Deep Learning-Based Interaction-Aware Trajectory Prediction for Autonomous Vehicles

Xiaoyu Mo

School of Mechanical and Aerospace Engineering

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

2022

Statement of Originality

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

30/06/2022

Date

Mo Xiaoyu

Xiaoyu Mo

Supervisor Declaration Statement

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

30/06/2022

Date

Prof. Chen Lyu

Authorship Attribution Statement

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

Chapter 3 is an extension of Mo, X., Xing, Y., & Lv, C. (2021). Graph and Recurrent Neural Network-based Vehicle Trajectory Prediction For Highway Driving. In 2021 IEEE International Intelligent Transportation Systems Conference (ITSC) (pp. 1934-1939). IEEE.

The contributions of the author and co-authors are listed as follows:

- I proposed the idea, designed the algorithms, conducted the experiments, analyzed data, and wrote the manuscript.
- Dr. Xing participated in regular discussions during the development of the algorithms.
- Prof. Lyu supervised the project and designed the system concept.
- The manuscript was reviewed and revised by Dr. Xing and Prof. Lyu.

Chapter 4 is published as Mo, X., Huang, Z., Xing, Y., & Lv, C. (2022). Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network. IEEE Transactions on Intelligent Transportation Systems, In Press.

The contributions of the author and co-authors are listed as follows:

- I proposed the idea, designed the algorithms, conducted the experiments, analyzed data, and wrote the manuscript.
- Mr. Huang and Dr. Xing participated in regular discussions during the development of the algorithms.
- Prof. Lyu supervised the project and designed the system concept.
- The manuscript was reviewed and revised by all authors.

30/06/2022

Date

Mo Xiaoyu

Xiaoyu Mo

Acknowledgements

First of all, I wish to express my sincerest gratitude to my supervisor, Prof. Chen Lyu, who has provided much encouragement, guidance, support, and freedom for my research and made this thesis possible.

Secondly, I wish to express my thanks to all the members and visitors of our AutoMan research group.

Thirdly, I want to show my thanks to all the people who have helped me during my academic journey.

Lastly, I would like to express my special thanks to my parents and sister for their enduring and unconditional love.

"It is far better to foresee even without certainty than not to foresee at all."

— Henri Poincaré (The Foundations of Science)

To my dear family

Abstract

Predicting future trajectories of surrounding agents and conducting motion planning based on interaction predictions are of great importance for ensuring the safety and efficiency of autonomous driving in real-world scenarios, especially under critical driving scenarios. However, trajectory prediction is challenging due to its highly interactive and dynamic attributes. The future trajectory of an agent is usually affected by multiple factors, including the agent's dynamics, its interaction with other surrounding agents, and the road structure. The motion patterns of different traffic participants, such as vehicles and pedestrians, are different and need to be considered separately. Therefore, in real-world applications, trajectory predictors should be able to simultaneously predict the motions of heterogeneous traffic participants. Besides, as for a target agent, there are a variable number of possible future trajectories existing, and this inherent multimodality characteristic should also be considered. Further, as trajectory prediction is not the end goal, it should be incorporated into downstream motion planning and control modules to further improve the overall performance of autonomous vehicles. This integration is also challenging and worthwhile investigating.

In this thesis, a series of data-driven algorithms are developed using deep neural networks to address the opening challenges in trajectory prediction and predictive motion planning for safe and smart autonomous driving.

In order to tackle the trajectory prediction problem and consider those key features (e.g., the target agent's dynamics, interactions, and map features) in a unified way, a deterministic trajectory prediction framework in which the vehicles and the road map are represented using a heterogeneous graph, is proposed. Under this framework, a novel heterogeneous graph social (HGS) pooling method is developed to model the interdependencies among all associated traffic participants and the infrastructures.

Besides, to address the heterogeneity of traffic participants for simultaneous prediction of multi-agent trajectories, a novel Heterogeneous Edge-enhanced graph

ATtention network (HEAT) with a three-channel architecture is proposed. The inter-agent interactions are represented through an edge-featured heterogeneous graph and processed by the designed HEAT network for interaction modeling. Map features are shared across all agents by introducing a selective gate mechanism. Type-specific trajectory decoders are designed for various categories of target agents.

Further, to address the multimodality characteristic of vehicle motion, a map-adaptive multimodal trajectory predictor is proposed. The proposed method can predict a variable number of a target vehicle's possible future trajectories based on the number of candidate centerlines (CCLs). In this approach, the driving scene is first represented using a heterogeneous hierarchical graph, wherein one node represents either an agent or its CCL. Then, a hierarchical graph operator (HGO) with an edge-masking technology is proposed to further model the scene graph. The HGO regulates information flow in the graph operations via edge-masking and outputs the encoded scene features for the downstream map-adaptive trajectory decoder. The map-adaptive predictor, which associates driving modalities with lane options, predicts single-CCL guided, cross-CCL guided, and motion-based predictions in an integrated manner. The motion-based prediction can handle the corner cases where the target vehicle only follows its own dynamics.

Finally, how trajectory prediction results can facilitate downstream motion planning is also investigated. A neural network-based predictive planner is designed by integrating an oracle planner and a trajectory predictor to generate the future reference path for the ego vehicle. In addition to vehicles' dynamics, interactions, and map features, during the training stage of the oracle planner, target agents' ground truth future motions are also taken as input. At the implementation stage of the trajectory planning algorithm, the target agents' ground truth future paths are replaced with the predicted trajectories generated by the pre-trained predictor. Results of the experimental validations on a real-world dataset show that both the oracle and predictive planners outperform the non-predictive baselines, demonstrating the effectiveness of the trajectory planning incorporated with the prediction module.

Results of the experimental validations on different real-world driving datasets show that the proposed trajectory prediction methods achieve state-of-the-art performance with additional capabilities of multi-agent simultaneous prediction and

strong scene adaptability. Besides, the algorithms developed based on the proposed HGS pooling technique and the HEAT network won the championships of two worldwide autonomous vehicle prediction challenges, respectively. These outcomes demonstrate the feasibility and effectiveness of the proposed methods. In addition, the high-level algorithm architectures, methodologies employed, and models developed in this work will expand the current theories of autonomous driving and intelligent transportation systems. They can also be expanded to a wide range of robotics and automation applications.

Contents

List of Figures

List of Tables

Acronyms

ADE	Average Displacement Error
BEV	Bird's Eye View
CSAA	Constant Steering Angle and Acceleration
CCL	Candidate Centerline
CSAV	Constant Steering Angle and Velocity
CNN	Convolutional Neural Network
CTRA	Constant Turn Rate and Acceleration
CTRV	Constant Turn Rate and Velocity
CVAE	Conditional Variational Auto-Encoder
FDE	Final Displacement Error
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GNN	Graph Neural Networks
GRU	Gated Recurrent Unit
HEAT	Heterogeneous Edge-enhanced Graph ATtention network
HGO	Hierarchical Graph Operator
HGS	Heterogeneous Graph Social pooling
LSTM	Long Short-Term Memory
MLP	Multilayer perceptron
OGM	Occupancy Grid Map
RNN	Recurrent Neural Networks
minADE	Minimum Average Displacement Error
minFDE	Minimum Final Displacement Error
RMSE	Root-Mean-Square Error
TDE	Displacement Error over Time

Chapter 1

Introduction

Integrating advanced communication, sensing, prediction, decision-making, planning, and control technologies, intelligent transportation systems (ITS) are believed to make our mobility safer, smarter, more efficient and sustainable [2–8]. Among many promising technologies for ITS, autonomous vehicles play an essential role and have been drawing more and more attention from both academia and industry [9, 10]. In order to operate safely and efficiently in highly complex, dynamic, and interactive driving scenarios, autonomous vehicles will need to smartly reason like human beings via forecasting future motions of surrounding traffic participants during navigation [11, 12]. Hence, trajectory prediction has become a hot research topic, as well as a key module, for autonomous driving in the past decade. However, achieving accurate and robust trajectory prediction is challenging due to the following reasons.

- The variable number of interactive agents. The number of associated traffic participants within a certain area would vary from time to time. Taking an intersection as an example, there could be many traffic participants at times of congestion but few at other times. Thus, during prediction, a comprehensive representation of the driving scene should be able to accommodate an arbitrary number of involved traffic participants.

- The heterogeneity of traffic participants. Traffic participants can be roughly classified as vehicles, pedestrians, and cyclists. However, fine-grained classifications would be more beneficial to the extraction of heterogeneous features

of traffic agents. Thus, the scene representation should be capable of differentiating the motion patterns of heterogeneous traffic participants.

- The complexity of road structures. Road structures in urban areas are complex and diverse. They would affect the motion behaviors of traffic participants to a great extent. So different road structures and various infrastructure elements should be considered in the scene representation for trajectory prediction.

- The interdependencies among traffic participants and infrastructures. The interdependencies among traffic participants and infrastructures are of great importance for interaction modeling and prediction. However, the heterogeneity of traffic participants and the complexity of road structures mentioned above make the interdependencies extraction even harder. For example, an agent's future motion can be affected by the behaviors of other surrounding agents, road boundaries, lane lines, traffic rules, and its own driving objectives. Thus an interaction-aware trajectory prediction method is expected to be able to encode the interdependencies from the associated objects and further improve the prediction accuracy.

- The multimodality of driving behaviors. The motion patterns of agents can be considered inherently multimodal. In real-world driving, there is usually more than one reasonable option for a driver to choose against a specific driving situation, and so does an autonomous vehicle, and the number of options varies from case to case. Thus, an intelligent trajectory prediction method is expected to be human-like and can capture all potential motion modalities of a set of target surrounding vehicles that may affect the behavior of the ego car.

Beyond the aforementioned challenges, the trajectory prediction itself is not the end goal. The predicted results need to be incorporated into the downstream decision-making, planning, and control modules in appropriate manners to contribute to the entire pipelines and closed-loop for autonomous driving. In this thesis, the solutions to these challenges will be investigated and discussed progressively.

