^{k-1}$. For each initial graph $\mathcal{G}_{init}$, we randomly sample r times of different $\mathbf{z}$ and use the decoder to generate r multi-agent trajectories. We denote the generated data as $\hat{\mathcal{D}}^k$. Hence, $\hat{\mathcal{D}}^k$ includes the generated real-like data of the past scenarios $\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^{k-1}$, which serves as the replay data for the predictor and generator.

Step II: After we get the current new dataset $\mathcal{D}^k$ and the generated dataset $\hat{\mathcal{D}}^k$ from Step I, we optimize our predictor and generative memory separately.

$$\begin{aligned} \psi^k &= \min_{\psi} \gamma\mathcal{L}_{\mathrm{P}}(\psi;\mathcal{D}^k) + (1-\gamma)\mathcal{L}_{\mathrm{P}}(\psi;\hat{\mathcal{D}}^k), \\ \theta^k, \phi^k &= \min_{\theta,\phi} \gamma\mathcal{L}_{\mathrm{G}}(\theta,\phi;\mathcal{D}^k) + (1-\gamma)\mathcal{L}_{\mathrm{G}}(\theta,\phi;\hat{\mathcal{D}}^k), \end{aligned} \tag{3.7}$$

where γ is the hyperparameter to determine the importance of the current scenario. The current scenario is more important if γ is larger.

Step III: The final step is to construct the new episode memory buffer $\mathcal{B}^k$. We randomly sample a small portion of the whole number of cases in the new dataset and store their initial graphs into the episodic memory buffers. In our application, we only allow storing the initial graphs of ten percent of the cases for each $\mathcal{D}^k$. The three steps above will be repeatedly executed as long as the new dataset is available.

3.5 Experiments

We conduct our experiments with INTERACTION dataset [183] collected in the USA, InD dataset [19] and RounD dataset [85] collected in Germany. First, we show that baselines indeed suffer from catastrophic forgetting and our method can reduce it. Then we conduct an ablative analysis to show that our method is better than applying the generative replay method directly and our approach can achieve similar performance with significantly less memory compared with real data replay.

3.5.1 Dataset and Preprocessing

To evaluate our approach's effectiveness, we use INTERACTION dataset, InD and RounD dataset. The aforementioned datasets include many complicated interaction scenarios like roundabouts and intersections, which is suitable to test the multi-agent interaction behaviors. Also, the cases of these datasets are divided by their scenarios originally. Since the driving behaviors are significantly different between different scenarios, these datasets are suitable to test the continual learning performance. We selected four scenarios for each dataset, and each scenario has a similar number of cases. Both datasets provide the HD map from which we can extract the references information. For both datasets, we use four time steps as observation and predict eight time steps of future trajectories. The interval between time steps is 0.5s for INTERACTION datasets and 0.4s for RounD/InD datasets.

3.5.2 Metrics

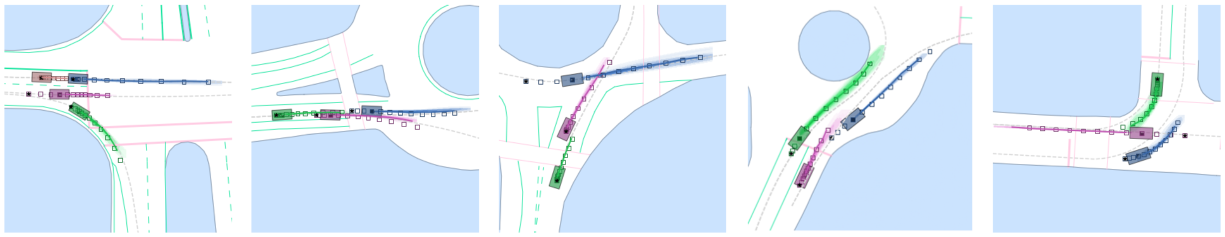
Prediction Metrics To evaluate the probabilistic prediction, we use the similar metrics in [96], the minimum average displacement error (ADE) and minimum final displacement error (FDE) over k samples:

$$\begin{aligned} \text{ADE} &= \frac{1}{Nt_f} \sum_{i=1}^{N} \min_{j \in \{1,\ldots,k\}} \{ \sum_{t=1}^{t_f} ||\mathbf{p}_t^i - \hat{\mathbf{p}}_t^i(j)||_2 \}, \\ \text{FDE} &= \frac{1}{N} \sum_{i=1}^{N} \min_{j \in \{1,\ldots,k\}} \{ ||\mathbf{p}_{t_f}^i - \hat{\mathbf{p}}_{t_f}^i(j)||_2 \}. \end{aligned} \tag{3.8}$$

Continual Learning Metrics For evaluating the continual learning performance, we adopt the concepts mentioned in [43] and adapt them to our application. Here we define two metrics: average error (AER) and average forgetting (FGT). We denote the testing error (ADE / FDE) at the j-th scenario after training on the i-th scenario as $R_{i,j}$. Let M be the number of scenarios, AER and FGT are defined as:

$$\begin{aligned} \text{AER} &= \frac{1}{M(M+1)/2} \sum_{j \le i}^{M} R_{i,j}, \\ \text{FGT} &= \frac{1}{M(M-1)/2} \sum_{i=2}^{M} \sum_{j<i}^{M} R_{i,j} - R_{j,j}. \end{aligned} \tag{3.9}$$

AER measures the average performance of all scenarios across all training datasets. FGT measures the average decrease of performance on the old scenarios after seeing the new scenarios, which can represent the degree of catastrophic forgetting. The greater the value of FGT is, the more forgetting occurs. Please refer to [43] for more details.



(a) MA (b) FT (c) SR (d) EP-R (e) EP-T

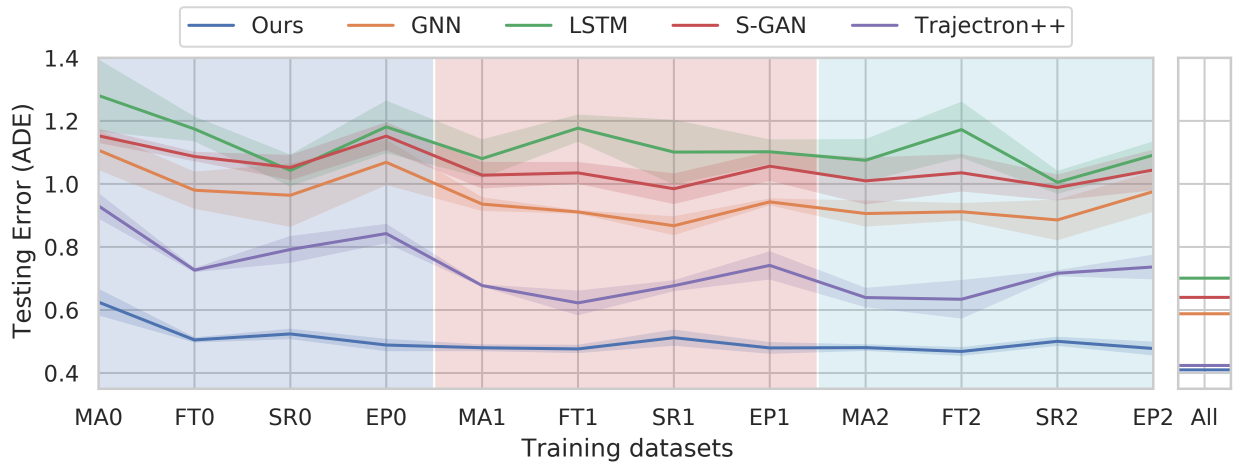
Figure 3.4: The visualization of multi-agent trajectory prediction results. We represent the ground truth trajectories as box markers. The star markers represent the starting points of the historical trajectory. We use kernel density estimation (KDE) to fit the sampled trajectories, shown as density maps in the pictures. The grey dash lines are the references of each vehicle. The predicted trajectory with the minimum ADE is illustrated in a solid line.

3.5.3 Overall Performance Evaluation

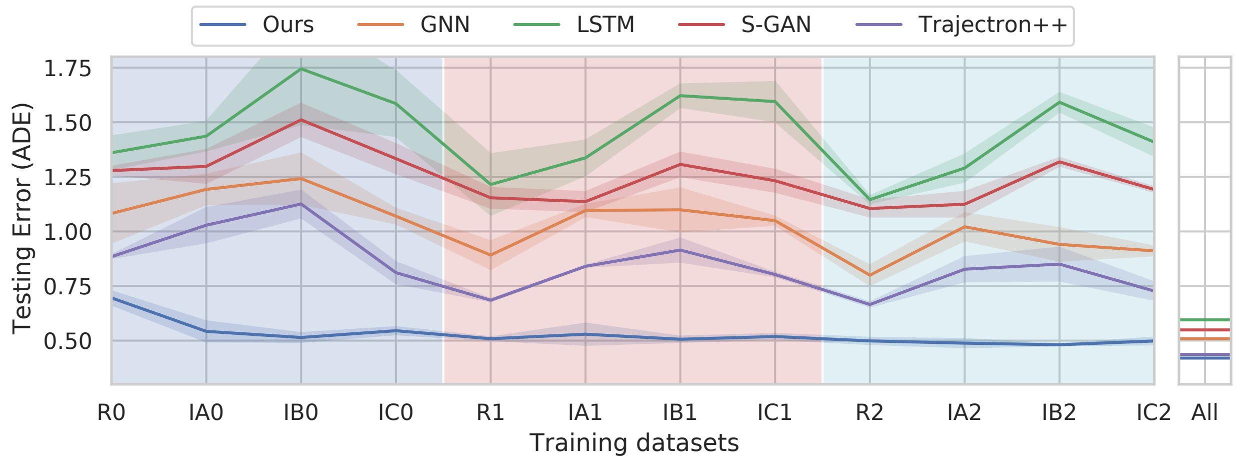
Experiment Settings For this experiment, we use four different locations from both the INTERACTION dataset and the RounD/InD dataset. The cases of each location are divided into three parts equally by time stamps since we intend to simulate that we repeatedly collect data from different locations. Different parts are regarded as different datasets. The four scenarios from the INTERACTION dataset are MA, FT, SR, and EP (Including EP-R and EP-T). The sequence of tasks is $\mathcal{D}^{\mathrm{MA0}}, \mathcal{D}^{\mathrm{FT0}}, \mathcal{D}^{\mathrm{SR0}}, \mathcal{D}^{\mathrm{EP0}}, \mathcal{D}^{\mathrm{MA1}}, \ldots, \mathcal{D}^{\mathrm{EP2}}$, where under the same scenario, e.g., $\mathcal{D}^{\mathrm{MA0}}, \ldots, \mathcal{D}^{\mathrm{MA2}}$ means three different datasets collected at different time at MA. Our framework will be trained in this order. The four scenarios from the RounD/InD dataset are R, IA, IB, and IC. R indicates Neuweiler of the RounD dataset. IA, IB, and IC indicate Bendplatz, Frankenburg, and Heckstrasse in the InD dataset. We evaluate the prediction performance for each scenario after we train the model with the current dataset. We compare our method with several baselines and recent works including long-short term memory (LSTM), GNN, S-GAN [59] and Trajectron++ [138]. We provide the same input information for all the methods. LSTM is equipped with a GMM model in order to generate the probabilistic prediction. GNN is a simple message passing graph neural network without any special design. We trained five models randomly to get the means and variances of performance.

Qualitative Analysis We illustrate the prediction performance of several representative cases with different numbers of agents at different scenarios of the INTERACTION dataset in Figure 3.4. We demonstrate that we can get accurate prediction at all the scenarios after training on a sequence of datasets. Furthermore, we find that if the references are relatively straight, the variance of predicted trajectories is small.

Quantitative Analysis We calculate the average performance of the testing data across all scenarios. This value represents the overall performance of the method after training on a

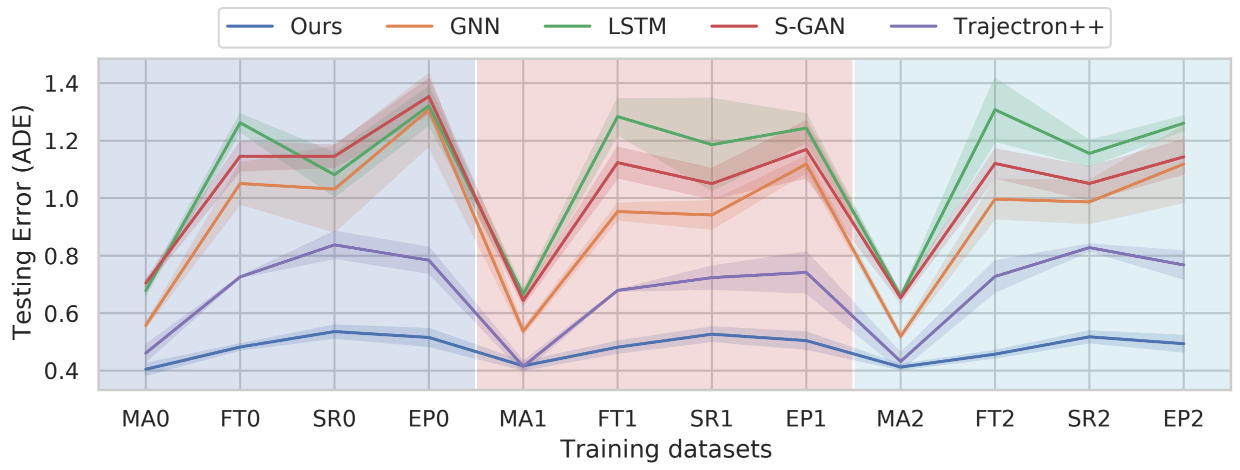


(a) INTERACTION dataset

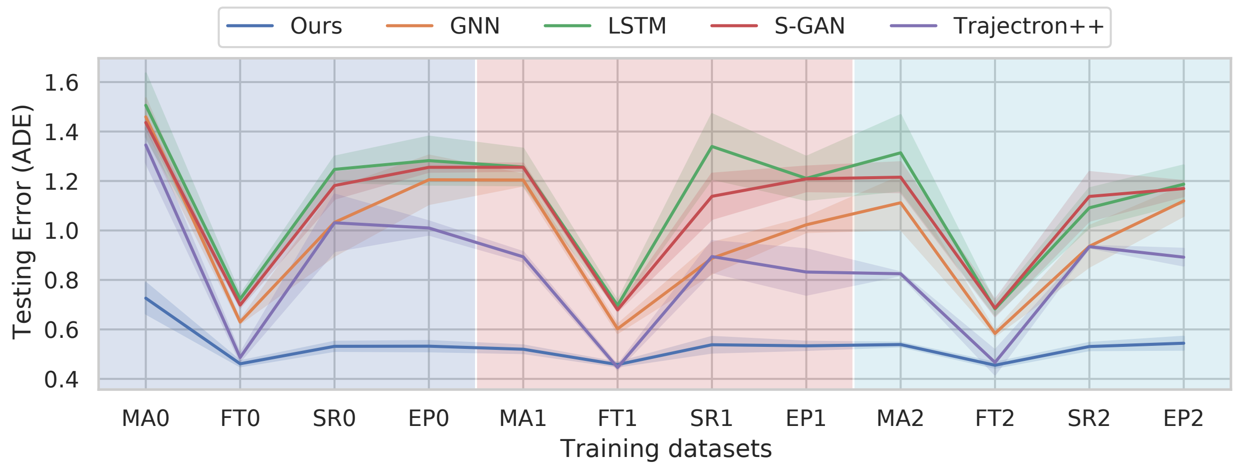


(b) RounD/InD dataset

Figure 3.5: Evaluation of overall performance (tested with a mixed dataset of all scenarios) in the continual learning setting. The column “All” means that the methods are trained with the mixed datasets of different scenarios, i.e., the non-continual learning setting.



(a) Evaluation on MA



(b) Evaluation on FT

Figure 3.6: ADE for selected scenarios, MA and FT in INTERACTION dataset. Models are tested on a particular scenario to show the catastrophic forgetting issue clearly.

sequence of datasets. We illustrate the ADE error for both INTERACTION and RounD/InD datasets in Figure 3.5.

In Figure 3.5, the other baseline models have bad performance even they could be trained on the same scenario again. We observed that baselines have large errors during the whole testing phase. It implies that it may only have good performance when the training scenario and testing scenario are the same. This conjecture is validated in Figure 3.6.

Notice that the average error is for the data of all scenarios. The first point at MA0 means that we train models on MA0 dataset and evaluate on a mixed testing data of all scenarios.

We observe that our model has better performance at MA0 than the other baselines. This advantageous performance of our predictor could be attributed to the reason that we directly use the reference information $\mathbf{c}$ (by using Frenét coordinate transformation). It enables our predictor to have a relatively better performance in the other unseen scenarios. While the other baseline models incorporate the information of references from rasterized images, they have to extract the useful information from the image directly. It also can be validated in Figure 3.6b. For instance, in Figure 3.6b, we find that our approach can have a better performance on FT than the others even though we only used the first dataset MA0 for training.

To find out why baseline models have bad performance when the training datasets come in a sequential way, we investigate the prediction performance on each scenario. We illustrate the prediction error of two representative scenarios of the INTERACTION dataset in Figure 3.6. We notice that the testing performances of all the models are good when the current training and the testing scenarios are the same. For instance, the state-of-the-art model, Trajcetron++, can achieve a very low prediction error when testing on the same scenario as the current training one. It implies that the bad overall performances in Figure 3.6 are not only caused by the capability of the model itself but also attributed to the catastrophic forgetting issues. For instance, in Figure 3.6a, when we test our prediction performance on MA, the errors of baseline models rapidly increase when they are trained on the following scenarios, which is called the catastrophic forgetting phenomenon. Our model significantly reduces the catastrophic forgetting problems compared with all the methods without continual learning, i.e., the fluctuation in Figure 3.6 is significantly smaller than the others.

We also compare the performance of different approaches between the continual learning setting and non-continual learning setting, i.e., training on the entire dataset of all scenarios. The non-continual learning setting (column "All" in Figure 3.5) can serve as the lower bound of the error of the models in the continual learning settings. In Figure 3.5 column "All", we see that Trajectron++ and Ours can achieve the similar best performance when training on the entire datasets. It demonstrates that our predictor module can achieve state-of-the-art performance. In addition, comparing the testing performance at column EP2 in which the models have seen all the datasets with the testing performance at column "All" in Figure 3.5 (a), our method can also achieve similar performance in the continual learning setting to the non-continual learning setting. It shows that our method reduces catastrophic forgetting remarkably. Meanwhile, we observe large gaps between the continual learning and the non-continual learning setting for each baseline.

3.5.4 Ablation Analysis

In this section, we demonstrate the effectiveness of our approach by comparing the following ablation models: i) our framework (Ours); ii) our predictor without double memory system (Ours w/o CL); iii) our predictor with vanilla generative replay (GR), i.e., the full-length trajectories are generated purely from a Gaussian random variable; iv) our predictor with elastic weighted consolidation (EWC). In order to show that the comparisons are invariant to the order of scenarios, we randomly sample four different orders. Without loss of generality, we use the whole dataset for each scenario. We show the results in Table 3.1 and Table 3.2 for both datasets.

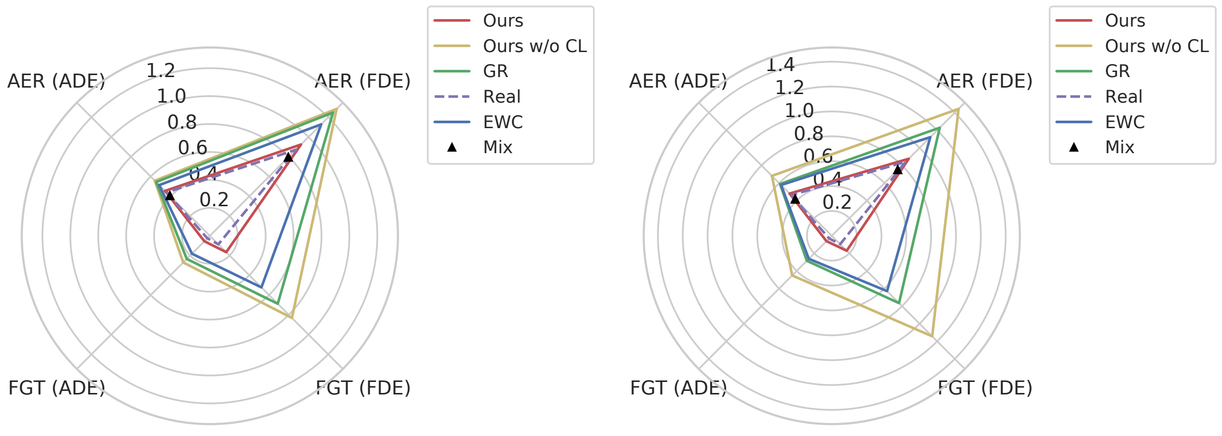
Table 3.1: The Performance of ADE / FDE (meters) in Ablation Analysis for INTERACTION Dataset

	Order 1		Order 2	
	AER	FGT	AER	FGT
Ours	**0.441/0.900**	**0.058/0.189**	**0.464/0.952**	**0.045/0.122**
Ours w/o CL	0.558/1.247	0.249/0.814	0.603/1.431	0.322/0.952
GR	0.543/1.278	0.270/0.836	0.567/1.263	0.243/0.718
EWC	0.502/1.109	0.198/0.609	0.543/1.235	0.217/0.661
	Order 3		**Order 4**	
	AER	FGT	AER	FGT
Ours	**0.466/0.891**	**0.060/0.188**	**0.449/0.956**	**0.058/0.166**
Ours w/o CL	0.577/1.265	0.245/0.755	0.556/1.237	0.265/0.806
GR	0.534/1.198	0.179/0.486	0.537/1.240	0.245/0.731
EWC	0.512/1.115	0.129/0.308	0.489/1.084	0.175/0.506

We demonstrate that the results are consistent under different orders. From Table 3.1 and 3.2, we show that Ours improves at least 19% in ADE for AER and 73% in ADE for FGT compared with Ours w/o CL. It implies that our conditional generative memory system is effective, and only using the Frenét-based predictor is not enough to reduce the catastrophic forgetting. Comparing Ours w/o CL with GR, we find that GR only slightly improves the performance. It shows that directly using generative replay does not work well in our application. This comparison illustrates the necessity of the proposed episodic memory buffer to store the initial graph information. Although GR and EWC do not use replay buffer, they cannot mitigate catastrophic forgetting effectively in our application. In Figure 3.7, we compute the average performance across different orders and demonstrate the comparison. We also include the model (Real), which uses full-length ground truth data replay to serve as the ideal performance bound of our methods. The difference between Real and Ours is that Ours only stores initial graph information and generates the whole trajectories by conditional generative memory, while Real stores the full-length ground truth multi-agent trajectories. This difference allows Ours reduces about 87% memory usage compared with Real. Meanwhile, Ours can achieve almost the same performance as Real (shown in Figure 3.7). It demonstrates our method's effectiveness and shows that utilizing generative models is possible to reduce catastrophic forgetting in multi-agent trajectory prediction.

Table 3.2: The Performance of ADE / FDE (meters) in Ablation Analysis for RounD/InD Dataset

	Order 1		Order 2	
	AER	FGT	AER	FGT
Ours	**0.509/0.938**	**0.112/0.284**	**0.466/0.832**	**0.036/0.113**
Ours w/o CL	0.669/1.376	0.424/1.083	0.674/1.453	0.442/1.135
GR	0.625/1.310	0.320/0.841	0.568/1.177	0.243/0.674
EWC	0.617/1.236	0.325/0.791	0.548/1.028	0.205/0.464
	Order 3		**Order 4**	
	AER	FGT	AER	FGT
Ours	**0.463/0.846**	**0.029/0.080**	**0.484/0.889**	**0.067/0.203**
Ours w/o CL	0.740/1.609	0.559/1.407	0.644/1.337	0.379/0.947
GR	0.576/1.203	0.288/0.786	0.590/1.226	0.286/0.762
EWC	0.558/1.085	0.227/0.579	0.576/1.116	0.278/0.665



(a) INTERACTION dataset

(b) RounD/InD dataset

Figure 3.7: Average performances across different orders for both datasets. "Mix" shows the prediction error of models trained on the whole datasets including all scenarios, i.e. the common settings without continual learning. Details and memory comparison analysis are in Section 3.5.4.

3.6 Implementation Details

Our predictor and conditional generative memory are trained using Adam optimizer with the batch size 64. We use 0.0005 as the learning rate for each task. The MLP modules have three fully connected layers with the activation function ReLU. The RNN modules use the GRU blocks. The CNN modules have five convolution layers with channels $16, 32, 64, 128, 256$. The GNN modules have two rounds of message passing. The hidden dimension of our models is 256. For the conditional generative memory, the dimension of the Gaussian random variable z^i for each node i is 4 and the hyperparameter β in the VAE loss is 1.0. For the predictor, the Gaussian mixture model has 4 kernels, and all of the methods sample 20 trajectories to calculate the metrics. We use $\gamma = 0.5$ for training.

3.7 Chapter Summary

How to continually learn a good trajectory predictor is a potential problem in the near future. In this chapter, we attempt to reduce the catastrophic forgetting in the complicated multi-agent spatial-temporal prediction. We give an approach based on graph representation and the conditional generative memory model to show the possibility of realizing continual learning in the multi-agent trajectory prediction tasks. Such a problem is under-investigated since most of the existing continual learning works focus on image classification and reinforcement learning while relational reasoning and feature representation appeal more attention in the trajectory prediction literature. Hence, this work paves the way for both the continual learning and the trajectory prediction communities. For the continual learning community, we demonstrate that continual learning can also work in high-dimensional regression tasks. For the trajectory prediction community, we show that there is an elegant solution to deal with sequentially increasing data. This work also raises many interesting questions and future directions including but not limited to i) Is there any way to completely abandon the episodic memory buffer? ii) Is it possible to design a predictor which has good zero-shot prediction in new scenarios? We believe that solving such questions can improve performance in the future.

Chapter 4

Uncertainty Analysis and Valuable Scenario Selection for Vehicle Trajectory Prediction

4.1 Introduction

Vehicle trajectory prediction has a pivotal role in autonomous driving scenarios. The safety of an autonomous vehicle system should access an accurate trajectory prediction of the surrounding traffic participants, such that the following decision-making, planning, and control modules can guarantee safety with a high probability. There is a growing body of literature investigating how to improve vehicle trajectory prediction from different perspectives. Several works focus on directions such as multi-agent interaction [109, 100, 151], multi-modality [133, 92, 95], and conditioned prediction [159], etc. There are also some works investigating how to design more efficient representations to capture more information, including the high-definition map and traffic rules [54, 58]. Recently, people are getting more interested in obtaining accurate vehicle trajectory prediction efficiently, especially about whether the existing approaches can also have good performance when the testing scenarios are different from the training scenarios, i.e., domain shift problems. Such problems can be investigated from different application settings such as transfer learning, continual (lifelong) learning, and active learning. One of the common problems of such application settings is that we need to know the difference between the source domain (training domain) and the target domain (testing domain).

Uncertainty estimation [89, 78, 55] has been investigated for decades in different areas, such as computer vision, natural language processing, etc. There is also some existing work [29, 112, 154] focusing on how to obtain a confidence score per predicted trajectory given one case. However, such work can only provide the relative confidence score given the predicted possible future trajectories. We still do not know how confident the current predictor has for one case, especially when the case is out of distribution. Thanks to the recently released Shifts Dataset [111], which provides an evaluation pipeline for investigating the uncertainty estimation of domain shift problems, it becomes possible to systematically investigate the uncertainty estimation of domain shift problems for vehicle trajectory prediction. Although several works have been proposed and evaluated on Shifts Dataset by using different uncertainty quantification techniques, such as Gaussian process [38, 90], ensemble models [73, 89, 52], and direct error prediction [72], etc., they do not provide experiment results about how different features contribute to uncertainty estimation. Also, the uncertainty estimation procedure in the approaches mentioned above is usually incorporated with the predictors, i.e., we have to know the details of predictors if we want to get access to the uncertainty score. It may not be legal or possible for a third party, such as an insurance company or department of transportation, to require detailed information about a commercial product from an autonomous driving company to evaluate. Therefore, in our work, we propose a predictor-agnostic direct-error-prediction-based uncertainty estimation framework to provide an analysis of the importance of different features: i) static information (maps, traffic rules, etc.) and ii) agent information (traffic participants' positions, velocities, etc.). We find that only using static information can achieve an acceptable uncertainty estimation results. Based on this observation, we conduct an experiment to show how the proposed framework can improve transfer learning training efficiency by only using static information. The proposed pipelines are shown in Figure 4.1. In summary, we highlight contributions as follows:

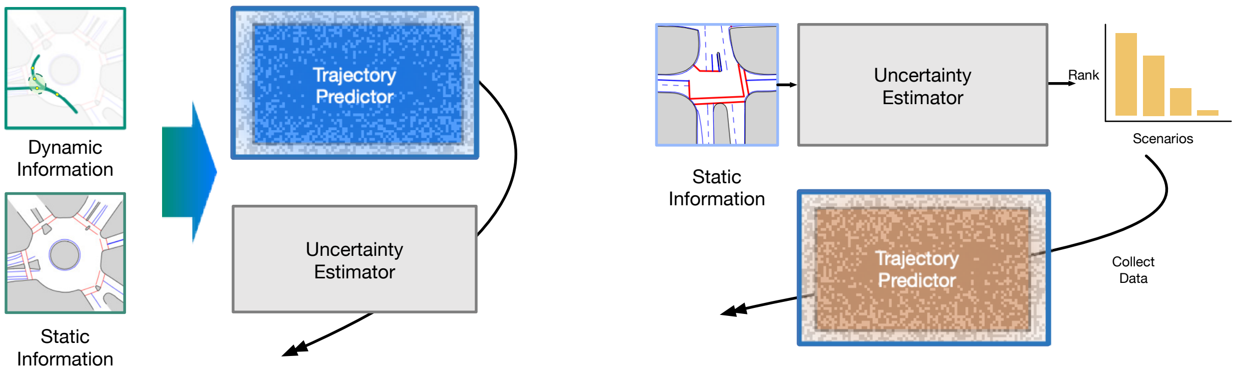


Figure 4.1: The proposed uncertainty estimation pipelines. The left figure illustrates the predictor-agnostic uncertainty estimation procedure. The details of the predictor (stippled blue box) are assumed unknown. In the right figure, we show how we finetune the predictor with the pre-trained uncertainty estimator. The uncertainty estimator only utilizes static information.

- We propose a predictor-agnostic direct-error-prediction-based uncertainty estimation framework for vehicle trajectory prediction. Potentially, The proposed framework can be utilized with any black-box predictors.

- We conduct the experiments with a vehicle trajectory prediction distribution shift dataset and compare the importance of static and dynamic features to uncertainty estimation in the vehicle trajectory prediction application with a transformer-based predictor.
- Based on the analysis, we propose a simple data selection algorithm that only leverages the static information and show that our proposed approach can improve the training efficiency compared with the approach trained with randomly selected data.

4.2 Related Works

4.2.1 Vehicle Trajectory Prediction

Several approaches have been proposed to solve the vehicle trajectory prediction with different traffic participants, such as vehicles [23] and pedestrians [59, 137, 4, 165]. In the earlier literature, probabilistic inference approach such as graphical models [45] and dynamic Bayesian network [77, 152] are proposed. Lately, several learning-based techniques such as generative models [133, 92, 95] are applied. Recently, some approaches based on graph neural network (GNN) [64, 83, 164] have been utilized in vehicle trajectory prediction [109, 100, 151]. Such methods can represent the multi-agent interaction efficiently. How to get good representation is another important direction. For instance, how to use high-definition maps efficiently is investigated in [63, 23, 54]. The authors introduce a convolutional-neural-network-based method in [23]. An approach to jointly detecting vehicles and predicting future trajectories leveraging high-definition maps and LiDAR information is introduced in [22]. Also, several conditional trajectory prediction methods [159, 117] are proposed.

4.2.2 Uncertainty Estimation

Uncertainty estimation has been widely investigated from both theoretical and application perspectives. There are several different approaches to achieving uncertainty estimation. Ensemble methods [89, 88, 130, 73, 34] are simple and strong techniques. For instance, [89] introduces a simple deep ensemble framework that currently has become a strong baseline. Otherwise, the authors in [73] propose to estimate uncertainty by considering diversity in ensemble predictions. Gaussian process [38, 90] can also be used for uncertainty quantification. [90] proposes a distance-preserved Gaussian process for uncertainty estimation. Also, several Bayesian deep learning methods [78, 116] are proposed to quantify the uncertainty. For the uncertainty estimation with vehicle trajectory prediction, [52] proposes using ensemble models to get the uncertainty score for prediction or imitative planning. Such methods are also used in the Shifts dataset. [129] uses spectral normalized Gaussian process [90] to obtain uncertainty; [128] directly produces an uncertainty score along with the predicted trajectories, and using the prediction error supervises the uncertainty score. However, the methods mentioned above are not designed to analyze what regimes of features have more influence on the uncertainty estimation. In our work, we adopt the direct-error-prediction strategy. We propose a framework that can select different types of features. By using this framework, we provide a detailed analysis of uncertainty estimation in the vehicle trajectory prediction application.

4.3 Problem Formulation

4.3.1 Vehicle Trajectory Prediction

Vehicle trajectory prediction aims to provide several possible future trajectories of one target vehicle given historical information, which includes static information and dynamic agent information. Static information includes high-definition maps, traffic rules, etc. Although certain traffic rule information, such as traffic signals, is not static in general, it usually has a fixed pattern and can be obtained directly without deploying data collection devices. Hence, this work will categorize such information without uncertainty as static non-agent information. The dynamic agent information consists of the motion information (e.g., position, velocity, acceleration, etc.) of homogeneous and heterogeneous agents (e.g., vehicles, pedestrians, cyclists, etc.). Such information usually has high uncertainty and cannot be obtained without deploying data collection devices. In this chapter, we assume that we can obtain the historical information $X \in \mathbb{R}^{d_X \times H \times W}$ in the image format, where H and W are the height and width, respectively, and d_X is the number of features. Also, we denote $X = [X_s \in \mathbb{R}^{d_s \times H \times W}, X_a \in \mathbb{R}^{d_a \times H \times W}]$, where X_s denotes the static information, and X_a denotes the agent information. We denote the T_f-time-step future trajectories as $Y = [y_1, y_2, \dots, y_t, \dots, y_{T_f}]$, where $y_t \in \mathbb{R}^2$ is the position of the target vehicle at time step t. The objective of vehicle trajectory prediction is to estimate the distribution $p(Y|X)$.

