Title

Relational Reasoning for Multi-Agent Systems

Author

Li, Jiachen

Publication Date

2021

Peer reviewed|Thesis/dissertation

Relational Reasoning for Multi-Agent Systems

by

Jiachen Li

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Professor Anca Dragan
Professor Mark Mueller

Fall 2021

Relational Reasoning for Multi-Agent Systems

Abstract

Relational Reasoning for Multi-Agent Systems

by

Jiachen Li

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

Multi-agent interacting systems are prevalent in the world, from purely physical systems to complicated social dynamic systems. The interactions between entities or components can give rise to very complex behavior patterns at the level of both individual and the whole multi-agent system. The involved entities in these systems need to interact with each other and their behaviors tend to have mutual influence. They need to perceive and behave compliant to the physical environment as well. This brings the necessity of effective reasoning on the agent-agent and agent-context relations/interactions. This especially plays a significant role in safety-critical applications such as autonomous driving and social robot navigation.

Technically, it is challenging to model the dynamics of multi-agent interacting systems due to the internal heterogeneity of agents, uncertainty and multi-modality in the future behavior, evolution or change of the context, etc. The fundamental research question to address in this dissertation is *how to model the multi-agent relations and interactions in a unified, generalizable framework with an effective relational representation.*

The focus of this dissertation is 1) to design a multi-agent behavior modeling framework with relational reasoning in dynamically evolving uncertain environments for heterogeneous agents; 2) to design a generic importance estimation framework with relational reasoning for scene understanding.

This dissertation is divided into two parts. Part I focuses on the multi-agent prediction and tracking problems. In Chapter 2, a hierarchical time-series prediction model is introduced for situation and behavior recognition based on probabilistic graphical models, which can be applied to the scenarios with a single autonomous agent or under a fixed multi-agent setting. In Chapter 3, deep generative modeling techniques are employed to learn the data distribution, which can generate more diverse and realistic prediction hypotheses. In Chapter 4, a graph representation is further leveraged to capture spatio-temporal interaction patterns, which is a natural way to represent multiple agents in the scene and their relations.

Different from the method discussed in Chapter 4 where the graph topology is determined by distance-based heuristics, in Chapter 5 we propose to learn a latent relational graph structure from observation data, which can evolve over time to enable dynamic relational reasoning. An accurate prediction model plays a significant role in multi-target tracking frameworks, especially in highly dynamic and interactive scenarios. In Chapter 6, a unified tracking and prediction framework based on a modified sequential Monte Carlo method is discussed, which can adopt any of the above prediction models as the implicit proposal distribution. Part II addresses another related downstream problem (i.e., importance estimation) of relational reasoning under a multi-agent setting. In Chapter 7, a hybrid attention inference network is presented to recognize relative importance of objects in the scene based on observation data, which enables dynamic key information selection. In Chapter 8, we further investigate how to inject human knowledge by proving human annotations with a self-supervised learning pipeline, which enables the model to learn from unlimited, unlabeled data.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

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

First and foremost, my deep and sincere gratitude goes to my Ph.D. advisor Professor Masayoshi Tomizuka, who has been a great mentor to me during the past five years for his profound knowledge, insightful perspectives, and enthusiasm on work and research, and positiveness towards life. Without his guidance and persistent help, this dissertation would not have been possible. I sincerely wish I could be a great person, an extraordinary mentor as he is in the future. What I have learned from him will accompany and influence my entire academic life.

I am very grateful to Professor Anca Dragan and Professor Mark Mueller for serving in my dissertation committee as well as in my qualifying exam committee. Meanwhile, I am thankful to Professor Roberto Horowitz and Professor Kameshwar Poolla for serving in my qualifying exam committee. They provided me with great support for my major milestones in the Ph.D. study.

Being a member in the Mechanical System Control (MSC) laboratory has been wonderful in the past five years. I sincerely thank my major collaborators at MSC Lab for all the inspiring discussions: Dr. Wei Zhan, Dr. Liting Sun, Dr. Yeping Hu, Dr. Zhuo Xu, Hengbo Ma and Jinning Li. I also would like to express my thanks to the other former and current MSC lab members: Dr. Minghui Zheng, Dr. Shiying Zhou, Dr. Kevin Haninger, Dr. Changliu Liu, Dr. Yu Zhao, Dr. Xiaowen Yu, Dr. Te Tang, Dr. Hsien- Chung Lin, Dr. Chen-Yu Chan, Dr. Cheng Peng, Dr. Yongxiang Fan, Dr. Yu-Chu Huang, Dr. Daisuke Kaneishi, Dr. Shuyang Li, Dr. Kiwoo Shin, Dr. Jianyu Chen, Dr. Zining Wang, Chen Tang, Zhuo Xu, Yujiao Cheng, Shiyu Jin, Jessica Leu, Changhao Wang, Xinghao Zhu, Yiyang Zhou, Lingfeng Sun, Ge Zhang, Huidong Gao, Zheng Wu, Jinning Li, Catherine Faulkner, Wu-Te Yang, Xiang Zhang, Ting Xu, Chengfeng Xu, Chenran Li, Akio Kodaira, and Ran Tian, for their help during my graduate study and their efforts to maintain a good research atmosphere in the group. Meanwhile, I would like to thank the visiting scholars and students in the MSC lab: Shanshan Xie, Fan Yang, Zhihao Zhang and Yaofeng Sun. 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. The valuable discussions with them enhanced this dissertation from the industrial perspective. In particular, I would like to thank Dr. Chiho Choi, Dr. David Isele, Haiming Gang, Srikanth Malla, Dr. Xiaobai Ma, Dr. Behzad Dariush and Dr. Kikuo Fujimura at Honda Research Institute US, Blake Wulfe, Dr. Jin Ge and Dr. Adrien Gaidon at Toyota Research Institute, Dr. Xinwei Shi, Tian Lan, Dr. Jonathan Stroud, Dr. Zhishuai Zhang, Dr. Junhua Mao, Dr. John Kang, Dr. Khaled Refaat and Dr. Dragomir Anguelov at Waymo LLC, for their helpful instructions and collaborations during my internships.

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

Chapter 1

Introduction

1.1 Multi-Agent Systems: An Overview

Multi-agent interacting systems are prevalent in the world, from purely physical systems to complicated social dynamic systems. The interactions between entities or components can give rise to very complex behavior patterns at the level of both individual and the whole multi-agent system. The involved entities in these systems need to interact with each other and their behaviors tend to have mutual influence. They need to perceive and behave compliant to the physical environment as well. This brings the necessity of reasoning on the agent-agent and agent-context relations/interactions. More specifically, the model is expected to reason on the interaction patterns between entities, or the relative importance of a certain entity with respect to another one. If the model can recognize the underlying relational structures, it will provide an informative intermediate representation for downstream problems (e.g., prediction, tracking, decision making, motion planning).

In this dissertation, we cover three typical instances of multi-agent systems: 1) physical systems where multiple entities (e.g., particles) interact with each other and move under the physical law, which can be very large-scale with complex dynamics; 2) intelligent transportation systems where different types of traffic participants need to share the same space and coordinate with each other when they have conflicts; 3) human teams/crowds where people need to interact with each other and behave compliant to the social norms. These systems share a similar property: the involved entities tend to have mutual influence on each other's behavior, which may result in highly complicated system dynamics or behavior patterns. Therefore, recognizing the relations and modeling the interactions between interactive agents in a multi-agent system play a significant role in various downstream tasks (e.g., tracking, prediction, importance estimation).

Technically, it is challenging to model the dynamics of multi-agent interacting systems. The fundamental research question to address in this dissertation is *how to model the multi-agent relations and interactions in a unified, generalizable framework with an effective relational representation*. Due to the broadness of application domains and complexity of multi-agent

interaction, the following aspects need to be considered when addressing the fundamental question.

- Internal heterogeneity of agents

  There may exist internal heterogeneity in a group of autonomous agents, which leads to distinct behavior and interaction patterns. Therefore, adopting a homogeneous behavior model for all the entities in the system may not be sufficient. For example, in the domain of intelligent transportation systems, the vehicle behavior is strictly constrained by road geometry and its kinematic model, while the pedestrian behavior is much more flexible. Even for the same type of traffic participants, the agents may also have different behavior styles/traits. Therefore, it is necessary to consider the internal heterogeneity of agents in behavior model design.

- Flexibility and scalability

  There may exist different number of interactive agents in a multi-agent system, which requires the relational reasoning models or behavior models to be flexible with agent number fluctuations or agent appearance/disappearance. Moreover, the model needs to be scalable to be applied to large-scale multi-agent systems (e.g., complex particle systems, human crowds). Therefore, it is necessary to find out a unified representation/formulation for multi-agent systems with flexible, large numbers of agents.

- Uncertainty and multi-modality

  There may exist uncertainty or multi-modality in multi-agent interactive behaviors. In the domain of intelligent transportation systems, a human driver may have different intentions in mind and choose a certain exit to leave an intersection or roundabout, which leads to multiple potential modes in the driving behavior. Even with the same intention, the velocity/acceleration could be uncertain in a reasonable range. In the domain of team-based sports games (e.g., basketball games), a player may either pass the ball to other team members or shoot at the basket, which brings a large uncertainty for behavior modeling. Therefore, it is necessary to take into account all the potential modes and model the uncertainties.

- Generalizability and adaptability

  The designed model may be applied to various settings or situations, which requires a good generalizability and adaptability. On the one hand, the model is desired to be generalizable to the multi-agent systems with different number of agents or various situations. On the other hand, the model is desired to be adaptable to evolving situations. More specifically, it is necessary to capture dynamic interaction/relation patterns as the multi-agent system evolves over time.

The focus of this dissertation is 1) to design a multi-agent behavior modeling framework with relational reasoning in dynamically evolving, uncertain environments for heterogeneous

agents; 2) establish a methodology to model multi-agent relations/interactions to address the fundamental problem.

1.2 Relational Reasoning and Interaction Modeling

There exist many techniques for modeling the relations and interactions between interacting agents, such as game theoretical modeling and deep learning methods. In this dissertation, we focus on the latter to design generic, highly flexible, and generalizable models for relational reasoning and interaction modeling in multi-agent systems.

In the context of multi-agent systems within the scope of this dissertation, there exist two major types of relations/interactions between different components in the system: *agent-context* and *agent-agent*.

- Agent-context: The autonomous agents need to interact with the environmental context (e.g., walls, road boundaries, traffic lights/signs), which are usually static objects with semantic meanings. The agents need to follow the restrictions brought by the context and avoid collision with obstacles.
- Agent-agent: The autonomous agents need to interact with each other when they share a common space. There can be cooperative and competitive behaviors in different scenarios or settings.

The agent-context relations/interactions has been widely studied in the area of behavior prediction, decision making and motion planning, etc. In this dissertation, we mainly investigate how to model agent-agent relations/interactions in the following sections.

1.3 Relational Feature Representation

An effective and generalizable feature representation plays a significant role in multi-agent relational reasoning and interaction modeling. The existing deep feature representations can be basically divided into three categories: *grid-based representation*, *entity-based representation* and *graph-based representation*. An illustrative diagram of each category in the context of traffic scenarios is shown in Figure 1.1.

In the grid-based representation, the state information (e.g., position, velocity) of interacting agents is typically encoded into a rasterized grid map. However, the grid-based representation is not aware of individual entities, so it can only model interactions implicitly. In the entity-based representation, the agent information is encoded individually and explicitly, which results in a feature embedding for each entity. Then, a certain feature aggregation strategy (e.g., concatenation, pooling, attention) is applied to fuse the information of multiple interacting agents to generate an interaction-aware feature embedding for downstream tasks. Although this representation is aware of individual objects, the model performance highly depends on the choice of aggregation strategies.

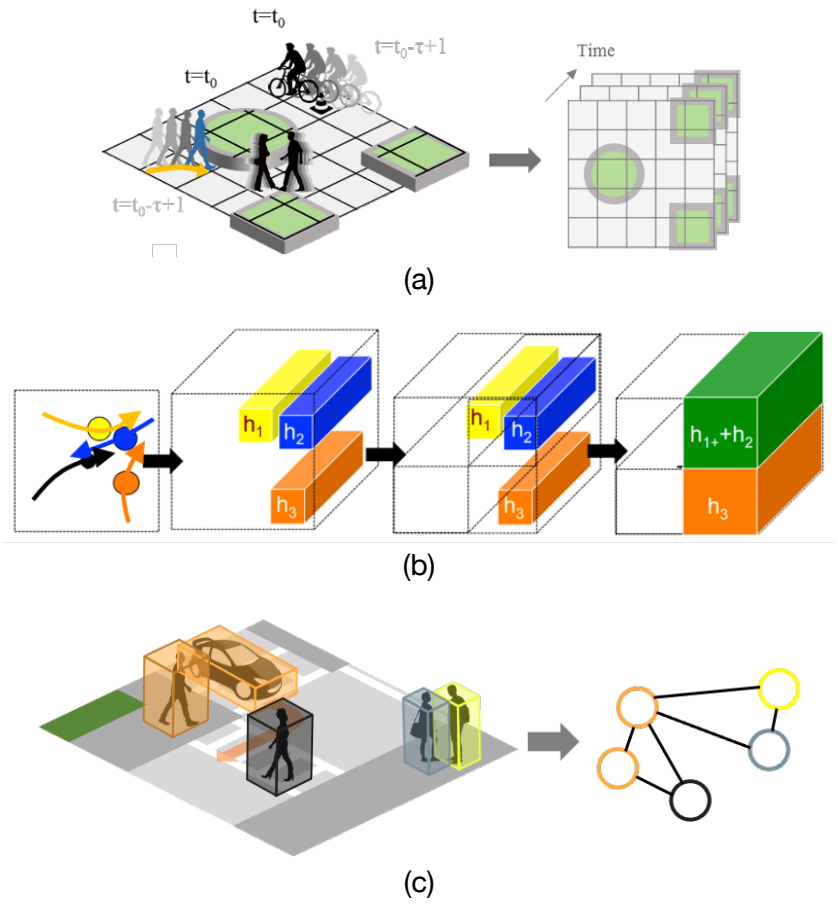


Figure 1.1: Relational feature representations of interactive agents: (a) Grid-based representation; (b) Entity-based representation; (c) Graph-based representation.

In this dissertation, we mainly employ a graph-based representation where each node represents a certain agent and the edge between a pair of nodes represents the relation/interaction of those two agents, which is a natural and uniform representation of the scene. The node attribute encodes agent state information and the edge attribute encodes relations among different agents. Graph neural networks are widely applied to process the graph information and extract relational feature embeddings. The *node function* and *edge function* are shared across the whole graph, which can handle flexible number of agents in the scene and bring the property of permutation invariance. The node/edge functions can be implemented with deep neural networks, which can capture highly flexible relation/interaction patterns and incorporate arbitrary inductive biases. With message passing over the graph, we can model arbitrary relations among nodes with flexible numbers. Higher-order interaction can be modeled by multiple rounds of updates.

1.4 Dissertation Contributions and Outline

This dissertation aims to establish a set of generic approaches to model the relations and interactions between autonomous agents or interactive entities in a multi-agent system, which can be applied to a variety of domains (e.g., physical systems, human crowds, traffic participants) and downstream tasks (e.g., prediction, tracking, importance estimation).

The remainder of the dissertation is divided into two parts according to the type of downstream tasks. Part I focuses on behavior prediction and motion tracking for autonomous agents with an emphasis on interaction modeling under a multi-agent setting. Part II focuses on object importance estimation for scene understanding with an emphasis on relational reasoning. Each chapter is intended to be self-contained with sufficient background, technical details and empirical results. The relationship among different chapters are shown in Figure 1.2. Some of the work in this dissertation has been published in [99, 96, 100, 93, 104, 98, 102] or under review in [101]. The other related work that is not included in this dissertation has been published in [94, 97, 13, 120, 118, 119, 121, 20, 36].

1.4.1 Part I: Prediction and Tracking

Chapter 2

Accurate and robust recognition and prediction of traffic situation plays an important role in autonomous driving, which is a prerequisite for risk assessment and effective decision making. Although there exist a lot of works dealing with modeling driver behavior of a single object, it remains a challenge to make predictions for multiple highly interactive agents that react to each other simultaneously. In this chapter, we propose a hierarchical time-series prediction model (HTSPM), which consists of a behavior recognition module and a state evolution module. Both modules in the proposed model are generic and flexible so as to be applied to a class of time-series prediction problems where behaviors can be separated into different levels. Instead of only focusing on forecasting trajectory of a single entity, we jointly predict continuous motions for interactive entities simultaneously. Moreover, due to the decoupling property of the layered structure, our model is suitable for knowledge transfer from simulation to real world applications as well as among different traffic scenarios, which can reduce the computational efforts of training and the demand for a large data amount. The proposed method is applied to a numerical case study as well as on-road vehicle behavior recognition and prediction in highway scenarios.

Chapter 3

Effective understanding of the environment and accurate trajectory prediction of surrounding dynamic obstacles are critical for intelligent systems such as autonomous vehicles and wheeled mobile robotics navigating in complex scenarios to achieve safe and high-quality decision making, motion planning and control. Due to the uncertain nature of the future, it is desired to make inference from a probability perspective instead of deterministic prediction. In

this chapter, we propose a conditional generative neural system (CGNS) for probabilistic trajectory prediction to approximate the data distribution, with which realistic, feasible and diverse future trajectory hypotheses can be sampled. The system combines the strengths of conditional latent space learning and variational divergence minimization, and leverages both static context and interaction information with soft attention mechanisms. We also propose a regularization method for incorporating soft constraints into deep neural networks with differentiable barrier functions, which can regulate and push the generated samples into the feasible regions. The proposed system is evaluated on several public benchmark datasets for pedestrian trajectory prediction and a roundabout naturalistic driving dataset collected by ourselves. The experimental results demonstrate that our model achieves better performance than various baseline approaches in terms of prediction accuracy.

Chapter 4

Due to the existence of frequent interactions between autonomous agents and uncertainty in the scene evolution, it is desired for the prediction system to enable relational reasoning on different entities and provide a distribution of future trajectories for each agent. In this chapter, we propose a generic generative neural system (STG-DAT) for multi-agent trajectory prediction involving heterogeneous agents. The system takes a step forward to interaction modeling by incorporating relational inductive biases with a dynamic graph representation and leverages both trajectory and scene context information. We also employ an efficient kinematic constraint layer applied to vehicle trajectory prediction. The constraint not only ensures physical feasibility but also enhances model performance. The proposed system is evaluated on three public benchmark datasets for trajectory prediction, where the agents cover pedestrians, cyclists and on-road vehicles. The experimental results demonstrate that our model achieves better performance than various baseline approaches in terms of prediction and tracking accuracy.

Chapter 5

Multi-agent interacting systems are prevalent in the world, from purely physical systems to complicated social dynamic systems. In many applications, effective understanding of the situation and accurate trajectory prediction of interactive agents play a significant role in downstream tasks, such as decision making and planning. In this chapter, we propose a generic trajectory forecasting framework (EvolveGraph) with explicit relational structure recognition and prediction via latent interaction graphs among multiple heterogeneous, interactive agents. Considering the uncertainty of future behaviors, the model is designed to provide multi-modal prediction hypotheses. Since the underlying interactions may evolve even with abrupt changes, and different modalities of evolution may lead to different outcomes, we address the necessity of dynamic relational reasoning and adaptively evolving the interaction graphs. We also introduce a double-stage training pipeline which not only improves training efficiency and accelerates convergence, but also enhances model performance. The proposed framework is

evaluated on both synthetic physics simulations and multiple real-world benchmark datasets in various areas. The experimental results illustrate that our approach achieves state-of-the-art performance in terms of prediction accuracy.

Chapter 6

Accurately tracking and predicting behaviors of surrounding objects are key prerequisites for intelligent systems such as autonomous vehicles to achieve safe and high-quality decision making and motion planning. However, there still remain challenges for multi-target tracking due to object number fluctuation and occlusion. To overcome these challenges, we propose a constrained mixture sequential Monte Carlo (CMSMC) method in which a mixture representation is incorporated in the estimated posterior distribution to maintain multi-modality. Multiple targets can be tracked simultaneously within a unified framework without explicit data association between observations and tracking targets. The framework can incorporate an arbitrary prediction model as the implicit proposal distribution of the CMSMC method. An example is the hierarchical time-series prediction model presented in Chapter 2. The proposed framework is applied to a numerical case study as well as on-road vehicle tracking in highway scenarios. The proposed approaches are evaluated from multiple aspects, which demonstrate great potential for intelligent vehicular systems and traffic surveillance systems.

1.4.2 Part II: Importance Estimation

Chapter 7

Motion forecasting plays a significant role in various domains (e.g., autonomous driving, human-robot interaction), which aims to predict future motion sequences given a set of historical observations. However, the observed elements may be of different levels of importance. Some information may be irrelevant or even distracting to the forecasting in certain situations. To address this issue, we propose a generic motion forecasting framework (RAIN) with dynamic key information selection and ranking based on a hybrid attention mechanism. The general framework is instantiated to handle multi-agent trajectory prediction and human motion forecasting tasks, respectively. In the former task, the model learns to recognize the relations between agents with a graph representation and to determine their relative significance. In the latter task, the model learns to capture the temporal proximity and dependency in long-term human motions. We also propose an effective double-stage training pipeline with an alternating training strategy to optimize the parameters in different modules of the framework. We validate the framework on both synthetic simulations and motion forecasting benchmarks in different domains, demonstrating that our method not only achieves state-of-the-art forecasting performance, but also provides interpretable and reasonable hybrid attention weights.

Chapter 8

Accurate identification of important objects in the scene is a prerequisite for safe and high-quality decision making and motion planning of intelligent agents (e.g., autonomous vehicles) that navigate in complex and dynamic environments. Most existing approaches attempt to employ attention mechanisms to learn importance weights associated with each object indirectly via various tasks (e.g., trajectory prediction), which do not enforce direct supervision on the importance estimation. In contrast, we tackle this task in an explicit way and formulate it as a binary classification ("important" or "unimportant") problem. We propose a novel approach for important object identification in egocentric driving scenarios with relational reasoning on the objects in the scene. Besides, since human annotations are limited and expensive to obtain, we present a semi-supervised learning pipeline to enable the model to learn from unlimited unlabeled data. Moreover, we propose to leverage the auxiliary tasks of ego vehicle behavior prediction to further improve the accuracy of importance estimation. The proposed approach is evaluated on a public egocentric driving dataset (H3D) collected in complex traffic scenarios. A detailed ablative study is conducted to demonstrate the effectiveness of each model component and the training strategy. Our approach also outperforms rule-based baselines by a large margin.

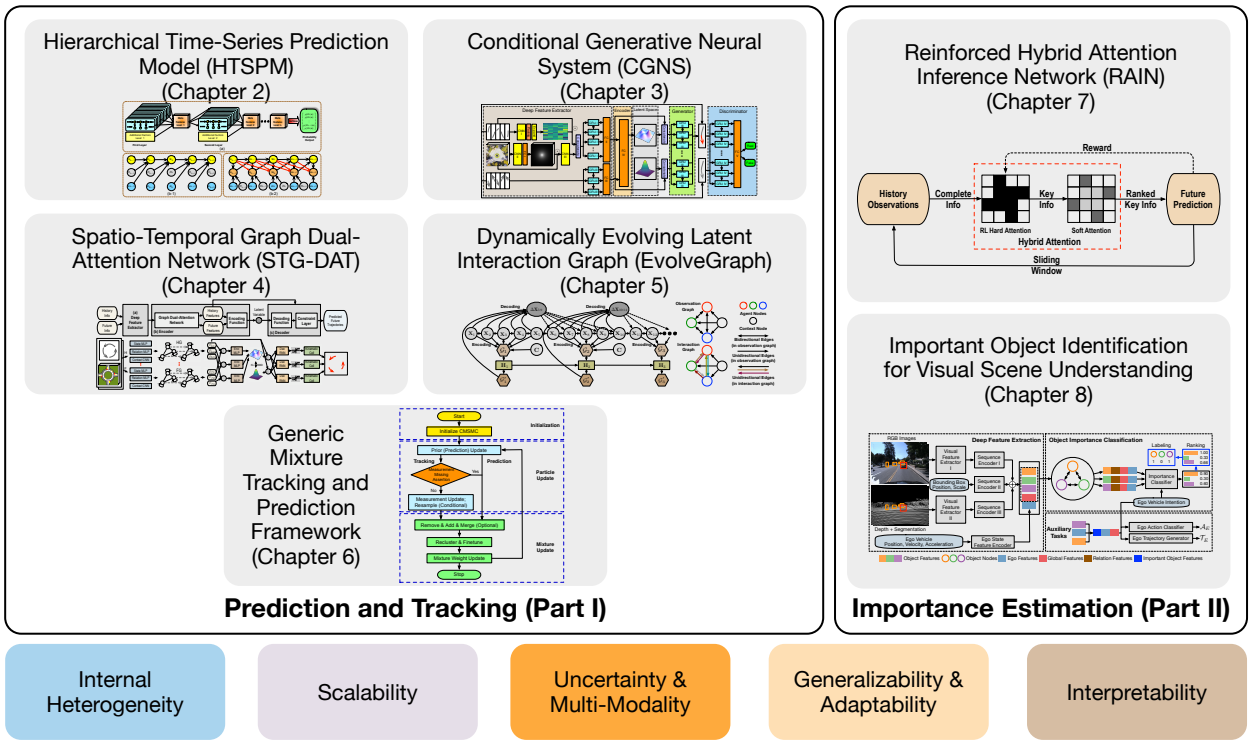


Figure 1.2: Dissertation outline.

Part I

Prediction and Tracking

Chapter 2

HTSPM: Hierarchical Time-Series Prediction Model

2.1 Introduction

Accurate and efficient recognition and prediction of future traffic scene evolution plays a significant role in autonomous driving which is a prerequisite for risk assessment, decision making and high-quality motion planning. Although a lot of research efforts have been devoted to the driver behavior recognition and prediction, most of them only focused on a single entity [78, 199, 55, 184, 205] while the information on surrounding vehicle is obtained from onboard sensor measurement and utilized as prior knowledge when making predictions. In recent years, more attention has been paid to model the interaction among multiple agents. A review on motion prediction and risk assessment was provided in [90] where the behavior models are classified into three categories: physics-based, maneuver-based and interaction-aware models. While the first two types are crucial for motion planning and control purpose, the models involving more interaction factors are important for the threat warning system and the decision making module in the Advanced Driver Assistant System (ADAS) as well as in fully autonomous vehicles [41, 30, 60, 86].

There remain several limitations and challenges when modeling human-like interactions: 1) It is hard to collect real-world driving data containing strong interactions among multiple traffic participants; 2) There is no ground truth for the driver intention since the human desire can vary from time to time. Therefore, it is desired to have an effective and robust recognition and prediction framework which takes into account uncertainty in human behavior and does not require a large amount of data.

In this work, we design a hierarchical framework which consists of a behavior recognition module and a state evolution module. Both modules in the proposed model are generic and flexible so as to be applied to a class of time-series prediction problems where behaviors can be separated into different levels. Instead of only focusing on forecasting trajectory of a single entity, we jointly predict continuous motions for interactive entities simultaneously.

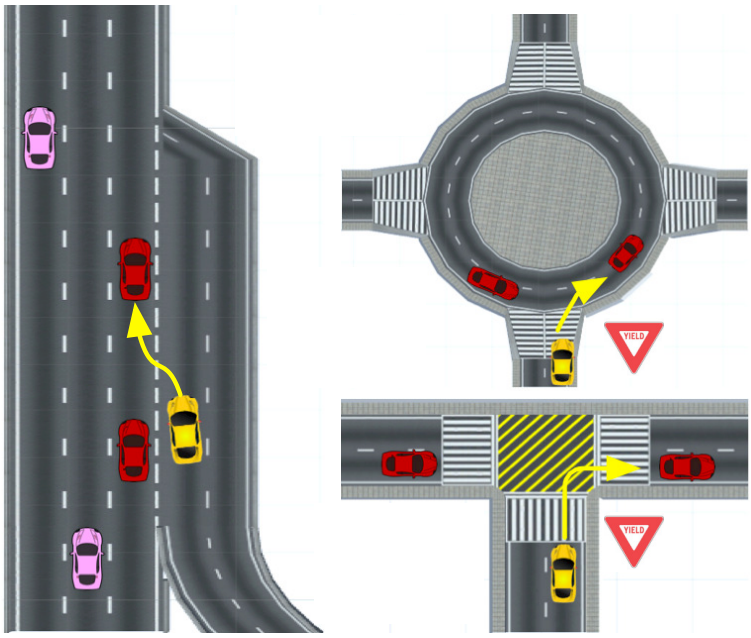


Figure 2.1: Typical traffic scenarios that lead to strong interaction among vehicles: (a) highway entrance; (b) roundabout; (c) uncontrolled T-intersection. The figures come from the bird-view of several road modules of the driving simulator developed by the authors.

The proposed method is applied to a numerical case study as well as on-road vehicle behavior recognition and prediction in highway scenarios.

2.2 Related Work

2.2.1 Driver Behavior Recognition and Prediction

Driver behavior recognition and vehicle trajectory prediction problems have been extensively investigated in literature. Widely used probabilistic models include Hidden Markov Model (HMM) [85, 183, 166], Gaussian Mixture Regression (GMR) [96, 99], Mixture Density Network (MDN) [206], Gaussian process (GP) [85], dynamic Bayesian network (DBN) [71], Rapidly-exploring Random Tree (RRT) [6], Variational Auto-Encoder (VAE) [89, 120], Generative Adversarial Network (GAN) [95, 93, 97] and multiple model approaches [157]. In this paper, we propose a hierarchical probabilistic model structure that can incorporate any of the above models for tracking and prediction. Moreover, instead of modeling each entity individually, we treat multiple interactive agents as a whole system and model the joint distribution of their future behaviors and motions.

2.2.2 Layered Hidden Markov Model

Layered Hidden Markov Model (LHMM) has a hierarchical architecture which consists of multiple layers of standard HMM. This layered representation is of great advantages when modeling multi-level activities such as human motions. To the best of our knowledge, the concept of LHMM was first proposed in [135] to recognize human states in an static office environment. In [1], the method was further applied to human motion intention recognition where the high-level complicated tasks are divided into several low-level primary sub-tasks. The results revealed much better performance for LHMM than a canonical one-layer HMM. In [166], a Monte Carlo Layered HMM capable of online learning was proposed to predict robot leader's motions, which is used to improve tracking accuracy.

This layer-structured model is suitable for dynamic situation modeling due to the fact that a multi-agent highly interactive process can be decomposed into certain stages, which will be explained in detail in Section III. Moreover, a novel formulation and interpretation for the model is provided.

2.3 Method: HTSPM

In this section, we present a hierarchical prediction model for time-series problems which consists of two modules: recognition module and evolution module. The recognition module aims at solving a probabilistic classification problem while the evolution module aims at propagating the current state to the future.

Consider a general nonlinear discrete-time state space system with equality and/or inequality constraints on the state which can be formulated as

$$\begin{aligned} \mathbf{x}_k &= q_{k-1}(\mathbf{x}_{k-1}, \mathbf{e}_{k-1}, \mathbf{v}_{k-1}), \\ \mathbf{z}_k &= h_k(\mathbf{x}_k, \mathbf{w}_k), \\ \mathbf{x}_k &\in \mathbb{S}_{\mathbf{x}_k}, \end{aligned} \tag{2.1}$$

where the subscript k denotes the time step, $\mathbf{x}, \mathbf{e}, \mathbf{z}, \mathbf{v}, \mathbf{w}$ denotes the state vector, the exterior information, the measurement vector, the process noise and the measurement noise, respectively. Note that the random variable $\mathbf{e}$ is involved in our work since the state evolution can be affected by exterior factors, which rarely emerges in the canonical formulation. $\mathbb{S}_{\mathbf{x}_k}$ denotes the feasible state set satisfying all the constraints. $q(\cdot)$ represents the process model (a.k.a. system dynamics model) and $h(\cdot)$ represents the measurement model. The process model and measurement model can be time-invariant or time-variant and the noise value can be sampled from arbitrary distributions. Mathematically, the proposed model is used to approximate the state transition distribution

$$\begin{aligned} \hat{f}(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{e}_{k-1}) = \sum_B \int_{\mathbf{u}_{k-1}} \int_{\mathbf{z}^k} \hat{f}(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \\ \times \hat{f}(\mathbf{u}_{k-1}|\mathbf{x}_{k-1}, \mathbf{e}_{k-1}, B_{k-1}) \hat{f}(B_{k-1}|\mathbf{z}^k) \mathrm{d}\mathbf{u}_{k-1} \mathrm{d}\mathbf{z}^k, \end{aligned} \tag{2.2}$$

where B denotes behavior class set elements.

2.3.1 Recognition Module

The recognition module aims at obtaining the posterior probabilities of classes $\hat{f}(B_{k-1}|\mathbf{z}^k)$ where B_{k-1} is a discrete random variable representing the class at time step $k-1$. In many applications, it is reasonable to make an assumption that $\hat{f}(B_{k-1}|\mathbf{z}^k) \approx \hat{f}(B_{k-1}|\mathbf{z}^{k-T:k})$ where $\mathbf{z}^{k-T:k} = (\mathbf{z}_{k-T}, ..., \mathbf{z}_k)$ and T is a properly chosen period length according to specific problem setups since the most recent several observations have far more significance than the past ones when deciding the current behavior class probabilities. There is no limitation on the specific architecture of recognition model provided it provides posterior class probabilities. Although there exist many widely used probabilistic classifiers that can be directly employed in this module, we propose a *Deep Hidden Markov Model (DHMM)* which has advantages over existing ones in multiple aspects.

The DHMM has a multi-layer architecture, which is shown in Figure 2.2. Each layer is composed of a group of canonical Hidden Markov Models (HMM). In general, the l-th layer HMM take in the outputs of the $(l-1)$-th layer and extract meta features which are used as observations for the $(l+1)$-th layer HMM. For instance, the first-layer HMM take in the lowest-level observations such as raw sensor measurement of state as well as exterior information and extract meta features which are utilized as observations of the second-layer HMM. Since there may be additional factors of different levels affecting the recognition results, they are incorporated into the meta features at proper levels. This information extraction and message passing mechanism applies to both training and recognition phases. After extracting the meta features of the last layer, there is a $Softmax$ layer calculating the posterior probabilities. In this work, we use the observation log-likelihood of each layer HMM as the meta features. An alternative for meta features are the class indices with highest likelihood which results in degeneracy into deterministic recognition except the output layer, which is suitable when large distinctions exist among different classes. This can also reduce computational efforts. In this work, we used the former in all the experiments.

Training Phase

The whole training trajectories are divided into proper segments at different levels and properly labeled. The DHMM is trained from the first layer to the last layer successively. More specifically, assuming that there are h_l classes in the l-th layer, then h_l HMM are trained with log-likelihood sequences obtained from the $(l-1)$-th layer output using the Baum-Welch algorithm (a.k.a Forward-Backward algorithm) [190]. In order to choose the best hidden state number, we can use the Bayesian information criterion (BIC) as a performance indicator by pre-fitting a Gaussian mixture distribution to the training feature sequences.

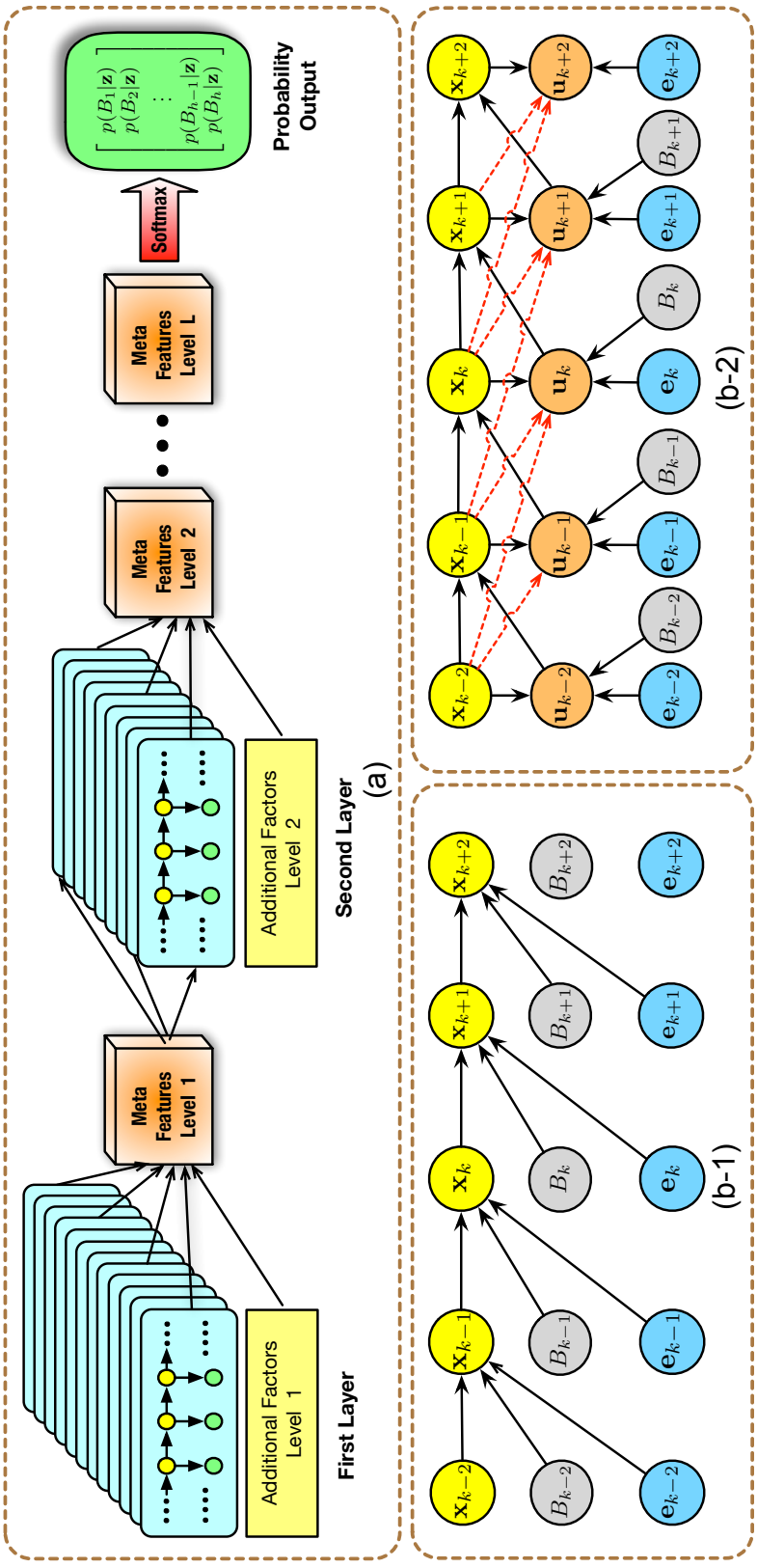


Figure 2.2: The diagram of hierarchical time-series prediction model. (a) Recognition module: Deep Hidden Markov Model (DHMM); (b-1) Evolution module: the exterior information affects the state directly; (b-2) Evolution module: the exterior information and behavior pattern affect the state indirectly through an action (input) term. The black solid arrows represent first-order Markov assumption and the red dashed ones imply higher order assumptions.

Recognition Phase

Given a new observation sequence, we also use a bottom-up procedure similar to the training phase to obtain the posterior class probabilities. The implementation details are summarized in the first half of **Algorithm 1**, where T_l is the period length for the l-th layer which needs to be tuned properly for different problems and application scenarios. If it is too large, the extracted meta feature sequence will have a short length, which passes less observations to future layers; if it is too small, the quality of current-layer output will decrease. In practice, a possible issue is the observation sequence likelihood obtained by different class HMM may be in different number scales, which will lead to a class with an absolute dominant probability (≈ 1) at all time. Therefore, it is recommended to incorporate a set of calibration parameters $\alpha_h, h = 1, 2, ..., h_l$ to make equal probabilities for each class at the initial time step and keep the same parameter values afterwards.

We select several popular probabilistic classifiers as baseline models and make comparisons in the numerical case study.

- **Standard Hidden Markov Model (HMM)**: Instead of partitioning the high-level behaviors into multiple stages like DHMM, HMM classifier treats the entire trajectory as a whole. In the training phase, the whole sequences are utilized to train the HMM by Baum-Welch algorithm. In the recognition phase, a segment of historical information are fed to HMM to obtain the likelihood and normalized into probabilities.

- **Gaussian Discriminant Analysis (GDA)**: The essence of GDA is to obtain a linear decision surface for Linear Discriminant Analysis (LDA) or a quadratic decision surface for Quadratic Discriminant Analysis (QDA) through proper transformations of raw features which can distinguish among categories and perform classification in the transformed space according to some distance metric such as Euclidean distance. While LDA assumes that all the classes share the same covariance matrix, QDA fits a particular covariance matrix for each class. The feature matrix consists of historical observation information and the labels are behavior indices.

- **Gaussian Naive Bayes (GNB)**: NB is a typical probabilistic classifier which employs Bayes' theorem and assumes that features are mutually independent [194]. In this work, we assume that the feature likelihood to be Gaussian distribution, which establishes a GNB.

The advantages of the proposed DHMM over above baseline models are four folds:

- Compared with the models which only take raw observations as input features, our model is able to extract multi-level features and robust to measurement noise and sensor failures.

- Compared with other deep models such as deep neural networks, our model requires a significantly less amount of training data and computational cost as well as maintains interpretability from a probability perspective.

- Thanks to the layered representation and decomposability between layers, DHMM has potential knowledge transferability among similar tasks. It will reduce much training efforts if we can utilize several parameters directly or finetune from well-trained models for new tasks.

- The training and recognition processes can be parallelized since the learning and inference of HMM are independent within a layer.

2.3.2 Evolution Module

The evolution module is designed to obtain the conditional state transition distribution given a certain behavior pattern $\hat{f}(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{e}_{k-1}, B_{k-1})$ which is demonstrated in Figure 2.2(b-1) where the exterior information has effects on the state directly. When the exterior information and behavior pattern affect the state indirectly through an action term, the conditional distribution can be further extended to $\hat{f}(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_{k-1})\hat{f}(\mathbf{u}_{k-1}|\mathbf{x}_{k-1}, \mathbf{e}_{k-1}, B_{k-1})$ which is presented in Figure 2.2(b-2). The detailed procedures of evolution phase can be found in the second half of **Algorithm 1**. In this work, we demonstrate three learning-based state evolution models and compare their performance in the experiments.

- **Conditional Gaussian Mixture Regression (CGMR)**: The driver behavioral model proposed in the authors' previous work [96] is adapted and generalized as CGMR which is based on a Gaussian mixture model (GMM). The conditional Gaussian mixture distribution is a linear combination of multiple Gaussians with the form $f(\zeta^i) = \sum_{g=1}^{N} \pi_g^i \mathcal{N}(\zeta^i|\mu_g^i, \mathbf{\Sigma}_g^i)$ where i is the behavior class index, $\sum_{g=1}^{N} \pi_g^i = 1$, μ_g^i and $\mathbf{\Sigma}_g^i$ are the mean and covariance of the g-th Gaussian distribution, and ζ^i is the training dataset for the i-th behavior. In each training sample, the input and output are stacked into a column vector which is denoted as $\zeta^i = [\ \mathcal{I}^i \mid \mathcal{O}^i\ ]^T$, where $\mathcal{I}^i$ denotes the conditional variables and $\mathcal{O}^i$ denotes the predicted variables. The dimensions of the two variables are arbitrary. For instance, in Fig. 2(b-2) the $\mathbf{e}_{k-1}, B_{k-1}$ and $\mathbf{x}_{k-1}$ can be treated as conditional variables while $\mathbf{u}_{k-1}$ and $\mathbf{x}_k$ can be treated as the corresponding predicted variables. The training and prediction method are identical to [96].

- **Conditional Probabilistic Multi-Layer Perceptrons (CP-MLP)**: MLP is a subclass of deep neural network which consists of a directed acyclic feed-forward architecture [29]. The network input is historical state information and the output are the actions in a certain length of time horizon. To improve generality, an $L2$-regularization term is added in the loss function and dropout layers are incorporated. However, since canonical MLP is a deterministic model, we add a noise term sampled from normal distribution to the network input to incorporate uncertainty during both training and test process. We train a P-MLP for each behavior class, which establishes a set of CP-MLP.

Algorithm 1: HTSPM Prediction Algorithm

Require:

1. The number of layers L and well-trained HMM of all layers HMM-l-h $(l = 1, ..., L; h = 1, ..., h_l)$;
2. The test raw observation sequence;
3. The last step particle hypotheses $\{\mathbf{x}_{k-1}^{(j)}, j = 1, ..., N_p\}$.

Ensure:

The current particle hypotheses $\{\mathbf{x}_k^{(j)}, j = 1, ..., N_p\}$.

Recognition phase:
$obs \leftarrow$ raw observation sequence;
$len \leftarrow$ Length(obs);
for $l = 1, 2, ..., L$ **do**
  for $h = 1, 2, ..., h_l$ **do**
    for $i = 1, 2, ..., len - T_l$ **do**
      $\mathcal{L}_{lh}$.append(Likelihood(HMM-l-h, $obs[i : i + T_l]$));
    end for
  end for
  $\mathcal{L}_l \leftarrow$ concatenating $\mathcal{L}_{lh}$;
  $obs \leftarrow \mathcal{L}_l$; $len \leftarrow$ Length(obs);
end for
$probability \leftarrow$ Softmax(obs);
Evolution phase:
for $j = 1, 2, ..., N_p$ **do**
  Sample $B_{k-1}^{(j)}$ from $probability$;
  Sample $\mathbf{u}_{k-1}^{(j)}$ from $\hat{f}(\mathbf{u}_{k-1}|\mathbf{x}_{k-1}, \mathbf{e}_{k-1}, B_{k-1}^{(j)})$;
  if $\mathbf{u}_{k-1}^{(j)}$ not feasible **then**
    Resample from proposal distribution;
  else
    $\mathbf{x}_k^{(j)} \leftarrow \mathcal{F}(\mathbf{x}_{k-1}^{(j)}, \mathbf{u}_{k-1}^{(j)})$;
  end if
end for
return $\{\mathbf{x}_k^{(j)}, j = 1, ..., N_p\}$

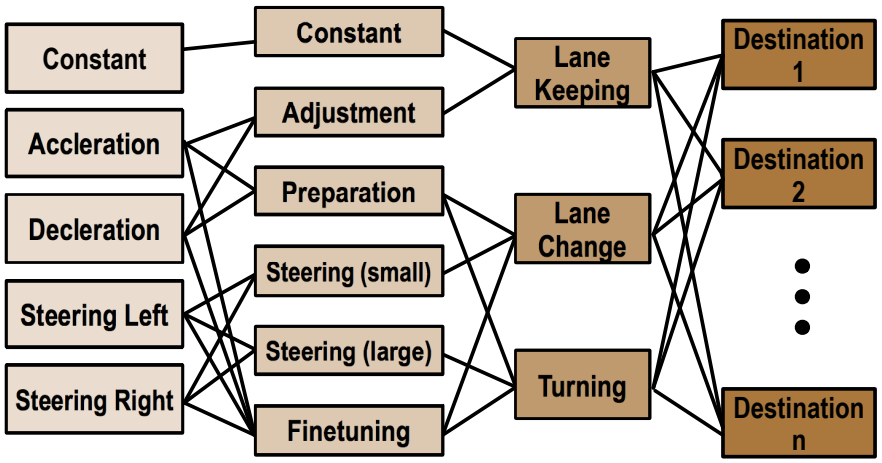


Figure 2.3: The hierarchical representation of driver behaviors. In daily driving situations, there are three common behaviors that can be arranged to get to any destination accordingly: lane keeping, lane change and turning. These behaviors can also be decomposed to more primary actions such as speed adjustment and steering which also have composing elements. Each behavior level corresponds to a layer and each behavior class at a certain level corresponds to an HMM.

- **Conditional Probabilistic Long Short-Term Memory (CP-LSTM)**: LSTM is a widely used variant of recurrent neural network (RNN) which is suitable for time-series data modeling and can effectively avoid gradient explosion and vanishing issues [54]. The network takes in a sequence of historical state and gives out a sequence of future actions. Similar to CP-MLP, a noise term is also appended to the input features to involve uncertainty. The CP-LSTM has a similar architecture to CP-MLP except that the first hidden layer is replaced with a LSTM layer.

2.3.3 Application Scopes

The proposed hierarchical time-series model is especially suitable for recognition and prediction of complicated events consisted of multi-level sub-stages, such as driver behaviors and human activities; or classification problems in which the high-level category has several sub-classes, such as in natural language processing. An illustration of exemplar application scenarios specifically for driver behaviors can be found in Figure 2.3. The proposed method is validated and discussed in the following case studies.

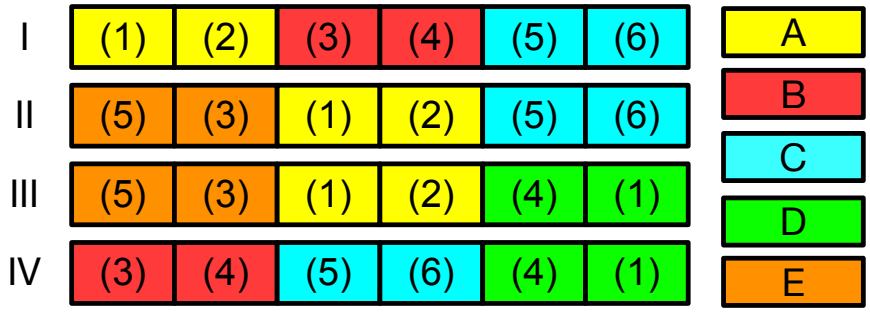


Figure 2.4: The diagram of hierarchical behavior representation in the numerical case. There are four high-level behaviors I, II, III and IV which are composed of three of the five stages A, B, C, D and E while each of these stages can also be separated into two of the six sub-stages (1)-(6). The order of stages and sub-stages are fixed in each high-level behavior.

Table 2.1: The function $g(k)$ for each sub-stage.

Sub-stages	$g(k)$	Sub-stages	$g(k)$	Sub-stages	$g(k)$
(1)	1.5	(2)	$1.5\cos(0.1k)$	(3)	-0.75
(4)	$3\sin(0.1k)$	(5)	-3	(6)	3

Table 2.2: The Semantic Labels of DHMM (Numerical Example)

Index	**Stage**	**Index**	**Stage**	**Index**	**Stage**
HMM-1-1	(1)	HMM-1-6	(6)	HMM-2-5	E
HMM-1-2	(2)	HMM-2-1	A	HMM-3-1	I
HMM-1-3	(3)	HMM-2-2	B	HMM-3-2	II
HMM-1-4	(4)	HMM-2-3	C	HMM-3-3	III
HMM-1-5	(5)	HMM-2-4	D	HMM-3-4	IV

2.4 Case Study I: Multivariate Dynamic System

In this section, we use a general numerical case to demonstrate the effectiveness and accuracy of the HTSPM prediction framework. In particular, we compare the recognition and prediction performance of the proposed HTSPM with widely used probabilistic classifiers and canonical state evolution models, respectively.

2.4.1 Problem Formulation

In this example, we set a fixed prediction entity number and each entity is assigned one of the four high-level behaviors shown in Figure 2.4. The training, validation and test trajectories for the i-th entity are generated by a nonlinear state space model with state constraints

$$\begin{aligned}
x^i_{1,k} &= x^i_{1,k-1} + 2x^i_{2,k-1}\Delta T + x^i_{3,k-1}\Delta T^2 + v^i_{1,k-1},\\
x^i_{2,k} &= x^i_{2,k-1} + x^{i2}_{3,k-1}\Delta T + v^i_{2,k-1},\\
x^i_{3,k} &= x^i_{3,k-1} + g^i(k-1) + v^i_{3,k-1},\\
z^i_{1,k} &= x^i_{1,k} + w^i_{1,k}, \quad w^i_1 \sim \mathcal{N}(0, 0.5),\\
z^i_{2,k} &= x^i_{2,k} + w^i_{2,k}, \quad w^i_2 \sim \mathcal{N}(0, 0.5),\\
x^i_{2,k} &\geq 0,\ -10 \leq x^i_{3,k} \leq 10,\\
x^i_{1,0} &\sim \mathrm{U}[0, 20],\ x^i_{2,0} \sim \mathrm{U}[10, 20],\ x^i_{3,0} \sim \mathcal{N}(0, 0.1),\\
v^i_1 &\sim \mathcal{N}(0, 0.5),\ v^i_2 \sim \mathrm{U}[-1, 1],\ v^i_3 \sim \mathrm{U}[-0.1, 0.1],
\end{aligned} \tag{2.3}$$

where ΔT is the period length between two time steps, $v_{j,k-1}$ $(j = 1, 2, 3)$ is process noise, w_k is measurement noise, $\mathrm{U}[\cdot, \cdot]$ and $\mathcal{N}(\cdot, \cdot)$ denotes uniform and Gaussian distribution respectively and $g(k)$ is a manually defined function whose detailed forms for each sub-stages (1)-(6) are provided in Table 2.1. This process model is nonlinear with non-Gaussian noise. The generated state trajectories of four behaviors can be spatially distinguished in the state space. We do not endow any physical meanings to states for generalization purpose.

2.4.2 Experiment Details and Results

We demonstrate the validity and effectiveness of the proposed framework and models by the following experiments. The experiment details are provided and results are analyzed.

Under this problem formulation, the DHMM has three layers corresponding to the three stage levels whose semantic labels are presented detailedly in Table 2.2. Figure 2.5 demonstrates the recognition results of each layer in DHMM for four cases in different behavior classes which possess a good interpretability. The comparisons of recognition results between DHMM and widely used probabilistic classifiers are shown in Figure 2.6. Detailed analysis can be found in the captions.

2.5 Case Study II: Highway Ramp Merging

In this section, a typical highway ramp merging scenario is investigated to validate the effectiveness and accuracy of the proposed approach. The data source, experiment details are presented and results are analyzed.

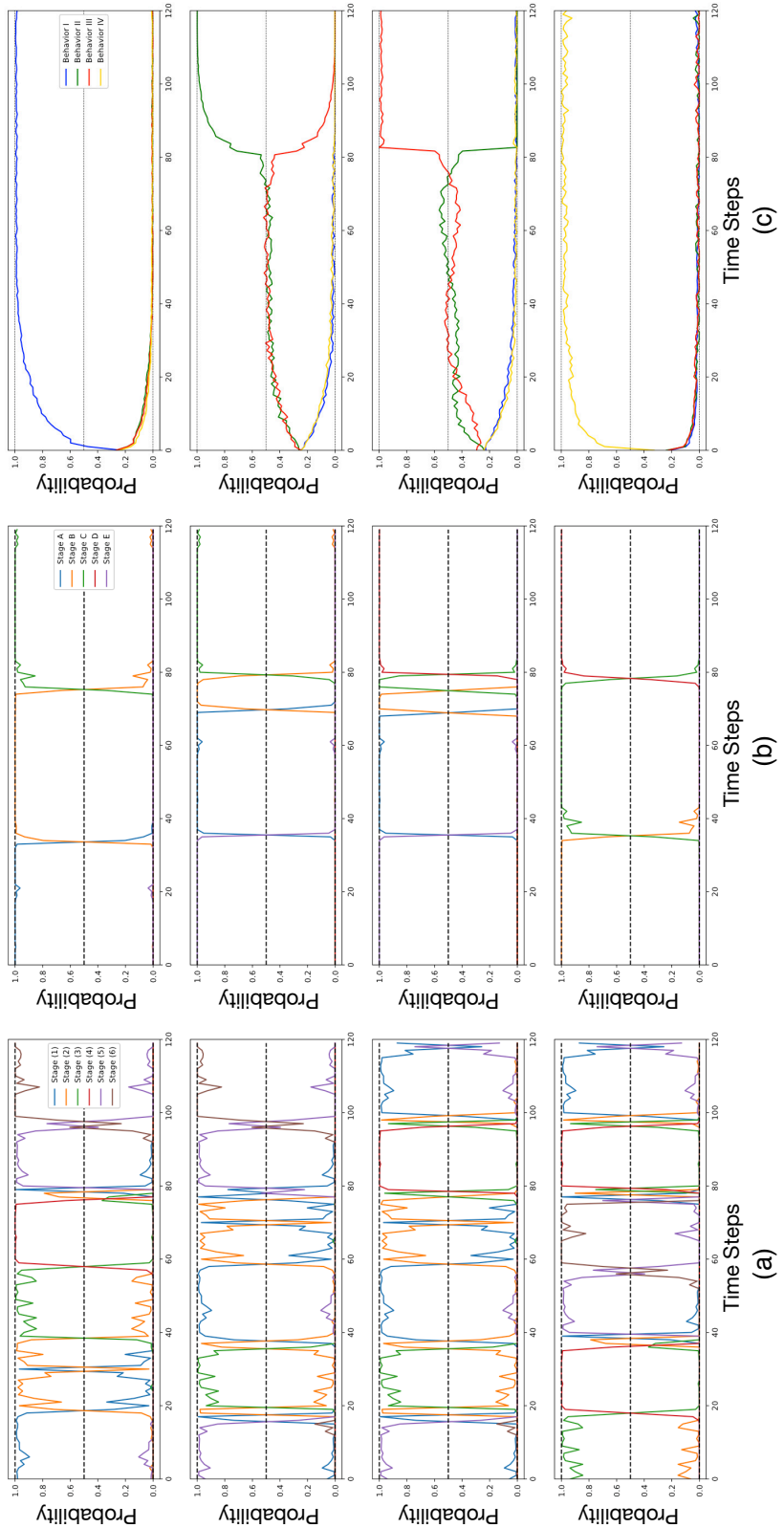


Figure 2.5: The visualization for outputs of three-layer DHMM. Each row illustrates a case from a certain behavior. (a) The first layer output: the first-layer HMM can recognize six lowest-level stages and provide proper probability sequences despite some noise; (b) The second layer output: the second-layer HMM recognize five mid-level stages based on the output of first layer, which gives probability sequences with smaller noise; (c) The third layer output (behavior class probabilities): since there is an overlap of low-level stages between Behavior II and III, it is reasonable that the DHMM cannot distinguish them until the trajectories evolve differently, thus predicts a probability around 0.5 for each before divergence and recognize the right class quickly thereafter.