1.1 Scope and Overview

Prediction for autonomous driving is a broad topic that includes, but is not limited to, predictions of vehicles' intentions, trajectories, and driving scenes. The trajectory prediction task studied in this thesis focuses on the short-term prediction (three to eight seconds ahead) of focused traffic participants. The trajectory discussed here is a sequence of plane coordinates over a short-term time horizon. More specifically, in this work, the predicted trajectory can be a deterministic trajectory of a single target agent, a cluster of deterministic trajectories of multiple heterogeneous traffic participants, or a bunch of potential trajectories of a single agent for multimodal prediction. Historical states of focused agents are used as part of the inputs across different sub-work, while the Bird's Eye View (BEV) maps or high-definition maps are selectively used for road information representation in different works. An agent's historical states are composed of a temporal sequence of measurements of its plane coordinates. Besides, velocities and yaw angles can be included as well if they are available.

In this thesis, different interaction-aware trajectory prediction algorithms are proposed based on deep learning for addressing multiple issues associated with urban driving. In addition, the impacts of trajectory prediction on data-driven path planning for autonomous vehicles are also investigated. More specifically, in the scene representation for prediction, the interactive objects are described as nodes in a directed graph, either homogeneous or heterogeneous, to adapt to different numbers of focused objects, including moving agents and road elements. Beyond the common graph-based representations, the agents and their candidate centerlines (CCLs) are further studied and represented in a hierarchical heterogeneous graph. First, widely used graph neural networks (GNNs) are applied for interaction modeling, and then a novel heterogeneous edge-enhanced graph attention network is designed to model the heterogeneous interactions for simultaneous multi-agent trajectory prediction. Next, a hierarchical graph operator, which can extend homogeneous graph networks to heterogeneous graphs via edge-masking, is developed. Further, a map-adaptive multimodal trajectory predictor is proposed to simultaneously predict lane-following, lane-crossing, and non-cooperative maneuvers in an integrated manner. It predicts a variable number of optional lane-following trajectories of a target vehicle based on the number of CCLs. Finally, integrated prediction and path planning is investigated using a data-driven approach.

Beyond the scope of trajectory prediction, the proposed high-level frameworks, methodologies, and algorithms can be extended to a wide range of autonomous vehicles, intelligent transportation, robotics, and AI domains. The proposed scene representation methods can be integrated into other important algorithm modules in autonomous driving, such as decision-making and planning. The proposed trajectory prediction methods can be used for driver cognitive model development, as well as intelligent traffic monitoring and management. Moreover, the proposed heterogeneous edge-enhanced graph attention network has great potential to be used in other graph-based learning approaches, including but not limited to traffic forecasting [13], multi-agent systems [14], internet of things [15], social networks [16], medical diagnosis [17], drug discovery [18], and even virus-spreading modeling.

1.2 Major Contributions

The main contributions of this thesis are summarized as follows.

The heterogeneous graph social pooling. A novel heterogeneous graph social pooling (HGS) approach is proposed for interaction-aware deterministic trajectory prediction under urban driving scenarios. It is an important component within the proposed high-level interaction-aware prediction framework. It represents the interactions among vehicles and infrastructure using a newly designed heterogeneous graph with multiple nodes. Within the heterogeneous graph representation, a vehicle node contains the dynamics features extracted from its historical states, and an infrastructure node contains the spatial features extracted from the image of the local map. The graph representation can deal with a variable number of associated objects and can be further modeled and processed by GNNs. As an improvement to the convolutional social pooling method for urban driving, the proposed HGS also takes the vehicle-infrastructure interaction modeling into consideration. By jointly considering the heterogeneous features extracted from multimodal interacting agents through the GNNs; as a result, the trajectory prediction accuracy is significantly improved.

The heterogeneous edge-enhanced graph attention network (HEAT) for deterministic multi-agent trajectory prediction. To further address the trajectory prediction problem for multiple heterogeneous agents, a novel three-channel

framework with a heterogeneous edge-featured interaction graph is designed to model the agents' interactions. The agents' dynamics, interactions, and map features are jointly considered during prediction. More specifically, the agents' dynamics information is first extracted from their historical states using type-specific encoders. A novel heterogeneous edge-enhanced graph attention network (HEAT) is then proposed to process the interaction graph. A HEAT network can be composed by stacking several HEAT layers. A HEAT layer is an extension of a graph attention network (GAT) layer for heterogeneous graphs with edge features. Given a heterogeneous graph with edge features, it rebuilds the node features by aggregating features from neighboring nodes via an edge-enhanced masked attention mechanism. For a target node, the HEAT layer first calculates the attention coefficients over its neighborhood, considering the features of both nodes and edges. Then the target node's feature is rebuilt as a weighted sum over mixtures of its source nodes' features and corresponding edge features. Besides, the map features are shared across all focused agents by introducing a selective gate mechanism. The proposed framework realizes simultaneous multi-agent trajectory prediction while achieving state-of-the-art performance. The proposed HEAT network can be extended and used in many other application domains with graph representations.

The map-adaptive multimodal trajectory predictor. Different from other existing methods, which predict a fixed number of possible future motions of an agent, the proposed new map-adaptive predictor can predict a variable number of future trajectories of an agent based on all feasible CCLs. This predictor can provide not only single-CCL guided motion predictions but also a prediction cross CCLs together with a motion-based prediction. These three kinds of predictions are generated integrally via a single graph operation. The driving scene is represented with a heterogeneous hierarchical graph, wherein a node can represent either an agent or its CCL. An agent node contains its dynamics feature encoded from its historical states, and a CCL node contains the CCL's sequential feature. A hierarchical graph operator with the edge-masking technique is proposed for this predictor to regulate the information flow via graph operations and obtain the encoded scene feature for the trajectory decoder. Experimental results on a real-world driving dataset show that our new method can simultaneously predict map-compliant and motion-based trajectories within a single graph operation and surpass other existing approaches with respect to prediction accuracy, validating its feasibility and effectiveness.

The neural network-based predictive motion planner. Most of the existing studies address the trajectory prediction and path planning problems separately. However, the impacts of prediction on trajectory planning have rarely been studied. Since both trajectory predictor and motion planner can generate a sequence of waypoints, in this thesis, a novel data-driven predictive planner is proposed based on neural networks, integrating the prediction results into the downstream planning module. More specifically, the predictive planner is a combination of an oracle planner and a trajectory predictor. The oracle planner is firstly trained with the ground truth of the target agents' future trajectories, and then it is converted to a predictive planner when replacing the ground truth with the predicted trajectories. Next, the impacts of trajectory prediction accuracy on the planning performance are further investigated via comparison with ground truth-guided trajectory planning. The results show that the added prediction module does improve planning performance. However, the prediction accuracy may not necessarily be as high as possible, as there is a saturation effect on the performance improvement brought by the prediction.

Inference time and performance. Tab. 1.1 lists the inference time and displacement errors of the methods developed in this thesis and two strong baselines. The inference time is obtained by running these methods on similar Nvidia GeForce RTX graphics processing units (GPUs) with batch size (BS) set to 32. The displacement errors or the methods developed in the thesis are obtained by submitting the prediction results to the Argoverse online benchmark (test set) [19]. A short dash means that the corresponding value is not available. For example, the displacement errors of HEAT-MATP and PrePlan are not shown because these methods are developed using the INTERACTION dataset [20]. It can be seen that the methods developed in this thesis show competitive performance and greatly reduce the inference time.

1.3 Outline of the Thesis

Chapter 1 introduces the trajectory prediction problem of autonomous driving, defines the scope of the thesis, and provides an overview of it. Major contributions are listed, followed by an outline of the thesis.

TABLE 1.1: Comparison with existing methods

Method	Inference time (sec / batch, BS=32)	Device	FDE (K=1)	FDE (K=6)
HGS-SATP	0.02	2080	3.89	-
HEAT-MATP	0.05	2080	-	-
HGO-MMTP	0.02	2080Ti	4.18	1.40
PrePlan	0.03	2080	-	-
LaneGCN [21]	0.08	2080Ti	3.78	1.36
DenseTNT [22]	0.63	3080	-	1.49

Chapter 2 reviews the physics-based and learning-based trajectory prediction methods. The physics-based methods are reviewed according to a taxonomy similar to existing reviews. The learning-based methods are reviewed based on two classification criteria: scene representation and trajectory decoding.

Chapter 3 proposes the high-level trajectory prediction framework with a novel heterogeneous graph social pooling approach. This is our first attempt to design a unified trajectory predictor that can accommodate an arbitrary number of interactive agents and simultaneously consider the target agent's dynamics, interaction with surrounding agents, and the road structure.

Chapter 4 proposes the multi-agent trajectory predictor that can simultaneously predict trajectories of an arbitrary number of heterogeneous agents. The predictor represents the interaction among agents as an edge-featured heterogeneous graph and leverages the proposed heterogeneous edge-enhanced graph attention network (HEAT) to model the interaction graph. A map selector based on the gate mechanism is proposed to share the same map across different agents in the scene. The predictor then combines the target agent's dynamics, interaction, and map features to generate trajectory predictions for all the target agents.

Chapter 5 addresses the inherent multimodality of driving behaviors by proposing a map-adaptive multimodal predictor. It can predict a variable number of future trajectories of an agent according to the availability of all CCLs. In addition to single-CCL guided predictions, it also provides a cross-CCL prediction and a motion-based prediction integrally. A hierarchical graph operator with edge-masking technology is proposed for modeling the scene graph wherein a node represents either an agent or its CCL.

Chapter 6 designs a predictive neural motion planner that is obtained by training an oracle planner with access to the ground truth of future trajectories of target agents and replacing the ground truth with predicted trajectories during implementation. The relationships between trajectory prediction and trajectory planning are investigated based on the developed predictive planner.