4.3.2 Uncertainty Estimation

We assume that our predictor is trained over a training dataset $\mathcal{D}_{\text{train}}$ collected at locations A. The objective of uncertainty estimation is to get a score to indicate the confidence of the prediction given a specific predictor. The performance will be evaluated on a testing dataset $\mathcal{D}_{\text{test}}$ including the data collected at locations B, which are different from A. We denote the uncertainty score for each case as $u(X)$, and the performance of uncertainty estimator on $\mathcal{D}_{\text{test}}$ as $s(\mathcal{D}_{test})$. We adopt the error-retention curve (ERC) [111] as the evaluation metric for uncertainty estimation. Such metric will only depend on the relative value of $u(X)$. Please see the details of the metric in Section 4.7.1.

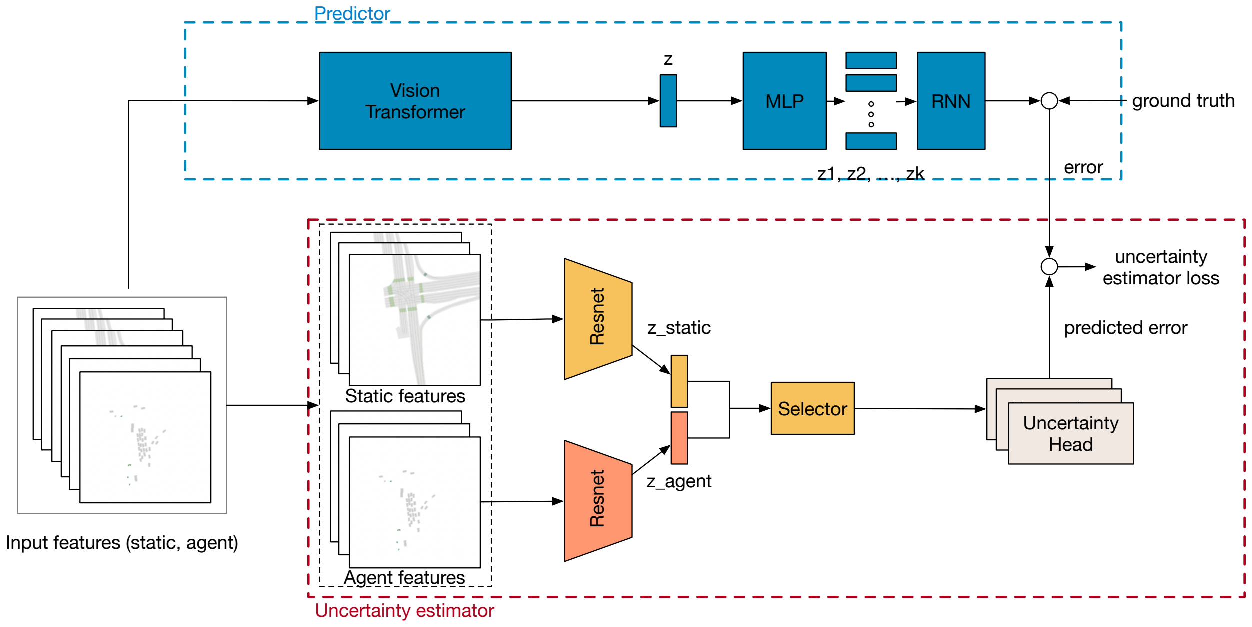


Figure 4.2: Overall framework. The dashed blue and red blocks indicate the predictor module and uncertainty estimator module, respectively. Input features are divided into two parts: static information and agent information. Different colors of ResNets represent different parameters.

4.4 Methodology

The proposed overall framework consists of two components: predictor and uncertainty estimator. We illustrate the proposed overall framework in Figure 4.2. The details of the proposed predictor are introduced in Section 4.4.1, and the proposed uncertainty estimator is introduced in Section 4.4.2. In Section 4.4.3, the training losses and procedure are provided. Please see the implementation details in Section 4.7.

4.4.1 Predictor

In this chapter, we use an image-based predictor, which can achieve comparable performance with the state-of-the-art approach [128] provided by the Shifts Challenge [111]. Recently, transformers [47, 163] have been proved effective in both natural language processing and computer vision. There exists several works [128, 118] using a transformer in trajectory prediction problems. Similar to [128], we use vision transformer [47] as the encoder to get the latent representation $z \in \mathbb{R}^d$ of historical observations X, i.e., z = ViT(X). The details of ViT are provided in Section 4.7. After getting the latent representation z, we intend to use a Gaussian mixture model (GMM) as the decoder. First, we project the latent variable z into k different latent variables $\tilde{z}_{i=1:k}$ by using a feed-forward neural network Π:

$$\begin{aligned}
&[vec(\tilde{\mathbf{Z}})^T, vec(\mathbf{W}(z))^T]^T = \Pi(z),\\
&\tilde{\mathbf{Z}} = [\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_k] \in \mathbb{R}^{d\times k},\\
&\mathbf{W}(z) = [w_1(z), w_2(z), \ldots, w_k(z)] \in \mathbb{R}^{1\times k},
\end{aligned} \tag{4.1}$$

and at the same time, Π also outputs a confidence score $w_i(z)$ for each $\tilde{z}_i$. Then we use a GRU-based recurrent neural network to get the mean of predicted trajectories $\mu_i(z)$:

$$\begin{aligned}
&Y \sim Q = \sum_{i=1}^{k} w_i \mathcal{N}(\mu_i(z), \Sigma_j(z)),\\
&h_t^i = GRU([\mu(z)_{t-1}^i, \tilde{z}_i]),\\
&\mu(z)_t^i = MLP(h_t^i), \forall t = 1, 2, \ldots, T_f,
\end{aligned} \tag{4.2}$$

where h is the state variable of GRU cell. We set $\{\Sigma_j\}_{j=1,\ldots,k}$ equal to the identity matrix.

4.4.2 Uncertainty Estimator

In this chapter, we focus on the direct-error-prediction-based uncertainty estimator, i.e., learning an estimator to forecast the prediction error. Our proposed framework can be divided into representation module $z = enc(x)$ and uncertainty projection head module $u = h(z)$. The proposed framework does not require any intermediate information of predictors. Hence, the proposed uncertainty estimation framework is predictor-agnostic. Due to the disentanglement of the predictor and uncertainty estimator, our uncertainty estimator can be used as an analyzer by third-party institutions.

Encoder In order to analyze the importance of different regimes of features for uncertainty estimation in vehicle trajectory prediction, we propose using a feature extractor to choose different regimes of latent representations of features. The features from different regimes, i.e., static features X_s and X_a, are encoded by ResNet [60] with different parameters:

$$\begin{aligned}
&z_s = \text{ResNet}(X_s; \psi_s), z_a = \text{ResNet}(X_a; \psi_a),\\
&enc(X) = \text{S}(z_s, z_a),
\end{aligned} \tag{4.3}$$

where ψ_s ans ψ_a are the parameters of ResNet, respectively. S is defined as a function of z_s and z_a. It can be $\text{S}(z_s, z_a) = z_s$ or $\text{S}(z_s, z_a) = z_a$, which represents static-information-based uncertainty estimation and agent-information-based uncertainty estimation. It can also be a combination of different information such as $S(z_s, z_a) = z_s + z_a$. Since we intend to analyze the different influence of X_s and X_a, we use the selector $\text{S}(z_s, z_a) = z_s$ and $\text{S}(z_s, z_a) = z_a$.

Decoder After we get the latent embedding $enc(X)$, in order to get a stable and robust results, we use an ensemble of feed-forward neural networks as the uncertainty projection module:

$$\hat{e}^i(X) = h(z, \theta_i), i = 1, \ldots, M; u(X) = \frac{1}{M}\sum_{i=1}^{M} \hat{e}^i(X), \tag{4.4}$$

where θ_i represents i-th MLP's parameter. M is the number of ensembles. After we get the uncertainty estimator, we use the mean of the ensembles as the final uncertainty score $u(X)$.

4.4.3 Training Process

We use a two-stage training strategy. First, we train the predictor introduced in Section 4.4.1. After the optimized predictor is obtained, we evaluate the predictor on the training data and get the produced uncertainty dataset $\mathcal{D}_{unc} = \{X_i, e_i\}_{i=1:n}$. Then we train our proposed uncertainty estimator on $\mathcal{D}_{unc}$. The prediction loss and uncertainty estimation loss are introduced in the following paragraphs.

Prediction loss function We adopt the corrected negative loglikelihood (cNLL) which is used in [111] as the prediction loss function:

$$\begin{aligned}
&\mathcal{L}_{cNLL}(x, y; Q) = \\
&-\log \sum_{i=1}^{k} w_i(x) \prod_{t=1}^{T} \mathcal{N}(y_t | \mu_t^i(x), \Sigma_t(x)) - \sum_{t=1}^{T} \frac{1}{2} \log |2\pi\Sigma_t(x)| \\
&= -\log \sum_{i=1}^{k} w_i(x) e^{-\frac{1}{2}\sum_{t=1}^{T}(y_t - \mu_t^i(x))^T \Sigma_t^{-1}(x)(y_t - \mu_t^i(x))},
\end{aligned} \tag{4.5}$$

where Q represents the learned GMM. As mentioned in [111], such prediction loss function can capture multi-modality compared with weighted average displacement error (wADE). The cNLL metric will be used in both the training and testing procedure. For the variance, Σ, we set it as an identical matrix, which is also suggested in the Shifts Dataset.

Uncertainty loss function Directly predicting the error as uncertainty estimation has been mentioned in [72, 128]. It is reasonable to use the corrected negative log-likelihood as the supervised signal for uncertainty estimation. Considering the optimal solution $\hat{e}^\star$ of the following optimization:

$$\begin{aligned}
&\min_{\hat{e}} \mathbb{E}_{X \sim P_X} \mathbb{E}_{Y|X \sim P_{Y|X}} [(cNLL(Y|X) - \hat{e}(X))^2], \\
&\hat{e}^\star(x) = \mathbb{E}_{Y|x \sim P_{Y|X}} [cNLL(Y|x)] \\
&\qquad = \mathcal{H}(P_{Y|x}) + D_{KL}[P_{Y|x} \| Q] + C,
\end{aligned} \tag{4.6}$$

where C is a constant. P represents the ground truth distribution. Q is the learned Gaussian mixture model. When the learned distribution Q is the same as the ground truth distribution, then the optimal total uncertainty estimator $\hat{e}^{\star}$ will be the entropy of P added a constant, i.e., $\mathcal{H}(P)+C$. In the other word, if there is no distributional shift and $Q = P$, then $\hat{e}^{*}(x)-C$ will represent the aleatoric uncertainty. The divergence between P and Q can represent the model uncertainty. Please see the details of such decomposition in [72]. In this chapter, we focus on total uncertainty estimation and we utilize a normalized version of MSE loss for cNLL:

$$\mathcal{L}_{unc}(e, \hat{e}) = (s \cdot \text{clip}(e, 0, e_{upp}) - \hat{e})^2, \tag{4.7}$$

where s is the scale of error and e_{upp} is the upper bound of clip value. We notice that the training will be more stable by using such a normalization procedure.

4.5 Experiments

In this section, we first introduce the dataset and metrics in Section 4.5.1. Then we show the performance of the proposed predictor in Section 4.5.2 to confirm that the trained predictor is good enough compared with the existing state-of-the-art baselines. After that, we show the empirical analysis of the proposed uncertainty estimator in Section 4.5.3. Finally, we show the transfer learning results based on the uncertainty estimator in Section 4.5.4. The detailed experiment settings are in Section 4.7.

4.5.1 Dataset and Metrics

We use the vehicle motion dataset in Shifts Dataset [111], which is a large-scale public dataset providing distributional shift partition. The dataset was collected by a fleet and is currently the largest vehicle motion prediction dataset. The dataset was collected across different scenarios in terms of space, time, and weather conditions. For instance, the dataset was collected across six different cities from different countries, three different seasons, and four weather conditions. The state of traffic participants and a high-definition map are provided. The sampling frequency is 50 Hz. The historical horizon and future horizon are both 5 seconds. The original dataset is partitioned into training, development (dev), and evaluation (eval) sets. The static information includes crosswalk occupancy, crosswalk availability, lane availability, lane occupancy, lane priority, lane speed limit, and road polygons. Each feature occupies one channel. The agent information includes the occupancy, velocity, acceleration, and yaw angle of vehicles, and the occupancy and velocity for pedestrians. Hence, the total number of static features is 8, and the total number of agent features is 9. We use the pre-rendered images with size $17 \times 128 \times 128$, which are provided by the dataset, as the features. We use the official canonical version of the dataset, which provides the training and evaluation splits. Please see the details in [111]. For the transfer learning experiments, we include the training dataset collected in Ann Arbor and Tel Aviv with the weather condition "no participation", which is a similar setting to the official canonical dataset for the training

Table 4.1: Prediction Evaluation Results, (mean/standard deviation/median)

Partition	cNLL	mADE	mFDE
Evaluation	19.42/190.50/1.94	0.56/0.85/0.28	1.07/1.85/0.39
ID	17.75/182.73/1.94	0.54/0.81/0.28	1.02/1.74/0.39
OOD	26.05/218.68/1.98	0.65/0.99/0.29	1.27/2.21/0.41

dataset collected in Moscow. The intentions of target vehicles are provided as trajectory tags in the dataset. The evaluation dataset is same as the one in the official canonical dataset.

For the prediction metric, we use the three metrics in [111], i.e., cNLL, minimum average displacement error (mADE), and minimum final displacement error (mFDE).

4.5.2 Predictor Performance

Since this chapter focuses on the uncertainty analysis and how to use the uncertainty estimator for transfer learning, we assume that a comparable predictor is sufficient to be used in our experiments. From Table 4.1, we show that the proposed predictor can achieve similar performance to the best one [128] (15.68 for cNLL, 0.53 for mADE, and 1.02 for mFDE) in the challenge [111]. It also shows that the mean, standard deviation, and median of the prediction errors in OOD (out-of-distriubtion) cases are larger than the corresponding statistics in ID (in-distribution) cases.

4.5.3 Importance of Features for Uncertainty Estimation

In this section, we show the quantitive results and analysis by using the proposed uncertainty estimator. The evaluation dataset is divided into different groups: Group A and Group B. The groups are separated by the intention of the target vehicle, including moving forward, moving right, moving left, moving back, stationary, starting, and stopping. Group A represents the set of intentions, including moving forward, moving right, moving left, moving backward, and stationary. Group B consists of the intention of starting and stopping. The elements in each group are exclusive, e.g., if a case is labeled by “Left”, then it cannot be labeled by “Right”. However, A case can be labeled as both “Left” and “Starting”. First, we analyze the overall performance of the whole evaluation dataset. Then we provide the evaluation conditioned on different intentions. We also show the evaluation results on the data partition within different speed ranges. We compare four methods: i) agent-information-only uncertainty estimator (agent); ii) static-information-only uncertainty estimator (static); iii) full-information-based uncertianty estimator (full), which use both agent and static information for uncertainty estimation; iv) predictor-dependent uncertainty estimator (pred-latent), the output of the predictor’s encoder is used as input of the uncertainty head.

Overall Performance

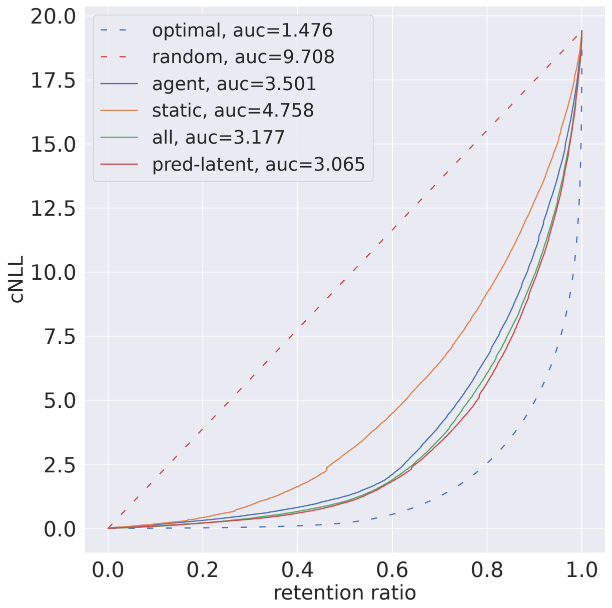
Error-Retention Curve Results The ERC of overall performance is shown in Figure 4.3. We notice that both agent-information-based and static-information-based approaches are much better than the random uncertainty estimator baseline and close to the optimal lower bound. The agent-information-based approach is better than the static-information-based one in general. It is reasonable to see that agent-only or static-only uncertainty estimator performance is worse than full-info uncertainty estimator. We also illustrate that the predictor-agnostic uncertainty estimator can achieve almost same performance as the predictor-dependent one. It shows that our proposed framework is effective. The observation is also consistent with the OOD partition of the evaluation dataset and the whole evaluation dataset. However, we cannot conclude that the comparison results are similar with different intentions of the target vehicle's driver. Hence, in the next section, we provide the results conditioned on different intentions.

Evaluation Conditioned on Human Driver Intention

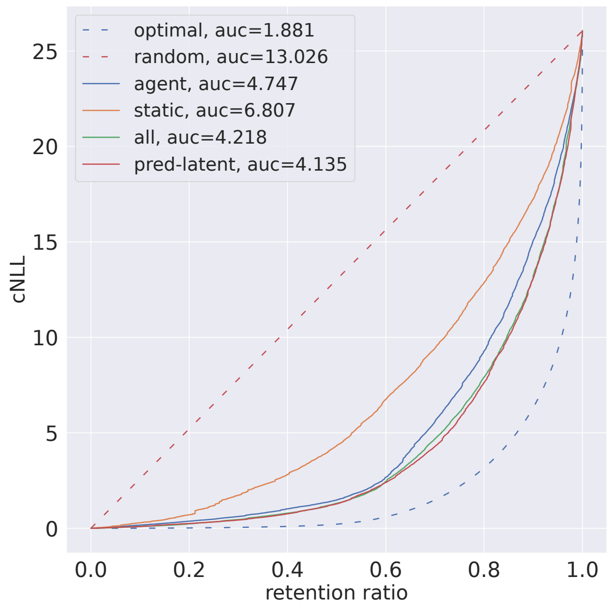
Stationary Performance The ERC performance evaluated on "stationary" is illustrated in Figure 4.3. We can observe that the agent-information-based approach is much better than the static-information-based approach. There are two potential explainations: i) The agent information is more important than static information in this scenario. For instance, when the traffic is heavy, and thus the target vehicle is surrounded by many vehicles, only observing the agent occupancy information is enough to determine the future trajectories. ii) If the vehicle's current velocity and acceleration are zero, the average speed of future trajectories will be low. In such cases, the scale of the total uncertainty of future trajectories can be small. In the aforementioned cases, agent information is enough to get an accurate uncertainty estimation, and the information gain from adding static information is not very significant. There are illustrations of prediction results conditioned on "stationary" in Figure 4.5a and Figure 4.5b.

Turning Performance We illustrate the ERC performance conditioned on "Turing left" and "Turning right" in Figure 4.4. We notice that static-information-based uncertainty estimation has a similar performance to the agent-information-based one. From the illustration, we would like to give some intuitive explanations of the empirical results:

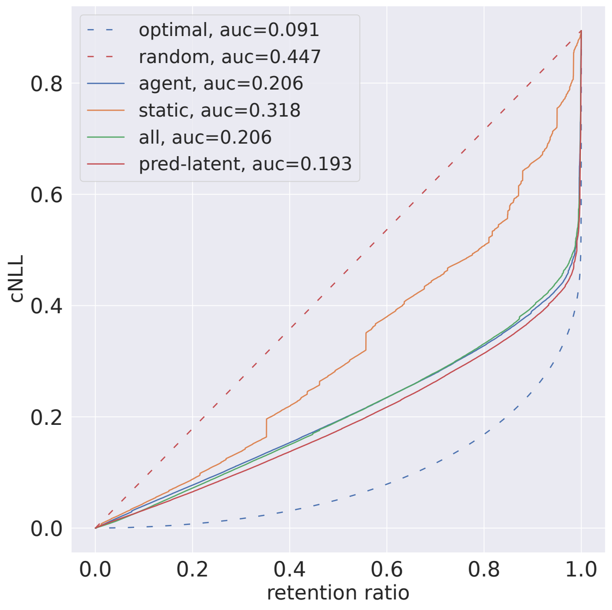
Why does the static-information-only uncertainty estimator work? In order to predict the future trajectory of a target vehicle that has the intention of "Turning left" and "Turning right" accurately, the information of the lane is significant since the future trajectories of vehicles turning left or right highly depend on the available routes. Also, it is difficult to predict the future trajectories in the turning left and turning right scenarios due to the multi-modality. The uncertainty estimator can identify such difficulty of scenarios by providing static information. Hence, the static-information-only uncertainty estimator



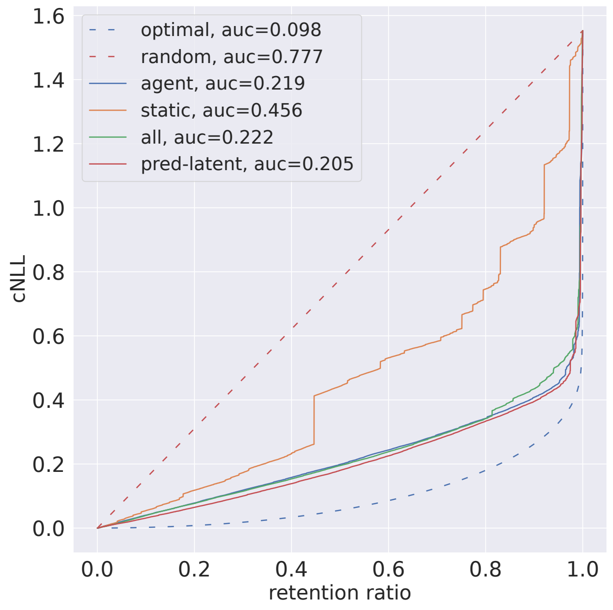
(a) Overall performance.



(b) OOD part performance.

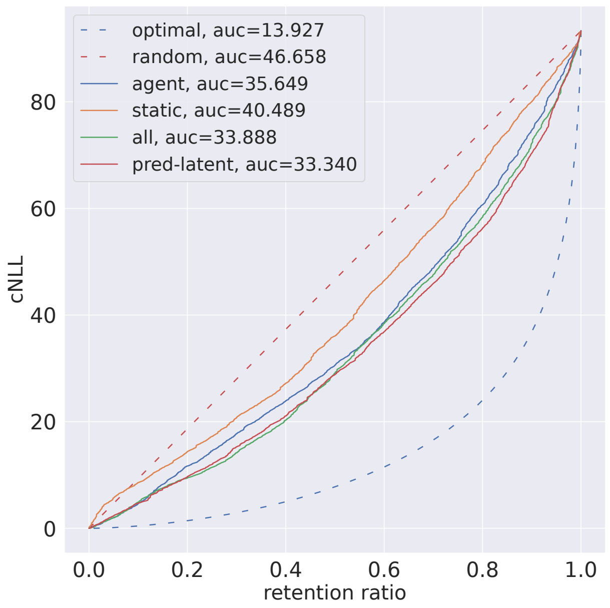


(c) Stationary.

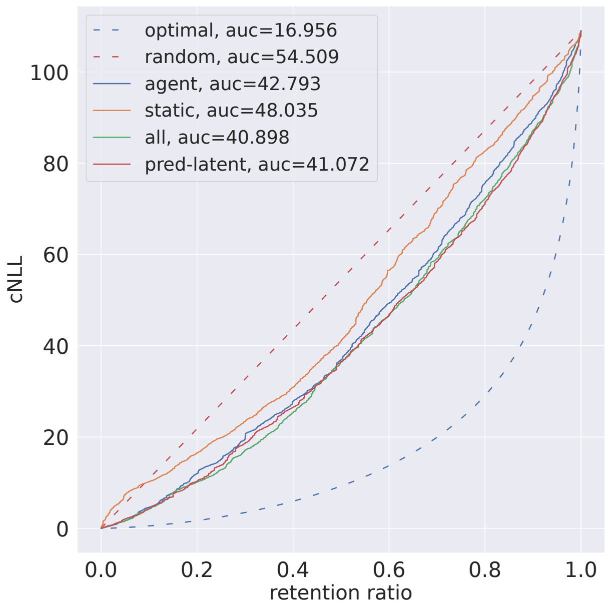


(d) Stationary, OOD part.

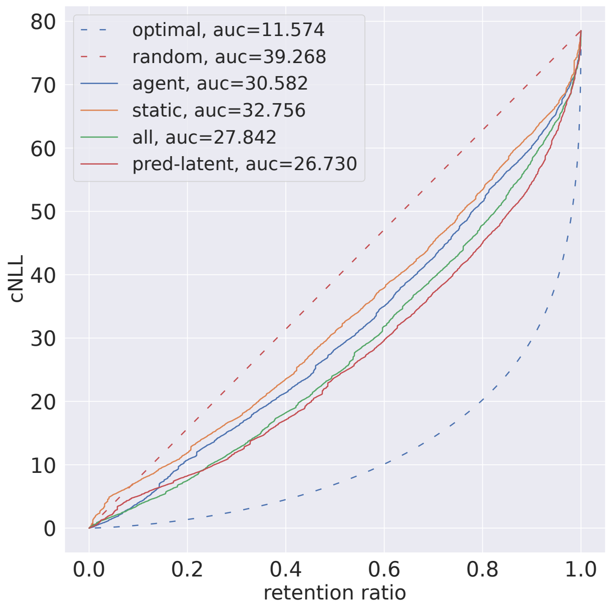
Figure 4.3: ERC results on the whole evaluation dataset and the part with intention "Stationary".



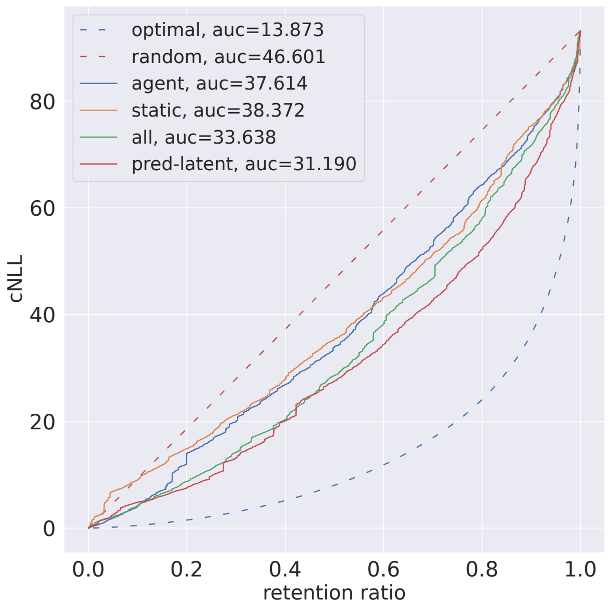
(a) Turing left.



(b) Turing left, OOD part.

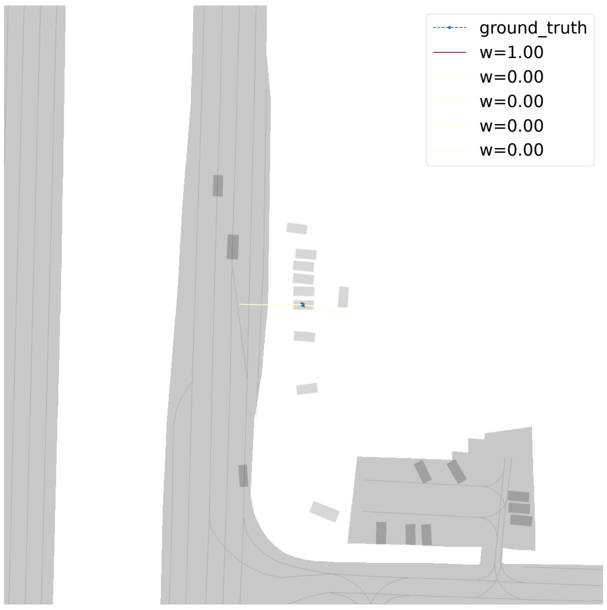


(c) Turning right.

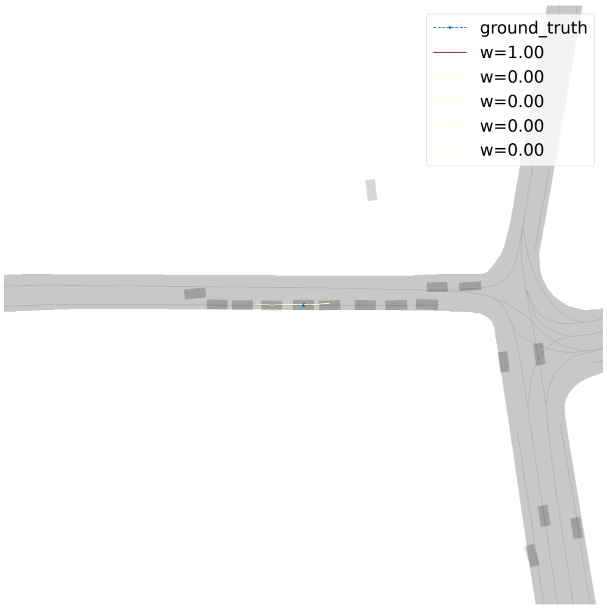


(d) Turing right, OOD part.

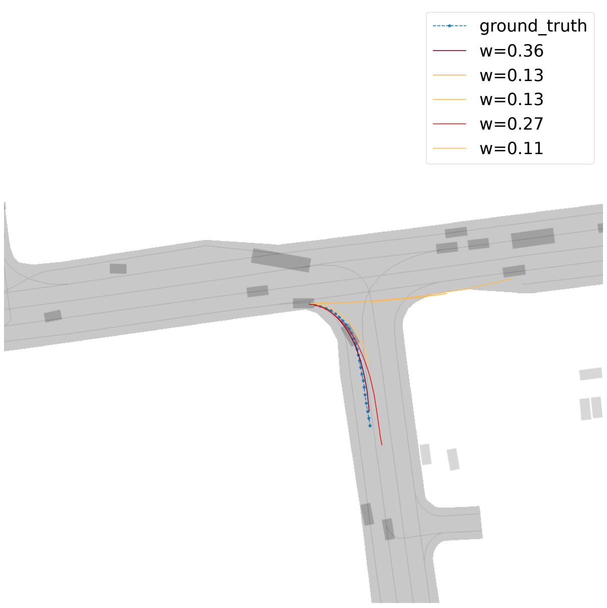
Figure 4.4: ERC results on the parts of evaluation dataset with intention "Turning left and right".



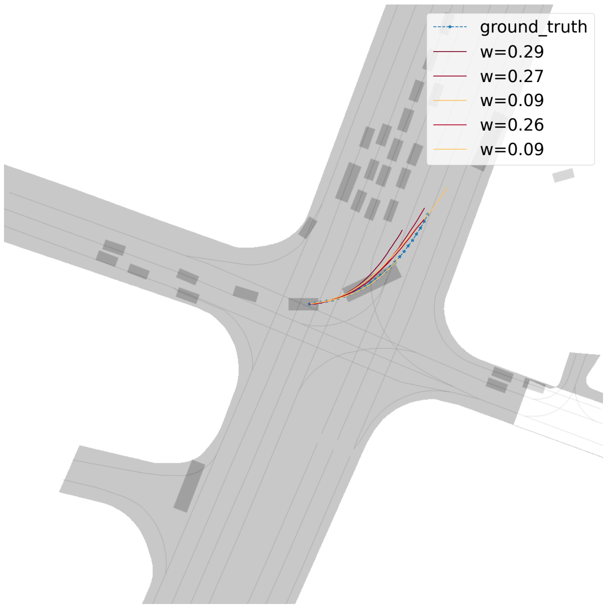
(a) Stationary, parking.



(b) Stationary, jammed.



(c) Turning right.



(d) Turing left.

Figure 4.5: Visualization of prediction results. Curves with different colors represent the predicted possible future trajectories with different confidence score w. The ground truth trajectory is shown as a blue dashed line with star markers.

indeed works in both scenarios. We illustrate a prediction result in Figure 4.5c, where it is impossible to predict the future trajectories without the lane information.

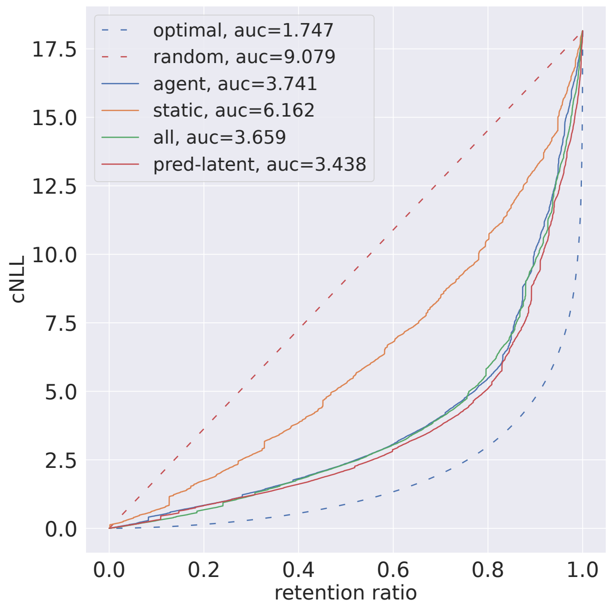
Why does the agent-information-only uncertainty estimator work, too? As we mentioned above, the static information may provide useful insight into where the target vehicle will go ahead. However, we find that the agent-information-based uncertainty estimator can also achieve comparable or better performance. We suggest that the correlation between the static information and the agent information will also help. For instance, when there are many agents in one case, the position of agents will imply the lane information. Besides, if the traffic is jammed, the agent's behavior will heavily depend on the other agents, too. We illustrate an example in Figure 4.5d. From Figure 4.5d, we can see that the position of agents can basically reflect the lane information.