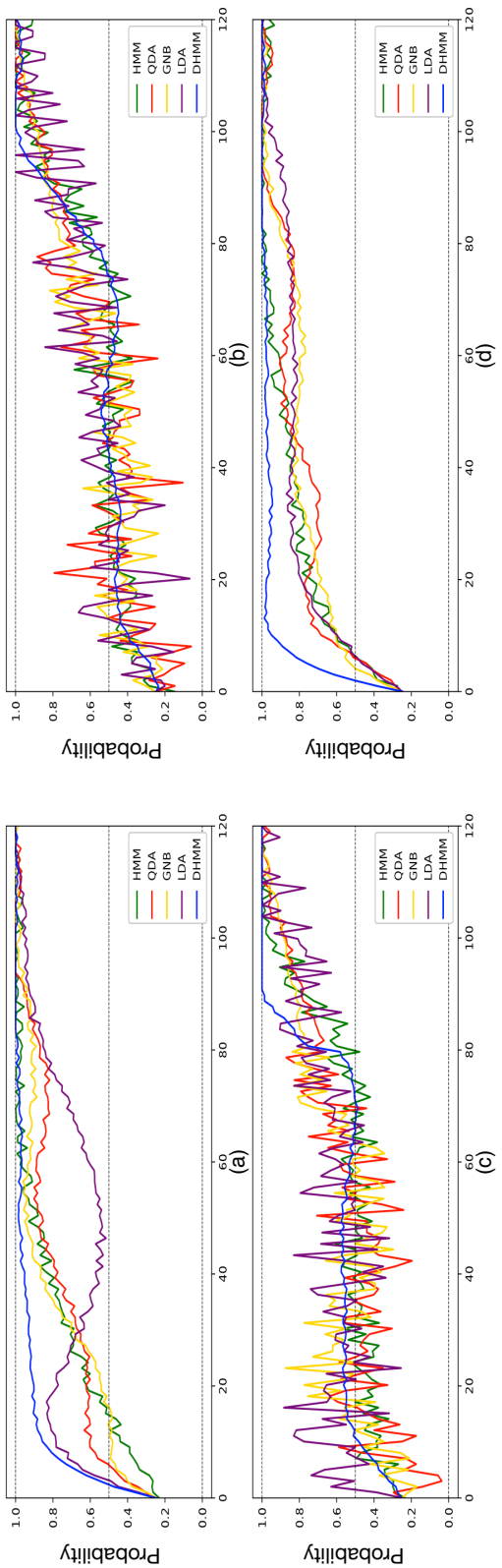


Figure 2.6: The comparisons of recognition results of probabilistic classifiers for four behavior classes. (a) Behavior I case; (b) Behavior II case; (c) Behavior III case; (d) Behavior IV case. In (a) and (d), our DHMM is able to recognize the true behavior class earliest among the five classifiers as well as output a relatively stable probability. In (b) and (c), there exist much larger fluctuations and noise in the recognition results of the other classifiers than DHMM especially in the first 80 time steps during which Behavior II and III are identical, which indicates our DHMM is more robust to feature fluctuations and noise. The reason is that while the inference outputs of other classifiers are highly dependent on raw feature sequences, our model is able to extract different level meta feature sequences which are more stable and easier to classify, which reduces the influences of raw feature noise.

2.5.1 Problem Statement

Among various highway ramps with different road curvature, the schematic diagram can be simplified as Figure 2.7. The gray car represents the merging vehicle while the red car stands for the car in the main lane with which the merging agent has major interaction. The yellow car is the leading vehicle which affects the available gap for the merging car. There are two potential interaction outcomes: the red car yields the gray car; or the opposite. The goal is to obtain the posterior probability of each outcome and the joint distribution of future trajectories of the two entities given a sequence of historical information.

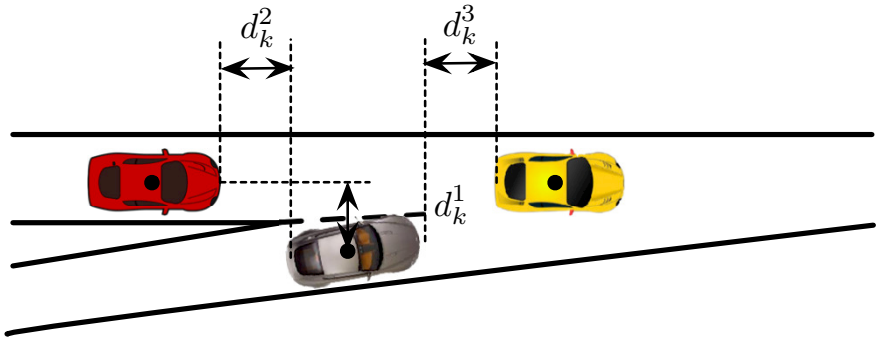


Figure 2.7: A simplified diagram of highway ramp merging process.

2.5.2 Data Collection and Pre-processing

Our training data and test data come from two sources: virtual simulation and real-world driving dataset. We developed a driving simulator based on Unity3d for data collection and visualization where a steering wheel and two pedals for acceleration and braking are used to collect human inputs for each involved driver. The position, velocity and yaw angle information of all the agents in the scene can be recorded in real time. We collected 128 interaction events in total with balanced distribution of two situations.

We also adopted the Next Generation Simulation (NGSIM) US I-80 Freeway dataset as naturalistic driving data which is available online [23]. We selected 100 interaction events in total. Since the original data is noisy which results in highly fluctuating velocities and accelerations, we applied an Extended Kalman filter (EKF) to smooth the trajectories. For both simulation data and real-world data, we randomly selected 80% as the training set and the others were used as test set.

2.5.3 Experiments and Results

There are two potential situations in this scenario: the merging agent cuts in ahead of or behind the main lane agent, corresponding to the two gaps shown in Figure 2.7. A complete

Table 2.3: The Semantic Labels of Two-Layer Hidden Markov Model

Index	**Situation**	**Stage**
HMM-1-1	Main/Merge lane car yields	Ambiguity
HMM-1-2	Main lane car yields	Preparation
HMM-1-3	Main lane car yields	Merging
HMM-1-4	Main lane car yields	Car following
HMM-1-5	Merge lane car yields	Preparation
HMM-1-6	Merge lane car yields	Merging
HMM-1-7	Merge lane car yields	Car following
HMM-2-1	Main lane car yields	—
HMM-2-2	Merge lane car yields	—

merging process can be typically partitioned into four stages:

- **Ambiguity**: The two cars advance in their own lanes without strong interaction with each other;
- **Preparation**: At this stage, the two agents are getting closer and the one willing to yield tends to decelerate to avoid collision and provide more space for the other one which may accelerate or keep a constant speed;
- **Merging**: At this stage, the merging agent cuts into one of the possible gaps;
- **Car following**: The interaction terminates and the yielding one starts car-following-like behavior.

The motion patterns and vehicle states in the *Ambiguity* stage have no obvious distinctions between the two situations since the two agents have not made decisions on whether to yield or not. However, there exist large differences in the other three stages (e.g. in the merging stage and car following stage, the relative longitudinal positions in the two situations are opposite).

Our experiment contains two parts. The proposed model is first trained and tested with the simulation dataset to demonstrate the recognition and inference performance; then we sample a set of future trajectories to approximate the joint distribution of the two vehicles' states. The well-trained LHMM is also tested on the real-world dataset with or without finetuning to illustrate the transferability. The details are introduced in the following subsections.

Table 2.4: State Features and Action Labels

	Notations	Descriptions
State features	y_k^1	Longitudinal position of the main lane car
	y_k^2	Longitudinal position of the merging car
	d_k^1	Lateral distance of two cars
	$\dot{y}_k^1$	Longitudinal velocity of the main lane car
	$\dot{y}_k^2$	Longitudinal velocity of the merging car
	$\ddot{y}_k^1$	Longitudinal acceleration of main lane car
	$\ddot{y}_k^2$	Longitudinal acceleration of main lane car
Action labels	$\Delta x_{k\to k+1}^1$	Traveled lateral distance of main lane car
	$\Delta x_{k\to k+1}^2$	Traveled lateral distance of merging car
	$\dot{y}_{k+1}^1$	Longitudinal velocity of main lane car
	$\dot{y}_{k+1}^2$	Longitudinal velocity of merging car

Virtual Simulation

In this part, we assume that the available gap between the main lane vehicle and the leading vehicle is large enough for the merging vehicle to cut in which implies that the leading car has little influence on the interaction process.

The whole trajectories of each interaction event were divided into four segments corresponding to four stages based on the road information and vehicle states. We trained 7 first-layer HMM individually using the segmented feature sequences. Then we obtained the log-likelihood sequences of both situations from the inference output and fed them to the second layer HMM as training observation sequences. The labels of each HMM are shown in Table I, where HMM-i-j refers to the j-th HMM in the i-th layer. In order to obtain the conditional distribution of actions $f(a_k|SE_k)$, we trained a Gaussian Mixture Model with the state features and action labels listed in Table 2.4.

Three typical test cases are used to demonstrate the performance of the proposed approach. The inference results of the first-layer HMM are shown in Figure 2.8 (a-1), (b-1) and (c-1). We can see that in the first-layer inference output, the observation likelihood of HMM-1-1 is the highest at the early stage for all the test cases, which implies vague decisions of two agents corresponding to the *Ambiguity* stage. Afterwards, the likelihood of HMM-1-2, HMM-1-3 and HMM-1-4 tend to dominate in the "*main lane car yields*" cases successively while HMM-1-5, HMM-1-6 and HMM-1-7 tend to dominate in the "*merge lane car yields*" cases successively, which implies that the first-layer HMMs are able to capture different evolution patterns of the raw feature sequences in both situations.

Moreover, we compared the performance of the two-layer HMM with a standard HMM classifier and Quadratic Discriminant Analysis (QDA), which is illustrated in Figure 2.8 (a-2),

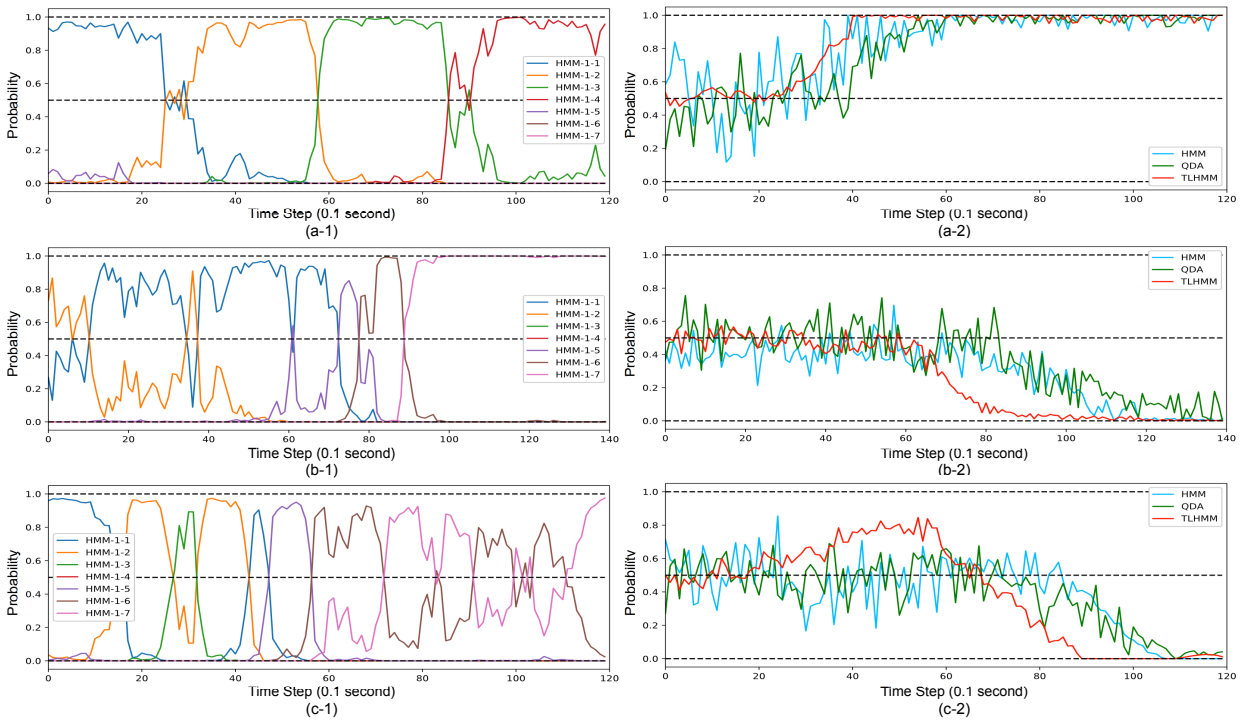


Figure 2.8: The DHMM inference output for three test case: (a) The main lane car yields the merging car; (b) The merging car yields the main lane car; (c) The merging car tends to cut in ahead of the main lane car while the main lane car does not yield. (a-1), (b-1) and (c-1) show the first-layer HMM inference output; and (a-2), (b-2) and (c-2) show the probability output of the second-layer HMM for "*main lane car yields*".

(b-2) and (c-2). We can see that for all the cases our model is able to recognize the true interaction result earliest among the three models. On the other hand, there is much less fluctuation in the probability output of our model than the other models, which implies that our model is more robust to the raw feature fluctuations and observation noise from the sensor measurement. The reason is that while the inference output of standard HMM and QDA are very sensitive to raw feature evolution, our model are sensitive to meta feature evolution, which reduces the effects of raw feature noise.

An occupancy heatmap is provided in Figure 2.9 to illustrate the predicted distribution of future trajectories generated by the GMM. The ground truth trajectory is located near the mean of the distribution, which implies that our model is able to make accurate long-term prediction.

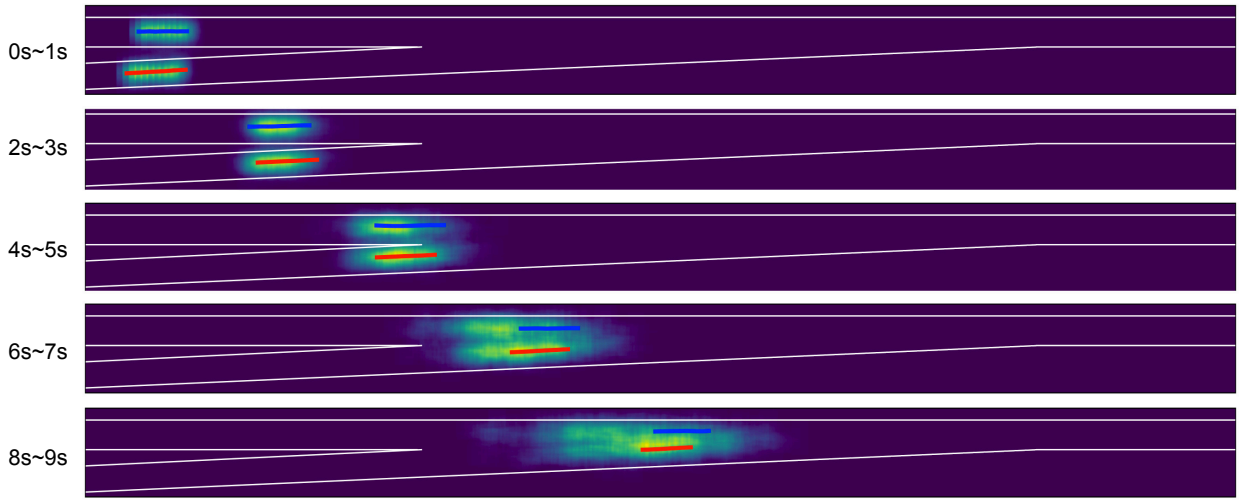


Figure 2.9: An illustrative heat map of long-term dynamic scene evolution. It contains predictions of the same case from the same initial state with different propagation length. The red line is the ground truth of the merging vehicle and the blue line is the ground truth of the main lane vehicle.

Simulation to Real World Transfer

In this part, we investigate the capability of knowledge transfer of the proposed model. In the NGSIM dataset, since the distances among vehicles are relatively small and the traffic speed is low, the influence of the leading vehicle cannot be ignored. Therefore, we added the position and velocity information of the leading vehicle into the raw feature sequences.

Since the road geometry and vehicle speed range in the NGSIM dataset are different from the simulation data, the first-layer HMM has to be retrained. For the second-layer HMM, three model setups were evaluated: 1) directly employ the well-trained second layer HMM without finetuning the parameters; 2) employ the well-trained parameters as the initial values and finetune the second-layer HMM; and 3) train the two-layer HMM from scratch using the NGSIM dataset. It took 112 iterations in average for the EM algorithm to converge when training the model from scratch but only 35 iterations when finetuning the pre-trained model. The inference results of two test cases are demonstrated in Figure 2.10. We can see that the transferred model with finetuning has comparable performance to the model trained from scratch with much less computation cost.

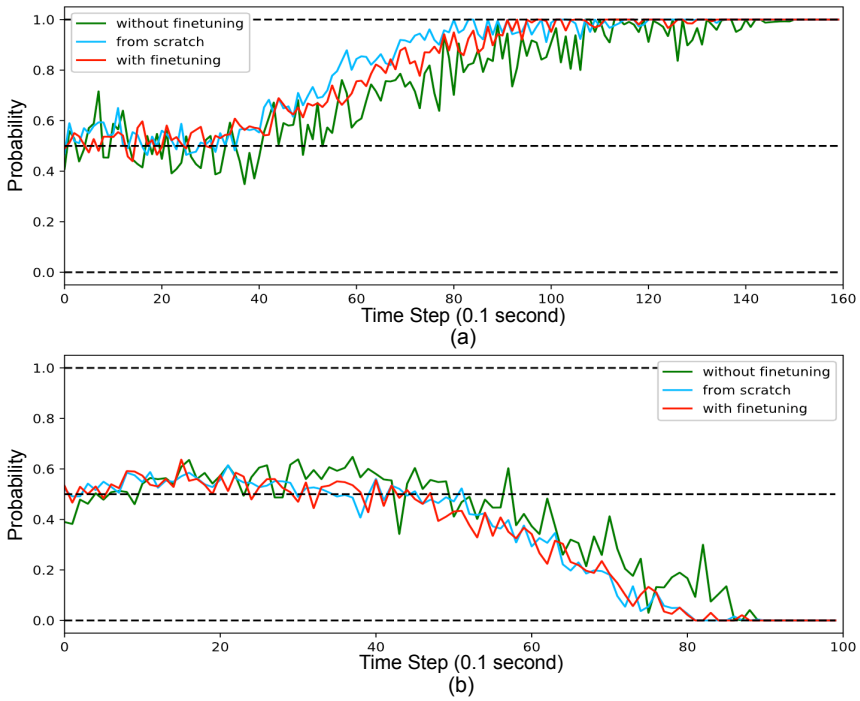


Figure 2.10: The comparison of inference results among three model setups. (a) a *"main lane car yields"* case; (b) a *"merging car yields"* case.

2.6 Case Study III: Highway Lane Changing

In this section, we apply the proposed framework and hierarchical time-series prediction model to solve real-time vehicle motion prediction problems in a highway lane keeping and lane changing setting. We investigate a highway scenario as an illustrative example in which we only consider lane keeping and lane change behaviors due to the restriction of road geometry. The data source, experiment details, results and comparisons of different models are illustrated and discussed.

2.6.1 Problem Statement

We consider two observation perspectives: from the ego vehicle or from traffic surveillance systems. For the ego vehicle, the surrounding environment information is provided by onboard sensors which covers a certain range. It aims at tracking surrounding objects as well as forecasting their future behaviors. For surveillance systems, the traffic situations can be obtained by camera based monitors. Unlike the setup in numerical case, the number of prediction targets around the ego vehicle or within the monitor area may fluctuate as time goes by. We make a reasonable simplification for situation representation with a group of

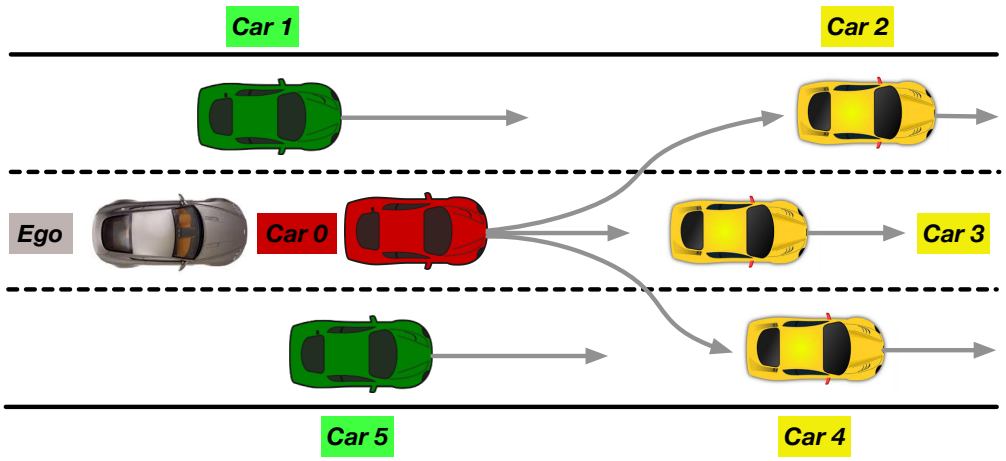


Figure 2.11: A simplified representation of highway scenario. The gray car represents the ego autonomous vehicle with onboard sensors detecting its surrounding objects and the red car is object of study which may have interactions with the green ones and be affected by the motions of leading yellow ones.

six cars which is shown in Figure 2.11, where we assume that only the red car can make a left lane change (LCL) or right lane change (LCR) while surrounding cars maintain the lane keeping behavior. This is a reasonable assumption since it is rare in realistic driving scenarios that two or more vehicles change lane simultaneously under this representation. Our goal is to make probabilistic predictions for their future behaviors.

2.6.2 Data Source and Pre-processing

The Next Generation Simulation (NGSIM) dataset is employed as the data source for extracting training, validation and test trajectories of vehicles, which can be found on [23]. The original dataset provides the estimated vehicle position, velocity, acceleration and other environment information extracted by image processing techniques from videos recording the traffic flow on a approximately 640 meters highway in California, USA. However, in some cases there is large detection noise and error especially on velocity and acceleration information which leads to unsmooth or unfeasible motions as indicated in [22]. Therefore, we applied an Extended Kalman Filter (EKF) to smooth and calibrate vehicle trajectories before experiments. We randomly selected 1,000 lane keeping cases and 200 lane change cases and split them into 70% as training data, 10% as validation data and 20% as test data.

2.6.3 Vehicle Motion Models

In this work we investigate and compare two types of vehicle motion models: pure kinematic model and proposed hierarchical time-series model.

Table 2.5: The Semantic Labels of DHMM (Vehicle Behavior Prediction)

Index	**Behavior**	**Stage**
HMM-1-1	Lane Keeping (LK)	Car following
HMM-1-2	Lane Change Left (LCL)	Preparation
HMM-1-3	Lane Change Left (LCL)	Deviation
HMM-1-4	Lane Change Left (LCL)	Adjustment
HMM-1-5	Lane Change Right (LCR)	Preparation
HMM-1-6	Lane Change Right (LCR)	Deviation
HMM-1-7	Lane Change Left (LCL)	Adjustment
HMM-2-1	Lane Keeping (LK)	—
HMM-2-2	Lane Change Left (LCL)	—
HMM-2-3	Lane Change Right (LCR)	—

Pure Kinematics Models

In most of the vehicle tracking and motion prediction literature, kinematic models are naturally employed as the state transition model to propagate particle hypotheses. A comparison study of various motion models and their state transition equations are presented in [156]. The simplest models are *constant velocity model* (CVM) and *constant acceleration model* (CAM) which are linear models treating all 2D motions as translations in both longitudinal and lateral directions without considering rotations. More complicated models such as bicycle models also consider the yaw rate. However, the yaw rate can be assumed to be zero, which is reasonable in highway scenarios due to small yaw angle variations.

Hierarchical Time-Series Prediction Model

There are three high-level behaviors in the studied scenario. The recognition module is a two-layer DHMM whose semantic labels are introduced in Table IV. *Lane Keeping (LK)* behavior only consists of *Car Following* stage; *Lane Change Left (LCL)* and *Lane Change Right (LCR)* behaviors both consist of *Preparation*, *Deviation* and *Adjustment* stages successively. All of the six vehicles are taken into account to obtain the distribution of red car's future high-level behaviors.

The evolution module consists of three independent models corresponding to three behaviors respectively which are trained separately. Each behavior model forecasts motions of a portion of the six entities according to relevance. Specifically, the *LK* model considers *Car 0* and *Car 3* while the *LCL* model and *LCR* model consider *Car 0, Car 1, Car 2* and *Car 0, Car 4, Car 5*, respectively. The recognition module determines the proportion of sampled trajectories by each model in the evolution module.

Behavior-unconditional Learning Based Models

Behavior-unconditional models treat all the entities as a whole system and predict the joint distribution of their motions. The objective is thus to capture the multi-modality of data distribution raised by multiple behavior patterns, which is much harder to achieve due to the demand for large representation capacity. To allow for a fair comparison, these models should have more complicated architectures than the evolution module of HTSPM.

2.6.4 Experiments Details and Results

For the recognition module of HTSPM, we decided the number of hidden states of each HMM according to BIC score. Since the Baum-Welch algorithm can reach different solutions with diverse initialization, we trained each HMM multiple times and selected the model with highest final BIC score. For the evolution module, the CGMR has 10 mixture components; the CP-MLP consists of four hidden fully-connected layers with 64 units followed by a leaky ReLU activation function; the CP-LSTM has the same architecture as CP-MLP except that the first fully-connected layer is replaced with a LSTM layer. For the behavior-unconditional models we increased the model capacity. The GGMR has 30 mixture components; the P-MLP and P-LSTM also consist of four hidden layers but with 192 units. We trained all the models multiple times and selected the ones with smallest prediction error on the validation set. Moreover, we chose CAM from vehicle kinematic models as a baseline.

We adopted a unified input feature representation for models of the same type. Specifically, for behavior-conditional models (i.e. *CGMR, CP-MLP* and *CP-LSTM*) the feature contains a sequence of historical relative positions of only model-related surrounding vehicles with respect to the middle red vehicle $\{x_i(k-T:k), y_i(k-T:k), i=0,...,5\}$ as well as their absolute advancing velocities $\{v_i(k-T:k-1), i=0,...,5\}$, where T is history horizon; while for behavior-unconditional models (i.e. *GGMR, GP-MLP* and *GP-LSTM*), the feature covers the same information of all the six vehicles.

Quantitative Analysis

We sampled 100 particle hypotheses for each entity to make predictions in all the experiments. Table 2.6 provides the ADE value comparisons of vehicle position prediction using both HTSPM and baseline models. It is shown that employing proposed HTSPM can achieve the lowest prediction error. The *DHMM+CP-LSTM* has superiority over the others in most time steps, which implies that recurrent neural network is more capable of learning long-term dependencies. Although the *CAM* can achieve acceptable performance in the first second, the error increases greatly as prediction horizon expands, which indicates pure kinematic models are only suitable for short-term predictions.

Table 2.6: ADE Value Comparisons of Vehicle Position Prediction

Cases	Prediction Horizon (s)	DHMM + CGMR (m)	DHMM + CP-MLP (m)	DHMM + CP-LSTM (m)	GGMR (m)	GP-MLP (m)	GP-LSTM (m)	CAM (m)
Lane Keeping (LK)	1.0	**0.12**/**0.11**	0.21/0.20	0.18/0.21	0.19/0.25	0.33/0.30	0.32/0.31	0.41/0.39
	2.0	0.93/0.99	**0.68**/0.65	0.72/**0.62**	1.01/0.97	0.80/0.74	0.76/0.74	1.91/1.33
	3.0	2.21/1.67	1.14/**1.36**	**1.13**/1.38	2.33/1.69	1.26/1.48	1.29/1.42	3.36/2.90
	4.0	3.37/2.29	1.69/**2.20**	**1.65**/2.31	3.68/2.34	1.81/2.32	1.78/2.29	4.65/3.63
	5.0	4.05/3.83	2.91/3.33	**2.90**/**3.04**	4.41/3.88	3.03/3.45	3.14/3.28	5.88/5.02
Lane Change Left (LCL)	1.0	0.36/**0.13**	0.31/0.22	**0.29**/0.24	0.42/0.28	0.41/0.34	0.38/0.35	0.51/0.42
	2.0	0.99/1.06	0.96/0.87	**0.91**/**0.83**	1.15/1.04	1.11/0.99	1.14/0.98	2.39/1.51
	3.0	1.98/1.79	1.79/1.82	**1.69**/**1.57**	2.11/1.93	1.91/1.94	1.83/1.90	3.73/5.64
	4.0	2.89/2.86	**2.61**/**2.41**	2.91/2.69	3.26/2.94	2.73/2.98	2.71/2.98	6.65/3.77
	5.0	4.63/4.23	3.82/3.78	**3.74**/**3.58**	5.37/4.11	4.04/3.90	3.95/3.73	7.68/5.36
Lane Change Right (LCR)	1.0	0.29/0.23	0.27/**0.18**	**0.24**/0.25	0.49/0.45	0.38/0.35	0.35/0.37	0.59/0.46
	2.0	**0.83**/1.12	1.02/0.86	0.97/**0.81**	0.91/1.08	1.14/0.97	1.09/0.92	2.23/1.49
	3.0	1.94/1.66	1.73/1.84	**1.68**/**1.64**	2.15/1.68	1.83/1.95	1.79/1.98	3.58/2.71
	4.0	3.01/2.52	2.58/**2.41**	**2.44**/2.56	3.28/2.82	2.74/2.67	2.78/2.70	6.23/3.68
	5.0	4.58/3.92	3.76/**3.61**	**3.71**/3.82	5.05/4.31	3.91/3.82	3.84/3.79	7.21/5.54

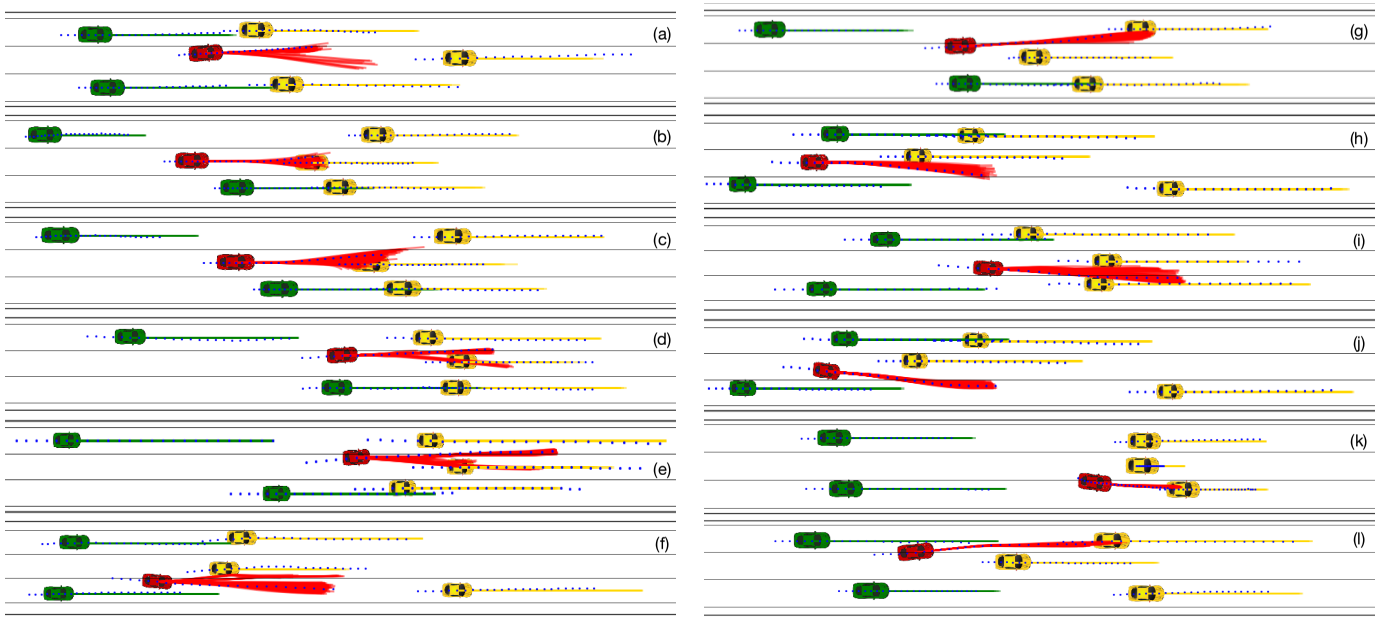


Figure 2.12: Visualization of prediction results of selected typical cases of LCL and LCR behaviors using the *DHMM+CP-LSTM* model. The colored solid lines are sampled future trajectory hypotheses and the blue scatters are ground truth trajectories.

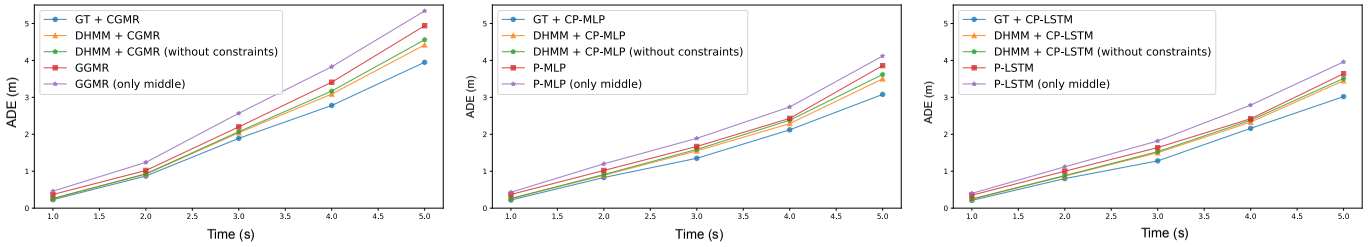


Figure 2.13: The ablative analysis of prediction accuracy of middle vehicle position in terms of Average Distance Error (ADE) using different learning-based state evolution models with different recognition results.

Qualitative Analysis

The prediction results of several test cases are visualized in Figure 2.12 to demonstrate the model performance. In the first column of figures, it can be seen that our model is able to make multi-modal trajectory predictions considering uncertainty both on the behavior and motion level. In Figure 2.12(a), the red car is predicted to make a left lane change according to its heading tendency. However, due to the small gap between the green and yellow cars on the target lane and their relative velocities, it is also possible and reasonable for the red car to change its mind to continue the lane keeping behavior, which is also captured by the proposed model. Figure 2.12(b) and Figure 2.12(c) show two chronological time steps in the same test case where at first the red car may choose all three possible behaviors while after a moment the probability of LCL increases due to the large gap on its left, small gap on its right and low relative velocity of its leading car, which demonstrates the online evolution of prediction results. The second column of figures mainly show the later stage of lane change where only one of the three behaviors dominates the future trajectories, where the variance of samples is still maintained to present different driving patterns.

Ablative Analysis

We also conducted an ablative analysis to demonstrate the relative importance of recognition and evolution module through comparing the prediction errors under four model settings:

- **GT + Behavior-conditional Model**: We directly use the state evolution model corresponding to the ground truth behavior. This can be treated as an upper limit of prediction performance of evolution module.
- **DHMM + Behavior-conditional Model**: This is just to use the complete proposed HTSPM with state constraints.
- **DHMM + Behavior-conditional Model (no constraints)**: This is to use the proposed HTSPM but without considering constraints on the vehicle state.
- **Behavior-unconditional Model**: This is the learning-based baseline model without classification on behaviors.
- **Behavior-unconditional Model (only middle car)**: This model does not consider the surrounding vehicles and make predictions for the middle car with only its historical trajectories, which is used to illustrate the significance of considering interactions among entities.

Figure 2.13 shows the ADE values of prediction for the above model settings. We find that models considering adjacent vehicles outperforms those only focusing on the middle vehicle, implying that the motions of surrounding cars have significant influence on the target vehicle. The *DHMM+Behavior-conditional Model* leads to further improvement, suggesting the effectiveness of behavior recognition prior to motion forecasting. Both factors become

more remarkable as prediction horizon extends. It is also shown that incorporating kinematic constraints on vehicle state can achieve better prediction accuracy than otherwise despite that the improvement is not significant, which indicates that there are not many violations of constraints in the output action of proposed models. Moreover, from the performance of ground truth behavior model we observe some space for improvement if the recognition module becomes more powerful.

2.7 Chapter Summary

In this chapter, a generic learning-based hierarchical time-series prediction model (HTSPM) was put forward and applied to a numerical case study and real-world on-road vehicle behavior prediction tasks in highway scenarios. The results show that the DHMM in recognition module of HTSPM can better capture the behavior distribution than other probabilistic classifiers in terms of response time and robustness. Multiple state evolution models including learning-based ones and pure kinematics-based ones were compared under the framework settings. An ablative analysis was also conducted to demonstrate the significance of the recognition module. Future research directions include enhancing the capability of both recognition module and evolution module and applying the proposed framework to more complicated scenarios with more interactions and mutual reactions such as roundabout and unsignalized intersections.

Chapter 3

CGNS: Conditional Generative Neural System

3.1 Introduction

It is desired for a multi-agent prediction system to satisfy the following requirements to generate diverse, realistic future trajectories.

- **Context-aware**: The system should be able to forecast trajectories which are inside the traversable regions and collision-free with static obstacles in the environment. For instance, when the vehicles navigate in a roundabout (see Figure 3.1(a)) they need to advance along the curves and avoid collisions with road boundaries.
- **Interaction-aware**: The system needs to generate reasonable trajectories compliant to traffic or social rules, which takes into account interactions and reactions among multiple entities. For instance, when the vehicles approach an unsignalized intersection (see Figure 3.1(b)), they need to anticipate others' possible intentions and motions as well as the influences of their own behaviors on surrounding entities.
- **Feasibility-aware**: The system should anticipate naturalistic and physically-feasible trajectories which are compliant to vehicle kinematics or dynamics constraints, although these constraints can be ignored for pedestrians due to the large flexibility of their motions.
- **Probabilistic prediction**: Since the future is full of uncertainty, the system should be able to learn an approximated distribution of future trajectories close to data distribution and generate diverse samples which represent various possible behavior patterns.

In this work, we propose a generative neural system that satisfies all the aforementioned requirements for predicting trajectories in highly interactive scenarios. The system takes

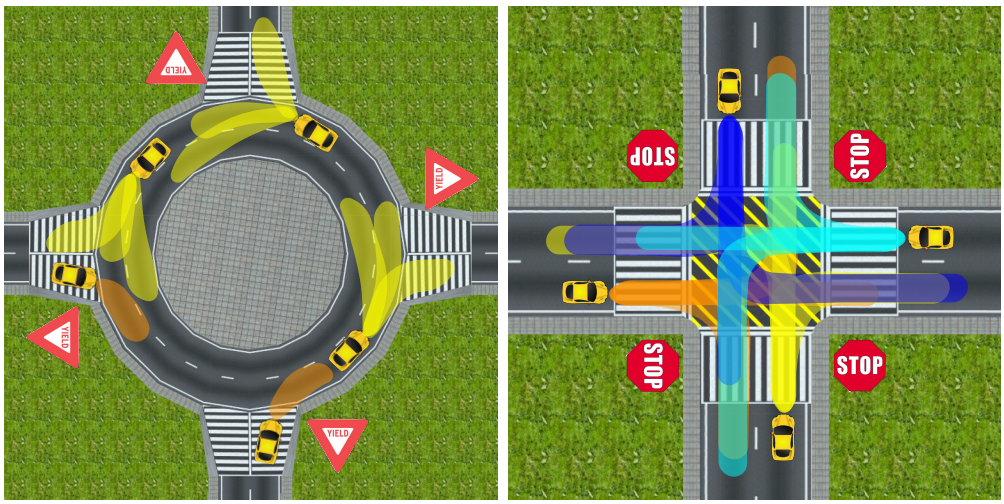


Figure 3.1: Typical urban traffic scenarios with large uncertainty and interactions among multiple entities. The shaded areas represent the reachable sets of possible trajectories. (a) Unsignalized roundabout with four-way yield signs; (b) Unsignalized intersection with four-way stop signs.

advantage of both explicit and implicit density learning in a unified generative system to predict the distributions of trajectories for multiple interactive agents, from which the sampled hypotheses are not only reasonable and feasible but also cover diverse possible motion patterns.

The main contributions of this work are summarized as follows:

- A Conditional Generative Neural System (CGNS) is proposed to jointly predict future trajectories of multiple highly-interactive agents, which takes into account the static context information, interactions among multiple entities and feasibility constraints.
- A block attention mechanism and a Gaussian mixture attention mask are proposed and applied to historical trajectories and scene image sequences respectively, which are computationally efficient.
- An effective strategy for soft constraint incorporation into deep neural networks is presented.
- The latent space learning and variational divergence minimization approaches are integrated into a unified framework in a novel fashion, which combines their strengths on distribution learning.

- The proposed CGNS is validated on multiple pedestrian trajectory forecasting benchmarks and is used to solve a task of anticipating motions of on-road vehicles navigating in highly-interactive scenarios.

3.2 Related Work

In this section, we provide a brief overview on related research and illustrate the distinction and advantages of the proposed generative system.

3.2.1 Trajectory and Sequence Prediction

Many research efforts have been devoted to predict behaviors and trajectories of pedestrians and on-road vehicles. Many classical approaches were employed to make time-series prediction, such as variants of Kalman filter based on system process models, time-series analysis and auto-regressive models. However, such methods only suffice for short-term prediction in simple scenarios where interactions among entities can be ignored. More advanced learning-based models have been proposed to cope with more complicated scenarios, such as hidden Markov models [99, 206], Gaussian mixture regression [96, 99], Gaussian process, dynamic Bayesian networks, and rapidly-exploring random tree. However, these approaches are nontrivial to handle high-dimensional data and require hand-designed input features, which confines the flexibility of representation learning. Moreover, these methods only predict behaviors for a certain entity. A few works also took advantage of both recurrent neural networks [3, 73] and generative modeling to learn an explicit or implicit trajectory distribution, which achieved better performance [97, 120, 95]. However, they either leveraged only static context images or only trajectories of agents, which is not sufficient to make predictions for the agents that interact with both static and dynamic obstacles. In this paper, we propose a conditional generative neural system which can leverage both historical scene evolution information and trajectories of multiple interactive agents and generate realistic and diverse trajectory hypotheses.

3.2.2 Soft Attention Mechanisms

Soft attention mechanisms have been widely used in neural networks to enable the capability of focusing on a subset of input features, which have been extensively studied in the field of image captioning [195], visual object tracking [81] and natural language processing. Several works also brought attention mechanisms into trajectory prediction tasks to figure out the most informative and related obstacles [151, 150, 179, 15]. In this paper, we put forward a block attention mask mechanism for trajectories to extract the most critical features of each entity as well as a Gaussian mixture attention mechanism for context images to extract the most crucial static features.

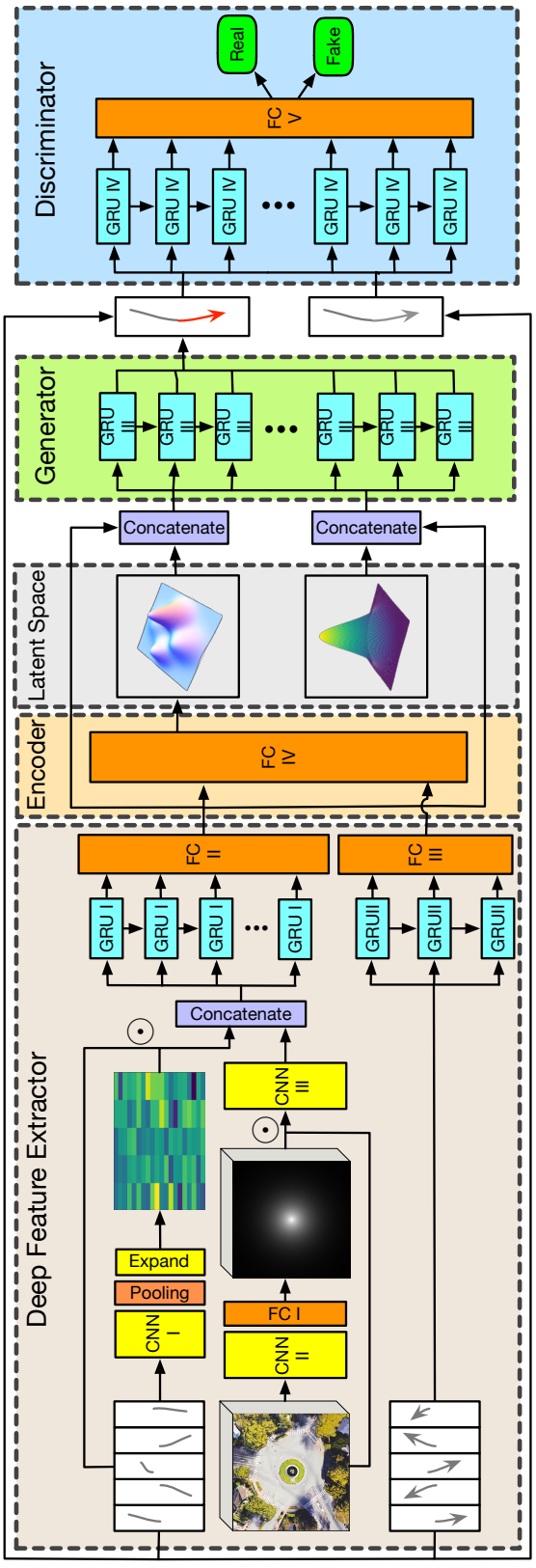


Figure 3.2: The overview of proposed conditional generative neural system (CGNS), which consists of four key components: (a) A deep feature extractor with soft attention mechanism, which extracts multi-level features from scene context image sequences and trajectories; (b) An encoder to learn conditional latent space representations; (c) A generator (decoder) to sample future trajectory hypotheses; (d) A discriminator to distinguish predicted trajectories from groundtruth.

3.2.3 Deep Bayesian Generative Modeling

The objective of generative models is to approximate the true data distribution, with which one can generate new samples similar to real data points with a proper variance. Generative models have been widely employed in tasks of representation learning and distribution approximation in literature, which basically fall into two categories: explicit density models and implicit density models. In recent years, since deep neural networks have been leveraged as universal distribution approximators thanks to its high flexibility, two deep generative models have been widely studied: Variational Auto-Encoder (VAE) [75] and Generative Adversarial Network (GAN) [42]. Since in trajectory forecasting tasks the predicted trajectories are sampled from the posterior distribution conditioned on historical information, the two models were extended to their conditional versions which results in conditional VAE (CVAE) [89] and conditional GAN (CGAN) [130, 151]. In this paper, we combine the strengths of conditional latent space learning via CVAE and variational divergence minimization via adversarial training.

3.3 Problem Formulation

The objective of this paper is to develop a deep generative system that can accurately forecast motions and trajectories for multiple agents simultaneously. The system should take into account the historical state information, static context and interactions among dynamic entities.

Assume there are in total N entities in the observation area, which may vary in different cases. We denote a set of trajectories covering the history and prediction horizons (T_h and T_f) as

$$\mathbf{T}_{k-T_h:k+T_f} = \{\mathbf{t}^i_{k-T_h:k+T_f} | \mathbf{t}^i_k = (x^i_k, y^i_k), i = 1, ..., N\} \tag{3.1}$$

where (x, y) is the 2D coordinate in the pixel space or world space. The latent random variable is denoted as $\mathbf{z}_k$, where k is the current time step. The sequence of context images up to time step k is denoted as $\mathbf{I}_{k-T_h:k}$. Our goal is to predict the conditional distribution of future trajectories given the historical context images and trajectories $p(\mathbf{T}_{k+1:k+T_f} | \mathbf{T}_{k-T_h:k}, \mathbf{I}_{k-T_h:k})$. The long-term prediction is realized by propagating the generative system multiple times to the future. To simplify the notations in the following sections, we denote the condition variable as $C = \{\mathbf{T}_{k-T_h:k}, \mathbf{I}_{k-T_h:k}\}$, the sequence of predicted variables as $Y = \{\mathbf{T}_{k+1:k+T_f}\}$.

3.4 Model Design: CGNS

In this section, we first provide an overview of the key components and the architecture of the proposed Conditional Generative Neural System (CGNS). The detailed theories and models of each component are then illustrated.

3.4.1 System Overview

The architecture of CGNS is shown in Figure 3.2 where there is a deep feature extractor (DFE) with an environment attention mechanism (EAM) as well as a generative neural sampler (GNS). First, the DFE extracts deep features from a sequence of historical context images and trajectories of multiple interactive agents to obtain both the information of static and dynamic obstacles, where the EAM tells which areas and dynamic entities should be paid more attention to than others when predicting the trajectory of a certain entity. The above information is utilized as the input of GNS which takes advantage of a deep latent variable model and a variational divergence minimization approach to generate a set of feasible, realistic and diverse future trajectories of all the involved entities. All the components are implemented with deep neural networks thus can be trained end-to-end efficiently and consistently.

3.4.2 Environment-Aware Deep Feature Extraction

We take advantage of both context images and historical trajectories of interactive agents to extract deep features of both static and dynamic environments. In order to figure out the most crucial parts to consider when forecasting behaviors of certain agents, we propose a soft block attention mechanism applied to trajectories and a Gaussian mixture attention mechanism applied to context images. The details are illustrated below.

The historical and future trajectories are constructed as matrices which are treated as 2D images. The former is fed into a convolutional neural network (CNN) and an average pooling layer to obtain a contractable attention mask over the whole trajectory matrix, which is then expanded to the same size as the trajectory matrix by duplicating each column twice corresponding to coordinates x and y. The original trajectory matrix is multiplied by the block attention mask elementwisely. This mechanism is not applied to the future trajectory matrix since it is unreasonable to have particular attention on the future evolution. The context image sequences are also fed into a CNN followed by fully connected layers to obtain a set of parameters of the Gaussian mixture distribution, which is used to calculate the context attention mask. The elementwise multiplication of original images and attention masks is fed to a pre-trained feature extractor, which is the convolution base of VGG-19 [164] in this paper. The interaction-aware features and context-aware features are concatenated and fed into a recurrent layer followed by fully connected layers to obtain a comprehensive and consistent feature embedding.

3.4.3 Deep Generative Sampling

The GNS is composed of an encoder E and a generator G. The goal of encoder is to learn a consistent distribution in a lower-dimensional latent space, from which the latent variable can be sampled efficiently. The generator aims to produce trajectories as real as possible. An auxiliary discriminator D is adopted, which aims to distinguish fake trajectories from

groundtruth. The generator G and discriminator D formulates a minimax game. The three components can be optimized jointly via conditional latent space learning and variational divergence minimization.

Conditional Latent Space Learning (CLSL)

The conditional latent variable model defined in this paper contains three classes of variables: condition variable C, predicted variable Y and latent variable z. We aim to obtain the conditional distribution $p(Y|C)$. Given the training data (C, Y), the model first samples z from an arbitrary distribution Q. Our goal is to maximize the variational lower bound, which is written as

$$\begin{aligned} &\log p(Y|C) - \mathcal{D}_{KL}[Q(z|C,Y)||p(z|C,Y)] = \\ &\mathbb{E}_{z\sim Q}[\log p(Y|z,C)] - \mathcal{D}_{KL}[Q(z|C,Y)||p(z|C)]. \end{aligned} \tag{3.2}$$

where $p(z|C) = \mathcal{N}(0, I)$. This process can be realized with a Conditional Variational Auto-Encoder which consists of an encoder network E to obtain $Q(z|C, Y)$ and a decoder (generator) network G to model $p(Y|z, C)$. The loss function can be formulated as a weighted sum of the reconstruction error and KL divergence:

$$\mathcal{L}_{\text{RC}}^{G,E} = \mathop{\mathbb{E}}_{\mathbf{t}_{k+1:k+T_f}, \mathbf{z}_k \sim Q} \left[\|\mathbf{t}_{k+1:k+T_f} - G(C_k, \mathbf{z}_k)\|^2 \right], \tag{3.3}$$

$$\mathcal{L}_{\text{KL}}^{E} = \mathop{\mathbb{E}}_{\mathbf{t}_{k+1:k+T_f}} \left[\mathcal{D}_{\text{KL}}(E(C_k)||p(\mathbf{z}_k)) \right], \tag{3.4}$$

where $\mathbf{z}_k \sim \mathcal{N}(0, I)$. The optimal encoder and generator can be obtained by

$$G^*, E^* = \arg\min_{G,E} \lambda_1 \mathcal{L}_{RC}^{G,E} + \lambda_2 \mathcal{L}_{\text{KL}}^{E}. \tag{3.5}$$

Variational Divergence Minimization (VDM)

Given two conditional distributions $P_{\text{data}}(Y|C)$ and $P_{\text{GNS}}(Y|C)$ with absolutely continuous density function $p_{\text{data}}(Y|C)$ and $p_{\text{GNS}}(Y|C)$ which denotes the real data distribution and its approximation with GNS, the f-divergence [133] is defined as

$$\mathcal{D}_f(P_{\text{data}} \mid\mid P_{\text{GNS}}) = \int_{\mathcal{Y}} p_{\text{GNS}}(Y|C) f\left(\frac{p_{\text{data}}(Y|C)}{p_{\text{GNS}}(Y|C)}\right) \mathrm{d}Y, \tag{3.6}$$

where $f : \mathbb{R}_+ \to \mathbb{R}$ is a convex and lower-semicontinuous function with $f(1) = 0$. A lower bound of f-divergence can be derived with the convex conjugate function f^*

$$\begin{aligned} &\mathcal{D}_f(P_{\text{data}} \mid\mid P_{\text{GNS}}) \\ &\geq \sup_{T\in\mathcal{T}} \left(\int_{\mathcal{Y}} p_{\text{data}}(Y|C) T(Y|C)) \mathrm{d}Y - \int_{\mathcal{Y}} p_{\text{GNS}}(Y|C) f^*(T(Y|C)) \mathrm{d}Y \right) \\ &= \sup_{T\in\mathcal{T}} (\mathbb{E}_{Y\sim P_{\text{data}}}[T(Y|C)] - \mathbb{E}_{Y\sim P_{\text{GNS}}}[f^*(T(Y|C))]), \end{aligned} \tag{3.7}$$

where $\mathcal{T}$ is an arbitrary class of mapping $T : \mathcal{Y} \to \mathbb{R}$. In order to minimize the variational lower bound in (3.7), we can formulate a minimax game of $p_{\text{GNS}}(Y|C)$ and $T(Y|C)$, which are parameterized by θ and ϕ, respectively. Then the optimal θ^* and ϕ^* can be obtained by

$$\begin{aligned}\theta^*, \phi^* = \arg\min_{\theta}\max_{\phi} \ & \mathbb{E}_{Y\sim p_{\text{data}}(Y|C)}[T_\phi(Y|C)] \\ & - \mathbb{E}_{Y\sim p_\theta(Y|C)}[f^*(T_\phi(Y|C))].\end{aligned} \tag{3.8}$$

In this work, we propose to minimize the Pearson-χ^2 divergence between $P_{\text{data}}+P_{\text{GNS}}$ and $2P_{\text{GNS}}$

$$\mathcal{D}_{\chi^2_{\text{Pearson}}} = \int_{\mathcal{Y}} \frac{\left(2p_{\text{GNS}} - (p_{\text{data}} + p_{\text{GNS}})\right)^2}{p_{\text{data}} + p_{\text{GNS}}} \mathrm{d}Y. \tag{3.9}$$

Since (3.9) is intractable, we leverage the adversarial learning techniques with a generator G and a discriminator D implemented as deep networks. The adversarial loss functions are derived as

$$\mathcal{L}^G_{\text{VDM}} = \frac{1}{2}\mathbb{E}_{z_k\sim p(z)}[(D(G(C_k, z_k)))^2], \tag{3.10}$$

$$\begin{aligned}\mathcal{L}^D_{\text{VDM}} =& \frac{1}{2}\mathbb{E}_{\mathbf{t}_{k+1:k+T_f}}[(D(\mathbf{t}_{k+1:k+T_f}) - 1)^2] \\ &+ \frac{1}{2}\mathbb{E}_{z_k\sim p(z)}[(D(G(C_k, z_k)) + 1)^2],\end{aligned} \tag{3.11}$$

To discriminate the effect of latent space learning, we also involve two additional terms $\mathcal{L}^{G,E}_{\text{VDM}}$ and $\mathcal{L}^{D,E}_{\text{VDM}}$ where the input z_k are sampled from the encoded latent distribution. Thus, the optimal encoder, generator and discriminator by variational divergence minimization can be obtained as

$$E^*, G^*, D^* = \arg\min_{G,E}\max_{D} \lambda_3(\mathcal{L}^G_{\text{VDM}} + \mathcal{L}^D_{\text{VDM}}) + \lambda_4(\mathcal{L}^{G,E}_{\text{VDM}} + \mathcal{L}^{D,E}_{\text{VDM}}). \tag{3.12}$$

3.4.4 Soft Constraint Incorporation

In order to make generated samples compliant to feasibility constraints of vehicle kinematics, we propose to incorporate a differentiable barrier (indicator) function $I(\cdot)$ in the loss function, which enables soft constraints in deep neural networks via pushing predicted trajectories to the feasible regions. In this work, we denote the empirical upper bounds on the absolute values of accelerations $\mathbf{a}_{k+1:k+T_f}$ and path curvatures $\kappa_{k+1:k+T_f}$ as $a_{\max}$ and $\kappa_{\max}$, respectively. Then the feasibility loss can be calculated as

$$\begin{aligned}\mathcal{L}^{G,E}_{\text{F}} = \ & \alpha_1\mathbb{E}_{\mathbf{a}_{k+1:k+T_f}}\left[\sum_{t=k+1}^{k+T_f}\max\left(0, \text{sgn}(|\mathbf{a}_t| - \mathbf{a}_{\max})\right)\right] \\ + \ & \alpha_2\mathbb{E}_{\kappa_{k+1:k+T_f}}\left[\sum_{t=k+1}^{k+T_f}\max\left(0, \text{sgn}(|\kappa_t| - \kappa_{\max})\right)\right],\end{aligned} \tag{3.13}$$

where sgn(·) refers to the sign function and $\mathbf{a}_t$,κ_t can be calculated with the predicted waypoints. This loss term is not applied to human trajectory prediction.