Chapter 7 summarizes the thesis and provides some promising directions for future work in the areas of trajectory prediction and planning.

Chapter 2

Literature Review

In this chapter, the trajectory prediction methods are reviewed with the following two categories first: the physics-based and learning-based methods. Then, they are further classified and discussed according to different criteria. The dichotomy in this chapter shows not only the difference between the two categories but also the two-stage development of trajectory prediction methods. The physics-based methods dominate the first stage, while the learning-based methods dominate the second stage. The physics-based methods are simple and suitable for short-term prediction, but they are no longer state-of-the-art, so our taxonomy of these methods is similar to the previous surveys [23, 24]. The learning-based methods show much better performance than physics-based ones because of their potential to handle different factors for prediction. They are developing rapidly and require new taxonomy for the latest works. We divide a learning-based prediction system into two stages: 1) scene representation and encoding followed by 2) trajectory decoding, and categorize them according to their differences in these two stages, respectively. This is different from previous surveys [11, 24].

2.1 The Physics-Based Trajectory Prediction

Physics-based trajectory prediction (PTP) methods represent the motion of vehicles with dynamic or kinematic motion models compliant with the laws of physics. A future trajectory is predicted via state evolution based on these models. Linear

models (constant velocity (CV) and constant acceleration (CA) models), which assume straight motion, are the simplest models. The straight motion assumption of CV and CA oversimplifies vehicular motion because of rotation ignorance. Curvilinear models take into consideration the turn rate (TR) of the vehicle. Similar to linear models, curvilinear models contain two types, namely Constant Turn Rate and Velocity (CTRV) and Constant Turn Rate and Acceleration (CTRA) models. Coupling steering angle and velocity leads to Constant Steering Angle and Velocity (CSAV) and Constant t Steering Angle and Acceleration (CSAA) models. Authors of [25] compare and evaluate these models. More details of these models can be found in their work. These models can be used for physics-based trajectory prediction in many ways.

Inspired by the taxonomies provided in [23, 24], we subdivide physics-based methods into state retention-based methods, reachable set-based methods, Monte Carlo simulation-based methods, and state estimation-based methods. State retention methods produce a single trajectory according to the target vehicle's current states, while reachable set-based methods produce a union of all reachable states. Monte Carlo simulation-based methods try to approximate the distribution of the target vehicle's future motion, and state estimation-based methods introduce uncertainty estimation into physics-based methods.

PTP via State Retention. State retention is a straightforward approach to trajectory prediction. These methods assume that the target vehicle's current states are available, and it will maintain its states (e.g., velocity, acceleration, and curvature) and follow a motion model. Authors of [26] propose a multilevel collision mitigation approach. They model the target vehicle's future motion via a constant acceleration model along curved coordinate axes and determine the time to react for collision circumvention based on the vehicles' current states, predicted states, and physics constraints. Rather than using oversimplified constant velocity (CV) and constant acceleration (CA) models, authors of [27] use the Constant Turn Rate and Constant Tangential Acceleration (CTRA) model as the motion model for cooperative path prediction. The state retention-based methods are simple and computationally efficient for real-time applications with limited resources. However, these methods ignore uncertainties about the future motions of traffic participants. State retention can only produce reliable predictions within a very short

horizon, so these methods are often used in object tracking rather than long-term trajectory prediction.

PTP via Reachable Set. Reachability analysis has been widely applied to the safety assessment of autonomous vehicles, where the reachable set, the union of all reachable states of a moving object within a time horizon, is computed as a prediction of traffic scene [28]. The original scene transition is approximated via Markov chain abstraction then the probability distribution can be computed by the Markov chain. Authors of [29] propose a Set-Based Prediction Of Traffic Participants (SPOT). SPOT can predict the future occupancy of other traffic participants considering their optional maneuvers under physical constraints. Traffic regulations are included to reduce the size of the occupancy set. However, a larger occupancy set can be provided as soon as a traffic participant violates traffic rules. The larger occupancy set assures collision-free planning of the ego vehicle. Reachability analysis of vulnerable road users is proposed in [30], where the authors argue that it is impossible to predict the exact movements of pedestrians because of the unknowable intentions of pedestrians. The reachable sets of pedestrians are computed based on a dynamic model with maximal physical limits that can be predefined or iteratively estimated. Authors of [31] calculate a forward reachable set of the ego vehicle and backward reachable sets of other agents, which may be occluded, to identify risk-inducing regions and focuses on only potentially dangerous agents for motion planning. Compared to state retention methods that produce a deterministic prediction according to a vehicle's current state, reachable set-based methods consider all possible motions of a vehicle. Even though the size of the reachable set can be reduced via constraints [32, 33], this kind of prediction can lead to safe but conservative motion when applied to downstream decision-making and planning modules.

PTP via Monte Carlo Simulation. Monte Carlo simulation methods try to approximate the unknown distribution of a vehicle's future motion by simulating motion models with randomly selected initial states. They can be described in three steps: 1) generate inputs and initial states randomly; 2) perform a deterministic simulation starting at each initial state according to the inputs generated at the previous step; 3) aggregate the results of individual simulations into the final result [34]. Monte Carlo simulation is used to predict the future motion of

surrounding vehicles in [35, 36]. These methods rely on deterministic simulations and ignore the uncertainty in the process.

PTP via State Estimation. Uncertainties introduced by imperfect models, process noise, and measurement noise can be addressed via Bayesian filtering-based state estimation methods. Bayesian filtering operates basically in two steps: prediction and update. In the prediction step, the estimated state at the current time t is fed to the evolution model to produce a predicted state for the next time step $t+1$. In the update step, the measured state at time $t+1$ is combined with the predicted state into an estimated state for time $t+1$. A predicted trajectory with associated uncertainty can be obtained by iterating the prediction step of Bayesian filters. Assuming linear models and normally distributed process and measurement noises, Kalman filter [37] can be applied for uncertainty-aware trajectory prediction. Authors of [38] use the simple bicycle motion model with a Kalman filter to predict trajectories of the ego and surrounding agents. The predicted trajectories are then used to compute possible collisions between paired trajectories for collision risk estimation. For problems with non-linear models, the extended Kalman filter (EKF) and the unscented Kalman filter (UKF) can be used in place of the Kalman filter [39]. Authors of [39] use CTRA as the motion model and apply UKF for prediction. For problems with arbitrary models and random noises, a particle filter can be used in place of the above-mentioned filters [40, 41]. The above-mentioned methods rely on a single motion model, ignoring that different motion models are suitable for different driving situations. For example, a CA model is suitable when the vehicle moves with a constant longitudinal acceleration but inappropriate when the vehicle is making a turn. To employ multiple kinematic models for different situations, authors of [42] propose heuristic rules based on driving recordings to select the suitable model for a specific scenario. Rather than heuristically selecting appropriate motion models, authors of [43] propose to continuously estimate probabilities of motion models and select the one with the highest probability. Three motion models are used in [43] for driving straight, turning left, and turning right, respectively. Authors of [44] implement four motion models, namely constant location, constant velocity, constant acceleration, and constant jerk, for interacting multiple models (IMM) estimation. The IMM filter used in [44] can provide a combined prediction considering all motion models. IMM estimation can be applied with a combination of linear and non-linear motion models. Authors of [45] apply

KF and EKF for linear (CV and CA) and non-linear (CTRV) models, respectively, and combine these estimations via IMM estimation.

The physics-based methods are simple and computationally efficient. However, they ignore interactions among traffic participants and other factors, making them more suitable for short-term prediction in tracking and risk assessment tasks rather than longer-term prediction tasks. Despite their inaccuracy in long-term prediction, physics-based methods can be used by learning-based methods to generate valid predictions [46].

2.2 The Learning-Based Trajectory Prediction

The learning-based trajectory prediction (LTP) methods are attracting more and more interest in the field of trajectory prediction because of the availability of large-scale datasets [20, 47–53] recorded in the real world and the success of machine learning methods, especially deep learning, in plenty of research areas [54, 55]. For a comparison of widely used datasets, please refer to [56].

Similar to the widely used Encoder-Decoder structure [57] in the field of deep learning, where the system consists of two main components: encoder and decoder, most LTP methods can be divided into two main stages: encoding and decoding. The encoding stage takes as input the scene representation and generates a hidden state with a fixed shape as an intermediate understanding of the scene, while the decoding stage maps the hidden state to trajectories according to task requirements. The selection of encoding methods is often representation-oriented while decoding methods are often task-oriented. For example, convolutional neural networks (CNNs) are often chosen as encoders for rasterization-based scene representations [58–60], and graph neural networks (GNNs) are more suitable for map-adaptive multimodal predictions [61]. We also observe that encoding and decoding stages are often decoupled, and the same encoding method can be used for different prediction tasks by using different decoders. Based on the above-mentioned modularization and observation, we propose to classify LTP methods according to two criteria: scene representation and trajectory decoding. This is different from the classifications

in [11], where the authors present three classifications of deep learning-based prediction methods according to input representation, output type, and prediction method, respectively.

2.2.1 Classification of LTP According to Scene Representation

In this subsection, we provide a classification of LTP methods according to how the driving scene is represented. We divide scene representations into five categories: raw sensor input, sequence, grid, graph, and hybrid representations. We also discuss encoding methods for these representations.

2.2.1.1 LTP with Raw Sensor Input

These methods rely purely on raw data available with onboard sensors. These methods are more suitable for autonomous driving in rural areas without assumptions about perfect observation and map availability. Without high-level feature extraction, these methods avoid potential information loss and can learn to extract useful features for prediction. However, raw sensor inputs are often high-dimensional and thus require more computational efforts for encoding. This problem can be mitigated via parameter sharing among different modules of an autonomous driving system [62].