Starting and Stopping Performance

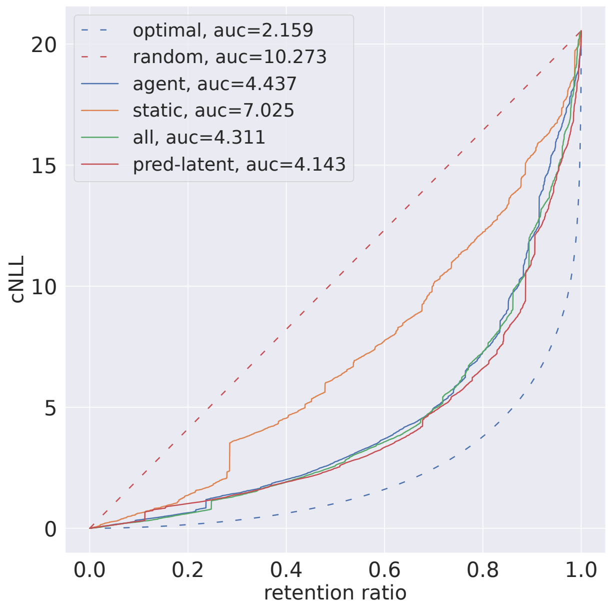
We illustrate the uncertainty estimators' performance on the datasets with the intention “starting” and “stopping” in Figure 4.6. We find that static-information-based and agent-information-based estimators can achieve similar performance on the dataset with the intention of “starting”, while the agent-information-based estimator is much better than the static-information-based one on the dataset with intention of “stopping”. One of reasonable explainations is that, some of the scenarios where the target vehicles were stopping are highly depended on the other agents' behavior. For instance, the deceleration of the target vehicle depends on the speed profile of the leading vehicle in Figure 4.7a and Figure 4.7b. Without the agent information, we cannot anticipate the future stopping of the target vehicle. The performance of the prediction and uncertainty estimation will be largely improved by giving the other vehicles information in such scenarios. For the scenarios of “starting”, the influence of the other agents will be smaller. For instance, in Figure 4.7c and Figure 4.7d, we note that the target vehicles just start moving, and the front vehicles may also just start moving. The distribution of future trajectories may be similiar across different scenarios and thus the difficulty and uncertainty of prediction is reduced. Please note that it is different from the moving forward, where the speed at the first time step or the last time step is not necessary to be zero.

Forward and Backward Performance

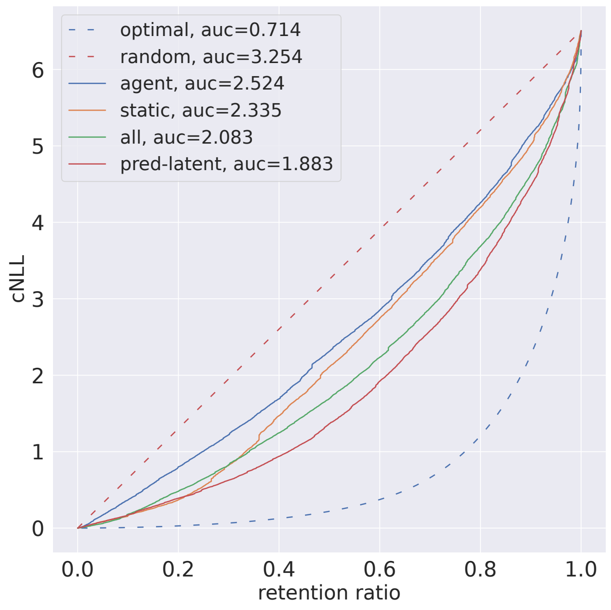
The evaluation results of the dataset with intention “forward” and “backward” are shown in Figure 4.8. We observe that the agent-information-based estimator is better than the static-information-based one in both forward and backward scenarios. It is reasonable since if the target vehicle's intention is just moving along one lane, the most important factor that the human driver should consider is the other traffic participants' behaviors. We observe that there is a mismatch between the performance of ERC on the evaluation dataset and the OOD part of the evaluation dataset in Figure 4.8c and 4.8d. The main reason is that the number of cases with “backward” intention is significantly smaller than the number of cases



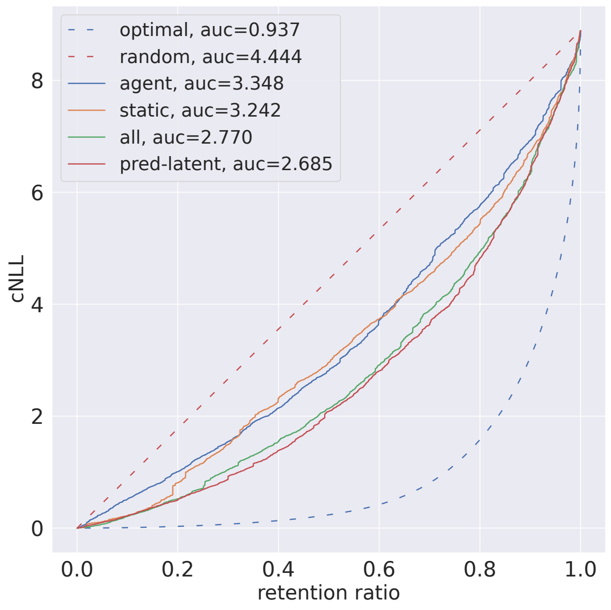
(a) Stopping.



(b) Stopping, OOD part.

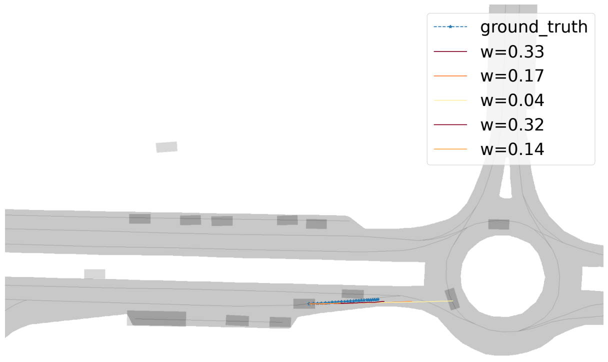


(c) Starting.

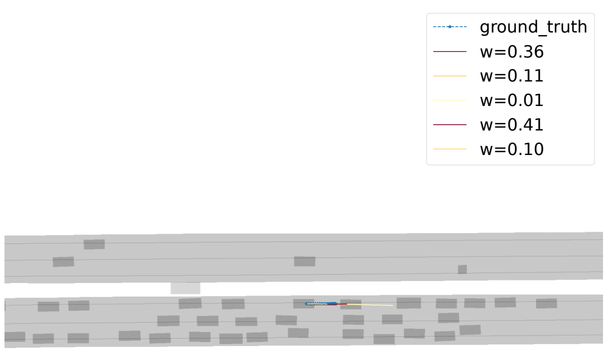


(d) Starting, OOD part.

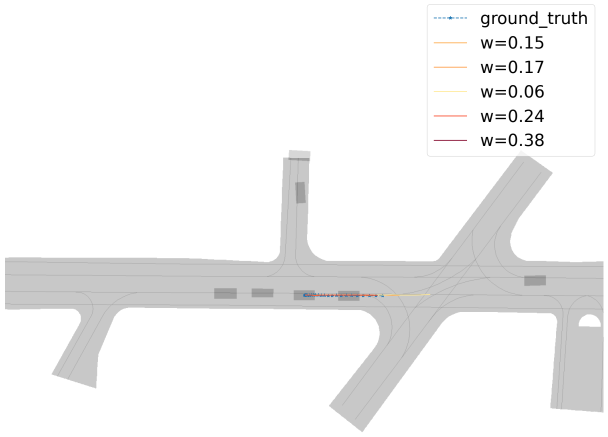
Figure 4.6: ERC results evaluated on testing dataset with intention “starting” and “stopping”.



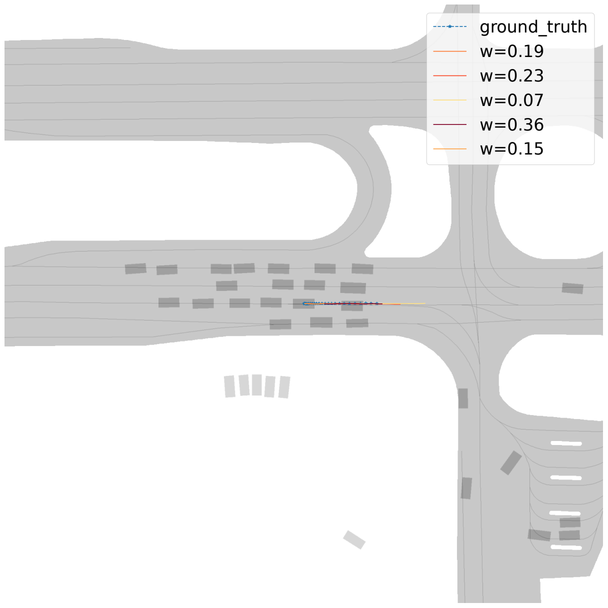
(a) Stopping, roundabout



(b) Stopping, jammed.

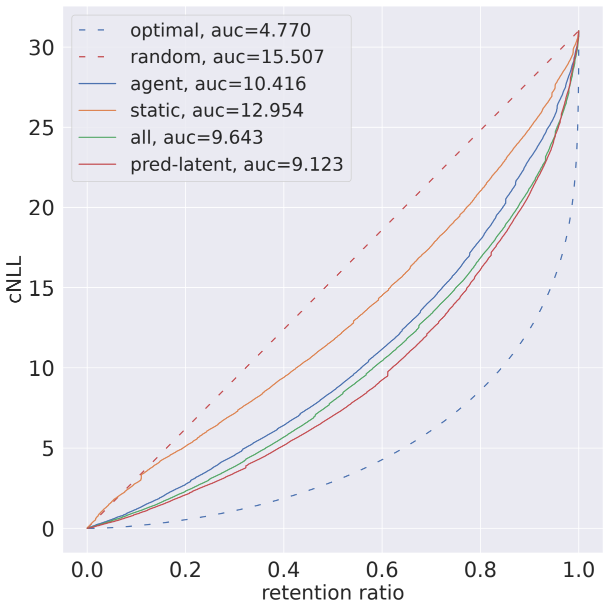


(c) Starting, Case 1.

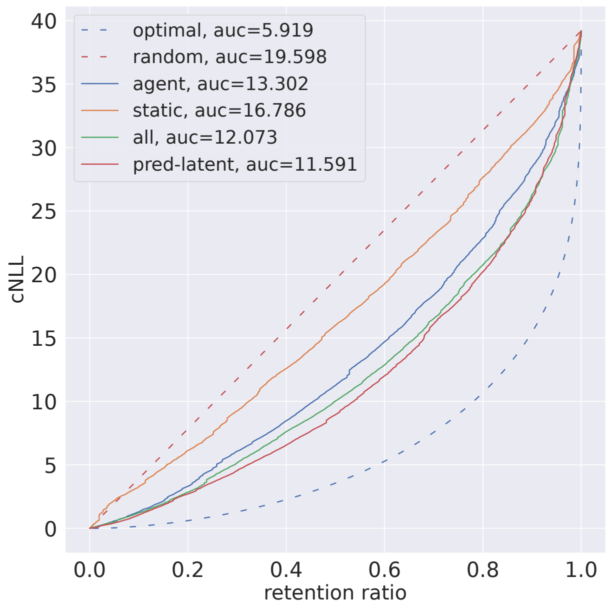


(d) Starting, Case 2.

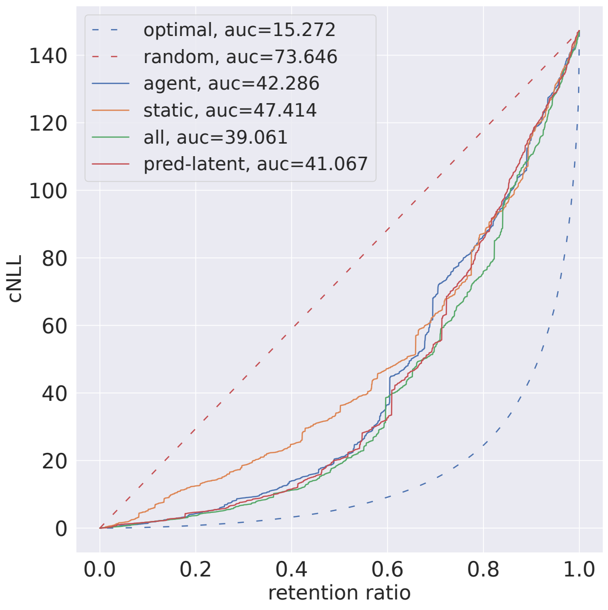
Figure 4.7: Visualization of prediction results with intention "starting" and "stopping". Curves with different colors represent the predicted possible future trajectories with different confidence score w. The ground truth trajectory is shown as a blue dashed line with star markers.



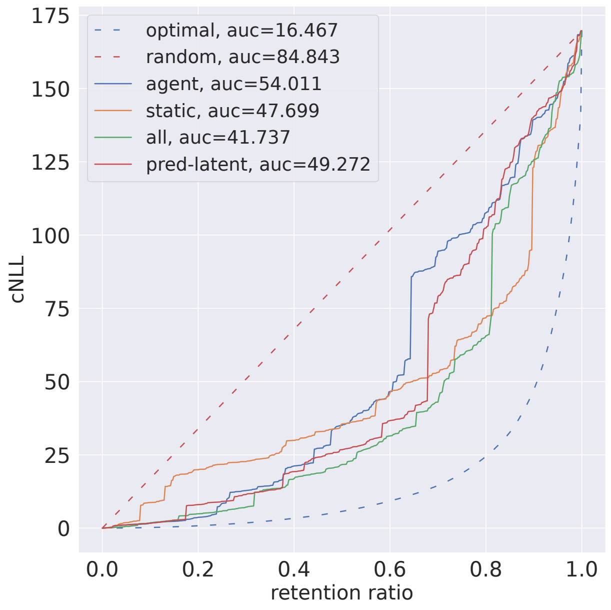
(a) Forward



(b) Forward, OOD part.

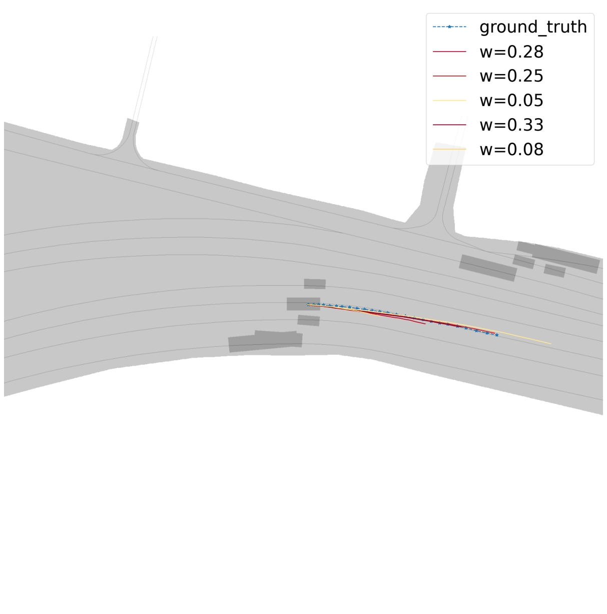


(c) Backward.

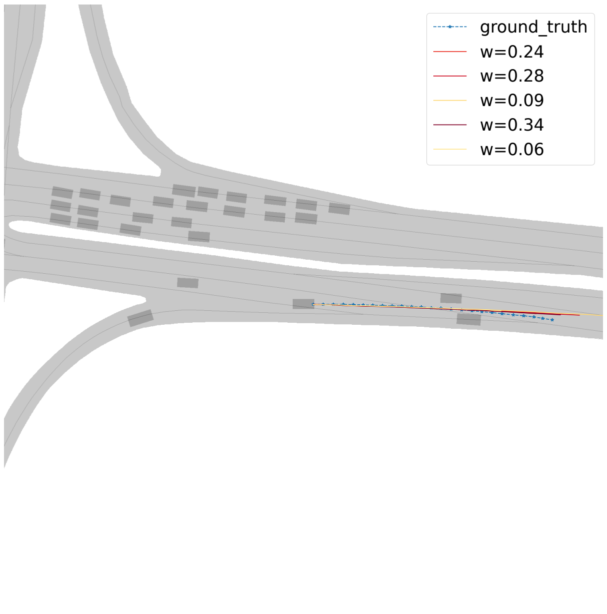


(d) Backward, OOD part.

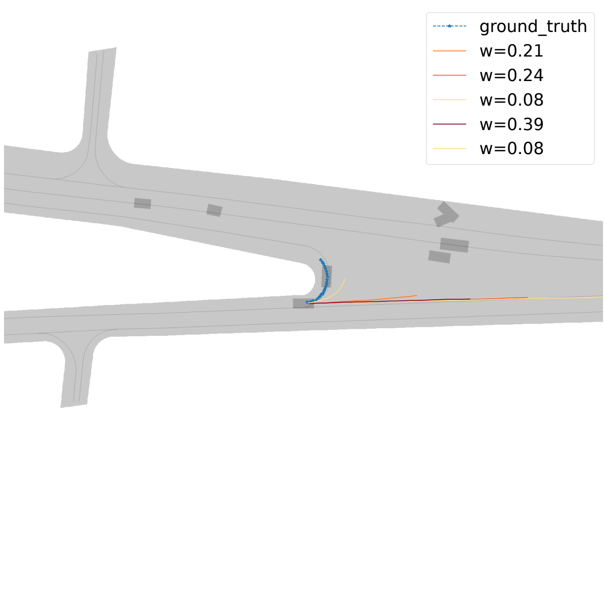
Figure 4.8: ERC results evaluated on testing dataset with intention “forward” and “backward”.



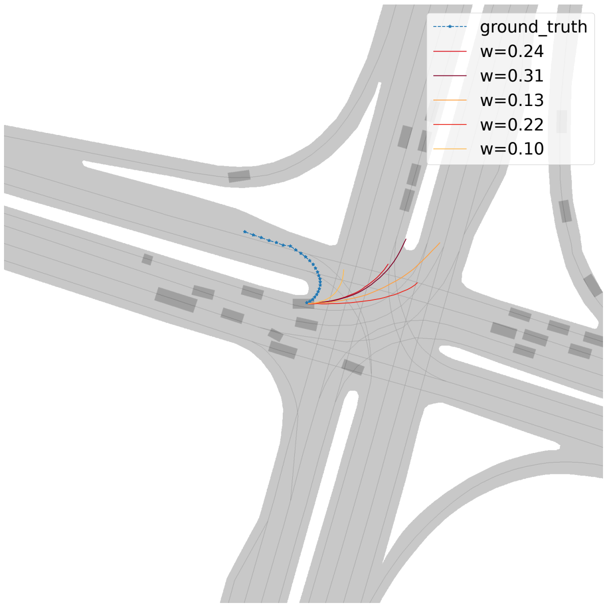
(a) Forward, Case 1.



(b) Forward, Case 2.

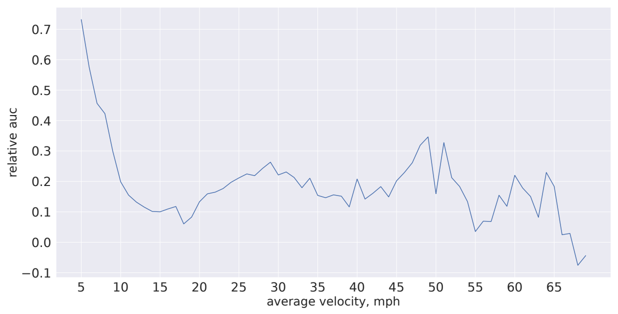


(c) Backward, Case 1.

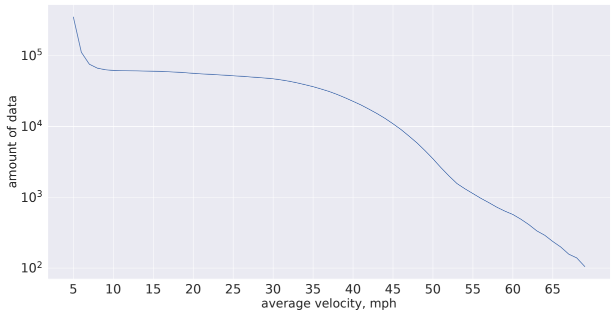


(d) Backward, Case 2.

Figure 4.9: Visualization of prediction results with intention "moving forward" and "moving backward". Curves with different colors represent the predicted possible future trajectories with different confidence score w. The ground truth trajectory is shown as a blue dashed line with star markers.



(a) $v - \delta$AUC performance.



(b) The number of data w.r.t v.

Figure 4.10: The relative performance between agent-information-based and static-information-based uncertainty estimator conditioned on different speed ranges.

with the other intentions. Hence the results shown in Figure 4.8c and Figure 4.8d may have less statistical significance. We illustrate the prediction results in Figure 4.9. In Figure 4.9c and Figure 4.9d, the predictor cannot learn the correct future trajectories as we suggest.

Performance in Different Average Speed Ranges

We also compare the performance of static-information-based and agent-information-based uncertainty estimation with the data associated with different average speed ranges. The data are filtered by the average speed range $[v - \Delta, v + \Delta]$, where the anchor v is the average speed of the future trajectory of the target vehicle. Δ is set as 5 mph. We use the relative improvement $\delta\text{AUC}(v)$ which is defined as:

$$\delta\text{AUC}(v) = \frac{\text{AUC}_{static}(v) - \text{AUC}_{agent}(v)}{\text{AUC}_{agent}(v) - \text{AUC}_{optimal}(v)}. \tag{4.8}$$

Such value reflects how much the agent-information-based uncertainty estimator is better than the static-information-based one. The value is normalized by the difference of AUC between agent-information one and the optimal solution. The results are visualized in Figure 4.10. From Figure 4.10a, we observe that when v is smaller than 10 mph, The agent-information-based uncertainty estimation is better than the static-information-based one. For instance, when the average speed of future trajectories belongs to $[0, 10]$, the agent-information-based one is about 70% better than the static-information-based one. However, the agent-information-based one seems not to improve so much when v is larger than 10 mph. Especially when v belongs to $[15, 20]$, the agent-information-based uncertainty estimator only improves less than 10%. We also illustrate the amount of data with respect to v in Figure 4.10b. The relative AUC has large fluctuation when the average velocity is larger than 40 mph. It may attribute to the number of data decreases rapidly when $v > 40$ mph as shown in Figure 4.10b.

Algorithm 1: The algorithm for adaptation to new environments

```
Input: $r$: The proportion of dataset. predictor $\mathcal{P}$. Static-infomation-based
       uncertainty estimator.
1 Collect the static information $\mathcal{D}_s = \{X_{s,i}\}_{i=1:n}$ in new environments.
2 Get the uncertainty score $u_i$ by the static-information uncertainty estimator for each
    $X_{s,i}$.
3 Get the sorted $\mathcal{E}_{sorted} = \{(X_{s,[i]}, u_{[i]})\}_{i=1:n}$, where $u_{[i]} \geq u_{[i+1]}, \forall i = 1, \ldots, n-1$.
4 Select the top-$k$ data points in $\mathcal{E}_{sorted}$, where $k = \lfloor rn \rfloor$.
5 According the selected $X_s$, collect data $\mathcal{D}_{adapt} = \{(X_i, y_i)\}_{i=1:k}$.
6 Finetune the decoder of $\mathcal{P}$ with $\mathcal{D}_{adapt}$.
```

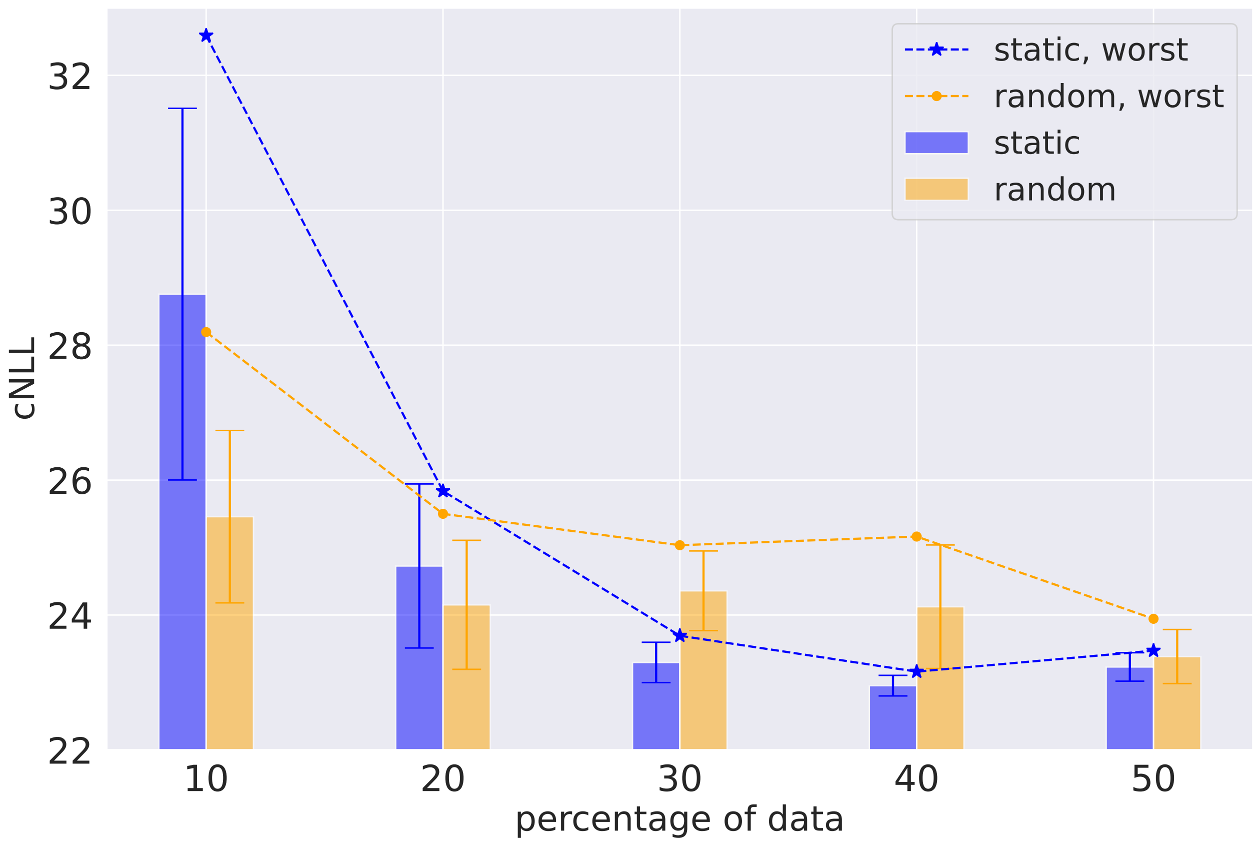
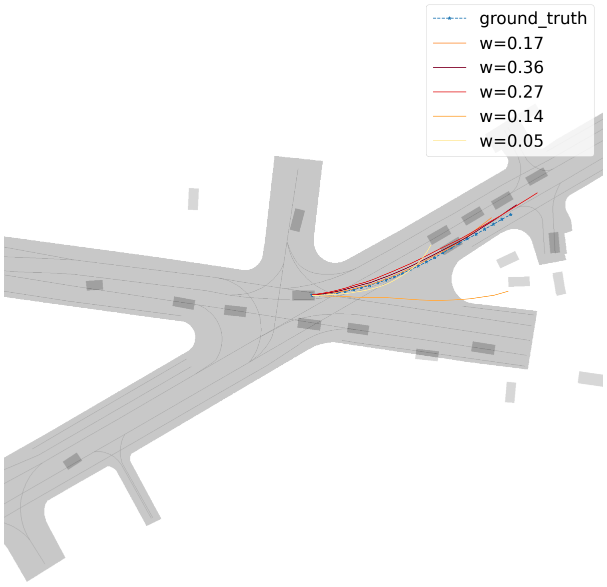
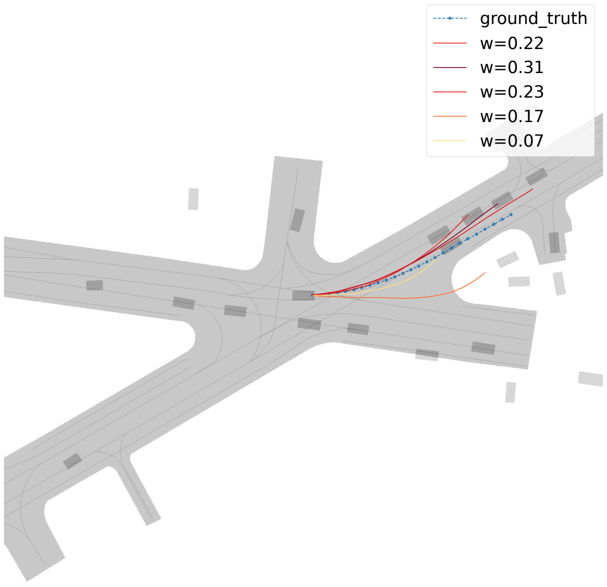


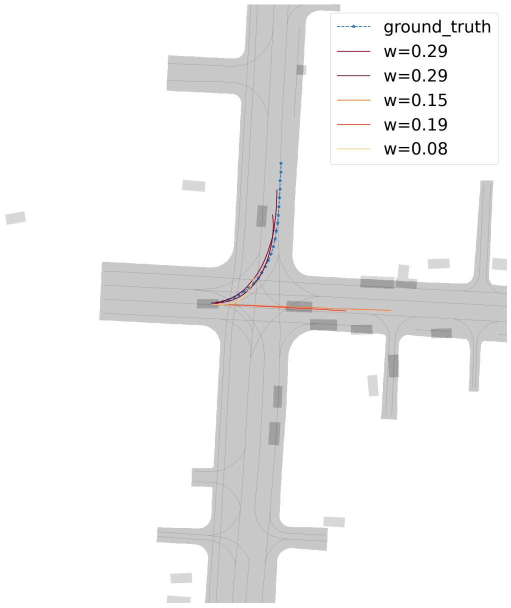
Figure 4.11: Adaptation to new environments.



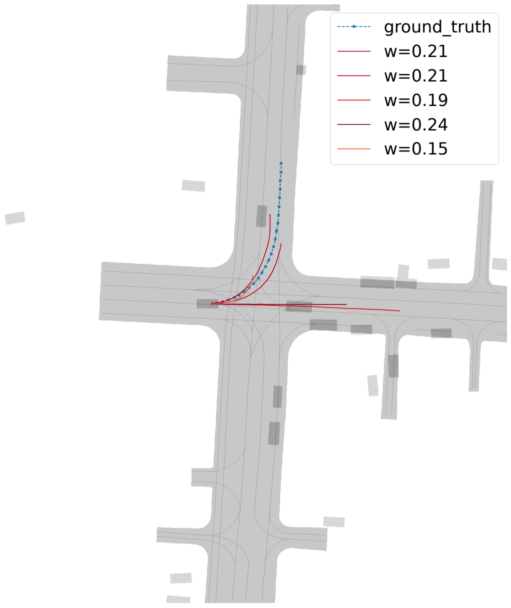
(a) Turning left, Case 1, fine-tuned model.



(b) Turning left, Case 1, original model.



(c) Turning left, Case 2, fine-tuned model.



(d) Turning left, Case 2, original model.

Figure 4.12: Visualization of prediction results with intention "turning left" for transfer learning experiments. Curves with different colors represent the predicted possible future trajectories with different confidence score w. The ground truth trajectory is shown as a blue dashed line with star markers.

4.5.4 Adaptation to New Environments

From the analysis in Section 4.5.3, we notice that the static-information-only uncertainty estimator can also have a good performance. Since the static information can be obtained directly without the deployment of data collection fleets, it is efficient and budget-friendly for transfer learning, continual learning, or active learning. In this section, we investigate whether using the static-information-based uncertainty estimator indeed helps improve the transfer learning efficiency. In this setting, we try to finetune the predictor trained with the training dataset (collected in Moscow) in order to adapt to the new environments Ann Arbor and Tel Aviv, which are used in the evaluation dataset.

We use a simple algorithm listed in Algorithm 1. We assume that we first collect the static information X_s at some locations. Then we select some important locations according to the static-information-based uncertainty estimator. Then we collect the agent information X_a. We finetune the decoder of the predictor based on the selected data via th early stopping technique. We compare our proposed method (static-UE) with the predictor finetuned with randomly selected data (random-UE). The result is shown in Figure 4.11. In Figure 4.11, the heights of bars represent the mean of cNLL. The errorbars represent the standard deviation. Meanwhile, the dashed lines indicate the worst performances of both methods. We notice that the performance of our proposed approach is better than the method with randomly selected data in terms of mean, variance and worst-performance. When only using 10% of data, static-UE achieves worse performance. One of the explainations is that the 10% of the data only represents the most worst cases according to the uncertainty esitmator, and only using the static-information-selected data is not good enough for predictor to attain new knowledge of the majority of new scenarios. We observe that static-UE can reduce the prediction error gradually compared with random-UE. When the number of selected data is larger than 20%, static-UE can achieve smaller error in terms of both mean and variance. However, random-UE does not improve so much from 20% to 40%, and the variance of random-UE keeps large. One of the reason why the variance of static-UE is smaller is that we choose the training data according to the uncertainty score, there is almost no randomness from the data. When the number of data increases to 50%, we find that both approaches achieve similar performance. It means that 50% data may be enough to represent the whole data distribution.