3.4.5 Conditional Generative Neural System (CGNS)

We leverage both CLSL and VDM in the proposed system, which provides complementary strengths. The objective function of the whole system is formulated as

$$\mathcal{L}_{\text{CGNS}} = \lambda_1 \mathcal{L}_{\text{RC}}^{G,E} + \lambda_2 \mathcal{L}_{\text{KL}}^{E} + \lambda_3 (\mathcal{L}_{\text{VDM}}^{G} + \mathcal{L}_{\text{VDM}}^{D}) + \lambda_4 (\mathcal{L}_{\text{VDM}}^{G,E} + \mathcal{L}_{\text{VDM}}^{D,E}) + \lambda_5 \mathcal{L}_{\text{F}}^{G,E}, \quad (3.14)$$

which can be trained end-to-end. In practice, due to the existence of reconstruction loss, the generator tends to improve faster than the discriminator, which may result in unbalanced training. Therefore, we compensate the unbalance by training the discriminator multiple times in each iteration.

3.5 Experiments

In this section, we validate the proposed CGNS on three benchmark datasets for trajectory prediction which are available online and solve a task of probabilistic behavior prediction for multiple interactive on-road vehicles in a roundabout scenario. The model performance is compared with several state-of-the-art baselines.

3.5.1 Datasets

- **ETH** [142] **and UCY** [87]: These datasets include bird-eye-view videos and image annotations of pedestrians in various outdoor and indoor scenarios. The trajectories were extracted in the world space.
- **Stanford Drone Dataset (SDD)** [148]: The dataset also contains a set of bird-eye-view videos and the corresponding trajectories of involved entities, which was collected in multiple scenarios within a university campus full of pedestrians, bikers and vehicles. The trajectories were extracted in the pixel space instead of the world space.
- **INTERACTION Dataset (ID)** [209]: The raw dataset was collected by a drone with camera and our testing vehicle equipped with LiDAR. The trajectories were extracted by visual detection. We visualized the real trajectories in our simulator to obtain the bird-eye-view images, where the static context information came from the Google Earth.

3.5.2 Evaluation Metrics and Baselines

We evaluate the model performance in terms of average displacement error (ADE) defined as the average distance between the predicted trajectories and the ground truth over all the

involved entities within the prediction horizon, as well as final displacement error (FDE) defined as the distance at the last predicted time step. To allow for fair comparisons with prior works, we predicted the future 12 time steps (4.8s) based on the previous 8 time steps (3.2s) for ETH and UCY in the Euclidean space. We used the standard training and testing split for SDD and make predictions in the pixel space. For our own dataset ID, we predicted the future 10 time steps (5s) based on the historical 4 time steps (2s) in the Euclidean space.

We compared the performance of our proposed system with the following baseline approaches on multiple datasets: *Constant Velocity Model (CVM)*, *Linear Regression (LR)*, *Probabilistic LSTM (P-LSTM)*, *Social LSTM (S-LSTM)* [3], *Social GAN (S-GAN and S-GAN-P)* [45], *Clairvoyant attentive recurrent network (CAR-Net)* [150], *SoPhie* [151] and *DESIRE* [89].

3.5.3 Implementation Details

Since the whole system consists of differentiable functions approximated by deep neural networks, it can be trained end-to-end efficiently. The detailed model architecture and hyper-parameters are introduced below.

In the deep feature extractor, the $\mathrm{CNN_I}$ contains one conv-layer with kernel size 5×5 and zero padding to keep the same dimension. The $\mathrm{CNN_{II}}$ contains three conv-layers with kernel size 3×3 and the $\mathrm{FC_I}$ contains two layers with 64 hidden units. The $\mathrm{CNN_{III}}$ is the convolution base of pre-trained VGG-19 whose weights are fixed during training. The $\mathrm{GRU_I}$, $\mathrm{GRU_{II}}$, $\mathrm{FC_{II}}$ and $\mathrm{FC_{III}}$ all have 128 hidden units. The Encoder $\mathrm{FC_{IV}}$ has three fully-connected layers with 256, 128 and 64 hidden units, respectively. The dimension of encoded latent space is two. The Generator $\mathrm{GRU_{III}}$ has 128 hidden units. The Discriminator $\mathrm{GRU_{IV}}$ has 128 hidden units and $\mathrm{FC_V}$ has three layers with 128, 128 and 1 units, respectively. In all the experiments, we set $\lambda_1 = 5.0, \lambda_2 = \lambda_3 = \lambda_4 = \lambda_5 = 1.0$ and $\alpha_1 = \alpha_2 = 1000$. The Adam optimizer was employed with a learning rate of 0.002. Moreover, we found that the Gaussian mixture in the context image attention mask does not lead to obvious improvement in terms of prediction accuracy and diversity than a single Gaussian in this task. Therefore, we utilized the latter to reduce model complexity and show the corresponding results.

3.5.4 Quantitative Analysis

ETH and UCY Dataset

To allow for fair comparisons with multiple baseline approaches which only leverage historical trajectory information, we deactivated the branch of context feature extraction in our system to illustrate its superiority on prediction accuracy based on the same input as prior works. The ADE and FDE of the proposed CGNS and baseline models in Euclidean space are compared in Table 3.5.2. Some of the reported statistics are adapted from the original papers.

It can be seen that the CVM performs the worst as expected since the constant velocity approximation is insufficient for a crowded scenario with highly interactive agents. The LR

Table 3.1: ADE/FDE Comparisons of Pedestrian Trajectory Prediction (ETH and UCY dataset).

	CVM	LR	P-LSTM	S-LSTM	S-GAN	S-GAN-P	SoPhie	**CGNS**
ETH	1.42/2.88	1.33/2.94	1.13/2.38	1.09/2.35	0.81/1.52	0.87/1.62	0.70/1.43	**0.62/1.40**
HOTEL	0.51/0.68	**0.39**/0.72	0.91/1.89	0.79/1.76	0.72/**0.61**	0.67/1.37	0.76/1.67	0.70/0.93
UNIV	0.73/1.63	0.82/1.59	0.63/1.36	0.67/1.40	0.60/1.26	0.76/1.52	0.54/1.24	**0.48/1.22**
ZARA1	0.59/1.36	0.62/1.21	0.44/0.84	0.47/1,00	0.34/0.69	0.35/0.68	**0.30**/0.63	0.32/**0.59**
ZARA2	0.84/1.55	0.77/1.48	0.51/1.16	0.56/1.17	0.42/0.84	0.42/0.84	0.38/0.78	**0.35/0.71**
AVG	0.82/1.62	0.79/1.59	0.72/1.53	0.72/1.54	0.58/1.18	0.61/1.21	0.54/1.15	**0.49/0.97**

Table 3.2: ADE/FDE Comparisons of Pedestrian Trajectory Prediction (SDD dataset).

	LR	P-LSTM	S-LSTM	S-GAN	SoPhie	CAR-Net	DESIRE	**CGNS**
SDD	37.1/63.5	35.8/55.4	31.2/57.0	24.8/38.6	17.8/32.1	25.7/51.8	19.3/34.1	**15.6/28.2**

Table 3.3: ADE/FDE Comparisons of Vehicle Trajectory Prediction (INTERACTION dataset). **T** represents only using interaction-aware (trajectory) features and **T** + **I** represents using additional context-aware (image) features.

	Baseline Models					Proposed CGNS			
	CVM	LR	P-LSTM	S-LSTM	S-GAN	**T** + CLSL	**T** + VDM	**T** + CLSL + VDM	**T** + **I** + CLSL+VDM
1.0s	**0.16**/0.29	0.24/0.32	0.23/0.28	0.24/0.30	0.22/0.28	0.19/**0.23**	0.22/0.27	0.17/0.25	0.21/0.26
2.0s	0.59/0.78	0.58/0.92	0.47/0.60	0.45/0.57	0.42/0.58	**0.34**/0.42	0.38/0.45	0.38/0.44	0.35/**0.40**
3.0s	1.21/1.92	1.43/2.28	0.84/1.53	0.80/1.48	0.81/1.54	0.72/1.33	0.75/1.37	0.69/1.24	**0.64/1.15**
4.0s	2.94/3.98	3.85/4.73	1.27/1.51	1.21/1.69	1.28/1.87	1.26/1.81	1.35/1.76	0.86/1.33	**0.79/1.23**
5.0s	4.28/6.12	5.89/6.91	1.78/2.21	1.69/2.77	1.65/2.68	1.85/3.20	1.72/2.89	1.54/2.37	**1.47/2.12**

performs slightly better in most scenarios than CVM but achieves the smallest error on the HOTEL dataset. A possible reason is that the human trajectories in this dataset tend to be more straight and smooth, which brings an advantage for linear fitting methods. The P-LSTM and S-LSTM provide an improvement with similar accuracy due to the exploitation of recurrent neural networks. The S-GAN and Sophie achieve a bigger progress thanks to the implicit generative modeling of trajectory distribution. Our approach makes a step forward on prediction accuracy, which implies the effectiveness of latent space learning.

Stanford Drone Dataset

We also compared the ADE and FDE of the CGNS and baseline models in pixel space, which is shown in Table 3.1. Similarly, the linear method LR performs the worst and the ordinary P-LSTM and S-LSTM give a slightly better accuracy. The CAR-Net makes a step forward by utilizing a physical attention module. The S-GAN and DESIRE provide better results than the above baselines since they solve the task from a probabilistic perspective by learning implicit data distribution and latent space representations, respectively. Our approach achieves the best performance in terms of prediction error, which implies the significance and necessity of leveraging both context and trajectory information. The combination of CLSL and VDM also contributes to the enhancement.

INTERACTION Dataset

We finally compared the model performance on our roundabout driving dataset in Table 3.2. The SoPhie, CAR-Net and DESIRE are not involved since their codes are not publicly available. It is shown that the linear models CVM and LR have similar performance to advanced learning-based models for short-term prediction since the velocity and yaw angle of vehicles cannot vary much in a short period due to kinematics feasibility constraints. However, as the prediction horizon increases, their performance deteriorates much faster. A potential reason is that due to the curving roads within the roundabout area, the vehicles tend to advance along the curving lines to avoid collisions, which is not able to be captured by linear approximations. The P-LSTM and S-LSTM provide similar results, which implies that the social pooling mechanism has little effects on feature extraction in this scenario. Our CGNS is able to achieve the smallest prediction error among baseline models in most cases especially for long-term prediction.

3.5.5 Qualitative Analysis

We provide a qualitative analysis of the prediction results on our INTERACTION dataset. To illustrate the effectiveness of the attention module, we visualize the context image masks and trajectory block masks of several typical testing cases in Figure 3.3. Detailed analysis can be found in the caption. The distribution of generated future trajectories is approximated by the kernel density estimation, which is visualized in Figure 3.4. We can see that the system

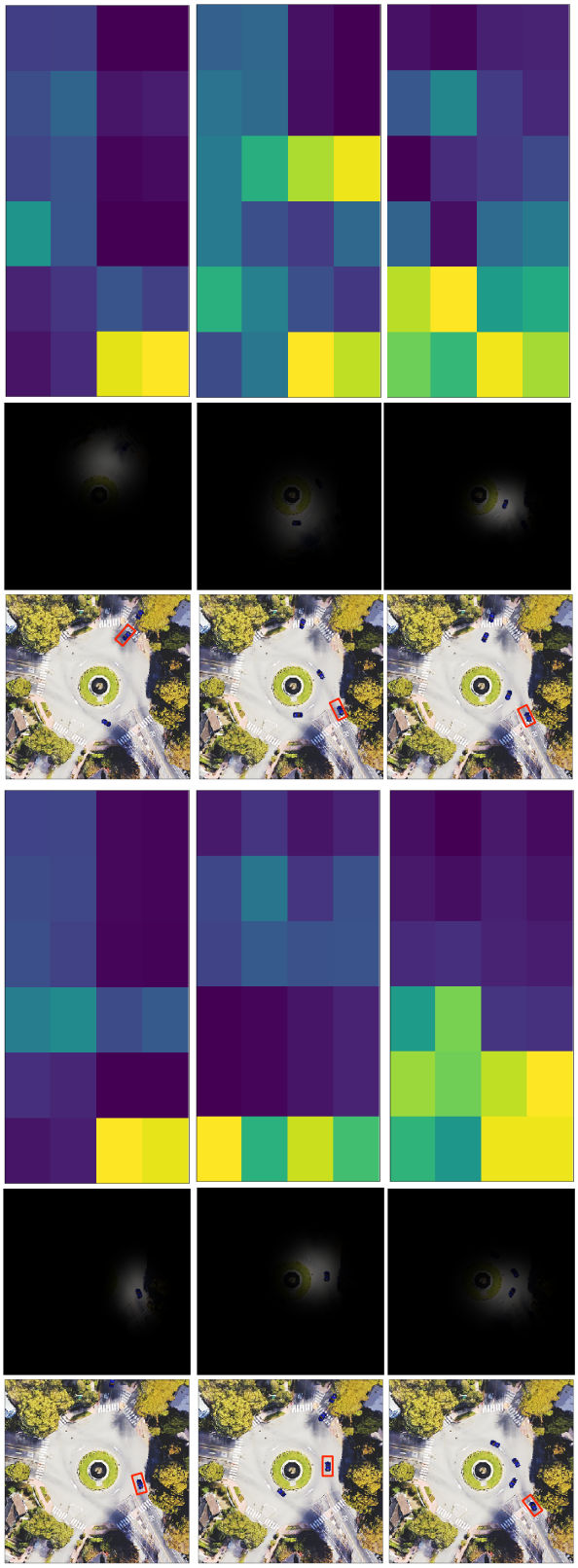


Figure 3.3: The visualization of the context image masks and trajectory block attention masks. Particularly, in the trajectory masks, there are four rows representing four historical time steps and six columns representing six vehicles in the scene. The first column corresponds to the predicted vehicle and the others corresponds to surrounding ones. Brighter colors indicate larger attention weights. The predicted vehicles are indicated with red bounding boxes. In all the cases, the image masks have a large weight around the predicted vehicle and the area of its heading direction. In the first three cases, only the historical trajectories of the predicted vehicle are assigned large attention weights, which implies that the other vehicles have little effect in these situations. However, in the last three cases, more attention is paid to other vehicles since there exist strong interactions which increases the inter-dependency.

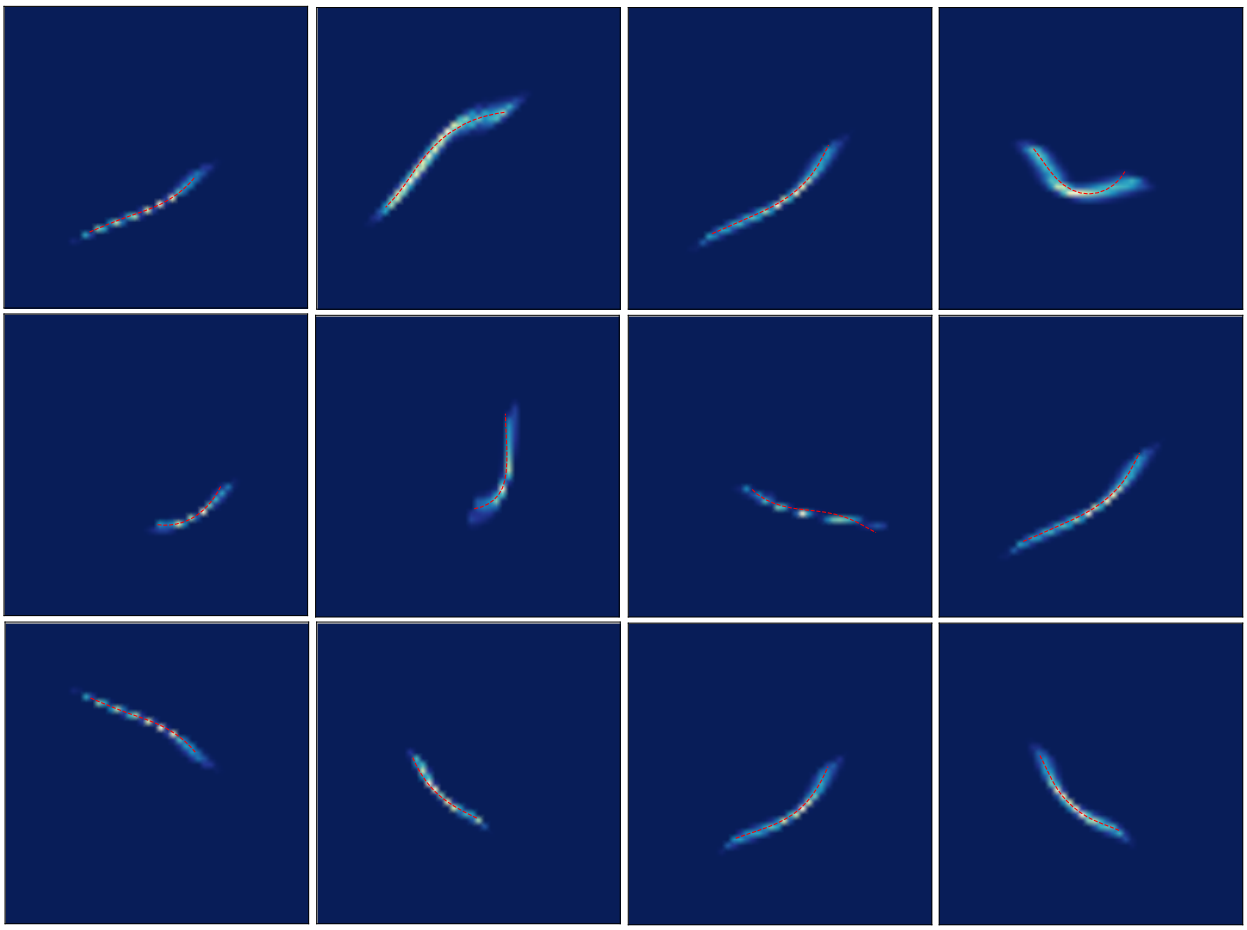


Figure 3.4: The visualization of sampled future trajectory hypotheses with CLSL+VDM training strategies. The red dashed lines denote groundtruth trajectories.

can generate smooth, feasible and realistic vehicle trajectories, which evolve along the road curves. The groundtruth is located at the most dense part of the distribution in most cases. In general, our proposed CGNS can achieve better generation performance in terms of realism and diversity.

3.5.6 Ablative Analysis

We conduct an ablative analysis on the RD dataset to demonstrate relative significance of each component in the proposed CGNS. The ADE and FDE of each model setting are shown in Table III. We notice that using the **T** + CLSL and **T** + VDM achieves similar performance in terms of prediction error while **T** + CLSL + VDM provides a notable improvement.

Moreover, it is demonstrated that the complete system **T** + **I** + CLSL + VDM does not lead to obvious improvement compared with three partial systems for short-term prediction while its superiority becomes more remarkable as the forecasting horizon increases. This is reasonable since the static context has little effect on driver behaviors in a short period. More specifically, since the trajectory segment within a short period can be approximated by a linear segment, learning the road curvature from context images does not provide much assistance for prediction. As the forecasting horizon increases, however, the restriction of road geometry on vehicle motions cannot be ignored any more, which results in larger performance gain of leveraging context information.

3.6 Chapter Summary

In this chapter, we propose a conditional generative neural system for long-term trajectory prediction, which takes into account both static context information through images and dynamic evolution of traffic situations through trajectories of interactive agents. We also incorporate attention mechanisms to figure out the most critical portions for predicting motions of a certain entity. The system combines the strengths of both latent space learning and variational divergence minimization to approximate the data distribution, from which realistic and diverse trajectory hypotheses can be sampled. The proposed system is validated on various benchmark datasets as well as a roundabout driving dataset collected by ourselves. The results show that our system can achieve better performance than various baseline models on most datasets in terms of prediction accuracy.

Chapter 4

STG-DAT: Spatio-Temporal Graph Dual-Attention Network

4.1 Introduction

In order to navigate safely in dense traffic scenarios or crowded areas full of vehicles and pedestrians, it is crucial for autonomous vehicles or mobile robots to forecast and track future behaviors of surrounding interactive agents accurately and efficiently [90]. For short-term prediction, it may be acceptable to use pure physics based methods. However, due to the uncertain nature of future situations, the system for long-term prediction is desired to not only allow for interaction modeling between different agents, but also to figure out traversable regions delimited by road layouts as well as right of way compliant to traffic rules. Figure 4.1 illustrates several traffic scenarios where interaction happens frequently and the drivable areas are heavily defined by road geometries. For instance, at the entrance of roundabouts or unsignalized intersections, the future behavior of an entering vehicle highly depends on whether the conflicting vehicles would yield and leave enough space for it to merge. In addition, for vehicle trajectory prediction, kinematic constraints should be satisfied to make the trajectories feasible and smooth.

There have been extensive studies on the prediction of a single target entity, which consider the influences of its surrounding entities [56, 14, 3, 45]. However, such approaches only care about one-way interactions, ignoring the potential interactions in the opposite way. Recent works have tried to address this issue by simultaneously forecasting for multiple agents [27, 26, 212]. However, most of these methods employ concatenation or social pooling operations to blend the features of different agents without explicit relational reasoning. Moreover, they are not able to model higher-order interactions (indirect influences) beyond adjacent entities. In this work, we take a step forward to model the interactions explicitly with a spatio-temporal graph representation and attention-based message passing rules. Our model enables permutation invariance and mutual effects between pairs of entities.

In the literature of deep learning methods, the soft attention mechanism is widely adopted

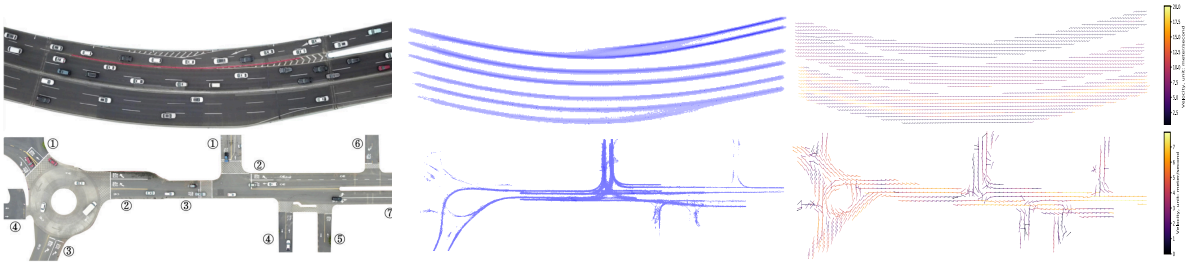


Figure 4.1: Typical traffic scenarios with large uncertainty and interactions among multiple entities. The left column is adopted from [204]. The upper figure in the first column was captured in a highway ramp merging scenario, where lane change behavior with negotiation happens frequently. The lower figure was captured in a roundabout and an unsignalized intersection scenario, where yielding and stopping behaviors happen frequently. The other two columns shows the occupancy density maps and the velocity fields of the scenarios, which are generated based on the training data to provide statistical context information.

in modeling spatial relations due to its flexibility and interpretability. The temporal relations are usually modeled by recurrent neural networks, which naturally attenuate the effects of distant history information. One of the most related work is Social Attention [179]. However, in some cases the earlier information may be also important. Therefore, in order to figure out which of the other agents have the most significant influence on a certain agent, as well as the relative importance of different time steps, we propose a spatio-temporal graph attention mechanism which is applied to both topological and temporal features.

Besides the trajectory prediction task, the proposed approach can also be incorporated into multi-target tracking frameworks, due to its property of permutation invariance and flexibility on agent numbers. More specifically, it can serve as the process (prediction) model in the prior update of recursive Bayesian state estimation. The model has advantages in handling occlusion issues and missing observations, due to its capability of long-term prediction. In this paper, we adopt the multi-target tracking framework proposed in [100] and compare the performance of tracking with our model and other widely used models.

This work is a significant extension of our prior work [120] where we presented a modified Wasserstein generative modeling method. This is adopted as the basis of training the proposed model in this paper. The method in [120] was only able to predict interactive behaviors of two vehicles in a single scenario, while the proposed approach is able to handle multiple, heterogeneous agents in different scenarios simultaneously.

The main contributions of this work are summarized as follows:

- We propose a multi-agent, generative trajectory forecasting system with relational reasoning on heterogeneous, interactive agents. The system is applied to predict pedestrian and vehicle trajectories across different scenarios.

- We propose a spatio-temporal dual-attention mechanism for representation learning on spatio-temporal dynamic graphs, which can figure out relative significance of the information about different surrounding agents at each time step.
- We incorporate an efficient kinematic constraint module similar to [120] to ensure physical feasibility for vehicle trajectory prediction. This constraint layer can not only smooth the trajectories and reduce prediction error, but also enhance the model robustness to noisy data.
- We validate the proposed system on multiple trajectory forecasting benchmark datasets. The approach achieves state-of-the-art prediction accuracy. The model also proves to enhance the multi-target tracking performance.

4.2 Related Work

In this section, we provide a concise literature review on related research and illustrate the distinctions and advantages of the proposed generative trajectory prediction approach, which can also be leveraged by multi-object tracking frameworks to enhance tracking performance.

4.2.1 Interaction-Aware Trajectory Prediction

Extensive research has been conducted on trajectory prediction of humans and autonomous agents (e.g. on-road vehicles, mobile robots, etc). Early literature mainly introduced physics-based or rule-based approaches, such as state estimation techniques based on kinematic models. These methods do not consider mutual influence between intelligent agents and they are only able to perform well in short-term prediction tasks with limited model flexibility [115, 155]. As machine learning techniques are studied more extensively, people began to employ learning-based models for prediction purpose, such as hidden Markov models [183], Gaussian mixture models [206], dynamic Bayesian network [71], and inverse reinforcement learning [172]. In recent years, researchers have proposed various learning-based prediction models, which enables more flexibility and capacity to capture underlying interactive behavior patterns [33, 89, 149, 196, 111, 122, 93, 110, 61, 170, 65, 14, 146, 98, 197, 151, 207, 20, 140, 188, 145]. The prediction hypotheses can be directly obtained from the model outputs. However, with an end-to-end fashion, physical feasibility constraints are usually ignored in these methods, which may result in implausible forecast. In this paper, we address multi-agent interaction modeling and introduce a probabilistic prediction system based on a deep generative framework. Our approach also explicitly considers the feasibility constraints of vehicles.

4.2.2 Relational Reasoning and Graph Networks

In general, the objective of relational reasoning is to reason about different entities and their relations from observed structured or unstructured data, such as image pixels [185],

words or sentences [113], human skeletons [77] and interactive navigating agents [203, 119, 19, 109, 121]. Popular techniques for relational reasoning and interaction modeling in earlier literature include, but are not limited to, social pooling mechanism [3], convolutional pooling mechanism [27], soft attention mechanism [179], etc. Recently, graph networks have proved to be effective for relational reasoning on graph-structured data, where there is no restriction on the message passing rules. In traffic scenarios, a typical representation of the whole scene is to formulate a graph, where nodes are agents and edges are their relationships. Most existing works focused on the approximation function parameterized by deep neural network due to its high flexibility, which leads to graph neural networks (GNN). In this paper, we present a graph neural network with both topological and temporal attention mechanisms to capture underlying interaction patterns and jointly predict future behaviors.

4.2.3 Deep Generative Models

Our approach is also related to deep generative models, which have been widely applied to representation learning and distribution approximation tasks [42, 75]. One of the advantages of generative modeling lies in the data distribution learning without supervision. Coupled with highly flexible deep networks, deep generative models have achieved satisfying performance in image generation, style transfer, sequence synthesis tasks, etc. Despite the variational auto-encoder, a highly flexible latent variable model with encoder-decoder architecture, tries its best to make the posterior of the latent variable and its prior (usually a normal distribution) as similar as possible, the two distributions do not match well in many tasks. Hence, it breaks the consistency of the model. Also, although generative adversarial networks have achieved satisfying performance on image generation tasks, it usually suffers from mode collapse problems, especially when applied to sequential data under the conditional setting. In order to mitigate these drawbacks, The Wasserstein auto-encoder (WAE) [177], proposed from the optimal transport point of view and combined with information theory, encourages the consistency between the encoded latent distribution and the prior distribution. A variant of variational auto-encoder was proposed in [211] and a modified approach was proposed in our previous work [120]. In this paper, we adopt the similar Wasserstein generative method in [120] as the basis of model training, and significantly extends pair-wise prediction to multi-agent prediction.

4.3 Preliminaries

In this section, we first provide a high-level summary of basics of graph neural networks. Then, we concisely introduce the Wasserstein generative modeling proposed in [120].

4.3.1 Graph Neural Network

The graph neural network is a type of deep learning models which are directly applied to graph structures. It naturally incorporates relational inductive bias into the model design. In the context of graph neural network, most graphs are attributed (with node attributes and/or edge attributes and/or global attributes). Generally, there are three basic operations in graph representation learning with GNN: edge update, node update and global update [9]. Note that the global update is optional, which is applied only if the graphs have global attributes. More formally, denote the graph with n nodes as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_i, i \in \{1, ..., n\}\}$ is a set of node attributes and $\mathcal{E} = \{e_{ij}, i, j \in \{1, ..., n\}\}$ is a set of edge attributes. Denote u as the global attribute. Then, the three update operation can be written as

$$\begin{aligned} e'_{ij} &= \phi^e(e_{ij}, v_i, v_j, u), & \bar{e}'_i &= \rho^{e\to v}(E'_i), \\ v'_i &= \phi^v(\bar{e}'_i, v_i, u), & \bar{e}' &= \rho^{e\to u}(E'), \\ u' &= \phi^u(\bar{e}', \bar{v}', u), & \bar{v}' &= \rho^{v\to u}(V'), \end{aligned} \tag{4.1}$$

where $E'_i = \{e'_{ij}, j \in N(i)\}$, $E' = \bigcup E'_i$, $V' = \{v'_i, i = 1, ..., n\}$, and $N(i)$ is the neighbors of node i. $\phi^e(\cdot)$,$\phi^v(\cdot)$ and $\phi^u(\cdot)$ are neural networks. $\rho^{e\to v}(\cdot)$, $\rho^{e\to u}(\cdot)$ and $\rho^{v\to u}(\cdot)$ are aggregation functions with the property of permutation invariance.

In this work, it is natural to represent intelligent agents as nodes, and their relation as edges. We only apply edge update and node update, since there is no global attribute in our setup.

4.3.2 Wasserstein Generative Modeling

The Wasserstein distance is defined in a metric space (χ, ρ):

$$W_\rho(Q, P) = \sup_{||f||_{\text{Lip}} \le 1} \int f(dQ - dP), \tag{4.2}$$

where $||f||_{\text{Lip}}$ is the Lipschitz constant of the function f. By the Kantorovich-Rubinstein duality, we can formulate Eq. (4.2) as an optimal transport problem, which is given by

$$W_\rho(Q, P) = \inf_{\mathbf{M}} \int \rho(x, x') d\mathbf{M} = \inf_{\mathbf{M}} \mathbf{E}_M[\rho], \tag{4.3}$$

where $\mathbf{M}(x, x')$ is the coupling distribution of $x \sim P$, $x' \sim Q$, which is a probability measure for $\chi \times \chi$.

Define

$$p_G(x) := \int_z p_G(x|z) p_z(z) \mathrm{d}z, \forall x \in \chi. \tag{4.4}$$

Following the WAE [177] with the assumption that $p_G(x|z)$ is deterministic, we have

$$\inf_{M \in \mathcal{P}(P_X, P_G)} \mathbf{E}_M[\rho(X, Y)] = \inf_{Q_z = P_z} \mathbf{E}_p \mathbf{E}_{q(z|x)}[\rho(X, G(Z))], \tag{4.5}$$

where $X \sim P_X$, $Z \sim Q(Z|X)$ and Q_z is the marginal distribution of Z. We relax the optimization problem as:

$$\begin{aligned} &\min_{Q(Z|X)\in\mathcal{Q}} \mathbf{E}_{P_X}\mathbf{E}_{Q(Z|X)}[\rho(X, G(Z))] \\ &s.t. \begin{cases} \mathbf{E}_{p(x)}[\mathcal{D}_{\mathrm{KL}}(Q(Z|X)||P(Z))] \leq \epsilon_1, \\ \mathcal{D}(Q_Z, P_Z) \leq \epsilon_2, \end{cases} \end{aligned} \tag{4.6}$$

where ϵ_1 and ϵ_2 are pre-defined constants. Then, the dual optimal problem is formulated as

$$\begin{aligned} &\max_{\alpha,\beta} \min_{Q(Z|X)\in\mathcal{Q}} \mathbf{E}_{P_X}\mathbf{E}_{Q(Z|X)}[\rho(X, G(Z))] \\ &+\alpha\mathbf{E}_{p(x)}[\mathcal{D}_{\mathrm{KL}}(Q(Z|X)||P(Z))] + \beta\mathcal{D}(Q_Z, P_Z), \end{aligned} \tag{4.7}$$

where α and β are chosen from a proper range.

After some algebra derivations, we obtain the following equivalent optimization problem

$$\begin{aligned} &\begin{cases} \max I_\phi(x, z) \\ s.t. \begin{cases} \mathcal{D}_{\mathrm{KL}}[p_\phi(z)||p(z)] \leq \epsilon_1, \\ \mathcal{D}_{\mathrm{KL}}[p_\phi(x, z)||p_\theta(x, z)] \leq \epsilon_2. \end{cases} \end{cases} \\ \Leftrightarrow &\begin{cases} \min\limits_{0<1-\alpha<\beta} -(1-\alpha)I_\phi(x, z) \\ \qquad +(\beta+\alpha-1)\mathcal{D}_{\mathrm{KL}}[p_\phi(z)||p(z)] \\ \qquad +\mathcal{D}_{\mathrm{KL}}[p_\phi(x, z)||p_\theta(x, z)], \end{cases} \end{aligned} \tag{4.8}$$

where $I_\phi(x, z)$ is the mutual information between x and z.

Please refer to our prior work [120] for more details on the derivation. The goal of both WAE and vanilla variational auto-encoder (VAE) is to learn the data distribution. According to the numerical experiment results in [120], the vanilla VAE tends to learn a distribution with smaller variance and have mode collapse issue; while WAE is better at capturing the true data distribution. Also, WAE is able to learn a better latent representation than VAE due to the regularization terms.

4.4 Problem Formulation

The objective is to predict future trajectories for multiple interactive agents, based on their historical states and context information. The prediction system can be also incorporated into any multi-target tracking frameworks. Without loss of generality, we assume N agents are navigating in the observation area, which are divided into M categories. In this work, the involved agents include vehicles, pedestrians and cyclists. We denote a set of agent trajectories as

$$\mathbf{T}^{1:T} = \{\boldsymbol{\tau}_i^{1:T} | \boldsymbol{\tau}_i^k = (x_i^k, y_i^k, v_i^k, \psi_i^k),\ T = T_h + T_f, i = 1, ..., N\}, \tag{4.9}$$

where T_h is the history horizon and T_f is the forecasting horizon. (x_i^k, y_i^k) is the position, v_i^k is the velocity, and ψ_i^k is the heading angle of agent i at time k. The coordinates can be either in the world space or image pixel space. We also denote a sequence of context information (raw images, semantic maps or the tensors which includes other relevant information) as

$$\begin{aligned} \mathbf{C}_g^{1:T} &= \{\mathbf{c}^{1:T}, T = T_h + T_f\} \text{ (global)}, \\ \mathbf{C}_l^{1:T} &= \{\mathbf{c}_i^{1:T}, T = T_h + T_f, i = 1, ..., N\} \text{ (local)}, \end{aligned} \tag{4.10}$$

which indicates components in the high-definition maps (e.g. road geometries, road lanes, drivable areas, traffic signs, etc). The future information is accessible during training. We aim to approximate the conditional distribution $p(\mathbf{T}^{T_h+1:T_h+T_f} | \mathbf{T}^{1:T_h}, \mathbf{C}^{1:T_h})$. The number of involved agents can be flexible in different cases. In the multi-target tracking tasks, the prediction model is iteratively applied.

4.5 Method: STG-DAT

In this section, we first provide an overview of the key modules and the architecture of the proposed generative trajectory prediction system. The detailed model design of each module will then be further illustrated.

4.5.1 System Overview

The detailed architecture of STG-DAT is shown in Figure 4.2, where a standard encoder-decoder architecture is employed. There are three key components: a deep feature extractor, an encoder with spatio-temporal graph generation and dual-attention network, and a decoder with a kinematic constraint layer. First, the feature extractor takes in both history and future information and outputs state, relation, and context feature embeddings. The information contains the trajectories of the involved interactive agents, and a sequence of context density maps and mean velocity fields. The scene images or semantic maps can also be included, if they are available in the dataset. Since the neural networks have the capability of extracting highly flexible features, we choose multi-layer perceptron (MLP) to generate state and relation embeddings and convolutional neural network (CNN) to generate context embedding. The extracted features are utilized to generate a spatio-temporal graph for both the history and the future, respectively. The node attributes are updated by a spatio-temporal graph attention mechanism. Then, the updated node attributes are transformed from the feature space into a latent space by an encoding function according to the derivation of the conditional generative modeling. Finally, the decoder based on the recurrent neural network generates feasible and human-like future trajectories for all the involved agents. The number of agents can be flexible in different cases due to the weight sharing and permutation invariance of the graph representation. All the components are implemented with deep neural networks, thus they can be trained end-to-end efficiently and consistently.

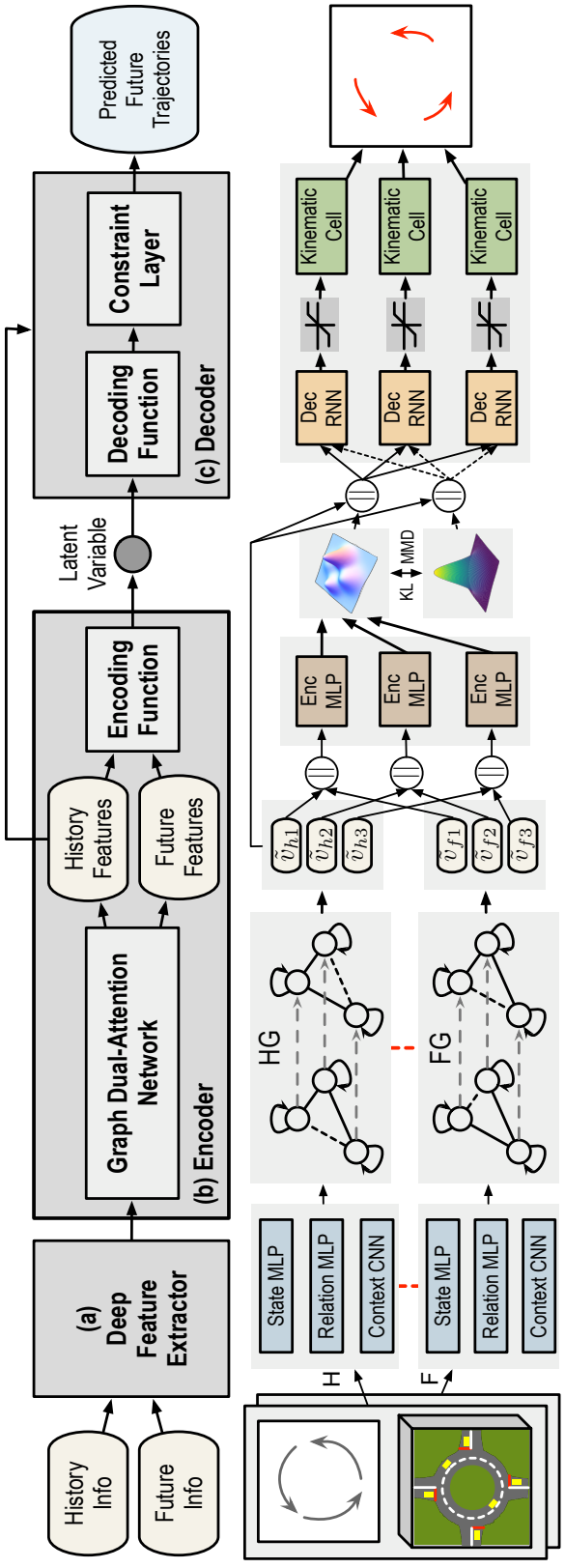


Figure 4.2: The detailed architecture of STG-DAT, which consists of three key components: (a) A deep feature extractor which extracts state, relation and context features from the trajectories of agents, the sequences of occupancy density maps and velocity fields. The red dashed lines indicate sharing parameters. (b) An encoder which includes a graph dual-attention network that processes spatio-temporal graphs and generates abstract node attributes containing interaction information, and an encoding function which maps the node attributes to a latent space. During the testing phase, the encoding function is not used. (c) A decoder which samples future trajectory hypotheses satisfying physical constraints for each agent. The bottom portion of the figure presents some details of (a)-(c). || denotes the concatenation operation. MLP refers to multi-layer perceptron. CNN refers to convolutional neural network.

4.5.2 Feature Extraction

The feature extractor consists of three parts: *State MLP*, *Relation MLP*, and *Context CNN*. The operations below are applied at each time step, and a sequence of state, relation, and context feature embeddings can be obtained.

- *State MLP*: It embeds the position, velocity, and heading information into a state feature vector for each agent. Different types of agents use distinct state embedding functions and the same type of agents share the same one. In this paper, we consider three types: vehicles, cyclists and pedestrians. The state embedding (SE) of agent i at time k is obtained by
$$SE_i^k = \mathrm{MLP}_s(\boldsymbol{\tau}_i^k). \tag{4.11}$$

- *Relation MLP*: It embeds the relative information between each pair of agents into a relation feature vector. We differentiate the edges with opposite directions between the same pair of nodes. The relative information can be either the distance and relative angle (in a 2D polar coordinate), or the differences between the positions of the two agents along two perpendicular axes (in a 2D Cartesian coordinate). We use the latter in this work, since it is simpler to compute and the performance is comparable to the former one. More specifically, consider a pair of agents i and j. When calculating the relation embedding associated with edge e_{ij} in a Cartesian coordinate, we set agent i as the origin and its heading as the positive direction. The relative position, velocity and heading angle of agent j with respect to agent i can be calculated and denoted as $\boldsymbol{\phi}_{ij}^k$. The relation embedding (RE) is obtained by
$$RE_{ij}^k = \mathrm{MLP}_r(\boldsymbol{\phi}_{ij}^k). \tag{4.12}$$

- *Context CNN*: It extracts spatial features for each agent from a local occupancy density map ($H \times W \times 1$) as well as heuristic features from a local velocity field ($H \times W \times 2$) centered on the corresponding agent. The reason of using occupancy density maps instead of real scene images is to remove redundant information and efficiently represent data-driven drivable regions. This information provides a prior knowledge of common driving behaviors at specific areas of the scene. The context embedding (CE) of agent i at time k is obtained by
$$CE_i^k = \mathrm{CNN}(\boldsymbol{c}_i^k). \tag{4.13}$$

4.5.3 Encoder with Graph Dual-Attention Network

After obtaining the extracted features, a history spatio-temporal graph (HG) and a future spatio-temporal graph (FG) are generated to represent the information related to the involved agents. Here, the state features and context features are concatenated to serve as the (agent) node attributes, whereas the relation features serve as edge attributes. The HG and FG contain different time steps, and they are processed in a similar fashion with the graph

dual-attention network. In a specific case, the number of nodes (agents) in both HG and FG is assumed to be fixed, which implies the same agents appear in the whole horizon. The edges are eliminated at a certain time step if the Euclidean distance between two agents is larger than a threshold d. Therefore, the graph connectivity and topology at different time steps may vary.

The proposed graph dual-attention network consists of two consecutive layers: a *topological attention layer* which updates node attributes from the spatial or topological perspective, and a *temporal attention layer* which outputs a high-level feature embedding for each node. The temporal attention layer summaries both the topological and temporal information and figure out relative significance of the information at each time step. Following the notation in Section III, assume there are totally n nodes (agents) in a graph, we denote a graph as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_i \in \mathbb{R}^{D_n}, i \in \{1, ..., n\}\}$ and $\mathcal{E} = \{e_{ij} \in \mathbb{R}^{D_e}, i, j \in \{1, ..., n\}\}$. D_n and D_e are the dimensions of node attributes and edge attributes.

Topological Attention Layer

The inputs of this layer are the original spatio-temporal graphs. The output is a new set of node attributes $\bar{\mathcal{V}} = \{\bar{v}_i^k \in \mathbb{R}^{\bar{D}_n}, i \in \{1, ..., n\}, k \in \{1, ..., T\}, T = T_h + T_f\}$, which can capture local structural properties. The topological attention coefficients α_{ij}^k (showing the significance of node j w.r.t. node i) are calculated by

$$\alpha_{ij}^k = \frac{\exp\left(-A_{ij}(\lambda \left\|v_i^k - v_j^k\right\|^2 + \mu \left\|e_{ij}^k\right\|^2)\right)}{\sum_{p \in N(i)} \exp\left(-A_{ip}(\lambda \left\|v_i^k - v_p^k\right\|^2 + \mu \left\|e_{ip}^k\right\|^2)\right)}, \tag{4.14}$$

where $N(i)$ is the first-order neighbor nodes (including i). A_{ij} is a prior attention coefficient which provides inductive bias from prior knowledge, λ and μ are weight parameters to adjust the relative importance of node attributes and edge attributes for computing attention coefficients. The underlying intuition is that the agents, with similar node attributes to the objective agent or with small spatial distance, tend to have more correlation thus should be paid more attention to. In this work, we set $A_{ij} = 1$ implying no prior attention bias, while more exploration on incorporating prior knowledge is left for future work. Then the node attributes are updated by

$$\bar{v}_i^k = \sum_{j \in N(i)} f_{\text{act}}(\alpha_{ij}^k W_n v_j^k), \tag{4.15}$$

where $f_{\text{act}}(\cdot)$ is an activation function, and W_n is learnable parameters. The above procedures are applied to each time step, and the weight matrices are shared across different time steps. We also employ the multi-head attention mechanism [178] to boost model performance by adjusting λ and μ, where the node attributes obtained by using different attention coefficients are concatenated into a whole vector. The above message passing procedures can be applied multiple times to capture higher-order interactions with an additional edge update procedure following the form of Eq. (1). More specifically, the updated edge attributes can be computed

by

$$\bar{e}_{ij}^k = \text{MLP}\left(\left[\bar{v}_i^k, \bar{v}_j^k, e_{ij}^k\right]\right). \tag{4.16}$$

Temporal Attention Layer

The input of this layer is the output of the topological attention layer, which is a set of node attributes $\bar{\mathcal{V}} = \{\bar{v}_i^k \in \mathbb{R}^{\bar{D}_n}, i \in \{1, ..., n\}, k \in \{1, ..., T\}\}$. The output is a set of highly abstract node attributes $\widetilde{\mathcal{V}} = \{\widetilde{v}_i \in \mathbb{R}^{\widetilde{D}_n}, i \in \{1, ..., n\}\}$. These attributes will be further processed by the downstream modules. The temporal attention coefficients β_i^k is

$$\begin{aligned}
\beta_i^{hk} =& \frac{\exp\left(f_{\text{act}}(\bar{v}_i^{k\top} w)\right)}{\sum_{k'=1}^{T_h} \exp\left(f_{\text{act}}(\bar{v}_i^{k'\top} w)\right)}, \ (1 \le k \le T_h) \\
\beta_i^{fk} =& \frac{\exp\left(f_{\text{act}}(\bar{v}_i^{k\top} w)\right)}{\sum_{k'=T_h+1}^{T_h+T_f} \exp\left(f_{\text{act}}(\bar{v}_i^{k'\top} w)\right)}, \ (T_h + 1 \le k \le T_h + T_f)
\end{aligned} \tag{4.17}$$

where $w \in \mathbb{R}^{\bar{D}_n}$ is a weight vector parameterizing the attention function. Then the node attributes are updated by

$$\widetilde{v}_i^h = \sum_{k=1}^{T_h} f_{\text{act}}(\beta_i^{hk} w^\top \bar{v}_i^k), \quad \widetilde{v}_i^f = \sum_{k=T_h+1}^{T_h+T_f} f_{\text{act}}(\beta_i^{fk} w^\top \bar{v}_i^k). \tag{4.18}$$

The multi-head attention mechanism can also be employed by learning different w and fusing the information by averaging or concatenation operations.

Feature Encoding

For each agent, the history and future node attributes are concatenated and mapped by an encoding function f_{enc} to obtain a latent variable z_i, which is given by

$$z_i = f_{\text{enc}}([\widetilde{v}_i^h || \widetilde{v}_i^f]). \tag{4.19}$$

The underlying intuition is that, during the training phase, the latent variable is able to encode the future information conditioned on the given history information, and it is trained to be consistent with the prior distribution by a regularization term in the loss function. During the testing phase, although the future information is not available, it can be implicitly obtained by sampling the latent variable from the prior distribution.

4.5.4 Decoder with Kinematic Constraint

We impose a kinematic constraint layer after the decoder gated recurrent unit (GRU) to enforce feasible trajectory prediction. The same types of agents share the same GRU unit, while different types use distinct ones. This is reasonable since the behavior patterns, speed ranges and traversable areas may be diverse among different types.

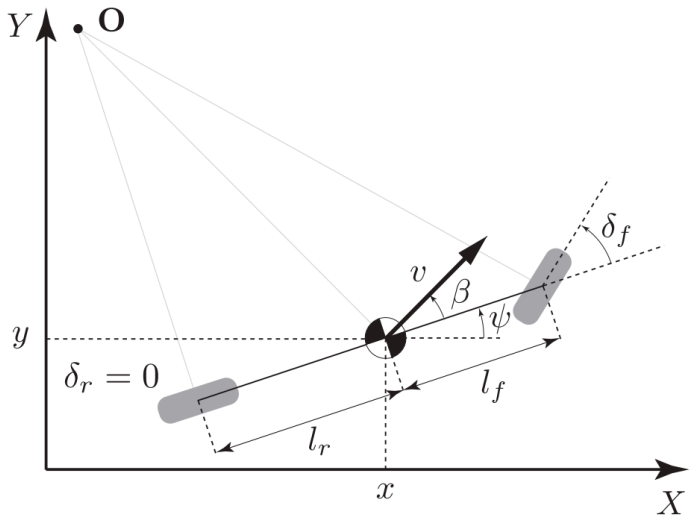


Figure 4.3: The diagram of the kinematic bicycle model adopted from [79]. The model equations are provided in Eq. (4.20).

The bicycle model is a widely used nonlinear model to approximate the kinematics of vehicles, which is shown in Fig. 4.3. Here we adopt the discretized form, which is given by

$$\begin{cases} x(k+1) = x(k) + v(k)\cos(\psi(k) + \beta(k))\Delta T, \\ y(k+1) = y(k) + v(k)\sin(\psi(k) + \beta(k))\Delta T, \\ \psi(k+1) = \psi(k) + \dfrac{v(k)}{l_r}\sin\beta(k)\Delta T, \\ v(k+1) = v(k) + a(k)\Delta T, \\ \beta(k+1) = \beta(k) + \dot{\beta}(k)\Delta T, \end{cases} \tag{4.20}$$

where (x, y) are the coordinates of the center of mass, ψ is the inertial heading and v is the speed of the vehicle. β is the angle of the current velocity of the center of mass with respect to the longitudinal axis of the car. l_f and l_r denote the distance from the center of mass of the vehicle to the front and rear axles, respectively. The state of vehicles at time step k is denoted as $\mathbf{s}(k) = [x(k), y(k), \phi(k), v(k), \beta(k)]^\top$, and the control input at time step k as $\mathbf{u}(k) = [a(k), \dot{\beta}(k)]^\top$. Eq. (4.20) can be expressed as $\mathbf{s}(k+1) = \mathbf{f}(\mathbf{s}(k), \mathbf{u}(k))$.

The prediction system is expected to provide the position distribution of each agent at each time step. The distribution of the control input $\mathbf{u}(k)$ is assumed to be a multi-variate Gaussian distribution at each time step, which is parameterized by the output of the GRU cell. We provide an illustrative example for the agent i. The inputs of the gated recurrent unit (GRU) are the node attribute $\widetilde{v}_i$ at the first step and zero paddings for the following steps. The outputs are the raw $\mathbf{u}(k)$ at each step which are truncated by a saturation function

in order to restrict the control actions in the feasible range. Then the kinematic cell takes in the current state and action, and outputs the state at the next step. This procedure is iterated until the prediction horizon is reached. If l_r is not a prior knowledge or cannot be observed during testing, then we can approximate it with a constant based on the statistics of training data.

In order to propagate the uncertainty to future time steps, two options are available with tradeoffs. The details are introduced in the following.

• **Nonlinear system with Monte Carlo sampling**:

We can approximate the distribution of position by a set of Monte Carlo samples, which is highly flexible without any restrictions on the nonlinear dynamic system. The particle samples are propagated by the nonlinear bicycle model. The prediction accuracy tends to be improved as the number of particles increases. However, the number of samples may need to be adjusted according to real-time requirements. If the explicit probability density function of the state variable is required, the kernel density estimation technique can be employed in a non-parametric way.

• **Linearized system with Gaussian assumption**:

Since $\mathbf{u}(k)$ follows Gaussian distribution, we can obtain an analytic distribution of position by linearizing the bicycle model at the current state. It is easy to show that the position distribution is also a Gaussian distribution. This is simple to implement and computationally efficient, while the flexibility is very limited. This restriction leads to lower prediction accuracy in general, especially when the real distribution is multi-modal.