Authors of [63] construct a 3D voxel grid from point cloud data, which is collected by a roof-mounted LiDAR and represented in the current vehicle coordinate system, for each frame. Then they stack 3D tensors over the past n frames to form a 4D tensor. Convolutional operations of different dimensions are applied for extracting temporal and spatial features. This representation is adopted in a follow-up work [62] that handles online mapping, perception, prediction, and planning in a unified way.

2.2.1.2 LTP with Sequence-Based Scene Representation

These methods simply represent the scene as a sequence of the target vehicle's historical states, where the states may include positions, velocities, and heading

angles and can be estimated with onboard sensors or obtained via vehicle-to-vehicle communications [64].

Authors of [65] represent the scene of a target vehicle with a sequence of its position (relative to the ego vehicle), heading angle (in radians), and speed (in meters per second) and apply a three-layer Long Short-Term Memory (LSTM [66]) network to predict its destinations at a T-shaped intersection. They investigate three observation lengths (0.2, 0.6, and 1.0 seconds) and report that a longer history horizon leads to better performance while diminishing returns are observed after 0.6 seconds. Rather than considering only the target vehicle's states, authors of [67] propose to take as input the ego vehicle's speed and yaw rate in addition to the target vehicle's position and velocity relative to the ego vehicle at each time step. They formulate the trajectory prediction task as a sequential multi-class classification problem by introducing a local occupancy grid map (OGM). Driver intention recognition and trajectory prediction are jointly generated via dual LSTMs in [68], where the sequential inputs are divided into lateral and longitudinal sequences. The first LSTM (intention recognizer) takes as input the lateral sequence and outputs an estimated driver intention. Then the second LSTM (trajectory approximator) takes as input both the longitudinal sequence and the embedded semantic understanding of driving intentions and generates the predicted trajectory. Lateral constraints can be applied for feasible trajectory prediction. Authors of [69] generalize their previous work on driver intention recognition [65] to predict intentions and trajectories jointly. A hyper LSTM is applied to the sequence of the target vehicle's position, heading, and speed for scene encoding.

Sequence-based representations are simple and require fewer computational resources. However, they can only be used for short-term trajectory prediction because they ignore the interdependencies among traffic participants and the infrastructures, which have a great impact on the target vehicle's future motion. These methods are often used as baselines in recent works to show the advantage of other LTP methods [58, 60, 70, 71].

2.2.1.3 LTP with Grid-Based Scene Representation

One way to model interdependencies between agents (and infrastructures) is to represent the scene as a grid, which can be either an occupancy grid or a Bird's

Eye View (BEV) image since the grid provides a spatial feature of the driving scene. Occupancy grids are often used for highway scenarios, where the road is well-structured, and the grid can be easily obtained. For urban scenarios, fine-grained grids (e.g., images and image-like arrays) are often used to represent driving scenes.

Highways are well-structured, and grids are often used to model interactions between vehicles. A dynamic occupancy grid map (DOGMa), which provides a spatial occupancy and velocity distribution over the local area, is adopted in [72] to represent the scene. The DOGMa, rather than raw sensor data, is fed into a CNN for scene encoding. Compared to raw sensor data inputs, DOGMa has two main advantages: 1) it provides a spatial occupancy and velocity distribution via sensor fusion techniques, which are hard to compensate for via learning-based methods; 2) it is independent of sensor setups and allows learning with various sensor inputs. DOGMa serves as an adapter between various raw data and the CNN encoder. The approach can be applied with any sensor inputs as long as a DOGMa can be constructed. The work in [72] is further extended with a sequence of DOGMa in [73], where a ConvLSTM [74] is used as the scene encoder. Authors of [58] represent the scene with a 13×3 spatial grid defined around the target vehicle and propose a convolutional social pooling technique to model the interdependencies of all vehicles in the scene. The grid is populated with dynamics features of corresponding vehicles according to their locations. The 13×3 grid may include redundant vehicles (up to 38) in congestion. Authors of [60] propose to use a 3×3 grid for scene representation so that only eight vehicles that have the most impact on the target vehicle's behavior are considered. Similar to [75], vehicles preceding and following the target vehicle and the three closest vehicles on the adjacent lanes are selected.

For urban driving, the roads often have complex structures. Thus, fine-grained grids are required to represent not only interactions among traffic participants but also the impacts of infrastructures. Multi-Agent Tensor Fusion (MATF) [70] represents the scene via a BEV image and applies fully convolutional layers to extract spatial features from it. Because dynamic features are ignored in the BEV representation, MATF needs to combine agents' dynamic features and the static spatial feature via spatial alignment. Authors of [76] propose a unified representation, which rasterizes an HD map surrounding the target agent and the motions of traffic participants into an agent-specific BEV image and uses CNNs for scene encoding. The rasterization represents map elements (e.g., roads, crosswalks, lanes) and agent

tracks with different colors. The image representation is straightforward, but the overlaps between road elements and agents may cause information loss. Rather than rasterizing the scene into a BEV image, authors of MultiPath [59] propose to represent the scene context as a three-dimensional array, where the first two dimensions represent locations from a top-down orthographic perspective, and the third dimension represents a stack of historical observations over a fixed traceback horizon via multiple channels. MultiPath provides a simple way to represent static and dynamic information with a single array without information loss caused by BEV images. CNN backbones are used for scene encoding. Despite the simplicity of the above methods, pixel resolution always needs to be carefully determined, considering the trade-off between image size and representation ability.

Despite grid-based representations' ability to model interaction for even urban driving, they have the following drawbacks. First, the resolution of grids and the size of receptive fields are two critical hyperparameters that need to be tuned carefully for performance and efficiency trade-offs. Second, rasterizing scenes into grids inevitably causes information loss. For example, overlapping lane elements and agents' states make it hard to distinguish the features of different objects. Third, grid representations cannot explicitly represent interactions, making it hard to inject expert knowledge of interaction identifications. For example, two cars that are moving in the opposite direction may have weak interaction despite being close to each other if a road median exists. Fourth, representing all road elements in a grid ignores the connectivity between road elements provided by HD maps. These disadvantages can be mitigated to some extent via graph representations.

2.2.1.4 LTP with Graph-Based Scene Representation

A graph is a natural way to represent objects and their relationships with nodes and edges. To model interactions among agents, we can represent agents as nodes of a graph and connect pairs of agents that interact with each other. Undirected edges can be applied to represent mutual influences, while directed edges can represent asymmetric influences. To represent a road network, we can construct a graph with junctions as nodes and roads between junctions as edges. To represent the whole driving scene, we can construct a heterogeneous graph to accommodate all the objects (e.g., vehicles, pedestrians, lanelets, waypoints, and traffic signs). Graph representations allow to accommodate a variable number of objects and explicitly

represent interdependencies among them with edges. Expert knowledge can be injected into graph construction by assigning edges between objects determined to have more interdependency.

Authors of [77] construct the graph with local connectivity, where each node contains a vehicle's feature at the current time, and apply adaptions of Graph Convolutional Networks (GCNs) [78] and Graph Attention Networks (GATs) [79] to model the interaction. It conceptually proves that modeling interaction as a graph improves prediction accuracy. However, it ignores temporal dependence between consecutive time steps by producing the final output with a simple feed-forward layer. Temporal features extracted using Recurrent Neural Networks (RNNs) are used as node features in [1]. GRIP [80] considers temporal features by representing the scene with an undirected spatio-temporal graph, where each node corresponds to an object at a time step in the scene. Objects that have interactions are connected via spatial edges at each time step. Temporal edges connect the same objects at different time steps. GRIP applies convolutional layers and graph operations to capture temporal and inter-object interaction features. SCALE-Net [81] proposes to represent the interaction with an edge-featured homogeneous graph, where the nodes contain physical states of corresponding agents, and the edges contain relative states between two connected agents. All agents are placed in their exclusive coordinate systems for generalization, and edge attributes are used to preserve spatial relationships among agents. Edge-enhanced Graph Convolutional Neural Networks (EGCNs) [82] are used to explore edge features in the constructed graph. One common issue of the above-mentioned scene representations is that they all use homogeneous graphs and ignore infrastructures and the heterogeneity of traffic participants. A heterogeneous graph is constructed for scene representation in [83], where an extra node containing the map feature is introduced to the interaction graph. The map feature is extracted via a CNN. VectorNet [84] uses a fully-connected hierarchical graph to model the interactions between vehicles and infrastructures, where each sub-graph represents an object, and applies GNNs to model interactions. It operates on vector representations, where all the agents' trajectories, lane lines, and crosswalks are represented by vectors. This vector representation is used in their follow-up work [85]. LaneGCN [21] represents the actor and map separately. It constructs a lane graph from the HD-map preserving connectivity between lane nodes and saves four types of connections for downstream encoders. Rather than representing agents as single nodes, LaneRCNN [86]

constructs a dynamics-related lane graph for each agent, with each node containing geometric, semantic, and agent information. EvolveGraph [87] constructs a fully-connected graph with heterogeneous agent nodes and one context node for multi-agent trajectory prediction. The same context node is linked to all target agents, while each agent should have a local context. Social-WaGDAT [88] proposes Wasserstein Graph Double-Attention Network to model spatio-temporal interactions and employs kinematic constraints for the final prediction. It uses occupancy density maps and velocity fields generated from training data as a priori knowledge of the context information.

There is a cluster of LTP methods that apply attention mechanisms for interaction modeling without specific interaction graph construction [89–91]. These methods are regarded as graph-based methods considering that attention coefficients between pairs of objects can be seen as weighted edges in graphs.

LTP with graph-based scene representation relies heavily on the graph's structure, while graph construction is still an open problem. One straightforward way is to connect every pair of nodes in the graph as did in VectorNet [84] and transformer-based methods. However, this approach introduces redundant edges and requires more computational efforts. Another way is to construct an interaction graph according to heuristics, for example, connecting two agents if the distance between them is below a predefined threshold. However, it is very hard to find a universal heuristic for graph construction. A promising way is to construct the graph via interaction identification [92], connecting two objects only if there is an interaction identified between them. Relying on immature interaction identification methods may lead to risk if a strong interaction is not identified. Graph representation provides a unified way to represent driving scenes with different kinds of objects for trajectory prediction. However, GNNs for heterogeneous graphs are less studied despite the rapid development of GNNs [71, 93] in recent years.