Prediction Visualization for Transfer Learning Experiments We provide several typical prediction visualizations generated by our proposed approach with 30% training data selected by the static-information-based uncertainty estimator. In Figure 4.12a, Figure 4.12b and Figure 4.12c, 4.12d, we illustrate the cases where future trajectories of vehicles have the turning left intentions. We notice that the fine-tuned model is indeed better than the original model. In Case 1, we can see that the fine-tuned model adjust each mode of possible future trajectories to align the lanes compared with the ones provided by the original model, i.e., the light orange ones, which are the predicted trajectories with small confidence scores. In Case 2, we can observe that the predicted trajectories with high confidence scores provided by the fine-tuned model are more accurate than the ones provided by the original model. In Figure 4.13a, 4.13b and Figure 4.13c, 4.13d, we illustrate the cases when future trajectories of vehicles have the turning right intentions. In Case 1, we notice that the fine-tuned model can adapt to the new scenarios and find the correct future lanes, while the original model cannot capture the new static information correctly. In our experiment, we freeze the encoder in order to maintain the knowledge learned before, however this strategy reduces the transferability at the same time. In the future, more advanced techniques can be applied.

4.6 Discussion

In this section, we discuss the limitation of our proposed framework. The major limitation of our work is that for transferring to the new environment, we assume that we can directly get the static information. However, the current input feature X_s is centered at the target vehicle position due to the dataset format. It is easily solved in future work by randomly sampling possible current locations of target vehicles instead of assuming we know the ground truth locations. Also, we only use a simple sampling strategy to show that static-information-only uncertainty estimator can improve the transfer learning strategy. Although Algorithm 1 is enough to support our argument, more advanced sampling techniques are desired to improve the performance further. In the future, we will also include the analysis for the other state-of-the-art predictors, such as [54, 22], etc. It will also be interesting to expand our analysis on the different large-scale datasets such as Argoverse 2 [173] and Waymo Dataset [153].

4.7 Implementation Details

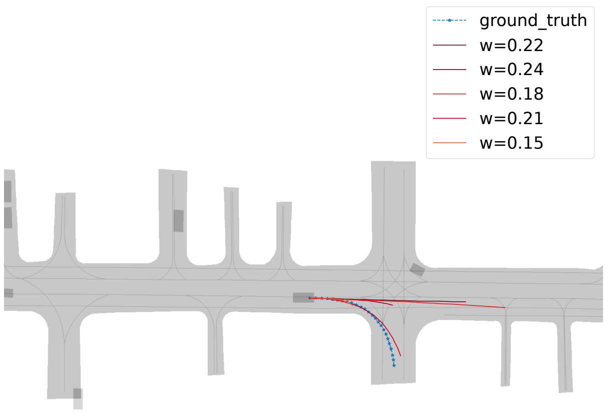
4.7.1 Error-Retention Curve

In order to make the paper consistently and clearly, we re-elaborate and formulate the definition of the error-retention curve as follows:

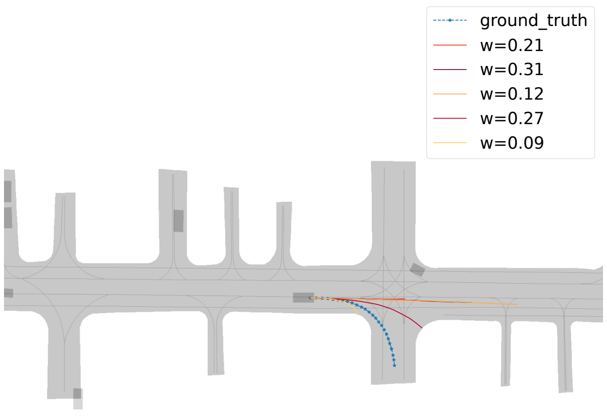
Definition 1 (Error-retention curve [111])**.** *Given a predictor $\mathcal{P}$, uncertainty estimator $\mathcal{U}$ and dataset $\mathcal{D}_{test}$ with size n, the error-retention curve (ERC) is defined as a map s : $\{\frac{i}{n}\}_{i=0:n} \rightarrow [0, \infty]$, which is obtained by the following procedures:*

1. Given a testing dataset $\mathcal{D}_{test} = \{(X_i, Y_i)\}_{i=1:n}$, calculate the error-uncertainty set $\mathcal{E}_{test} = \{(u_i, e_i)\}_{i=1:n}$ from $\mathcal{P}$ and $\mathcal{U}$.

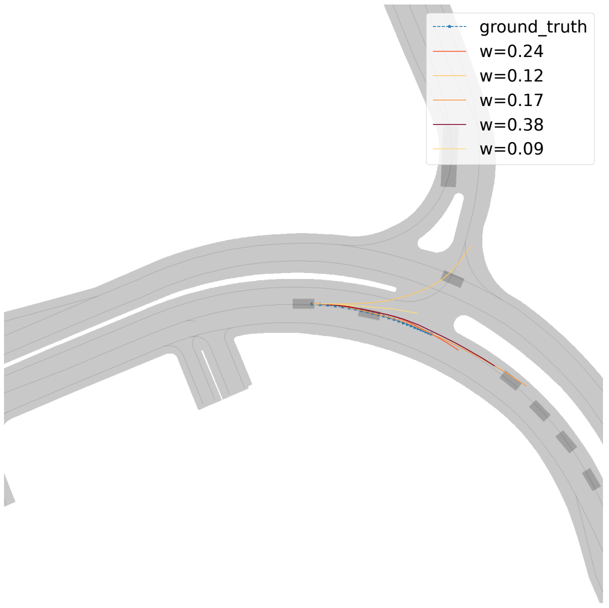
2. Sort $\mathcal{E}_{test}$ according to u, we have $\tilde{\mathcal{E}}_{test} = \{(u_{[i]}, e_{[i]})\}_{i=1:n}$, where $u_{[i]} \leq u_{[i+1]}, \forall i = 1, \ldots, n-1$.



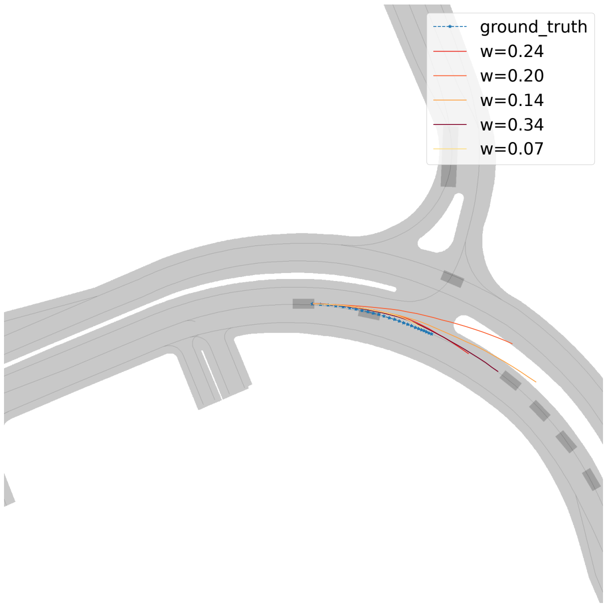
(a) Turning right, Case 1, fine-tuned model.



(b) Turning right, Case 1, original model.



(c) Turning right, Case 2, fine-tuned model.



(d) Turning right, Case 2, original model.

Figure 4.13: Visualization of prediction results with intention “turning right” for transfer learning experiments. Curves with different colors represent the predicted possible future trajectories with different confidence score w. The ground truth trajectory is shown as a blue dashed line with star markers.

3. s and $\text{AUC}(s) = \int s(r)d\mu(r)$ is calculated as

$$
\begin{aligned}
s(\frac{i}{n}) &= \frac{1}{n}\sum_{j=1}^{i} e_{[j]}, \forall i = 1, \dots, n, s(0) = 0, \\
\text{AUC}(s) &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{i} e_{[j]} = \sum_{i=1}^{n} \frac{(n-i+1)}{n} e_{[i]}.
\end{aligned}
\tag{4.9}
$$

As mentioned in [111], the minimum of AUC is obtained whenever u and e has a monotonic non-decreasing relation. We denote the minimal curve as "optimal" in the figures. The expected value of AUC will be $\frac{1}{2n}\sum_{i=1}^{n} e_i$, and s will be a linear function if the permutation of e is sampled uniformly. The curve is denoted as "random" in the figures, which is served as a "worst-case" indication. Notice that it is possible that the ERC is above the "random" curve, which means that the corresponding uncertainty estimator is even worse than a random estimator.

4.7.2 Predictor Implementation

For the encoder (ViT) in the proposed predictor, the patch size is 16, and the depth is 12. We use 8 multi-attention heads. The size of the feed-forward neural network in ViT is 768. The pooling strategy is "mean". The output size of ViT is 128. For the decoder of the predictor, we use GRU [28] as the recurrent neural network cell. The hidden layer sizes of the projection neural network Π is [300, 300], and the activation is ReLU. The number of GMM's kernels k is 5. During the predictor training phase, we use AdamW [104] as the optimizer. The learning rate is 1e-4, and the decay rate is 1e-3. We train the predictor with 21 epochs and the batch size is 1024. We notice that after 21 epochs, the training loss is stable. In order to reduce the influence of sub-optimality of the predictor, we choose the best model of three trained predictors for the following uncertainty estimators.

4.7.3 Uncertainty Estimator Implementation

For the uncertainty estimators, we use ResNet34 [60] as the encoder. We modify the first convolutional layer in order to fit the input image size: $c \times 128 \times 128$, where $c = 8$ for the static-information uncertainty estimator, and $c = 9$ for the agent-information uncertainty estimator. The number of ensembles $M = 3$. For each uncertainty head, we use a feed-forward network with hidden sizes: [512, 256, 128]. The activation is ReLU. The latent variable's dimension for the uncertainty estimator is 128. For the uncertainty estimation loss, we use the scale $s = 1/300$, and the $e_{upp} = 10000$. For the uncertainty estimator training, we use AdamW as the optimizer and the learning rate is 1e-4, too. The decay rate is $1e-3$. We train both the static-information-based and agent-information-based uncertainty estimators for 15 epochs and the batch size is 1024. We observe that the standard deviation of the ensemble models are very small and not influence the conclusion in the analysis.

4.7.4 Transfer Learning Settings

For the transfer learning experiment, we freeze the encoder of the predictor, i.e., ViT, and fine-tune the decoder part with 40 epochs. The batch size is 1024. We use the same optimizer and configurations of the trained predictor. For the transfer learning experiments, we train 3 different models for each approach to get the statistics. For 10% data, we observe that the variance of random-UE is remarkably large and hence we use three more random experiments to calcuate the statistics.

4.8 Chapter Summary

In this chapter, we introduce a generic direct error-prediction-based uncertainty estimation framework for vehicle trajectory prediction. The proposed error-prediction-based uncertainty estimation framework does not require the detail of predictors design, enabling the third party, such as a regular department or insurance company, to evaluate autonomous driving companies' prediction products. Meanwhile, in order to investigate how the different feature attributes influence the uncertainty estimation, we specifically design a pipeline to analyze it in an ablative way. We validate our proposed framework and conduct a thorough analysis of the Shifts Dataset. In the analysis, we design several experiments to show the uncertainty performance in different types of data, e.g., the dataset separated by the target vehicle intention, the average future speed, etc. We conclude that only using static information to estimate uncertainty can also achieve a good result through the analysis. The conclusion is meaningful for several downstream tasks, such as transfer learning, active learning, and continual learning, since the static information can be obtained with much less budget than collecting dynamic agent information. We conduct an experiment with the transfer learning setting to support our argument that the static-information uncertainty estimator can also improve the training efficiency. In the future, We intend to use the proposed framework on the other state-of-the-art predictors and compare it with the other uncertainty estimation techniques.

Part II

From Human Behavior Prediction to Generation

Chapter 5

Diverse Human Motion Prediction via Knowledge Distillation

5.1 Introduction

Human motion prediction plays a significant role in several applications such as human-robot interaction [11, 12], autonomous driving [30, 31, 94, 107], and animation [122]. For instance, an autonomous driving system can make a safe planning strategy given an accurate motion prediction of pedestrians. Moreover, robots can cooperate reasonably with people when they have a good understanding of human beings' future plans. However, since diversity and uncertainty are human future motion's intrinsic properties, it becomes a challenging problem in the computer science community. Unlike the vehicle trajectory prediction scenarios where we can get prior knowledge such as the traffic rules and routing information [106, 105] to constrain the different modes of trajectories, we can hardly get any prior knowledge about what humans will do in the future. Thus, we can only leverage the information from the given dataset, which increases the difficulty of diverse human motion prediction.

There are two lines of research in this area. First, several works attempt to get an accurate human motion prediction without considering the diversity, such as [99] based on graph neural network and [166] based on recurrent neural network. On the other line, some research investigates how to increase the diversity of human motion prediction based on deep generative models [5, 177, 180, 132] or diverse sampling techniques [178]. Deep generative models such as variational autoencoder and generative adversarial network naturally capture the stochastic behaviors, while they may suffer from the posterior collapse or mode collapse problems. Otherwise, even if we assume that the generative models can capture the actual data distribution, the data distribution can still be very imbalanced and skewed, which makes that sampling the minor modes is challenging within a limited number of samples. Several works [178, 179, 113] propose new losses to increase diversity while keeping the prediction natural and accurate. In [179], a multiple sampling function is designed to explicitly capture the different modes of the distribution based on a pre-trained conditional variational

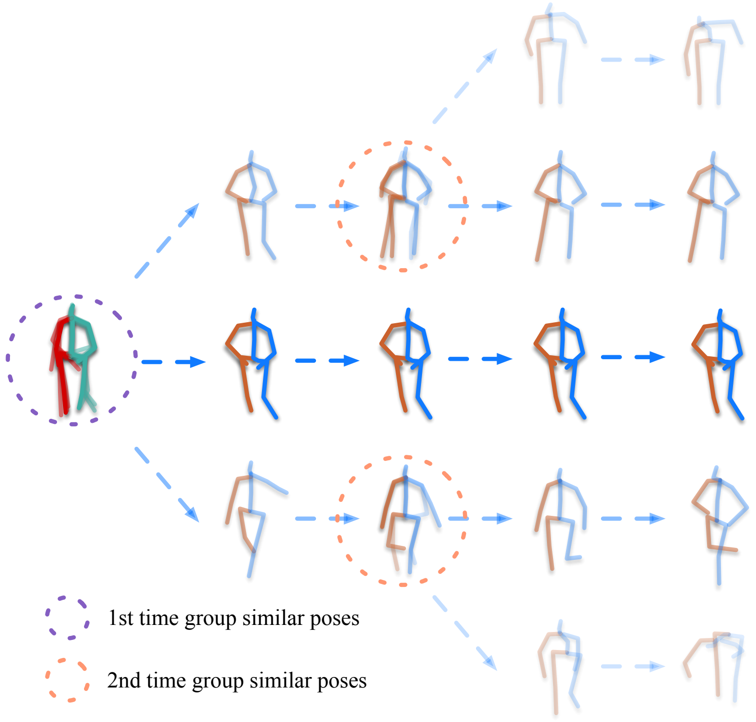


Figure 5.1: Illustration of obtaining multi-modal pseudo future motions in a dataset. We can cluster the similar initial poses (purple dashed circle) and share their future poses as the common ground truth. The solid poses are the ground truth and the transparent ones are the augmented poses. We argue that such an approach can be applied recursively (orange dashed circle), which will lead to discovering more different and realistic modes of motion in the data.

autoencoder. By using this pre-trained variational autoencoder, such methods can control the likelihood of predicted motion with a training hyperparameter. In [15, 113, 87], they proposed generative models to learn the distribution implicitly. However, these works still have to choose hyperparameters before training to balance the likelihood and diversity sampling. It implies that such approaches cannot be adjusted and controlled during the testing phase. Considering the real-world application such as pedestrian motion prediction in autonomous driving, we not only need to know most of the different possible modes of motion but also need to know which modes will most likely happen. It will be more practical if we can decide the balance of accuracy sampling and diversity sampling during the testing phase for the purpose of designing the risk-averse or risk-seeking planner of autonomous vehicles. Hence, we introduce a multi-objective variational inference framework with two different priors. The proposed structure makes it possible to adjust the ratio between accuracy and diversity sampling during the testing time.

Meanwhile, since there is only one ground-truth future motion poses given a historical observation, several works [142, 176] propose to use a similarity cluster-based technique to get the multi-modal pseudo-ground-truth future motions. Similar initial poses are grouped, and their corresponding future poses can be viewed as the pseudo possible future motions for each initial pose in the group. We argue that such logic can also be applied recursively. We can group similar poses again at certain steps and get the shared futures. A demonstration is shown in Figure 5.1. This strategy can boost the diversity of future motions. However, the sampling number will exponentially increase due to the recursive queries during training and make such direct implementation intractable. In order to solve this issue, we introduce an oracle that provides several possible future motions with a short-term horizon to instruct the predictor repeatedly. To summarize, our contributions are three folds:

- We propose a unified multi-objective conditional variational autoencoder based human motion prediction framework, which can adjust the ratio of sample numbers of accuracy and diversity sampling during testing.
- We propose to learn a short-term oracle system and distill the oracle's knowledge into the prediction framework to increase the diversity of human future motions. In order to achieve this goal, we propose a novel sample-based loss to supervise the predictor during the training phase.
- We evaluate the performance of our proposed approach on two human motion datasets. The experiments results show that our methods can achieve state-of-the-art performance.

5.2 Related Works

5.2.1 Human Motion Prediction

Human motion prediction has been investigated with many different approaches in the computer vision community. At the early stage, several methods [20, 145, 156, 167, 114, 3, 101] without deep learning techniques are proposed such as Gaussian process [168], hidden Markov model [20], and latent variable models [156]. Such methods can achieve good performance for recurrent human motion data. However, they may not be suitable for more complicated irregular human motions. As several promising deep learning models such as recurrent neural network (RNN) [42, 27, 16] and graph neural network (GNN) [139, 21, 84, 96, 97] are proposed recently, there are several research focusing on how to incorporate the models above to enhance the deterministic human motion prediction accuracy. Several works such as [53, 71, 115, 125, 188] are based on RNN, and [99, 113] utilize graph neural network (GNN) to capture both the temporal and spatial information. In order to get more diverse human motion prediction, several probabilistic models [17, 81, 82, 5, 87, 15, 102, 180, 132] are applied to capture the uncertainty of human motion. Deep generative models can be used to estimate the data distribution. There are several approaches based on variational autoencoders [17, 81, 82, 5], generative adversarial networks[87, 15, 102, 36] and normalizing flows[180, 132, 50].

5.2.2 Diverse Forecasting

In [177], the authors propose an approach that can learn a representation for motion reconstruction and transformation together. Also, GAN-like models are utilized in [15, 87] to capture the diverse human motion prediction. There are also some research using a different representation to improve the diversity [188]. In [178], a diversity sampling function which is formulated as a determinantal point process [56, 57, 86] is proposed. Especially in [179], the authors argue that even though the existed likelihood-based methods can have a good estimation of the data, they can still be challenging to sample some minor modes given a fixed number of samples. Hence, they propose to learn another diversity sampling function that can generate diverse motions based on one pre-trained variational autoencoder model. However, the proposed model needs to choose hyperparameters to balance the likelihood and diversity before training. We investigate the diverse human motion prediction in an orthogonal direction with the related work. We aim to get a unified model that can adjust the sample number ratio between accuracy and diversity samples during the testing phase. Besides, we attempt to explore more diverse and natural modes by utilizing pseudo future motions with a short-term oracle, and any models mentioned above can be integrated.

5.3 Problem Formulation

Our goal is to predict the possible future human motions given a dataset $\mathcal{D}$. We denote the human motion with time horizon $T = T_h + T_f$ as $\boldsymbol{X}_{t-T_h+1:t+T_f} = [\boldsymbol{X}_{t-T_h+1}, \dots, \boldsymbol{X}_{t+T_f}]$, where $\boldsymbol{X}_t \in \mathbb{R}^d$ is the human joints Cartesian coordinates at time step t. T_h and T_f are the historical horizon and future horizon respectively. Given an observation $\boldsymbol{C} = \boldsymbol{X}_{t-T_h+1:t}$, we intend to get the future motion distribution $P(\boldsymbol{X}_{t+1:t+T_f}|\boldsymbol{C}, \rho)$. Since such conditional probabilistic distribution may have several dominant modes, it is difficult to sample the other modes given a fixed sampling number. In contrast, if we focus on increasing the diversity of the samples, the prediction accuracy will be undermined. In this chapter, we introduce a variable $\rho \in [0, 1]$ to control the degree of diversity of prediction, i.e., we intend to get M samples $X^i_{t+1:t+T_f} \sim P(\boldsymbol{X}_{t+1:t+T_f}|\boldsymbol{C}, \rho), i = 1, \dots, M$. The larger ρ is, the more diverse samples will be generated and focuses on the rare cases, and the smaller ρ is, the prediction will focus more on the most likely modes. For simplicity, we use $\boldsymbol{X}$ represent $\boldsymbol{X}_{t+1:t+T_f}$ in the case that the time step index is not necessary.

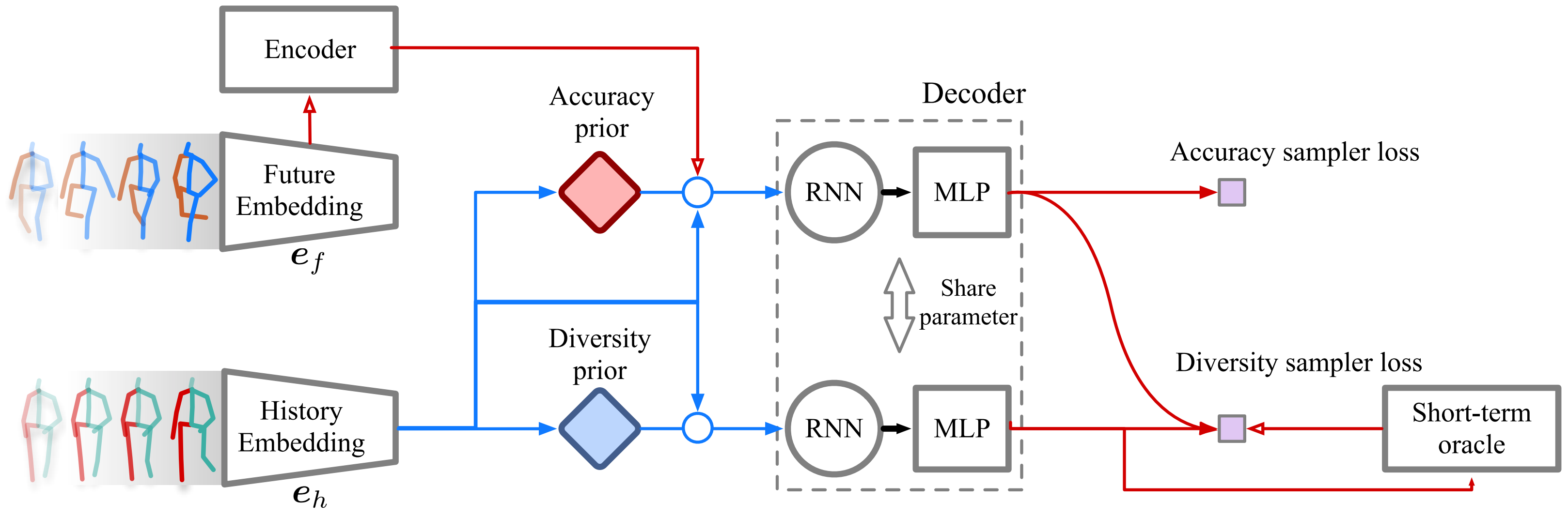


Figure 5.2: Overview of the proposed framework. Red lines indicate the pipeline, which are only used during training. Blue lines indicate the pipeline used in both the training and testing phase. During training, several samples are generated from both accuracy prior function (red diamond, Section 5.4.1) and diversity prior function (blue diamond, Section 5.4.1). The accuracy prior function will be only updated by accuracy sampler loss defined in Section 5.4.1. The diversity prior function will be updated by the diversity sampler loss, which depends on all the samples. The short-term oracle function is introduced in Section 5.4.2.

5.4 Methodology

We first introduce the multi-objective generative prediction framework based on conditional variational inference. Then we introduce the proposed short-term oracle, which provides multi-modal supervision to the prediction framework. Finally, we introduce our proposed approach's training strategy and testing procedure. The overall framework is illustrated in Figure 5.2.

5.4.1 Multi-Objective Predictor

In general, we can represent a probabilistic distribution via a latent variable model:

$$P(\boldsymbol{X}|\boldsymbol{C};Q) = \mathbb{E}_{\boldsymbol{Z}\sim Q(\boldsymbol{Z}|\boldsymbol{C})}[P(\boldsymbol{X}|\boldsymbol{C},\boldsymbol{Z})], \tag{5.1}$$

where $Q(\boldsymbol{Z}|\boldsymbol{C})$ is the conditional prior distribution of latent variable $\boldsymbol{Z} \in \mathbb{R}^{d_z}$ whose dimension is d_z. $P(\boldsymbol{X}|\boldsymbol{C},\boldsymbol{Z})$ is defined as the conditional likelihood given the observation information $\boldsymbol{C}$ and latent variable $\boldsymbol{Z}$. We can vary the prior distribution Q to achieve different distributions of $\boldsymbol{X}$ given the same observation $\boldsymbol{C}$. In our proposed approach, we introduce two different prior distributions $Q_{\text{acc}}(\boldsymbol{Z}|\boldsymbol{C})$ and $Q_{\text{div}}(\boldsymbol{Z}|\boldsymbol{C})$. We intend to estimate the data distribution $P_{\mathcal{D}}$ using $P(\boldsymbol{X}|\boldsymbol{C};Q)$ with prior $Q_{\text{acc}}(\boldsymbol{Z}|\boldsymbol{C})$, and get the most diverse distribution which mainly focuses on the minor modes by sampling from $Q_{\text{div}}(\boldsymbol{Z}|\boldsymbol{C})$. The overall framework is illustrated in Figure 5.2. Similar to [179], we define the historical observation encoder $\mathrm{e}_{\boldsymbol{h}}(\boldsymbol{C})$ and future information encoder $\mathrm{e}_{\boldsymbol{f}}(\boldsymbol{X})$ as

$$\begin{aligned} \mathrm{e}_{\boldsymbol{h}}(\boldsymbol{C}) &= [\text{MLP} \circ \text{RNN}](\boldsymbol{C}) \\ \mathrm{e}_{\boldsymbol{f}}(\boldsymbol{X}) &= [\text{MLP} \circ \text{RNN}](\boldsymbol{X}), \end{aligned} \tag{5.2}$$

where we first encode the temporal information of trajectories by using a recurrent neural network (RNN) and then use a forward neural network to map the states of RNN to the feature embedding space. Based on the historical embedding $\mathrm{e}_{\boldsymbol{h}}(\boldsymbol{C})$ and latent variable $\boldsymbol{Z}$, We denote the decoder function $\mathrm{d}_\theta(\boldsymbol{X}|\boldsymbol{C},\boldsymbol{Z})$ as:

$$\mathrm{d}_\theta(\boldsymbol{X}|\boldsymbol{C},\boldsymbol{Z}) = [\text{MLP} \circ \text{RNN}](\mathrm{e}_{\boldsymbol{h}}(\boldsymbol{C})||\boldsymbol{Z}), \tag{5.3}$$

where θ is the parameter of the decoder. In general, the outputs of the decoder are the parameters of a probabilistic distribution, e.g., the mean and variance of a Gaussian distribution. In this work, we use a deterministic decoder, and the output of the decoder is the predicted poses. For convenience, the output of the decoder is also denoted by Equation 5.3. The randomness of the decoder is only dependent on $\boldsymbol{Z}$. "||" represents the concatenate operator of two vectors. We use a similar neural network structure for the decoder with the encoders. The details of the operator $\circ$ are in Section 5.8.

Accuracy Sampler

The first objective is to infer the accuracy prior distribution $Q_{\text{acc}}(\boldsymbol{Z}|\boldsymbol{C})$. We intend to approximate the data distribution by sampling from the accuracy prior distribution. Hence, we apply the variational inference to maximize the evidence lower bound (ELBO) of the log-likelihood:

$$\begin{aligned}\mathcal{L}_{\text{ELBO}} &= \mathbb{E}_{Q_\psi(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{C})}[\log P_\theta(\boldsymbol{X}|\boldsymbol{Z},\boldsymbol{C})] \\ &- D_{KL}[Q_\psi(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{C})||Q_{\text{acc}}(\boldsymbol{Z}|\boldsymbol{C})],\end{aligned} \tag{5.4}$$

where $Q_\psi(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{C})$ is the posterior distribution of latent variable $\boldsymbol{Z}$ given the historical observation and future information. There are some works [18, 147, 160, 191] investigating the collapse problems for conditional variational inference. Those works argue that using a universal prior distribution, i.e., an independent isotropic Gaussian distribution, may not be a good choice for conditional distribution estimation [18, 147]. It is difficult to capture complex conditional multi-modal data and introduce strong model bias resulting in missing modes [160, 191]. Hence, instead of using an isotropic Gaussian distribution $\mathcal{N}(0,\mathbf{I})$ which is independent of $\boldsymbol{C}$, we model $Q_{\text{acc}}(\boldsymbol{Z}|\boldsymbol{C})$ as a Gaussian distribution $\mathcal{N}(\mu_{\phi_{\text{acc}}}(\boldsymbol{C}),\Sigma_{\phi_{\text{acc}}}(\boldsymbol{C}))$. The $D_{KL}[Q_\psi||Q_{\text{acc}}]$ is:

$$\frac{1}{2}[\log\frac{|\Sigma_{\phi_{\text{acc}}}|}{|\Sigma_\psi|} - n_z + Tr(\Sigma^{-1}_{\phi_{\text{acc}}}\Sigma_\psi) + ||\mu_{\phi_{\text{acc}}} - \mu_\psi||^2_{\Sigma^{-1}_{\phi_{\text{acc}}}}], \tag{5.5}$$

which can be calculated analytically. Since we have no control of the distribution $Q_{\text{acc}}(\boldsymbol{Z}|\boldsymbol{C})$, it could be arbitrarily distribution and it will increase the difficulty of training. In order to constrain the prior distribution, we use the best-of-many loss as the regularization of the prior model:

$$\begin{aligned}\mathcal{R}_{\text{acc}} &= \min_i \|\hat{\boldsymbol{X}}^i - \boldsymbol{X}\|^2 \\ \boldsymbol{z}^i &\sim Q(\boldsymbol{Z}|\boldsymbol{C}) \\ \hat{\boldsymbol{X}}^i &= \mathrm{d}_\theta(\boldsymbol{X}|\boldsymbol{C},\boldsymbol{z}^i), i = 1,\dots,n_{\text{acc}},\end{aligned} \tag{5.6}$$

where n_{acc} is the number of samples. Then the overall loss for the accuracy sampler is:

$$\mathcal{L}_{\text{A}}(\theta,\psi) = -\lambda_{\text{ELBO}}\mathcal{L}_{\text{ELBO}} + \lambda_{\text{acc}}\mathcal{R}_{\text{acc}}, \tag{5.7}$$

where λ_{elbo} and λ_{acc} are used to balance two losses.

Diversity Sampler

In order to explore the different modes of possible future poses, we propose to learn another prior distribution $Q_{\text{div}}(\boldsymbol{Z}|\boldsymbol{C})$ with parameter ϕ_{div}. We utilize a common diversity loss definition:

$$
\begin{aligned}
\mathrm{DIV}(\mathcal{X},\mathcal{Y}) &= \frac{1}{N_xN_y}\sum_{i,j} e^{-d(\boldsymbol{X}^i,\boldsymbol{Y}^j)} \\
\boldsymbol{X}^i, \boldsymbol{Y}^j &\in \mathcal{X},\mathcal{Y}, i=1,\dots,N_x, j=1,\dots,N_y,
\end{aligned}
\tag{5.8}
$$

where $\mathcal{X}$ and $\mathcal{Y}$ represent two sets of samples with size N_x and N_y. $d(\cdot,\cdot)$ is a metric defined in the Euclidean space. We define the metric as $d(x,y)=\eta||x-y||_2$, where η is a parameter to determine the sensitivity of the distance between two samples. We denote the set of the samples which are generated by the accuracy sampler as $\mathcal{X}_{\text{acc}}$ and the set of samples generated by the diversity sampler as $\mathcal{X}_{\text{div}}$. Then we define the diversity loss as:

$$
\mathcal{L}_{\text{div}} = \alpha_{\text{div}}\mathrm{DIV}(\mathcal{X}_{\text{div}},\mathcal{X}_{\text{div}}) + (1-\alpha_{\text{div}})\mathrm{DIV}(\mathcal{X}_{\text{div}},\mathcal{X}_{\text{acc}}),
\tag{5.9}
$$

where $\mathrm{DIV}(\mathcal{X}_{\text{div}},\mathcal{X}_{\text{div}})$ represents the diversity of samples generated by the diversity sampler. $\mathrm{DIV}(\mathcal{X}_{\text{div}},\mathcal{X}_{\text{acc}})$ represents the average pairwise distance between the samples from accuracy and diversity sampler. In the previous works, when the weight of diversity loss is large, it will have a negative influence on the accuracy sampler to approximate the data distribution. Since we intend to disentangle the accuracy objective and diversity objective, we only increase the pairwise distances between samples from the diversity sampler by using the first term in Equation 5.9 , and we make the samples from the diversity sampler dissimilar to the samples from the accuracy sampler by using the second term in Equation 5.9. We can determine the relative importance of the two items in 5.9 by a weight α_{div}. A larger α_{div} means that we focus on making the samples from Q_{div} more different.