For a nonlinear system with Gaussian assumption, the expectation and covariance of the state variable can be propagated in a fashion similar to the prior (prediction) update of extended Kalman filter [163]. We can linearize the system around the current state $\mathbf{s}(k)$,

$$\mathbf{s}(k+1) = \mathbf{Df}_s(k)\mathbf{s}(k) + \mathbf{Df}_u(k)\mathbf{u}(k) \tag{4.21}$$

$$\mathbf{u}(k) \sim \mathcal{N}(\boldsymbol{\mu}_u(k), \boldsymbol{\Sigma}_{uu}(k)) \tag{4.22}$$

where $\mathbf{Df}_s(k)$ and $\mathbf{Df}_u(k)$ are the Jacobian matrices defined as below,

$$\begin{aligned}
\mathbf{Df}_s(k) &= \begin{bmatrix} 1 & 0 & -\Delta T v_y(k) & \Delta T v_x(k) & -\Delta T v_y(k) \\ 0 & 1 & \Delta T v_x(k) & \Delta T v_y(k) & \Delta T v_x(k) \\ 0 & 0 & 1 & \dfrac{\Delta T}{l_r}\sin\beta(k) & \Delta T \dfrac{v(k)}{l_r}\cos\beta(k) \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \\
\mathbf{Df}_u(k) &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \Delta T & 0 \\ 0 & \Delta T \end{bmatrix},
\end{aligned} \tag{4.23}$$

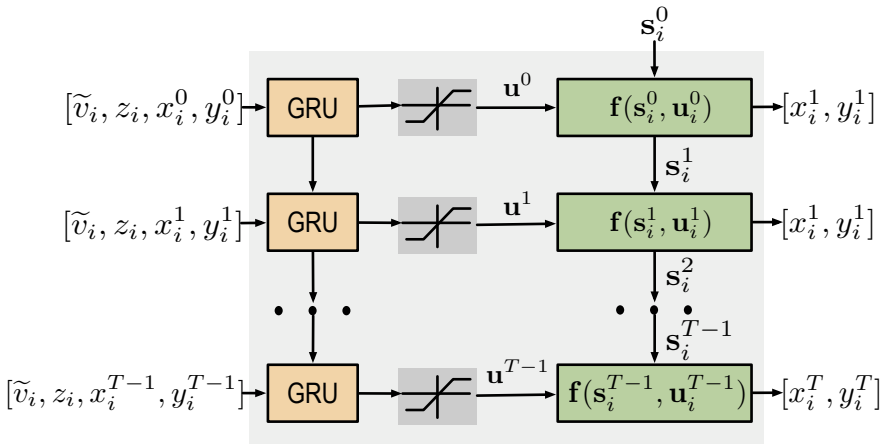


Figure 4.4: The diagram of the recurrent decoder with kinematic constraint layer. The recurrent process consists of two phases: burn-in phase and prediction phase. In the burn-in phase, the history groundtruth is used as the input of GRU at each step for initialization purpose. In the prediction phase, the output position at the last step will serve as the input of the next step. The iteration continues until the prediction horizon is reached.

where $\gamma(k) = \phi(k) + \beta(k)$, $v_x(k) = v(k)\cos\gamma(k)$, and $v_y(k) = v(k)\sin\gamma(k)$. Then we have the distribution of $\mathbf{s}(k+1)$, which is expressed as

$$\mathbf{s}(k+1) \sim \mathcal{N}(\boldsymbol{\mu}_s(k+1), \boldsymbol{\Sigma}_{ss}(k+1)), \tag{4.24}$$

where

$$\begin{aligned} \boldsymbol{\mu}_s(k+1) &= \mathbf{D}\mathbf{f}_s(k)\boldsymbol{\mu}_s(k) + \mathbf{D}\mathbf{f}_u(k)\boldsymbol{\mu}_u(k), \\ \boldsymbol{\Sigma}_{ss}(k+1) &= \mathbf{D}\mathbf{f}_s(k)\boldsymbol{\Sigma}_{ss}(k)\mathbf{D}\mathbf{f}_s(k)^\top + \mathbf{D}\mathbf{f}_u(k)\boldsymbol{\Sigma}_{uu}(k)\mathbf{D}\mathbf{f}_u(k)^\top, \end{aligned} \tag{4.25}$$

with the initial condition $\mathbf{s}(0) \sim \mathcal{N}(\boldsymbol{\mu}_s(0), \boldsymbol{\Sigma}_{ss}(0))$.

4.5.5 Loss Function and Training

In this part, we demonstrate the loss function of our model, which is based on the optimization formulation of Wasserstein generative modeling aforementioned in Section III. In order to keep consistent with Section III, we use the same notations: x denotes the predicted trajectories, z denotes the latent variable, and y denotes the condition variable.

The optimization problem can be formulated in the same way as in [120], which is written as

$$\begin{aligned} \min_{\theta,\phi,s.t.0<1-\alpha<\beta} \quad & -\mathbf{E}_{p_\phi(z|x,y)}[\log p_\theta(x|z,y)]] \\ & + \alpha \mathbf{E}_{p(x|y)}[D_{\mathrm{KL}}[p_\phi(z|x,y)||p(z|y)]] \\ & + \beta D(p_\phi(z|y), p(z|y)). \end{aligned} \tag{4.26}$$

The detailed derivation can be found in [120]. Since the whole system is fully differentiable, we can train the network in an end-to-end fashion by the Adam optimizer [74]. The loss function is given by

$$\begin{aligned}\mathcal{L} = & \gamma \mathbf{E}_{j\in\{1,\dots,N_b\}} \left\| \tau_j^{T_h+1:T_h+T_f} - \hat{\tau}_j^{T_h+1:T_h+T_f} \right\|^2 \\ & + \alpha \mathbf{E}_{p(x|y)}[D_{\mathrm{KL}}[p_\phi(z|x,y)||p(z|y)] \\ & + \beta \mathrm{MMD}(p_\phi(z|y), p(z|y)),\end{aligned} \tag{4.27}$$

where

$$\begin{aligned} & p_\phi(z|x,y) = \mathcal{N}(\mathrm{MLP}(\tilde{v}_h, \tilde{v}_f), \mathbf{I}), \\ & \tilde{v}_h = \mathrm{GDAT}(\mathrm{FE}(y)),\ \tilde{v}_f = \mathrm{GDAT}(\mathrm{FE}(x)), \\ & y = \{\mathbf{T}_{1:T_h}, \mathbf{C}_{1:T_h}\}, \\ & x = \{\mathbf{T}_{T_h+1:T_h+T_f}, \mathbf{C}_{T_h+1:T_h+T_f}\}. \end{aligned} \tag{4.28}$$

FE is the deep feature extractor and GDAT is the proposed graph dual-attention network. γ is a weight parameter to adjust the relative importance of the reconstruction loss, N_b is the total number of training agents, D_{KL} is Kullback-Leibler divergence, and MMD is maximum mean discrepancy. If $\gamma \gg \alpha, \beta$, then the loss function degenerates to the mean squared error loss. The whole model is trained in an end-to-end fashion.

4.6 Experiments

In this section, we validate the proposed method on three publicly available benchmark datasets for trajectory prediction of traffic participants. The results are analyzed and compared with state-of-the-art baselines.

4.6.1 Datasets

Here we briefly introduce the datasets below.

- **ETH** [142] **and UCY** [87]: These two datasets are usually used together in literature, which include bird-eye-view videos and annotations of pedestrians in both indoor and outdoor scenarios. The trajectories were extracted in the world space (unit: meters).
- **Stanford Drone Dataset (SDD)**[148]: The dataset also contains a set of bird-eye-view images and the corresponding trajectories of involved entities. It was collected in multiple scenarios in a university campus full of interactive pedestrians, cyclists and vehicles. The trajectories were extracted in the image pixel space.
- **INTERACTION Dataset (ID)**[204]: The dataset contains naturalistic motions of various traffic participants in a variety of highly interactive driving scenarios. Trajectory data was collected using drones and traffic cameras. The high-definition maps

of scenarios and agents' trajectories are provided. We consider three types of scenarios: roundabout (RA), unsignalized intersection (UI) and highway ramp (HR). The trajectories were extracted in the world space (unit: meters).

4.6.2 Global Context Information

In order to provide better global context information, we designed two different representations, namely occupancy density map and mean velocity field. After constructing such global context information offline, we did decentralized online localization for the corresponding target agent and obtained their local context information, which was used in both training and testing phases.

Occupancy Density Map

The density map describes the normalized frequency distribution of all the agents' locations. For a specific scene, we first split our map into a number of bin areas, which are 1m×1m squares. Without loss of generality, we denote this histogram as B, and all the agents in different frames as a set $\{o_{k,p}\}$, where p is the agent index and k is the frame index. We obtained the global representation of density by calculating $B_{i,j} = \sum_{t,k} \phi(o_{k,p}, i, j)$, where i, j are the indices of the histogram and $\phi(o_{k,p}, i, j)$ is an indicator function which equals 1 if $o_{k,p}$ is located in the bin area indicated by index i, j and 0 otherwise. Then we normalized this density map by dividing all bin values by the sum of the values in this histogram and used this normalized histogram as our occupancy density map.

Mean Velocity Field

Similarly, we also created a map of velocity field which contains 1m×1m square areas. We denote the whole map as VF and the bin item indexed by i, j as $VF(i, j)$. The $VF(i, j)$ is a two-dimensional vector representing the average speed along vertical and horizontal axes of all the agents in this area. More formally,

$$\begin{aligned} VF(i,j)_x &= \frac{1}{N} \sum_{k,p} \phi(v_{k,p}, i, j) v^x_{k,p}, \\ VF(i,j)_y &= \frac{1}{N} \sum_{k,p} \phi(v_{k,p}, i, j) v^y_{k,p}, \end{aligned} \tag{4.29}$$

where N is the number of points located at the bin area (i, j).

4.6.3 Localization for Local Context Information

After obtaining the global context offline, our model utilized a decentralized method to do localization for each agent during training and testing. Given the location and the moving

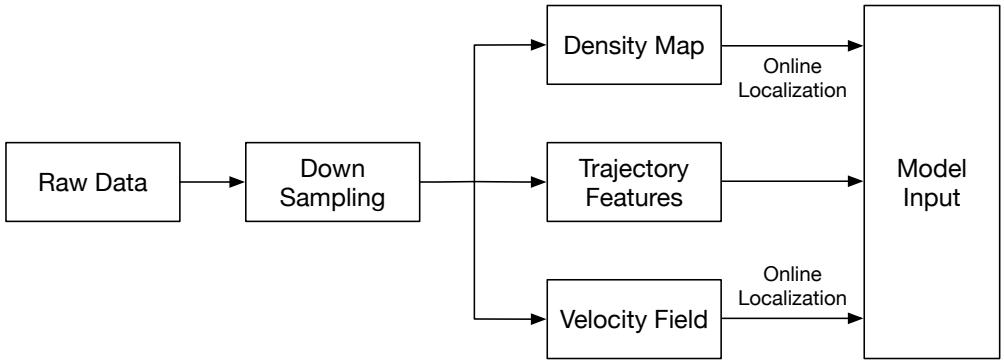


Figure 4.5: The pipeline of data preprocessing. The raw data was first down-sampled to be compatible to our experiment setup.

direction of the current agent at the current time step, we obtained a local context centered on this agent along its moving direction from the global context. All the agents share the same size of the local context map.

4.6.4 Evaluation Metrics

We evaluate the model performance in terms of average displacement error (ADE) and final displacement error (FDE), which are exactly the same as [3, 45, 80]. ADE is defined as the average distance between the predicted trajectories and the groundtruth over all the involved entities within the prediction horizon. FDE is defined as the deviated distance at the last predicted time step. For the ETH, UCY, and SDD dataset, we predicted the future 12 time steps (4.8s) based on the historical 8 time steps (3.2s). For the ID dataset, we predicted the future 10 time steps (5.0s) based on the historical 4 time steps (2.0s).

4.6.5 Baseline Methods

- **Probabilistic LSTM (P-LSTM)** [95]: The backbone of the model is the same as an encoder-decoder architecture with vanilla LSTM. In order to incorporate uncertainty in the model, a noise term sampled from the normal distribution is added in the input, which results in a probabilistic model.
- **Social LSTM (S-LSTM)** [3]: The trajectories are encoded with an LSTM layer, whose hidden states serve as the input of the proposed social pooling layer, which handles interaction modeling implicitly.
- **Social GAN (S-GAN)** [45]: The model introduces a generative adversarial learning scheme into S-LSTM to further improve performance.

- **Social Attention (S-ATT)** [179]: The model is based on the architecture of Structural-RNN [66], which deals with spatio-temporal graphs with recurrent neural networks.
- **DESIRE** [89]: The model is a deep stochastic inverse optimal control framework based on conditional variational auto-encoder with RNN encoders and decoders. A ranking module for sampled trajectories was introduced to indicate their likelihood.
- **Social-BiGAT** [80]: The model is a graph-based generative adversarial network, which is based on a graph attention network. A recurrent encoder-decoder architecture is trained via an adversarial scheme.
- **Trajectron** [65]: The model combines tools from recurrent sequence modeling and variational deep generative modeling to produce a distribution of future trajectories.

4.6.6 Implementation Details

A batch size of 64 was used and the models were trained for 100 epochs with early stopping using the Adam optimizer with an initial learning rate of 0.001. The models were trained on a single NVIDIA TITAN X GPU. We used a split of 70%, 10%, 20% as training, validation and testing data, respectively. During the testing phase, the data processing time for a single time step is around 8ms in average. According to our setting, we predict the future 10 time steps which needs around 80ms (equivalent to 12.5Hz).

The details of our model architecture are introduced in the following.

- **Deep Feature Extractor (FE)**: The *State MLP* and *Relation MLP* both have three hidden layers with 128 hidden units. The *Context CNN* adopts the same backbone structure of ResNet18 [51], which is trained from scratch.
- **Graph Dual-Attention Network (GDAT)**: The dimensions of node attributes and edge attributes are 64 and 16, respectively. These dimensions are fixed in different rounds of message passing. The activation functions in the attention mechanism are LeakyReLU.
- **Encoding Function**: The encoding function is a three-layer MLP with 128 hidden units. The dimension of latent variable is 32.
- **Decoding Function**: The decoding function is a GRU recurrent layer with 128 hidden units.

4.6.7 Quantitative Analysis

- **ETH and UCY Datasets**: The comparison of the proposed STG-DAT and baseline methods in terms of ADE and FDE is shown in Table 4.1. Some of the reported statistics are adopted from the original papers. All the baseline methods deal with interaction

Table 4.1: ADE/FDE (meters) Comparisons (ETH & UCY datasets).

Scenes	S-LSTM	S-GAN	CGNS	Social-BiGAT	Trajectron	**STG-DAT**
ETH	1.09 / 2.35	0.81 / 1.52	0.62 / 1.40	0.69 / 1.29	0.48 / 0.93	**0.38** / **0.77**
HOTEL	0.79 / 1.76	0.72 / 0.61	0.70 / 0.93	0.49 / 1.01	0.29 / 0.54	**0.25** / **0.39**
UNIV	0.67 / 1.40	0.60 / 1.26	0.48 / 1.22	0.55 / 1.32	0.44 / 0.93	**0.41** / **0.82**
ZARA1	0.47 / 1.00	0.34 / 0.69	0.32 / 0.59	0.30 / 0.63	0.35 / 0.68	**0.23** / **0.50**
ZARA2	0.56 / 1.17	0.42 / 0.84	0.35 / 0.71	0.36 / 0.75	0.36 / 0.70	**0.21** / **0.46**
AVG	0.72 / 1.54	0.58 / 1.18	0.49 / 0.97	0.48 / 1.00	0.46 / 0.94	**0.30** / **0.59**

Table 4.2: ADE/FDE (pixels) Comparisons (SDD dataset).

S-LSTM	S-GAN	CGNS	DESIRE	Trajectron	**STG-DAT** (same node)	**STG-DAT**
33.19 / 56.38	24.81 / 38.62	15.60 / 28.20	19.30 / 34.12	17.38 / 31.46	14.55 / 23.54	**13.25** / **21.94**

modeling in a specific way. The S-LSTM employs a social pooling mechanism to model the interactions between entities. The S-GAN improve the performance by introducing deep generative modeling. The CGNS combines conditional latent space learning and variational divergence minimization to further enhance the generation capability. The Trajectron adopts a graph-structured model with sequence modeling. Both Social-BiGAT and our method leverage the trajectory and context information, but in different ways. Our model can achieve better performance owing to the explicit interaction modeling with graph neural networks and more compact distribution learning with conditional Wasserstein generative modeling. In general, our approach achieves the smallest ADE and FDE across different scenes. The average ADE/FDE are reduced by 34.8%/37.2% compared to the best baseline (Trajectron).

- **Stanford Drone Dataset**: The comparison of results is provided in Table 4.2, where the ADE and FDE are reported in the pixel distance. Note that we also included cyclists and vehicles in the dataset besides pedestrians. In general, the relative performance of baseline methods is consistent with the observation on the ETH/UCY datasets. Our approach achieves the best performance in terms of prediction error, which implies the superiority of explicit interaction modeling and necessity of leveraging both trajectory and context information. In order to show the effectiveness of distinct node embedding functions for different types, we provide an ablative result by treating all the agents as the same type, which leads to an increase on the prediction error. For the full model, the ADE/FDE are reduced by 15.1%/22.2% with respect to the best baseline (CGNS).

- **INTERACTION Dataset**: We finally compare the model performance on the real-world driving dataset in Table 4.3. For fair comparison, we only involved the baseline approaches whose codes are publicly available and can be adapted to the same setup as our approach. Although we trained a unified prediction model on different scenarios

Table 4.3: ADE/FDE (meters) Comparisons (ID dataset).

Scenes	Time	Baseline Methods					
		P-LSTM	S-LSTM	S-GAN	S-ATT	CGNS	Trajectron
RA	1.0s	0.13 / 0.15	0.16 / 0.20	0.13 / 0.16	0.14 / 0.17	0.11 / 0.16	0.09 / 0.13
	2.0s	0.27 / 0.35	0.25 / 0.39	0.22 / 0.35	0.23 / 0.39	0.21 / 0.32	0.18 / 0.30
	3.0s	0.58 / 0.83	0.52 / 0.80	0.45 / 0.72	0.53 / 0.77	0.44 / 0.68	0.40 / 0.61
	4.0s	0.87 / 1.35	0.84 / 1.33	0.70 / 1.19	0.78 / 1.25	0.69 / 1.03	0.64 / 0.98
	5.0s	1.36 / 1.88	1.29 / 1.80	1.13 / 1.51	1.22 / 1.69	1.01 / 1.45	0.95 / 1.32
UI	1.0s	0.11 / 0.20	0.12 / 0.17	0.11 / 0.16	0.12 / 0.18	0.11 / 0.14	0.10 / 0.14
	2.0s	0.28 / 0.49	0.25 / 0.48	0.24 / 0.40	0.26 / 0.43	0.24 / 0.38	0.23 / 0.35
	3.0s	0.45 / 0.88	0.41 / 0.83	0.39 / 0.77	0.41 / 0.80	0.38 / 0.76	0.36 / 0.72
	4.0s	0.83 / 1.64	0.77 / 1.47	0.74 / 1.31	0.75 / 1.39	0.70 / 1.18	0.69 / 1.10
	5.0s	1.34 / 2.36	1.31 / 2.20	1.21 / 1.98	1.29 / 2.06	1.06 / 1.90	1.01 / 1.84
HR	1.0s	0.07 / 0.11	0.06 / 0.10	0.06 / 0.08	0.06 / 0.09	0.05 / 0.08	0.05 / 0.07
	2.0s	0.21 / 0.39	0.19 / 0.35	0.17 / 0.31	0.17 / 0.32	0.16 / 0.29	0.15 / 0.27
	3.0s	0.55 / 1.02	0.49 / 0.92	0.43 / 0.81	0.45 / 0.84	0.40 / 0.75	0.38 / 0.70
	4.0s	0.90 / 1.58	0.81 / 1.43	0.71 / 1.26	0.74 / 1.31	0.66 / 1.17	0.62 / 1.09
	5.0s	1.51 / 2.57	1.36 / 2.31	1.20 / 2.04	1.25 / 2.12	1.12 / 1.90	1.04 / 1.77

Scenes	Time	**STG-DAT**			
		T	**T + C − ATT**	**T + C**	**T + C + K**
RA	1.0s	0.07 / 0.11	0.09 / 0.13	0.08 / 0.13	**0.06** / **0.10**
	2.0s	0.19 / 0.31	0.22 / 0.34	0.16 / 0.29	**0.14** / **0.25**
	3.0s	0.35 / 0.57	0.43 / 0.66	0.31 / 0.54	**0.26** / **0.47**
	4.0s	0.58 / 0.92	0.70 / 1.05	0.51 / 0.83	**0.42** / **0.71**
	5.0s	0.90 / 1.25	1.08 / 1.56	0.85 / 1.17	**0.68** / **1.01**
UI	1.0s	**0.06** / 0.10	0.09 / 0.16	0.08 / 0.14	0.07 / **0.11**
	2.0s	0.20 / 0.32	0.23 / 0.36	0.19 / 0.30	**0.16** / **0.28**
	3.0s	0.37 / 0.65	0.41 / 0.71	0.34 / 0.59	**0.30** / **0.55**
	4.0s	0.60 / 0.98	0.72 / 1.19	0.55 / 0.91	**0.49** / **0.83**
	5.0s	0.97 / 1.69	1.22 / 1.91	0.88 / 1.50	**0.77** / **1.26**
HR	1.0s	**0.04** / 0.07	0.05 / 0.07	**0.04** / 0.07	**0.04** / **0.06**
	2.0s	0.14 / 0.25	0.15 / 0.27	0.13 / 0.24	**0.12** / **0.22**
	3.0s	0.35 / 0.66	0.37 / 0.70	0.34 / 0.64	**0.31** / **0.58**
	4.0s	0.58 / 1.02	0.61 / 1.08	0.56 / 0.99	**0.51** / **0.90**
	5.0s	0.97 / 1.65	1.03 / 1.75	0.95 / 1.61	**0.86** / **1.46**

simultaneously, we analyzed the results for each type of scenario separately. In the HR scenarios, the results of baseline methods are comparable while our model achieves the best performance. The behavior patterns of vehicles in HR scenarios are relatively easy to forecast, since most vehicles are doing car following without highly interactive behaviors. In the RA and UI scenarios, however, the superiority of the proposed system is more distinguishable due to frequent interactions. The P-LSTM performs the worst since it predicts future trajectories for each agent individually without considering their relations. Although all the other baseline approaches incorporate interaction modeling by different strategies which further reduce the prediction error, our model still performs the best. This implies the advantages of graph dual-attention network for interaction modeling, as well as kinematic layer for feasibility constraints. By using our full model $\mathbf{T}+\mathbf{C}+\mathbf{K}$, the 5.0s ADE/FDE are reduced by 28.4%/23.5%, 23.8%/31.5% and 17.3%/17.5% in RA, UI and HR with respect to the best baseline (Trajectron), respectively.

We also tested the tracking performance using our approach and baseline methods, which is shown in Table IV. The constant velocity model (CVM) and constant acceleration model (CAM) are widely used linear vehicle kinematics models with a constant velocity / acceleration assumption. These models are employed frequently in multi-target tracking literature. A Gaussian noise term is injected at each time step to model uncertainty. It shows that tracking with learning-based models performs consistently better than CVM and CAM due to the ability of interaction-aware prediction. Our approach STG-DAT achieves a significantly higher accuracy.

4.6.8 Qualitative and Ablative Analysis

We qualitatively evaluated on prediction hypotheses of typical testing cases on the SDD dataset and ID dataset in Fig. 4.6 and Fig. 4.7, respectively. Although we jointly predict all the agents in a scene, we show predictions for a subset for clearness. It shows that our approach can handle different challenging scenarios (e.g. intersection, roundabout) and diverse behaviors (e.g., going straight, turning, waiting, stopping) of vehicles and pedestrians. Generally, the ground truth trajectories are close to the mean of predicted distribution and the model also allows for uncertainty.

We also conducted comprehensive ablative analysis on the ID dataset to demonstrate relative significance of context information, dual-attention mechanism and the kinematic constraint layer for vehicle trajectory prediction. The descriptions of compared model settings are provided below:

- $\mathbf{T}$: This is the model without the kinematic layer, which only uses trajectory information.
- $\mathbf{T}+\mathbf{C}-\mathbf{ATT}$: This is the model without the dual-attention mechanism or the kinematic constraint layer. We used equal attention in this model setting instead.
- $\mathbf{T}+\mathbf{C}$: This is the model without kinematic constraint layer.

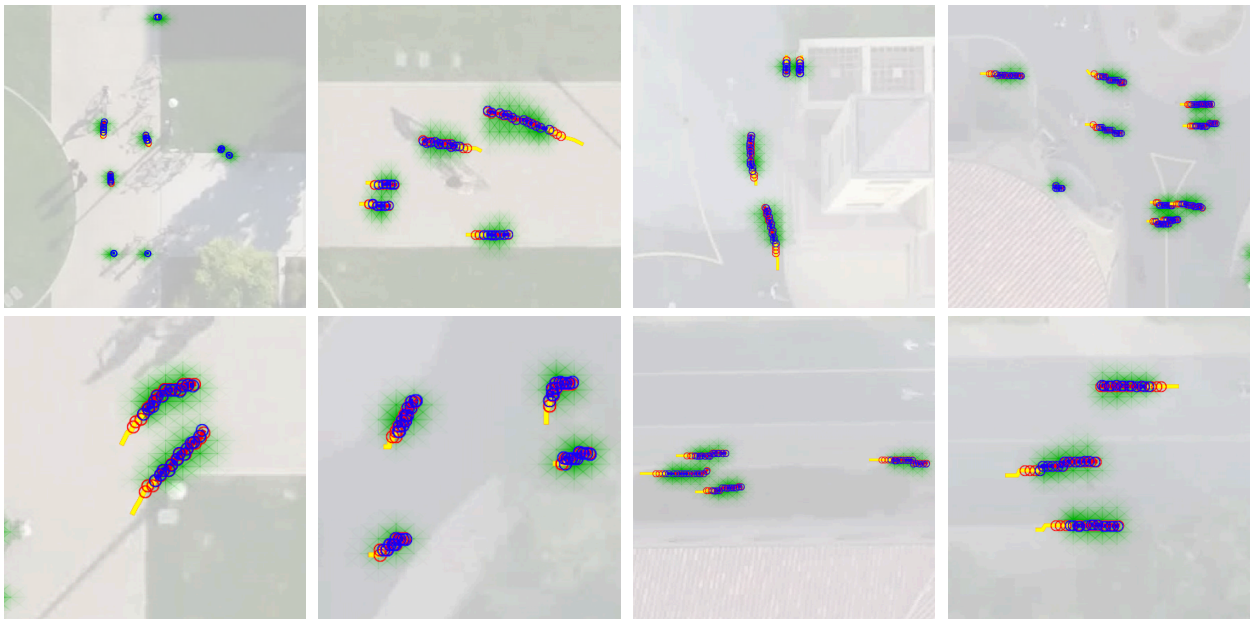


Figure 4.6: Qualitative results on the SDD dataset. The green mask represents the predicted distribution and the yellow, blue and red lines represent historical observation, groundtruth and a trajectory hypothesis sampled from the distribution with the smallest error, respectively.

- $\mathbf{T}+\mathbf{C}+\mathbf{K}$: This is the whole proposed model including all the components.

The ADE/FDE of each model setting are shown in the lower part of Table 4.3.

- $\mathbf{T}$ versus $\mathbf{T}+\mathbf{C}$: We show the effectiveness of employing scene context information. $\mathbf{T}$ is the model without the kinematic layer, which only uses trajectory information, while $\mathbf{T}+\mathbf{C}$ further employs context information. The models directly output the position displacements $(\Delta x^k, \Delta y^k)$ at each step, which are aggregated to get complete trajectories. We can see little difference on prediction errors over short horizons, while the gap becomes larger as the horizon extends. The reason is that the vehicle trajectories within a short period can usually be well approximated by a constant velocity model, which are not heavily restricted or affected by the static environmental context. However, as the forecasting horizon increases, the effects of context constraints cannot be ignored anymore, which leads to larger performance gain of leveraging context information. Compared with $\mathbf{T}$, the 5.0s ADE/FDE of $\mathbf{T}+\mathbf{C}$ are reduced by 5.6%/6.4%, 9.3%/7.7% and 2.1%/2.4% in RA, UI and HR scenarios, respectively. This implies that the context information has larger effects on the prediction in RA and UI scenarios, where the influence of road geometries cannot be ignored. The context information does little help to HR scenarios, since most vehicles go straight on highways. In Figure 4.7, the

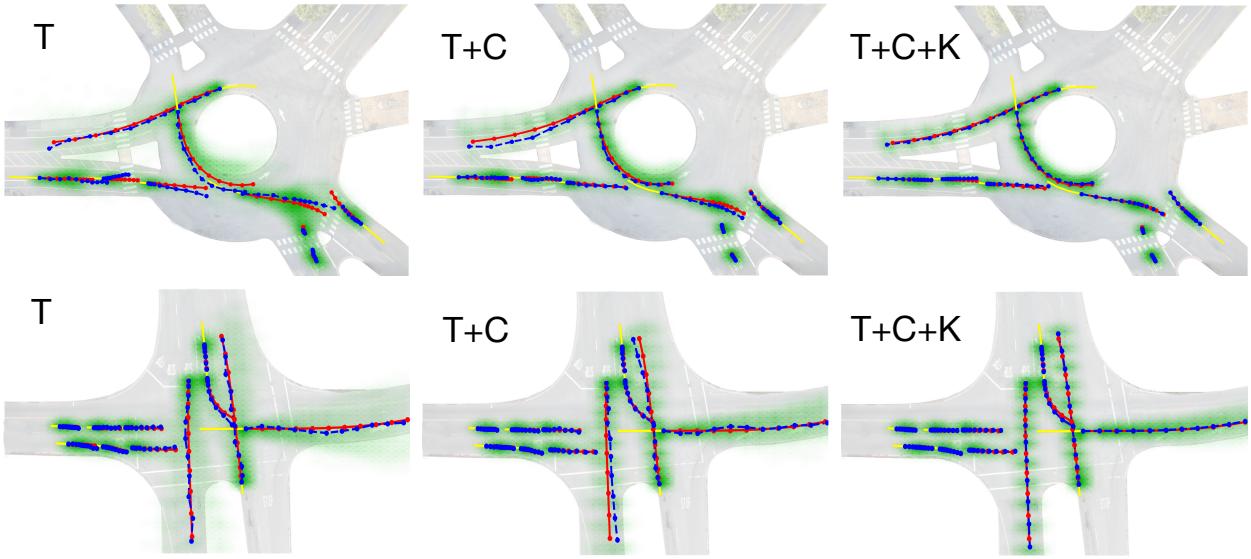


Figure 4.7: Qualitative and ablative results on the ID dataset. The green mask represents the predicted distribution and the yellow, blue and red lines represent historical observation, groundtruth and a trajectory hypothesis sampled from the distribution with the smallest error, respectively.

predicted distribution of $\mathbf{T}+\mathbf{C}$ is more compliant to roadways to avoid collisions and the vehicles near the "yield" or "stop" signs tend to yield or stop. However, $\mathbf{T}$ generates samples which are outside of feasible areas or violating traffic rules.

- $\mathbf{T}+\mathbf{C}-\mathbf{ATT}$ versus $\mathbf{T}+\mathbf{C}$: We show the effectiveness of the proposed dual-attention mechanism. $\mathbf{T}+\mathbf{C}-\mathbf{ATT}$ uses equal attention coefficients in both topological and temporal layers. According to the statistics reported in Table 4.3, compared with equal attention, employing the dual-attention mechanism to figure out relative importance within the topological structure and along different time steps can reduce the 5.0s ADE/FDE by 21.3%/25.0%, 27.9%/21.5% and 7.8%/8.0% in RA, UI and HR scenarios, respectively. The improvement in RA and UI are more significant due to frequent interactions.

- $\mathbf{T}+\mathbf{C}$ versus $\mathbf{T}+\mathbf{C}+\mathbf{K}$: We show the effectiveness of the kinematic constraint layer. Different from $\mathbf{T}$ which directly output position displacement, the outputs of the GRU unit in $\mathbf{T}+\mathbf{C}+\mathbf{K}$ are control actions, which are aggregated by the bicycle model to obtain complete trajectories. According to Table III, employing the kinematic constraint layer to regularize the learning-based prediction hypotheses can further reduce the 5.0s ADE/FDE by 20.0%/13.7%, 12.5%/16.0% and 9.5%/9.3% in RA, UI and HR scenarios, respectively. Due to the restriction from the kinematic model, unfeasible movements

can be filtered out and the model is unlikely to overfit noisy data or outliers. Moreover, the improvement in RA and UI is more significant than in HR. The reason is that most vehicles go straight along the road in HR, whose behaviors can be well approximated by linear models. However, there are frequent turning behaviors in RA and UI which need constraints by more sophisticated models. We also visualize the predicted trajectories in Figure 4.7, where the ones in $\mathbf{T} + \mathbf{C} + \mathbf{K}$ are smoother and more plausible.

4.7 Chapter Summary

In this chapter, we propose a generic system for multi-agent trajectory prediction named STG-DAT, which considers context information, trajectories of heterogeneous, interactive agents and physical feasibility constraints. In order to effectively model the interactions between different entities, we design a graph dual-attention network to extract features from spatio-temporal dynamic graphs. The Wasserstein generative modeling is employed as the basis of training the whole framework. The STG-DAT is validated by both pedestrian and vehicle trajectory prediction tasks on multiple benchmark datasets. The experimental results show that our approach achieves the state-of-the-art prediction performance compared with multiple baseline methods. Moreover, the proposed prediction model can be easily adopted by multi-target tracking frameworks, which empirically proves to enhance tracking accuracy.

Chapter 5

EvolveGraph: Dynamic Relational Reasoning with Evolving Interaction Graphs

5.1 Introduction

Multi-agent trajectory prediction is critical in many real-world applications, such as autonomous driving, mobile robot navigation and other areas where a group of entities interact with each other, giving rise to complicated behavior patterns at the level of both individuals and the multi-agent system as a whole. Since usually only the trajectories of individual entities are available without any knowledge of the underlying interaction patterns, and there are usually multiple possible modalities for each agent, it is challenging to model such dynamics and forecast their future behaviors.

There have been a number of existing works trying to provide a systematic solution to multi-agent interaction modeling. Some related techniques include, but not limited to social pooling layers [3], attention mechanisms [179, 80, 58, 168, 93], message passing over graphs [47, 154], etc. These techniques can be summarized as implicit interaction modeling by information aggregation. Another line of research is to explicitly perform inference over the structure of the latent interaction graph, which allows for relational structures with multiple interaction types [77, 4]. Our approach falls into this category but with significant extension and performance enhancement over existing methods.

A closely related work is NRI [77], in which the interaction graph is static with homogeneous nodes during training. This is sufficient for the systems involving homogeneous type of agents with fixed interaction patterns. In many real-world scenarios, however, the underlying interactions are inherently varying even with abrupt changes (e.g. basketball players). And there may be heterogeneous types of agents (e.g. cars, pedestrians, cyclists, etc.) involved in the system, while NRI cannot distinguish them explicitly. Moreover, NRI does not deal with the multi-modality explicitly in future system behaviors. In this work, we address the

problems of

- extracting the underlying interaction patterns with a latent graph structure, which is able to handle different types of agents in a unified way;
- capturing the dynamics of interaction graph evolution for dynamic relational reasoning;
- predicting future trajectories (state sequences) based on the historical observations and the latent interaction graph;
- capturing the multi-modality of future system behaviors.

The main contributions of this work are summarized as:

- We propose a generic trajectory forecasting framework with explicit interaction modeling via a latent graph among multiple heterogeneous, interactive agents. Both trajectory information and context information (e.g., scene images, semantic maps, point cloud density maps) can be incorporated into the system.
- We propose a dynamic mechanism to evolve the underlying interaction graph adaptively along time, which captures the dynamics of interaction patterns among multiple agents. We also introduce a double-stage training pipeline which not only improves training efficiency and accelerates convergence, but also enhances model performance in terms of prediction accuracy.
- The proposed framework is designed to capture the uncertainty and multi-modality of future trajectories in nature from multiple aspects.
- We validate the proposed framework on both synthetic simulations and trajectory forecasting benchmarks in different areas. Our EvolveGraph achieves the state-of-the-art performance consistently.

5.2 Related Work

The problem of multi-agent trajectory prediction has been considered as modeling behaviors among a group of interactive agents. Social forces was introduced by [53] to model the attractive and repulsive motion of humans with respect to the neighbors. Some other learning-based approaches were proposed, such as hidden Markov models [100, 206], dynamic Bayesian networks [71], inverse reinforcement learning [172]. In recent years, the conceptual extension has been made to better model social behavior with supplemental cues such as motion patterns [208, 200] and group attributes [198]. Such social models have motivated the recent data-driven methods in [3, 89, 45, 196, 49, 207, 56, 117, 14, 212, 122, 151, 170, 146, 93, 38, 61, 125, 120]. They encode the motion history of individual entities using the recurrent operation of neural networks. However, it is nontrivial for these methods to find acceptable

future motions in heterogeneous and interactively changing environments, partly due to their heuristic feature pooling or aggregation, which may not be sufficient for dynamic interaction modeling.

Interaction modeling and relational reasoning have been widely studied in various fields. Recently, deep neural networks applied to graph structures have been employed to formulate a connection between interactive agents or variables [179, 122, 80, 153, 210, 12]. These methods introduce nodes to represent interactive agents and edges to express their interactions with each other. They directly learn the evolving dynamics of node attributes (agents' states) and/or edge attributes (relations between agents) by constructing spatio-temporal graphs. However, their models have no explicit knowledge about the underlying interaction patterns. Some existing works (e.g. NRI [77]) have taken a step forward towards explicit relational reasoning by inferring a latent interaction graph. However, it is nontrivial for NRI to deal with heterogeneous agents, context information and the systems with varying interactions. In this work, we present an effective solution to handle aforementioned issues. Our work is also related to learning on dynamic graphs. Most existing works studied representation learning on dynamically evolving graphs [138, 72], while we attempt to predict evolution of the graph.

5.3 Problem Formulation

We assume that, without loss of generality, there are N homogeneous or heterogeneous agents in the scene, which belongs to M (≥ 1) categories (e.g. cars, cyclists, pedestrians). The number of agents may vary in different cases. We denote a set of state sequences covering the historical and forecasting horizons (T_h and T_f) as $\mathbf{X}_{1:T} = \{\mathbf{x}^i_{1:T}, T = T_h + T_f, i = 1, ..., N\}$. We also denote a sequence of historical context information as $\mathbf{C}_{1:T_h} = \{\mathbf{c}_{1:T_h}\}$ for dynamic scenes or fixed context information $\mathbf{C}$ for static scenes. In the scope of this paper, we define $\mathbf{x}^i_t = (x^i_t, y^i_t)$, where (x, y) is the 2D coordinate in the world space or image pixel space. The context information includes images or tensors which represent attributes of the scene. We denote the latent interaction graph as $\mathcal{G}_\beta$, where β is the graph index. We aim to estimate $p(\mathbf{X}_{T_h+1:T_h+T_f}|\mathbf{X}_{1:T_h}, \mathbf{C}_{1:T_h})$ for dynamic scenes or $p(\mathbf{X}_{T_h+1:T_h+T_f}|\mathbf{X}_{1:T_h}, \mathbf{C})$ for static scenes. For simplicity, we use $\mathbf{C}$ when referring to the context information in the equations. More formally, if the latent interaction graph is inferred at each time step, then we have the factorization of $p(\mathbf{X}_{T_h+1:T_h+T_f}|\mathbf{X}_{1:T_h}, \mathbf{C})$ below:

$$\int_{\mathcal{G}} p(\mathcal{G}_0|\mathbf{X}_{1:T_h}, \mathbf{C}) p(\mathbf{X}_{T_h+1}|\mathcal{G}_0, \mathbf{X}_{1:T_h}, \mathbf{C}) \prod_{\beta=1}^{T_f-1} p(\mathcal{G}_\beta|\mathcal{G}_{0:\beta-1}, \mathbf{X}_{1:T_h+\beta}, \mathbf{C}) p(\mathbf{X}_{T_h+\beta+1}|\mathcal{G}_{0:\beta}, \mathbf{X}_{1:T_h+\beta}, \mathbf{C}).$$

5.4 Model Design: EvolveGraph

An illustrative graphical model is shown in Figure 5.1 (left part) to demonstrate the essential procedures of the prediction framework with explicit dynamic relational reasoning. Instead of

Figure 5.1: (a) The left part is a high-level graphical illustration of the proposed approach, where the encoding horizon and decoding horizons (re-encoding gap) are both set to 5. $\mathbf{X}_t$ denotes the state of all the agents at time t, $\Delta\mathbf{X}_t$ denotes the change in state, and $\mathbf{C}$ denotes context information. $\mathcal{G}_\beta$ denotes the latent interaction graph obtained from the static encoding process, and $\mathcal{G}'_\beta$ denotes the adjusted interaction graph with time dependence. At each encoding-decoding iteration, $\mathcal{G}_\beta$ is obtained through the encoding of previous trajectories and context information, which goes through a recurrent unit ($\mathbf{H}_\beta$) to get the adjusted interaction graph $\mathcal{G}'_\beta$. The previous trajectories and $\mathcal{G}'_\beta$ are combined as the input of the decoding process, which generates distributions of state changes to get future trajectories. (b) The right part is an illustration of observation graph and interaction graph. In the observation graph, the edges between agent nodes are homogeneous and bidirectional, while in the interaction graph, colored edges are unidirectional with a certain type.

end-to-end training in a single pipeline, our training process contains two consecutive stages:

- **Static interaction graph learning**: A series of encoding functions are trained to extract interaction patterns from the observed trajectories and context information, and generate a distribution of static latent interaction graphs. A series of decoding functions are trained to recurrently generate multi-modal distributions of future states. At this stage, the prediction is only based on the static interaction graph inferred from the history information, which means the encoding process is only applied once and the interaction graph does not evolve with the decoding process.
- **Dynamic interaction graph learning**: The pre-trained encoding and decoding functions at the first stage are utilized as an initialization, which are finetuned together with the training of a recurrent network which captures the dynamics of interaction graph evolution. The graph recurrent network serves as a high-level integration which considers the dependency of the current interaction graph on previous ones. At this stage, the prediction is based on the latest updated interaction graph.

5.4.1 Static interaction graph learning

Observation Graph

A fully-connected graph without self-loops is constructed to represent the observed information with node/edge attributes, which is called observation graph. Assume that there are N heterogeneous agents in the scene, which belongs to M categories. Then the observation graph consists of N agent nodes and one context node. Agent nodes are bidirectionally connected to each other, and the context node only have outgoing edges to each agent node. We denote an observation graph as $\mathcal{G}_{obs} = \{\mathcal{V}_{obs}, \mathcal{E}_{obs}\}$, where $\mathcal{V}_{obs} = \{\mathbf{v}_i, i \in \{1, ..., N\}\} \cup \{\mathbf{v}_c\}$ and $\mathcal{E}_{obs} = \{\mathbf{e}_{ij}, i, j \in \{1, ..., N\}\} \cup \{\mathbf{e}_{ic}, i \in \{1, ..., N\}\}$. $\mathbf{v}_i$, $\mathbf{v}_c$ and $\mathbf{e}_{ij}$, $\mathbf{e}_{ic}$ denote agent node attribute, context node attribute and agent-agent, context-agent edge attribute, respectively. More specifically, the $\mathbf{e}_{ij}$ denotes the attribute of the edge from node j to node i. Each agent node has two types of attributes: *self-attribute* and *social-attribute*. The former only contains the node's own state information, while the latter only contains other nodes' state information. The calculations of node/edge attributes are given by

$$\mathbf{v}_i^{\text{self}} = f_a^m(\mathbf{x}_{1:T_h}^i),\ i \in \{1, ..., N\},\ m \in \{1, ..., M\}, \tag{5.1}$$

$$\mathbf{v}_c = f_c(\mathbf{c}_{1:T_h}) \quad \text{or} \quad \mathbf{v}_c = f_c(\mathbf{c}), \tag{5.2}$$

$$\mathbf{e}_{ij}^1 = f_e^1([\mathbf{v}_i^{\text{self}},\ \mathbf{v}_j^{\text{self}}]),\ \mathbf{e}_{ic}^1 = f_{ec}^1([\mathbf{v}_i^{\text{self}},\ \mathbf{v}_c]), \tag{5.3}$$

$$\mathbf{v}_i^{\text{social-1}} = f_v^1([\textstyle\sum_{i \neq j} \alpha_{ij} \mathbf{e}_{ij}^1,\ \mathbf{e}_{ic}^1]),\ \textstyle\sum_{i \neq j} \alpha_{ij} = 1, \tag{5.4}$$

$$\mathbf{v}_i^1 = [\mathbf{v}_i^{\text{self}}, \mathbf{v}_i^{\text{social-1}}], \quad \mathbf{e}_{ij}^2 = f_e^2([\mathbf{v}_i^1,\ \mathbf{v}_j^1]), \tag{5.5}$$

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}\mathbf{v}_i || \mathbf{W}\mathbf{v}_j])\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}\mathbf{v}_i || \mathbf{W}\mathbf{v}_k])\right)}, \tag{5.6}$$

where α_{ij} are learnable attention coefficients computed similar as [178], $f_a^m(\cdot)$, $f_c(\cdot)$ are agent, context node embedding functions, and $f_e(\cdot)$, $f_{ec}(\cdot)$ and $f_v(\cdot)$ are agent-agent edge, agent-context edge, and agent node update functions, respectively. Different types of nodes (agents) use different embedding functions. Note that the attributes of the context node are never updated and the edge attributes only serve as intermediates for the update of agent node attributes. These $f(\cdot)$ functions are implemented by deep networks with proper architectures. At this time, we obtain a complete set of node/edge attributes which include the information of direct (first-order) interaction. The higher-order interactions can be modeled by multiple loops of equations (5.4)-(5.6), in which the social node attributes and edge attributes are updated by turns. Note that the *self-attribute* is fixed in the whole process.

Interaction Graph

The interaction graph represents interaction patterns with a distribution of edge types for each edge, which is built on top of the observation graph. We set a hyperparameter L to denote the number of possible edge types (interaction types) between pairwise agent nodes to model *agent-agent* interactions. Also, there is another edge type that is shared between the context node and all agent nodes to model *agent-context* interactions. Note that "no edge" can also be treated as a special edge type, which implies that there is no message passing along such edges. More formally, the interaction graph is a discrete probability distribution $q(\mathcal{G}|\mathbf{X}_{1:T_h}, \mathbf{C}_{1:T_h})$ or $q(\mathcal{G}|\mathbf{X}_{1:T_h}, \mathbf{C})$, where $\mathcal{G} = \{\mathbf{z}_{ij}, i, j \in \{1, ..., N\}\} \cup \{\mathbf{z}_{ic}, i \in \{1, ..., N\}\}$ is a set of interaction types for all the edges, and $\mathbf{z}_{ij}$ and $\mathbf{z}_{ic}$ are random variables to indicate pairwise interaction types for a specific edge.

Encoding

The goal of the encoding process is to infer a latent interaction graph from the observation graph, which is essentially a multi-class edge classification task. We employ a softmax function with a continuous approximation of the discrete distribution [123] on the last updated edge attributes to obtain the probability of each edge type, which is given by

$$q(\mathbf{z}_{ij}|\mathbf{X}_{1:T_h}, \mathbf{C}) = \text{Softmax}((\mathbf{e}_{ij}^2 + \mathbf{g})/\tau), \ i, j \in \{1, ..., N\}, \tag{5.7}$$

where $\mathbf{g}$ is a vector of independent and identically distributed samples drawn from Gumbel$(0, 1)$ distribution and τ is the Softmax temperature, which controls the sample smoothness. We also use the repramatrization trick to obtain gradients for backpropagation. The edge type between context node and agent nodes $\mathbf{z}_{ic}$, without loss of generality, is hard-coded with probability one. For simplicity, we summarize all the operations in the observation graph and the encoding process as $q(\mathbf{z}|\mathbf{X}_{1:T_h}, \mathbf{C}) = f_{\text{enc}}(\mathbf{X}_{1:T_h}, \mathbf{C})$, which gives a factorized distribution of $\mathbf{z}_{ij}$.

Decoding

Since in many real-world applications the state of agents has long-term dependence, a recurrent decoding process is applied to the interaction graph and observation graph to approximate the distribution of future trajectories $p(\mathbf{X}_{T_h+1:T_h+T_f}|\mathcal{G}, \mathbf{X}_{1:T_h}, \mathbf{C})$. The output at each time step is a Gaussian mixture distribution with K components, where the covariance of each Gaussian component is manually set equal. The detailed operations in the decoding process consists of two stages: burn-in stage ($1 \leq t \leq T_h$) and prediction stage ($T_h+1 \leq t \leq T_h+T_f$), which are given by

$$\tilde{\mathbf{e}}_t^{ij} = \sum\nolimits_{l=1}^{L} z_{ij,l} \tilde{f}_e^l([\tilde{\mathbf{h}}_t^i,\ \tilde{\mathbf{h}}_t^j]), \quad \mathrm{MSG}_t^i = \sum\nolimits_{j \neq i} \tilde{\mathbf{e}}_t^{ij}, \tag{5.8}$$

- $1 \leq t \leq T_h$ (Burn-in stage):

$$\tilde{\mathbf{h}}_{t+1}^i = \mathrm{GRU}^i([\mathrm{MSG}_t^i, \mathbf{x}_t^i, \mathbf{v}_c], \tilde{\mathbf{h}}_t^i), \quad w_{t+1}^{i,k} = f_{weight}^k(\tilde{\mathbf{h}}_{t+1}^i), \tag{5.9}$$

$$\boldsymbol{\mu}_{t+1}^{i,k} = \mathbf{x}_t^i + f_{\mathrm{out}}^k(\tilde{\mathbf{h}}_{t+1}^i), \quad p(\hat{\mathbf{x}}_{t+1}^i|\mathbf{z}, \mathbf{x}_{1:t}^i, \mathbf{c}) = \sum\nolimits_{k=1}^{K} w_{t+1}^{i,k} \mathcal{N}(\boldsymbol{\mu}_{t+1}^{i,k}, \sigma^2\mathbf{I}), \tag{5.10}$$

- $T_h+1 \leq t \leq T_h+T_f$ (Prediction stage):

$$\tilde{\mathbf{h}}_{t+1}^i = \mathrm{GRU}^i([\mathrm{MSG}_t^i, \hat{\mathbf{x}}_t^i, \mathbf{v}_c], \tilde{\mathbf{h}}_t^i),\ w_{t+1}^{i,k} = f_{weight}^k(\tilde{\mathbf{h}}_{t+1}^i),\ \boldsymbol{\mu}_{t+1}^{i,k} = \hat{\mathbf{x}}_t^i + f_{\mathrm{out}}^k(\tilde{\mathbf{h}}_{t+1}^i), \tag{5.11}$$

$$p(\hat{\mathbf{x}}_{t+1}^i|\mathbf{z}, \hat{\mathbf{x}}_{T_h+1:t}^i, \mathbf{x}_{1:T_h}^i, \mathbf{c}) = \sum\nolimits_{k=1}^{K} w_{t+1}^{i,k} \mathcal{N}(\boldsymbol{\mu}_{t+1}^{i,k}, \sigma^2\mathbf{I}), \tag{5.12}$$

$$\text{Sample a Gaussian component from the mixture based on } \mathbf{w}_{t+1}^i, \quad \text{Set } \hat{\mathbf{x}}_{t+1}^i = \boldsymbol{\mu}_{t+1}^{i,k}, \tag{5.13}$$

where MSG is a symbolic acronym for "message" here without specific meanings, $\tilde{\mathbf{h}}_t^i$ is the hidden state of GRU^i at time t, $w_{t+1}^{i,k}$ is the weight of the kth Gaussian distribution at time step $t+1$ for agent i. $\tilde{f}_e^l(\cdot)$ is the edge update function of edge type l, $f_{weight}^k(\cdot)$ is a mapping function to get the weight of the kth Gaussian distribution, and $f_{\mathrm{out}}^k(\cdot)$ is a mapping function to get the mean of the kth Gaussian component. Note that the predicted $\hat{\mathbf{x}}_t^i$ is needed in equation (5.11). In the previous decoding step, we only have its corresponding distribution $p(\hat{\mathbf{x}}_t^i|\mathbf{z}, \hat{\mathbf{x}}_{T_h+1:t-1}^i, \mathbf{x}_{1:T_h}^i, \mathbf{c})$ from the previous step. We first sample a Gaussian component from the mixture based on the component weights w_{t+1}^i. Say we get the kth component, then we set $\hat{\mathbf{x}}_t^j$ as $\boldsymbol{\mu}_t^{j,k}$, which is the trajectory with maximum likelihood within this component. We fix the covariance σ as a constant. The nodes (agents) of the same type share the same GRU decoder. During the burn-in stage, the ground-truth states are used; while during the prediction stage, the state prediction hypotheses are used as the input at the next time step iteratively. For simplicity, the whole decoding process is summarized as $p(\mathbf{X}_{T_h+1:T_h+T_f}|\mathcal{G}, \mathbf{X}_{1:T_h}, \mathbf{C}) = f_{\mathrm{dec}}(\mathcal{G}, \mathbf{X}_{1:T_h}, \mathbf{C})$.

We provide an illustrative diagram of the decoding process, which is shown in Figure 5.2. In this figure, without loss of generality we demonstrate the decoding process for only one node in a five-node observation graph to illustrate how the decoding process works. Figure

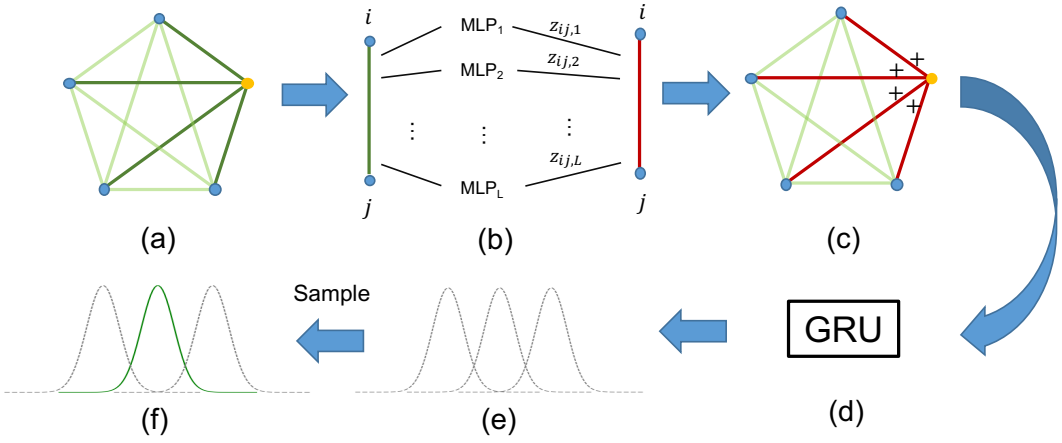


Figure 5.2: An illustrative diagram of the decoding process.

5.2(a) shows the observation graph, we choose the node on the right as an example. Figure 5.2(b) shows the process of using MLPs to process a specific edge, where $z_{ij,l}, l = 1, 2, ..., L$ denotes the probability of the edge belonging to a certain edge type l. The processed edges are shown in red. Figure 5.2(c) shows the sum over every incoming edge attribute of this node. Then we input the result into the decoding GRU. The decoding GRU outputs several Gaussian components and their corresponding weights. We sample one specific Gaussian component based on the weights. Then we use the $\boldsymbol{\mu}$ of the sampled Gaussian distribution as the output state at this step. $\boldsymbol{\mu}$ is used as the input into the next decoding step (if it's not the burn-in step). We iterate the decoding process several times until the desired prediction horizon is reached.

5.4.2 Dynamic Interaction Graph

In many applications, the interaction patterns recognized from the past time steps are likely not static in the future. Instead, they are rather dynamically evolving throughout the future time steps. Moreover, many interaction systems have multi-modal properties in its nature. Different modalities afterwards are likely to result in different interaction patterns. A single static interaction graph is neither sufficiently flexible to model dynamically changing situations (especially those with abrupt changes), nor to capture all the modalities. Therefore, we introduce an effective dynamic mechanism to evolve the interaction graph.

The encoding process is repeated every τ (re-encoding gap) time steps to obtain the latent interaction graph based on the latest observation graph. Since the new interaction graph also has dependence on previous ones, we also need to consider their effects. Therefore, a recurrent unit (GRU) is utilized to maintain and propagate the history information, as well

as adjust the prior interaction graphs. More formally, the calculations are given by

$$q(\mathbf{z}_\beta|\mathbf{X}_{1+\beta\tau:T_h+\beta\tau}, \mathbf{C}) = f_{\text{enc}}(\mathbf{X}_{1+\beta\tau:T_h+\beta\tau}, \mathbf{C}), \tag{5.14}$$

$$q(\mathbf{z}'_\beta|\mathbf{X}_{1+\beta\tau:T_h+\beta\tau}, \mathbf{C}) = \text{GRU}(q(\mathbf{z}_\beta|\mathbf{X}_{1+\beta\tau:T_h+\beta\tau}, \mathbf{C}), \mathbf{H}_\beta) \tag{5.15}$$

where β is the re-encoding index starting from 0, $\mathbf{z}_\beta$ is the interaction graph obtained from the static encoding process, $\mathbf{z}'_\beta$ is the adjusted interaction graph with time dependence, and $\mathbf{H}_\beta$ is the hidden state of the graph evolution GRU. After obtaining $\mathcal{G}'_\beta = \{\mathbf{z}'_\beta\}$, the decoding process is applied to get the states of the next τ time steps,

$$p(\mathbf{X}_{T_h+\beta\tau+1:T_h+(\beta+1)\tau}|\mathcal{G}'_\beta, \mathbf{X}_{1:T_h}, \hat{\mathbf{X}}_{T_h+1:T_h+\beta\tau}, \mathbf{C}) = f_{\text{dec}}(\mathcal{G}'_\beta, \mathbf{X}_{1:T_h}, \hat{\mathbf{X}}_{T_h+1:T_h+\beta\tau}, \mathbf{C}). \tag{5.16}$$

The decoding and re-encoding processes are iterated to obtain the distribution of future trajectories.