2.2.1.5 LTP with Hybrid Scene Representations

Hybrid scene representation provides an easy way to fuse and balance different types of information for the trajectory prediction task [71, 94–97]. IntentNet [94] represents the scene with voxelized 3D point cloud data as in [63] and a rasterized map containing static (e.g., lanes and traffic signs) and dynamic (e.g., traffic

lights) information. Multiple CNNs are used for scene encoding. Similar hybrid representations are adopted in [95, 98]. Trajectron++ [96] represents agent interactions with a directed graph and encodes a local map for each modeled agent to make use of map information. Interaction and map information are concatenated to produce a representation, which can be extended to include additional information. HOME [97] uses multi-channel rasterization to represent maps and agents' historical states while agents' scalar histories are also represented separately. This representation can be regarded as a combination of rasterization and graph representations since the inter-agent interactions are modeled via an attention mechanism, where attention coefficients can be seen as edge weights of a fully connected graph. Authors of [71] propose to represent the scene with a heterogeneous edge-enhanced graph and a static BEV map of the local area. The heterogeneous graph is constructed to represent interactions among traffic participants of different types, and the BEV map is used for spatial information. A heterogeneous edge-enhanced graph attention network (HEAT) is proposed to model the interaction, and a CNN is applied to retrieve spatial information from the map.

2.2.2 Classification of LTP According to Trajectory Decoding

In this section, we review existing trajectory decoding options, which reflect the objectives of different trajectory prediction methods. We roughly categorize them into two groups: unimodal and multimodal trajectory predictions. For a review of intention and occupancy map prediction, please refer to [11]. As multimodal prediction has received rising interest in recent years, we subdivide multimodal prediction methods into four types as we will present.

2.2.2.1 Unimodal Trajectory Prediction

Unimodal trajectory prediction (UMTP) methods predict a single trajectory of a traffic participant over a prediction horizon. The predicted trajectory here is not restricted to a sequence of XY coordinates of the target vehicle [1, 70, 71, 80]. It can also be displacements between two consecutive time steps [77], combinations

of longitudinal velocities and lateral positions [75], sequence of positions with estimated uncertainties [99, 100], or sequence of bounding boxes [63]. Above unimodal methods tend to converge to an average trajectory over multiple driving intentions. This drawback can be mitigated by conditioning unimodal trajectories on driving intentions [68, 94, 101]. Authors of [68] classify driving intentions into lane keeping, left lane change, and right lane change and propose to predict trajectories conditioned on semantic understanding of driving intentions using LSTMs for highway driving. IntentNet [94] further classifies driving intentions into eight categories for urban driving and outputs a sequence of bounding boxes conditioned on embedded intention scores as the predicted trajectory. Authors of [101] propose to classify driving intentions via LSTMs and then use optimization-based methods to predict intention-aware trajectory. Even though intention-aware unimodal prediction methods mitigate mode-collapse risks of other unimodal prediction methods, their dependence on intention recognition makes them susceptible to mistakes in intention classification and the definition of intention set. Besides, intention labeling for intention recognition is time-consuming and error-prone.

2.2.2.2 Multimodal Trajectory Prediction

Multimodal trajectory prediction (MMTP) methods try to capture the multimodality of driving behaviors and output trajectories for all intentions with or without mode probabilities. We classify multimodal prediction methods into intention-aware, sampling and scoring-based, multimodal regression-based, and map-adaptive methods.

Intention-aware MMTP. These MMTP methods can be constructed by generalizing intention-aware unimodal predictors to output trajectories for all intentions in the set rather than a selected intention. Authors of [58, 102] classify driving intention on highways into six maneuver classes and propose a maneuver-based trajectory decoder to estimate maneuver probabilities and generate parameters of a sequence of bivariate Gaussian distributions over the prediction horizon as final outputs. Softmax layers and an LSTM decoder are adopted for maneuver estimation and trajectory generation, respectively. Rather than defining intentions in a handcrafted manner, MultiPath [59] proposes to learn a set of K anchor trajectories with assigned probabilities as intentions via the k-means algorithm. It then

regresses offsets from anchor waypoints along with uncertainties for multimodal prediction.

Intention-aware MMTP methods can handle the multimodality of driving behaviors. However, similar to intention-aware UMTP, there are some drawbacks of these approaches. First, their performance depends on the predefined intention set, which cannot cover all situations in real-world driving. Second, these methods rely on manually labeled intentions, which is time-consuming and inflexible. For example, re-labeling is needed as long as the intention set is updated. Third, driving intention is coarse-grained, while driving behavior is fine-grained. The former may not be able to capture the subtlety of the latter for accurate prediction.

Sampling and scoring-based MMTP. These methods try to model the subtlety of driving behaviors via sampling from continuous latent space or discrete Euclidean space, then capture multimodality by scoring sampled trajectories according to some criteria. DESIRE [103] uses conditional variational auto-encoder (CVAE) [104] to learn a stochastic latent variable that is able to generate a diverse set of predictions via sampling and decoding. Sampled trajectories are further scored and refined to output a predefined number of trajectories as the prediction. Authors of [69] propose to represent multimodal predictions as a weighted mixture of Gaussian distributions obtained via a Mixture Density Network (MDN) [105] output layer. They apply a clustering algorithm to produce a ranked set of predictions according to probabilities. Authors of [67] use OGM to discretize the region of interest, thus formulating the trajectory prediction as a sequential multi-class classification problem. Beam search is applied to generate a set of trajectories with the most probabilities. TNT [85] first selects M target points from a large number of points uniformly sampled on lane centerlines, then produce M trajectories for each target. These trajectories are then scored to produce a set of K trajectories as the final prediction. HOME [97] generates a probability heatmap for a local area and proposes two target sampling algorithms for miss rate (MR) and Final Displacement Error (FDE) optimization, respectively. Full trajectories are generated for all targets via fully connected layers.

Sampling and scoring-based MMTP methods are expected to model fine-grained driving behaviors and can potentially generalize to different driving situations. However, this generalizability comes with costs. First, It is hard to explain what the latent space is and how it corresponds to driving multimodality. Second, Euclidean

space sampling makes the sampled trajectories explainable and intuitive. However, these methods often rely on a large number (e.g., 1,000) of samples to capture the distribution. Abundant sampling requests much time and computational resources, and the subsequent scoring is even more time-consuming.

Multimodal regression-based MMTP. These methods directly output a fixed number (M) of predictions that are expected to cluster driving behaviors into M modes. The mode collapse problem is addressed by optimizing the winning mode(s). MTP [76] directly outputs a fixed number of trajectories via fully connected layers for multimodal prediction. Different numbers of modes ($M = 1, 2, 3, 4$) are implemented for comparison, and MTP with $M = 3$ is reported to show the best performance on all metrics in [76]. WIMP [106] learns a mixture of M predictors with LSTM-based decoders to produce M different trajectories for multimodal prediction. Similarly, LaneGCN [21] learns a list of K decoders and a classifier for stochastic multimodal prediction.

Multimodal regression-based MMTP methods avoid predetermining an intention set. However, these methods require predetermining the number of driving modes (M), and the selection of M has an impact on the performance. It is obvious that a predefined M is not applicable to different situations. For example, MTP with $M = 3$ is not able to capture the situations where the target vehicle has four or more options. In addition, the modality, in this case, is hard to explain. For instance, an MTP method with $M = 2$ may output *fast* and *slow* modes when the behavior is restricted to going straight. However, it may output *left* and *right* modes when the target vehicle is approaching a T-shaped intersection with the same decoder.

Map-adaptive MMTP. Map-adaptive trajectory prediction is less studied in the literature compared to other multimodal trajectory prediction problems. However, map-adaptive methods can generalize to different lane topologies, such as intersections, roundabouts, and other unusual road structures [61].

GoalNet [61] represents the inputs and outputs in a path-relative coordinate frame and proposes to use a GNN to generate a variable number of trajectories based on a set of available paths of the target vehicle. In addition to lane-following predictions, GoalNet also provides a separate channel to cover non-map-compliant

driving behaviors. GoalNet links up driving modalities with paths available in specific driving situations, thus making itself explainable, scalable, and map-adaptive. Conditioning prediction on a lane addresses the mode collapse problem since lanes are separated from each other.

Despite the map-adaptive capability of GoalNet, there are some problems that should be addressed in the future on map-adaptive prediction. GoalNet ignores the vehicles behind the target and those in other lanes, which can have a critical impact on the target vehicle's behavior. For example, the behavior of a target vehicle operating an unprotected left turn will be affected by the vehicles in the lane on its left side. However, GoalNet does not consider this information. In addition, the goal-free prediction, although also named motion-based, is not purely based on the target vehicle's past motion, thus not able to cover the critical corner case where a vehicle drives in an uncooperative way.

This chapter divides trajectory prediction methods into physics-based and learning-based categories and reviews them according to new subdivisions (A table summarizing these methods can be found in Appendix-E). The reviewed learning-based methods mainly focus on those based on supervised learning since they are most related to this project compared to other learning methods, such as clustering and reinforcement learning. We refer the readers to [11, 24, 107] to gain a broader understanding of trajectory prediction methods. The technical contents of this project fit into the literature as follows. Chapter 3 represents the driving scene as a directed heterogeneous graph and produces a unimodal trajectory for a single target vehicle. Chapter 4 uses hybrid scene representation with interaction and map information processed separately and combined via concatenation for simultaneous multi-agent trajectory prediction. Chapter 5 tackles the multimodal prediction task in a map-adaptive manner. A directed hierarchical graph is constructed to represent interdependencies among agents and their candidate centerlines.