Only using the diversity loss is not enough to get a realistic prediction since it is possible to increase diversity in the wrong way. For instance, one model can generate random noises or arbitrary invalid poses. Hence, we need to use human motion in the data to constrain the prediction. In order to constrain each generated poses from the diversity sampler, we assume that there exists an oracle:

$$
\tilde{\boldsymbol{X}}_{t+1:t+\tau} \sim \mathcal{O}(\boldsymbol{X}_t,\tau),
\tag{5.10}
$$

where $\mathcal{O}(\boldsymbol{X}_t,\tau)$ is the probabilistic distribution of future poses with horizon τ given the current initial pose $\boldsymbol{X}_t$. The oracle $\mathcal{O}$ can be seen as a teacher to distill the "knowledge" of the future poses into the predictor. Based on the oracle, we define a sample-based loss:

$$
\begin{aligned}
\mathcal{L}_{\text{ref}}(\tau) = \frac{1}{n_{\text{div}}}\sum_{i,s}\min_j \|\hat{\boldsymbol{X}}^i_{s\tau+1:(s+1)\tau} - \tilde{\boldsymbol{X}}^j_{s\tau+1:(s+1)\tau})\|^2, \\
s.t. \boldsymbol{z}^i \sim Q_{\text{div}}(\boldsymbol{Z}|\boldsymbol{C}), \hat{\boldsymbol{X}}^i_{1:T} = \mathrm{d}_\theta(\boldsymbol{X}|\boldsymbol{C},\boldsymbol{z}^i), \\
\tilde{\boldsymbol{X}}^j_{s\tau+1:(s+1)\tau} \sim \mathcal{O}(\hat{\boldsymbol{X}}_{s\tau},\tau), \\
i=1,\dots,n_{\text{div}}, j=1,\dots,n_{\text{o}}, s=0,\dots,T/\tau-1,
\end{aligned}
\tag{5.11}
$$

where τ represents the time interval of predicted poses from the oracle. n_{div} is the number of samples generated from diversity prior, and n_{o} is the number of samples which the oracle provides. W.l.o.g, we assume that the current time step is 0, and the prediction horizon is T. Given one sample $\hat{\boldsymbol{X}}^i_{1:T}$, the oracle provides several possible short-term futures $\tilde{\boldsymbol{X}}^j_{s\tau+1:(s+1)\tau}$ given the current predicted pose $\hat{\boldsymbol{X}}^i_{s\tau}$ recursively. We enforce the short-term predicted sequence $\hat{\boldsymbol{X}}^i_{s\tau+1:(s+1)\tau}$ to be similar with one of the provided futures $\tilde{\boldsymbol{X}}^j_{s\tau+1:(s+1)\tau}$. Notice that the diversity loss $\mathcal{L}_{\text{div}}$ defined in Equation 5.9 will encourage the predictor to choose one of the provided future human motions which is useful to increase the diversity. The illustration of the oracle supervision procedure is shown in Figure 5.3.

We also adopt several widely-used physical feasibility losses [179, 113, 180] such as the limbs' constraint $\mathcal{L}_{\text{limb}}$ and the velocity constraint $\mathcal{L}_{\text{vel}}$ as $\mathcal{L}_{\text{phy}}$:

$$\mathcal{L}_{\text{phy}} = \lambda_{\text{vel}}\mathcal{L}_{\text{vel}} + \mathcal{L}_{\text{limb}}. \tag{5.12}$$

The details of each item in Equation 5.12 are provided in Section 5.8. Therefore, the overall loss for the diversity sampler is:

$$\mathcal{L}_{\text{D}} = \lambda_{\text{ref}}\mathcal{L}_{\text{ref}} + \lambda_{\text{div}}\mathcal{L}_{\text{div}} + \mathcal{L}_{\text{phy}}, \tag{5.13}$$

where λ_{ref} and λ_{div} decide the importance of losses. Besides, we use a low-pass filter to smooth the predicted poses generated by the diversity sampler after training. Please see the details in Section 5.8.

5.4.2 Short-term Oracle Design

We introduce an oracle to supervise the predictor in Section 5.4.1. In this section, we discuss how to obtain the oracle. We propose to learn a short-term oracle $\mathcal{O}(\boldsymbol{X}, \tau)$ by using another conditional variational autoencoder to capture the pseudo-ground-truth multi-modality. In order to achieve such goal, several works utilize the similarity search techniques [176]. This method is also used in [178, 179] as the multi-modality evaluation metrics. In our work, we define:

$$\begin{aligned}
&\Omega(\boldsymbol{X}_t) = \mathbf{S}(\mathcal{X}_{\text{o}}; \tau, K)\\
&\mathcal{X}_{\text{o}} = \{\boldsymbol{X}^1_{t+1:t+\tau} \dots \boldsymbol{X}^{|\mathcal{X}_{\text{o}}|}_{t+1:t+\tau}\}\\
&d(\boldsymbol{X}^j_t, \boldsymbol{X}_t) \leq \delta, \forall j = 1, \dots, |\mathcal{X}_{\text{o}}|,
\end{aligned} \tag{5.14}$$

where $\mathcal{X}_o$ represents the set of all the future poses whose corresponding initial poses $\boldsymbol{X}^j_t$ are in a ball with radius δ which centered at the given initial pose $\boldsymbol{X}_t$. The ball is defined by metric $d(\cdot, \cdot)$. $\Omega(\boldsymbol{X}_t)$ represents the set of K selected future poses which has time horizon τ given the initial pose $\boldsymbol{X}_t$. Since there can be many similar poses to the given initial poses and most of the corresponding future poses are very similar, we need to select a proper fixed number of future poses in $\mathcal{X}_{\text{o}}$ in order to capture the different modes. Here we use the k-determinantal point process (k-DPP) as the selection strategy $\mathbf{S}$ to choose the future poses.

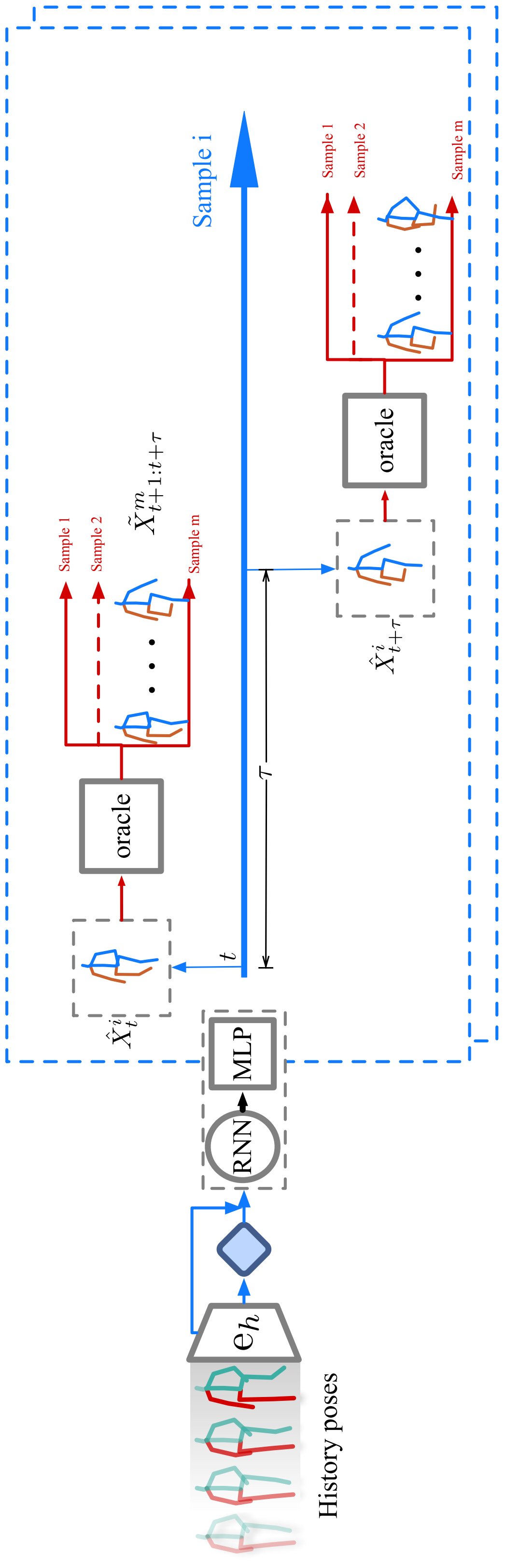


Figure 5.3: The procedure of short-term oracle supervision. During training, we can get several predicted human motions. For each sample (indicated by the blue arrow), the poses will be fed to the oracle after each τ time steps. The oracle will provide several possible future poses as options. The predicted human motions only need to be similar to one of the options in each short time horizon.

k-Determinantal Point Process

k-determinantal point process [86] is widely used to sample the diverse points given a fixed number of samples. Given a set $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$, a k-determinantal point process defined on $\mathcal{X}$ is a probability measure on $2^{\mathcal{X}}$:

$$\mathbf{Pr}(\mathcal{S}) = \frac{\det(L_{\mathcal{S}})\mathbf{1}(|\mathcal{S}| = k)}{\sum_{\mathcal{S}\subset[n],|\mathcal{S}|=k} \det(L_{\mathcal{S}})}, \tag{5.15}$$

where we denote $\mathcal{S}$ as a subset of $\mathcal{X}$ and $L_S \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ as the similarity matrix:

$$\{L_{\mathcal{S}}\}_{ij} = e^{-d(X^i_{t+1:t+\tau}, X^j_{t+1:t+\tau})}. \tag{5.16}$$

We preprocess the training data to augment each case with K futures poses. Several sampling algorithms [40, 65] for the determinantal point process can be used directly.

Short-term Oracle Model

The short-term oracle can be trained with any approach proposed in Section 5.2. In our experiments, we use a conditional variational autoencoder similar to the likelihood sampler defined above after getting the augmented data with the prediction horizon τ. Now, we can provide more diverse futures given the exact same historical observation. Since the augmented data is balanced by the k-determinantal point process, there will be fewer extremely minor modes and hence mitigate the trouble of rare-case sampling. The details of the short-term oracle neural network structure are provided in Section 5.8.

5.5 Training and Testing Process

The training procedure is summarized in Algorithm 2. We generate the same number of samples from both accuracy prior and diversity prior for training. Notice that the diversity loss $\mathcal{L}_{\text{div}}$ does not backpropagate to the accuracy prior $Q_{\text{acc}}(\boldsymbol{Z}|\boldsymbol{C})$ since we do not want the diversity loss influence the accuracy prior.

After we get the optimized model, we can decide the ratio of diverse samples, which mainly focus on the most different modes compared with the major modes by adjusting the ratio number ρ. The testing procedure is summarized in Algorithm 3.

5.6 Experiments

In this section, we introduce the datasets and evaluation metrics first. Then the quantitative, qualitative analysis, and ablation analysis are provided. Implementation details, additional results, limitations, and future work are provided in Section 5.8.

Algorithm 2: Training Procedure

```
Input: $N$: number of epoches. $n_{\text{acc}}$: number of samples for accuracy sampler $Q_{\text{acc}}$.
       $n_{\text{div}}$: number of samples for diversity sampler $Q_{\text{div}}$. $n_o$: number of samples
       generated from oracle $\mathcal{O}$.
Output: $\theta$, $\phi_{\text{acc}}$, $\phi_{\text{div}}$
Data: Training dataset $\mathcal{D}_{train}$
1  while $epoch \leq N$ do
2      Sample $\mathcal{B} = \{\boldsymbol{X}^i, \boldsymbol{C}^i\}_i \sim \mathcal{D}_{train}$
3      foreach $\boldsymbol{X}, \boldsymbol{C} \in \mathcal{B}$ do
4          Generate $n_{\text{acc}}$ samples:
5          $\hat{\boldsymbol{X}}^i_{\text{acc}} = \text{d}(\boldsymbol{X}|\boldsymbol{C}, \boldsymbol{z}^i), \boldsymbol{z}^i \sim Q_{\text{acc}}(\boldsymbol{Z}|\boldsymbol{C})$
6          Generate $n_{\text{div}}$ samples:
7          $\hat{\boldsymbol{X}}^i_{\text{div}} = \text{d}(\boldsymbol{X}|\boldsymbol{C}, \boldsymbol{z}^i), \boldsymbol{z}^i \sim Q_{\text{div}}(\boldsymbol{Z}|\boldsymbol{C})$
8          for $s = 0, \ldots, T_f/\tau - 1$ do
9              Generate $n_o$ samples:
10             $\tilde{\boldsymbol{X}}^j_{t+s\tau+1:t+(s+1)\tau} \sim \mathcal{O}(\hat{\boldsymbol{X}}_{\text{div},t+s\tau}, \tau)$
11     Update $\theta, \psi, \phi_{\text{acc}}$ with $\mathcal{L}_{\text{A}}$
12     Update $\theta, \phi_{\text{div}}$ with $\mathcal{L}_{\text{D}}$
```

Algorithm 3: Testing Procedure

```
Input: $\rho$: The proportion of samples from $Q_{\text{div}}$ in the total samples , $M$: the total
       number of samples
Output: $\hat{\boldsymbol{X}}$, The predicted poses
Data: Testing Dataset $\mathcal{D}_{test}$
1  foreach $\boldsymbol{X}, \boldsymbol{C} \in \mathcal{D}_{test}$ do
2      Generate $(1-\rho)M$ samples from $Q_{\text{acc}}$
3      Generate $\rho M$ samples from $Q_{\text{div}}$
```

5.6.1 Datasets

We evaluate our method on Human3.6M [68] and HumanEva-I dataset [146] and use identical settings with the other baselines. Human3.6M dataset consists of 11 subjects and 3.6 million video frames. There are 15 actions for each subject. The human motion is recorded at 50Hz. We adopt a 17-joint skeleton representation in our work. We use five subjects (S1, S5, S6, S7, S8) for training and testing with the other two subjects (S9 and S11). The predicted future motion horizon is 2 seconds (100 time steps), and the historical motion horizon is 0.5 seconds (25 time steps). HumanEva-I dataset includes three subjects. The record rate of human motion is 60Hz. We choose to use the 15-joint skeleton representation. We use the same training and testing datasets which are provided by the official website. We predict future motion for 1 second (60 time steps) with 0.25 seconds (15 time steps) observation.

5.6.2 Evaluation Metrics

The following metrics are used to evaluate the performance of methods. For accuracy, we use Average Displacement Error (ADE) which is defined as the average Euclidean distance over the prediction time steps between the ground truth motion $\boldsymbol{X}_{t+1:t+T_f}$ and the closest sample [179], and Final Displacement Error (FDE), which is the Euclidean distance between the final ground truth pose and the final predicted pose, i.e., $\min_i \|\hat{\boldsymbol{X}}^i_{t+T_f} - \boldsymbol{X}_{t+T_f}\|$. For diversity, we use Average Pairwise Distance (APD), which is the L2 distance between all pairs of motion samples, which is computed as $\frac{1}{K(K-1)}\sum_{i\neq j}\|\hat{\boldsymbol{X}}^i_{t+1:t+T_f} - \hat{\boldsymbol{X}}^j_{t+1:t+T_f}\|$.

5.6.3 Quantitative Analysis

We compare our approach with several baselines in Table 5.1. The baselines include deterministic methods such as acLSTM [101] and ERD [53], probabilistic approaches such as MT-VAE [177] and Dlow [179], etc. We use 50 samples to evaluate the prediction performance for all methods. We directly use the results of baselines from [179] and [188].

In Table 5.1 we can conclude that our method with $\rho = 0.46$ can achieve a better performance compared with the other baselines in terms of all the metrics. In general, probabilistic methods such as Best-of-Many and GMVAE can achieve better accuracy and diversity than deterministic ones such as acLSTM and ERD. We can observe that most of the methods will have worse ADE and FDE if the APD is larger in general. It is because there exists a trade-off between diversity and accuracy. Compared with DLow, our approach improve the performance on both Human3.6M and HumanEva-I datasets. We also compare our results with DCT5 and DCT20 in [188], which use the frequency representation with the CVAE framework. Our results are on par with their performance on both datasets.

Table 5.1: Quantitative results on Human3.6M and HumanEva-I dataset. Our results and the best results of baselines are highlighted.

Human3.6M						
	ERD	acLSTM	Pose-Knows	MT-VAE	HP-GAN	BoM
APD ↑	0	0	6.723	0.403	7.214	6.265
ADE ↓	0.722	0.789	0.461	0.457	0.858	0.448
FDE ↓	0.969	1.126	0.560	0.595	0.867	0.533

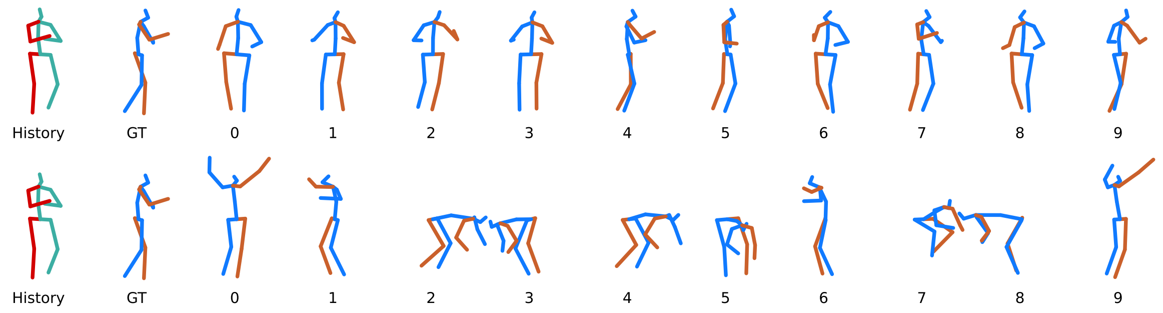
Human3.6M						
	GMVAE	DeLiGAN	DSF	Dlow	DCT5/DCT20	Ours
APD ↑	6.769	6.509	9.330	11.74	12.579/**15.920**	**14.24**
ADE ↓	0.461	0.483	0.493	0.425	**0.412**/0.416	**0.414**
FDE ↓	0.555	0.534	0.592	0.518	**0.514**/0.522	**0.516**

HumanEva-I						
	ERD	acLSTM	Pose-Knows	MT-VAE	HP-GAN	BoM
APD ↑	0	0	2.308	0.021	1.139	2.846
ADE ↓	0.382	0.429	0.269	0.345	0.772	0.271
FDE ↓	0.461	0.541	0.296	0.403	0.749	0.279

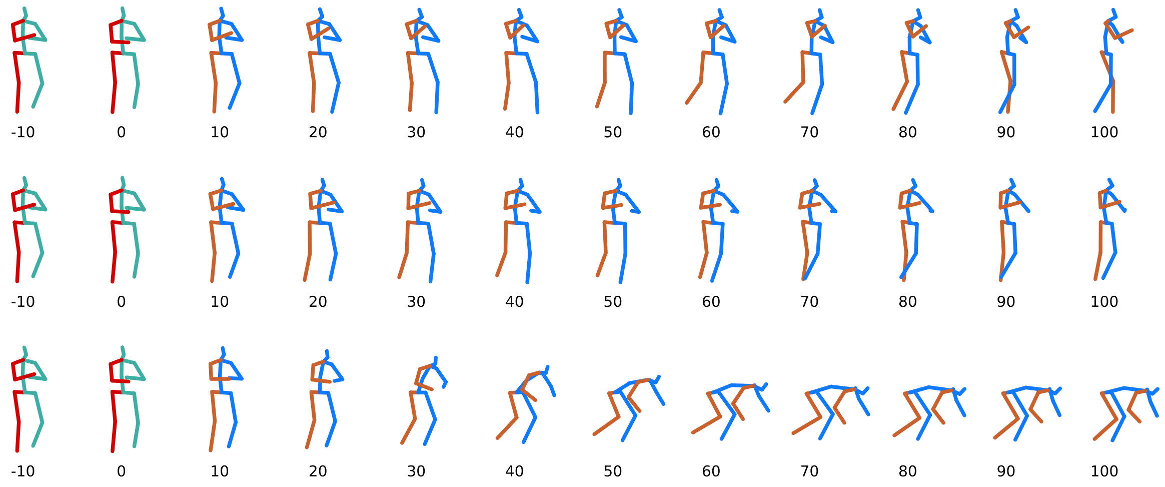
HumanEva-I						
	GMVAE	DeLiGAN	DSF	Dlow	DCT5/DCT20	Ours
APD ↑	2.443	2.177	4.538	4.855	4.181/**6.266**	**5.786**
ADE ↓	0.305	0.306	0.273	0.251	**0.234**/0.239	**0.228**
FDE ↓	0.345	0.322	0.290	0.268	**0.244**/0.253	**0.236**

5.6.4 Qualitative Analysis

We illustrate 10 end poses of random samples generated from both accuracy prior function and diversity function in Figure 5.4a and 5.5a. The first row shows the samples from the accuracy prior function. We notice that most samples are similar to the ground truth, which represents that the accuracy sampler can generate the predicted future human motions with high accuracy. The second row shows the samples from the diversity sampler. We notice



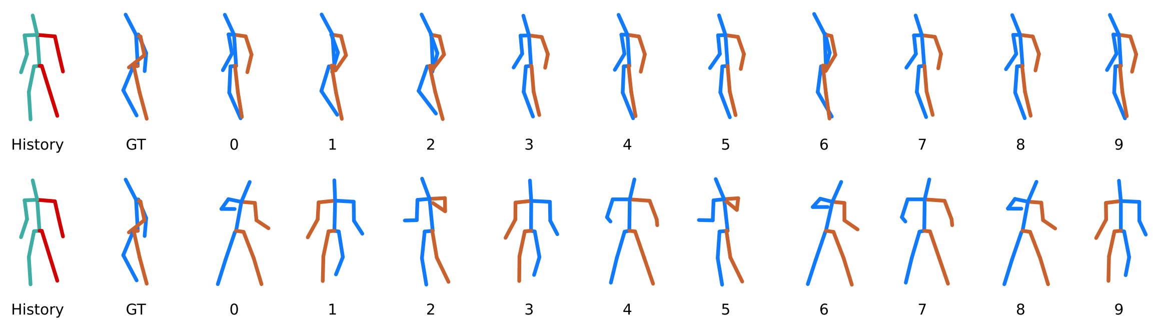
(a) Predicted end poses on Human3.6M dataset.



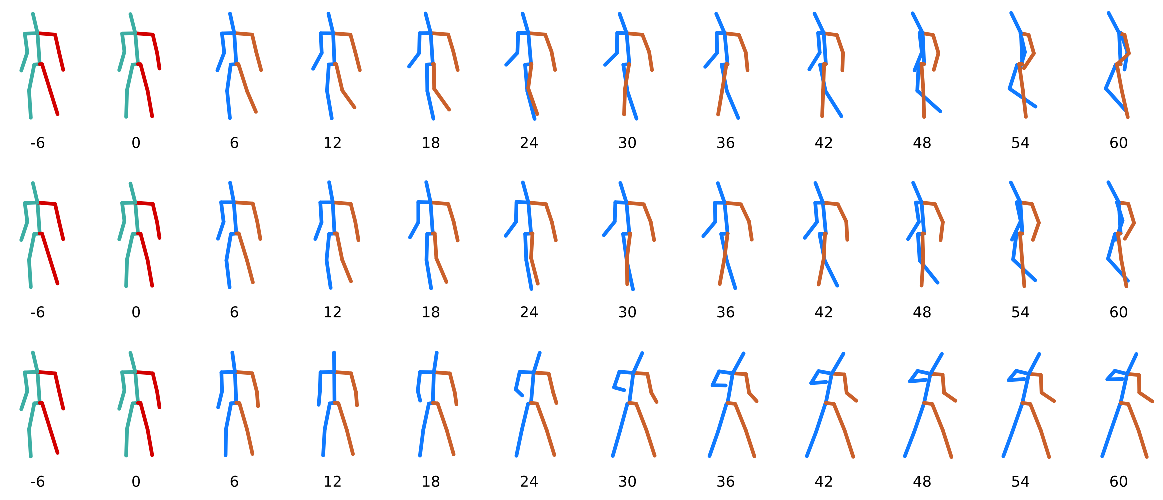
(b) Predicted human motion on Human3.6M dataset.

Figure 5.4: Visualization of prediction results on Human3.6M dataset. Figure 5.4a illustrates ten predicted end poses generated from both accuracy prior (first row) and diversity prior (second row). Figure 5.4b shows the predicted time sequences. The sequence in the first row is the ground truth motion. The sequence in the second row is one of the samples generated from accuracy prior and the sequence in the third row is one of the samples generated from diversity prior.

that the predicted poses from the diversity sampler have more different modes and are not similar to the samples generated from the accuracy sampler. It can be attributed to the second item in the diversity loss $\mathcal{L}_{\text{div}}$ in Equation 5.9, where we encourage our diversity prior to generating dissimilar samples to the ones generated from the accuracy sampler. We also illustrate two samples of predicted human motion from both accuracy and diversity sampler for both datasets in Figure 5.4b and Figure 5.5b. We notice that the predicted



(a) Predicted end poses on HumanEva-I dataset.



(b) Predicted human motion on HumanEva-I dataset.

Figure 5.5: Visualization of prediction results on HumanEva-I dataset. Figure 5.5a illustrates ten predicted end poses generated from both accuracy prior (first row) and diversity prior (second row). Figure 5.5b shows the predicted time sequences. The sequence in the first row is the ground truth motion. The sequence in the second row is one of the samples generated from accuracy prior and the sequence in the third row is one of the samples generated from diversity prior.

time sequences are smooth. The samples from the accuracy sampler can be very accurate compared with the ground truth. More visualization results are provided in Section 5.8.

5.6.5 Different Sampling Ratio

In Figure 5.6, we illustrate the different metrics values with respect to the number of samples n_{acc} generated from the accuracy sampler during testing. When n_{acc} equals 0, it means that we only sample from the diversity prior distribution. We can see that ADE and FDE increase since the diversity sampler is designed to focus on exploring more different possible modes instead of matching the likelihood of data. Hence, we observe that APD can achieve around 18 when $n_{\text{acc}} = 0$. When n_{acc} increases, we observe that both the accuracy metrics (ADE and FDE) and diversity metric (APD) decrease. The accuracy metrics decrease slowly when the n_{acc} is large enough. When $n_{\text{acc}} = 50$, which means that all the samples are generated from the accuracy sampler, we observe that APD decreases to around 6 and the accuracy metrics achieve the best performance.

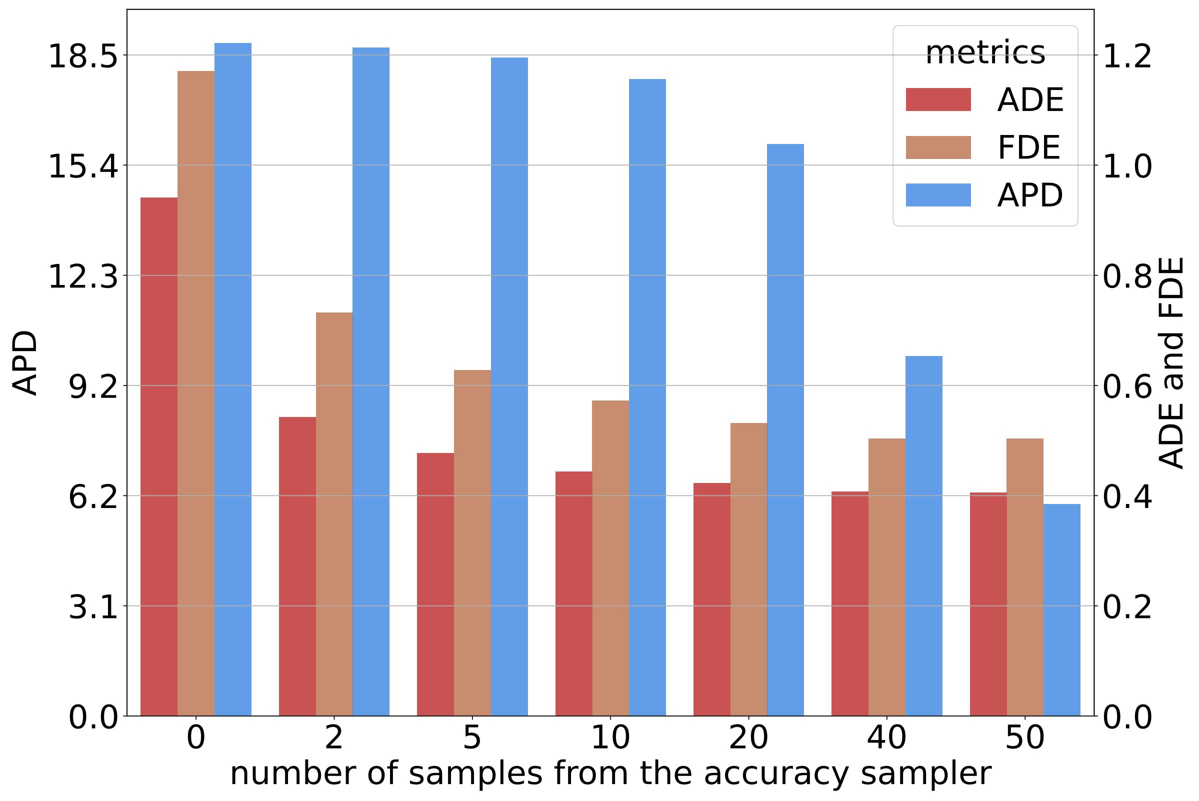
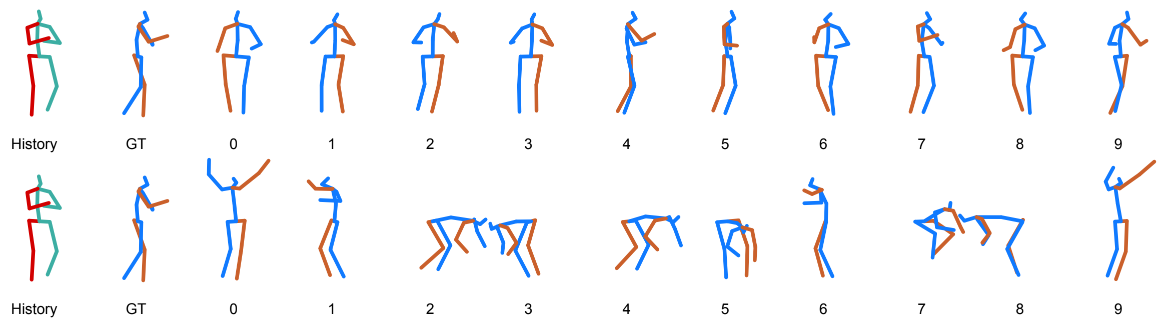
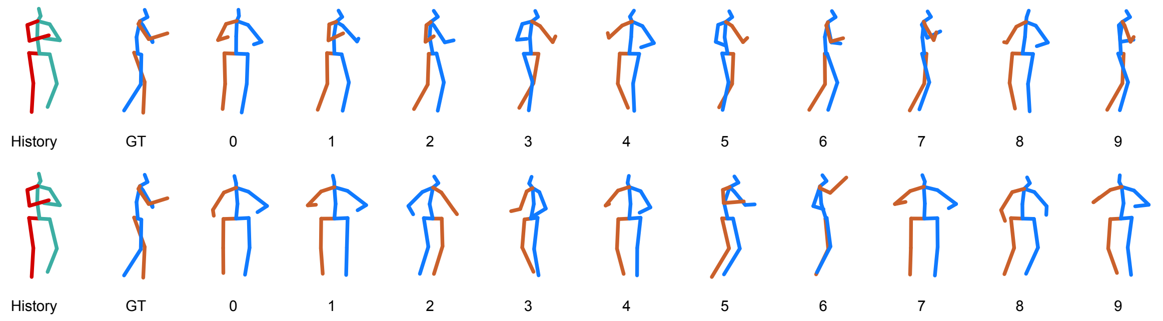


Figure 5.6: APD, ADE and FDE with respect to n_{acc} on Human3.6M dataset. Red and brown bars indicate the accuracy metrics ADE and FDE. The blue bar indicates the APD.



(a) Samples from the predictor with $\tau = 25$.



(b) Samples from the predictor with $\tau = 100$.

Figure 5.7: Visualization of predicted end poses of motions on Human3.6M dataset with different oracles. Figure 5.7a illustrates the performance of the predictor with oracle ($\tau = 25$). Figure 5.7b illustrates the performance of the predictor with oracle ($\tau = 100$). In each figure, the first row are the samples generated from accuracy prior function. The second row are the samples generated from the diversity prior function.

5.6.6 Ablation Analysis

Using Short-term Oracle Prediction Horizon τ In order to investigate whether dividing the prediction horizon into several short-term ones can help the predictor discover more possible modes, we evaluate our models with the oracles which have different prediction horizon length τ. We compare our framework supervised by a short-term oracle with $\tau = 25$ and our framework supervised by the one with the full-length of prediction horizon, i.e., the prediction is not divided into short-term subsequences. We show the results of Human3.6M dataset in Figure 5.7. We can see that when using the oracle with $\tau = 100$, i.e., the prediction horizon of the oracle is not short-term and the horizon is the same as the target prediction horizon, and the diversity is lower than the one which has $\tau = 25$. It shows that the oracle with a short prediction horizon indeed increases the diversity. We also compare the different

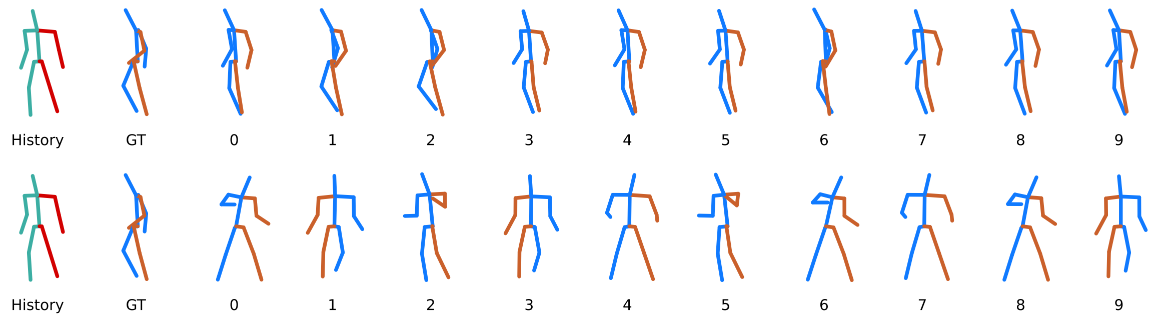
Table 5.2: The comparison with different τ on Human3.6M dataset.