5.4.3 Uncertainty and Multi-Modality

Here we emphasize the efforts to encourage diverse and multi-modal trajectory prediction and generation. In our framework, the uncertainty and multi-modality mainly come from three aspects. First, in the decoding process, we output Gaussian mixture distributions indicating that there are several possible modalities at the next step. We only sample a single Gaussian component at each step based on the component weights which indicate the probability of each modality. Second, different sampled trajectories will lead to different interaction graph evolution. Evolution of interaction graphs contributes to the multi-modality of future behaviors, since different underlying relational structures enforce different regulations on the system behavior and lead to various outcomes. Third, directly training such a model, however, tends to collapse to a single mode. Therefore, we employ an effective mechanism to mitigate the mode collapse issue and encourage multi-modality. During training, we run the decoding process d times, which generates d trajectories for each agent under specific scenarios. We only choose the prediction hypothesis with the minimal loss for backpropagation, which is the most likely to be in the same mode as the ground truth. The other prediction hypotheses may have much higher loss, but it doesn't necessarily imply that they are implausible. They may represent other potential reasonable modalities.

5.4.4 Loss Function and Training

In our experiments, we first train the encoding / decoding functions using a static interaction graph. Then in the process of training dynamic interaction graph, we use the pre-trained encoding / decoding functions at the first stage to initialize the parameters of the modules used in the dynamic training. This step is reasonable since the encoding / decoding functions used in these two training process play similar roles and their optima are supposed to be close. And if we train dynamic graphs directly, it will lead to longer convergence time and is likely to be trapped into some bad local optima due to large number of learnable parameters.

It is possible that this method may accelerate the whole training process and avoid some bad local optima.

In the training process, our loss function is defined as follows:

$$\mathcal{L}_S = -\mathbb{E}_{q(\mathbf{z}|\mathbf{X}_{1:T_h},\mathbf{C})}\left[\sum_{i=1}^{N}\sum_{t=T_h+1}^{T_h+T_f}\sum_{k=1}^{K} w_t^{i,k}\log p_t^{i,k}(\mathbf{x}_t|\mathbf{z},\mathbf{X}_{1:T_h},\hat{\mathbf{X}}_{T_h+1:t-1},\mathbf{C})\right] \text{ (Static)}, \tag{5.17}$$

$$\mathcal{L}_D = -\mathbb{E}_{q(\mathbf{z}'_{\beta(t)}|\mathbf{X}_{1+\beta\tau:T_h+\beta\tau},\mathbf{C})}\left[\sum_{i=1}^{N}\sum_{t=T_h+1}^{T_h+T_f}\sum_{k=1}^{K} w_t^{i,k}\log p_t^{i,k}(\mathbf{x}_t|\mathbf{z}'_{\beta(t)},\mathbf{X}_{1:T_h},\hat{\mathbf{X}}_{T_h+1:t-1},\mathbf{C})\right] \text{ (Dynamic)}, \tag{5.18}$$

where $q(\cdot)$ denotes the encoding and re-encoding operations, which return a factorized distribution of $\mathbf{z}_{ij}$ or $\mathbf{z}'_{ij}$. The $p_t^{i,k}(\mathbf{x}_t|\mathbf{z},\mathbf{X}_{1:T_h},\hat{\mathbf{X}}_{T_h+1:t-1},\mathbf{C})$ and $p_t^{i,k}(\mathbf{x}_t|\mathbf{z}'_{\beta(t)},\mathbf{X}_{1:T_h},\hat{\mathbf{X}}_{T_h+1:t-1},\mathbf{C})$ denote a certain Gaussian distribution.

5.5 Experiments

In this work, we validated the proposed framework EvolveGraph on one synthetic dataset and three benchmark datasets for real-world applications: Honda 3D Dataset (H3D) [141], NBA SportVU Dataset (NBA), and Stanford Drone Dataset (SDD) [148].

For the synthetic dataset, since we have access to the ground truth of the underlying interaction graph, we quantitatively and qualitatively evaluate the model performance in terms of both interaction (edge type) recognition and average state prediction error. For the benchmark datasets, we evaluate the model performance in terms of two widely used standard metrics: minimum average displacement error (minADE_{20}) and minimum final displacement error (minFDE_{20}) [14]. The minADE_{20} is defined as the minimum average distance between the 20 predicted trajectories and the ground truth over all the involved entities within the prediction horizon. The minFDE_{20} is defined as the minimum deviated distance of 20 predicted trajectories at the last predicted time step. We also provide ablative analysis (right part of Table 2-4), analysis on double-stage training, analysis on the selection of edge types and re-encoding gap, and additional qualitative results in supplementary materials.

5.5.1 Datasets

Synthetic Particle Simulations

We designed a synthetic particle simulation to validate the performance of our model. In this simulation, we have n particles in an x-y plane and all the locations of particles are randomly initialized on the $y > 0$ half plane. The movement of these particles contains two phases, corresponding to two interaction graphs. Initially, particles are rigidly connected to each other and form a "star" shape. More specifically, there is a virtual centroid and each particle is rigidly connected to the centroid. it is equivalent to using a stick to connect the particle

and the virtual centroid. And the distance between a certain particle and the centroid keeps the same. The particles are uniformly distributed around the centroid, which means the angle between two adjacent "sticks" is $\frac{2\pi}{n}$. In the first phase, particles move as a whole, with both translational and rotational motions. The velocity and angular velocity of the whole system are randomly initialized in a certain range. Once any one of the particles reaches $y = 0$ (the switching criterion), the movement of all the particles will transfer to the second phase. In the second phase, particles are no longer connected to each other. The motion of one particle will by no means affect the motion of the others. In other words, each particle keeps uniform linear motions once the second phase begins. We generated 50k sample in total for training, validation and testing.

Benchmark Datasets

- H3D [141]: A large scale full-surround 3D multi-object detection and tracking dataset, which provides point cloud information and trajectory annotations for heterogeneous traffic participants (e.g. cars, trucks, cyclists and pedestrians). We selected 90k samples in total for training, validation and testing.
- NBA: A trajectory dataset collected by NBA with the SportVU tracking system, which contains the trajectory information of all the ten players and the ball in real games. We randomly selected 50k samples in total for training, validation and testing.
- SDD [148]: A trajectory dataset containing a set of top-down-view images and the corresponding trajectories of involved entities, which was collected in multiple scenarios in a university campus full of interactive pedestrians, cyclists and vehicles. We randomly selected 50k samples in total for training, validation and testing.

5.5.2 Baseline Methods

We compared the performance of our proposed approach with the following baseline methods. Please refer to the reference papers for more details.

For Synthetic Particle Simulations

- Corr. (LSTM): The baseline method for edge prediction in [77].
- LSTM (single) / LSTM (joint): The baseline methods for state sequence prediction in [77].
- NRI (static): The NRI model with static latent graph [77].
- NRI (dynamic): The NRI model with latent graph re-evaluation at each time step [77].

For Benchmark Datasets

- Social-LSTM [3]: The model encodes the trajectories with an LSTM layer whose hidden states serve as the input of a social pooling layer.
- Social-GAN [45]: The model introduces generative adversarial learning scheme into S-LSTM to improve performance.
- Social-Attention [179]: The model deals with spatio-temporal graphs with recurrent neural networks, which is based on the architecture of Structural-RNN [66].
- Gated-RN [19]: The model infers relational behavior between road users and the surrounding environment by extracting spatio-temporal features.
- Trajectron++ [152]: The approach represents a scene as a directed spatio-temporal graph and extract features related to the interaction. The whole framework is based on conditional variational auto-encoder.
- NRI [77]: The model is formulated as a variational inference task with an encoder-decoder structure. This is the most related work.
- STGAT [62]: The model is a variant of graph attention network, which is applied to spatio-temporal graphs.
- Social-STGCNN [132]: The model is a variant of graph convolutional neural network, which is applied to spatio-temporal graphs.

5.5.3 Implementation Details

For all the experiments, a batch size of 32 was used and the models were trained for up to 20 epochs during the static graph learning stage and up to 100 epochs during the dynamic graph learning stage with early stopping. We used Adam optimizer with an initial learning rate of 0.001. The models were trained on a single TITAN Xp GPU. We used a split of 65%, 10%, 25% as training, validation and testing data.

Specific details of model components are introduced below:

- **Agent node embedding function**: for each different node type, a distinct two-layer gated recurrent unit (GRU) with hidden size = 128.
- **Context node embedding function**: four-layer convolutional blocks with kernel size = 5 and padding = 3. The structure is [[Conv, ReLU, Conv, ReLU, Pool], [Conv, ReLU, Conv, ReLU, Pool]].
- **Agent node update function**: a three-layer MLP with hidden size = 128.

Table 5.1: Comparison of Accuracy (Mean ± Std in %) of Interaction (Edge Type) Recognition.

	Corr. (LSTM)	NRI (dynamic)	**EvolveGraph** (static)	**EvolveGraph** (RNN re-encoding)	**EvolveGraph** (dynamic)	Supervised
No Change	63.2±0.9	91.3±0.3	**95.6**±0.2	91.4±0.3	93.8±1.1	98.1±0.4
Change	—	71.5±3.1	64.1±0.8	75.2±1.4	**82.3**±3.2	94.3±1.5

- **Edge update function**: for both agent-agent edges and agent-context edges, a distinct three-layer MLP with hidden size = 128.
- **Encoding function**: a three-layer MLP with hidden size = 128.
- **Decoding function**: a two-layer gated recurrent unit (GRU) with hidden size = 128.
- **Recurrent graph evolution module**: a two-layer GRU with hidden size = 256.

Specific experimental details of different datasets are introduced below:

- **Synthetic simulations**: 2 edge types, re-encoding gap = 1, encoding horizon = 20.
- **H3D dataset**: 5 edge types, re-encoding gap = 5, encoding horizon = 5.
- **NBA dataset**: 6 edge types, re-encoding gap = 4, encoding horizon = 5.
- **SDD dataset**: 4 edge types, re-encoding gap = 5, encoding horizon = 5.

5.5.4 Synthetic Simulations: Particle Physics System

We experimented with a simulated particle system with change of relations. Multiple particles are initially linked and move together. The links disappear as long as a certain criterion on particle state is satisfied and the particles move independently thereafter. The model is expected to learn the criterion by itself, and perform edge type prediction and trajectory prediction. Since the system is deterministic in nature, we do not consider multi-modality in this task.

We predicted the particle states at the future 50 time steps based on the observations of 20 time steps. We set two edge types in this task, which correspond to "with link" and "without link". The results of edge type prediction are summarized in Table 1, which are averaged over 3 independent runs. *No Change* means the underlying interaction structure keeps the same in the whole horizon, while *Change* means the change of interaction patterns happens at some time. It shows that the supervised learning baseline, which directly trains the encoding functions with ground truth labels, performs the best in both setups and serves as a "gold standard". Under the *No Change* setup, NRI (dynamic) is comparable to EvolveGraph (RNN re-encoding), while EvolveGraph (static) achieves the best performance. The reason is that dynamic evolution of interaction graph leads to higher flexibility but may result in larger

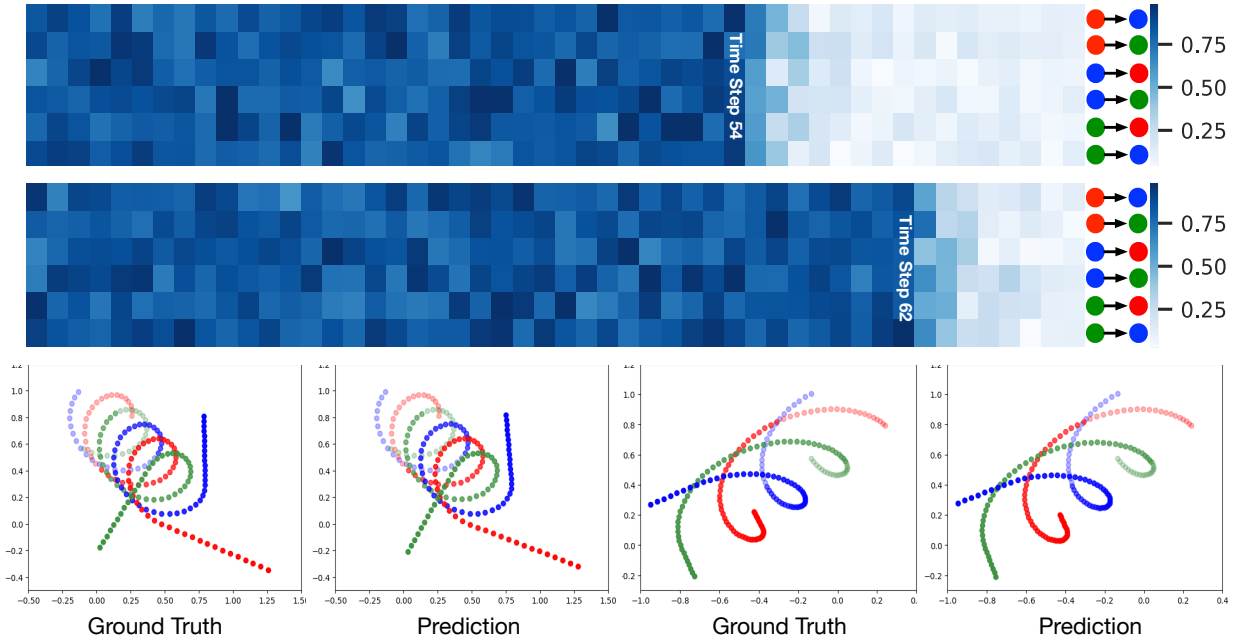


Figure 5.3: Visualization of latent interaction graph evolution and particle trajectories. (a) The top two figures show the probability of the first edge type ("with link") at each time step. Each row corresponds to a certain edge (shown in the right). The actual times of graph evolution are 54 and 62, respectively. The model is able to capture the underlying criterion of relation change and further predict the change of edge types with nearly no delay. (b) The figures in the last row show trajectory prediction results, where semi-transparent dots are historical observations.

uncertainty, which affects edge prediction in the systems with static relational structures. Under the *Change* setup, NRI (dynamic) re-evaluates the latent graph at every time step during the testing phase, but it is hard to capture the dependency between consecutive graphs, and the encoding functions may not be flexible enough to capture the evolution. EvolveGraph (RNN re-encoding) performs better since it considers the dependency of consecutive steps during the training phase, but it still captures the evolution only at the feature level instead of the graph level. EvolveGraph (dynamic) achieves significantly higher accuracy than the other baselines (except Supervised), due to the explicit evolution of interaction graphs.

We also provide visualization of interaction graphs and particle trajectories of random testing cases in Figure 5.3. In the heatmaps, despite that the predicted probabilities fluctuate within a small range at each step, they are very close to the ground truth (1 for "with link" and 0 for "without link"). The change of relation can be quickly captured within two time steps. The results of particle state prediction are shown in Figure 5.4. The standard deviation

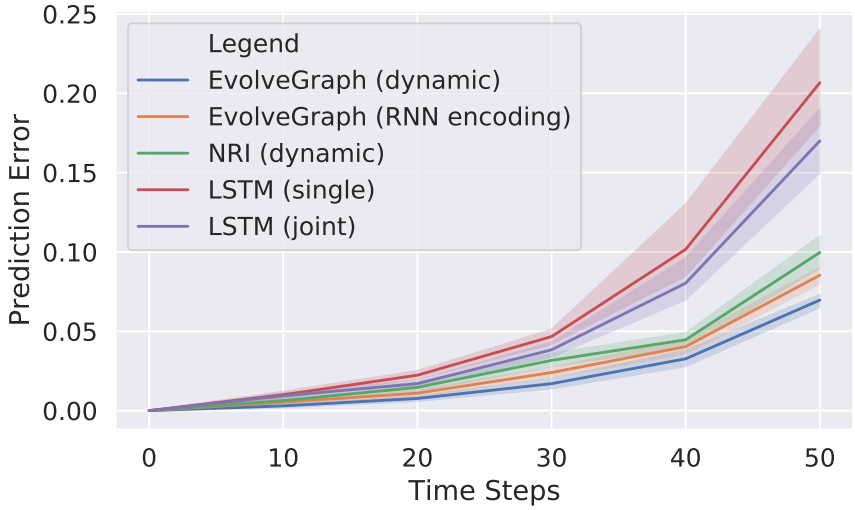


Figure 5.4: Average prediction error of particle state.

was calculated over 3 runs. Within the whole horizon, EvolveGraph (dynamic) consistently outperforms the other baselines with stable performance (small standard deviation).

5.5.5 H3D Dataset: Traffic Scenarios

We predicted the future 10 time steps (4.0s) based on the historical 5 time steps (2.0s). The comparison of quantitative results is shown in Table 5.2, where the unit of reported minADE_{20} and minFDE_{20} is meters in the world coordinates. Note that we included cars, trucks, cyclists and pedestrians in the experiments. All the baseline methods consider the relations and interactions among agents. The Social-Attention employs spatial attention mechanisms, while the Social-GAN demonstrates a deep generative model which learns the data distribution to generate human-like trajectories. The Gated-RN and Trajectron++ both leverage spatio-temporal information to involve relational reasoning, which leads to smaller prediction error. The NRI infers a latent interaction graph and learns the dynamics of agents, which achieves similar performance to Trajectron++. The STGAT and Social-STGCNN further take advantage of the graph neural network to extract relational features in the multi-agent setting. Our proposed method achieves the best performance, which implies the advantages of explicit interaction modeling via evolving interaction graphs. The 4.0s minADE_{20} / minFDE_{20} are *significantly* reduced by 20.0% / 27.1% compared to the best baseline approach (STGAT).

We also provide visualization of results. Figure 5.5(a) and Figure 5.5(b) show two random testing samples from H3D results. We can tell that our framework can generate accurate and plausible trajectories. More specifically, in Figure 5.5(a), for the blue prediction hypothesis

Table 5.2: minADE_{20} / minFDE_{20} (Meters) of Trajectory Prediction (H3D dataset).

Time	Baseline Methods							**EvolveGraph (Ours)**				
	STGAT	Social-Attention	Social-STGCNN	Social-GAN	Gated-RN	Trajectron++	NRI (dynamic)	SG (same node type)	SG	RNN re-encoding	DG (single stage)	DG (double stage)
1.0s	0.24/0.33	0.29/0.45	0.23/0.32	0.27/0.37	**0.18**/0.32	0.21/0.34	0.24/0.30	0.28/0.37	0.27/0.35	0.25/0.32	0.24/0.31	0.19/**0.25**
2.0s	0.34/0.48	0.53/0.96	0.36/0.52	0.45/0.77	0.32/0.64	0.33/0.62	0.32/0.60	0.40/0.58	0.38/0.55	0.35/0.51	0.33/0.46	**0.31**/**0.44**
3.0s	0.46/0.77	0.87/1.62	0.49/0.89	0.68/1.29	0.49/1.03	0.46/0.93	0.48/0.94	0.51/0.80	0.48/0.76	0.44/0.70	0.40/0.60	**0.39**/**0.58**
4.0s	0.60/1.18	1.21/2.56	0.73/1.49	0.94/1.91	0.69/1.56	0.71/1.63	0.73/1.56	0.64/1.21	0.61/1.14	0.57/1.07	0.50/0.90	**0.48**/**0.86**

Table 5.3: minADE_{20} / minFDE_{20} (Meters) of Trajectory Prediction (NBA dataset).

Time	Baseline Methods						**EvolveGraph (Ours)**				
	STGAT	Social-STGCNN	Social-Attention	Social-LSTM	Social-GAN	Trajectron++	NRI (dynamic)	SG (same node type)	SG	RNN re-encoding	DG (single stage)	DG (double stage)
1.0s	0.42/0.71	0.46/0.76	0.87/1.36	0.92/1.34	0.82/1.25	0.55/0.90	0.60/0.87	0.70/1.09	0.59/0.92	0.58/0.89	0.48/0.76	**0.31**/**0.52**
2.0s	0.91/1.39	0.90/1.43	1.58/2.51	1.64/2.74	1.52/2.45	0.99/1.58	1.02/1.71	1.51/2.38	1.38/2.12	1.09/1.88	0.84/1.43	**0.74**/**1.10**
3.0s	1.62/2.87	1.59/2.67	2.78/4.66	2.93/5.03	2.63/4.51	1.89/3.32	1.83/3.15	2.10/3.53	1.88/3.23	1.77/2.87	1.43/2.55	**1.28**/**2.07**
4.0s	2.47/3.86	2.35/3.71	3.76/6.64	4.00/7.12	3.60/6.24	2.62/4.70	2.48/4.30	2.83/4.85	2.52/4.57	2.39/3.89	2.08/3.74	**1.83**/**3.16**

Table 5.4: minADE_{20} / minFDE_{20} (Pixels) of Trajectory Prediction (SDD dataset).

Time	Baseline Methods							**EvolveGraph (Ours)**				
	STGAT	Social-STGCNN	Social-Attention	Social-LSTM	Social-GAN	Trajectron++	NRI (dynamic)	SG (same node type)	SG	RNN re-encoding	DG (single stage)	DG (double stage)
4.8s	18.8/31.3	20.6/33.1	33.3/55.9	31.4/55.6	27.0/43.9	19.3/32.7	25.6/43.7	22.5/40.3	20.6/36.4	18.4/32.1	16.1/26.6	**13.9**/**22.9**

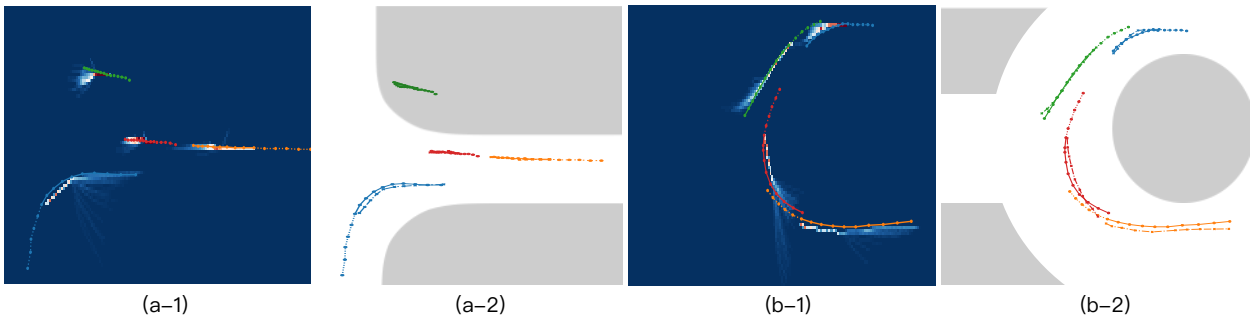


Figure 5.5: Qualitative results of testing cases of H3D dataset. Dashed lines are historical trajectories, solid lines are ground truth, and dash-dotted lines are prediction hypothesis. White areas represent drivable areas and gray areas represent sidewalks. We plotted the prediction hypothesis with the minimal ADE, and the heatmap to represent the distributions. (a) Intersection; (b) Roundabout.

at the left bottom, there is an abrupt change at the fifth prediction step. This is because the interaction graph evolved at this step (Our re-encoding gap τ was set to be 5 in this case). Moreover, in the heatmap, there are multiple possible trajectories starting from this point, which represent multiple potential modalities. These results show that the evolving interaction graph can reinforce the multi-modal property of our model, since different samples of trajectories at the previous steps lead to different directions of graph evolution, which significantly influences the prediction afterwards. In Figure 5.5(b), each car may leave the roundabout at any exit. Our model can successfully show the modalities of exiting the roundabout and staying in it. Moreover, if exiting the roundabout, the cars are predicted to exit on their right, which implies that the modalities predicted by our model are plausible and reasonable.

5.5.6 NBA Dataset: Sports Games

We also predicted the future 10 time steps (4.0s) based on the historical 5 time steps (2.0s). The comparison of quantitative results is shown in Table 5.3, where the unit of reported minADE_{20} and minFDE_{20} is meters in the world coordinates. Note that we included both players and the basketball in the experiments. The players are divided into two different types according to their teams. The basketball players are highly interactive and behaviors often change suddenly due to the reaction to other players. The baselines all consider the relations and interactions among agents with different strategies, such as soft attention mechanisms, social pooling layers, and graph-based representation. Owing to the dynamic interaction modeling by evolving interaction graph, our method achieves *significantly* better performance than state-of-the-art, which reduces the 4.0s minADE_{20} / minFDE_{20} by 22.1% / 18.1% with

respect to the best baseline (Social-STGCNN). Qualitative results and analysis can be found in Section 7.3 in the supplementary materials.

Figure 5.6 show some visualizations of testing results on the NBA dataset. First, we tell that in such cases the ball follows a player at most times, which implies that the predicted results represent plausible situations. Second, most prediction hypotheses are very close to the ground truth, even if some predictions are not similar to the ground truth, they represent a plausible behavior. Third, the heatmaps show that our model can successfully predict most reasonable future trajectories and their multi-modal distributions. More specifically, in the first case of Figure 5.6, for the player of the green team in the middle, the historical steps move forward quickly, while our model can successfully predict that the player will suddenly stop, since he is surrounded by many opponents and he is not carrying the ball. In the second case of Figure 5.6, our model shows that three pairs of players from different teams competing against each other for chances. the defensing team is closer to the basket. and the player carrying the ball is running quickly towards the basket. Two opponents are trying to defend him. Such case is a very common situation in basketball games. In general, not only does our model achieve high accuracy, it can also understand and predict most moving, stopping, offending and defensing behaviors in basketball games.

5.5.7 SDD Dataset: University Campus

We predicted the future 12 time steps (4.8s) based on the historical 8 time steps (3.2s). The comparison of quantitative results is shown in Table 5.4, where the unit of reported minADE_{20} and minFDE_{20} is pixels in the image coordinates. Note that we included all the types of agents (e.g. pedestrians, cyclists, vehicles)in the experiments, although most of them are pedestrians. Our proposed method achieves the best performance. The 4.8s minADE_{20} / minFDE_{20} are reduced by 26.1% / 26.8% compared to the best baseline approach (STGAT).

5.5.8 Ablative Analysis

We conducted ablative analysis on the benchmark datasets to demonstrate the effectiveness of heterogeneous node types, dynamically evolving interaction graph and two-stage graph learning. The best minADE_{20} / minFDE_{20} of each model setting are shown in the right parts of Table 5.2, Table 5.3 and Table 5.4. The detailed descriptions of each model setup are provided as follows:

- **SG (same node type)**: This is the simplest model setting, where only a static interaction graph is extracted based on the history information. The same node embedding function is shared among all the agent nodes.
- **SG**: This setting is similar to the previous one, except that different node embedding functions are applied to different types of agent nodes.

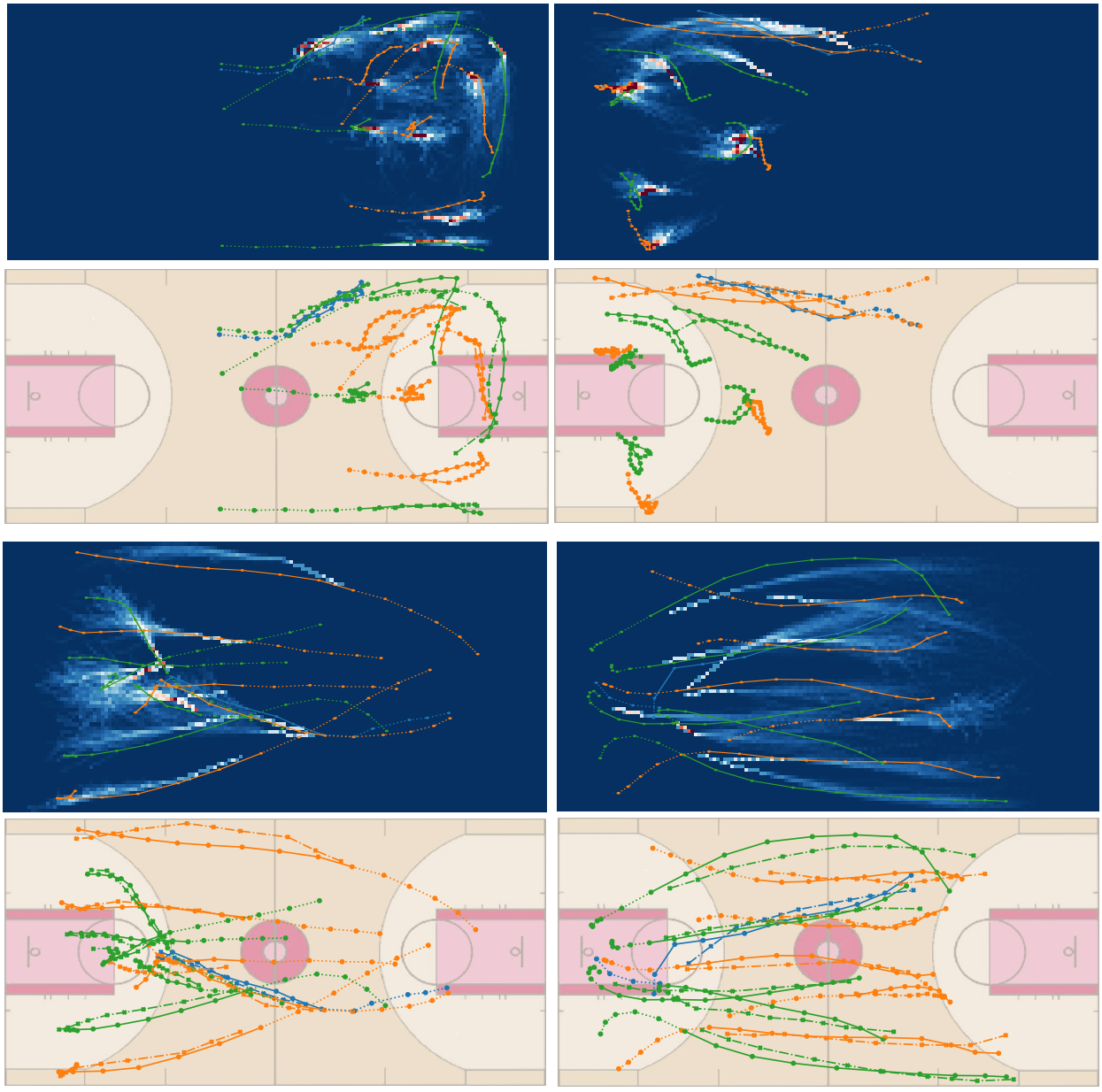


Figure 5.6: Qualitative results of testing cases of the NBA dataset. The upper figures are the visualization of predicted distributions, and the lower figures are the best prediction hypotheses. The line colors indicate teams and blue lines are the trajectories of basketball.

- **RNN re-encoding**: The interaction graph is re-encoded every τ time steps using an RNN encoding process. Note that this is different from our model, since the RNN encoding process only captures evolution of node attributes without explicitly modeling the dependency of consecutive underlying interaction graphs.

- **DG (single stage)**: This is our whole model, where the encoding, decoding functions and the graph evolving GRU are all trained together from scratch.

- **DG (double stage)**: This is our whole model with double stage interaction graph learning, where the encoding, decoding functions trained at the first stage are employed as an initialization in the second stage.

We conducted the ablative analysis in the following perspectives:

- **SG (same node type) v.s. SG**: We show the effectiveness of the distinction of agent node types. According to the prediction results in Table 2, Table 3 and Table 4, utilizing distinct agent-node embedding functions for different agent types achieves consistently smaller minADE_{20} / minFDE_{20} than a universal embedding function. The reason is that different types of agents have distinct behavior patterns or feasibility constraints. For example, the trajectories of on-road vehicles are restricted by roadways, traffic rules and physical constraints, while the restrictions on pedestrian behaviors are much fewer. Moreover, since vehicles usually have to yield pedestrians at intersections, it is helpful to indicate agent types explicitly in the model. With differentiation of agent types, the 4.0s minADE_{20} / minFDE_{20} are reduced by 4.7% / 5.8% on the H3D dataset, 8.6% / 5.8% on the NBA dataset. The 4.8s minADE_{20} / minFDE_{20} are reduced by 8.9% / 9.9% on the SDD dataset.

- **SG v.s. RNN re-encoding v.s. DG (double stage)**: We compare the performance of our method and SG / RNN encoding baselines. It is shown that the improvement of RNN re-encoding is limited and the prediction errors are slightly smaller than SG. Although both RNN re-encoding baseline and our method attempt to capture the potential changes of underlying interaction graph at each time step, our method achieves a significantly smaller prediction error consistently. A potential reason is despite that the RNN re-encoding process is iteratively extracting the patterns from node attributes (agent states), its capability of inferring graph evolution is limited.

- **DG (single stage) v.s. DG (double stage)**: We show the effectiveness and necessity of double-stage dynamic graph learning. It is shown that the double-stage training scheme leads to remarkable improvement in terms of minADE_{20} / minFDE_{20} on all three datasets. During the first training stage, the encoding / decoding functions are well trained to a local optimum, which is able to extract a proper static interaction graph. According to empirical findings, the encoding / decoding functions are sufficiently good as an initialization for the second stage training after several epochs' training. During the second training stage, the encoding / decoding functions are initialized from the

first stage and finetuned, along with the training of graph evolution GRU. This leads to faster convergence and better performance, since it may help avoid some bad local optima at which the loss function may be stuck if all the components are randomly initialized. With the same hyperparameters, the single-stage / double-stage training took about 25 / 14 epochs to reach their smallest validation loss on the NBA dataset and 41 / 26 epochs on the H3D dataset. Compared to single-stage training, the 4.0s minADE_{20} / minFDE_{20} of double-stage training are reduced by 12.0% / 15.5% on the NBA dataset and 9.4% / 12.2% on the H3D dataset. The 4.8s minADE_{20} / minFDE_{20} are reduced by 13.7% / 13.9% on the SDD dataset.

5.5.9 Analysis on Edge Types and Re-encoding Gap

We also provide a comparison of minADE_{20} / minFDE_{20} (in meters) and testing running time on the NBA dataset to demonstrate the effect of different numbers of edge types and re-encoding gaps. In Figure 5.7(a), it is shown that as the number of edge type increases, the prediction error first decreases to a minimum and then increases, which implies too many edge types may lead to overfitting issues, since some edge types may capture subtle patterns from data which reduces generalization ability. The cross-validation is needed to determine the number of edge types. In Figure 5.7(b), it is illustrated that the prediction error increases consistently as the re-encoding gap raises, which implies more frequent re-identification of underlying interaction pattern indeed helps when it evolves along time. However, we need to trade off between the prediction error and testing running time if online prediction is required. The variance of minADE_{20} / minFDE_{20} in both figures are small, which implies the model performance is stable with random initialization and various settings in multiple experiments.

5.6 Chapter Summary

In this work, we present a generic trajectory forecasting framework with explicit relational reasoning among multiple heterogeneous, interactive agents with a graph representation. Multiple types of context information (e.g. static / dynamic, scene images / point cloud density maps) can be incorporated in the framework together with the trajectory information. In order to capture the underlying dynamics of the evolution of relational structures, we propose a dynamic mechanism to evolve the interaction graph, which is trained in two consecutive stages. The double-stage training mechanism can both speed up convergence and enhance prediction performance. The method is able to capture the multi-modality of future behaviors. The framework is validated by synthetic physics simulations and multiple trajectory forecasting benchmarks for different applications, which achieves state-of-the-art performance in terms of prediction accuracy. For the future work, we will handle the prediction task involving a time-varying number of agents with an extended adaptive framework. EvolveGraph can also be applied to find the underlying patterns of large-scale interacting systems which involve a large number of entities, such as very complex physics systems.

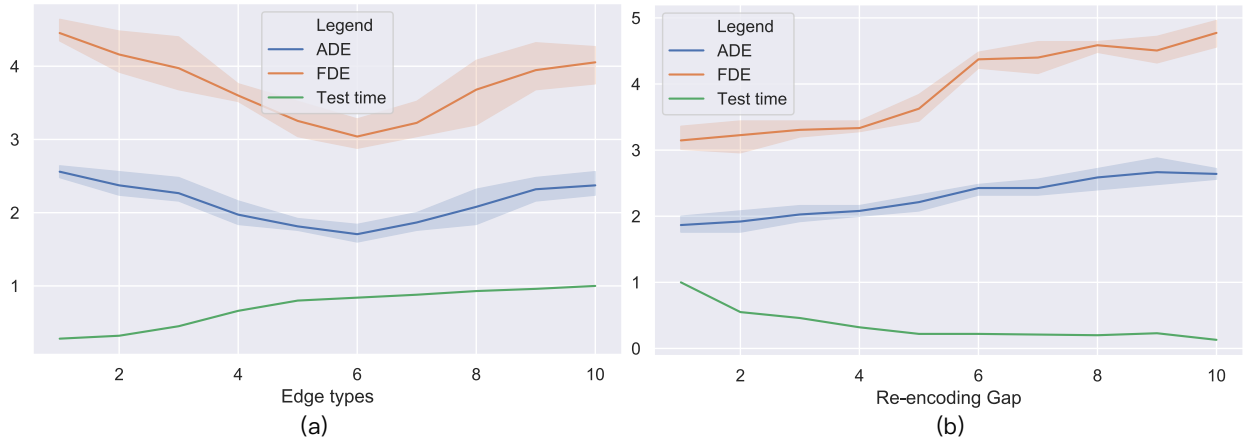


Figure 5.7: The comparison of minADE_{20} / minFDE_{20} (in meters) and testing running time of different model settings on the NBA dataset. We trained three models for each setting to illustrate the robustness of the method. (a) Different numbers of edge types; (b) Different re-encoding gaps. The testing running time is re-scaled to [0,1] for better illustration.

Chapter 6

CMSMC: Generic Mixture Tracking and Prediction Framework

6.1 Introduction

Effective tracking of surrounding objects and inference of their future motions are critical for intelligent systems (e.g. autonomous vehicles and industrial robotics) to achieve safe and high-quality decision making, motion planning and control. Although single-target tracking problems have been well studied in literature, it still remains a challenge for multi-target tracking due to multi-modality and selection of data association methods. The tracking performance is also dependent on the prediction quality of state evolution models. In simple scenarios where each entity behaves independently or with few interactions, state transition models based on pure kinematics or dynamics can be accurate enough to make a short-term forecasting. In many real-world applications where highly-interactive agents exist, however, these models are not sufficient due to the inherent uncertainty of future behaviors and interdependency among multiple entities. Also these models may be even not available due to the essential complexity such as pedestrian behaviors. Above all, it is desired to have a unified framework for tracking multiple agents and predicting their future motions simultaneously, which takes the uncertainties and interactions into account.

Many studies on multi-target tracking have been conducted in recent decades, which can be classified into two main categories. One category employs deep learning and computer vision techniques to track objects by real-time detection on images and videos [165, 28], where the bounding boxes of tracking targets can be obtained. The other category estimates the state distribution of tracking targets through Bayesian inference methods. Kalman filter (KF) [106] is a widely used estimator for linear systems while Extended Kalman filter (EKF) [115, 70] and Unscented Kalman filter (UKF) [201] are utilized in nonlinear systems. However, in practice the state distribution cannot be well approximated by simple multivariate Gaussian distribution. The sequential Monte Carlo (SMC) method (a.k.a. particle filter) [7, 46, 155, 175] thus has superiority over KF variants since no assumption on system model and state

distribution is made. There are also research lying in the intersection of both categories [155], in which Bayesian filters are utilized to provide a heuristic for forecasting the positions of bounding boxes in future frames. In this paper, we only focus on tracking methods based on recursive Bayesian state estimation where observation sequences can be obtained by sensor fusion.

There are two popular solutions for multi-target tracking with SMC method. One is using multiple instances of single object tracking where each entity is assumed to be independent. However, independence is not an appropriate assumption for interactive targets. The other is using dynamic state space extension to model the joint distribution of all the objects' states. However, the dimension of state space will blow up as target number increases. To overcome the challenges and deficiencies of existing methods, we propose a uniform framework which makes a bridge for multi-target tracking and multi-agent prediction. Instead of assuming each tracking target behaves independently, we take the interactions into consideration to enhance tracking.

The main contributions of this work are summarized as follows:

- We propose a constrained mixture sequential Monte Carlo (CMSMC) method in which a mixture representation is incorporated in the estimated posterior distribution to maintain multi-modality. Multiple targets can be tracked simultaneously within a unified framework without explicit data association between observations and tracking targets. Any form of prediction models can be employed as the implicit proposal distribution of CMSMC method.

- We propose to employ the prediction model as an implicit proposal distribution of CMSMC method and formulate a unified tracking and prediction framework, which is able to handle occlusions and sensor failures.

6.2 Related Work

In real-world tracking problems, there are commonly linear or nonlinear constraints on the estimated states. For instance, there are upper limits for the absolute value of acceleration and steering angle raised by dynamics feasibility restriction for vehicle tracking; the positions and velocities are also limited in proper ranges due to traffic rules. To enable incorporation of constraints, a constrained version of KF and its variants was proposed to handle linear or linearized systems [169]. The constrained particle filter was also investigated to cope with nonlinear systems with complicated constraints through an acceptance/rejection procedure [84]. In [158], an additional optimization technique is employed to further improve the efficiency and robustness. The concept of mixture tracking was first proposed in [180] and applied to football player tracking in a sequence of video frames.

In this work, we modify the original formulation of mixture tracking and generalize it to be adapted to general nonlinear discrete-time systems. Also, we add a constraint handling step in the mixture sequential Monte Carlo method to enhance tracking and prediction

performance as well as to suggest potential constraint incorporation strategies. This step does not affect the theoretical convergence properties.

6.3 Constrained Mixture Sequential Monte Carlo

In this section, we first present the theory of recursive constrained Bayesian state estimation with a mixture model representation. Since it is intractable and hard to obtain a closed-form estimated distribution, the constrained mixture sequential Monte Carlo approach is proposed to approximate the prior and posterior state distributions. The convergence analysis and practical implementation guides are provided.

6.3.1 Recursive Constrained Mixture Bayesian State Estimation

Consider a general nonlinear discrete-time state space system with equality and (or) inequality constraints on the state which can be formulated as

$$\begin{aligned} \mathbf{x}_k &= q_{k-1}(\mathbf{x}_{k-1}, \mathbf{e}_{k-1}, \mathbf{v}_{k-1}), \\ \mathbf{z}_k &= h_k(\mathbf{x}_k, \mathbf{w}_k), \\ \mathbf{x}_k &\in \mathbb{S}_{\mathbf{x}_k}, \end{aligned} \tag{6.1}$$

where the subscript k denotes the time step, $\mathbf{x}, \mathbf{e}, \mathbf{z}, \mathbf{v}, \mathbf{w}$ denote the state vector, the exterior information, the measurement vector, the process noise and the measurement noise, respectively. Note that the random variable $\mathbf{e}$ is involved in our work since the state evolution can be affected by exterior factors, which rarely emerges in the canonical formulation. $\mathbb{S}_{\mathbf{x}_k}$ denotes the feasible state set satisfying all the constraints. $q(\cdot)$ represents the process model (a.k.a. system dynamics model) and $h(\cdot)$ represents the measurement model. The process model and measurement model can be time-invariant or time-variant and the noise value can be sampled from arbitrary distributions.

The recursive state-constrained Bayesian estimation consists of two steps:

Step 1: Prior (Prediction) Update:

$$f(\mathbf{x}_k|\mathbf{z}^{k-1}) = \int_{\mathbf{e}_{k-1}} \int_{\mathbf{x}_{k-1}\in\mathbb{S}_{\mathbf{x}_{k-1}}} [f(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{e}_{k-1}) f(\mathbf{x}_{k-1}, \mathbf{e}_{k-1}|\mathbf{z}^{k-1})] \mathrm{d}\mathbf{e}_{k-1} \mathrm{d}\mathbf{x}_{k-1}, \tag{6.2}$$

Step 2: Measurement Update:

$$f(\mathbf{x}_k|\mathbf{z}^k) = \frac{f(\mathbf{z}_k|\mathbf{x}_k) f(\mathbf{x}_k|\mathbf{z}^{k-1})}{\int_{\mathbf{x}_k\in\mathbb{S}_{\mathbf{x}_k}} f(\mathbf{z}_k|\mathbf{x}_k) f(\mathbf{x}_k|\mathbf{z}^{k-1}) \mathrm{d}\mathbf{x}_k}, \tag{6.3}$$

where $f(\cdot)$ represents the probability density function and $\mathbf{z}^k = (\mathbf{z}_1, \cdots, \mathbf{z}_k)$ represents the measurement up to time step k. The initial state distribution is set to be $f(\mathbf{x}_0|\mathbf{z}_0)$ which is adaptive to the initial measurement.

The above formulation works well for estimating unimodal distributions which is widely used in single object tracking. However, it does not perform well in multi-modal distribution estimation as well as in multi-object tracking problems since the estimated distribution tends to degenerate to be unimodal along time (e.g. due to particle weight degeneracy in SMC). Therefore, the mixture model formulation is utilized to maintain multiple modalities, which requires no distribution parameterization assumptions. The posterior state distribution can be written as

$$f(\mathbf{x}_k|\mathbf{z}^k) = \sum_{m=1}^{M} \pi_{m,k} f_m(\mathbf{x}_k|\mathbf{z}^k), \tag{6.4}$$

where M denotes the component number of the mixture model, $\pi_{m,k}$ denotes the m-th component weight at time step k and $\sum_{m=1}^{M} \pi_{m,k} = 1$. Assuming that the posterior state distributions for each mixture component at time step $k-1$, i.e. $f_m(\mathbf{x}_{k-1}|\mathbf{z}^{k-1})$ has been obtained from the last measurement update and the exterior information $\mathbf{e}_{k-1}$ is independent from $\mathbf{x}_{k-1}$, we can calculate the new prior state distribution by

$$\begin{aligned} f(\mathbf{x}_k|\mathbf{z}^{k-1}) = \sum_{m=1}^{M} \pi_{m,k-1} \int_{\mathbf{e}_{k-1}} \int_{\mathbf{x}_{k-1}} [f_m(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{e}_{k-1}) \\ \times f_m(\mathbf{x}_{k-1}|\mathbf{z}^{k-1}) f_m(\mathbf{e}_{k-1}|\mathbf{z}^{k-1})] \mathrm{d}\mathbf{x}_{k-1} \mathrm{d}\mathbf{e}_{k-1}. \end{aligned} \tag{6.5}$$

When a new measurement is taken in, the prior state distribution is substituted into (6.3), which leads to

$$f(\mathbf{x}_k|\mathbf{z}^k) = \frac{\sum_{m=1}^{M} \pi_{m,k-1} f_m(\mathbf{z}_k|\mathbf{x}_k) f_m(\mathbf{x}_k|\mathbf{z}^{k-1})}{\sum_{n=1}^{M} \pi_{n,k-1} \int_{\mathbf{x}_k \in \mathbb{S}_{\mathbf{x}_k}} f_n(\mathbf{z}_k|\mathbf{x}_k) f_n(\mathbf{x}_k|\mathbf{z}^{k-1}) \mathrm{d}\mathbf{x}_k}. \tag{6.6}$$

The new posterior distribution and mixture weight for the m-th component can be obtained through following equations:

$$f_m(\mathbf{x}_k|\mathbf{z}^k) = \frac{f_m(\mathbf{z}_k|\mathbf{x}_k) f_m(\mathbf{x}_k|\mathbf{z}^{k-1})}{\int_{\mathbf{x}_k \in \mathbb{S}_{\mathbf{x}_k}} f_m(\mathbf{z}_k|\mathbf{x}_k) f_m(\mathbf{x}_k|\mathbf{z}^{k-1}) \mathrm{d}\mathbf{x}_k}, \tag{6.7}$$

$$\begin{aligned} \pi_{m,k} &= \frac{\pi_{m,k-1} \int_{\mathbf{x}_k \in \mathbb{S}_{\mathbf{x}_k}} f_m(\mathbf{z}_k|\mathbf{x}_k) f_m(\mathbf{x}_k|\mathbf{z}^{k-1}) \mathrm{d}\mathbf{x}_k}{\sum_{n=1}^{M} \pi_{n,k-1} \int_{\mathbf{x}_k \in \mathbb{S}_{\mathbf{x}_k}} f_m(\mathbf{z}_k|\mathbf{x}_k) f_m(\mathbf{x}_k|\mathbf{z}^{k-1}) \mathrm{d}\mathbf{x}_k} \\ &= \pi_{m,k-1} f_m(\mathbf{z}_k|\mathbf{z}^{k-1}) / \sum_{n=1}^{M} \pi_{n,k-1} f_n(\mathbf{z}_k|\mathbf{z}^{k-1}). \end{aligned} \tag{6.8}$$

The above recursive process can be applied to each individual component which only interacts with others by the adaptive adjustment of component weights in each iteration.

6.3.2 CMSMC Approximation

In order to approximate the constrained mixture state estimation recursion, we propose a constrained mixture sequential Monte Carlo approach which can adopt arbitrary state evolution models as implicit proposal distribution. The CMSMC approximation is represented by six sets of variables: state vector $\mathbf{x}$, particle normalized weight w, particle unnormalized weight $\bar{w}$, the component identity c that indicates which component the particle pertains to, and the feasibility indicator $\mathbb{I}$ which reveals whether the state vector of this particle is inside the feasible region. We set $\mathbb{I} = 1$ for feasible particles while $\mathbb{I} = 0$ for unfeasible ones. A self-contained particle state formulation is defined as

$$\mathbf{p}_k^{(i)} = [\mathbf{x}_k^{(i)} \ w_k^{(i)} \ \bar{w}_k^{(i)} \ c_k^{(i)} \ \pi_k^{(i)} \ \mathbb{I}_k^{(i)}], \tag{6.9}$$

where the subscript k denotes the time step and the superscript (i) denotes the particle identity.

Rather than only estimate the state vector at the current time step, the proposed CMSMC method is able to estimate the whole state profile, which gives

$$\hat{f}(\mathbf{x}^{k(i)}|\mathbf{z}^{k-1}) = \sum_{m=1}^{M} f_m(\mathbf{x}^{k(i)}|\mathbf{x}^{k-1(i)}, \mathbf{e}^{k-1}, \mathbf{z}^{k-1}) f_m(\mathbf{x}^{k-1(i)}, \mathbf{e}^{k-1}|\mathbf{z}^{k-1}), \tag{6.10}$$

where $\mathbf{x}^{k(i)} = (\mathbf{x}_1^{(i)}, \cdots, \mathbf{x}_k^{(i)})$ represents the whole trajectory of the i-th particle. Since the algorithm complexity will increase much if a marginalization process is implemented to obtain $\hat{f}(\mathbf{x}_k|\mathbf{z}^k)$ from $\hat{f}(\mathbf{x}^k|\mathbf{z}^k)$, a widely used simplification [155] is employed as

$$\begin{aligned} f(\mathbf{x}_k|\mathbf{z}^k) \approx \hat{f}(\mathbf{x}_k|\mathbf{z}^k) &= \sum_{m=1}^{M} \pi_{m,k} \sum_{i \in \mathcal{C}_m} w_k^{(i)} \delta(\mathbf{x}_k^{(i)} - \mathbf{x}_k), \\ \sum_{m=1}^{M} \pi_{m,k} = 1, \quad \sum_{i \in \mathcal{C}_m} w_k^{(i)} &= 1, \ m = 1, ..., M, \end{aligned} \tag{6.11}$$

where $\delta(\cdot)$ denotes the Dirac delta function and $\mathcal{C}_m$ denotes the particle identity set corresponding to the m-th component. The details of the CMSMC procedure are introduced below, where we denote N_p as the total amount of particles.

Initialization

The initial particles are randomly sampled from the feasible regions of initial state distribution $f(\mathbf{x}_0|\mathbf{z}_0)$ with equal weight $1/N_p$, which automatically guarantees that the state constraints are satisfied.

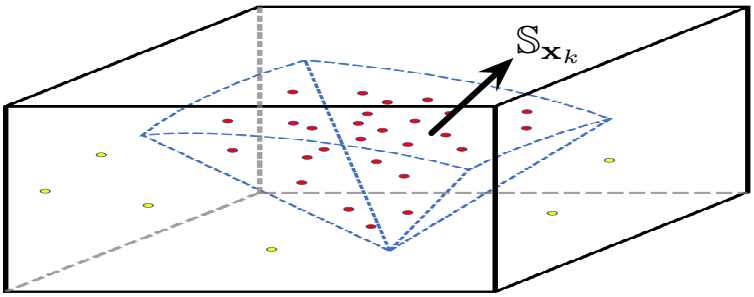


Figure 6.1: An illustrative diagram of incorporating state constraints. (The whole state space is represented by the cube and the feasible region is represented by a polytope which is restricted by four linear inequality constraints and one nonlinear inequality constraint. The red points and yellow points signify the feasible particles and unfeasible ones which may be rejected or assigned a zero weight, respectively.)

Prior (Prediction) Update & Measurement Update

The sequential importance sampling (SIS) technique [155] is employed in this step. We incorporate a proposal distribution $f_m^p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{e}_{k-1}, \mathbf{z}_k)$ for each component from which the new particles are sampled and obtain that

$$f(\mathbf{x}_k|\mathbf{z}^{k-1}) = \sum_{m=1}^{M} \pi_{m,k-1} \int_{\mathbf{e}_{k-1}} \int_{\mathbf{x}_{k-1}\in\mathbb{S}_{\mathbf{x}_{k-1}}} [f_m^p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{e}_{k-1}, \mathbf{z}_k) \frac{f_m(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{e}_{k-1})}{f_m^p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{e}_{k-1}, \mathbf{z}_k)} f_m(\mathbf{x}_{k-1}, \mathbf{e}_{k-1}|\mathbf{z}^{k-1})] \mathrm{d}\mathbf{e}_{k-1}\mathrm{d}\mathbf{x}_{k-1}. \tag{6.12}$$

Here we provide two strategies of incorporating constraints:

(i) For the m-th mixture component, sample a new state vector $\mathbf{x}_k$ for each existing particle from the proposal distribution $f_m^p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{e}_{k-1}, \mathbf{z}_k)$ only once and check whether $\mathbf{x}_k \in \mathbb{S}_{\mathbf{x}_k}$ is satisfied. Set $\mathbb{I}_k^{(i)}$ to be unity if $\mathbf{x}_k \in \mathbb{S}_{\mathbf{x}_k}$; otherwise, set $\mathbb{I}_k^{(i)}$ to be zero. Then calculate the new weights for the particles by

$$\bar{w}_k^{(i)} = \mathbb{I}_k^{(i)} \cdot w_{k-1}^{(i)} \frac{f_m(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, \mathbf{e}_{k-1}) f_m(\mathbf{z}_k|\mathbf{x}_k^{(i)})}{f_m^p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, \mathbf{e}_{k-1}, \mathbf{z}_k)}, \tag{6.13}$$

$$w_k^{(i)} = \bar{w}_k^{(i)} / \sum_{j\in\mathcal{C}_m} \bar{w}_k^{(j)}. \tag{6.14}$$

The posterior distribution of the m-th component can be properly approximated by the new particle set.

(ii) An alternative strategy is similar to (i) except that the new state vector $\mathbf{x}_k$ needs re-sampling until $\mathbf{x}_k \in \mathbb{S}_{\mathbf{x}_k}$ is satisfied. The corresponding particle weight is updated by (6.15)

instead of (6.13)

$$\bar{w}_k^{(i)} = w_{k-1}^{(i)} \frac{f_m(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, \mathbf{e}_{k-1}) f_m(\mathbf{z}_k|\mathbf{x}_k^{(i)})}{f_m^p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, \mathbf{e}_{k-1}, \mathbf{z}_k)}. \tag{6.15}$$

After obtaining the new particle weights, the component weights can be maintained accordingly by

$$\begin{aligned} \pi_{m,k} &= \frac{\pi_{m,k-1} \int_{\mathbf{x}_k \in \mathbb{S}_{\mathbf{x}_k}} f_m(\mathbf{z}_k|\mathbf{x}_k) f_m(\mathbf{x}_k|\mathbf{z}^{k-1}) \mathrm{d}\mathbf{x}_k}{\sum_{n=1}^{M} \pi_{n,k-1} \int_{\mathbf{x}_k \in \mathbb{S}_{\mathbf{x}_k}} f_m(\mathbf{z}_k|\mathbf{x}_k) f_m(\mathbf{x}_k|\mathbf{z}^{k-1}) \mathrm{d}\mathbf{x}_k} \\ &\approx \quad \pi_{m,k-1} \sum_{i \in \mathcal{C}_m} \bar{w}_k^{(i)} / \sum_{n=1}^{M} \pi_{n,k-1} \sum_{j \in \mathcal{C}_n} \bar{w}_k^{(j)}. \end{aligned} \tag{6.16}$$

There can be a particle re-sampling step for each component like canonical sequential Monte Carlo methods to avoid weight degeneracy if necessary.

Reclustering

Due to the long-term evolution of particles, the particle belonging to one component may become spatially close to another component, which requires a re-clustering step. Without loss of generality, in this work we employ the k-medoids approach where k equals the number of components. This does not change the estimated posterior distribution thus not affect the convergence analysis.