Chapter 3

Deterministic Trajectory Prediction for a Single Target Vehicle

This chapter develops an interaction-aware trajectory prediction framework to predict the future trajectory of a single target agent. A single heterogeneous graph is constructed to represent traffic participants and infrastructure. A novel heterogeneous graph social (HGS) pooling approach is proposed to model the vehicle-infrastructure interactions for urban driving.

3.1 Introduction

Researchers in the field of autonomous driving have proposed many works for trajectory prediction, and these methods fall into three categories: physics-based, maneuver-based, and interaction-aware methods [23]. Physics-based methods consider the object's individual dynamics to predict its motion ignoring possible maneuvers restricted by the road structure and neighboring agents' impacts [38]. Maneuver-based methods consider maneuver options and predict trajectory conditioned on maneuvers ignoring the impact of surrounding vehicles [108]. Interaction-aware methods have attracted more and more interest recently in that they: 1)

naturally treat driving as an interactive activity; 2) show better performance compared to pure physics-based and maneuver-based methods; 3) can be extended to take physics and maneuvers into account [1, 58, 60, 61, 80, 83, 88].

Interaction-aware approaches, which usually leverage learning-based algorithms and consider driving as an interactive activity, mainly use either or both graph neural networks (GNNs) and convolutional neural networks (CNNs) for interaction modeling [71]. In [77], a graph with local connectivity is constructed to model the interactions. It conceptually proves that modeling interaction as a graph improves the prediction accuracy, but it ignores vehicles' time serial features. GRIP applies graph operations on sequential features to take inter-vehicular interactions into account, ignoring the effects brought by infrastructures [80]. SCALE-Net uses an Edge-enhanced Graph Convolutional Neural Network (EGCN) on the graph with physical states as node features to explore edge features in the constructed graph [81]. Homogeneous graphs, where all nodes only represent vehicles with neglect of infrastructures' effects, are used to model interactions in the above-mentioned works. VectorNet uses a fully-connected hierarchical graph to model interactions among vehicles and infrastructures, where each sub-graph represents an object, and GNNs are used for extracting the interaction features [84]. It operates on vector representations, where all the agents' trajectories, lane lines, and crosswalks are represented by vectors. The fully-connected global graph used in the VectorNet is not efficient since the number of edges increases exponentially with the increasing number of nodes. EvolveGraph constructs a fully-connected graph with heterogeneous agent nodes and one context node for multi-agent trajectory prediction [87]. The same context node is linked to all target agents. However, we argue that each agent should have a local context.

CNNs are also widely explored for modeling either or both interaction and road geometry for trajectory prediction [58, 60, 70]. Convolutional social pooling (CS-LSTM) defines an occupancy grid around the target vehicle, where the cell occupied by a vehicle contains the vehicle's dynamics feature encoded using Long Short-Term Memory (LSTM) networks, and applies a CNN to the grid to extract interaction features without considering road structures [58]. Multi-agent tensor fusion (MATF) uses CNNs to encode the static scene context from a Bird's Eye View (BEV) image and then fuses the scene context and multiple agents' dynamics features to predict their future trajectories [70]. The spatial structure is retained

in MATF. It considers vehicle-infrastructure interactions, but the infrastructure is limited to a BEV map. Thus, it is hard to extend the MATF to consider other infrastructures in the future.

This work focuses on vehicle trajectory prediction. Surrounding vehicles, as well as infrastructures, are jointly considered. Assuming that the information of vehicles' dynamic states and the local map are available, we propose a novel interaction-aware trajectory prediction framework with a heterogeneous graph social (HGS) pooling method. To represent the scene in a unified and efficient way, we construct a star-like heterogeneous graph with an additional map node containing the target vehicle's local map feature. This representation is agnostic to map formats, and the map feature can be obtained from either a BEV map or a high-definition map (HD map). The star-like graph is sparse, and the number of edges increases linearly with the number of nodes. As shown in Fig. 3.1, the proposed framework shares the vehicle dynamics encoder across all vehicles to extract their individual features. Then, a rotation-sensitive convolutional network is designed to obtain the map feature, and a target-centered graph is constructed to represent vehicle-vehicle and vehicle-infrastructure relationships. Further, a novel heterogeneous graph social (HGS) pooling mechanism, which can handle an arbitrary number of vehicles, is proposed to explore and leverage vehicle-infrastructure interactions for prediction. Within the proposed framework, we consider the historical dynamics of the vehicles and the local BEV map. Other information, such as HD maps, can also be included in the designed structure if available.

Compared to CS-LSTM [58], which is designed for highway driving, HGS represents the driving scene with a unified heterogeneous graph containing an additional map node for complex urban driving. Compared to existing works for urban scenarios [70, 109] that highly rely on CNN extracted map features for spatial alignment between temporal and spatial features, HGS is agnostic to map representation as long as the spatial feature is provided for the map node.

The main contributions of this work are summarized as follows: 1) Jointly considering the temporal features of individual vehicles, the structure of the driving scenario, and their interactive relationships, a novel spatial-temporal trajectory prediction framework is proposed for urban driving scenarios; 2) Under the proposed framework, a heterogeneous graph social pooling module is developed to

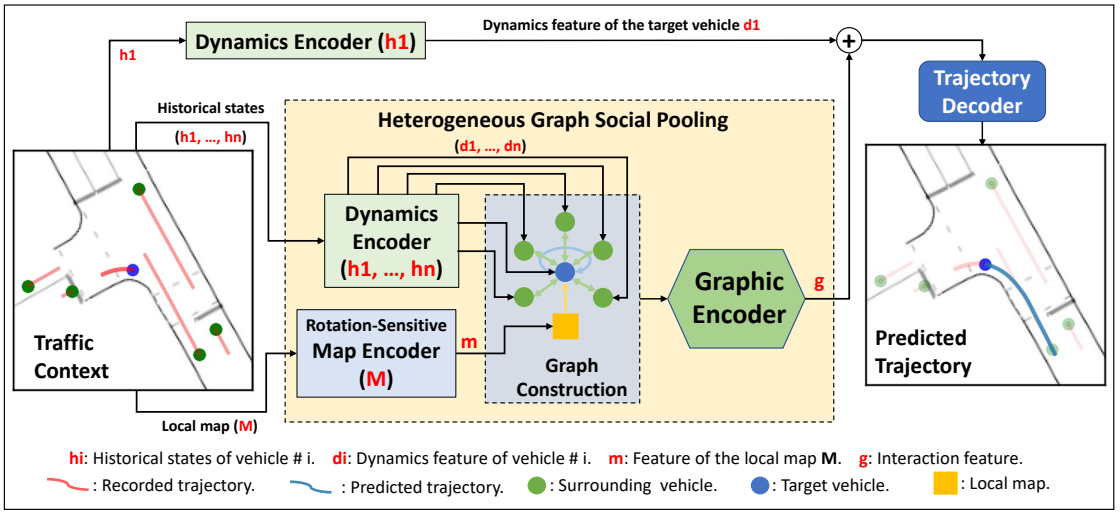

FIGURE 3.1: **The proposed scheme.** It consists of three encoders: a temporal dynamics encoder for vehicle dynamics feature extraction, a rotation-sensitive map encoder for road structure information, and a heterogeneous graph social pooling for obtaining high-level heterogeneous interaction features. The encoded dynamics and interaction features are then concatenated and sent to the trajectory decoder for trajectory prediction.

model high-level interactions among vehicles and the road. It can handle various driving situations with a varying number of interacting vehicles.

The remaining sections of this work are structured as follows. Sec. 3.2 elaborates on the proposed methodology and algorithms. Sec. 3.3 illustrates the setup of the experimental validation. Sec. 3.4 shows the experimental results. Sec. 3.5 shows how the proposed HGS model can be downgraded for highway driving scenarios. Sec. 3.6 provides further discussions on the results and the proposed method. Sec. 3.7 concludes this study and outlines future works.

3.2 Methodology

In this section, the problem is first formulated. Then the high-level architecture of the proposed trajectory prediction framework using heterogeneous graph social pooling is introduced in detail.

3.2.1 Input and Output

The trajectory prediction task of this work is to predict the future trajectory of a target vehicle for urban driving. Prediction is conducted based on the target vehicle's interactions with other surrounding traffic participants and the road structure. Both temporal and spatial information is considered.

The input $\mathbf{X}_t$ to the model consists of two parts: historical states of all considered vehicles and a local map.

$$\mathbf{X}_t = [\mathcal{H}_t, \mathcal{M}_t], \tag{3.1}$$

where $\mathcal{H}_t = \{h_t^1, \ldots, h_t^n\}$ represents historical states of n vehicles at current time t, and $h_t^i = [s_{t-T_h+1}^i, s_{t-T_h+2}^i, \ldots, s_t^i]$ represents vehicle i's ($i = 1$ for target vehicle) historical states at time t. T_h is the traceback horizon. The number of considered vehicles n may vary from case to case. $\mathcal{M}_t$ is the local map centered at the position of the target vehicle at time t. The number of considered agents n depends on the radius of the considered area and the real-time traffic density. When the radius is fixed, n is the number of agents whose distance to the target agent is below a certain threshold. The number n can be determined by the different graph construction strategies. Because a graph can always accommodate an arbitrary number of objects, and GNNs can process all nodes in parallel, the graph representation can always consider a variable number of agents, no matter how the graph connection strategy is.

The output is the predicted trajectory of the target vehicle:

$$f_t^1 = [(x_{t+1}^1, y_{t+1}^1), (x_{t+2}^1, y_{t+2}^1), \ldots, (x_{t+T_f}^1, y_{t+T_f}^1)], \tag{3.2}$$

where T_f is the prediction horizon.