	$\tau = 25$, short-term							
n_{acc}	0	7	14	21	28	35	42	50
ADE $\downarrow$	0.941	0.459	0.433	0.421	0.413	0.411	0.407	0.404
FDE $\downarrow$	1.170	0.598	0.551	0.529	0.515	0.510	0.504	0.501
APD $\uparrow$	18.79	18.16	17.14	15.73	14.04	12.02	9.265	5.927
	$\tau = 100$, non-short-term							
n_{acc}	0	7	14	21	28	35	42	50
ADE $\downarrow$	0.504	0.431	0.417	0.409	0.406	0.402	0.401	0.402
FDE $\downarrow$	0.580	0.523	0.505	0.495	0.491	0.486	0.488	0.497
APD $\uparrow$	7.346	7.397	7.360	7.234	6.997	6.683	6.255	5.651

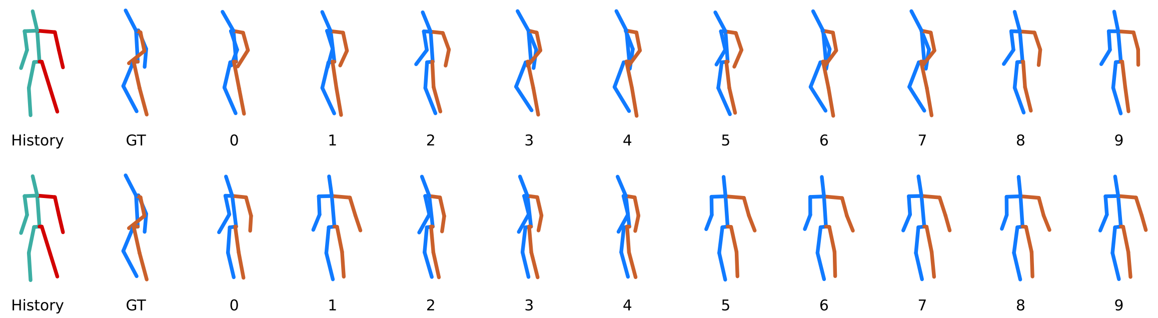
metrics of two models with different τ, and the results are summarized in Table 5.2. We notice that ADE and FDE with $n_{\text{acc}} = 50$ of both models with $\tau = 100$ and $\tau = 25$ are similar since all the samples are from the accuracy samplers. However, when n_{acc} decreases, we observe that the diversity of the model supervised by the oracle with $\tau = 100$ does not increase so much. We also observed that ADE and FDE of the model supervised with oracle (τ=100) does not change too much when n_{acc} is greater than 28 and APD does not change too much when n_{acc} is smaller than 14. It is reasonable since the model supervised by the oracle with $\tau = 100$ only explores limited and less possible diverse modes than the one supervised by the oracle with $\tau = 25$. It also supports our suggestion that the short-term oracle indeed helps the predictor discover more possible future motions meanwhile the accuracy of prediction is maintained.

The following context is the additional ablation analysis results which are corresponding to the experiments on HumanEva-I dataset. We visualize the predicted future motions in Figure 5.8 and show the quantitative comparison in Table 5.3.

The experiment results are consistent with the ones on Human3.6M dataset. Similarly, we observe that the diversity of our proposed framework with short-term oracle ($\tau = 15$) is significantly improved compared with the one with non-short-term oracle ($\tau = 100$). Since HumanEva-I is a small dataset and evaluated with a short prediction horizon, the number of different modes is intrinsically limited. We can also observe that the predicted end poses in Figure 5.8 are not diverse as the ones in Human3.6M dataset. Also, we observe that the model with a non-short-term oracle doesn't increase the diversity so much. This is also reasonable since all the modes provided by grouping the similar initial poses once can still be similar. These results also imply that using short-term oracle, i.e., grouping similar poses several times every $\tau = 15$ discover more modes.



(a) Samples from the predictor with $\tau = 15$.



(b) Samples from the predictor with $\tau = 60$.

Figure 5.8: Visualization of predicted end poses on HumanEva-I dataset with different oracles. Figure 5.8a and 5.8b illustrate the performance of the predictor with oracle ($\tau = 15$) and the predictor with oracle ($\tau = 60$) respectively. In each figure, the first and second row are the samples generated from accuracy and diversity prior function respectively.

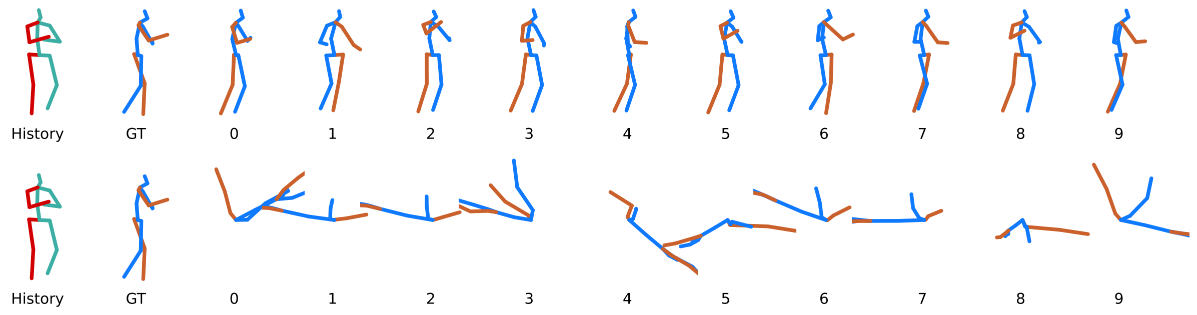
Ablation of Oracle In order to show that the short-term oracle loss in Equation 5.11 is necessary, we provide the qualitative results of the model without short-term oracle supervision in Figure 5.9. The diversity prior w/o oracle supervision indeed produces infeasible motions.

5.7 Additional Visualization Results

We illustrate more prediction cases on Human3.6M dataset in Figure 5.10 and Figure 5.11. In Figure (a), the first and second rows show the samples from the accuracy and diversity sampler, respectively. In Figure (b), the first row shows the ground truth human motion, and the second and third rows show the samples from accuracy and diversity sampler respectively. We also illustrate two more prediction cases on HumanEva-I dataset in Figure

Table 5.3: The comparison with different τ on HumanEva-I dataset.

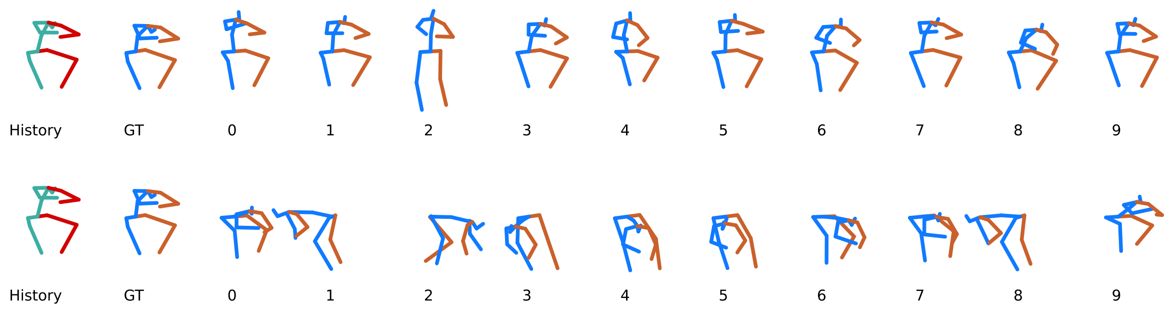
	$\tau = 15$, short-term							
n_{acc}	0	7	14	21	28	35	42	50
ADE ↓	0.561	0.241	0.231	0.227	0.228	0.227	0.228	0.229
FDE ↓	0.623	0.249	0.243	0.236	0.235	0.235	0.236	0.244
APD ↑	6.966	6.943	6.943	6.139	5.354	4.441	3.217	1.619
	$\tau = 60$, non-short-term							
n_{acc}	0	7	14	21	28	35	42	50
ADE ↓	0.345	0.236	0.229	0.227	0.226	0.226	0.226	0.233
FDE ↓	0.346	0.243	0.234	0.232	0.228	0.227	0.229	0.237
APD ↑	2.120	2.330	2.449	2.476	2.429	2.309	2.059	1.706



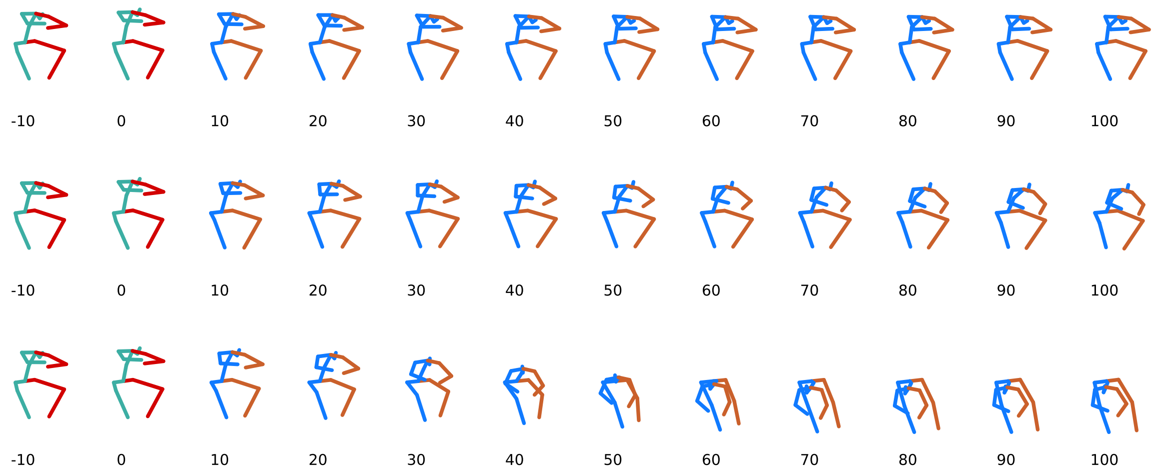
(a) Samples from the predictor with $\tau = 15$.

Figure 5.9: Predicted end poses of Ours w/o oracle. The 1st row shows samples from the accuracy prior. The 2nd row shows samples from the diversity prior trained without oracle's supervision.

5.12 and Figure 5.13. In Figure (a), the first and second rows show the samples from the accuracy and diversity sampler respectively. In Figure (b), the first row shows the ground truth human motion, and the second and third rows show the samples from accuracy and diversity sampler respectively.



(a) 10 predicted end poses.



(b) Predicted human motion.

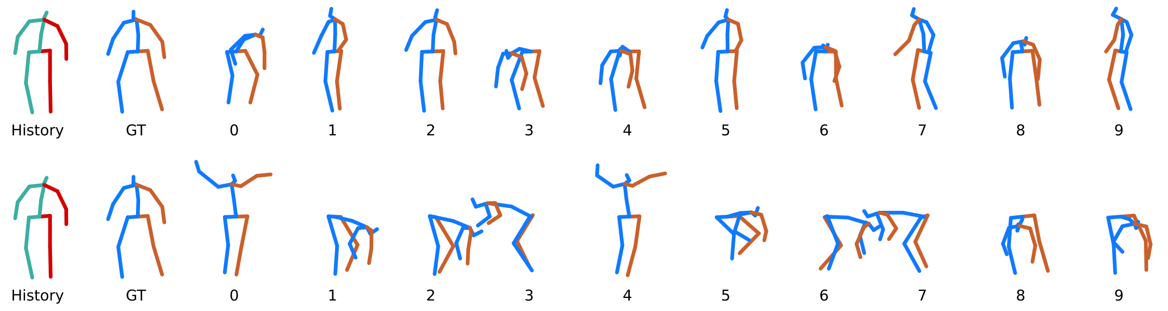
Figure 5.10: Additional Prediction Case 1 on Human3.6M dataset.

5.8 Implementation Details

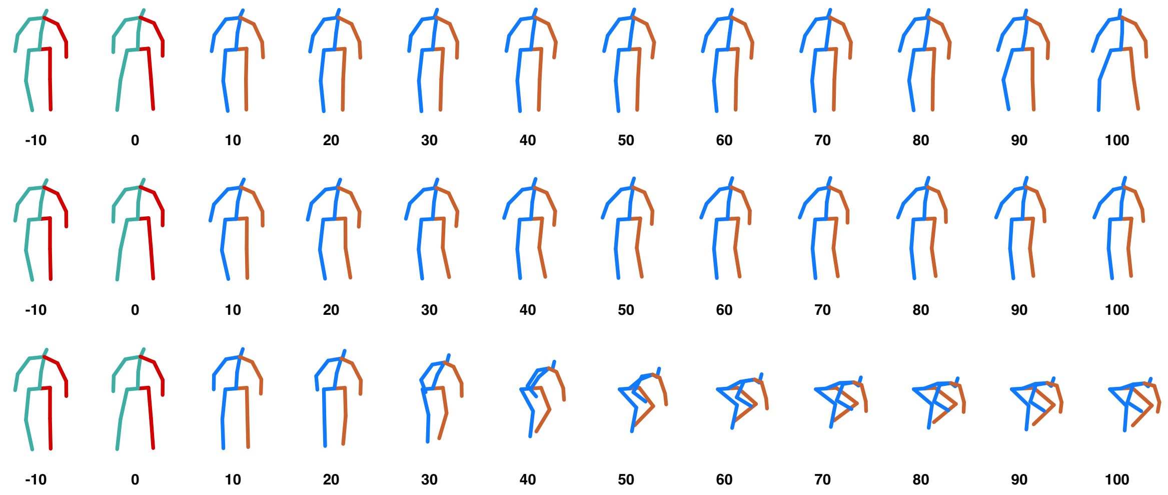
5.8.1 Physical Feasibility Loss

In this section we provide the details of the physical feasibility loss used in Equation 5.12. The velocity loss $\mathcal{L}_{\text{vel}}$ defined as the average difference between each two successive poses:

$$\mathcal{L}_{\text{vel}}(\boldsymbol{X}) = \frac{1}{T}\sum_{t=0}^{T-1} \|\boldsymbol{X}_{t+1} - \boldsymbol{X}_t\|^2, \tag{5.17}$$



(a) 10 predicted end poses.



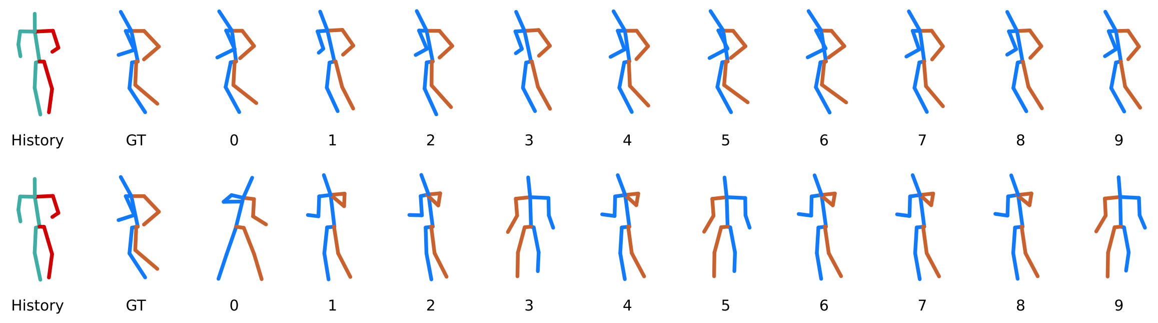
(b) Predicted human motion.

Figure 5.11: Additional Prediction Case 2 on Human3.6M dataset.

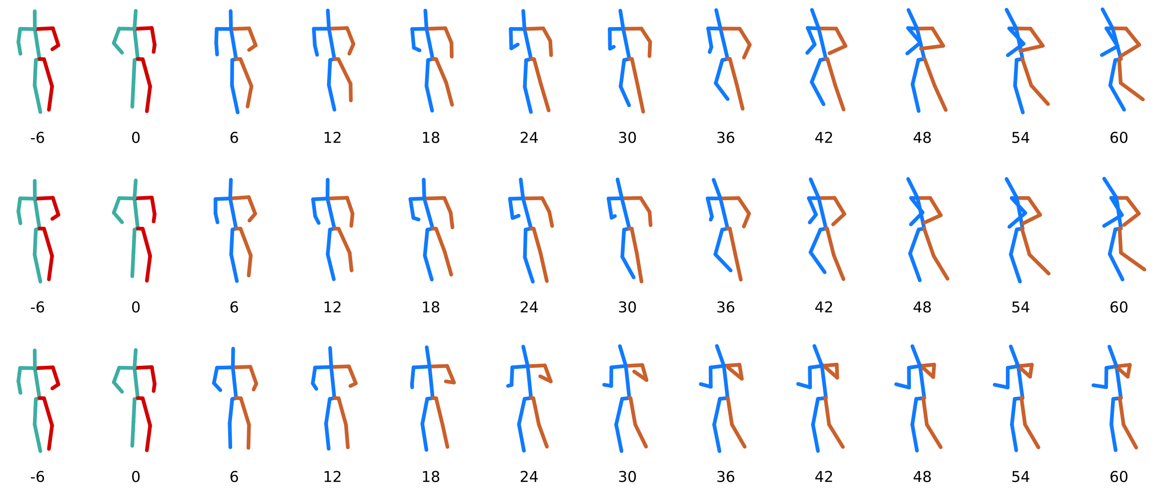
We also constraint limbs by using the following loss:

$$\mathcal{L}_{\text{limb}}(\boldsymbol{X}) = -\lambda_{\text{dir}} \log P(\boldsymbol{n}) + \frac{\lambda_{\text{len}}}{n_l T} \sum_i \sum_t (\|\hat{\boldsymbol{l}}_i(t)\| - \|\boldsymbol{l}_i\|)^2, \tag{5.18}$$

where the likelihood $\log P(\boldsymbol{n})$ is approximated by a neural spline normalizing flow [50, 113, 180]. $\boldsymbol{n} = [\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots, \boldsymbol{n}_m]$, where $\boldsymbol{n}_i$ is the normalized direction of the i-th limb. Meanwhile, we also enforce the predicted limbs' length $\|\hat{\boldsymbol{l}}_i(t)\|$ should be same as the ground truth $\|\boldsymbol{l}_i\|$. where $\hat{\boldsymbol{l}}_i(t)$ is the length of the i-th predicted limb at time step t. Besides, we use a low-pass filter to smooth the predicted poses generated by the diversity sampler after training. We use the first 4 lowest frequencies calculated by the real Fourier transform. The



(a) 10 predicted end poses.



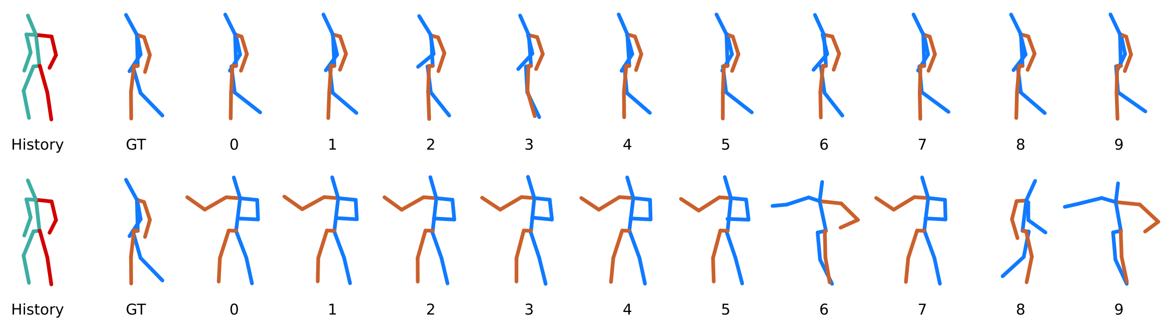
(b) Predicted human motion.

Figure 5.12: Additional Prediction Case 2 on HumanEva-I dataset.

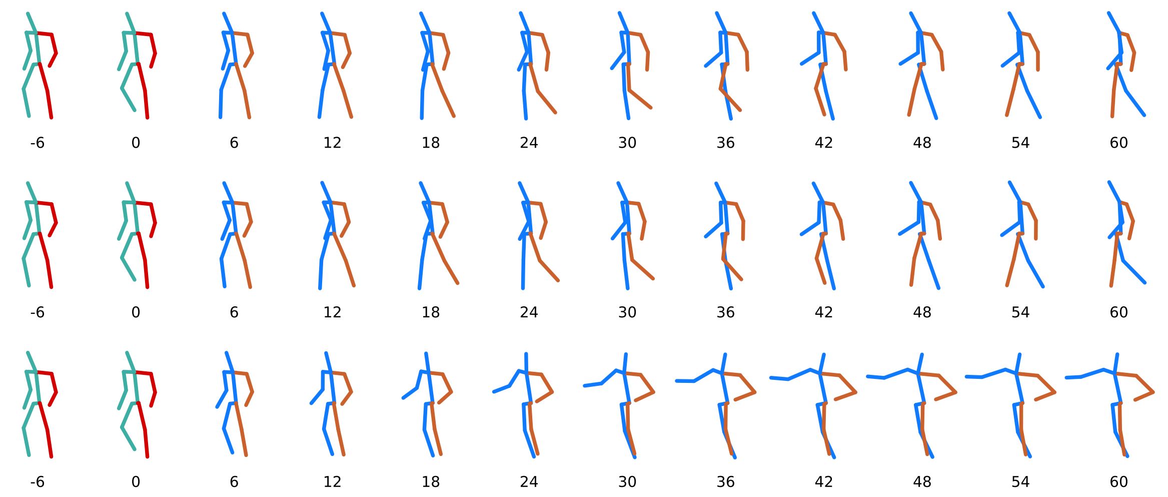
parameters $(\lambda_{\text{vel}}, \lambda_{\text{dir}}, \lambda_{\text{limb}})$ is set as $(800, 0.01, 100)$ for both Human3.6M and HumanEva-I dataset.

5.8.2 Short-term Oracle Details

We select $K = 10$ as the number of augmented future motions for k-determinantal point process. We use the same structure and loss as the accuracy sampler. The only different is the training process. Since we have multiple future motions given an observation, the loss $\mathcal{L}_o = -\mathcal{L}_{\text{ELBO}} + \lambda_{\text{acc}}\mathcal{R}_{\text{acc}}$ becomes:



(a) 10 predicted end poses.



(b) Predicted human motion.

Figure 5.13: Additional Prediction Case 2 on HumanEva-I dataset.

$$\begin{aligned}\mathcal{L}_{\text{ELBO}} = \frac{1}{K}\sum_{j=1}^{K}\mathbb{E}_{Q_\psi(\boldsymbol{Z}|\boldsymbol{X}_j,\boldsymbol{C})}[\log P_\theta(\boldsymbol{X}_j|\boldsymbol{Z},\boldsymbol{C})] \\ - D_{KL}[Q_\psi(\boldsymbol{Z}|\boldsymbol{X}_j,\boldsymbol{C})||Q_{\text{acc}}(\boldsymbol{Z}|\boldsymbol{C})],\end{aligned} \tag{5.19}$$

where, K is the number of augmented pseudo future motions. Similarly, the regularization becomes:

$$\begin{aligned} \mathcal{R}_{\text{acc}} &= \frac{1}{K}\sum_{j=1}^{K}\min_{i}\|\hat{\boldsymbol{X}}^i - \boldsymbol{X}^j\|^2 \\ \boldsymbol{z}^i &\sim Q(\boldsymbol{Z}|\boldsymbol{C}), \hat{\boldsymbol{X}}^i = \mathrm{d}_\theta(\boldsymbol{X}|\boldsymbol{C}, \boldsymbol{z}^i), i = 1, \ldots, K. \end{aligned} \tag{5.20}$$

Notice that we set the sample number of the decoder as K, too. Hence, every prediction sample will be supervised by the augmented pseudo future motions. We use one time step observation as $\boldsymbol{C}$ to predict τ time steps future motions for both Human3.6M and HumanEva-I datasets.

5.8.3 Training Parameters

The dimension n_z of latent variable $\boldsymbol{Z}$ is 128. The diversity prior function is a multiple layer neutron with two hidden layers and each layer has 512 neurons. For the accuracy prior function, we use the same historical embedding and MLP. We only use the diagonal of the covariance matrix. The dimension of the hidden state of RNN is 128. We use Adam optimizer with an exponential decay learning rate. The training batch size is 64, and the total number of epochs is 300 for Human3.6M dataset and 100 for HumanEva-I dataset. The hyperparameters for the accuracy sampler are $(\lambda_{\text{elbo}}, \lambda_{\text{acc}}) = (1.0, 2.0)$. $\lambda_{\text{div}}, \lambda_{\text{ref}}$ for the diversity sampler are set as 15, 0.3 for Human3.6M dataset with $\tau = 25$ and 10, 0.3 for HumanEva-I dataset with $\tau = 15$. For the grouping threshold, we use the same number in [179, 178]. We set the diversity sensitivity $\eta = 15$ for both datasets. For HumanEva-I dataset, we augment the dataset first for all the experiments by grouping the similar last historical pose first since the dataset is small. We use the same decoder and encoder structure as the ones in [179], and MLP$\circ$RNN represents that we use an MLP after the RNN outputs, please see details in [179].

5.9 Chapter Summary

In this chapter, we propose a multi-objective diverse human motion prediction framework, which can enable adjustable sampling during the testing time. In order to enhance the diversity of predicted poses, we introduce a short-term oracle to instruct the predictor to discover more diverse possible modes of future poses. Such a framework overcomes the trade-off between likelihood sampling and diversity sampling. Thanks to both the multi-objective structure and short-term oracle, Our proposed approach achieves state-of-the-art performance in terms of accuracy and diversity. The experiments results and ablation studies demonstrate the effectiveness of the proposed method. Several future directions could be investigated. First, since our proposed approach is a general framework, more complicated structures such as graph neural network and transformer can be incorporated. Second, we currently assume that the horizon of short-term oracle is fixed. How to dynamically decide the short-term horizon will be the future work.

Chapter 6

Differentiable Safety-Critical Control for Motion Generation

6.1 Introduction

Safety plays a critical role in autonomous systems that interact with people, such as autonomous driving and robotics. There are several approaches to developing a safe control strategy, e.g., Hamilton-Jacobi reachability analysis [13] and model predictive control [79]. However, such methods may have high computational costs in real-time applications. Control barrier functions (CBFs) [7] have gained more attention recently since these methods only depend on the current state and do not require heavy computation. CBFs are usually encoded as constraints in a quadratic program (CBF-QP) for safety-critical tasks [6]. With a properly chosen class $\mathcal{K}$ function in CBFs, a system can avoid unsafe sets. Meanwhile, it does not reduce the stabilizing performance from a high-level controller [7]. However, the performance of the overall controller, which consists of a high-level controller and CBF-QP, can be easily undermined if the environment changes. In other words, each safe set in CBFs necessitates a unique class $\mathcal{K}$ function that maximizes the overall control performance for a specific environment. In the real world, the environment information for safety-critical tasks is usually not fully known a priori, and a system might also face different environments during its deployment. Thus, it is hard to tune a class $\mathcal{K}$ function for each environment beforehand to reconcile performance and safety. Moreover, the tuning process for choosing a class $\mathcal{K}$ function becomes tedious when there are multiple control barrier function constraints in the CBF-QP [169], or some are with high relative-degree in the ECBF-QP [119]. This challenge impedes the progress towards deploying CBF-based safety-critical controllers in the real world.

To address this challenge, we investigate how to model the relation between environment information and safety-critical control. We propose a learning safety-critical control framework using an environment-dependent neural network which satisfies the forward invariance condition. Thanks to the development of differentiable convex optimization [2], we can enable the learning procedure in an end-to-end style. After offline training, we can directly deploy the proposed safety-critical control framework in different environments without any adaption.

6.1.1 Related Works

Safe environment generalization

Cluttered environments have been considered in the safe control literature. A provably approximately correct-bayes framework is proposed to synthesize controllers that provably generalize to novel environments in [110]. A control Lyapunov function and control barrier function based quadratic program (CLF-CBF-QP) is utilized with a high-level path plan to navigate through obstacle-scattered environments in [14]. Moreover, for hostile environments with adversarial agents, a probabilistic tree logic method is proposed in [35] to assure safety. Safe generalization problem with control barrier functions is considered with a weighted mixture of existing controllers in [148]. Yet the generalization ability is largely limited by the number of existing controllers. With the help of reinforcement learning, safe environment adaptation is studied through a risk-averse approach in [187]. However, this approach can only provide relative safety instead of safety guarantee. For robotic applications, bipedal robot walking on stepping stones is addressed in [120] using a robust control barrier function method, where the distances between adjacent stones are different at each step. Predictive control with CBFs tackles the safe car overtaking problems in [181], where different leading cars serve as novel environments. Our approach adopts a different way using class $\mathcal{K}$ function to generalize a controller to enforce safety under different environments.

Safe learning control

A safe reinforcement learning (RL) framework under constrained Markov decision process is proposed in [33] using a Lyapunov based method. A learning-based control barrier function from expert demonstration is proposed in [134] to ensure safety. In [141], a CBF is created using RL for risk mitigation in adversarial environments. In [32] and [155], they address the model uncertainty problem by learning CBF constraints. In [39], the authors design a learning robust control Lyapunov barrier function that can generalize despite model uncertainty. A model-free safe reinforcement learning is studied by synthesizing a barrier certificate and querying a black-box dynamic function in [190]. A game theoretic approach is adopted in [158] to reduce conservatism while maintaining robustness during human robot interaction. Differentiable optimization layers have emerged as a new approach for safe learning control recently. In [123], a differentiable layer is applied to control barrier function based quadratic program in order to enhance the recursive feasibility, where the parameters are adapted online. In [51], safety is framed as a differentiable robust CBF layer in model-based RL. We also utilize the differentiable optimization layer as a tool. However, we focus on generalizing the safety-critical control to novel environments.

6.1.2 Contributions

The contribution of this paper is as follows:

- We present an approach to generalizing safety-critical control to novel environments by integrating control barrier functions and differentiable optimization.
- We introduce a neural network based ECBF-QP and formulate the safety-critical control as a differentiable optimization layer.
- We show that the proposed neural network module based on the exponential control barrier function assures the forward invariance of a safe set.
- We numerically validate the proposed learning control design using systems with different relative-degrees and novel environments with randomly generated obstacles.

6.1.3 Organization

This paper is organized as follows: in Sec. 6.2, we introduce the background of control barrier functions and differentiable optimization. The problem formulation is illustrated in Sec. 6.3, where we motivate the formulation with a simple case study. Then, in Sec. 6.4, we present the methodology of learning differentiable safety-critical control using control barrier functions. In Sec. 6.5, we test the proposed control logic on 2D double and quadruple integrator systems with different environment settings. Secs. 6.6 and 6.7 provides discussion and concluding remarks.

6.2 Background

Throughout this paper, we will consider a nonlinear control affine system:

$$\dot{x} = f(x) + g(x)u, \tag{6.1}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$ represents the state of the system, $u \in \mathbb{R}^m$ is the control input, and $f : \mathcal{X} \to \mathbb{R}^n$ and $g : \mathcal{X} \to \mathbb{R}^m$ are locally Lipschitz continuous.

6.2.1 Control Barrier Functions

Definition 2. *[80] A Lipschitz continuous function $\alpha : [0, a) \to [0, \infty), a > 0$ is said to belong to class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$. Moreover, α is said to belong to class $\mathcal{K}_\infty$ if it belongs to class $\mathcal{K}$, $a = \infty$, and $\lim_{r\to\infty} \alpha(r) = \infty$.*

Definition 3. *[8, Def. 2] Consider a continuously differentiable function $h : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ and a set $\mathcal{C}$ defined as the superlevel set of h, $\mathcal{C} = \{x \in \mathcal{X} : h(x) \geq 0\}$, then h is a control* *barrier function (CBF) if there exists an extended class $\mathcal{K}_\infty$ function α such that for the control system* (6.1)*:*

$$\sup_{u\in\mathbb{R}^m}[L_f h(x) + L_g h(x)u] \geq -\alpha(h(x)). \tag{6.2}$$

If h is a control barrier function on $\mathcal{X}$ and $\frac{\partial h}{\partial x} \neq 0$ for all $x \in \partial\mathcal{C}$, any Lipschitz continuous controller satisfying (6.2) renders the set $\mathcal{C}$ forward invariant [8, Thm.2]. By incorporating (6.2) as a constraint, a quadratic program based safety-critical controller is proposed in [6]:

CBF-QP:

$$\begin{aligned} u^*(x) = \quad & \underset{u\in\mathbb{R}^m}{\arg\min} \quad \|u - u_{\text{perf}}\|^2 && \text{(6.3a)}\\ \text{s.t.} \quad & L_f h(x) + L_g h(x)u \geq -\alpha(h(x)), && \text{(6.3b)} \end{aligned}$$

where u_{perf} is the reference control input that can be from a high-level performance controller, which is expected to achieve the control objective. For instance, model predictive control is a popular choice as a high-level performance controller [135]. In the context of safety-critical control, a control Lyapunov function is often used in a quadratic program formulation (CLF-QP) to realize stability.

Remark 1. *In Definition 3, an extended class $\mathcal{K}_\infty$ function is required for CBF. Here, we restrict ourselves to a subclass: class $\mathcal{K}$ function, which can facilitate our learning algorithm. Typically, $\alpha(x)$ is simplified as αx, with α being a positive constant, which we term as a linear class $\mathcal{K}$ function. Previous work [123, 175, 182] have investigated how to adjust the class $\mathcal{K}$ function in order to improve the feasibility. In this chapter, we focus on learning a neural network based class $\mathcal{K}$ function to safely generalize to different environments.*

The CBF constraint in (6.3b) has been so far assumed to be relative-degree one, which typically does not held for most safety-critical constraints in robotic systems [66]. A special type of CBFs called exponential control barrier functions (ECBFs) has been introduced to enforce arbitrarily high relative-degree CBF constraints in [119].