Mixture Component Weight Update

Since particles may transfer among different components, the component weights need to be updated to maintain the same distribution by

$$\begin{aligned} \hat{f}(\mathbf{x}_k|\mathbf{z}^k) &= \sum_{m=1}^{M} \pi'_{m,k} \sum_{i \in \mathcal{C}'_m} w'^{(i)}_k \delta(\mathbf{x}_k^{(i)} - \mathbf{x}_k), \\ \pi'_{m,k} = \sum_{i \in \mathcal{C}'_m} \pi_{c_k^{(i)},k} w_k^{(i)}, &\quad w'^{(i)}_k = (\pi_{c_k^{(i)},k} / \pi'_{c_k^{(i)},k}) w_k^{(i)}. \end{aligned} \tag{6.17}$$

6.3.3 Convergence Analysis

Many research efforts have been devoted to investigate the theoretical convergence properties of canonical sequential Monte Carlo methods such as [24, 83]. This subsection provides a concise convergence analysis of the proposed CMSMC method based on the following propositions for canonical SMC adapted from [83] where the exhaustive proofs can be found.

Proposition 1. If the state transition distribution $f(\mathbf{x}_k|\mathbf{x}_{k-1})$ is continuous and for $\forall k$, $0 < f(\mathbf{z}_k|\mathbf{x}_k) \leq C(k, \mathbf{z}_k) < \infty$ is satisfied, then for $\forall k$ and $\forall \mathbf{z}^k$, $||\hat{f}(\mathbf{x}_k|\mathbf{z}^k) - f(\mathbf{x}_k|\mathbf{z}^k)||_1 \to 0$ as the particle number $N_p \to \infty$.

Proposition 2. Under the conditions in Proposition 1, the approximated distribution by particles $\hat{f}(\mathbf{x}_k|\mathbf{z}^k)$ converges to the true posterior distribution at the rate of $1/\sqrt{N_p}$.

Due to the decomposability of mixture components in the CMSMC formulation, the convergence property can be evaluated at component level. The state transition distribution can be calculated as

$$f(\mathbf{x}_k|\mathbf{x}_{k-1}) = \sum_{m=1}^{M} \pi_{m,k-1} \int_{\mathbf{e}_{k-1}} f_m(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{e}_{k-1}), \tag{6.18}$$

which is continuous and the likelihood value $f_m(\mathbf{z}_k|\mathbf{x}_k)$ is bounded. Moreover, the reclustering process does not essentially modify the particle representation. Therefore, the convergence property in Proposition 1 also applies.

6.3.4 Practical Implementation Guides

Constraint Incorporation

The two constraint incorporation strategies have advantages on different aspects. Therefore, the choice should be made according to emphasis on the performance in a particular problem. The first strategy can guarantee the feasibility of the nonzero-weighted particles as well as keep the least computational cost since there is no multiple sampling process for a particular particle. This works well if the proposal distribution is properly chosen and the constraints are not hard to satisfy. Otherwise, there will be more and more rejected particles along time which results in a significant reduction of particle amount and low estimation quality even divergence. Under such situations, the second strategy which maintains a constant size of the particle set is the better choice despite larger computational cost.

There is another intuitive strategy which only samples the new particles once and pushes the unfeasible ones to the boundary of the feasible region. However, this is very hard to implement, especially when the boundary is highly nonlinear. Moreover, it may result in a high density around the boundary which leads to a large deviation to the true distribution.

Proposal Distribution

In order to obtain a approximated posterior distribution, the optimal proposal distribution is

$$f_m^p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)}, \mathbf{e}_{k-1}, \mathbf{z}_k) = f_m(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)}, \mathbf{e}_{k-1}, \mathbf{z}_k), \tag{6.19}$$

which has no influence on the particle weight variance. However, it is usually difficult to properly sample from this distribution and the weight update process brings much computational cost. A widely used alternative is

$$f_m^p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)}, \mathbf{e}_{k-1}, \mathbf{z}_k) = f_m(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)}, \mathbf{e}_{k-1}), \tag{6.20}$$

which is easy to sample from and can significantly simplify the weight update equation (6.13) and (6.15) into

$$\bar{w}_k^{(i)} = \mathbb{I}_k^{(i)} \cdot w_{k-1}^{(i)} f_m(\mathbf{z}_k|\mathbf{x}_k^{(i)}), \tag{6.21}$$

$$\bar{w}_k^{(i)} = w_{k-1}^{(i)} f_m(\mathbf{z}_k|\mathbf{x}_k^{(i)}). \tag{6.22}$$

In this work, we employed this proposal distribution in all the experiments.

Resampling Strategy

In order to avoid weight degeneracy, the particles need to be resampled when necessary. A straightforward way is to resample at each iteration. However, this is only suitable for offline implementation since it may ruin the real-time capability of the algorithm due to the large computational cost. A better choice is to use the effective number criterion [155] where

$$N_{\text{eff},k} = \frac{N_p}{1 + \frac{\Sigma(w_k^{(i)})}{[\mu(w_k^{(i)})]^2}} \approx \frac{1}{\sum_{i=1}^{N_p}(w_k^{(i)})^2}. \tag{6.23}$$

If $N_{\text{eff},k}$ is less than a proper threshold N_{th}, then the resampling process applies. A comparison study of the resampling algorithms is provided in [31]. In this work, we utilized systematic resampling in all the experiments.

Real-time Capability

The real-time performance of proposed CMSMC approach is mainly dependent on the following factors: i) common: particle amount, resampling frequency and constraint incorporation strategy; ii) problem-specific: target number, model complexity and proposal distribution sampling efforts. It is natural that more particles lead to a better approximation of distributions. However, in many cases it is likely that when the particle amount is large enough, adding particles will increase computational cost but bring little performance improvement. Therefore, a tradeoff on particle amount should be made by choosing the best setup through multiple experiments. Another consideration is to dynamically adjust particle number such as reduce particles after convergence and add particles when they tend to diverge or the tracking accuracy tends to decrease.

Divergence Alert

In real-time applications, it is crucial to set up a divergence alert mechanism to decide whether to adjust particle amount or even reinitialize the algorithm. There are two efficient divergence indicators: the effective number N_{eff} and raw likelihood values of particles. If they are less than properly chosen thresholds, the divergence alert will be activated.

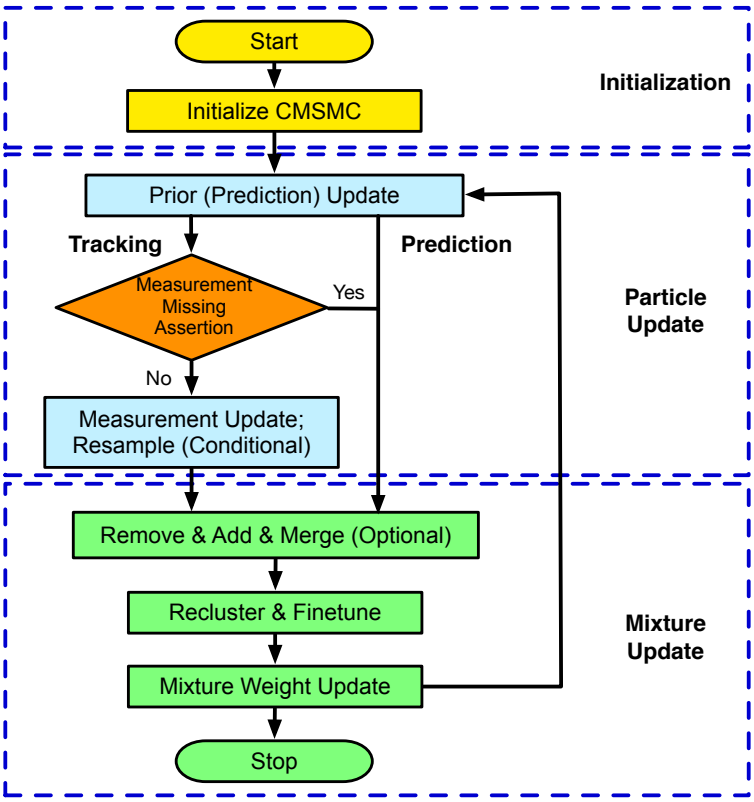


Figure 6.2: The diagram of CMSMC-based tracking and prediction framework.

6.4 Generic Tracking and Prediction Framework

In this section, we propose a generic tracking and prediction framework based on the constrained mixture sequential Monte Carlo approach, whose flow diagram is illustrated in Figure 6.2. The framework has a closed-loop structure which falls into three stages: initialization, particle update and mixture update. A summarized implementation procedure of the framework can be found in Algorithm 2.

The framework has two function modes: tracking mode and prediction mode. In each iteration of the tracking mode, there is a "measurement missing assertion" step through setting a proper distance threshold to check whether the new measurement of tracking targets are lost due to complete occlusion or sensing failure. If so, the current step is treated as a prediction problem thus without measurement update.

In the real-world applications, the number of tracking targets may fluctuate along time due to object emergence and disappearance as well as merging and splitting. Therefore, an adaptive adjustment mechanism is required so that the difference between component number and true target quantity is minimized. Therefore, a "Remove & Add & Merge" step is employed to adaptively adjust the component number, which is introduced in detail below.

Algorithm 2: CMSMC-based Tracking and Prediction

Require:

1. Function Mode (FM): tracking (0) or prediction (1);
2. Mixture Update Mode (MUM): fixed component number (0) or adaptive component number (1);
3. Initial component number M_0;
4. Initial particle amount for each component n_c;
   the total particle amount is $N_p = M_0 n_c$ accordingly;

1: $StopFlag \leftarrow 0$; $k \leftarrow 1$;
2: Initialization: draw initial particles $\{\mathbf{x}_0^{(i)} : i = 1, ..., N_p\}$ according to a known $f(\mathbf{x}_0|\mathbf{z}_0)$ with equal weights;
3: **while** $StopFlag = 0$ **do**
4: Prior Update: sample $\mathbf{x}_k^{(i)}$ from the proposal distribution $\mathbf{x}_k^{(i)} \sim f_m^p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i-1)}, \mathbf{e}_{k-1}, \mathbf{z}_k)$ using one of the constraint incorporation strategies;
5: **if** $FM = 0$ **then**
6: Measurement Update: calculate the unnormalized particle weights by (6.13) or (6.15) and normalized weights by (6.14); If $N_{\text{eff},k} < N_{th}$, resample by the systematic resampling algorithm or any other proper strategies;
7: **end if**
8: **if** $MUM = 1$ **then**
9: Adjust the component number adaptively as illustrated and obtain the new component number M'; $M \leftarrow M'$;
10: **end if**
11: Recluster the particles and calculate the new component weights by (6.17); $k \leftarrow k + 1$;
12: **end while**

(i) *Remove*: the components are removed if the corresponding weights are less than π_{th} or the mean point is outside the observation area;

(ii) *Add*: the component number will increase by one if the amount of particles assigned to a certain measurement in the last iteration is less than N_{mth} which we treat as a new target emergence. New particles are drawn around the new target;

(iii) *Merge*: the two components are merged when their distance is less than d_{th}. We employ the distance metric proposed in [48] for component m and n

$$\text{dist}(m, n) = \frac{\int [\hat{f}_m(\mathbf{x}) - \hat{f}_n(\mathbf{x})]^2 \mathrm{d}\mathbf{x}}{\int \hat{f}_m(\mathbf{x})^2 \mathrm{d}\mathbf{x} + \int \hat{f}_n(\mathbf{x})^2 \mathrm{d}\mathbf{x}}. \tag{6.24}$$

Since the mixture representation is non-parametric which makes the distance metric intractable to evaluate, we fit a Gaussian distribution to each component and obtain the approximated means $\hat{\mu}_m$ and variances $\hat{\Sigma}_m$ which are then substituted into (6.24). The distance metric is

approximated by

$$\text{dist}(m,n) \approx \frac{|4\pi\hat{\Sigma}_m|^{-\frac{1}{2}} + |4\pi\hat{\Sigma}_n|^{-\frac{1}{2}} - 2\mathcal{N}(\hat{\mu}_m|\hat{\mu}_n, \Sigma_m+\Sigma_n)}{|4\pi\hat{\Sigma}_m|^{-\frac{1}{2}} + |4\pi\hat{\Sigma}_n|^{-\frac{1}{2}}}. \tag{6.25}$$

6.5 Case Study I: Multivariate Dynamic System

In this section, we use the same numerical case as in Section 2.4 to demonstrate the effectiveness and accuracy of the CMSMC-based tracking and prediction framework. Please refer to the details of the multivariate dynamic system in Section 2.4. The superiority of CMSMC method is demonstrated by a comparison with EKF and UKF. We use the prediction model HTSPM introduced in Section 2.3.

6.5.1 CMSMC v.s. EKF/UKF

To illustrate the advantages of proposed CMSMC method, we compared its tracking performance with EKF and UKF. Since linearization process is necessary for EKF, we adopted a differentiable state equation

$$\begin{aligned} x_{1,k} &= x_{1,k-1} + 2x_{2,k-1}\Delta T + x_{3,k-1}\Delta T^2 + v_{1,k-1}, \\ x_{2,k} &= x_{2,k-1} + x_{3,k-1}^2\Delta T + v_{2,k-1}, \\ x_{3,k} &= x_{3,k-1} + v_{3,k-1}, \end{aligned} \tag{6.26}$$

which is an approximation of the original state space model. We used 100 particles for each mixture component in CMSMC and there were four tracking targets corresponding to four different high-level behaviors. The performance comparisons are provided in Figure 6.3. It is shown in Figure 6.3(a) that the UKF and CMSMC have comparable accuracy on state mean values while EKF has a larger error especially during highly nonlinear stages, which indicates that using a first-order approximation at the current state (which is only employed in EKF) is not sufficient for estimating a general highly nonlinear system. Apart from mean values, covariance is another critical indicator when evaluating an approximated distribution. To be comparable with CMSMC, we sampled the same amount of particles using the estimated means and covariance matrices of EKF and UKF and computed the Mean Absolute Error (MAE) which is shown in Figure 6.3(b). The results indicate that CMSMC can achieve the smallest MAE and the most stable tracking performance, which shows that CMSMC has greater advantages when handling nonlinear systems with non-Gaussian noise. The particles of CMSMC are visualized in Figure 6.3(c) which achieves a smooth and coherent tracking performance.

6.5.2 HTSPM v.s. Other State Evolution Models

The HTSPM consists of the aforementioned DHMM and four independent state evolution models corresponding to four high-level behaviors I, II, III and IV. We compared the tracking

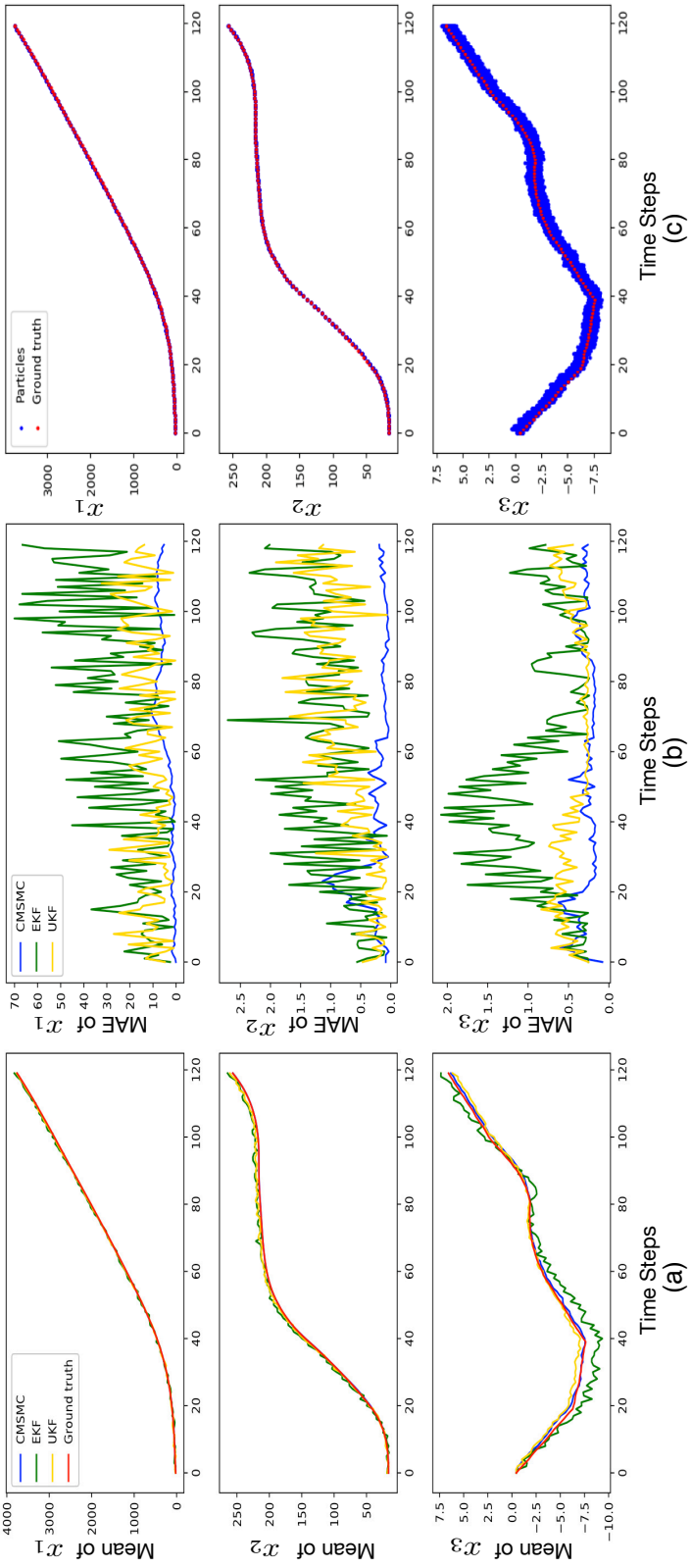


Figure 6.3: The performance comparisons of CMSMC, EKF and UKF. (a) The mean values of state tracking results; (b) The mean absolute error (MAE) of state tracking results; (c) The visualized particles of CMSMC.

Table 6.1: MAE Value Comparisons of Tracking Performance for Numerical Case Study.

Model	x_1	x_2	x_3
DHMM + CGMR	1.43	0.51	0.71
DHMM + CP-MLP	1.15	0.38	0.57
DHMM + CP-LSTM	**0.91**	**0.16**	**0.24**
GGMR	2.98	0.74	1.12
P-MLP	1.67	0.61	0.88
P-LSTM	1.33	0.35	0.62
SSM	1.89	0.81	0.96

performance of proposed HTSPM with other state evolution models based on the CMSMC framework in terms of the average of MAE values over the tracking horizon, which is presented in Table 6.1. The *Global Gaussian Mixture Regression (GGMR), P-MLP* and *P-LSTM* are unified models which are trained without separating different behaviors, which means a single model is able to make predictions of all the behavior classes. The *SSM* refers to the aforementioned approximated state space model (6.26). For both conditional and unconditional models, the GMM has 20 mixture components and the neural network has three hidden layers each with 64 units followed by a ReLU activation function. The input noise is sampled from a three-dimensional normal distribution.

It can be seen that in general HTSPM can achieve lower MAE than the corresponding behavior-unconditional versions, which illustrates the significance of recognition module. The reasons are two folds. On the one hand, the behavior-unconditional models need to learn a much more complex data distribution than behavior-conditional ones due to the variety of motion patterns, which demands a more sophisticated architecture with larger learning capacity. On the other hand, there tends to be mode collapse in behavior-unconditional models since the optimization algorithm usually gets stuck at local optimums and provide an averaged output of training cases which leads to undesired minimization of the loss functions. Among the compared models, using LSTM achieved the lowest tracking error which implies it is better at learning time dependencies of time-series. Moreover, note that overall learning-based models can achieve better tracking accuracy than the approximated state space model, which indicates high practicability and superiority of learning-based models in real-world applications where true system models are unavailable.

6.6 Case Study II: On-Road Vehicles

In this section, we apply the proposed tracking and prediction framework to solve real-time vehicle tracking problems in a highway scenario. We adopt the HTSPM presented in Section 2.3 as the prediction model in the tracking framework. We also use the same dataset and experimental settings as Section 2.6. The details are omitted to avoid repetition.

Table 6.2: ADE Value Comparisons of Tracking Performance of Vehicle Positions and Velocities

Model	Position (m)	Velocity (m/s)
DHMM + CGMR	0.046 / 0.028	0.636 / 0.541
DHMM + CP-MLP	0.041 / **0.021**	**0.547** / 0.488
DHMM + CP-LSTM	**0.038** / 0.024	0.582 / **0.473**
GGMR	0.051 / 0.033	0.941 / 0.557
GP-MLP	0.043 / 0.029	0.812 / 0.617
GP-LSTM	0.044 / 0.031	0.847 / 0.656
CAM	0.082 / 0.067	1.936 / 1.614

For multi-target tracking, we sampled 100 initial particles for each tracked vehicle from a Gaussian distribution with the mean at initial observations. The particle state contains vehicle positions x, y and velocities $\dot{x}, \dot{y}$ except that accelerations $\ddot{x}, \ddot{y}$ are additionally considered when using CAM. The comparisons of tracking performance in terms of Average Distance Error (ADE) are illustrated in Table 6.2 where the first column corresponds to middle car while the second one corresponds to the average of five surrounding cars. The bold numbers indicate best performance. It is shown that learning-based vehicle motion models can achieve much lower tracking errors than pure kinematic model. Moreover, with the behavior recognition module on top of evolution module, the HTSPM is more capable of capturing the true vehicle state evolution distribution.

6.7 Chapter Summary

In this chapter, a generic multi-target tracking and multi-agent probabilistic behavior prediction framework based on constrained mixture sequential Monte Carlo (CMSMC) method is proposed, which can track multiple entities simultaneously without explicit data association with a unified representation and predict the joint distribution of their future motions or states. A generic learning-based hierarchical time-series prediction model (HTSPM) was also used to serve as an implicit proposal distribution in the prior update of Bayesian state estimation. The proposed framework and models are validated by a numerical case study. The results show that the proposed CMSMC method can achieve better tracking accuracy than variants of KF in terms of both mean and variance of posterior distribution.

Part II

Importance Estimation

Chapter 7

RAIN: Reinforced Hybrid Attention Inference Network

7.1 Introduction

Motion forecasting has been widely studied in various domains, such as physical systems, human skeletons, and multi-agent interacting systems (e.g., traffic participants, sports players, etc). The problem is formulated as to predict future states or trajectories based on historical spatio-temporal observations. However, the observed information may be of different levels of significance and in some situations not all the information is relevant for the forecasting. Moreover, the key information may be varying as the situation evolves, which motivates the forecasting approach to dynamically adjust its attention to different subsets of observations. Here we provide two illustrative real-world examples where key information is naturally selected based on either spatial relations or temporal dependencies. An on-road vehicle usually only needs to pay attention to the traffic participants that are interacting or having a conflict with itself, so only a subset of observations are indeed relevant when predicting its future behavior. For human motion forecasting, it is observed that humans tend to repeat their motions, which motivates dynamic attention to different segments of previous motions given the current observation.

Attention mechanisms have been widely adopted to learn the relative importance of elements. There are two major types of attention mechanisms in literature: soft attention and hard attention [195]. The soft attention is usually performed by applying a score function to input features followed by a softmax function to obtain the attention weights in the range of $[0, 1]$. These operations are fully differentiable which can be trained by back-propagation with typical gradient-based optimizers. However, the softmax function tends to assign non-zero attention weights to irrelevant or unimportant elements, which dilutes the attention given to the truly significant information [179, 159]. In contrast, the hard attention mechanism can force the model to only pay attention to the relevant information while discard the others entirely to reduce information redundancy. The hard attention weights can be only

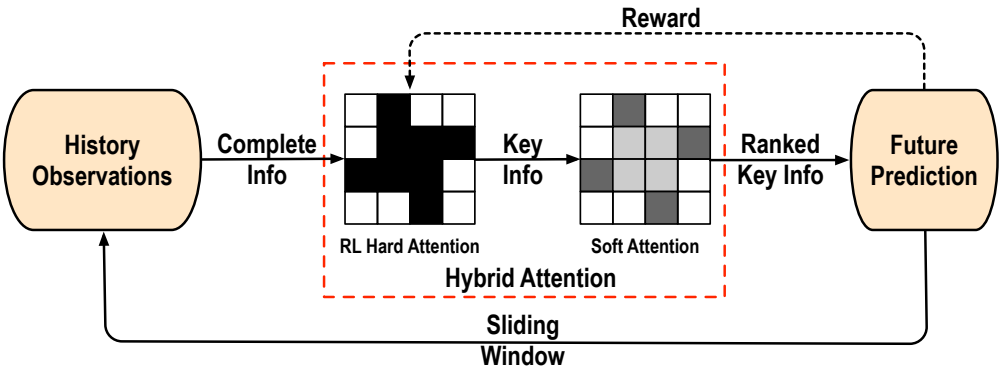


Figure 7.1: A high-level diagram of the proposed general prediction framework with key information/element selection and ranking, which consists of two major parts: an RL-based hard attention mechanism to discriminate key information from complete observations and a soft attention mechanism to further figure out relative significance of the key information. The whole procedure can be iteratively applied over time with a sliding window to enable dynamic selection of key information to adapt to evolving situations.

binary: 0 (discarded) or 1 (retained). However, the hard attention is not differentiable due to the argmax operation, which needs to be optimized by reinforcement learning (RL) algorithms (e.g., deep Q-learning [131], policy gradient [191]). Recently, some alternatives to the traditional hard attention have been proposed based on the approximation of evidence lower bound (ELBO) [124], which can be trained end-to-end.

The general idea of selecting the most important information/elements with hard attention has been applied to several different domains, such as computer vision and natural language processing. Wu et al [193] introduced a key frame selection framework based on multi-agent reinforcement learning for video based human activity recognition. Wang et al [181] presented a framework for informative view selection from multiple indoor cameras to recognize human actions. Gao et al [37] proposed a hard and channel-wise attention network for graph representation learning. Shen et al [159] illustrated a reinforced self-attention network to figure out sparse dependencies between tokens in a sentence.

However, the efficacy of hard attention in motion forecasting tasks, to the best of our knowledge, still remains largely unexplored so far. Besides, many existing works including the aforementioned ones, pre-define a fixed number of elements to pay attention to, which may be unsatisfactory in the scenarios where the amount of key information/elements is varying. For example, the motion of a certain entity in an interacting system may be affected by a varying number of entities at different time, thus a fixed number of selected elements may be redundant or insufficient in different situations. To address this issue, we propose a reinforcement learning based hard attention mechanism for motion forecasting, which does not enforce any constraints on the amount of key elements. It is even possible in some

situations that no element or all the elements are selected based on their significance. In the multi-agent setting with a graph representation, learning hard attention can also be treated and interpreted as graph structure/topology learning [35, 108]. Since the selected key information may be still at different levels of importance, we propose to employ soft attention as a ranking mechanism to further discriminate relative importance.

To the best of our knowledge, we are the first to propose a hybrid attention based framework for motion forecasting, which is illustrated in Figure 7.1. The main contributions of this work are summarized as follows:

- We propose a general motion forecasting framework (named RAIN) with dynamic key information/element selection and ranking via a hybrid attention mechanism.
- We propose an effective double-stage training pipeline with an alternating training strategy to improve different modules in the framework alternatively.
- We instantiate the general framework and propose a novel graph-based model for multi-agent trajectory forecasting. We also demonstrate the general idea on human skeleton motion forecasting, where a state-of-the-art model [189] is employed as a part of our framework. We validate the proposed framework on both domains and our method achieves the state-of-the-art performance consistently.

7.2 Related Work

7.2.1 Trajectory/Motion Forecasting

Many research efforts have been devoted to motion forecasting in various domains. Here we particularly provide a brief review of literature on physical systems, highly interactive traffic scenarios and skeleton based human motions, which are closely related to this work.

Physical systems

Learning the dynamics of physical systems involving multiple interacting elements have been studied in some recent works, either from simulated trajectories [8, 58, 77, 98] or from generated videos [187, 168, 82], where graph neural network is used to model interactions. Some of them assume a known graph topology based on prior knowledge, while the others infer the underlying structure explicitly or implicitly. Our approach falls into the second category, where we propose a novel fashion of relational inference based on reinforcement learning.

Traffic scenarios

A number of works attempt to forecast future trajectories or behaviors of heterogeneous traffic participants (e.g., pedestrians, vehicles, cyclists) [118, 62, 36, 80, 192, 93, 143, 121,

19, 14, 119]. In order to model the relations or interactions between different entities, many information aggregation techniques have been developed, including but not limited to, feature concatenation [93], attention mechanisms [179, 104, 62, 80], social pooling [3, 27], message passing across graphs [98, 62, 20], etc. Moreover, recent works are also putting more emphasis on environmental modeling to leverage physical and semantic constraints such as road layout and traffic rules in driving scenarios [38, 173, 143].

Human motions

Recurrent neural networks are widely utilized in human motion forecasting [66, 43, 34]. Recent works also proposed to adopt feed-forward methods for effective encoding of long-term motion history [126, 92]. Besides, attention mechanisms were also employed due to its flexibility and efficacy. In [174], Tang et al adopted a frame-wise attention mechanism to summarize the pose history. In [189], Mao et al proposed a motion attention based method, which achieved the previous state-of-the-art performance. In this paper, we apply our hard attention module on top of their soft attention model [189] to illustrate the efficacy of our proposed hybrid attention based framework.

7.2.2 Attention Mechanisms

Attention mechanisms have been widely used in deep neural networks due to their efficacy and efficiency for sequence modeling and information fusion. Existing works have demonstrated the superiority of attention in a broad range of domains, such as natural language processing [139, 159, 186], image captioning [195, 182, 5], saliency detection [114, 213], time-series modeling [32, 144], human activity recognition [161, 18, 167]. Most of these papers either solely apply soft attention to obtain contextual embeddings or solely apply hard attention to select important elements. The only exception is [159] in which a hybrid attention was presented to figure out sparse dependencies between sequential tokens in a sentence. However, their method can only be applied to sequence modeling, which is hard to generalize to the motion forecasting task of multi-agent interacting systems. For the motion forecasting task, some existing works only adopt soft attention to weight the complete observed information [179, 93, 62, 104, 80, 189], while our method can discriminate the most relevant elements via the hybrid attention.

7.3 Problem Overview

Since the proposed approach can be widely applied to various domains, we introduce the problem formulation in a general way. Define a multivariate dynamic system

$$\mathbf{X}_{t+1:t+T_f} = f(\mathbf{X}_{t-T_h+1:t}, \mathbf{C}), \tag{7.1}$$

where $\mathbf{X}_t = \{\mathbf{x}_t^i, i = 1, ..., N\}$ denotes the system state at time t and $\mathbf{C} = \{\mathbf{c}^i, i = 1, ..., N\}$ denotes optional context information or external factors. N is the total number of variables

which have a specific meaning in different domains. The goal of this work is to approximate the conditional distribution $p(\mathbf{X}_{t+1:t+T_f}|\mathbf{X}_{t-T_h+1:t}, \mathbf{C})$, where T_h and T_f denote the history and prediction horizon.

For a multi-agent interacting system, the variables refer to the involved homogeneous / heterogeneous interacting entities, where the state may include position, velocity, etc. For a multivariate time series such as human motions, the variables refer to a set of human skeletons, where the state may include joint coordinates or relative angles.

In this work, we instantiate and apply the proposed general prediction framework illustrated in Figure 7.1 to both multi-agent systems with spatially interacting agents (e.g., physical systems, traffic participants), and multivariate time series with temporal dependency (e.g., skeleton-based human motions). For long-term prediction of multi-agent interacting systems, we propose a novel, complete model architecture based on graph representation in Section 7.4. For multivariate time series (i.e., human motions), we present an effective way to build our proposed hard attention module on top of a state-of-the-art soft attention based model for human motion prediction [189] in Section 7.6.

7.4 Multi-Agent Interacting System: Model

7.4.1 Model Overview

An illustrative diagram is shown in Figure 7.2 to introduce the pipeline and three major components: graph message passing module (GMP), RL based hard attention module (RL-HA) and soft graph attention based motion generator (SGA-MG), which cooperate closely to improve the final prediction performance. More specifically, for the prediction of a certain target entity, GMP collects information from other entities across graph $\mathcal{G}$. RL-HA discriminates the key relevant elements from the complete observations and provides SGA-MG with an inferred relation graph $\mathcal{G}'$ with only selected edges, which is a natural generalization of the traditional hard attention to graph representation. SGA-MG uses soft attention weights to rank the relative importance of key information and generates future trajectories. The prediction together with the ground truth provides rewards to RL-HA during the training phase to guide the improvement of the RL edge selector. GMP is pre-trained to collect contextual information across the whole graph. SGA-MG is pre-trained with a fully connected topology in order to improve training efficiency and stability as well as to enable informative initial reward.

7.4.2 Graph Message Passing

It is natural to represent a multi-agent system with N entities as a fully connected (FC) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{\mathbf{v}_i, i = 1, ..., N\}$ and $\mathcal{E} = \{\mathbf{e}_{ij}, i, j = 1, ..., N\}$. $\mathbf{v}_i$ denotes node i's attribute and $\mathbf{e}_{ij}$ denotes the edge attribute from sender node j to receiver node i. The node attribute consists of a self-attribute to store the individual information, a social-

Figure 7.2: The diagram of the proposed hybrid attention based prediction framework for multi-agent interacting systems, which consists of three major components: a graph message passing module (GMP), a RL based hard attention module (RL-HA) and a soft graph attention based motion generator (SGA-MG). GMP aims to pass the information across the fully connected graph $\mathcal{G}$. RL-HA aims to figure out truly relevant information (edges) and output an inferred relation graph $\mathcal{G}'$, while SGA-MG aims to further rank the significance of the key information with soft attention and generate future motions with recurrent neural networks. The prediction process can be either one-shot (with static relation graph $\mathcal{G}'$) or progressive with a sliding window (with dynamic relation graph $\mathcal{G}'$).

attribute to store other entities' information, and a context-attribute to include the agent's context information. More formally, we have $\mathbf{v}_i^{\text{self}} = f_s^m(\mathbf{x}_{t-T_h+1:t}^i)$, $\mathbf{v}_i^{\text{neighbor}} = f_n^m(\mathbf{x}_{t-T_h+1:t}^i)$, $\mathbf{v}_i^{\text{context}} = f_c(\mathbf{c}^i)$, where $m \in \{1, ...M\}$, M is number of agent types. f_s^m and f_n^m are state embedding functions, and f_c is context embedding function. Different state embedding functions corresponding to certain agent types are applied to heterogeneous agents.

Since the relationship between a pair of agents is not only determined by their own behaviors but also affected by other agents in the scene, it is not sufficient to use only a pair of self node attributes to determine the existence of an edge. Therefore, we apply a round of message passing to collect the contextual information across the FC graph $\mathcal{G}$ by

$$\alpha_{ij} = \frac{\exp\left(\text{MLP}\left(\left[\mathbf{v}_i^{\text{self}} || \mathbf{v}_j^{\text{neighbor}}\right]\right)\right)}{\sum_{k\in\mathcal{N}_i} \exp\left(\text{MLP}\left(\left[\mathbf{v}_i^{\text{self}} || \mathbf{v}_k^{\text{neighbor}}\right]\right)\right)}, \tag{7.2}$$

$$\mathbf{v}_i^{\text{social}} = f_v\left(\sum\nolimits_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{v}_j^{\text{neighbor}}\right), \tag{7.3}$$

where f_v is the social attribute update function and $||$ denotes the concatenation operation. $\mathcal{N}_i$ denotes the set of one-hop neighbors of node i. MLP refers to multi-layer perceptron. The complete node attribute is

$$\mathbf{v}_i = f_{\text{enc}}\left(\left[\mathbf{v}_i^{\text{self}},\ \mathbf{v}_i^{\text{social}},\ \mathbf{v}_i^{\text{context}}\right]\right). \tag{7.4}$$

7.4.3 Hard Attention: Key Information Selection

As shown in Figure 7.1, the RL-based hard attention module serves as a key information selector, which takes the complete history observations as input and discriminates the truly relevant information while totally discard the rest. More specifically, in the context of multi-agent system, the RL-HA module is expected to figure out truly influencing factors when predicting the motions of a certain agent. In other words, the goal is to assert the existence of each edge in the FC graph based on the observations so that the redundant information is discarded in the prediction.

The selection of key edges naturally fits into a reinforcement learning framework. The definition of observations, actions and reward functions are elaborated in the following, while the training procedures are left to Section 7.4.6.

Observations

The observation O of RL-agent at RL-step η ($\leq T_{\text{RL}}$) includes a pair of node attributes $\mathbf{v}_i$ and $\mathbf{v}_j$ as well as the current edge selection status s_{ij} (0: "retained" or 1: "discarded"). T_{RL} is the upper bound of RL-steps. The observation O_η is obtained by $O_\eta = [\mathbf{v}_i,\ \mathbf{v}_j,\ s_{ij,\eta}]$. Note that the dimension of O_η only depends on the dimension of node attributes, which enables the applicability to the systems with varying numbers of entities. The policy network of RL-agent takes the observation O_η as input and decides the action at the current RL-step.

Actions

There are two possible actions for the RL-agent: "staying the same" (action 0) and "changing to the opposite" (action 1). At each RL-step, the RL-agent makes decision for each edge in the FC graph. The policy can be written as $a = \pi(O)$. We do not enforce any constraints on the selection of edges, i.e., there is no lower/upper bound on the number of selected edges. The actions of RL-agent may change the topology of the inferred graph $\mathcal{G}'$ after each RL-step, which further influence the SGA-MG module.

Rewards

In general, the reward indicates how good the action taken by the RL-agent is with respect to the current situation. In our method, the reward is designed to indicate the performance of motion forecasting in various aspects. The acquisition of high rewards depends on the collaboration of all the modules in the framework.

The reward consists of three parts: regular reward R_{reg}, improvement reward R_{imp} and stimulation/punishment $R_{\text{sti}}/R_{\text{pun}}$. More specifically, the *regular reward* is the negative mean squared error of future predictions calculated by

$$R_{\text{reg},\eta} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t'=t+1}^{t+T_f} ||\mathbf{x}_{t'}^{i} - \hat{\mathbf{x}}_{t',\eta}^{i}||^2. \tag{7.5}$$

The *improvement reward* encourages the decrease of prediction error via applying a sign function to the error change between consecutive RL-steps, which is obtained by

$$R_{\text{imp},\eta} = \text{sign}(R_{\text{reg},\eta} - R_{\text{reg},\eta-1}). \tag{7.6}$$

The reason of applying a sign function instead of directly using the raw improvement is to avoid reward vanishing when improvement becomes smaller towards convergence. The *stimulation/punishment* is applied when there is a large improvement or deterioration in terms of a certain metric, which is given by

$$R_{\text{sti},\eta} = \Omega_s, \quad R_{\text{pun},\eta} = -\Omega_p, \tag{7.7}$$

where Ω_s and Ω_p are manually defined positive constants. These rewards depend on the metrics in specific domains.

Then the whole reward is calculated by

$$R_\eta = R_{\text{reg},\eta} + \beta_{\text{imp}} R_{\text{imp},\eta} + \beta_{\text{sti}} R_{\text{sti},\eta}\mathbb{I}(\text{sti}) + \beta_{\text{pun}} R_{\text{pun},\eta}\mathbb{I}(\text{pun}), \tag{7.8}$$

where β_{imp}, β_{sti} and β_{pun} are hyperparameters and $\mathbb{I}(\cdot)$ is an indicator function to indicate the occurrence of large improvement or deterioration.

7.4.4 Soft Attention: Key Information Ranking

After finalizing the key information (edges) by the RL-HA module, the soft graph attention mechanism [178] is applied over the inferred graph $\mathcal{G}'$ to further determine the relative significance of the selected key information at each time step. Here we take time step t as an example to illustrate the soft graph attention mechanism. Note that unlike the one-shot state embedding in GMP, here we construct a spatio-temporal graph to incorporate the state information with a stepwise embedding strategy.

In order to avoid confusion on notation with the RL-HA section, here we denote i-th node attribute at time t as

$$\bar{\mathbf{v}}_{i,t} = [\bar{\mathbf{v}}_{i,t}^{\text{self}},\ \bar{\mathbf{v}}_{i,t}^{\text{social}},\ \mathbf{v}_i^{\text{context}}], \tag{7.9}$$

where the context attribute $\mathbf{v}_i^{\text{context}}$ is obtained in Section 7.4.2, the self-attribute $\bar{\mathbf{v}}_{i,t}^{\text{self}}$ and the intermediate attribute $\bar{\mathbf{v}}_{i,t}^{\text{neighbor}}$ are introduced in Section 7.4.5, and the social attribute $\bar{\mathbf{v}}_{i,t}^{\text{social}}$ is calculated by the graph soft attention mechanism as follows

$$\begin{aligned} \bar{\alpha}_{ij}^t =& \frac{\exp\left(\text{MLP}\left(\left[\bar{\mathbf{v}}_{i,t}^{\text{self}} || \bar{\mathbf{v}}_{j,t}^{\text{neighbor}}\right]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{MLP}\left(\left[\bar{\mathbf{v}}_{i,t}^{\text{self}} || \bar{\mathbf{v}}_{k,t}^{\text{neighbor}}\right]\right)\right)}, \\ \bar{\mathbf{v}}_{i,t}^{\text{social}} =& \bar{f}_v\left(\sum\nolimits_{j\in\mathcal{N}_i} \bar{\alpha}_{ij}^t \bar{\mathbf{v}}_{j,t}^{\text{neighbor}}\right), \end{aligned} \tag{7.10}$$

where $\bar{\alpha}_{ij}^t$ are learnable attention weights. We also use multi-head attention to stabilize training [178].

7.4.5 Spatio-Temporal Motion Generator

The motion generator consists of two LSTM networks (E-LSTM/G-LSTM) with the soft graph attention in between. The E-LSTM takes in agent state information and outputs $\bar{\mathbf{v}}_{i,t}^{\text{self}}$ at each time step, while the G-LSTM takes in the complete node attribute $\bar{\mathbf{v}}_{i,t}$ and outputs the predicted change in state $\Delta\hat{\mathbf{x}}_t^i$ at the current time t, which is used to calculate the state $\hat{\mathbf{x}}_{t+1}^i$ with the system model (e.g., discrete-time linear dynamics). More specifically at time t,

$$\text{Embedding: } \bar{\mathbf{v}}_{i,t}^{\text{self}} = \text{E-LSTM}^{\text{s}}(\mathbf{x}_t^i; \mathbf{h}_t^{s,i}), \tag{7.11}$$

$$\bar{\mathbf{v}}_{i,t}^{\text{neighbor}} = \text{E-LSTM}^{\text{n}}(\mathbf{x}_t^i; \mathbf{h}_t^{n,i}), \tag{7.12}$$

$$\text{Generation: } \Delta\hat{\mathbf{x}}_t^i = \text{G-LSTM}(\bar{\mathbf{v}}_{i,t}; \tilde{\mathbf{h}}_t^i), \tag{7.13}$$

$$\hat{\mathbf{x}}_{t+1}^i = f_{\text{system}}(\mathbf{x}_t^i, \Delta\hat{\mathbf{x}}_t^i) + \boldsymbol{\epsilon}, \tag{7.14}$$

where $\mathbf{h}_t^{s,i}$, $\mathbf{h}_t^{n,i}$, and $\tilde{\mathbf{h}}_t^i$ are the hidden states of E-LSTM and G-LSTM respectively, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ is a random noise to incorporate uncertainty, and f_{system} is the system model. Note that the whole generation process is divided into two stages: burn-in stage (from $t - T_h + 1$

to t) and prediction stage (from $t+1$ to $t+T_f$). At burn-in stage, the true state is fed into E-LSTM while at prediction stage, the last prediction is fed instead.

If the topology of the inferred graph $\mathcal{G}'$ is assumed to remain static over time, we can use one-shot generation to obtain the complete future trajectory. Otherwise, we can first generate the trajectory segment within a certain future horizon $\tau < T_f$, and push the predicted segment into the observations. This process can be iteratively propagated to generate the whole trajectory. We set $\tau = 2$ in Figure 7.2.

7.4.6 Framework Training Strategy

Since the RL-HA module is not differentiable, we cannot use an end-to-end fashion to train the whole model. Therefore, we propose an alternating training algorithm to improve RL-HA and the other modules separately. There are two training stages: pre-training stage and formal-training stage. In the *pre-training* stage, we pre-train the parameters of GMP module with an auto-encoder structure by unsupervised learning, where GMP serves as the encoder together with an auxiliary decoder. The decoder is discarded after the GMP is well trained. In addition, we pre-train the SGA-MG module with a fully connected topology to enable informative initial reward for the RL-HA module. The pre-training is necessary for the convergence of formal-training stage. In the *formal-training* stage, the pre-trained parameters of GMP are fixed to serve as a feature extractor, and we perform alternating optimization of RL-HA and SGA-MG modules: (a) train the RL-HA module with fixed parameters of GMP and SGA-MG with Double Deep Q-Learning (DDQN) method [50]; (b) finetune SGA-MG with fixed parameters of GMP and RL-HA using back-propagation methods.

Pre-Training Stage: GMP Module

We employ a standard encoder-decoder structure to pre-train the GMP by unsupervised learning, with the purpose of enabling informative and effective feature extraction in the GMP module. In the auto-encoder structure, the GMP module serves as the encoding process to generate latent embeddings for each node. And an auxiliary decoding function is trained to reconstruct the history information with the latent embeddings. After the model is well-trained, the GMP module is able to extract good representation of the observation information.

The loss function is the standard mean squared error reconstruction loss, which is calculated as

$$L_{\text{GMP}} = \frac{1}{NT_h} \sum_{i=1}^{N} \sum_{t'=t-T_h+1}^{t} ||\mathbf{x}_{t'}^i - \hat{\mathbf{x}}_{t'}^i||^2. \tag{7.15}$$

After convergence, we save the parameters of the GMP module and discard those of the decoder, since we only use GMP in the following formal-training stage.

Algorithm 3: Double-Stage Training Algorithm

Input: history $\mathbf{X}_{t-T_h+1:t}$, true future $\mathbf{X}_{t+1:t+T_f}$, context $\mathbf{C}$, hyperparameters N_{ft}, N_s, E

Initialize the parameters in GMP (ϕ), RL-HA (ψ) and SGA-MG (θ);

`/* Pre-training Stage */`

Pre-train GMP by unsupervised learning with (7.15);

Pre-train SGA-MG by supervised learning with (7.16);

`/* Initialize RL */`

Initialize the replay buffer $\mathcal{D}$;

Initialize the RL-step index $i \leftarrow 0$;

`/* Formal-training Stage */`

for *epoch* $\leftarrow$ *1,2,...,E* **do**

  Generate rollout ξ with ϕ, ψ, θ;

  Add rollout ξ into replay buffer $\mathcal{D}$;

  $i \leftarrow i + 1$;

  `/* Train RL-HA */`

  if $i > N_s$ **then**

    Sample a rollout ξ' from $\mathcal{D}$;

    Update policy and ψ with DDQN;

  end

  `/* Finetune SGA-MG */`

  for $m \leftarrow 1, 2, ..., N_{ft}$ **do**

    Sample a case of $\mathbf{X}_{t-T_h+1:t}$ and $\mathbf{C}$;

    Use GMP to obtain node attributes on $\mathcal{G}$;

    Use RL-HA to generate $\mathcal{G}'$;

    Use SGA-MG to generate $\hat{\mathbf{X}}_{t+1:t+T_f}$;

    Compute loss by equation (7.16);

    Update θ by back-propagation;

  end

end

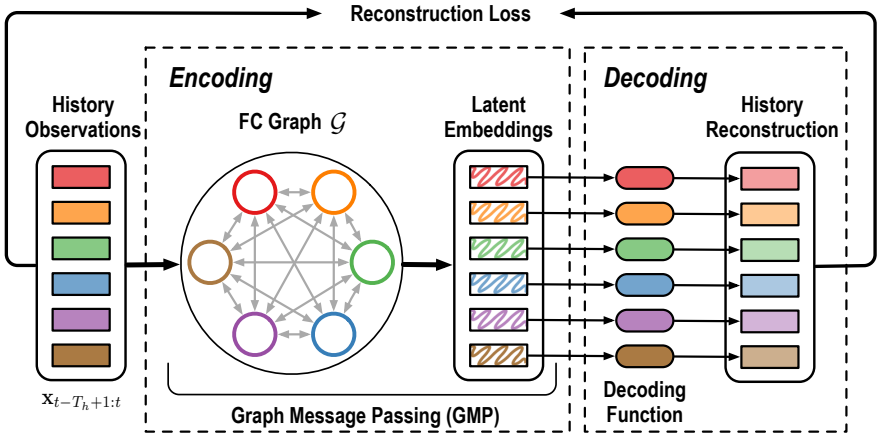


Figure 7.3: The diagram of the auto-encoder structure for pre-training the GMP module, which consists of an encoding procedure and a decoding procedure.

Pre-Training Stage: SGA-MG Module

In order to enable informative initial reward for the RL-HA module, we pre-train the SGA-MG module with a fully connected topology. The model architecture is the same as in Figure 7.2 except that the GAT is applied to a fully connected graph. The loss function is a standard mean squared error loss, which is calculated as

$$L_{\text{SGA-MG}} = \frac{1}{NT_f} \sum_{i=1}^{N} \sum_{t'=t+1}^{t+T_f} ||\mathbf{x}_{t'}^i - \hat{\mathbf{x}}_{t'}^i||^2. \tag{7.16}$$

Formal-Training Stage

In the formal-training stage, we initialize the GMP and SGA-MG with pre-trained parameters. Then we perform an alternating training strategy to train the RL-HA and SGA-MG alternatively until convergence. The detailed pseudo-code of the training pipeline of the whole framework is provided in Algorithm 3.

7.5 Multi-Agent Interacting System: Experiments

In this work, we validated the proposed hybrid attention approach on the datasets in multi-agent systems in different domains, including a synthetic physics motion dataset and a real-world driving dataset (nuScenes [11]).

For the mixed particle simulation, since we have access to the ground truth of the particle charging state (i.e., positively/negatively-charged or uncharged), we can quantitatively and

qualitatively evaluate the model performance in terms of both attention learning and mean square error (MSE) of particle position prediction.

For the nuScenes driving dataset, we employ three widely used metrics: minimum average displacement error (minADE_{20}), minimum final displacement error (minFDE_{20}) and miss rate (MR) in trajectory prediction [89]. We report the results for vehicles and pedestrians separately. We also provide ablative analysis on these datasets.

7.5.1 Datasets and Evaluation Metrics

Mixed Particle Simulation

In the mixed particle system, there are two types of particles: charged particles and uncharged particles. The charged particles are uniformly sampled to carry positive or negative charges $q_i \in \{\pm q\}$, which interact with each other via Coulomb forces, which is given by

$$F_{ij} = C \cdot (q_i \cdot q_j) \cdot \frac{r_i - r_j}{||r_i - r_j||^3}, \tag{7.17}$$

where C is a constant. These charged particles may either attract or repel each other, although the forces may be weak when the distance in between is large. However, the motions of uncharged particles are totally independent since there is no force applied to them. They move straight with a constant velocity the same as the random initialization. In this paper, we have 3 charged particles and 3 non-charged ones in each case. We generated 8k samples for training, and 4k samples for validation and testing, respectively.

The simulation process of charged particles is mainly adopted from NRI [77]. In order to prevent the force divergence issue when two particles move with a very small distance, we adopt the same strategy as suggested in [77] to avoid numerical issues, which is to clip the forces to some maximum threshold. Despite that this is not exactly physically precise, the generated trajectories are not distinguishable to human observers and do not affect the conclusion of the paper.

The evaluation metric for trajectory prediction in this experiment is the mean squared error, which is calculated by

$$\text{MSE} = \frac{1}{NT_f} \sum_{i=1}^{N} \sum_{t'=t+1}^{t+T_f} ||\mathbf{x}_{t'}^i - \hat{\mathbf{x}}_{t'}^i||^2. \tag{7.18}$$

nuScenes Dataset [11]

The nuScenes dataset is a widely used large-scale driving dataset with a full set of sensor suite, which was collected in Boston and Singapore. It provides the point cloud information, trajectory annotations of heterogeneous traffic participants (e.g., cars, pedestrians and cyclists), as well as the map information. We processed the original data into segments with a length of 6 seconds to construct our dataset (2 seconds as history and 4 seconds as

future). We generated about 8k samples for training, and 2k samples for validation and testing, respectively.

We adopt the standard evaluation metrics in trajectory prediction, which include minADE_K, minFDE_K and miss rate (MR). In this paper, we use the same $K = 20$ as most baselines. The MR(@dm) is calculated by

$$\text{MR}(@d\text{m}) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(\min_k ||\mathbf{x}_{t+T_f}^{i,k} - \hat{\mathbf{x}}_{t+T_f}^{i,k}||_2 > d\right), \tag{7.19}$$

where $\mathbb{I}(\cdot)$ denotes an indicator function to indicate whether the current case is a failure case, and d is a manually defined threshold.

7.5.2 Baseline Methods

Ablative Baselines

- Ours (true+soft): This is the model that only applies soft graph attention to the true relation graph. Note that this model is only used for the experiments on mixed particle simulation since the true relation graph is not accessible in real-world scenarios and dataset.
- Ours (full+soft): This is the model that only applies soft graph attention to a fully connected relation graph.
- Ours (ELBO+soft): This is the model where only the RL-HA module is replaced by an ELBO based module with other modules not changed, which is trained end-to-end. The purpose of this ablation setting is to provide a baseline based on an alternative way to obtain hard attention.
- Ours (hybrid, static): This is the model that applies both RL hard attention to obtain the inferred relation graph and soft attention to figure out relative importance. The inferred relation graph remains static during the whole prediction horizon and the model performs one-shot prediction.
- Ours (hybrid, dynamic): This model setup is very similar to Ours (hybrid, static). The difference is that the model performs iterative prediction with a fixed horizon of sliding window. The inferred relation graph is dynamically evolving over time.
- Ours (hybrid): This model setup is only used for human motion prediction. The RL hybrid attention and soft attention work together to extract informative history features for the motion generator in [189].

For Mixed Particle Simulation

- Corr. (LSTM): The baseline method for edge recognition used in [77].
- LSTM (single) / LSTM (joint): The baseline methods for state sequence prediction in [77].
- NRI: The NRI model with static latent interaction graph [77].
- DNRI: A model for neural relational inference with dynamic interaction graphs [44].
- Supervised: Since the true relation graph is accessible in the simulation data, we can use supervised learning to train a binary classifier to infer the existence of the edges in the graph. The ground truth labels include 0 (w/o edge) and 1 (w/ edge).

7.5.3 Implementation Details

In this section, we introduce the details of model architecture, hyperparameters and specific experimental settings for each dataset. We trained the models for 100 epochs for both particle simulation and nuScenes dataset. They shared the same model architecture and specific details of model components are introduced below:

- GMP: The state embedding functions f_s^m, f_n^m, node attribute update function f_v, and the encoding function f_{enc} are all three-layer MLPs with hidden size = 64. During the pre-training stage, the decoding function is also a three-layer MLP with hidden size = 64. The context embedding function f_c is a four-layer convolutional block with kernel size = 5. The layer structure is [[Conv, ReLU, Conv, ReLU, Pool], [Conv, ReLU, Conv, ReLU, Pool]]. The context embedding is only applied to the "traffic scenario", where the context information is the projected point cloud images.
- RL-HA: The maximum RL-step η is set to 10. In the total reward, the hyperparameters are $\beta_{\text{imp}} = \beta_{\text{sti}} = \beta_{\text{pun}} = 0.01$. We also set $\Omega_s = \Omega_p = 1.0$. In the "traffic scenario", we define a "success case" where the end-point error is less than the miss rate threshold and a "failure case" as the opposite. The stimulation reward is applied when the current case changes from "failure case" to "success case", and the punishment reward is applied for the opposite situation.

  All the coefficients and thresholds in reward function were decided empirically. $R_{\text{reg}}/R_{\text{imp}}$ reward the overall improvement of prediction while $R_{\text{sti}}/R_{\text{pun}}$ are mainly related to endpoint prediction. We found that increasing the weights of $R_{\text{sti}}/R_{\text{pun}}$ could improve both minADE and minFDE in a certain range while overly large weights could have negative effects on minADE. The miss rate thresholds should be specifically decided for various types of agents.

- SGA-MG: The Embedding LSTM (E-LSTM) and Generation LSTM (G-LSTM) both have a hidden size of 128. For the particle simulation, we performed one-shot prediction with a static inferred relation graph; for the nuScenes dataset, we performed progressive forecasting with a sliding window of 2 seconds (10 frames) with dynamic relation graphs.

In all the experiments, we used a batch size of 32 and the Adam optimizer with an initial learning rate of 0.001 to train the models with a single NVIDIA Quadro RTX 6000 GPU.

7.5.4 Synthetic Simulation: Mixed Particle System

We applied our approach to a simulated physical system with mixed charged and uncharged particles, where the charged ones are randomly assigned positive or negative charges with a uniform distribution. The particles with the same charge repel and the ones with opposite charges attract according to the fundamental law of electricity. The uncharged particles perform uniform motion independently. The proposed hybrid-attention based prediction model is expected to learn the relations between particles and forecast their long-term motions.