3.2.2 Interaction-Aware Trajectory Prediction with Heterogeneous Graph Social Pooling

The proposed interaction-aware trajectory prediction framework, as shown in Fig. 3.1, is comprised of shared temporal dynamics encoders, a rotation-sensitive map encoder, a heterogeneous graph social pooling block, and a trajectory decoder. Dynamics encoders are applied to process the historical states of individual vehicles,

and the rotation-sensitive map encoder is adopted to process the local map for the spatial feature. Then vehicle dynamics and road structure features are jointly represented in a target-centered heterogeneous graph, and the heterogeneous graph social pooling extracts high-level interaction features from the constructed graph. Finally, the encoded dynamics and interaction features are fed into the trajectory decoder for prediction. These four components are elaborated on below.

3.2.2.1 Temporal Dynamics Encoder

This work adopts Recurrent Neural Networks (RNNs) to encode vehicles' dynamics features from their past states since a vehicle's past states compose a sequence with temporal dependence. RNNs, like LSTM [66] and Gated Recurrent Unit (GRU) [110], have been proven to be capable of modeling sequences in many challenging tasks, such as statistical machine translation, where the inputs and outputs are sequences with different lengths. Note that the selection of dynamics encoders may affect the prediction accuracy, but the proposed framework is agnostic to sequence models.

A shared temporal dynamics encoder is applied to each vehicle to capture their sequential features from historical states. Historical states, which are not restricted to the positions in the XY coordinates, can be extended with other available information, such as the vehicles' velocities and yaw angles. The extendability of the proposed model is further demonstrated in Sub.Sec. 3.4.2. Eq. 3.3 shows the dynamics encoder that is applied to process the target vehicle's historical states.

$$d_t^1 = \mathrm{FC}_1(\mathrm{Dyn}_{\mathrm{enc}}(\mathrm{Emb}(h_t^1))), \tag{3.3}$$

where Emb() is a linear function embedding low dimensional inputs into a higher dimensional space. $\mathrm{Dyn}_{\mathrm{enc}}$ is the RNN used in this work for dynamics feature extracting. FC_1 is a fully connected layer applied to the RNN-encoded feature. d_t^1 is the dynamics feature of the target vehicle. $\mathrm{Dyn}_{\mathrm{enc}}$ here is not restricted to a specific form in this work and can be implemented using LSTM, GRU, or other state-of-the-art RNNs. In other words, the dynamics encoder of the proposed model can be updated with the latest types of sequence models. Vehicles' dynamics features also serve as vehicle-node features in the spatio-temporal interaction graph (See Fig. 3.2).

3.2.2.2 Rotation-Sensitive Map Encoder

A map, which can be either a high-definition (HD) map or an image one, provides the road structure information and formulates vehicles' driving behaviors in the scene so that it should be considered in the trajectory prediction, especially for urban situations. In this work, we choose a pictorial BEV map because it is easy to get and contains enough road information. Specifically, a gray-scale image of the local map centered at the target vehicle is used for representing road information. A CNN is designed to extract the map features from the local map in the image. Even though many state-of-the-art CNNs have been widely used for image processing tasks, such as image classification [111] and object detection [112], and their pre-trained models can be easily obtained, we argue that they are not suitable for map processing. The max-pooling layer used in most CNNs covers the rotation invariance of images for image classification. It is useful for image classification tasks, as a rotated object in an image is still the object without any change in the classification result. However, for driving, a map also contains directional information, and a rotated map may mislead a vehicle's navigation. Instead of using pre-trained existing CNNs, we design a new CNN without max-pooling layers for map feature extraction. The detailed structure of the designed rotation-sensitive map encoder is given in Sub.Sec. 3.3.3.

Eq. 3.4 shows the map encoder applied to the local map at time t.

$$m_t = \mathrm{FC}_2(\mathrm{Map}_{\mathrm{enc}}(\mathcal{M}_t)), \tag{3.4}$$

where $\mathrm{Map}_{\mathrm{enc}}$ is the designed map encoder applied to the map image. FC_2 is a fully connected layer applied to the encoded raw map feature. And m_t is the feature vector of the local map. The size of m_t is the same as d_t^i. The map feature serves as a map-node feature in the spatio-temporal interaction graph (Fig. 3.2).

3.2.2.3 Heterogeneous Graph Social (HGS) Pooling

In this work, vehicle-vehicle and vehicle-infrastructure relationships are represented as a directed heterogeneous graph. In addition, a heterogeneous graph social (HGS) pooling block is designed for interaction modeling. In the constructed interaction graph, each node represents either a vehicle or the map, and an edge from one

node to another represents the directed influence from the former onto the latter. This representation is selected considering that: 1) Graphs can naturally represent objects (as nodes) and their relationships (as edges), e.g., the social networks. 2) The graph representation can accommodate a variable number of objects and is applicable to traffic situations with a variable number of interactive vehicles. 3) The heterogeneous graph allows different types of objects, such as vehicles and roads, to be considered jointly.

Definition 1 (Directed Heterogeneous Graph)**.** A graph can be represented by $\mathbb{G} = (V, E)$, where $V = \{v_1, \ldots, v_N\}$ is the set of N nodes, and $E \subset V \times V$ is the set of edges. If the edge from node i to node j is different from the one from node j to node i, the graph can be seen as a directed graph. If there are several kinds of nodes in the graph, the graph is considered a heterogeneous graph.

One problem with using graphs for traffic representation is that the graph needs to be constructed for the task, and the structure of the graph matters. Consider two extreme cases: 1) A graph with only self-connections, where each node is only connected with itself, ignores the relationships between nodes. 2) A graph with all connections, where each node is connected to all nodes, includes unnecessary connections. The number of edges increases quadratically with the increasing number of nodes. It is more reasonable to construct the interaction graph with neighboring connections, where each node is connected to nodes in its neighborhood. Specifically, we construct a sparse graph with self-loops to represent vehicle-infrastructure interactions.

Target-centered interaction graph. To model interactions as a graph, without loss of generality, we take v_1 as the target vehicle, v_N as the map node, and $\{v_2, \ldots, v_{N-1}\}$ as the neighboring vehicles. Then the edge set is determined as:

$$E = \{e_{1,j}\}_{(j=1,\ldots,N-1)} \cup \{e_{i,1}\}_{(i=1,\ldots,N)}, \tag{3.5}$$

where $e_{i,j}$ represents the directed edge from node i to node j. An example of the constructed graph, which contains two kinds of nodes: the *vehicle node* and the *map node*, is shown in Fig. 3.2.

A vehicle node in the graph contains the sequential feature of the corresponding vehicle obtained from the dynamics encoder, and a map node contains the spatial

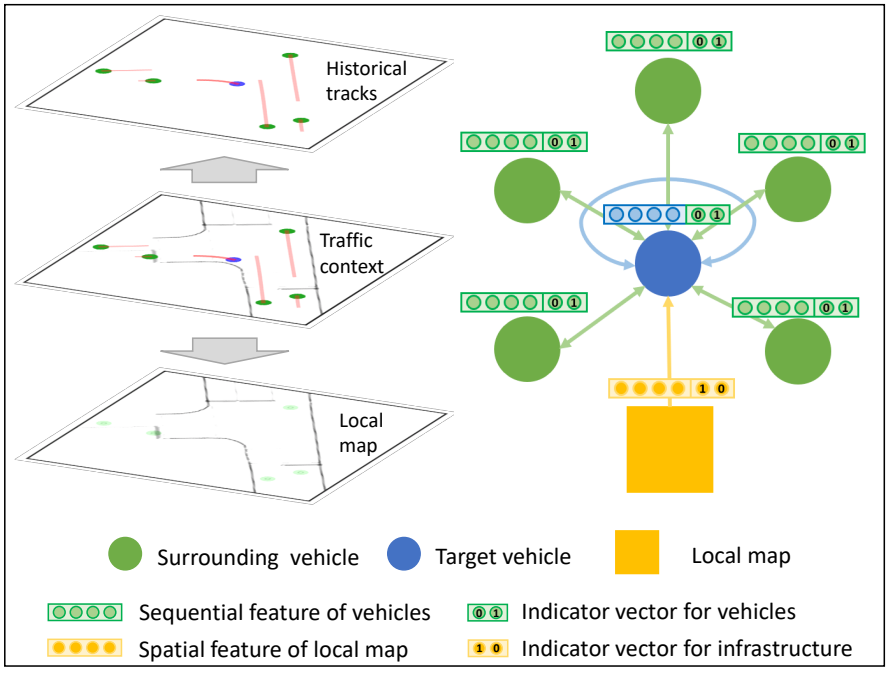

FIGURE 3.2: **Target-centered interaction graph.** *Left*, the traffic context is split into two parts: vehicles' historical states and the local map. *Right*, the target-centered heterogeneous graph is constructed with local connections.

feature of the local map extracted from the map encoder. Specifically, a vehicle is selected as the target vehicle's neighbor if its distance to the target vehicle is within 20 meters. The local map is set as a $40 \times 40m^2$ square, centered at the target vehicle. Node features are concatenated with an indicator one-hot vector ([0,1] for the vehicle nodes and [1,0] for the road node) to handle the heterogeneity.

The graph represented interaction can be processed by numerous GNNs to extract the interaction features. Compared to traditional graph methods, GNNs are designed to apply neural networks to tasks with graph-like inputs, such as social network prediction and protein interface prediction, and they show better performance in many tasks with graph data [79, 113, 114]. GNNs encode interactions among nodes into a feature vector by integrating information from variable-size neighborhoods.