Definition 4. *[119, Def. 1] Consider a r-times continuously differentiable function $h : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ and a set $\mathcal{C}$ defined as the superlevel set of h, $\mathcal{C} = \{x \in \mathcal{X} : h(x) \geq 0\}$, then h is an exponential control barrier function (ECBF) if there exists a row vector $K_\alpha \in \mathbb{R}^r$ such that for the system* (6.1)*:*

$$\sup_{u\in\mathbb{R}^m}[L_f^r h(x) + L_g L_f^{r-1} h(x)u] \geq -K_\alpha \eta_b(x), \tag{6.4}$$

for $\forall x \in \{x \in \mathbb{R}^n | h(x) \geq 0\}$, *with*

$$\eta_b(x) = \begin{bmatrix} h(x) \\ \dot{h}(x) \\ \ddot{h}(x) \\ \vdots \\ h^{(r-1)}(x) \end{bmatrix} = \begin{bmatrix} h(x) \\ L_f h(x) \\ L_f^2 h(x) \\ \vdots \\ L_f^{r-1} h(x) \end{bmatrix}. \tag{6.5}$$

We define $\mu = L_f^r h(x) + L_g L_f^{r-1} h(x) u$, then the above dynamics of $h(x)$ can be written as the linear system

$$\begin{aligned} \dot{\eta}_b(x) &= F\eta_b(x) + G\mu, \\ h(x) &= C\eta_b(x), \end{aligned} \tag{6.6}$$

where

$$\begin{aligned} F &= \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \\ C &= \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}. \end{aligned} \tag{6.7}$$

If $\mu \geq -K_\alpha \eta_b(x)$, with $(F - GK_\alpha)$ being Hurwitz and total negative, then we can guarantee that $h(x_0) \geq 0 \implies h(x(t)) \geq 0, \forall t \geq 0$ where x_0 is the initial condition.

Let $-p_i$ be the negative real eigenvalues of $(F - GK_\alpha)$. We can then define a family of functions $v_i : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ with corresponding superlevel sets $\mathcal{C}_i$,

$$\begin{aligned} v_0(x) &= h(x), & \mathcal{C}_0 &= \{x : v_0(x) \geq 0\}, \\ v_1(x) &= \dot{v}_0 + p_1 v_0(x), & \mathcal{C}_1 &= \{x : v_1(x) \geq 0\}, \\ &\vdots & &\vdots \\ v_r(x) &= \dot{v}_{r-1} + p_r v_{r-1}(x), & \mathcal{C}_r &= \{x : v_r(x) \geq 0\}, \end{aligned} \tag{6.8}$$

where $\mathcal{C}_0$ plays the role of the safe set $\mathcal{C}$ as defined in Definition 3 for a relative-degree one CBF. Then, we have:

Theorem 1. *[119, Thm.2] A valid exponential CBF should satisfy two conditions: suppose* K_α *is chosen such that* $p_i > 0$ *and the eigenvalues* $-p_i$ *satisfy* $p_i \geq -\frac{\dot{v}_{i-1}(x_0)}{v_{i-1}(x_0)}$, *then* (6.9b) *guarantees* $h(x)$ *is an exponential control barrier function.*

Given an ECBF, we can extend the CBF-QP in (6.3) to enforce high relative-degree safety-critical constraints:

ECBF-QP:

$$u^*(x) = \underset{u \in \mathbb{R}^m}{\arg\min} \quad \|u - u_{\text{perf}}\|^2 \tag{6.9a}$$

$$\text{s.t.} \quad L_f^r h(x) + L_g L_f^{r-1} h(x) u \geq -K_\alpha \eta_b(x), \tag{6.9b}$$

where u_{perf} is the reference control input. Note that the control barrier function constraint (6.9b) can be extended to multiple constraints in order to account for different safety criteria. Furthermore, a general formulation of high order control barrier functions can be seen in [174].

Remark 2. *In ECBF-QP* (6.9b)*, due to the relation between the coeeficients in K_α and the eigenvalues $-p_i$, $K_\alpha \eta_b$ can be reformulated as $\Pi_{i=1}^r [L_f + p_i] \circ h(x) - L_f^r h(x)$. This will be used to develop our differentiable safety-critical control formulation. Note that L_f here is the Lie derivative operator s.t. $L_f \circ h(x) = L_f h(x) = \frac{\partial}{\partial x} h(x) f(x)$.*

6.2.2 Differentiable Optimization

A differentiable optimization problem is a class of optimization problems whose solutions can be backpropagated through. This functionality enables an optimization problem to serve as layers within deep learning architectures, which can encode constraints and complex dependencies through optimization that traditional convolutional and fully-connected layers usually cannot capture [9]. Some successful differentiable layer examples include differentiable model predictive control [10, 49], Pontryagin's maximum principle [75], robust control [46], and meta learning [91], etc. In this chapter, we utilize the differentiable optimization layer presented in [2] for the ECBF-QP.

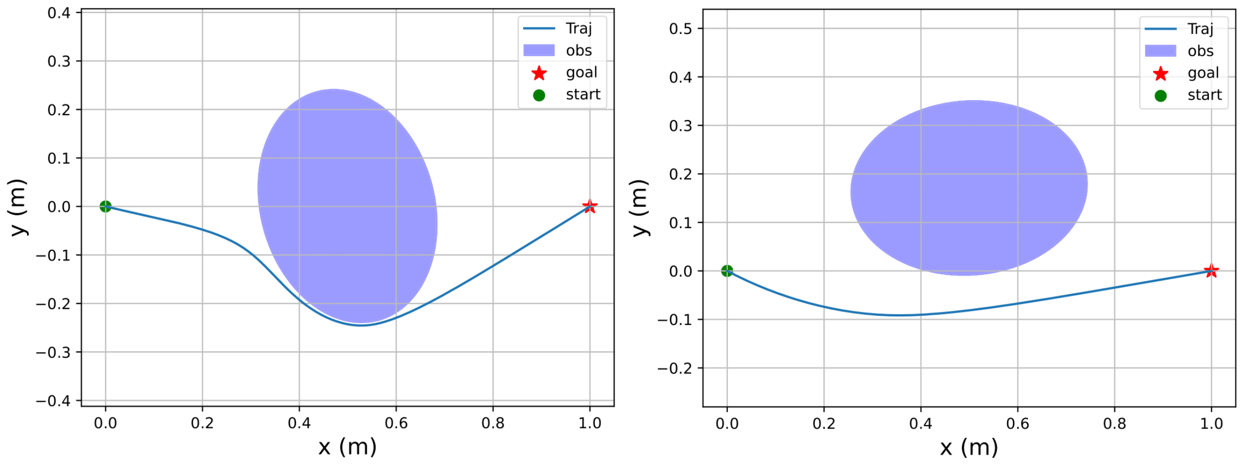
Remark 3. *While CBFs are continuously differentiable functions [8], here a differentiable safety-critical control using CBF does not mean the CBF itself is differentiable but rather that the backpropagation can go through the CBF-QP differentiable optimization layer.*

6.3 Problem Statement

Having established the background of CBFs and differentiable optimizations, we now present our problem formulation for generalizing to novel environments.

6.3.1 Motivating Example

Using a 2D double integrator as an illustrative example, we design a linear quadratic regulator (LQR) to drive the system to a goal location while avoiding different obstacles using the ECBF-QP in (6.9). The LQR controller serves as the reference performance controller



(a) Environment 1 (b) Environment 2

Figure 6.1: Motivating example for safety-critical control for generalization to novel environments using a 2D double integrator. A hand-tuned K_α for Environment 1 in (a) is used in the novel Environment 2 in (b). As can be seen, this results in a trajectory with a larger deviation from the obstacle in Environment 2. Thus, a well-tuned K_α for one environment does not necessarily generalize to a different environment.

u_{perf}. The simulation results are demonstrated in Figure 6.1. Note that the K_α for ECBF-QP is tuned manually in Figure 6.1(a), which leads to a short and smooth trajectory, i.e., a smooth trajectory that goes around the obstacle with minimal detour from a straight-line trajectory from start to goal. However, the ECBF-QP with the same K_α results in a large detour in Figure 6.1 (b) in a different environment. While larger detours are conservative, they potentially require more control effort, and result in energy inefficient motions. The motion in Environment 2 could be shorter by getting closer to the obstacle. This example demonstrates that the K_α plays an important role in generating desirable trajectories in different environments.

Remark 4. *In order to get a trajectory with desired properties, e.g., smoothness and minimum distance, it is necessary to choose a proper K_α using ECBF-QP. Moreover, certain fixed K_α that works in a particular environment could actually fail in a different environment, resulting in violation of the safety constraint $h(x) \geq 0$.*

6.3.2 Problem Formulation

Building upon the motivating example, we are motivated to optimize the class $\mathcal{K}$ function in CBF-QP or K_α in ECBF-QP with respect to different environments, which can result in a safe trajectory satisfying a user-defined metric.

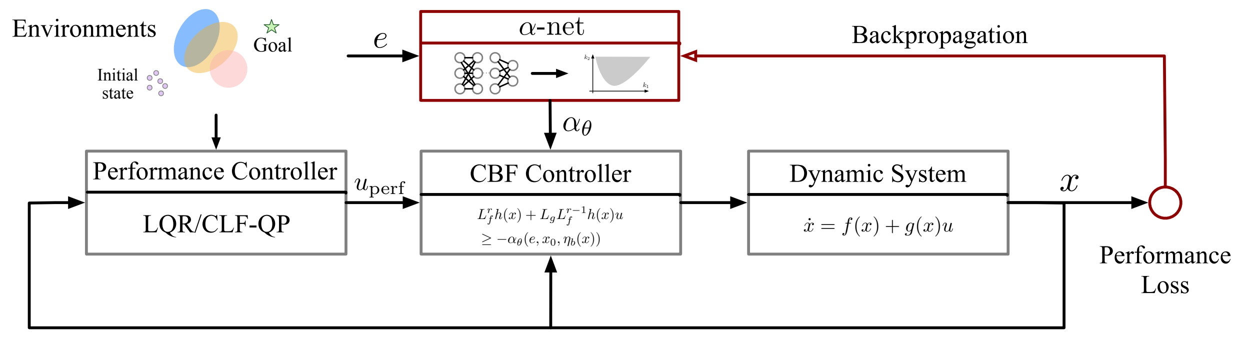


Figure 6.2: The overall framework of the proposed approach, which includes two main components: a performance controller and a differentiable CBF-QP. The novel environment information, e, is an input to the performance controller and α net. The performance loss computed along a trajectory will be backpropagated through the α net, then the α net outputs the parameters to construct the class $\mathcal{K}$ function.

To this end, we represent the $K_\alpha \eta_b(x)$ in (6.4) with a neural network parameterized with θ. $\Pi_\theta(u_{\text{perf}}, e, x_0)$ represents the solution of ECBF-QP mentioned in (6.9b). Such an ECBF-QP can be embedded as a layer in a deep learning pipeline by using differentiable convex optimization technique. Then we minimize a performance cost $\mathcal{L}$ in an episodic setting. The formulation is given as follows:

$$\begin{aligned} &\arg\min_{\theta} \quad \mathbb{E}_{x_0 \sim P_0, e \sim P_e}[\mathcal{L}(\tau, e, x_0)] \\ \text{s.t.} \quad &u = \Pi_\theta(u_{\text{perf}}(x), e, x_0), \\ &\dot{x} = f(x) + g(x)u, \end{aligned} \tag{6.10}$$

where e is an environment sampled from a distribution of environments P_e, e.g., e consists of center and radius of the obstacle. x is the state where we evaluate cost at each time step, and x_0 is the initial state which is sampled from a distribution P_0. Note that $\mathcal{L}$ is the cost along a trajectory instead of the cost at each time step, and τ represents the trajectory with time horizon T. u_{perf} is the performance control input provided by a high-level performance controller. Once the training procedure is done offline, we can deploy the neural network based controller Π_θ to novel environments.

6.4 Methodology

Having seen the problem formulation, we will next introduce how to enable the generalization to novel environments via learning differentiable ECBF-QP.

The overall control architecture is shown in Figure 6.2, which basically includes two parts: a performance controller and a differentiable ECBF-QP safety filter. The performance controller is mainly responsible for achieving the control objective, and the differentiable ECBF-QP serves as a safety filter, which will be explained in detail in this section.

6.4.1 Differentiable Safety-Critical Control using CBFs

We formulate our differentiable safety-critical control based on exponential control barrier functions in (6.9). Differentiable CBFs have been used in [123] and [51]. However, they used systems with relative-degree one or solved a relative-degree two system using a cascaded approach. Here, we extend it to a general formulation as follows:

Differentiable ECBF-QP:

$$\begin{aligned} \Pi_\theta(u_{\text{perf}}, e, x_0) = &\underset{u\in\mathbb{R}^m, \delta\in\mathbb{R}}{\arg\min}\ \|u - u_{\text{perf}}\|^2 + \zeta\delta^2 \\ \text{s.t.}\quad & L_f^r h(x) + L_g L_f^{r-1} h(x) u \geq -\alpha_\theta(e, x_0, \eta_b(x)) - \delta^2, \end{aligned} \tag{6.11}$$

where, $\Pi_\theta(u_{\text{perf}}, e, x_0)$ is the safe policy filtered by the ECBF-QP and conditioned on the high-level performance control input u_{perf} and the environment information e. $\alpha_\theta(e, x_0, \eta_b(x))$ is denoted by α-net, where θ represents parameters of the network.

As we will see next, the α function is a linear function (of η_b) that encodes an ECBF constraint within the differentiable ECBF-QP so as to deal with high relative-degree safety constraints, which are common in many robotic applications. We include a slack variable δ which guarantees that such an optimization is feasible during the training procedure, and ζ is a hyperparameter. Note that we do not use δ as in (6.11) during the test time.

6.4.2 The Structure of α-net

Exponential control barrier function provides a formal structure to guarantee safety with a vector parameter K_α. In general, it is not easy and probably time-consuming to find the best K_α directly in order to generalize to novel environments. We thus encode the structure of exponential CBF into our neural network. As noted in Remark 2, our formulation is shown as follows

$$\alpha_\theta(e, x_0, \eta_b(x)) = \prod_{i=1}^{r} [L_f + p_i(e, x_0; \theta)] \circ h(x) - L_f^r h(x), \tag{6.12}$$

where L_f is the lie derivative operator as mentioned in Remark 2. Notice that for the ECBF, the right side of (6.9b) only includes the lie derivative with respect to f. The function $\mathbf{p}(e, x_0; \theta) \in \mathbb{R}^r$ outputs $[p_1, p_2, \dots, p_r]$, and $p_i(e, x_0; \theta)$ represents p_i as defined in (6.8). We use the following neural network structure:

$$\begin{aligned}
\mathbf{p}(e, x_0; \theta) &= \mathrm{ReLU}(\prod_{k=0}^{m} \sigma(W_k l_k) - b(x_0)) + b(x_0), \\
b_i(x_0) &= \mathrm{ReLU}(-\frac{\dot{v}^{i-1}(x_0)}{v^{i-1}(x_0)} - \epsilon) + \epsilon, i = 1, \dots, r, \\
l_0 &= [e, x_0],
\end{aligned} \tag{6.13}$$

where $b_i(x_0)$ is the i-th element of $b(x_0) \in \mathbb{R}^r$, and it represents the bounds of $p_{i=1\dots r}$ in Thm.1. The parameter θ represents the weights $\{W_0, W_1, \dots, W_m\}$. l_k represents the outputs of the k-th layer of the neural network. We concatenate the environment information e and initial state x_0 together as the input l_0 and choose the ReLU function as the activation, with $\sigma(\cdot)$ being any activation function. Then, we randomly initialize the neural network parameters with positive numbers.

Theorem 2. *Given the function $\boldsymbol{p}(e, x_0; \theta)$ defined in* (6.13)*, p_i satisfies the conditions in Thm.1. Thus, $h(x)$ is an exponential CBF and $\mathcal{C}_0$ in* (6.8) *is forward invariant.*

Proof. Since $b_i(x) = \max\{-\frac{\dot{v}^{i-1}(x_0)}{v^{i-1}(x_0)}, \epsilon\}$ and $p_i(e, x_0, \theta) \geq b_i(x_0)$, we have $p_i(e, x_0, \theta) \geq \max\{-\frac{\dot{v}^{i-1}(x_0)}{v^{i-1}(x_0)}, \epsilon\}$. It follows that $p_i(e, x_0, \theta)$ satisfies i) $p_i(e, x_0, \theta) \geq -\frac{\dot{v}^{i-1}(x_0)}{v^{i-1}(x_0)}$ and ii) $p_i(e, x_0, \theta) \geq \epsilon > 0$, which are the conditions in Thm.1. From [119, Thm.1], the set $\mathcal{C}_0$ is forward invariant given $h(x)$ is a valid exponential CBF. □

6.4.3 Loss Function

In general, the loss function $\mathcal{L}(\tau, e, x_0)$ can be designed with any performance evaluation metric $\mathcal{L}_{\text{perf}}$. In our work, we propose a loss function which includes two components:

$$\mathcal{L}(\tau, e, x_0) = \mathcal{L}_{\text{perf}}(\tau, e, x_0) + \lambda_\delta \sum_{t=1}^{T} \delta_t^2, \tag{6.14}$$

where T is the number of simulation time steps with a fixed simulation interval Δt, τ is the simulated trajectory represented by $[x_0, x_1, \dots, x_T]$. The coefficient λ_δ is for slack variable penalty. Notice that we use a slack variable δ in (6.11) to make sure that the optimization program will not be interrupted by the infeasibility issue of solving the ECBF-QP. Here, δ_t represents the value of the slack variable δ at each time step. However, the gradient descent of the neural network α_θ may lead to a solution such that δ_t is large. Hence, we include the penalty of δ_t in the loss function. The ideal situation is that δ_t is zero.

6.4.4 Algorithm

The training algorithm is shown in Algorithm 4. The input is a distribution of environments P_e, and the output is the network weights θ of the α-net. For each iteration, we sample n environments and rollout trajectories τ, then the weights of the neural network are updated after each iteration.

Algorithm 4: Training algorithm

```
Input: Environment distribution $P_e$, initial state distribution $P_0$, simulation time
       interval $\Delta t$, simulation time horizon $T$.
Output: The network weights $\theta$.
1 while $t \leq$ number of iteration do
2     for $i$=1:$n$ do
3         Sample $e_i \sim P_e$, $x_{0,i} \sim P_0$
4         Collect the trajectory $\tau_i$ by using the designed controller.
5     Update $\theta_t \to \theta_{t-1} - \lambda \frac{1}{n} \nabla_\theta \sum_{e_i} \mathcal{L}_\theta(\tau_i, e_i, x_{0,i})$
```

When the task is obstacle avoidance, we can iteratively use the learned policy for m obstacles during the test time. The differentiable ECBF-QP in (6.11) becomes

$$\begin{aligned} \Pi_\theta(u_{\text{perf}}, e, x_0) = \underset{u \in \mathbb{R}^m}{\arg\min} \ & \|u - u_{\text{perf}}\|^2 \\ \text{s.t.} \quad & L_f^r h_j(x) + L_g L_f^{r-1} h_j(x) u \geq -\alpha_\theta(e_j, x_0, \eta_{b,j}(x)), \\ & j = 1, \ldots, m, \end{aligned} \tag{6.15}$$

where h_j represents the j-th exponential CBF. The environment e can have multiple obstacles and each of them e_j can be captured by an ECBF constraint.

6.5 Experiment Results

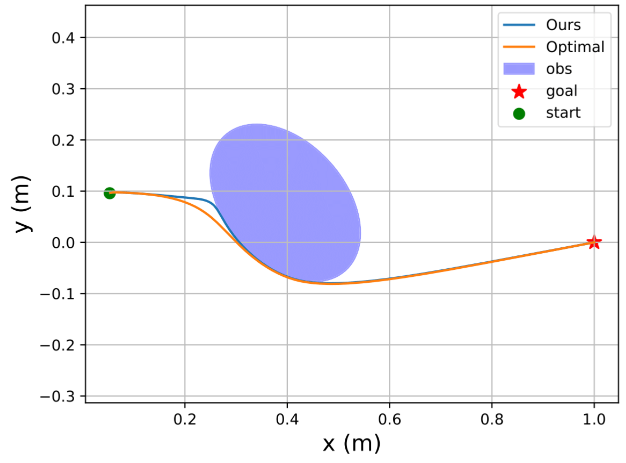
After developing our methodology for learning differentiable safety-critical control using CBFs, we now present the simulation results of our proposed framework using 2D double and quadruple integrator systems.

6.5.1 Simulation Setup

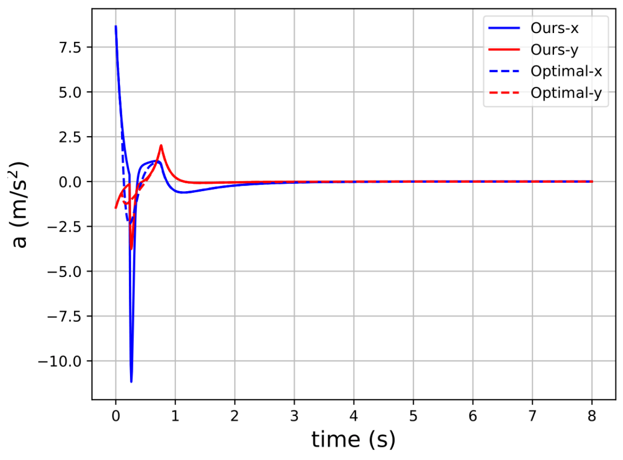
We focus on the collision avoidance problem. We set up different environments with different obstacles, which are represented by ellipses:

$$\begin{aligned} & h(y) = (y - y_c)^\top Q (y - y_c) - 1, \\ & y = Cx, Q = R(\theta)^\top \Lambda R(\theta), \end{aligned} \tag{6.16}$$

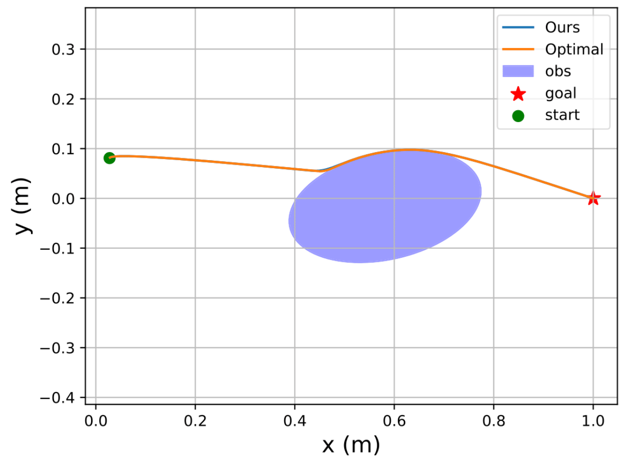
where y is the measurement variable, i.e., the position in Cartesian space. $R(\theta)$ is the rotation matrix defined by the orientation θ of the ellipse. y_c is the center of the obstacle. Λ is a diagonal matrix which represents the size of the obstacle. We define the environment



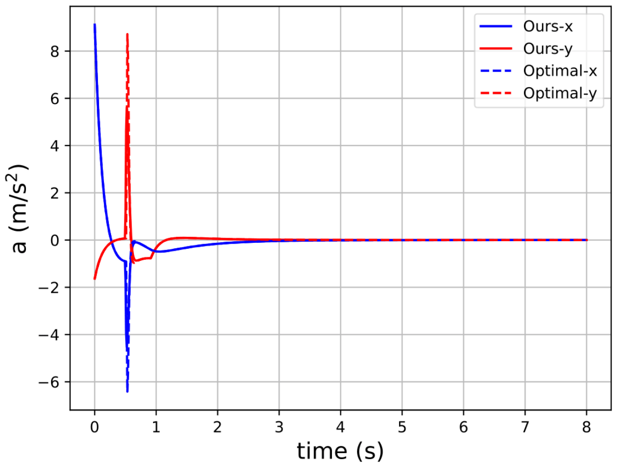
(a) Environment 1: Trajectory



(b) Environment 1: Control input



(c) Environment 2: Trajectory



(d) Environment 2: Control input

Figure 6.3: 2D double integrator (n=4, r=2) avoids a randomly generated obstacle in two different environments. The blue trajectory uses the proposed method, and the orange trajectory is the optimal performance reference that was generated by learning specifically for that environment. The corresponding control inputs are shown on the right side.

information as $e = [y_c, diag(\Lambda)^\top, \theta]^\top \in \mathbb{R}^5$. For different environments, we randomly sample e from a Gaussian distribution $\mathcal{P}_e$.

We use a linear quadratic regulator as the performance controller and a differentiable ECBF-QP as the safety filter. For the α-net in the differentiable ECBF-QP, we use a feedforward neural network with two hidden layers. Each hidden layer size is 100. Based on Algorithm 4, we train each system with 100 iterations. In each iteration, we sample 30 environments, and for each rollout, the simulation time is 8s. Futhermore, the initial condition of each system is selected randomly within a predefined region. We use the same loss function for both systems, which is the sum of the distance between each point and the goal location.

$$\mathcal{L}_{\text{perf}}(\tau, e, x_0) = \sum_{t=0}^{T} \|x_t - x_{\text{goal}}\|^2. \tag{6.17}$$

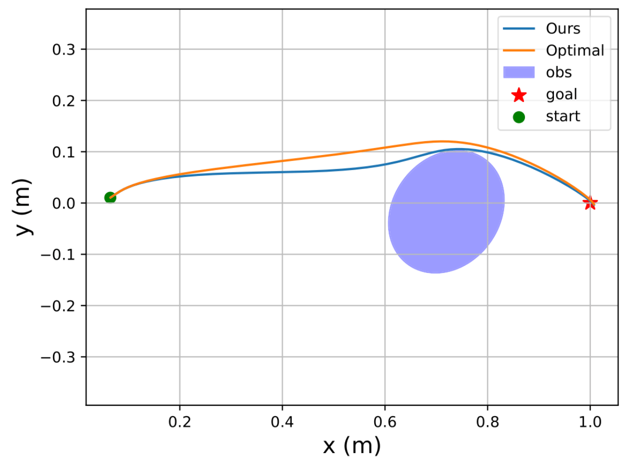
The numerical values of this loss will serve as means to compare performance of different controllers. Moreover, we train a α-net only conditioned on a specific environment to serve as the optimal solution for that specific environment.

6.5.2 Double Integrator Experiment

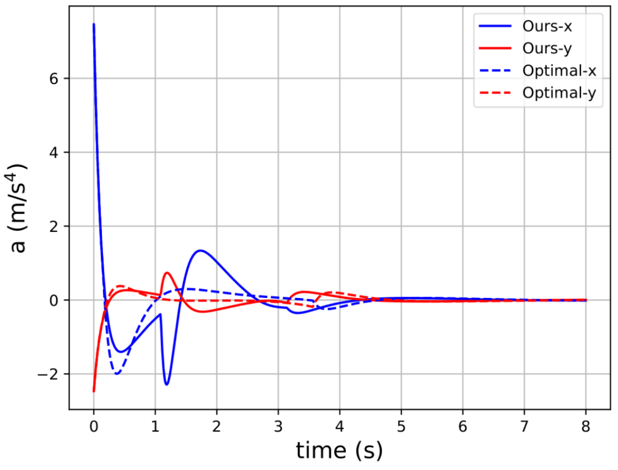
Two representative validation results for 2D double integrator avoiding an obstacle with the proposed approach are shown in Figure 6.3, including trajectory and control input. The start point is chosen randomly, the goal location is at $(1.0, 0.0)$, and the obstacle is colored as blue. Moreover, the proposed method is compared with the optimal performance solution, which is obtained by finding the best K_α based on the current environment, i.e., the environment in Figure 6.3. In Environment 1, the losses defined in (6.17) for our method and optimal performance solution are 26.88 and 26.41. In Environment 2, the losses are 25.88 and 25.86 for ours and optimal performance solution, respectively. The simulation result shows that the performance of our proposed method is close to the optimal performance solution in terms of the loss function in (6.17). Also, our control inputs (solid lines) is similar to the optimal ones (dashed lines) as shown in Figure 6.3 (b) and (d).

6.5.3 Quadruple Integrator Experiment

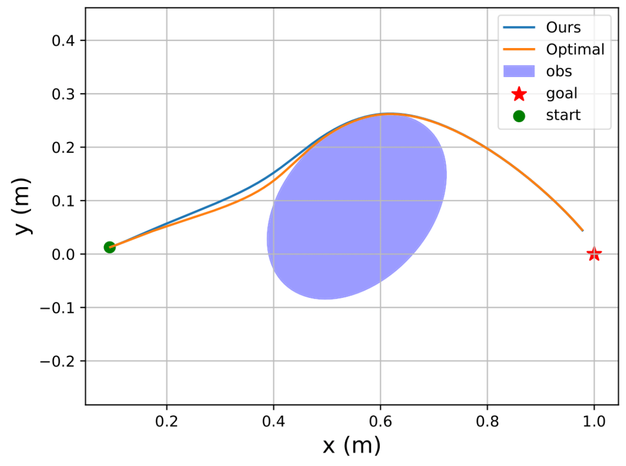
In Figure 6.4, we show that our approach can cope with a system with relative-degree four for generalization to novel environments. In both environments, the losses for the optimal performance solution is 26.71 and 35.03, whereas the losses for our method is 28.35 and 35.36, respectively. We also observe that the control inputs of the proposed method is similar to the optimal performance solution as shown in Figure 6.4 (b) and (d). The results imply that our approach can determine a proper K_α given the environment information without any manually tuning process for high relative-degree systems.



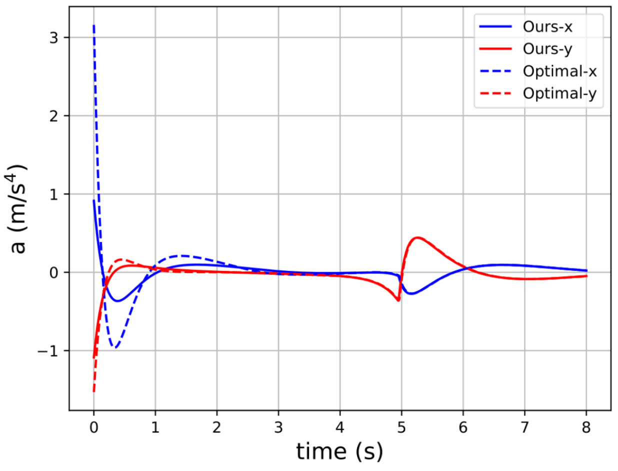
(a) Experiment 1: Trajectory



(b) Experiment 1: Control input



(c) Experiment 2: Trajectory



(d) Experiment 2: Control input

Figure 6.4: 2D quadruple integrator (n=8, r=4) avoids a randomly generated obstacle in two different environments. The blue trajectory uses the proposed method, and the orange trajectory is the optimal performance reference that was generated by learning specifically for that environment. The corresponding control inputs are shown on the right side.

6.5.4 Multiple Obstacles Experiment

To further validate the generalization ability, we extend the simulation setup of 2D double integrator from one obstacle to multiple obstacles. We randomly generate two obstacles and formulate one ECBF constraint for each object accordingly in (6.15). For each constraint, we use the same learned α-net. In this scenario, the proposed method needs to be able to generalize to more complex environments. The simulation results of two examples are

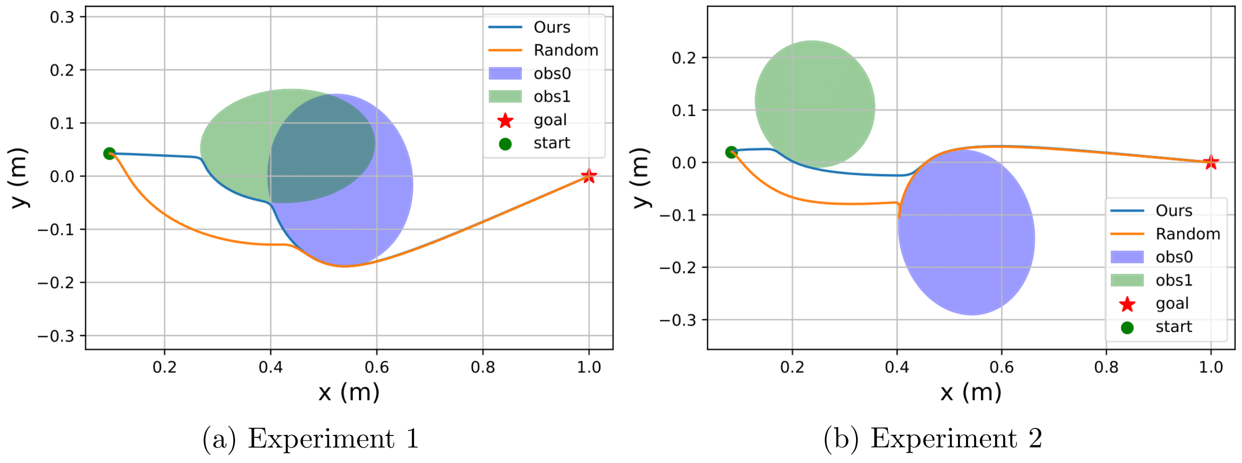


Figure 6.5: 2D double integrator is able to generalize to novel environments with two randomly generated obstacles, which are not experienced during training. The blue trajectory utilizes the proposed framework, and the orange trajectory uses a random valid K_α.

shown in Figure 6.5. In Experiment 1, the losses for our method and random valid K_α are 28.11 and 32.50, respectively, and in Experiment 2, the corresponding losses are 23.96 and 35.51. It shows that our approach successfully generalizes to multiple obstacle scenarios and outperforms the baseline by a large margin.

6.5.5 Ablation Study

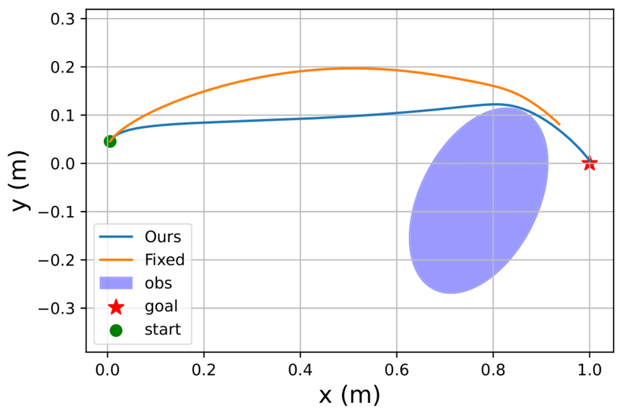
We conduct two ablation studies using the 2D quadruple integrator to validate that our proposed design is necessary for generalizing to novel environments.