The goal of RL-based hard attention mechanism is to infer whether there is a force between a pair of particles, which is essentially a binary classification task. The recognition of the force type (repel or attract) will be handled implicitly by the soft attention counterpart. The relation recognition results of different approaches are compared in Table 7.1, where the standard deviations are calculated based on three independent runs. It shows that the supervised learning method which directly trains a binary classifier with ground truth labels performs the best in terms of all the evaluation metrics. However, the accessibility of true relation labels is highly limited in real-world tasks. Among the approaches that do not require true labels during training, our method achieves the highest recognition accuracy, which improves by 6.8% over the strongest baseline DNRI and 4.7% over the ablative baseline *Ours (ELBO+soft)*. Based on the experiments, end-to-end training with ELBO tends to be more stable and converges faster than RL in general, but it did not achieve a better final performance. The accuracy alone cannot sufficiently prove the superiority due to the data imbalance of the two classes. Therefore, we also compared the precision, recall and F1-score. Our method achieves consistently better results on these metrics.

We further predicted the particle positions at the future 50 time steps given the historical observations of 30 time steps. The comparison of mean squared error (MSE) of different methods is provided in Figure 7.4. The solid lines and shaded areas represent the average value and standard deviation of three runs, respectively. *Ours (true+soft)* attains the smallest MSE consistently over the whole prediction horizon due to access to the true graph structures, which serves as a performance upper bound. *Ours (hybrid)* achieves a smaller MSE than *Ours (full+soft)* / *Ours (ELBO+soft)* and even approaches the performance of *Ours (true+soft)*, which indicates that the RL-based hard attention mechanism indeed helps on the relation recognition, which further improves prediction performance. *Ours (hybrid)* also outperforms NRI and DNRI where the interaction graph is encoded to a latent space.

Table 7.1: Evaluation (Mean±Std in %) of Relation Recognition (Mixed Particle System).

	Accuracy	Precision	Recall	F1-score
Corr. LSTM [77]	71.85±0.39	54.52±0.24	60.17±0.16	57.21±0.33
NRI [77]	85.47±0.98	78.57±0.59	64.71±0.66	70.97±0.51
DNRI [44]	87.54±1.66	82.48±0.91	67.23±0.78	74.08±0.83
Ours (ELBO+soft)	89.31±0.46	87.29±0.21	73.52±0.30	79.82±0.25
Ours (hybrid)	**93.52**±1.25	**92.34**±0.87	**80.55**±0.92	**86.04**±0.74
Supervised	97.48±0.23	95.24±0.09	89.55±0.14	92.31±0.11

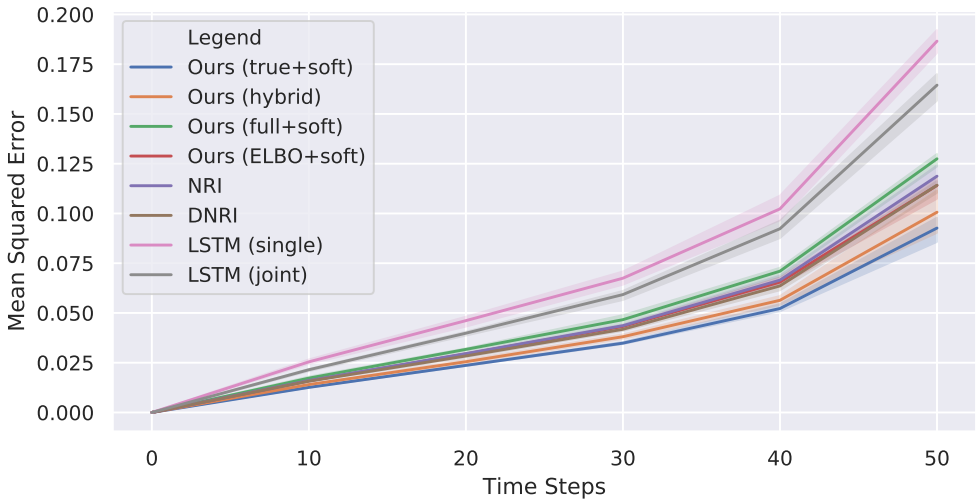


Figure 7.4: Mean squared error (MSE) in predicting future positions of mixed particles.

We also visualize the predicted trajectories and attention maps of typical cases in Figure 7.5 with descriptions in the caption. The results show that the RL agent can figure out truly relevant agents with a high accuracy and the soft attention counterpart further determines the relative significance of the selected entities. However, applying soft attention alone on fully connected graphs sometimes assigns trivial weights or even large weights to irrelevant agents as shown in the "Soft Attention" columns, which weakens interpretability and performance of the model. We address two interesting cases in (e) and (f), where the RL agent only selects a subset of interacting agents or even an empty set, but the system still generates very good prediction. The reason is that the distance between the interacting particles is too large, which leads to the ignorance of the weak forces by the RL agent. Particularly in (f), all the particles perform nearly uniform motions which results in an empty hybrid attention map. This demonstrates the capability of distinguishing key information of the hard attention mechanism.

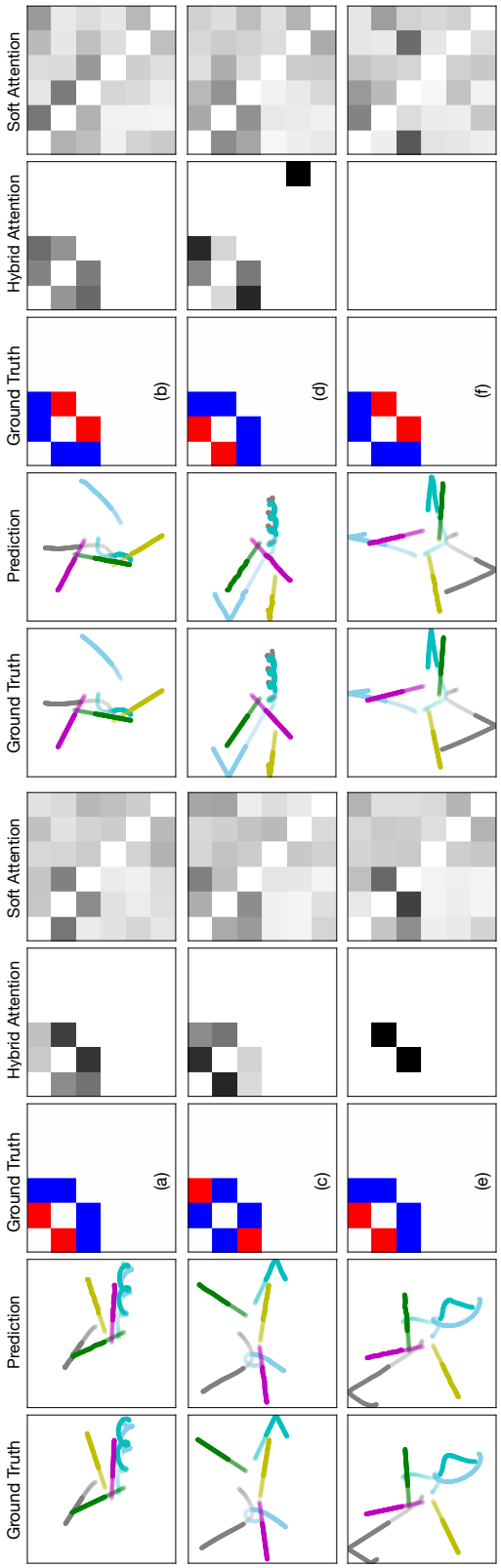


Figure 7.5: Visualization of particle trajectories and attention maps. The semi-transparent segments represent the observation of 30 time steps and the solid ones represent the prediction of future 50 time steps. In the ground truth relation map, red denotes "repel" and blue denotes "attract". In the attention maps, darker color indicates larger weight. Note that we do not consider self-attention (i.e. zero diagonal).

7.5.5 nuScenes Dataset: Traffic Scenarios

We validated our RAIN framework on the nuScenes dataset, which handles long-term prediction of heterogeneous traffic participants (i.e., vehicles and pedestrians). We predicted the future 4.0s (20 frames) given the history observations of 2.0s (10 frames). The comparison of quantitative results is shown in Table 7.2 (vehicles) and Table 7.3 (pedestrians). Note that we simultaneously included both types of agents during training, while reported the test results separately as done in [152].

Table 7.2: Comparison of minADE_{20} / minFDE_{20} (Meters) and Miss Rate@2.0m (MR, %) of Vehicle Trajectory Prediction.

Method	1.0s	2.0s	3.0s	4.0s	MR
NRI [77]	0.19/0.23	0.37/0.57	0.58/1.00	0.82/1.52	27.1
DNRI [44]	0.17/0.20	0.33/0.49	0.50/0.87	0.71/1.33	22.4
STGAT [62]	0.16/0.19	0.31/0.47	0.48/0.83	0.68/1.27	20.6
S-STGCNN [132]	0.18/0.21	0.34/0.52	0.53/0.92	0.75/1.41	24.7
Ours (full+soft)	0.07/0.10	0.17/0.41	0.40/0.82	0.63/1.28	21.0
Ours (ELBO+soft)	**0.04/0.07**	0.16/0.42	0.40/0.79	0.61/1.24	16.1
Ours (hybrid, static)	0.05/0.09	0.15/0.40	0.38/0.77	0.58/1.17	11.3
Ours (hybrid, dynamic)	0.06/0.09	**0.14/0.38**	**0.37/0.75**	**0.54/1.12**	**9.5**

Table 7.3: Comparison of minADE_{20} / minFDE_{20} (Meters) and Miss Rate@1.0m (MR, %) of Pedestrian Trajectory Prediction.

Method	1.0s	2.0s	3.0s	4.0s	MR
NRI [77]	0.10/0.13	0.22/0.32	0.35/0.58	0.48/0.82	28.5
DNRI [44]	0.09/0.12	0.20/0.31	0.32/0.53	0.44/0.78	25.6
STGAT [62]	0.08/0.11	0.18/0.27	0.29/0.48	0.40/0.71	17.8
S-STGCNN [132]	0.12/0.15	0.23/0.37	0.35/0.61	0.48/0.88	27.9
Ours (full+soft)	0.07/0.11	0.15/0.22	0.24/0.40	0.32/0.56	13.2
Ours (ELBO+soft)	**0.04/0.06**	0.13/0.20	0.21/0.39	0.30/0.56	12.9
Ours (hybrid, static)	0.06/0.08	0.12/0.20	0.19/0.36	0.29/0.54	11.7
Ours (hybrid, dynamic)	0.05/0.08	**0.11/0.18**	**0.17/0.34**	**0.26/0.51**	**9.8**

All the learning-based baseline methods consider the relation modeling via certain techniques. It is shown that *Ours (hybrid, dynamic)* achieves the smallest error and lowest miss rate in both vehicle and pedestrian prediction. We also present extensive ablation results. *Ours (full+soft)* achieves lower prediction error than STGAT with a similar model structure, especially in vehicle prediction. The reason is that the vehicle kinematics model is incorporated into the motion generator to propagate the vehicle state, which can guarantee the feasibility of generated trajectories. *Ours (hybrid, static)* and *Ours (ELBO+soft)* outperforms

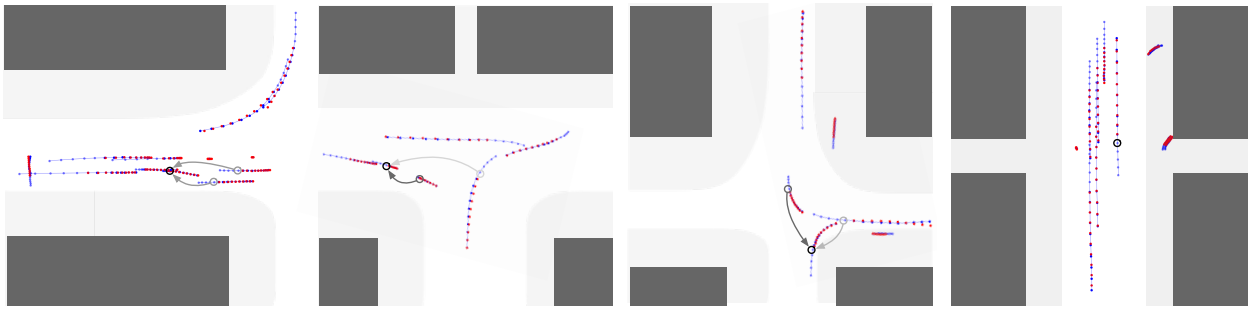


Figure 7.6: The visualization of testing cases in the nuScenes dataset. The light blue dots are history observations, dark blue dots are ground truth, and red dots are predictions with the minimum ADE. The black circles indicate the target agent and gray arrows indicate hybrid attention. The targets only attend to the agents with arrows selected by hard attention and darker colors imply larger weights. In the last case, there is no arrow, which implies that the model infers that the target is not influenced by any agent in the scene at the current frame.

Ours (full+soft), which implies the effectiveness of hard attention on selecting important agents. In particular, *Ours (hybrid, static)* achieves smaller prediction errors than *Ours (ELBO+soft)*, which implies the advantage of RL based hard attention over the ELBO based method. *Ours (hybrid, dynamic)* further reduces the error owing to the dynamic recognition of key elements, where the margin of improvement becomes larger as the prediction horizon increases.

The visualization of agent trajectories and hybrid attention weights in several typical test scenarios is provided in Figure 7.6. It is shown that our method can learn reasonable attention weights to exploit the key information, and generate plausible and accurate prediction hypotheses.

7.6 Human Skeleton Motions: Model

In this section, we address the application of our method to capture long-term temporal dependency in skeleton based human motions. Instead of selecting important or relevant entities introduced in Section 7.4, here we propose to utilize the RL-HA module to discriminate key information over the whole history horizon. It can be either frame-wise selection or segment-wise selection. We demonstrate an illustrative case study of the latter based on a state-of-the-art model proposed in [189].

More specifically, the model in [189] is employed as the soft attention based motion generator in our framework. The proposed RL hard attention mechanism is built on top of it, and they work as a whole to generate future human motions. First, an encoding function is applied to extract contextual representation of each history frame. Then, the

RL-HA module discriminates the key motion segments from history observations, and the soft attention mechanism further outputs relative significance of the selected segments for the current prediction.

Similar to RAIN for the multi-agent interacting systems, we also employ a double-stage training pipeline, including a pre-training stage and a formal-training stage. In the pre-training stage, we pre-train the parameters of a contextual encoder and the soft attention based motion generator. In the formal-training stage, we train the RL-HA module and finetune the motion generator alternatively.

We denote the complete history motions as $\mathbf{X}_{t-T_h+1:t}$ and the future motions as $\mathbf{X}_{t+1:t+T_f}$. We have the same assumption as [189] that $T_h > T_s + T_f$ where T_s is the length of the motion segments used to compute attention weights.

7.6.1 Pre-Training Stage

First, we use an auto-encoder structure to train an encoding function that can extract the contextual information from the complete history motion sequence. More formally, the auto-encoder can be written as

$$\mathbf{Z} = \text{Encoding}(\mathbf{X}_{t-T_h+1:t}), \tag{7.20}$$

$$\hat{\mathbf{X}}_{t-T_h+1:t} = \text{Decoding}(\mathbf{Z}), \tag{7.21}$$

where Encoding and Decoding functions are neural networks. The loss function of training the auto-encoder is the standard mean squared error reconstruction loss, which is calculated by

$$\text{MSE} = \frac{1}{JT_h} \sum_{j=1}^{J} \sum_{t'=t-T_h+1}^{t} ||\mathbf{x}_{t'}^j - \hat{\mathbf{x}}_{t'}^j||^2, \tag{7.22}$$

where J is the number of relative angles between joints in a skeleton. For the soft attention based motion generator, since the authors of [189] released their official code and pre-trained models, we directly load their pre-trained parameters in the formal-training stage.

7.6.2 Formal-Training Stage

In the formal-training stage, we alternatively train the RL-HA module and the motion generator. We define the motion segments in the same way as [189]. More specifically, we first divide the complete motion history $\mathbf{X}_{t-T_h+1:t}$ into $T_h - T_s - T_f + 1$ segments $\{\mathbf{X}_{i:i+T_s+T_f-1}\}_{i=t-T_h+1}^{t-T_s-T_f+1}$, each of which contains $T_s + T_f$ consecutive frames of human poses. We use the same setting as [189], where the motion generator exploits the past T_s frames to predict the future T_f frames. The first T_s frames of each segment is used as a key, and the whole segment is then the corresponding value. The query is defined as the latest segment $\mathbf{X}_{t-T_s+1:t}$.

RL-HA Module

In the domain of forecasting human skeleton motions, the RL-HA module is expected to select the key history motion segments for the current prediction based on the latest observation segment. Then the soft attention mechanism in [189] will further rank the relative importance of the selected key segments, which is employed by the motion predictor to generate future motions.

The selection of key segments naturally fits into a reinforcement learning framework. The definition of observations, actions and reward functions of the RL-agent are elaborated in the following.

Observations: The observation O of RL-agent at RL-step η ($\leq T_{\text{RL}}$) includes a tuple of key, query, contextual information $\mathbf{Z}$ as well as the current segment selection status s_i (0: "retained" or 1: "discarded"). T_{RL} is the upper bound of RL-steps. The observation O_η is obtained by

$$O_\eta = [f_k(\mathbf{X}_{i:i+T_s-1}),\ f_q(\mathbf{X}_{t-T_s+1:t}),\ \mathbf{Z},\ s_{i,\eta}]\,, \tag{7.23}$$

where f_k and f_q are mapping functions modeled by neural networks. Note that the dimension of O_η only depends on the dimensions of key, query and contextual information, which enables the applicability to the scenarios with varying numbers of history motion segments. The policy network of RL-agent takes the observation O_η as input and decides the action at each RL-step.

Actions: There are two possible actions for the RL-agent: "staying the same" (action 0) and "changing to the opposite" (action 1). At each RL-step, the RL-agent makes decision for each history motion segment. The policy can be written as $a = \pi(O)$. We do not enforce any constraints on the selection of motion segments, i.e., there is no lower / upper bound on the number of selected segments. The actions of RL-agent may change the key motion segments after each RL-step, which further influences the soft attention based motion generator.

Rewards: The reward consists of two parts: regular reward R_{reg} and improvement reward R_{imp}. More specifically, the *regular reward* is the negative mean squared error of future predictions calculated by

$$R_{\text{reg},\eta} = -\frac{1}{J}\sum_{j=1}^{J}\sum_{t'=t+1}^{t+T_f} ||\mathbf{x}_{t'}^j - \hat{\mathbf{x}}_{t',\eta}^j||^2. \tag{7.24}$$

The *improvement reward* encourages the decrease of prediction error via applying a sign function to the error change between consecutive RL-steps, which is obtained by

$$R_{\text{imp},\eta} = \text{sign}(R_{\text{reg},\eta} - R_{\text{reg},\eta-1}). \tag{7.25}$$

The whole reward is obtained by $R_\eta = R_{\text{reg},\eta} + \beta_{\text{imp}} R_{\text{imp},\eta}$, where β_{imp} is a hyperparameter.

Alternating Training Strategy

The contextual encoding function is initialized with well-trained parameters in the pre-training stage and fixed in the formal-training stage. The soft attention based motion generator is initialized with the pre-trained model in [189]. We perform alternating optimization of RL-HA and motion generator (MG) modules: (a) train the RL-HA module with fixed parameters of MG with Double Deep Q-Learning (DDQN) method [50]; (b) finetune MG with fixed parameters of RL-HA using back-propagation methods.

7.7 Human Skeleton Motions: Experiments

7.7.1 Dataset

The Human3.6M dataset [63] is a widely used skeleton-based human motion dataset for pose estimation and motion forecasting, which includes 15 different activities performed by 7 professional actors. The human skeleton information is provided in two representations: 3D joint positions and joint angles. The skeleton has 32 joints, the 3D coordinates of which can be computed by applying the forward kinematics. As in [189], we also down-sample the motion sequences to 25 frames per second, and remove the global rotation, translation and constant angles. In this work, we chose relative angles between joint to represent the skeleton state.

7.7.2 Implementation Details

We trained the models for 50 epochs on the Human3.6M dataset. We adopted exactly the same experimental settings as [189]. More specifically, during training, we trained the model to predict the future 10 frames based on the history 50 frames and the attention weights are calculated based on the latest observation sequence with 10 frames. During testing, we enabled progressive long-term prediction with a sliding window to generate future 25 frames.

Specific details of model components are introduced in the following:

- Encoding / Decoding (pre-training stage): They are three-layer MLPs with hidden size = 128.
- RL-HA: The maximum RL-step η is set to 10. In the total reward, the hyperparameter is $\beta_{\text{imp}} = 0.01$.
- Motion Generator: We adopted the same model architecture and hyperparameters as in [189].

Table 7.4: Comparison of mean angle errors (MAE) of different methods for both short-term ($\leq$ 400 milliseconds) and long-term prediction (1k milliseconds) on the 15 actions in the Human3.6M dataset. "–" indicates that the original paper did not report the result.

Motion	Walking					Eating					Smoking					Discussion				
milliseconds	80	160	320	400	1k	80	160	320	400	1k	80	160	320	400	1k	80	160	320	400	1k
Res-sup. [128]	0.27	0.46	0.67	0.75	1.03	0.23	0.37	0.59	0.73	1.08	0.32	0.59	1.01	1.10	1.50	0.30	0.67	0.98	1.06	1.69
CSM [92]	0.33	0.54	0.68	0.73	0.92	0.22	0.36	0.58	0.71	1.24	0.26	0.49	0.96	0.92	1.62	0.32	0.67	0.94	1.01	1.86
Traj-GCN [116]	0.18	0.32	0.49	0.56	0.67	0.17	0.31	0.52	0.62	1.12	0.22	0.41	0.84	0.79	1.57	0.20	**0.51**	0.79	0.86	1.70
DMGNN [105]	0.18	0.31	0.49	0.58	0.75	0.17	0.30	0.49	0.59	1.14	0.21	0.39	0.81	0.77	1.52	0.26	0.65	0.92	0.99	**1.45**
LTD-10-10 [126]	0.18	0.31	0.49	0.56	0.64	**0.16**	0.29	0.50	0.62	1.10	0.22	0.41	0.86	0.80	1.58	0.20	**0.51**	**0.77**	**0.85**	1.75
HisRepItself [189]	0.18	0.30	0.46	0.51	0.64	**0.16**	0.29	0.49	0.60	1.10	0.22	0.42	0.86	0.80	1.58	0.20	0.52	0.78	0.87	1.63
Ours (hybrid)	**0.17**	**0.28**	**0.43**	**0.49**	**0.60**	**0.16**	**0.26**	**0.45**	**0.56**	**1.01**	**0.20**	**0.38**	**0.80**	**0.74**	**1.48**	**0.19**	0.52	0.82	0.91	1.64
Motion	Directions					Greeting					Phoning					Posing				
millisecond	80	160	320	400	1k	80	160	320	400	1k	80	160	320	400	1k	80	160	320	400	1k
Res-sup [128]	0.41	0.64	0.80	0.92	–	0.57	0.83	1.45	1.60	–	0.59	1.06	1.45	1.60	–	0.45	0.85	1.34	1.56	–
CSM [92]	0.39	0.60	0.80	0.91	1.45	0.51	0.82	1.21	1.38	1.72	0.59	1.13	1.51	1.65	1.81	0.29	0.60	1.12	1.37	2.65
Traj-GCN [116]	0.26	0.45	0.70	0.79	–	**0.35**	0.61	0.96	1.13	–	0.53	1.02	1.32	1.45	–	0.23	0.54	1.26	1.38	–
DMGNN [105]	**0.25**	0.44	0.65	0.71	–	0.36	0.61	**0.94**	**1.12**	–	0.52	0.97	1.29	1.43	–	0.20	0.46	1.06	1.34	–
LTD-10-10 [126]	0.26	0.45	0.71	0.79	1.35	0.36	**0.60**	0.95	1.13	1.59	0.53	1.02	1.35	1.48	1.74	**0.19**	0.44	**1.01**	1.24	2.55
HisRepItself [189]	**0.25**	**0.43**	**0.60**	**0.69**	**1.27**	**0.35**	**0.60**	0.95	1.14	**1.57**	0.53	1.01	1.31	1.43	1.68	**0.19**	0.46	1.09	1.35	**2.32**
Ours (hybrid)	**0.25**	0.46	0.65	0.75	1.34	**0.35**	0.62	0.99	1.21	1.64	**0.49**	**0.95**	**1.23**	**1.35**	**1.59**	**0.19**	**0.43**	1.02	**1.23**	2.46
Motion	Purchases					Sitting					Sitting Down					Taking Photo				
millisecond	80	160	320	400	1k	80	160	320	400	1k	80	160	320	400	1k	80	160	320	400	1k
Res-sup [128]	0.58	0.79	1.08	1.15	–	0.41	0.68	1.12	1.33	–	0.47	0.88	1.37	1.54	–	0.28	0.57	0.90	1.02	–
CSM [92]	0.63	0.91	1.19	1.29	2.52	0.39	0.61	1.02	1.18	1.67	0.41	0.78	1.16	1.31	2.06	0.23	0.49	0.88	1.06	1.40
Traj-GCN [116]	0.42	0.66	1.04	1.12	–	0.29	0.45	0.82	0.97	–	0.30	0.63	0.89	1.01	–	0.15	0.36	0.59	0.72	–
DMGNN [105]	0.41	**0.61**	1.05	1.14	–	0.26	0.42	0.76	0.97	–	0.32	0.65	0.93	1.05	–	0.15	0.34	0.58	0.71	–
LTD-10-10 [126]	0.43	0.65	1.05	1.13	2.27	0.29	0.45	0.80	0.97	1.52	0.30	0.61	0.90	1.00	1.67	0.14	0.34	0.58	0.70	1.05
HisRepItself [189]	0.42	0.65	**1.00**	**1.07**	**2.22**	0.29	0.47	0.83	1.01	1.55	0.30	0.63	0.92	1.04	1.70	0.16	0.36	0.58	0.70	1.08
Ours (hybrid)	0.41	0.63	1.07	1.14	2.25	**0.24**	**0.40**	**0.75**	**0.96**	**1.47**	**0.27**	**0.58**	**0.85**	**0.97**	**1.58**	**0.14**	**0.32**	**0.53**	**0.64**	**0.91**
Motion	Waiting					Walking Dog					Walking Together					Average				
millisecond	80	160	320	400	1k	80	160	320	400	1k	80	160	320	400	1k	80	160	320	400	1k
Res-sup [128]	0.32	0.63	1.07	1.26	–	0.52	0.89	1.25	1.40	–	0.27	0.53	0.74	0.79	–	0.40	0.69	1.04	1.18	–
CSM [92]	0.30	0.62	1.09	1.30	2.50	0.59	1.00	1.32	1.44	1.92	0.27	0.52	0.71	0.74	1.28	0.38	0.68	1.01	1.13	1.77
Traj-GCN [116]	0.23	0.50	0.92	1.15	–	0.46	0.80	1.12	1.30	–	0.15	0.35	0.52	0.57	–	0.27	0.53	0.85	0.96	–
DMGNN [105]	0.22	0.49	0.88	1.10	–	0.42	0.72	1.16	1.34	–	0.15	0.33	**0.50**	0.57	–	0.27	0.52	0.83	0.95	–
LTD-10-10 [126]	0.23	0.50	0.91	1.14	2.37	0.46	0.79	1.12	1.29	1.86	0.15	0.34	0.52	0.57	1.16	0.27	0.52	0.83	0.95	1.62
HisRepItself [189]	0.22	0.49	0.92	1.44	2.30	0.46	0.78	1.05	1.23	1.82	**0.14**	0.32	**0.50**	**0.55**	1.16	0.27	0.52	0.82	0.94	1.57
Ours (hybrid)	**0.21**	**0.46**	**0.83**	**1.03**	**2.18**	**0.41**	**0.71**	**1.01**	**1.15**	**1.72**	**0.14**	**0.31**	**0.50**	0.56	**1.15**	**0.25**	**0.48**	**0.79**	**0.91**	**1.51**

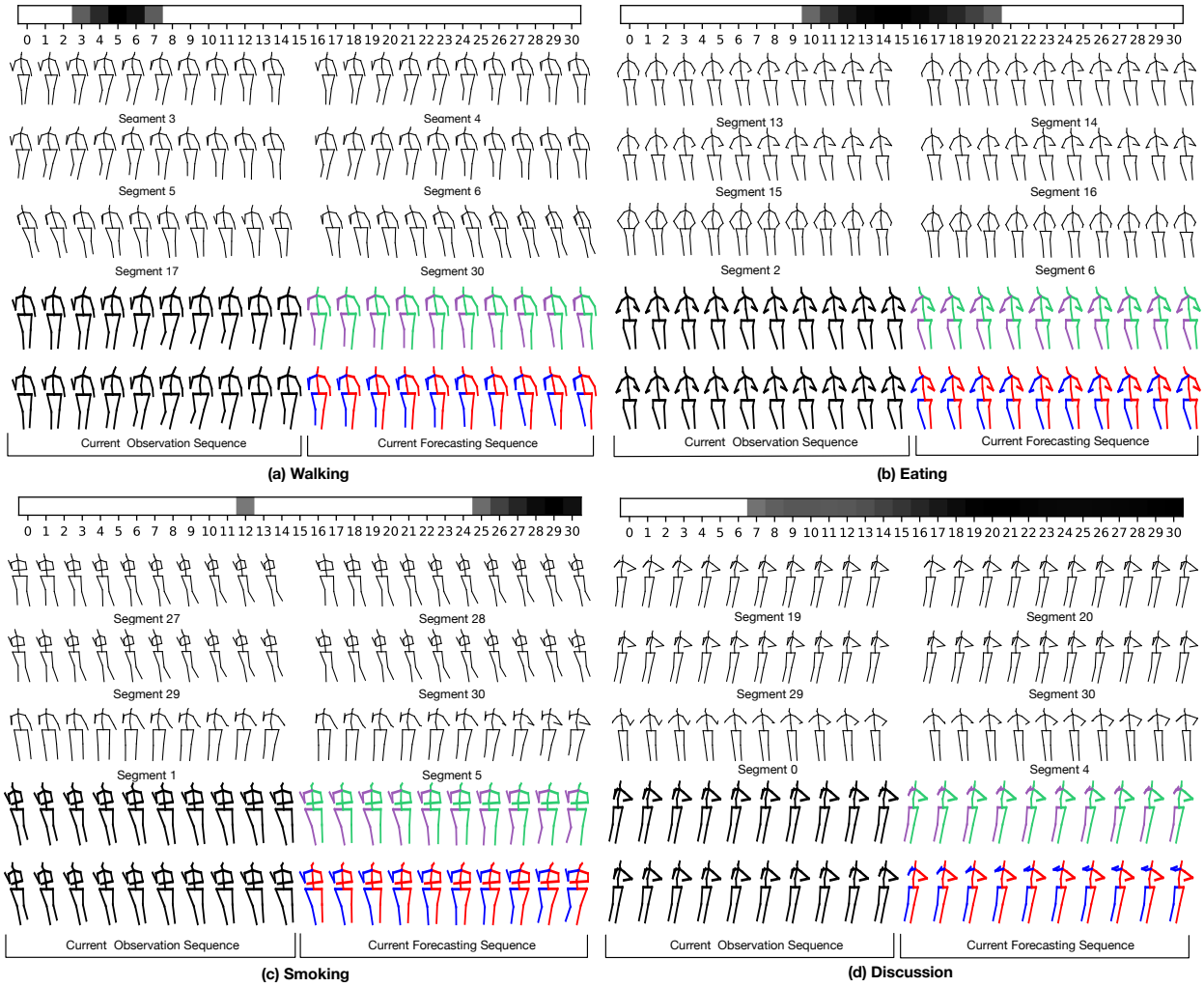


Figure 7.7: The visualization of human skeleton motion forecasting of four typical actions with hybrid attention maps. The black skeletons at the bottom are the latest observation sequences which are used to calculate current attention weights. The purple-green skeletons are the prediction hypotheses of our method. The blue-red skeletons are the ground truth. In our experimental setting, for each case there are 31 available history motion segments with a length of 10 frames for the RL hard attention module to select and the soft attention is only applied to the selected segments. In the hybrid attention maps, darker colors indicate larger attention weights. White color means the corresponding motion segment is not selected as key information. Best viewed in color.

7.7.3 Results and Analysis

We adopted the same experimental setting as the baseline approaches for fair comparison, which is to forecast the future 25 frames based on the past 50 frames. The model was trained to only predict 10 frames during the training phase, while the prediction was recursively applied by using predictions as new observations during the testing phase. The forecasting results are shown in Table 7.4. Among the baselines, HisRepItself [189] yields the previous state-of-the-art performance, which also serves as an ablation model with only soft attention on the motion history. In general, the results show that *Ours (hybrid)* achieves the best performance both in average and for most actions in terms of both short-term and long-term forecasting accuracy. In particular, *Ours (hybrid)* outperforms HisRepItself in average, indicating the additional benefits brought by the hard attention mechanism.

It is shown that *Ours (hybrid)* achieves the smallest MAE in most actions as well as in average compared with baselines. The action "Directions" is an interesting exception where HisRepItself outperforms our method. A potential reason is that in the "Directions" action, there is no clear pattern of the temporal dependency between the current observation and previous motions, which makes it hard for the RL-agent to discriminate and select the appropriate history motion segments to pay attention to.

We also visualize the prediction of human skeletons and the learned hybrid attention weights in typical testing cases in Figure 7.7. It shows that our method can accurately forecast the human motions. More specifically, we visualize the top four motion segments with the largest four hybrid attention weights in each case (i.e., the motion segments in the first two rows). It shows that these segments have similar patterns with the current observation sequence and thus are selected as key information, which implies that the learned hybrid attention is reasonable and interpretable. We also visualize some irrelevant segments that are discarded by the model (i.e., the motion segments in the third row). It can be easily found that these segments are dissimilar to the latest observation sequence, thus are unimportant for the current prediction.

In addition, in case (a), (b) and (c), the learned hybrid attention is sparse, which implies that the model is able to effectively discriminate and only focus on the key information. An interesting exception is case (d), where most history segments are selected. A potential reason is that most history segments in this case are very similar to the current observation sequence, which leads the model to take them all into account for current prediction.

7.8 Chapter Summary

In this chapter, we present a generic motion forecasting framework with key information selection and ranking procedures. The former is realized by a reinforcement learning based hard attention mechanism for the purpose of discriminating relevant information for the prediction from complete spatio-temporal observations. The latter is fulfilled by a soft attention mechanism for the purpose of determining relative importance of the selected key

elements. These dual procedures can be applied recurrently to dynamically adjust the focus of the model as the situation evolves over time. A double-stage training pipeline with an alternating training strategy is employed to train different parts of the model, which proves to be effective and stable in the experiments. The general framework is instantiated and applied to multi-agent trajectory prediction and human motion forecasting, which achieves state-of-the-art performance in terms of a wide range of evaluation metrics in different domains.

Chapter 8

IOS^2: Important Object Identification with Semi-Supervised Learning

8.1 Introduction

The autonomous agents that navigate in complex, highly dynamic environments need to accurately identify the most important objects in the scene which are relevant to their decision making and motion planning. In particular, autonomous vehicles should be able to figure out which *dynamic objects* (e.g., vehicles, pedestrians, cyclists) and *static objects* (e.g., traffic lights, stop signs) are critical to perceive the environment, detect potential risks, and determine the current action given their specific intentions/goals. Human drivers tend to focus on a subset of important surrounding objects when driving in a dense area. Similarly, the limited onboard computational resources can be allocated more efficiently on the perception and reasoning of the identified important objects. Moreover, this function can also enable the advanced driver-assistance systems to warn the drivers about risk objects in dangerous situations.

Existing related works can be mainly divided into three categories. First, some works focus on predicting the driver's gaze by imitating human drivers [127, 137]. The gaze information can be obtained by mounting a camera on the driver's head. However, most of these methods only provide pixel/region-level attention without indicating the importance of each object/instance. Driver gaze tends to be sequential and limited to a single region at a certain moment, while there may be multiple important objects out of the focused region simultaneously. Moreover, human drivers may not always pay attention to the truly important objects, thus degrades the reliability of the information. Second, some works attempt to train an attention-based model with specific tasks such as trajectory forecasting [179] and end-to-end driving [25, 2], in which there is no explicit supervision on the learned attention. Third, some recent works attempt to identify important objects by providing explicit supervision on the object importance with human annotations to inject human knowledge, in which the models are trained by standard supervised learning [39, 209]. However, these approaches

demand a large amount of labeled data. Our proposed method lies in this direction of research, but we propose to utilize semi-supervised learning techniques to reduce human efforts and enable the model to learn an unlimited amount of unlabeled data. Different from [39, 209] which only consider dynamic traffic participants, we also include traffic lights/signs in the driving scenes to enable semantic reasoning of the environment.

Learning object importance from human-labeled data can be naturally formulated as a binary classification problem, where each object is classified as important (1) or unimportant (0). However, since the importance of a certain object is not totally independent from others (e.g., the importance of a certain object may decrease given the existence of another object), it is necessary to reason about the relation among entities before the final classification. Therefore, we leverage a graph representation and message passing operations to extract relational features.

Moreover, since the behavior of ego vehicle may be influenced by the important objects, it may in turn provide helpful cues for important object identification. For example, if the ego vehicle is waiting before a green light, there is likely at least one important object which prevents it from moving forward. Therefore, we propose to use auxiliary tasks of ego behavior prediction to provide additional supervision signals for the importance classification. Equipped with the auxiliary branches, our framework is able to identify important objects in the scene and infer the ego behavior simultaneously.

The main contributions of this work are summarized as follows:

- We propose a novel method for important object identification in the egocentric driving scenarios with relational reasoning on the objects in the scene.
- We propose to employ a modified semi-supervised learning algorithm with a ranking-based strategy for the pseudo-label generation to enable the model to learn from unlabeled datasets in addition to a human-labeled dataset. To the best of our knowledge, we are the first to apply a modified semi-supervised learning algorithm to important object/people identification for autonomous driving.
- We propose to leverage the auxiliary tasks of ego vehicle behavior prediction to provide additional supervision signals and further improve the accuracy of importance estimation.
- We validate the proposed method on a public urban driving dataset with a detailed ablative study and comparison with baselines, which demonstrates the effectiveness of each model component and the training strategy.

8.2 Related work

8.2.1 Important Object/People Identification

Identifying important individuals automatically from a set of entities has attracted increasing research efforts in recent years, which can be applied to a wide range of application domains. While many visual saliency detection approaches have been proposed to solve various computer vision tasks such as visual question answering [5, 202], scene understanding [171], video summarization [68], most of them only provide pixel-level importance estimation without being aware of individual object instances. Some recent papers put more emphasis on the object/instance-level importance estimation and their applications. In [107], Li et al. proposed a framework for automatically detecting important people in still images of social events. In [162], Silver et al. introduced an algorithm to select a small set of important objects from large problem instances for a particular goal in planning problems. In the autonomous driving domain, Li et al. [91] presented a model to identify risk objects for risk assessment via causal inference. Gao et al. [39] proposed a method to estimate importance of traffic participants in egocentric driving videos, and Zhang et al. [209] further considered the interactions among the involved entities. However, most of these models are trained by supervised learning which demand a large amount of labeled data.

8.2.2 Semi-supervised Learning (SSL)

Learning from partially labeled data has emerged as an exciting research direction in deep learning, especially in classification problems [215, 69, 57]. It enables the models to effectively learn from a labeled data together with an unlimited amount of unlabeled data, reducing the efforts of human annotation and enlarging learning resources [136]. The typical SSL methods can be broadly divided into the following categories: consistency regularization [176], proxy-label methods [88], generative models [76], and graph-based methods [64]. However, these SSL methods were primarily proposed for the standard classification tasks where the object instances are classified independently without considering their relations. In this work, we propose a modified strategy for pseudo-label generation and reason about the relations between objects.

8.2.3 Relational Reasoning and Graph Neural Networks

In order to identify the important individuals in a given scene, the model should learn to recognize their relations. Relational reasoning on a group of entities has a wide range of applications such as trajectory forecasting [20], interaction detection [214], object detection [59], dynamics modeling [153], human-robot interaction [16]. In recent years, graph neural networks [9, 178] have attracted significantly increasing research efforts in various fields, which are suitable for tackling relation modeling and extraction [4, 98]. In this work, we

employ a graph neural network to model the relations among objects in the driving scenes, which improves the performance of important object identification.

8.3 Problem Formulation

The important object identification is formulated as a binary classification problem with a semi-supervised learning pipeline. Consider a labeled egocentric driving dataset that contains $|\mathcal{L}|$ labeled frontal image sequences $\mathcal{L} = \{\mathbf{I}^{\mathcal{L}}_{i,t}, i = 1, ..., |\mathcal{L}|, t = -T_h + 1, ..., 0\}$, where for image sequence $\mathbf{I}^{\mathcal{L}}_{i,t}$ there are $N^{\mathcal{L}}_i$ detected objects $\{\mathbf{x}^{\mathcal{L}}_j, j = 1, ..., N^{\mathcal{L}}_i\}$ at the current frame $t = 0$ and the corresponding importance labels $y^{\mathcal{L}}_j$. We set $y^{\mathcal{L}}_j = 1$ for "important" class and $y^{\mathcal{L}}_j = 0$ for "unimportant" class. We also have a set of unlabeled frontal image sequences $\mathcal{U} = \{\mathbf{I}^{\mathcal{U}}_{i,t}, i = 1, ..., |\mathcal{U}|, t = -T_h+1, ..., 0\}$, where for image sequence $\mathbf{I}^{\mathcal{U}}_{i,t}$ there are $N^{\mathcal{U}}_i$ detected objects $\{\mathbf{x}^{\mathcal{U}}_j, j = 1, ..., N^{\mathcal{U}}_i\}$ at the current frame $t = 0$ without importance labels. In this paper, we aim to design an importance classification model $\hat{\mathbf{y}} = f_\theta(\mathbf{x}_j)$ where $\mathbf{x}_j \in \mathbf{I}_{i,0}$ to learn from the augmented training set $\mathcal{L} \cup \mathcal{U}$. In other words, we care about the objects existing at the current frame, and the model f_θ takes in the information of all the detected objects and predicts their importance with respect to the ego vehicle. Besides, we have access to other onboard sensor measurements (e.g., point cloud, velocity, acceleration, yaw angle) and the current ego driving intention I_E (i.e., going forward, turning left, turning right).

8.4 Method

An illustrative model diagram is shown in Figure 8.1 to demonstrate the essential components and procedures of the important object identification approach. The model consists of three modules: (a) a deep feature extraction module which extracts object features from the frontal-view visual observations and the ego vehicle state information; (b) an importance classification module which takes in the extracted feature embeddings and reasons about relations between the objects in the scene and identifies the important ones; and (c) an auxiliary ego behavior prediction module which enhances important object identification by providing additional supervision. We employ a modified semi-supervised learning algorithm to enable the model to learn from a combination of labeled and unlabeled dataset, which achieves better performance compared with standard supervised learning on the labeled dataset. The details of each component are elaborated in the following sections.

8.4.1 Deep Feature Extraction

We assume that the bounding boxes of objects (i.e., traffic participants, traffic lights, stop signs) in the driving videos can be obtained by a detection and tracking system in advance. The depth images can be obtained by projecting the point cloud to the frontal camera view. The segmentation maps are obtained by applying DeepLabv3 [17] to the RGB images. The depth images and segmentation maps have the same size as the original RGB images,

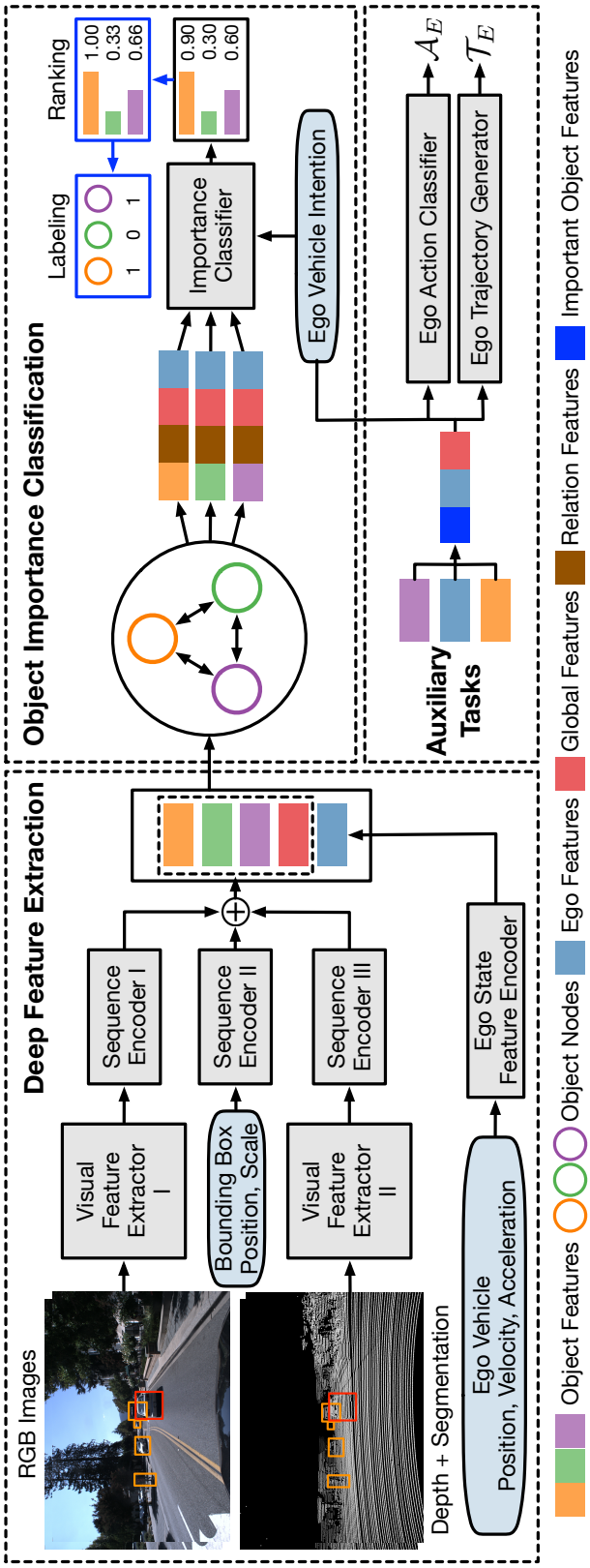


Figure 8.1: The framework of important object identification with a semi-supervised learning pipeline, which consists of three components: 1) deep feature extraction (DFE); 2) object importance classification (OIC); and 3) auxiliary tasks (AT). The procedures with blue arrows are only applied to unlabeled data samples for pseudo-label generation. In the input images, the red/orange boxes indicate the important/unimportant objects, respectively.

which are concatenated along the channel axis. The state information (i.e., position, velocity, acceleration) of the ego vehicle can be obtained from the synchronized 3D LiDAR SLAM and CAN bus.

Visual Features: The visual features $\mathbf{v}_{j,V}$ of a certain object consist of appearance features $\mathbf{v}_{j,A}$ extracted from RGB images, and depth/semantic features $\mathbf{v}_{j,DS}$ extracted from depth images and semantic segmentation maps. The appearance features contain the information of both the appearance and the local context of objects. We adopt the ResNet101 [51] pre-trained on the ImageNet dataset with Feature Pyramid Networks [112] on top as the backbone of Visual Feature Extractor I to obtain the features at all frames, which are fed into Sequence Encoder I to obtain the final features $\mathbf{v}_{j,A}$. We use Visual Feature Extractor II (i.e., ResNet18) trained from scratch to extract the depth/semantic features at all frames, which are fed into Sequence Encoder III to obtain the final features $\mathbf{v}_{j,DS}$. To extract the feature of each object from $\mathbf{v}_{j,A}$ and $\mathbf{v}_{j,DS}$, a ROIAlign [52] pooling layer is added before feeding into sequence encoder. The final visual feature of each object is concatenated by $\mathbf{v}_{j,V} = [\mathbf{v}_{j,A}, \mathbf{v}_{j,DS}]$ along the channel dimension. Similar procedures are applied to extract the global context information from the whole image. The global feature is denoted as $\mathbf{v}_{\text{global}} = [\mathbf{v}_{\text{global},A}, \mathbf{v}_{\text{global},DS}]$.

Bounding Box Features: The location and scale of the object bounding boxes in the frontal-view images can provide additional indications of the size and relative positions of the objects with respect to the ego vehicle, which may influence their importance. We represent this information by $(\frac{x_{j,t}}{W}, \frac{y_{j,t}}{H}, \frac{w_{j,t}}{W}, \frac{h_{j,t}}{H})$ where $x_{j,t}$, $y_{j,t}$, $w_{j,t}$ and $h_{j,t}$ denote the center coordinate, width and height of the bounding box. The stack of this vector along the time axes is fed into Sequence Encoder II to obtain the bounding box feature $\mathbf{v}_{j,B}$.

Ego Vehicle Features: We extract the ego state features $\mathbf{v}_{\text{ego}}$ from a sequence of state information (i.e., position, velocity, acceleration) with the Ego State Feature Encoder.

8.4.2 Importance Classification on Relation Graph

After obtaining the ego state features ($\mathbf{v}_{\text{ego}}$), global features ($\mathbf{v}_{\text{global}}$) and object features ($\mathbf{v}_j, j = 1, ..., N_i$) in the image sequence $\mathbf{I}_i$, we can construct a fully-connected object relation graph where the node attributes are the corresponding object features. In order to model the mutual influence and relations among individuals, we apply a message passing mechanism over the graph, which consists of an edge update ($v \to e$) and a node update ($e \to v$):

$$\mathbf{e}_{jk} = f_e\left([\mathbf{v}_j, \mathbf{v}_k]\right), \quad \bar{\mathbf{v}}_j = f_v\left(\sum\nolimits_{j \neq k} \mathbf{e}_{jk}\right), \tag{8.1}$$

where $\mathbf{e}_{jk}$ is the edge attribute from the sender node k to the receiver node j, $\bar{\mathbf{v}}_j$ is defined as the relation features of node j, and $f_e(\cdot)/f_v(\cdot)$ are the edge/node update functions (i.e., multi-layer perceptrons) which are shared across the whole graph. Note that the edges between the same pair of nodes with opposite directions have different attributes since their mutual influence of object importance may not be symmetric. The message passing procedure is applied multiple times to model higher-order relations in our model.

Since the importance of a certain object with respect to the ego vehicle at the current frame not only depends on its own state but also the global context, object relations and the ego vehicle intention, we generate a comprehensive object feature $\mathbf{o}_j = [\mathbf{v}_j, \bar{\mathbf{v}}_j, \mathbf{v}_{\text{global}}, \mathbf{v}_{\text{ego}}, I_E]$, which is fed into the classifier (i.e., multi-layer perceptron) to obtain its importance score $s_j \in [0, 1]$ (i.e., the probability that the object is important). During training phase, s_j is used to compute loss directly for labeled objects and generate pseudo-labels for unlabeled ones. During testing phase, s_j is used to predict importance by $\arg\max(1 - s_j, s_j)$.

8.4.3 Ranking-Based Pseudo-Label Generation

Pseudo-label generation is generally a crucial step in semi-supervised learning algorithms. In our task, a naive way is to use the learned importance classifier at the last iteration directly to assign pseudo-labels for the objects in the unlabeled data samples by $\arg\max(1 - s_j, s_j)$. However, in most cases only a small subset of objects are important, the naive version of pseudo-label assignment may lead to an data imbalance issue (i.e., assigning an "unimportant" label to all the objects) which degrades the model performance. In order to mitigate this problem, we adopt a modified ranking-based strategy similar to [57], which encourages the model to identify relative importance.

First, we label the objects with a raw importance score s_j larger than a threshold $\alpha_1 = 0.8$ as important objects. Similarly, those with a raw importance score s_j smaller than a threshold $1 - \alpha_1$ are labeled as unimportant objects. If all the objects in a certain case are labeled, there is no further operation and the data sample is appended to the training set. Second, in order to consider the relative importance of the rest objects, we calculate a set of ranking scores $\bar{s}_j$ via dividing the raw scores s_j by their maximum. Then we label the objects with a ranking scores $\bar{s}_j$ greater than a threshold $\alpha_2 = 0.8$ as important ones while the others as unimportant ones. This ranking strategy can not only incorporate relational reasoning in the pseudo-label generation process, but also mitigate data imbalance issue to some extent. We denote the pseudo-label of object $\mathbf{x}_j^{\mathcal{U}}$ as $\tilde{y}_j^{\mathcal{U}}$.

8.4.4 Auxiliary Tasks: Ego Vehicle Behavior Prediction

Since the behavior of ego vehicle can be affected by the existence of important objects, it could in turn enhance the identification of important objects. Therefore, we propose to predict the ego vehicle behavior at two levels as parallel auxiliary tasks to provide additional supervision signals. Since the ego information is always available without the requirement of human annotations, the auxiliary tasks are trained by supervised learning with ground truth for both labeled and unlabeled datasets. First, we predict the high-level action of the ego vehicle $\mathcal{A}_E$ at the current frame with the Ego Action Classifier (EAC), which is formulated as a classification problem. The ego actions can be *stop*, *speed up*, *slow down* or *constant speed*. The ground truth actions are automatically obtained by setting thresholds on the speed and acceleration of the ego vehicle. Second, we also forecast the low-level trajectory of

the ego vehicle $\mathcal{T}_E$ in the future 2 seconds with the Ego Trajectory Generator (ETG), which is formulated as a regression problem.

The EAC and ETG share the same input, which is a feature embedding including the ego intention, ego state features, important object features and global features while discarding the information of unimportant objects. The intuition is that the ego behavior is only influenced by the important objects, which serves as a regularization to encourage the model to identify the correct ones. If some important objects are mistakenly discarded, the predicted ego behavior may change accordingly which results in a discrepancy from the true behavior. The corresponding loss can help improve the importance classifier. However, since hard assignment is not differentiable, we employ the Gumbel-Softmax technique [67] to obtain the gradient approximation for back-propagation. More formally, we denote the weight associated to object $\mathbf{x}_j$ as z_j which can be drawn as

$$z_j = \frac{e^{((\log(s_j)+g_{j,1})/\tau)}}{e^{((\log(s_j)+g_{j,1})/\tau)} + e^{((\log(1-s_j)+g_{j,0})/\tau)}}, \tag{8.2}$$

where $\mathbf{g}_j \in \mathbb{R}^2$ is a vector of i.i.d. samples drawn from a Gumbel$(0, 1)$ distribution and τ is the softmax temperature which controls the smoothness of samples. This distribution converges to one-hot samples from the categorical distribution as $\tau \to 0$. Then the important object features $\mathbf{v}_{\text{imp}}$ is

$$\mathbf{v}_{\text{imp}} = \frac{1}{N_i} \sum_{j=1}^{N_i} z_j \mathbf{v}_j \text{ (training)} \tag{8.3}$$

$$\mathbf{v}_{\text{imp}} = \frac{1}{\hat{N}_i} \sum_{j=1}^{N_i} \arg\max(1 - s_j, s_j)\mathbf{v}_j \text{ (testing)}, \tag{8.4}$$

where $\hat{N}_i$ is the total number of predicted important objects. The combined feature for ego behavior prediction $[\mathbf{v}_{\text{imp}}, \mathbf{v}_{\text{ego}}, \mathbf{v}_{\text{global}}, I_E]$ is used to predict the ego action $\mathcal{A}_E$ and trajectory $\mathcal{T}_E$.

8.4.5 Loss Function and Training

The proposed model can be trained either by supervised learning on the labeled dataset $\mathcal{L}$ or by semi-supervised learning on the combined (labeled and unlabeled) dataset $\mathcal{L} + \mathcal{U}$. The former one serves as an ablative baseline. The detailed loss functions and training procedures of both settings are elaborated below.

Supervised Learning: The loss function L_{SL} consists of two parts: importance classifi-

cation loss L_{imp} and auxiliary loss L_{aux}. More specifically, the loss is computed by

$$L_{\text{SL}}^{\mathcal{L}} = L_{\text{imp}}^{\mathcal{L}} + \lambda L_{\text{aux}}^{\mathcal{L}} = -\frac{1}{|\mathcal{L}|}\sum_{i=1}^{|\mathcal{L}|}\frac{1}{N_i^{\mathcal{L}}}\sum_{j=1}^{N_i^{\mathcal{L}}} l_{\text{CE}}(y_j^{\mathcal{L}}, s_j^{\mathcal{L}}) + \lambda\left(-\frac{1}{|\mathcal{L}|}\sum_{i=1}^{|\mathcal{L}|} l_{\text{CE}}(\mathcal{A}_{Ei}, \hat{\mathcal{A}}_{Ei}) + \beta||\mathcal{T}_{Ei} - \hat{\mathcal{T}}_{Ei}||^2\right), \tag{8.5}$$

where $l_{\text{CE}}(\cdot,\cdot)$ denotes the binary cross-entropy (CE) loss. λ and β are used to adjust the ratio between different losses.

Semi-Supervised Learning: The loss function L_{SSL} consists of two parts: labeled data loss $L^{\mathcal{L}}$ and unlabeled data loss $L^{\mathcal{U}}$. More specifically, the loss is computed by

$$L_{\text{SSL}} = L_{\text{SL}}^{\mathcal{L}} + \gamma L_{\text{SL}}^{\mathcal{U}},\ w_j = \frac{\exp s_j^{\mathcal{U}}}{\sum_{j=1}^{N_i^{\mathcal{U}}}\exp s_j^{\mathcal{U}}}, \varepsilon_i = 1 - \frac{H(\mathbf{w})}{H(\mathbf{m})}, \tag{8.6}$$

$$L_{\text{SL}}^{\mathcal{U}} = \frac{1}{|\mathcal{U}|}\sum_{i=1}^{|\mathcal{U}|}\varepsilon_i\sum_{j=1}^{N_i^{\mathcal{U}}} w_j l_{\text{MSE}}(\tilde{y}_j^{\mathcal{U}}, s_j^{\mathcal{U}}) + \lambda\left(-\frac{1}{|\mathcal{U}|}\sum_{i=1}^{|\mathcal{U}|} l_{\text{CE}}(\mathcal{A}_{Ei}, \hat{\mathcal{A}}_{Ei}) + \beta||\mathcal{T}_{Ei} - \hat{\mathcal{T}}_{Ei}||^2\right), \tag{8.7}$$

where $l_{\text{MSE}}(\cdot,\cdot)$ denotes the mean square error (MSE) loss [10, 134], γ is used to balance the ratio between labeled and unlabeled data loss, $\mathbf{w} = (w_1, ..., w_{N_i^{\mathcal{U}}})$, $\mathbf{m} = (1/N_i^{\mathcal{U}}, ..., 1/N_i^{\mathcal{U}})$ with the same dimension as $\mathbf{w}$, and $H(\cdot)$ is the entropy function. ε_i and w_j are the weights associated to i-th unlabeled data case and the j-th objects in a certain case, respectively. The weight γ is initialized as 0, which implies that the unlabeled dataset is not used at the beginning of training. It increases to a maximum value over a fixed number of epochs with a linear schedule [10, 134] since the model becomes more accurate and confident thus can generate more reliable pseudo-labels as training goes on. The effects of unlabeled dataset increases as the number of epochs becomes larger.

Object Weighting: We apply a weighting mechanism on the loss of each object based on the corresponding predicted importance score $s_j^{\mathcal{U}}$ to strengthen the effect of important ones while weaken that of unimportant ones.

Unlabeled Data Weighting: The basic assumption of our task is that a small subset of objects in the scene are significantly more important to the ego vehicle than the others in most scenarios. In some situations, however, the model may predict vague and similar importance scores for all the objects in unlabeled scenarios. Such cases contribute little to important object identification thus their effects should be weakened. More specifically, we obtain the weight ε_i by leveraging the entropy function $H(\cdot)$ to indicate the similarity between importance scores of different objects. A set of more similar scores will lead to a smaller weight associated with the corresponding training case.

8.5 Experiments

8.5.1 Dataset and Preprocessing

We experimented with a public driving dataset to validate our model: Honda 3D dataset (H3D) [141]. The dataset provides the information collected by a full sensor suite (e.g., camera, LiDAR, radar) mounted on a testing vehicle that navigates in complex driving scenarios with highly interactive traffic participants and the annotated bounding boxes of detected objects in the frontal-view images. Note that in this work we directly used the ground truth bounding boxes provided in the dataset instead of including an upstream detection module. The purpose is to focus on the important object identification while minimizing the influencing factors from the upstream perception task.

Dataset Statistics: We have 7,517 labeled cases for supervised learning and 4,241 unlabeled cases for semi-supervised learning. The ratio of going forward/turning left/turning right is around 4:1:1. The labeled cases are randomly split into training/testing datasets with a ratio of 7:3.

Object Importance Annotation: In order to obtain the binary importance labels of the object in each bounding box, a group of annotators (i.e., experienced drivers) were asked to watch the egocentric driving videos and imagine themselves as the driver of ego vehicle. A segment of ego future trajectory was also provided to the annotators for intention annotation. For each video segment, the important objects and ego intention were annotated at a frequency of 2Hz. We define the important objects as the ones that can potentially influence the ego vehicle behavior given a certain ego intention/goal, which includes dynamic/static traffic participants, traffic lights and stop signs. Generally, the traffic lights and stop signs related to the ego vehicle are labeled as important objects since they provide crucial semantic information of the environment. The static/dynamic traffic participants that are located at the potential future path of the ego vehicles are important. The parked vehicles close to the ego vehicles are generally labeled as important objects since they may start to move and merge into the ego lane.

To demonstrate the consistency and validity of the annotations, we compute the intra-class correlation coefficient (ICC) [160], which is widely used for the assessment of consistency or reproducibility of quantitative measurements made by different observers measuring the same quantity. The ICC for ego intention annotations is 0.941, and the ICC for important object annotations is 0.934. According to the guideline in [21], an ICC score between 0.75 and 1.00 indicates "excellent", which implies the consistency of our annotations.

8.5.2 Evaluation Metrics and Baselines

We adopt the standard metrics for binary classification problems (i.e., accuracy, F1 score) to evaluate the performance of important object identification. To the best of our knowledge, our work is the first to consider the influence of traffic lights and stop signs for this specific task, thus no existing baseline is available to compare against directly. Instead, we conduct

Table 8.1: The Ablative Baseline Model Settings

Model	Ego Intention	Relation Graph	Auxiliary Task	Ranking Strategy	Pseudo Loss Weighting
Ours-S-1	✓			–	–
Ours-S-2		✓		–	–
Ours-S-3	✓	✓		–	–
Ours-S	✓	✓	✓	–	–
Ours-SS-1	✓	✓		✓	
Ours-SS-2	✓	✓			✓
Ours-SS-3	✓	✓		✓	✓
Ours-SS	✓	✓	✓	✓	✓

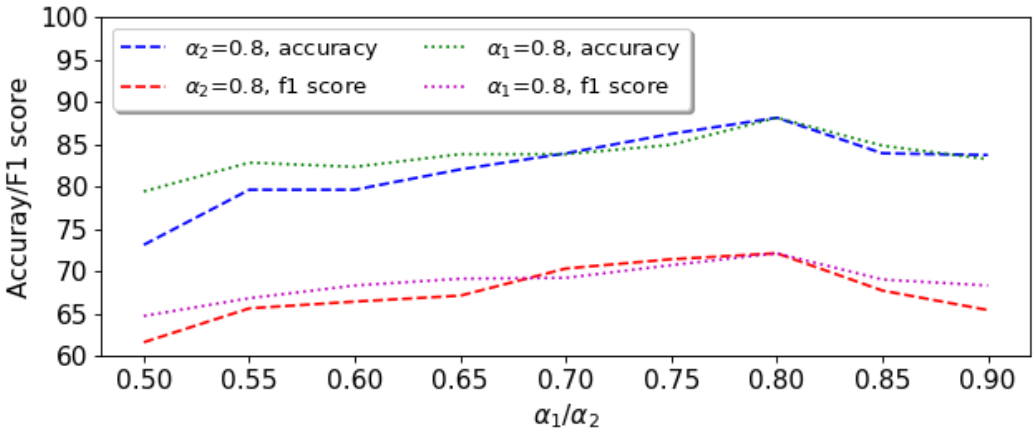


Figure 8.2: The comparison of model performance with different α_1 and α_2.

a detailed ablative study to demonstrate the effectiveness of each model component, and compare our approach with some rule/heuristics based methods.

The different model settings are elaborated in Table 8.1. Ours-S denotes our model trained only on labeled data by standard supervised learning. Ours-S-X denotes a variant of Our-S without certain model components or input information. Ours-SS denotes our model trained both on labeled and unlabeled data by semi-supervised learning. Ours-SS-X denotes a variant of Our-SS without certain model components. The check marks (✓) indicate containing the corresponding information or component. The dashes (–) indicate "not applicable". We also implement four rule-based baseline methods. B-1 selects the object with the largest bounding box. B-2 selects the object closest to the image center. B-3 selects the object closest to ego vehicle. B-4 selects the union of the important objects obtained by above three methods.

Table 8.2: Comparison of Important Object Identification Performance (%)

Ego Intention	Metric	B-1	B-2	B-3	B-4	Ours-S-1	Ours-S-2	Ours-S-3	Ours-S	Ours-SS-1	Ours-SS-2	Ours-SS-3	Ours-SS
Left Turn (LT)	Accuracy	78.1	67.3	76.1	70.2	79.9	76.3	81.0	82.4	81.3	81.3	81.7	87.6
	F1 Score	43.4	15.6	38.2	43.4	64.1	63.7	66.7	68.2	68.4	68.2	68.2	73.2
Straight Pass (SP)	Accuracy	79.1	76.9	75.0	72.7	79.0	75.0	82.2	83.6	84.3	84.7	85.1	89.5
	F1 Score	33.8	27.0	21.0	39.1	59.2	56.0	65.8	68.7	67.7	69.2	73.2	71.8
Right Turn (RT)	Accuracy	70.9	62.1	72.6	63.1	74.7	72.7	75.8	78.0	78.6	78.7	80.7	82.6
	F1 Score	33.8	23.1	44.4	48.0	62.4	59.1	68.6	70.2	72.3	69.2	74.0	72.3
All	Accuracy	77.6	73.2	74.8	70.8	78.5	75.5	81.0	82.6	82.9	83.3	83.9	88.1
	F1 Score	36.8	24.4	28.8	41.8	60.5	56.4	66.6	68.9	68.9	69.1	70.5	72.1

Table 8.3: Effect of Different Proportion of Labeled Training Data (%)

Metric	Ours-$S_{30\%}$	Ours-$S_{60\%}$	Ours-$S_{100\%}$	Ours-$SS_{30\%}$	Ours-$SS_{60\%}$	Ours-$SS_{100\%}$
Accuracy	80.9	82.1	82.6	80.7	84.5	88.1
F1 Score	64.8	66.8	68.9	65.2	68.1	72.1

8.5.3 Implementation Details

For all the experiments, a batch size of 32 was used and the models were trained for up to 100 epochs with early stopping. We used Adam optimizer with an initial learning rate of 0.0001. The models were trained on a single NVIDIA Quadro V100 GPU. We set the thresholds for pseudo-label generation as $\alpha_1 = 0.8, \alpha_2 = 0.8$ and the softmax temperature as $\tau = 0.1$. In the loss functions, we set $\lambda = 0.5$, $\beta = 1.0$, γ is initialized as 0.001 and exponentially increases at each iteration to its maximum value 1.0. Specific details of model components are introduced below:

- **Sequence Encoder I/II/III and Ego State Feature Encoder**: a single-layer LSTM with hidden size = 128 followed by a three-layer MLP with hidden size = 128.
- **Edge/Node Update Functions** $f_v(\cdot)/f_e(\cdot)$: a three-layer MLP with hidden size = 128.
- **Importance Classifier**: a three-layer MLP with hidden size = 256.
- **Ego Action Classifier and Ego Trajectory Generator**: a three-layer MLP with hidden size = 64 followed by a three-layer MLP with hidden size = 256.

8.5.4 Results and Analysis

We conducted a detailed ablation study with quantitative and qualitative results to demonstrate the effectiveness of each model component in the supervised learning setting and effectiveness of the ranking-based pseudo-label generation in the semi-supervised learning setting. We also compared our method with some rule/heuristics-based baselines. The numerical comparisons of performance are provided in Table 8.2. Some testing scenarios are visualized in Figure 8.3, 8.4, 8.5.

Rule-based baselines

We demonstrate the advantages of our method over simple rule-based methods in Table 8.2. We can observe a large drop in performance by using various heuristics to identify the important objects, which implies that fixed rules are not sufficient to capture complex relations and relative importance of objects in the scene.

Ego Intention

In a certain scenario, the important objects may be distinct with different ego driving intentions. We evaluated the model performance on the scenarios with different ego intentions separately to illustrate its effectiveness on H3D dataset. Comparing Ours-S-2 and Ours-S-3 in both tables, the accuracy and F1 score improve with ego intention by a large margin consistently in the scenarios with various ego intentions. More specifically, leveraging the

ego intention improves accuracy by 5.5% and F1 score by 10.2% in average. The intention information can guide the model to pay attention to specific regions thus helpful for important object identification. In order to qualitatively illustrate the influence of ego intention, we visualize the predicted important objects with both ground truth and manipulated intentions in Figure 8.4. The results show that our model can not only correctly identify the important objects with the true intention, but also generate reasonable predictions for manipulated intentions with which the attention may be paid to the objects in different corresponding regions.

Relation Graph

The relation graph is a critical component to aggregate the information of different objects and model their relations/interactions, which helps to discriminate relative importance of multiple objects. In Ours-S-1, the relation graph is skipped and the relation features $\bar{\mathbf{v}}_j$ is removed from $\mathbf{o}_j$, which leads to a decrease in performance compared with Ours-S-3, which implies the effectiveness of graph learning. The information aggregation of different objects boosts the model accuracy by 2.5% and F1 score by 6.1%.

Auxiliary Tasks

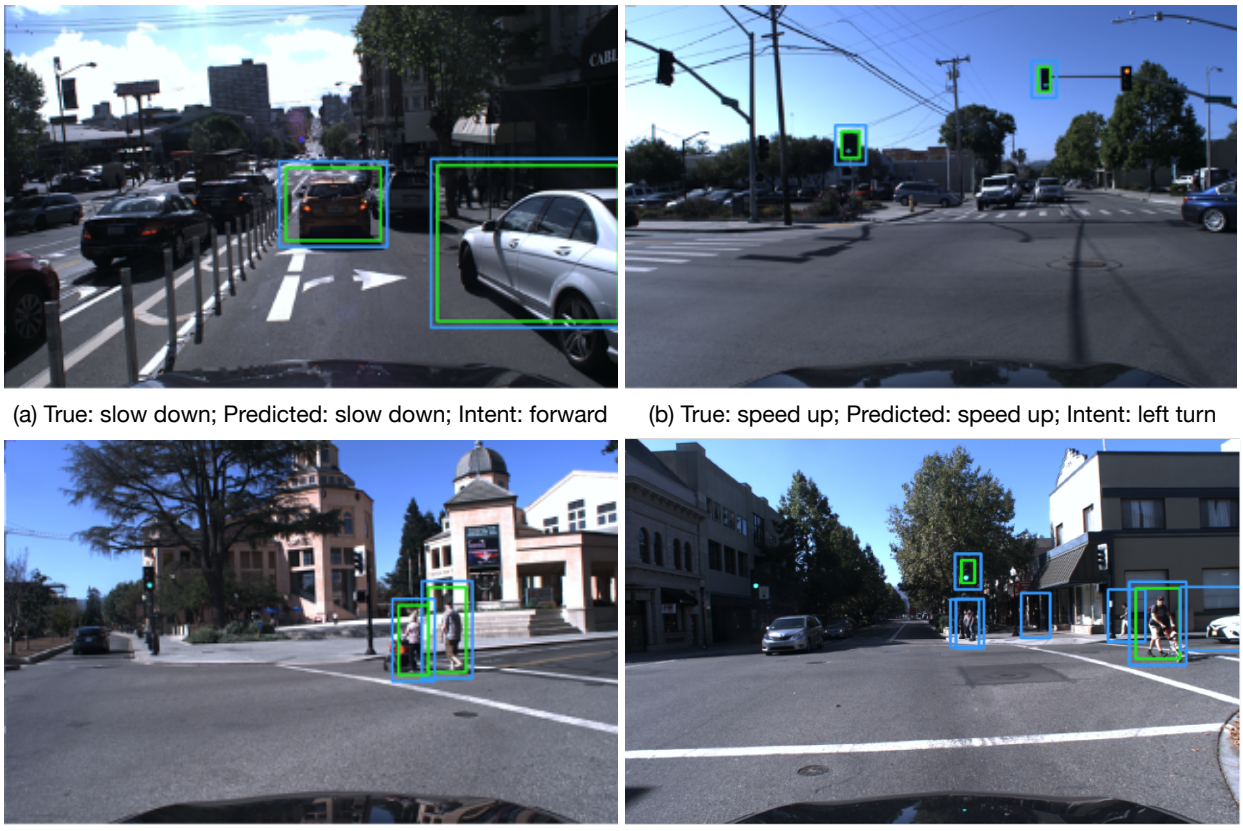
The comparison between Ours-S and Ours-S-3, Ours-SS and Ours-SS-3 can illustrate the effectiveness of the auxiliary tasks. The results show that adding auxiliary tasks during training improves accuracy and F1 score in both supervised learning and semi-supervised learning, which implies that the ego behavior indeed provides useful information and additional supervision. In Fig. 8.3, it shows that our model can predict accurate ego actions at the current frame in various scenarios with specific intentions.

Semi-Supervised Learning

The comparison between Ours-S and Ours-SS shows that the performance of our model improves by a large margin through learning from additional unlabeled data samples with semi-supervised learning. The accuracy and F1 score improve by 5.5% and 3.2%, respectively. The ablative results in Table 8.2 also illustrate the benefits brought by ranking-based strategy for pseudo-label generation and entropy-based loss weighting. In Fig. 8.5, the naive strategy tends to label all the objects in the scene as "unimportant" due to the data imbalance issue while the ranking-based strategy can mitigate this issue and generate reasonable pseudo-labels.

Effect of the Proportion of Labeled Training Data

We further demonstrate the effect of the proportion of labeled training data on the semi-supervised important object identification in Table 8.3. More specifically, we compare the performance of the models trained on 30%, 60% and 100% of the labeled dataset and the whole unlabeled dataset.



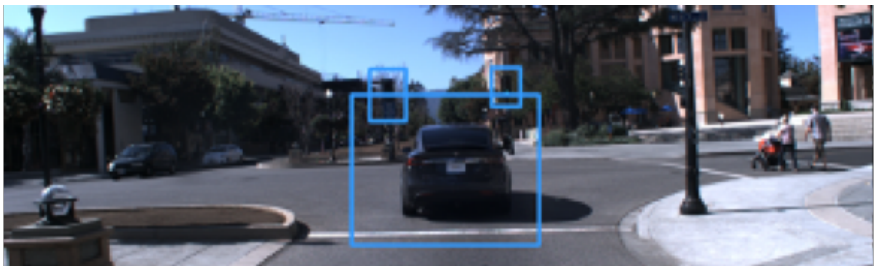
(a) True: slow down; Predicted: slow down; Intent: forward

(b) True: speed up; Predicted: speed up; Intent: left turn

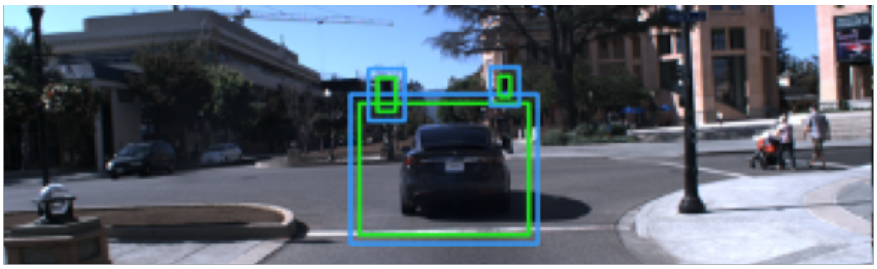
(c) True: speed up; Predicted: speed up; Intent: right turn

(d) True: slow down; Predicted: slow down; Intent: right turn

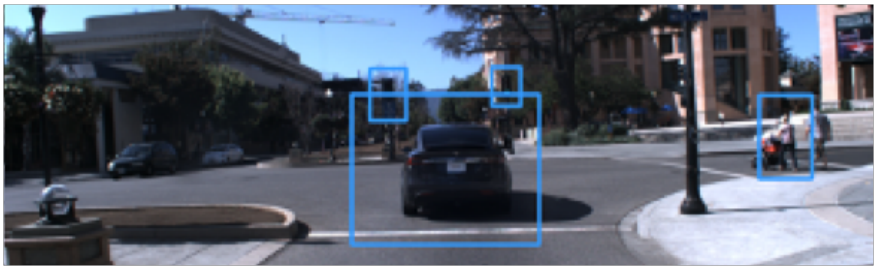
Figure 8.3: Testing scenarios with true ego vehicle intentions (Ours-SS). The green/blue boxes indicate the true/predicted important objects. Our model can identify the correct important objects and predict accurate ego actions. In (a), the model recognizes adjacent objects as important due to potential interactions. In (b), since the light for "turning left" is green, the vehicles in the opposite direction are not important thus the ego vehicle can speed up. In (c), the model identifies the important objects in the intended path of ego vehicle. In (d), since the ego intention is "turning right", the model predicts the objects in the right part of the image as important ones and still predicts the ego action correctly and keeps the ego vehicle safe.



(a) True intent: forward; Predicted: speed up; True: speed up



(b) Manipulated intent: left turn; Predicted: speed up



(c) Manipulated intent: right turn; Predicted: speed up

Figure 8.4: Testing scenarios with manipulated ego vehicle intentions (Ours-SS).

The results show that training on more labeled data can boost the overall model performance, which can also enable the semi-supervised models to generate more reasonable and accurate pseudo-labels for unlabeled scenarios and further enhance performance. It is observed that Ours-$S_{30\%}$ achieves comparable performance to Ours-$SS_{30\%}$. A potential reason is that with only 30% of the labeled dataset the model could not learn well enough to generate sufficiently accurate pseudo-labels, which weakens the advantages of learning from unlabeled dataset or even brings negative effects. As the amount of labeled data increases, the model performance improves by a larger margin with semi-supervised learning.

(a) True intent: forward (b) True intent: forward (c) True intent: forward

Figure 8.5: Visualization of generated "important" pseudo-labels (yellow boxes) with the ranking-based strategy (Ours-SS). The naive strategy labels all the objects as "unimportant" in these scenarios.

Effect of Thresholds α_1 and α_2

In the pseudo-label generation process, the thresholds α_1 and α_2 are determined empirically. The comparison of model performance with different α_1/α_2 combinations is shown in Fig. 8.2. In particular, $\alpha_1 = 0.5$ performs the worst when α_2 is fixed at 0.8, which implies that the naive threshold (i.e., 0.5) is not good for generating pseudo labels since the prediction confidence may not be high enough which results in noisy pseudo labels.

8.6 Chapter Summary

In this paper, we present a novel approach for important object identification in autonomous driving. We propose to incorporate human knowledge as direct supervision to recognize important objects. Since it is expensive to obtain human annotations for a large amount of data, we propose a semi-supervised learning pipeline to enable the model to learn from both labeled and unlabeled datasets. In order to consider the relations between objects when inferring their importance, we employ a graph neural network to extract relation features. Moreover, we propose to leverage the action/trajectory information of the ego vehicle to provide additional supervision signals as auxiliary tasks to improve the model performance. The model is validated on the H3D dataset with egocentric videos, 2D bounding box annotations and point clouds. The quantitative and qualitative results demonstrate the effectiveness of each model component, the ranking-based pseudo-label generation strategy and loss weighting strategies for semi-supervised learning.

Chapter 9

Final Words

In this dissertation, we investigated a wide range of generic approaches to model the relations and interactions between autonomous agents or entities in a multi-agent system, which can be applied to a variety of domains (e.g., physical systems, human crowds, traffic participants, robot swarms). We also explored the solutions to typical downstream tasks (e.g., prediction, tracking, importance estimation) under a multi-agent setting with relational reasoning.

In Part I, we focused on the multi-agent prediction and tracking problems. In Chapter 2, we introduced a hierarchical time-series prediction model for situation and behavior recognition based on probabilistic graphical models, which can be applied to the scenarios with a single autonomous agent or under a fixed multi-agent setting. This prediction approach has a hierarchical structure, which is particularly designed for a dynamic process consisting of multiple clearly defined sub-stages. When it is applied to a general dynamic system, the number of sub-stages will become a hyperparameter to tune to achieve the best performance. In Chapter 3, we employed deep generative modeling techniques to learn the data distribution, which can generate more diverse and realistic prediction hypotheses. In Chapter 4, we further leveraged a graph representation to capture spatio-temporal interaction patterns, which is a natural way to represent multiple agents in the scene and their relations. The graph topology is determined by human-defined heuristics (e.g., distanced-based), which is a straightforward way to impose inductive biases to the relational structure. However, the heuristics may not be applicable in all situations, which may lead to undesired outcomes. For example, a distant coming vehicle in the opposite direction may prevent the car from turning left in the intersection, but it might be ignored with a small distance threshold when modeling its influence. Therefore, in Chapter 5 we proposed to learn a latent relational graph structure from observation data instead of pre-defining the graph topology, which can evolve over time to enable dynamic relational reasoning.

An accurate prediction model plays a significant role in multi-target tracking frameworks, especially in highly dynamic and interactive scenarios with heterogeneous agents. In Chapter 6, we discussed a unified tracking and prediction framework based on a modified sequential Monte Carlo method, which can adopt any of the above prediction models as the implicit proposal distribution.

In Part II, we addressed another related downstream problem (i.e., importance estimation) of relational reasoning under a multi-agent setting. In Chapter 7, we discussed a hybrid attention inference network to recognize relative importance of objects in the scene based on observation data, which enables dynamic key information selection. Compared with the approach proposed in Chapter 5, this method emphasizes more on estimating the importance of a certain entity with respect to another one instead of recognizing their specific relations, which is more applicable when the interaction types are hard to define. In Chapter 8, we further investigated how to incorporate human knowledge from human annotations with a self-supervised learning pipeline, which enables the model to learn from both labeled data and unlimited, unlabeled data.

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

- To demonstrate interpretability of learning-based models

  In recent years, many learning-based methods (especially deep learning methods) have been proposed to solve multi-agent motion prediction and interaction-aware decision making problems for autonomous systems, which can achieve promising model performance in terms of various aspects. However, it is not trivial to understand and interpret the end-to-end models. In this dissertation, we tried to investigate and improve the model interpretability by visualizing or analyzing latent/intermediate representations. For example, the attention weights can be visualized to illustrate the relative importance of surrounding agents or feature elements, which enables a post-hoc interpretation of the model. In future work, more theoretical studies and analysis can be further conducted.

- To improve data efficiency of learning-based models

  Learning-based models (especially supervised deep learning methods) usually demand a large amount of training data for supervision. However, in many cases the data is expensive to collect and requires abundant human annotations. Therefore, in this dissertation we investigated potential solutions to leverage unlimited unlabeled data through the semi-supervised learning techniques. In future work, it is promising to explore approaches that have less reliance on human-annotated data and the ability to draw conclusions even with limited experience. Common sense reasoning or human prior knowledge could be incorporated for behavior modeling and decision making.

- To enhance generalizability and enable continual learning

  Many learning-based approaches achieve promising performance when the testing data are drawn from an underlying distribution that is the same as or very similar to training data distribution. However, there are often unseen or out-of-distribution testing cases not covered in the training dataset, which may result in a significant decrease in testing performance. Meanwhile, it is also possible that new data is collected in a streaming way. In future work, it is desired to investigate how data-driven methods can generalize better under unseen settings and learn from new data in a continuous way.

- To improve computation efficiency and scalability

  There may be a large number of autonomous agents communicating and interacting with each other in real-world multi-agent systems. For example, mobile robots share the space with human crowds; robot swarms collaborate together and communicate with each other to get a task done. Therefore, it is necessary to improve the computation efficiency and scalability of method to tackle large-scale multi-agent systems. In this dissertation, we have explored effective and efficient graph representations to represent multi-agent scenarios, in which node and edge functions are shared across the whole graph. In future work, we will how to further improve the computation efficiency and enhance scalability in a more systematic way.

- To integrate perception, prediction and decision making in a unified framework

  Perception, prediction and decision making are three major tasks for autonomous navigation. Effective perception of the context environment and accurate future prediction of the interactive agents in the scene serve as an important prerequisite for safe and high-quality decision making of autonomous agents. In this dissertation, we have explored the perception aspect with an emphasis on semantic scene understanding and important object identification. We also investigated a wide range of intention/motion prediction methods, which can achieve state-of-the-art performance in terms of prediction accuracy. In future work, we will explore decision making algorithms in a multi-agent setting and design a unified framework that incorporates the perception, prediction and decision making modules.

With the development of technology, it may no longer be a fantasy to have intelligent and autonomous robots that think, behave and interact with the world in the way that human beings do, so that they can better serve, assist and collaborate with people in their daily lives across work, home and leisure.

Bibliography

[1] Daniel Aarno and Danica Kragic. "Evaluation of Layered HMM for Motion Intention Recognition". In: *IEEE International Conference on Advanced Robotics*. 2007.

[2] Ekrem Aksoy, Ahmet Yazıcı, and Mahmut Kasap. "See, Attend and Brake: An Attention-based Saliency Map Prediction Model for End-to-End Driving". In: *arXiv preprint arXiv:2002.11020* (2020).

[3] 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.

[4] Ferran Alet et al. "Neural Relational Inference with Fast Modular Meta-learning". In: *Advances in Neural Information Processing Systems*. 2019, pp. 11804–11815.

[5] Peter Anderson et al. "Bottom-up and top-down attention for image captioning and visual question answering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6077–6086.

[6] Georges Aoude et al. "Mobile agent trajectory prediction using Bayesian nonparametric reachability trees". In: *Infotech@ Aerospace 2011*. 2011, p. 1512.

[7] M Sanjeev Arulampalam et al. "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking". In: *IEEE Trans. on signal processing* 50.2 (2002), pp. 174–188.

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

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

[10] David Berthelot et al. "MixMatch: A Holistic Approach to Semi-Supervised Learning". In: *Advances in Neural Information Processing Systems* 32 (2019).

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

[12] Defu Cao et al. "Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting". In: *Proceedings of the Neural Information Processing Systems (NeurIPS)*. 2020.

[13] Defu Cao et al. "Spectral Temporal Graph Neural Network for Trajectory Prediction". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021.

[14] Yuning Chai et al. "MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction". In: *Conference on Robot Learning*. 2020, pp. 86–99.

[15] Changan Chen et al. "Crowd-Robot Interaction: Crowd-aware Robot Navigation with Attention-based Deep Reinforcement Learning". In: *arXiv preprint arXiv:1809.08835* (2018).

[16] Changan Chen et al. "Relational graph learning for crowd navigation". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 10007–10013.

[17] Liang-Chieh Chen et al. "Rethinking atrous convolution for semantic image segmentation". In: *arXiv preprint arXiv:1706.05587* (2017).

[18] Sangwoo Cho et al. "Self-Attention Network for Skeleton-based Human Action Recognition". In: *The IEEE Winter Conference on Applications of Computer Vision*. 2020, pp. 635–644.

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

[20] 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, pp. 244–253.

[21] Domenic V Cicchetti. "Guidelines, criteria, and rules of thumb for evaluating normed and standardized assessment instruments in psychology." In: *Psychological assessment* 6.4 (1994), p. 284.

[22] Benjamin Coifman and Lizhe Li. "A critical evaluation of the Next Generation Simulation (NGSIM) vehicle trajectory dataset". In: *Transport. Res. Part B: Methodo.* 105 (2017), pp. 362–377.

[23] James Colyar and John Halkias. "US highway 101 dataset". In: *Federal Highway Administration (FHWA) Technique Report. FHWA-HRT-07-030*. 2007.

[24] Dan Crisan and Arnaud Doucet. "Convergence of sequential Monte Carlo methods". In: *Signal Processing Group, Department of Engineering, University of Cambridge, Technical Report CUEDIF-INFENGrrR38* 1 (2000).

[25] Luca Cultrera et al. "Explaining autonomous driving by learning end-to-end visual attention". In: *CVPR Workshop*. 2020, pp. 340–341.

[26] Nachiket Deo, Akshay Rangesh, and Mohan M Trivedi. "How would surround vehicles move? a unified framework for maneuver classification and motion prediction". In: *IEEE Transactions on Intelligent Vehicles* 3.2 (2018), pp. 129–140.

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

[28] Julie Dequaire et al. "Deep tracking in the wild: End-to-end tracking using recurrent neural networks". In: *The International Journal of Robotics Research* 37.4-5 (2018), pp. 492–512.

[29] Alberto Diaz-Alvarez et al. "Modelling the human lane-change execution behaviour through Multilayer Perceptrons and Convolutional Neural Networks". In: *Transport. Res. Part F: Traffic Psycho. and Behav.* 56 (2018), pp. 134–148.

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

[31] Randal Douc and Olivier Cappe. "Comparison of resampling schemes for particle filtering". In: *Image and Signal Processing and Analysis., Proc. 4th Int. Symp. on*. IEEE. 2005, pp. 64–69.

[32] Chenyou Fan et al. "Multi-horizon time series forecasting with temporal attention learning". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2527–2535.

[33] Tharindu Fernando et al. "Soft+ hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection". In: *Neural networks* 108 (2018), pp. 466–478.

[34] Katerina Fragkiadaki et al. "Recurrent network models for human dynamics". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 4346–4354.

[35] Luca Franceschi et al. "Learning Discrete Structures for Graph Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 1972–1982.

[36] Haiming Gang et al. "LOKI: Long Term and Key Intentions for Trajectory Prediction". In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2021.

[37] Hongyang Gao and Shuiwang Ji. "Graph representation learning via hard and channel-wise attention networks". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 741–749.

[38] 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.

[39] Mingfei Gao, Ashish Tawari, and Sujitha Martin. "Goal-oriented object importance estimation in on-road driving videos". In: *ICRA*. IEEE. 2019, pp. 5509–5515.

[40] Hossein Gholamalinezhad and Hossein Khosravi. "Pooling Methods in Deep Neural Networks, a Review". In: *arXiv preprint arXiv:2009.07485* (2020).

[41] Tobias Gindele, Sebastian Brechtel, and Rudiger Dillmann. "Learning driver behavior models from traffic observations for decision making and planning". In: *IEEE Intelligent Transportation Systems Magazine* 7.1 (2015), pp. 69–79.

[42] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[43] Anand Gopalakrishnan et al. "A neural temporal model for human motion prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12116–12125.

[44] Colin Graber and Alexander G Schwing. "Dynamic Neural Relational Inference". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8513–8522.

[45] 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.

[46] Fredrik Gustafsson. "Particle filter theory and practice with positioning applications". In: *IEEE Aerosp. and Elec. Syst. Magaz.* 25.7 (2010), pp. 53–82.

[47] Nicholas Guttenberg et al. "Permutation-equivariant neural networks applied to dynamics prediction". In: *arXiv preprint arXiv:1612.04530* (2016).

[48] Uwe D Hanebeck, Kai Briechle, and Andreas Rauh. "Progressive Bayes: a new framework for nonlinear state estimation". In: *Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2003*. Vol. 5099. International Society for Optics and Photonics. 2003, pp. 256–268.

[49] Irtiza Hasan et al. "MX-LSTM: mixing tracklets and vislets to jointly forecast trajectories and head poses". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6067–6076.

[50] Hado van Hasselt, Arthur Guez, and David Silver. "Deep reinforcement learning with double Q-Learning". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press. 2016, pp. 2094–2100.

[51] 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.

[52] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[53] Dirk Helbing and Peter Molnar. "Social force model for pedestrian dynamics". In: *Physical review E* 51.5 (1995), p. 4282.

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

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

[56] 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.

[57] Fa-Ting Hong, Wei-Hong Li, and Wei-Shi Zheng. "Learning to Detect Important People in Unlabelled Images for Semi-Supervised Important People Detection". In: *CVPR*. 2020, pp. 4146–4154.

[58] Yedid Hoshen. "Vain: Attentional multi-agent predictive modeling". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2701–2711.

[59] Han Hu et al. "Relation networks for object detection". In: *CVPR*. 2018, pp. 3588–3597.

[60] Yeping Hu, Wei Zhan, and Masayoshi Tomizuka. "Probabilistic Prediction of Vehicle Semantic Intention and Motion". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 307–313.

[61] Xin Huang et al. "Uncertainty-aware driver trajectory prediction at urban intersections". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 9718–9724.

[62] Yingfan Huang et al. "Stgat: Modeling spatial-temporal interactions for human trajectory prediction". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 6272–6281.

[63] 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 (2014), pp. 1325–1339.

[64] Ahmet Iscen et al. "Label propagation for deep semi-supervised learning". In: *CVPR*. 2019, pp. 5070–5079.

[65] Boris Ivanovic and Marco Pavone. "The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 2375–2384.

[66] 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.

[67] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax". In: *arXiv preprint arXiv:1611.01144* (2016).

[68] Zhong Ji et al. "Deep attentive video summarization with distribution consistency learning". In: *IEEE Trans. on Neural Networks and Learning Systems* (2020).

[69] Longlong Jing et al. "Videossl: Semi-supervised learning for video classification". In: *WACV*. 2021, pp. 1110–1119.

[70] Alexander Kamann et al. "Object tracking based on an extended Kalman filter in high dynamic driving situations". In: *Intell. Transport. Syst., 20th Int. Conf. on*. IEEE. 2017, pp. 1–6.

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

[72] Seyed Mehran Kazemi et al. "Relational representation learning for dynamic (knowledge) graphs: A survey". In: *arXiv preprint arXiv:1905.11485* (2019).

[73] ByeoungDo Kim et al. "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network". In: *arXiv preprint arXiv:1704.07049* (2017).

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

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

[76] Diederik P Kingma et al. "Semi-supervised learning with deep generative models". In: *NIPS*. 2014, pp. 3581–3589.

[77] Thomas Kipf et al. "Neural Relational Inference for Interacting Systems". In: *International Conference on Machine Learning*. 2018, pp. 2688–2697.

[78] Stefan Klingelschmitt and Julian Eggert. "Using context information and probabilistic classification for making extended long-term trajectory predictions". In: *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*. IEEE. 2015, pp. 705–711.

[79] Jason Kong et al. "Kinematic and dynamic vehicle models for autonomous driving control design". In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2015, pp. 1094–1099.

[80] Vineet Kosaraju et al. "Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks". In: *Advances in Neural Information Processing Systems*. 2019, pp. 137–146.

[81] Adam Kosiorek, Alex Bewley, and Ingmar Posner. "Hierarchical attentive recurrent tracking". In: *Advances in Neural Information Processing Systems*. 2017, pp. 3053–3061.

[82] Jannik Kossen et al. "Structured Object-Aware Physics Prediction for Video Modeling and Planning". In: *International Conference on Learning Representations*. 2020.

[83] Hans R Kunsch et al. "Recursive Monte Carlo filters: algorithms and theoretical analysis". In: *The Annals of Statistics* 33.5 (2005), pp. 1983–2021.

[84] Lixin Lang et al. "Bayesian estimation via sequential Monte Carlo sampling constrained dynamic systems". In: *Automatica* 43.9 (2007), pp. 1615–1622.

[85] Christian Laugier et al. "Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety". In: *IEEE Intell. Transport. Syst. Magazine* 3.4 (2011), pp. 4–19.

[86] Andreas Lawitzky et al. "Interactive scene prediction for automotive applications". In: *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE. 2013, pp. 1028–1033.

[87] Laura Leal-Taixé et al. "Learning an image-based motion context for multiple people tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3542–3549.

[88] Dong-Hyun Lee et al. "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks". In: *Workshop on challenges in representation learning, ICML*. Vol. 3. 2. 2013.

[89] 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.

[90] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. "A survey on motion prediction and risk assessment for intelligent vehicles". In: *ROBOMECH journal* 1.1 (2014), p. 1.

[91] C. Li, S. H. Chan, and Y. -T. Chen. "Who Make Drivers Stop? Towards Driver-centric Risk Assessment: Risk Object Identification via Causal Inference". In: *IROS*. 2020, pp. 10711–10718.

[92] Chen Li et al. "Convolutional sequence to sequence model for human dynamics". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5226–5234.

[93] 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.

[94] Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. "Interaction-aware Multi-agent Tracking and Probabilistic Behavior Prediction via Adversarial Learning". In: *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2019.

[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, Wei Zhan, and Masayoshi Tomizuka. "Generic Vehicle Tracking Framework Capable of Handling Occlusions Based on Modified Mixture Particle Filter". In: *Proceedings of 2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 936–942.

[97] Jiachen Li et al. "Coordination and Trajectory Prediction for Vehicle Interactions via Bayesian Generative Modeling". In: *in 2019 Intelligent Vehicles Symposium (IV)*. IEEE. 2019.

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

[99] Jiachen Li et al. "Generic probabilistic interactive situation recognition and prediction: From virtual to real". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 3218–3224.

[100] Jiachen Li et al. "Generic Tracking and Probabilistic Prediction Framework and Its Application in Autonomous Driving". In: *IEEE Transactions on Intelligent Transportation Systems* 21.9 (2020), pp. 3634–3649.

[101] Jiachen Li et al. "Important Object Identification with Semi-Supervised Learning for Autonomous Driving". In: *IEEE Robotics and Automation Letters (RA-L)* (under review).

[102] Jiachen Li et al. "RAIN: Reinforced Hybrid Attention Inference Network for Motion Forecasting". In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2021.

[103] Jiachen Li et al. "Social-WaGDAT: Interaction-aware Trajectory Prediction via Wasserstein Graph Double-Attention Network". In: *arXiv preprint arXiv:2002.06241* (2020).

[104] Jiachen Li et al. "Spatio-Temporal Graph Dual-Attention Network for Multi-Agent Prediction and Tracking". In: *IEEE Transactions on Intelligent Transportation Systems* (2021). DOI: 10.1109/TITS.2021.3094821.

[105] 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.

[106] Shengbo Eben Li et al. "Kalman filter-based tracking of moving objects using linear ultrasonic sensor array for road vehicles". In: *Mechanical Systems and Signal Processing* 98 (2018), pp. 173–189.

[107] Wei-Hong Li, Fa-Ting Hong, and Wei-Shi Zheng. "Learning to learn relation for important people detection in still images". In: *CVPR*. 2019, pp. 5003–5011.

[108] Yunzhu Li et al. "Causal Discovery in Physical Systems from Videos". In: *Advances in Neural Information Processing Systems* 33 (2020).

[109] Zirui Li et al. "A Hierarchical Framework for Interactive Behaviour Prediction of Heterogeneous Traffic Participants based on Graph Neural Network". In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[110] Zirui Li et al. "Development and evaluation of two learning-based personalized driver models for pure pursuit path-tracking behaviors". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 79–84.

[111] Junwei Liang et al. "Peeking into the future: Predicting future person activities and locations in videos". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5725–5734.

[112] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: *CVPR*. 2017.

[113] Zhouhan Lin et al. "A structured self-attentive sentence embedding". In: *arXiv preprint arXiv:1703.03130* (2017).

[114] Nian Liu, Junwei Han, and Ming-Hsuan Yang. "Picanet: Learning pixel-wise contextual attention for saliency detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3089–3098.

[115] Wei Liu, Hongwen He, and Fengchun Sun. "Vehicle state estimation based on Minimum Model Error criterion combining with Extended Kalman Filter". In: *J. of the Fran. Instit.* 353.4 (2016), pp. 834–856.

[116] Zhenguang Liu et al. "Towards natural and accurate future motion prediction of humans and animals". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10004–10012.

[117] Chao Lu et al. "Transfer Learning for Driver Model Adaptation in Lane-Changing Scenarios Using Manifold Alignment". In: *IEEE Transactions on Intelligent Transportation Systems* (2019).

[118] Hengbo Ma et al. "Continual Multi-agent Interaction Behavior Prediction with Conditional Generative Memory". In: *IEEE Robotics and Automation Letters* (2021).

[119] Hengbo Ma et al. "Multi-agent Driving Behavior Prediction across Different Scenarios with Self-supervised Domain Knowledge". In: *2021 International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2021.

[120] 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.

[121] 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.

[122] 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.

[123] C Maddison, A Mnih, and Y Teh. "The concrete distribution: A continuous relaxation of discrete random variables". In: *International Conference on Learning Representations*. 2017.

[124] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. "The concrete distribution: A continuous relaxation of discrete random variables". In: *International Conference on Learning Representations*. 2017.

[125] Srikanth Malla, Behzad Dariush, and Chiho Choi. "TITAN: Future Forecast using Action Priors". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11186–11196.

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

[127] Sujitha Martin et al. "Dynamics of driver's gaze: Explorations in behavior modeling and maneuver prediction". In: *IEEE Trans. on Intell. Vehicles* 3.2 (2018), pp. 141–150.

[128] 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.

[129] Dhanvin Mehta, Gonzalo Ferrer, and Edwin Olson. "Autonomous navigation in dynamic social environments using multi-policy decision making". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 1190–1197.

[130] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[131] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[132] Abduallah Mohamed et al. "Social-STGCNN: A Social Spatio-Temporal Graph Convolutional Neural Network for Human Trajectory Prediction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14424–14432.

[133] XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. "Estimating divergence functionals and the likelihood ratio by convex risk minimization". In: *IEEE Transactions on Information Theory* 56.11 (2010), pp. 5847–5861.

[134] Avital Oliver et al. "Realistic Evaluation of Deep Semi-Supervised Learning Algorithms". In: *NeurIPS* 31 (2018), pp. 3235–3246.

[135] Nuria Oliver, Eric Horvitz, and Ashutosh Garg. "Layered representations for human activity recognition". In: *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*. IEEE Computer Society. 2002, p. 3.

[136] Yassine Ouali, Céline Hudelot, and Myriam Tami. "An Overview of Deep Semi-Supervised Learning". In: *arXiv preprint arXiv:2006.05278* (2020).

[137] Andrea Palazzi et al. "Predicting the Driver's Focus of Attention: the DR (eye) VE Project". In: *IEEE Trans. on Patt. Analy. and Mach. Intell.* 41.7 (2018), pp. 1720–1733.

[138] Aldo Pareja et al. "EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs." In: *AAAI*. 2020, pp. 5363–5370.

[139] Ankur Parikh et al. "A Decomposable Attention Model for Natural Language Inference". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 2249–2255.

[140] Seong Hyeon Park et al. "Diverse and admissible trajectory forecasting through multimodal context understanding". In: *European Conference on Computer Vision*. Springer. 2020, pp. 282–298.

[141] Abhishek Patil et al. "The H3D Dataset for Full-Surround 3D Multi-Object Detection and Tracking in Crowded Urban Scenes". In: *International Conference on Robotics and Automation*. 2019.

[142] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. "Improving data association by joint modeling of pedestrian trajectories and groupings". In: *European conference on computer vision*. Springer. 2010, pp. 452–465.

[143] Tung Phan-Minh et al. "Covernet: Multimodal behavior prediction using trajectory sets". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14074–14083.

[144] Yao Qin et al. "A dual-stage attention-based recurrent neural network for time series prediction". In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 2017, pp. 2627–2633.

[145] 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.

[146] 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.

[147] Yara Rizk, Mariette Awad, and Edward W Tunstel. "Decision making in multiagent systems: A survey". In: *IEEE Transactions on Cognitive and Developmental Systems* 10.3 (2018), pp. 514–529.

[148] Alexandre Robicquet et al. "Learning social etiquette: Human trajectory understanding in crowded scenes". In: *European conference on computer vision*. Springer. 2016, pp. 549–565.

[149] Andrey Rudenko et al. "Human motion trajectory prediction: A survey". In: *The International Journal of Robotics Research* 39.8 (2020), pp. 895–935.

[150] Amir Sadeghian et al. "Car-net: Clairvoyant attentive recurrent network". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 151–167.

[151] 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, pp. 1349–1358.

[152] Tim Salzmann et al. "Trajectron++: Multi-Agent Generative Trajectory Forecasting With Heterogeneous Data for Control". In: *Proceedings of Europe Conference on Computer Vision (ECCV)*. 2020.

[153] Alvaro Sanchez-Gonzalez et al. "Learning to simulate complex physics with graph networks". In: *Proceedings of the International Conference on Machine Learning*. 2020, pp. 11648–11657.

[154] Adam Santoro et al. "A simple neural network module for relational reasoning". In: *Advances in neural information processing systems*. 2017, pp. 4967–4976.

[155] Jacob Scharcanski et al. "A particle-filtering approach for vehicular tracking adaptive to occlusions". In: *IEEE Transactions on Vehicular Technology* 60.2 (2010), pp. 381–389.

[156] Robin Schubert, Eric Richter, and Gerd Wanielik. "Comparison and evaluation of advanced motion models for vehicle tracking". In: *Inform. Fusion, 11th Int. Conf. on*. IEEE. 2008, pp. 1–6.

[157] Mao Shan, Stewart Worrall, and Eduardo Mario Nebot. "Probabilistic Long-Term Vehicle Motion Prediction and Tracking in Large Environments." In: *IEEE Trans. Intell. Transport. Syst.* 14.2 (2013), pp. 539–552.

[158] Xinguang Shao, Biao Huang, and Jong Min Lee. "Constrained Bayesian state estimation–A comparative study and a new particle filter based approach". In: *J. of Process Control* 20.2 (2010), pp. 143–157.

[159] Tao Shen et al. "Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2018, pp. 4345–4352.

[160] Patrick E Shrout and Joseph L Fleiss. "Intraclass correlations: uses in assessing rater reliability." In: *Psychological bulletin* 86.2 (1979), p. 420.

[161] Chenyang Si et al. "An attention enhanced graph convolutional lstm network for skeleton-based action recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pp. 1227–1236.

[162] Tom Silver et al. "Planning with learned object importance in large problem instances using graph neural networks". In: *arXiv preprint arXiv:2009.05613* (2020).

[163] Dan Simon. "Kalman filtering with state constraints: a survey of linear and nonlinear algorithms". In: *IET Control Theory & Applications* 4.8 (2010), pp. 1303–1318.

[164] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[165] Sayanan Sivaraman and Mohan Manubhai Trivedi. "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis". In: *IEEE Trans. Intell. Transport. Syst.* 14.4 (2013), pp. 1773–1795.

[166] Sina Solaimanpour and Prashant Doshi. "A layered HMM for predicting motion of a leader in multi-robot settings". In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 788–793.

[167] Sijie Song et al. "Spatio-temporal attention-based LSTM networks for 3D action recognition and detection". In: *IEEE Transactions on image processing* 27.7 (2018), pp. 3459–3471.

[168] Sjoerd van Steenkiste et al. "Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions". In: *International Conference on Learning Representations*. 2018.

[169] Ondvrej Straka, Jindvrich Dunik, and Miroslav Simandl. "Truncation nonlinear filters for state estimation with nonlinear inequality constraints". In: *Automatica* 48.2 (2012), pp. 273–286.

[170] Shan Su et al. "Potential Field: Interpretable and Unified Representation for Trajectory Prediction". In: *arXiv preprint arXiv:1911.07414* (2019).

[171] Guolei Sun et al. "Mining cross-image semantics for weakly supervised semantic segmentation". In: *ECCV*. Springer. 2020, pp. 347–365.

[172] Liting Sun, Wei Zhan, and Masayoshi Tomizuka. "Probabilistic prediction of interactive driving behavior via hierarchical inverse reinforcement learning". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 2111–2117.

[173] Charlie Tang and Russ R Salakhutdinov. "Multiple futures prediction". In: *Advances in Neural Information Processing Systems*. 2019, pp. 15424–15434.

[174] Yongyi Tang et al. "Long-term human motion prediction by modeling motion context and enhancing motion dynamic". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2018, pp. 935–941.

[175] Georg Tanzmeister and Dirk Wollherr. "Evidential grid-based tracking and mapping". In: *IEEE Trans. Intell. Transport. Syst.* 18.6 (2017), pp. 1454–1467.

[176] Antti Tarvainen and Harri Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results". In: *NIPS*. 2017, pp. 1195–1204.

[177] Ilya Tolstikhin et al. "Wasserstein auto-encoders". In: *arXiv preprint arXiv:1711.01558* (2017).

[178] Petar Veličković et al. "Graph Attention Networks". In: *International Conference on Learning Representations*. 2018.

[179] 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.

[180] Jaco Vermaak, Arnaud Doucet, Perez Patrick, et al. "Maintaining multi-modality through mixture tracking". In: *Proc. Ninth IEEE Int. Conf. on Computer Vision*. IEEE. 2003, p. 1110.

[181] Boyu Wang, Lihan Huang, and Minh Hoai. "Active Vision for Early Recognition of Human Actions". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1081–1091.

[182] Fei Wang et al. "Residual attention network for image classification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3156–3164.

[183] Wenshuo Wang, Junqiang Xi, and Ding Zhao. "Learning and Inferring a Driver's Braking Action in Car-Following Scenarios". In: *IEEE Trans. on Veh. Technol.* 67.5 (2018), pp. 3887–3899.

[184] Wenshuo Wang et al. "Development and evaluation of two learning-based personalized driver models for car-following behaviors". In: *American Control Conference (ACC), 2017*. IEEE. 2017, pp. 1133–1138.

[185] Xiaolong Wang et al. "Non-local neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7794–7803.

[186] Yequan Wang et al. "Attention-based LSTM for aspect-level sentiment classification". In: *Proceedings of the 2016 conference on empirical methods in natural language processing*. 2016, pp. 606–615.

[187] Nicholas Watters et al. "Visual interaction networks: Learning a physics simulator from video". In: *Advances in neural information processing systems*. 2017, pp. 4539–4547.

[188] Lianzhen Wei et al. "Autonomous Driving Strategies at Intersections: Scenarios, State-of-the-Art, and Future Outlooks". In: *2021 International Conference on Intelligent Transportation Systems (ITSC), to appear*. IEEE. 2021.

[189] Mao Wei, Liu Miaomiao, and Salzemann Mathieu. "History Repeats Itself: Human Motion Prediction via Motion Attention". In: *ECCV*. 2020.

[190] Lloyd R Welch. "Hidden Markov models and the Baum-Welch algorithm". In: *IEEE Inform. Theo. Soci. Newsletter* 53.4 (2003), pp. 10–13.

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

[192] Pengxiang Wu, Siheng Chen, and Dimitris N Metaxas. "MotionNet: Joint Perception and Motion Prediction for Autonomous Driving Based on Bird's Eye View Maps". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11385–11395.

[193] Wenhao Wu et al. "Multi-agent reinforcement learning based frame sampling for effective untrimmed video recognition". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 6222–6231.

[194] Xianghua Xie, Mark Jones, and Gary Tam. "Recognition, Tracking, and Optimisation". In: *Int. J. of Computer Vision* 122.3 (2017), pp. 409–410.

[195] Kelvin Xu et al. "Show, attend and tell: Neural image caption generation with visual attention". In: *International conference on machine learning*. 2015, pp. 2048–2057.

[196] Yanyu Xu, Zhixin Piao, and Shenghua Gao. "Encoding crowd interaction with deep neural network for pedestrian trajectory prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5275–5284.

[197] Peixin Xue et al. "Crossing-Road Pedestrian Trajectory Prediction via Encoder-Decoder LSTM". In: Oct. 2019, pp. 2027–2033. DOI: 10.1109/ITSC.2019.8917510.

[198] Kota Yamaguchi et al. "Who are you with and where are you going?" In: *CVPR 2011*. IEEE. 2011, pp. 1345–1352.

[199] Wen Yao et al. "Lane change trajectory prediction by using recorded human driving data". In: *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE. 2013, pp. 430–436.

[200] Shuai Yi, Hongsheng Li, and Xiaogang Wang. "Understanding pedestrian behaviors from stationary crowd groups". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3488–3496.

[201] Hongxiao Yu et al. "A model predictive control approach combined unscented Kalman filter vehicle state estimation in intelligent vehicle trajectory tracking". In: *Adv. in Mech. Engi.* 7.5 (2015).

[202] Zhou Yu et al. "Deep modular co-attention networks for visual question answering". In: *CVPR*. 2019, pp. 6281–6290.

[203] Vinicius Zambaldi et al. "Relational deep reinforcement learning". In: *arXiv preprint arXiv:1806.01830* (2018).

[204] 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).

[205] Wei Zhan et al. "Safe and feasible motion generation for autonomous driving via constrained policy net". In: *Industrial Electronics Society, IECON 2017-43rd Annual Conference of the IEEE*. IEEE. 2017, pp. 4588–4593.

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

[207] Pu Zhang et al. "Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12085–12094.

[208] Yanhao Zhang et al. "Abnormal crowd behavior detection based on social attribute-aware force model". In: *2012 19th IEEE International Conference on Image Processing*. IEEE. 2012, pp. 2689–2692.

[209] Zehua Zhang et al. "Interaction Graphs for Object Importance Estimation in On-road Driving Videos". In: *ICRA*. IEEE. 2020, pp. 8920–8927.

[210] Hang Zhao et al. "Multivariate Time-series Anomaly Detection via Graph Attention Network". In: *arXiv preprint arXiv:2009.02040* (2020).

[211] Shengjia Zhao, Jiaming Song, and Stefano Ermon. "Infovae: Balancing learning and inference in variational autoencoders". In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 5885–5892.

[212] Tianyang Zhao et al. "Multi-agent tensor fusion for contextual trajectory prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12126–12134.

[213] Ting Zhao and Xiangqian Wu. "Pyramid feature attention network for saliency detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3085–3094.

[214] Penghao Zhou and Mingmin Chi. "Relation parsing neural network for human-object interaction detection". In: *ICCV*. 2019, pp. 843–851.

[215] Yi Zhou et al. "Collaborative learning of semi-supervised segmentation and classification for medical images". In: *CVPR*. 2019, pp. 2079–2088.