Is the obstacle information indeed useful?

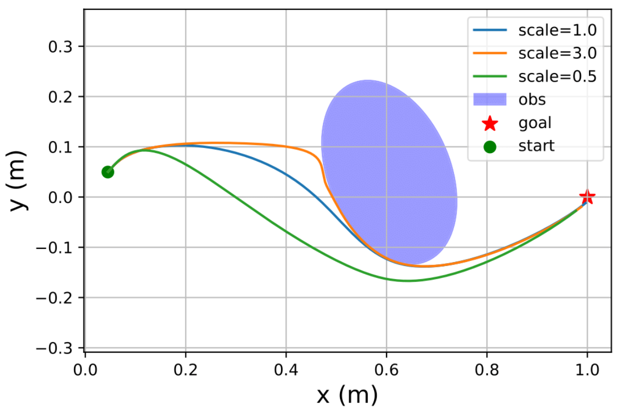
The first ablation study is to evaluate whether the obstacle information is necessary. We train an α-net with only one fixed obstacle during training as a baseline and then test it with novel environments. The resulting trajectories are shown in Figure 6.6(a). Our proposed method has a loss of 30.26, while the loss for baseline is 36.79. This indicates that the obstacle information is necessary as an input to our neural network.

Is a larger or smaller valid K_α better?

In Figure 6.6(b), we investigate whether a larger or smaller valid K_α can achieve a better performance with respect to our loss function. We scale K_α by multiplying the learned eigenvalues $\{p_i\}_{i=1,\dots,r}$ with coefficients 3.0 and 0.5. The losses for our method, 3.0 scale, and



(a) Ablation study 1



(b) Ablation study 2

Figure 6.6: Ablation study: (a) obstacle information is necessary for environment generation. The blue curve uses the proposed method, and the orange curve is the baseline with fixed obstacle information; (b) a larger or smaller valid K_α does not lead to a better performance. Different colors represent different scales.

Table 6.1: Benchmark of our proposed framework in three different scenarios: double integrator, quadruple integrator, and double integrator with two obstacles.

Scenario	Random	Ours
Double Integrator	26.8832±0.7259	**26.6774±0.5998**
Quadruple Integrator	34.2979±1.1136	**32.2187±0.8840**
Multiple Obstacles	62.2640±2.2653	**40.6258±3.6791**

0.5 scale are 33.90, 65.90, and 37.85, respectively. The resulting trajectories demonstrate that the proposed method outperforms the cases with the scales.

6.6 Discussion

We next provide the analysis of the proposed framework and a discussion on the limits and thoughts on future work.

We carry out a performance benchmark in three different scenarios: 2D double integrator with one obstacle (Double Integrator), 2D quadruple integrator with one obstacle (Quadruple Integrator), and 2D double integrator with two obstacles (Multiple Obstacles). We compare our method with random valid K_α using 200 experiments for each scenario. The mean and standard deviation of the losses are summarized in Table 6.1. We first conduct 4 subtasks, and each of them consists of 50 experiments. The mean and standard deviation are computed based on these subtasks. The benchmark for Double Integrator and Quadruple Integrator further validates that the proposed framework is useful for generalization to novel environments. Moreover, when we extend the learned α-net to multiple obstacles, our method shows better results.

In terms of the overall controller design, we assume that a high-level controller is given and fixed, which could be a valid assumption in many applications. Note that the overall performance is determined by both the high-level performance controller and the CBF-QP safety filter.

6.7 Chapter Summary

In this chapter, we presented a learning differentiable safety-critical-control framework using control barrier functions for generalization to novel environments, which uses a learning-based method to choose an environment-dependent K_α in exponential control barrier function. Moreover, based on the ECBF formulation, the proposed method ensures the forward invariance of the safe set. We numerically verified the proposed method with 2D double and quadruple integrator systems in novel environments. Our framework can be easily generalized to different shapes of obstacles and nonlinear dynamics. Also, different representation of environment information such as images and point cloud can be used. There are several interesting future directions. For instance, an integrated end-to-end framework can be designed for training the performance controller and ECBF-QP. Another promising future direction is to test our approach in more general scenarios.

Chapter 7

Final Words

In this dissertation, we proposed several approaches to predict and generate human behavior for autonomous driving and robotics within multiple problem settings. Part I presented how to improve the generalizability of human driving behavior prediction in transfer learning and continual learning scenarios. In Chapter 2, we investigated how to obtain a good representation of observation to improve the transferability of trajectory prediction. We leveraged the structured environment information and self-supervised learning technique to capture the multi-agent interaction. The proposed approach demonstrated superior performance, especially in the zero-shot and few-shot learning settings. Furthermore, Chapter 3 introduced a neural memory system to enable continual learning of human driving behavior prediction. We showed that the proposed method has less forgetting with less memory usage than other baselines. In Chapter 4, we first analyzed uncertainty estimation for trajectory prediction under the assumption that the prediction model is private. The proposed method showed that static-only information is also efficient in deciding which cases may have higher uncertainty than others. Then we utilized only the static-information-based uncertainty estimator to improve the data efficiency of transfer learning. The results validated that the proposed pipeline can be more effective than using randomly collected data to finetune predictors in unseen scenarios. Part II shifted from behavior prediction to generation. In this part, we focused on two essential properties of human behavior generation: diversity and feasibility. In Chapter 5, we developed a deep generative model to blur the boundary between prediction and generation. The proposed approach enabled testing-time adjustment between prediction and generation. Meanwhile, a teach-student system was proposed to improve the diversity of 3D human motion generation. Chapter 6 investigated how to incorporate deep learning and safety-critical control to generate feasible trajectories in different environments.

There are several promising future directions:

- **Evaluation of Human Behavior Models** Lord Kelvin said: *"If you cannot measure it, you cannot improve it."* Although there are many evaluation measurements of human behavior, such as geometric metric and probabilistic measurements, in the literature, researchers' consensus is that none of the existing measurements can perfectly reflect whether one model is good enough. For instance, distance-based metrics such as ADE or minADE can only evaluate the average performance of a prediction model; APD (Average Pairwise Distance) can only show how diverse the predicted behavior is among the generated samples. The difficulty of human behavior evaluation can be attributed to its inherent stochastic and high dimensionality nature. In particular, we usually cannot get enough data to represent the ground truth distribution of human driving trajectories conditioned on one observation, since it is almost impossible to find the same observation that includes different traffic participants, map information, etc. It brings difficulty to evaluate the multi-modality of human behavior. Moreover, even if one model could generate diverse human behavior, it is challenging to examine whether the estimated likelihood of each mode is correct without enough ground truth data. Meanwhile, considering the influence of downstream tasks for human behavior models is also a challenging and open problem. For instance, evaluating the human driving behavior prediction model and motion planning algorithm is not independent in autonomous driving scenarios. One of the cases is shown in [70], where a task-related evaluation metric is proposed. Meanwhile, a differentiable framework [76] is proposed to improve the prediction performance with the proposed task-related metric in the open-loop setting. In the future, investigating more complicated and realistic settings, such as closed-loop multi-agent settings, can remarkably enhance performance.

- **Human Behavior Foundation Models** Recently, we have witnessed the success of foundation models in different domains: conversational AI, text-to-image generation, etc. It is shown that using a large enough dataset and models could achieve superior performance and remarkable generalizability. However, we have yet to see a comparable foundation model proposed for human behavior modeling. Although there are some initial attempts, such as text-to-motion models [157, 126] for 3D human motion generation, we still need to scale the dataset size and the model to improve the generalizability. Moreover, improving the efficiency of data collection, including annotation and preprocessing, is one of the most challenging problems. It is interesting to see whether a human behavior foundation model also has zero-shot transferability. It is also vital to develop continual learning algorithms for foundation models.

With the ongoing advancement of technology, we will witness the coexistence of intelligent robots and humans and experience unprecedented levels of efficiency, safety, and high quality of life in the future.

Bibliography

[1] Tameem Adel, Han Zhao, and Richard E Turner. "Continual Learning with Adaptive Weights (CLAW)". In: *International Conference on Learning Representations*. 2020.

[2] Akshay Agrawal et al. "Differentiable Convex Optimization Layers". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 9562–9574.

[3] Emre Aksan, Manuel Kaufmann, and Otmar Hilliges. "Structured prediction helps 3d human motion modelling". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7144–7153.

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

[5] Mohammad Sadegh Aliakbarian et al. "Learning variations in human motion via mix-and-match perturbation". In: *arXiv preprint arXiv:1908.00733* (2019).

[6] Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. "Control barrier function based quadratic programs with application to adaptive cruise control". In: *53rd IEEE Conference on Decision and Control*. IEEE. 2014.

[7] Aaron D Ames et al. "Control barrier function based quadratic programs for safety critical systems". In: *IEEE Transactions on Automatic Control* 62 (2016), pp. 3861–3876.

[8] Aaron D Ames et al. "Control barrier functions: Theory and applications". In: *2019 18th European Control Conference (ECC)*. IEEE. 2019, pp. 3420–3431.

[9] Brandon Amos and J Zico Kolter. "Optnet: Differentiable optimization as a layer in neural networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 136–145.

[10] Brandon Amos et al. "Differentiable MPC for End-to-end Planning and Control". In: *Advances in Neural Information Processing Systems* 31 (2018), pp. 8289–8300.

[11] Andrea Bajcsy et al. "A robust control framework for human motion prediction". In: *IEEE Robotics and Automation Letters* 6.1 (2020), pp. 24–31.

[12] Andrea Bajcsy et al. "Analyzing human models that adapt online". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 2754–2760.

[13] Somil Bansal et al. "Hamilton-jacobi reachability: A brief overview and recent advances". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 2242–2253.

[14] Fernando S. Barbosa et al. "Provably Safe Control of Lagrangian Systems in Obstacle-Scattered Environments". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 2056–2061. DOI: 10.1109/CDC42340.2020.9304160.

[15] Emad Barsoum, John Kender, and Zicheng Liu. "Hp-gan: Probabilistic 3d human motion prediction via gan". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 1418–1427.

[16] Justin Bayer et al. "Evolving memory cell structures for sequence learning". In: *International conference on artificial neural networks*. Springer. 2009, pp. 755–764.

[17] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives". In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[18] Apratim Bhattacharyya et al. "Conditional flow variational autoencoders for structured sequence prediction". In: *arXiv preprint arXiv:1908.09008* (2019).

[19] Julian Bock et al. "The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections". In: *IEEE Intell. Vehicles Symp. (IV)*. IEEE. 2020, pp. 1929–1934.

[20] Matthew Brand and Aaron Hertzmann. "Style machines". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, pp. 183–192.

[21] Defu Cao et al. "Spectral temporal graph neural network for trajectory prediction". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 1839–1845.

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

[23] Yuning Chai et al. "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction". In: *arXiv preprint arXiv:1910.05449* (2019).

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

[25] Arslan Chaudhry et al. "Efficient lifelong learning with a-gem". In: *arXiv preprint arXiv:1812.00420* (2018).

[26] Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: *arXiv preprint arXiv:2002.05709* (2020).

[27] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[28] Kyunghyun Cho et al. “On the properties of neural machine translation: Encoder-decoder approaches”. In: *arXiv preprint arXiv:1409.1259* (2014).

[29] Chiho Choi and Behzad Dariush. “Looking to relations for future trajectory forecast”. In: *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.* 2019, pp. 921–930.

[30] Chiho Choi and Behzad Dariush. “Looking to relations for future trajectory forecast”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2019, pp. 921–930.

[31] Chiho Choi et al. “Shared cross-modal trajectory prediction for autonomous driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2021.

[32] Jason Choi et al. “Reinforcement Learning for Safety-Critical Control under Model Uncertainty, using Control Lyapunov Functions and Control Barrier Functions”. In: *Proceedings of Robotics: Science and Systems.* Corvalis, Oregon, USA, July 2020. DOI: `10.15607/RSS.2020.XVI.088`.

[33] Yinlam Chow et al. “A lyapunov-based approach to safe reinforcement learning”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems.* 2018, pp. 8103–8112.

[34] Kurtland Chua et al. “Deep reinforcement learning in a handful of trials using probabilistic dynamics models”. In: *Advances in neural information processing systems* 31 (2018).

[35] Igor Cizelj et al. “Probabilistically safe vehicle control in a hostile environment”. In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 11803–11808.

[36] Antonia Creswell et al. “Generative adversarial networks: An overview”. In: *IEEE Signal Processing Magazine* 35.1 (2018), pp. 53–65.

[37] Alexander Cui et al. “LookOut: Diverse Multi-Future Prediction and Planning for Self-Driving”. In: *arXiv preprint arXiv:2101.06547* (2021).

[38] Andreas Damianou and Neil D Lawrence. “Deep gaussian processes”. In: *Artificial intelligence and statistics.* PMLR. 2013, pp. 207–215.

[39] Charles Dawson et al. “Safe Nonlinear Control Using Robust Neural Lyapunov-Barrier Functions”. In: *5th Annual Conference on Robot Learning.* 2021.

[40] Michal Derezinski, Daniele Calandriello, and Michal Valko. “Exact sampling of determinantal point processes with sublinear time preprocessing”. In: *Advances in neural information processing systems* 32 (2019).

[41] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[42] Rahul Dey and Fathi M Salem. “Gate-variants of gated recurrent unit (GRU) neural networks”. In: *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. IEEE. 2017, pp. 1597–1600.

[43] Natalia Diaz-Rodriguez et al. “Don’t forget, there is more than forgetting: new metrics for Continual Learning”. In: *arXiv preprint arXiv:1810.13166* (2018).

[44] Frederik Diehl et al. “Graph neural networks for modelling traffic participant interaction”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 695–701.

[45] Chiyu Dong, John M Dolan, and Bakhtiar Litkouhi. “Intention estimation for ramp merging control in autonomous driving”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 1584–1589.

[46] Priya L Donti et al. “Enforcing robust control guarantees within neural network policies”. In: *International Conference on Learning Representations*. 2020.

[47] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[48] Alexey Dosovitskiy et al. “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[49] Ján Drgona, Aaron Tuor, and Draguna Vrabie. “Learning constrained adaptive differentiable predictive control policies with guarantees”. In: *arXiv preprint arXiv:2004.11184* (2020).

[50] Conor Durkan et al. “Neural spline flows”. In: *Advances in neural information processing systems* 32 (2019).

[51] Yousef Emam et al. “Safe Model-Based Reinforcement Learning Using Robust Control Barrier Functions”. In: *arXiv preprint arXiv:2110.05415* (2021).

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

[53] Katerina Fragkiadaki et al. “Recurrent network models for human dynamics”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4346–4354.

[54] Jiyang Gao et al. “VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11525–11533.

[55] Jakob Gawlikowski et al. “A survey of uncertainty in deep neural networks”. In: *arXiv preprint arXiv:2107.03342* (2021).

[56] Jennifer A Gillenwater et al. "Expectation-maximization for learning determinantal point processes". In: *Advances in Neural Information Processing Systems* 27 (2014).

[57] Boqing Gong et al. "Diverse sequential subset selection for supervised video summarization". In: *Advances in neural information processing systems* 27 (2014).

[58] 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*. 2021, pp. 15303–15312.

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

[60] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[61] Kaiming He et al. "Momentum contrast for unsupervised visual representation learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9729–9738.

[62] Olivier Henaff. "Data-efficient image recognition with contrastive predictive coding". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 4182–4192.

[63] Joey Hong, Benjamin Sapp, and James Philbin. "Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8454–8462.

[64] Yedid Hoshen. "Vain: Attentional multi-agent predictive modeling". In: *Advances in Neural Information Processing Systems*. 2017.

[65] J Ben Hough et al. "Determinantal processes and independence". In: *Probability surveys* 3 (2006), pp. 206–229.

[66] Shao-Chen Hsu, Xiangru Xu, and Aaron D Ames. "Control barrier function based quadratic programs with application to bipedal robotic walking". In: *2015 American Control Conference (ACC)*. IEEE. 2015, pp. 4542–4548.

[67] Steven C Y Hung et al. "Compacting, Picking and Growing for Unforgetting Continual Learning". In: *NeurIPS*. 2019, pp. 13669–13679.

[68] Catalin Ionescu et al. "Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments". In: *IEEE transactions on pattern analysis and machine intelligence* 36.7 (2013), pp. 1325–1339.

[69] David Isele and Akansel Cosgun. "Selective experience replay for lifelong learning". In: *Proc. AAAI*. Vol. 32. 1. 2018.

[70] Boris Ivanovic and Marco Pavone. "Injecting planning-awareness into prediction and detection evaluation". In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2022, pp. 821–828.

[71] Ashesh Jain et al. "Structural-rnn: Deep learning on spatio-temporal graphs". In: *Proceedings of the ieee conference on computer vision and pattern recognition.* 2016, pp. 5308–5317.

[72] Moksh Jain et al. "Deup: Direct epistemic uncertainty prediction". In: *arXiv preprint arXiv:2102.08501* (2021).

[73] Siddhartha Jain et al. "Maximizing overall diversity for improved uncertainty estimates in deep ensembles". In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol. 34. 04. 2020, pp. 4264–4271.

[74] Ashish Jaiswal et al. "A survey on contrastive self-supervised learning". In: *Technologies* 9.1 (2021), p. 2.

[75] Wanxin Jin, Shaoshuai Mou, and George Pappas. "Safe pontryagin differentiable programming". In: *Advances in Neural Information Processing Systems* 34 (2021).

[76] Peter Karkus et al. "DiffStack: A Differentiable and Modular Control Stack for Autonomous Vehicles". In: *Conference on Robot Learning.* PMLR. 2023, pp. 2170–2180.

[77] Dietmar Kasper et al. "Object-oriented Bayesian networks for detection of lane change maneuvers". In: *IEEE Intelligent Transportation Systems Magazine* 4.3 (2012), pp. 19–31.

[78] Alex Kendall and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?" In: *Advances in neural information processing systems* 30 (2017).

[79] Eric C Kerrigan and Jan M Maciejowski. "Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control". In: *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187).* Vol. 5. IEEE. 2000, pp. 4951–4956.

[80] Hassan K Khalil. "Nonlinear systems". In: (2002).

[81] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *International Conference on Learning Representations* (2013).

[82] Durk P Kingma et al. "Semi-supervised learning with deep generative models". In: *Advances in neural information processing systems* 27 (2014).

[83] Thomas Kipf et al. "Neural relational inference for interacting systems". In: *arXiv preprint arXiv:1802.04687* (2018).

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

[85] Robert Krajewski et al. "The rounD Dataset: A Drone Dataset of Road User Trajectories at Roundabouts in Germany". In: *IEEE Intell. Transp. Syst. Conf. (ITSC).* 2020, pp. 1–6.

[86] Alex Kulesza and Ben Taskar. “k-DPPs: Fixed-size determinantal point processes”. In: *ICML*. 2011.

[87] Jogendra Nath Kundu, Maharshi Gor, and R Venkatesh Babu. “Bihmp-gan: Bidirectional 3d human motion prediction gan”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 2019, pp. 8553–8560.

[88] Thanard Kurutach et al. “Model-ensemble trust-region policy optimization”. In: *arXiv preprint arXiv:1802.10592* (2018).

[89] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles”. In: *Advances in neural information processing systems* 30 (2017).

[90] Balaji Lakshminarayanan et al. “Simple and principled uncertainty estimation with deterministic deep learning via distance awareness”. In: (2020).

[91] Kwonjoon Lee et al. “Meta-learning with differentiable convex optimization”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10657–10665.

[92] Namhoon Lee et al. “Desire: Distant future prediction in dynamic scenes with interacting agents”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 336–345.

[93] Timothée Lesort et al. “Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges”. In: *Information fusion* 58 (2020), pp. 52–68.

[94] Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. “Conditional generative neural system for probabilistic trajectory prediction”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 6150–6156.

[95] Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. “Interaction-aware multi-agent tracking and probabilistic behavior prediction via adversarial learning”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 6658–6664.

[96] Jiachen Li et al. “EvolveGraph: Multi-Agent Trajectory Prediction with Dynamic Relational Reasoning”. In: *Advances in neural information processing systems*. 2020.

[97] Jiachen Li et al. “RAIN: Reinforced Hybrid Attention Inference Network for Motion Forecasting”. In: *Proc. IEEE Int. Conf. Comput. Vision*. 2021.

[98] Jiachen Li et al. “Spatio-Temporal Graph Dual-Attention Network for Multi-Agent Prediction and Tracking”. In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[99] Maosen Li et al. “Dynamic multiscale graph neural networks for 3d skeleton based human motion prediction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 214–223.

[100] Xin Li, Xiaowen Ying, and Mooi Choo Chuah. “Grip: Graph-based interaction-aware trajectory prediction”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 3960–3966.

[101] Xiu Li et al. “Structure from recurrent motion: From rigidity to recurrency”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3032–3040.

[102] Xiao Lin and Mohamed R Amer. “Human motion modeling using dvgans”. In: *arXiv preprint arXiv:1804.10652* (2018).

[103] David Lopez-Paz and Marc’Aurelio Ranzato. “Gradient Episodic Memory for Continual Learning”. In: *NIPS*. 2017.

[104] Ilya Loshchilov and Frank Hutter. “Decoupled weight decay regularization”. In: *arXiv preprint arXiv:1711.05101* (2017).

[105] Hengbo Ma et al. “Continual multi-agent interaction behavior prediction with conditional generative memory”. In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 8410–8417.

[106] Hengbo Ma et al. “Multi-agent Driving Behavior Prediction across Different Scenarios with Self-supervised Domain Knowledge”. In: *IEEE Intell. Transp. Syst. Conf. (ITSC)*. IEEE. 2021.

[107] Hengbo Ma et al. “Wasserstein generative learning with kinematic constraints for probabilistic interactive driving behavior prediction”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 2477–2483.

[108] Xiaobai Ma et al. “Reinforcement Learning for Autonomous Driving with Latent State Inference and Spatial-Temporal Relationships”. In: *2021 International Conference on Robotics and Automation (ICRA)*. IEEE. 2021.

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

[110] Anirudha Majumdar, Alec Farid, and Anoopkumar Sonar. “PAC-Bayes control: learning policies that provably generalize to novel environments”. In: *The International Journal of Robotics Research* 40.2-3 (2021), pp. 574–593.

[111] Andrey Malinin et al. “Shifts: A dataset of real distributional shift across multiple large-scale tasks”. In: *arXiv preprint arXiv:2107.07455* (2021).

[112] Srikanth Malla et al. “Nemo: Future object localization using noisy ego priors”. In: *arXiv preprint arXiv:1909.08150* (2019).

[113] Wei Mao, Miaomiao Liu, and Mathieu Salzmann. “Generating Smooth Pose Sequences for Diverse Human Motion Prediction”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13309–13318.

[114] Wei Mao et al. “Learning trajectory dependencies for human motion prediction”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9489–9497.

[115] Julieta Martinez, Michael J Black, and Javier Romero. “On human motion prediction using recurrent neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2891–2900.

[116] Rowan McAllister et al. “Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning”. In: International Joint Conferences on Artificial Intelligence, Inc. 2017.

[117] Rowan McAllister et al. “Control-Aware Prediction Objectives for Autonomous Driving”. In: *arXiv preprint arXiv:2204.13319* (2022).

[118] Jiquan Ngiam et al. “Scene Transformer: A unified architecture for predicting multiple agent trajectories”. In: *arXiv preprint arXiv:2106.08417* (2021).

[119] Quan Nguyen and Koushil Sreenath. “Exponential control barrier functions for enforcing high relative-degree safety-critical constraints”. In: *2016 American Control Conference (ACC)*. IEEE. 2016, pp. 322–328.

[120] Quan Nguyen and Koushil Sreenath. “Optimal Robust Control for Bipedal Robots through Control Lyapunov Function based Quadratic Programs.” In: *Robotics: Science and Systems*. Rome, Italy. 2015, pp. 1–9.

[121] Johannes von Oswald et al. “Continual learning with hypernetworks”. In: *International Conference on Learning Representations*. Sept. 2020.

[122] Soohwan Park et al. “Learning predict-and-simulate policies from unorganized human motion data”. In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–11.

[123] Hardik Parwana and Dimitra Panagou. “Recursive Feasibility Guided Optimal Parameter Adaptation of Differential Convex Optimization Policies for Safety-Critical Systems”. In: *arXiv preprint arXiv:2109.10949* (2021).

[124] Georgios Pavlakos et al. “Expressive Body Capture: 3D Hands, Face, and Body from a Single Image”. In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 10975–10985.

[125] Dario Pavllo, David Grangier, and Michael Auli. “Quaternet: A quaternion-based recurrent model for human motion”. In: *arXiv preprint arXiv:1805.06485* (2018).

[126] Mathis Petrovich, Michael J Black, and Gul Varol. “TEMOS: Generating diverse human motions from textual descriptions”. In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*. Springer. 2022, pp. 480–497.

[127] Jary Pomponi et al. “Efficient continual learning in neural networks with embedding regularization”. en. In: *Neurocomputing* (Feb. 2020).

[128] Aleksey Postnikov, Aleksander Gamayunov, and Gonzalo Ferrer. “Transformer based trajectory prediction”. In: *arXiv preprint arXiv:2112.04350* (2021).

[129] Alexey Pustynnikov and Dmitry Eremeev. “Estimating Uncertainty For Vehicle Motion Prediction on Yandex Shifts Dataset”. In: *arXiv preprint arXiv:2112.08355* (2021).

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

[131] Ievgen Redko et al. “A survey on domain adaptation theory: learning bounds and theoretical guarantees”. In: *arXiv preprint arXiv:2004.11829* (2020).

[132] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. “R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 772–788.

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

[134] Alexander Robey et al. “Learning control barrier functions from expert demonstrations”. In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 3717–3724.

[135] Ugo Rosolia and Aaron D Ames. “Multi-rate control design leveraging control barrier functions and model predictive control policies”. In: *IEEE Control Systems Letters* 5.3 (2020), pp. 1007–1012.

[136] Mohammad Rostami, Soheil Kolouri, and Praveen K Pilly. “Complementary Learning for Overcoming Catastrophic Forgetting Using Experience Replay”. In: *arXiv* (2019). URL: `http://arxiv.org/abs/1903.04566`.

[137] Amir Sadeghian et al. “Sophie: An attentive gan for predicting paths compliant to social and physical constraints”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[138] Tim Salzmann et al. “Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data”. In: *arXiv preprint arXiv:2001.03093* (2020).

[139] Franco Scarselli et al. “The graph neural network model”. In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80.

[140] Wegner DM Schacter DL Gilbert DT. “Semantic and Episodic memory”. In: *Psychology*. 2009.

[141] Edvards Scukins and Petter Ögren. “Using Reinforcement Learning to Create Control Barrier Functions for Explicit Risk Mitigation in Adversarial Environments”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Robotics and Automation Society. 2021.

[142] Jan Sedmidubsky and Pavel Zezula. “Similarity search in 3D human motion data”. In: *Proceedings of the 2019 on International Conference on Multimedia Retrieval*. 2019, pp. 5–6.

[143] Hanul Shin et al. “Continual Learning with Deep Generative Replay”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I Guyon et al. Curran Associates, Inc., 2017, pp. 2990–2999.

[144] Wenwen Si, Tianhao Wei, and Changliu Liu. “Agen: Adaptable generative prediction networks for autonomous driving”. In: *IEEE Intell. Vehicles Symp. (IV)*. IEEE. 2019, pp. 281–286.

[145] Hedvig Sidenbladh, Michael J Black, and Leonid Sigal. “Implicit probabilistic models of human motion for synthesis and tracking”. In: *European conference on computer vision*. Springer. 2002, pp. 784–800.

[146] Leonid Sigal, Alexandru O Balan, and Michael J Black. “Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion”. In: *International journal of computer vision* 87.1 (2010), pp. 4–27.

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

[148] Lin Song et al. “Generalization of Safe Optimal Control Actions on Networked Multi-Agent Systems”. In: *arXiv preprint arXiv:2109.09909* (2021).

[149] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. “Curl: Contrastive unsupervised representations for reinforcement learning”. In: *arXiv preprint arXiv:2004.04136* (2020).

[150] Jong-Chyi Su, Subhransu Maji, and Bharath Hariharan. “When does self-supervision improve few-shot learning?” In: *European Conference on Computer Vision*. Springer. 2020, pp. 645–666.

[151] Chen Sun et al. “Stochastic prediction of multi-agent interactions from partial observations”. In: *arXiv preprint arXiv:1902.09641* (2019).

[152] Lingfeng Sun et al. “Interactive Prediction for Multiple, Heterogeneous Traffic Participants with Multi-Agent Hybrid Dynamic Bayesian Network”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 1025–1031.

[153] Pei Sun et al. “Scalability in perception for autonomous driving: Waymo open dataset”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2446–2454.

[154] Bohan Tang et al. “Collaborative uncertainty in multi-agent trajectory forecasting”. In: *Advances in Neural Information Processing Systems* 34 (2021).

[155] Andrew Taylor et al. “Learning for safety-critical control with control barrier functions”. In: *Learning for Dynamics and Control*. PMLR. 2020, pp. 708–717.

[156] Graham W Taylor, Geoffrey E Hinton, and Sam Roweis. “Modeling human motion using binary latent variables”. In: *Advances in neural information processing systems* 19 (2006).

[157] Guy Tevet et al. “Human motion diffusion model”. In: *arXiv preprint arXiv:2209.14916* (2022).

[158] Ran Tian et al. “Safety Assurances for Human-Robot Interaction via Confidence-aware Game-theoretic Human Models”. In: *arXiv preprint arXiv:2109.14700* (2021).

[159] 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.

[160] Jakub Tomczak and Max Welling. “VAE with a VampPrior”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 1214–1223.

[161] 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.

[162] Ashish Vaswani et al. “Attention is all you need”. In: *arXiv preprint arXiv:1706.03762* (2017).

[163] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[164] Petar Veličković et al. “Graph Attention Networks”. In: *International Conference on Learning Representations* (2018).

[165] 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.

[166] Jacob Walker et al. “The pose knows: Video forecasting by generating pose futures”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 3332–3341.

[167] Borui Wang et al. “Imitation learning for human pose prediction”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7124–7133.

[168] Jack M Wang, David J Fleet, and Aaron Hertzmann. “Gaussian process dynamical models for human motion”. In: *IEEE transactions on pattern analysis and machine intelligence* 30.2 (2007), pp. 283–298.

[169] Li Wang, Aaron D Ames, and Magnus Egerstedt. “Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 2659–2664.

[170] Wenshuo Wang, Junqiang Xi, and Ding Zhao. “Learning and inferring a driver’s braking action in car-following scenarios”. In: *IEEE Transactions on Vehicular Technology* 67.5 (2018), pp. 3887–3899.

[171] Moritz Werling et al. "Optimal trajectory generation for dynamic street scenarios in a frenet frame". In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 987–993.

[172] Wikipedia. *Human behavior — Wikipedia, The Free Encyclopedia*. `http://en.wikipedia.org/w/index.php?title=Human%20behavior&oldid=1143286354`. [Online; accessed 04-April-2023]. 2023.

[173] Benjamin Wilson et al. "Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting". In: (2021).

[174] Wei Xiao and Calin Belta. "High Order Control Barrier Functions". In: *IEEE Transactions on Automatic Control* (2021).

[175] Wei Xiao, Calin A. Belta, and Christos G. Cassandras. "Feasibility-Guided Learning for Constrained Optimal Control Problems". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 1896–1901. DOI: `10.1109/CDC42340.2020.9303857`.

[176] Jingwei Xu et al. "Video prediction via example guidance". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 10628–10637.

[177] Xinchen Yan et al. "Mt-vae: Learning motion transformations to generate multimodal human dynamics". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 265–281.

[178] Ye Yuan and Kris Kitani. "Diverse trajectory forecasting with determinantal point processes". In: *International Conference on Learning Representations* (2019).

[179] Ye Yuan and Kris Kitani. "Dlow: Diversifying latent flows for diverse human motion prediction". In: *European Conference on Computer Vision*. Springer. 2020, pp. 346–364.

[180] Andrei Zanfir et al. "Weakly supervised 3d human pose and shape reconstruction with normalizing flows". In: *European Conference on Computer Vision*. Springer. 2020, pp. 465–481.

[181] Jun Zeng, Bike Zhang, and Koushil Sreenath. "Safety-critical model predictive control with discrete-time control barrier function". In: *2021 American Control Conference (ACC)*. IEEE. 2021, pp. 3882–3889.

[182] Jun Zeng et al. "Safety-Critical Control using Optimal-decay Control Barrier Function with Guaranteed Point-wise Feasibility". In: *2021 American Control Conference (ACC)*. 2021, pp. 3856–3863. DOI: `10.23919/ACC50511.2021.9482626`.

[183] 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]* (2019).

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

[185] Wei Zhan et al. “Spatially-partitioned environmental representation and planning architecture for on-road autonomous driving”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 632–639.

[186] Amy Zhang et al. “Learning invariant representations for reinforcement learning without reconstruction”. In: *arXiv preprint arXiv:2006.10742* (2020).

[187] Jesse Zhang et al. “Cautious adaptation for reinforcement learning in safety-critical settings”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 11055–11065.

[188] Yan Zhang, Michael J Black, and Siyu Tang. “We are more than our joints: Predicting how 3d bodies move”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 3372–3382.

[189] Hang Zhao et al. “Tnt: Target-driven trajectory prediction”. In: *Conference on Robot Learning*. PMLR. 2021, pp. 895–904.

[190] Weiye Zhao, Tairan He, and Changliu Liu. “Model-free Safe Control for Zero-Violation Reinforcement Learning”. In: *5th Annual Conference on Robot Learning*. 2021.

[191] Zachary Ziegler and Alexander Rush. “Latent normalizing flows for discrete sequences”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7673–7682.