Under the proposed framework, leveraging the power of GNNs, a heterogeneous graph social (HGS) pooling block is designed to extract the high-level interactions among vehicles and the road from the constructed spatio-temporal graph. HGS is agnostic to the selection of graphic encoders. The calculation of the HGS block is

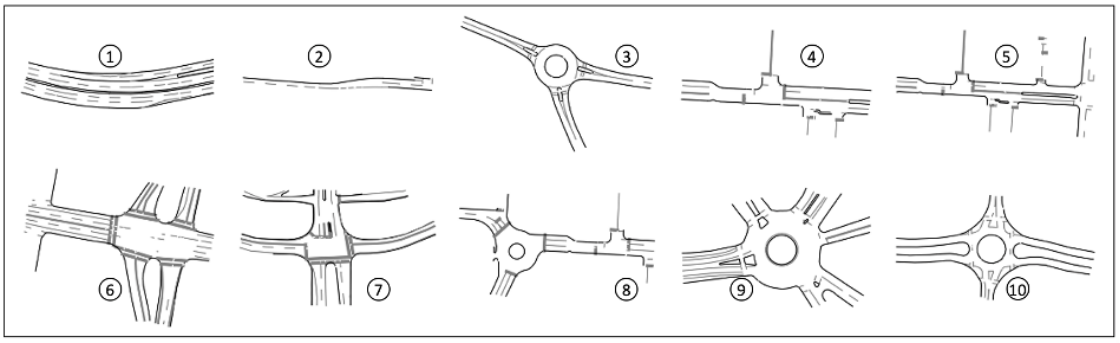

FIGURE 3.3: **The interactive driving scenarios [20].** The driving scenarios considered in this work include roundabouts (3, 8, 9, 10), unsignalized intersections (4, 5, 6, 7), and highway ramps (1, 2).

shown in Eq. 3.6 and Eq. 3.7.

$$D_t = \mathrm{FC}_1(\mathrm{Dyn}_{\mathrm{enc}}(\mathrm{Emb}(\mathcal{H}_t))), \tag{3.6}$$

$$g_t = \mathrm{FC}_3(\mathrm{Inter}_{\mathrm{enc}}(D_t, m_t, E_t)), \tag{3.7}$$

where D_t is the dynamics features of all vehicles, and E_t indicates the edge set at time t. $\mathrm{Inter}_{\mathrm{enc}}$ denotes the GNN used for interaction. FC_3 represents the fully connected layer applied to the raw feature extracted out from $\mathrm{Inter}_{\mathrm{enc}}$, and g_t is the target vehicle's interaction feature.

3.2.2.4 Trajectory Decoder

Finally, an RNN, rather than a multilayer perceptron (MLP), is applied to the concatenation of the target vehicle's dynamics and interaction features for trajectory prediction. The RNN is adopted here as it inherently handles the sequential dependence in a trajectory (the next position of a vehicle is partially dependent on its previous positions), while the MLP treats contiguous positions independently in prediction. Eq. 3.8 shows the calculation of the trajectory decoder.

$$f_t^1 = \mathrm{FC}_4(\mathrm{Traj}_{\mathrm{dec}}([g_t \| d_t^1])), \tag{3.8}$$

where $[g_t \| d_t^1]$ is the concatenation of g_t and d_t^1. $\mathrm{Traj}_{\mathrm{dec}}$ is the trajectory decoder for prediction, and FC_4 is the fully connected layer mapping decoded features to proper outputs. In this work, the output is a sequence of XY coordinates.

3.3 Experimental Setup

In this section, the dataset (3.3.1), evaluation metrics (3.3.2), and implementation details (3.3.3) for experiments are introduced.

3.3.1 Data

3.3.1.1 Dataset

The proposed scheme is trained and validated using the recently published INTERACTION dataset [20]. It consists of various highly interactive driving situations, including highway ramps, roundabouts, and intersections, recorded worldwide using drones and fixed cameras. For each recorded scenario, it provides a high-definition (HD) map and vehicle and pedestrian/bicyclist tracks.

As shown in Fig. 3.3, ten driving scenarios, as suggested by the online benchmark, are considered in this work. The whole dataset is split into training and validation sets, as suggested by authors of the INTERACTION dataset. The trajectories are split into segments of 10 seconds, where five seconds are saved as historical states, and the following five seconds are considered the ground truth. The gap between the two segments is one second. After the pre-processing stage described in the following paragraph, the training set contains $317,335$ pieces of data, and the validation set contains $90,219$ pieces of data.

3.3.1.2 Data Processing

We use a stationary frame of reference with the origin fixed at the target vehicle's position at time t for all trajectories. The velocities remain the same as their recorded values. The yaw angle of the target vehicle at time t is set as zero, and other values are changed accordingly. The local map is a $40 \times 40 m^2$ square centered at the target vehicle, and it is kept parallel to the vehicle's current direction. It is represented by a 160×160 array.

3.3.2 Evaluation Metrics

In this work, the performance of the prediction method is evaluated using three metrics, i.e., the Displacement Error at time τ (DE_τ), the Average Displacement Error (ADE) over trajectories, and the Final Displacement Error (FDE). They are all measured in meters. The lower, the better. DE_τ, ADE, and FDE between two trajectories can be calculated by Eq. 3.9, Eq. 3.10, Eq. 3.11, respectively.

$$DE_\tau = \sqrt{(\hat{x}_\tau - x_\tau)^2 + (\hat{y}_\tau - y_\tau)^2}, \tag{3.9}$$

$$ADE = \frac{1}{T_f}\sum_{\tau=1}^{T_f} DE_\tau = \frac{1}{T_f}\sum_{\tau=1}^{T_f} \sqrt{(\hat{x}_\tau - x_\tau)^2 + (\hat{y}_\tau - y_\tau)^2}, \tag{3.10}$$

$$FDE = DE_{T_f} = \sqrt{(\hat{x}_{T_f} - x_{T_f})^2 + (\hat{y}_{T_f} - y_{T_f})^2}, \tag{3.11}$$

where $(\hat{x}_\tau, \hat{y}_\tau)$ is the predicted position in the XY coordinate at time τ, (x_τ, y_τ) is the ground truth of the position at time τ, and T_f is the prediction horizon.

3.3.3 Implementation Details

In this work, the algorithms are implemented using PyTorch [115] for the overall structure and PyTorch Geometric [116] for GNN layers, respectively. The model is trained in an end-to-end manner for ten epochs using Adam [117] with a scheduled learning rate, which starts from 0.001 and reduces by half at the end of the epochs 1,2,4, and 6 for minimizing ADE as defined in Eq. 3.10. The historical states are firstly embedded into a 64-dimension space and then sent to the temporal dynamics encoder. The dynamics encoder is a one-layer RNN with a hidden size of 64, and the trajectory decoder is a two-layer RNN with a hidden size of 128. The rotation-sensitive map encoder is a three-layer convolutional block with $[\#\_filters, kernel\_size, stride] = [8, 16, 4]$ for the first layer, $[16, 8, 4]$ for the second layer, and $[32, 4, 2]$ for the third layer. The structure is [[Conv, LeakyReLU, BatchNorm], [Conv, LeakyReLU, BatchNorm], [Conv, LeakyReLU, BatchNorm], FC, FC]. Max-pooling layers after convolutional layers are all removed to reserve direction information in the map image. This is designed specifically for the trajectory prediction task. The heterogeneous graph social pooling block uses two GNN

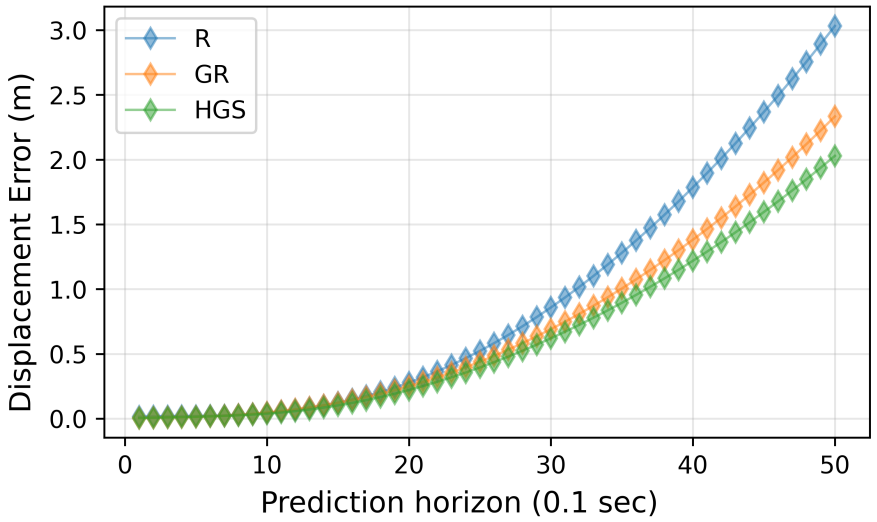

FIGURE 3.4: **Results of the ablation study.** The ablation study takes (x, y, v_x, v_y, ψ) as input states. Traceback horizon: $T_h = 30$ (3 sec). Prediction horizon: $T_f = 50$ (5 sec). Graphic encoder: GAT. Dynamics encoder: GRU.

layers to extract the higher-level interaction features. Leaky-ReLU with a negative slope of 0.1 is the only activation function throughout the implementation.

3.4 Results

In this section, we first investigate the proposed method via ablation study (3.4.1) and then show the extendibility and renewability of HGS (3.4.2). Next, we study the impacts of the traceback horizon on the prediction performance (3.4.3) followed by an inference time study (3.4.4). Finally, we compare HGS with existing methods by submitting it to test sets of two naturalistic driving datasets (3.4.5).

3.4.1 Ablation Study

The ablation study is conducted to validate the effectiveness and superiority of the proposed approach. The following methods are implemented and compared:

- **R**. This model predicts the future trajectory of a vehicle using only its individual dynamics feature extracted by a single dynamics encoder. No interaction is considered in this model.

- **GR**. This model leverages a homogeneous graph to reflect the interactions between vehicles, where each node represents a vehicle. It considers the interactions among vehicles but ignores the effects of infrastructures.
- **HGS**. This is an implementation of the proposed scheme HGS, which models the interactions among vehicles and the infrastructure via a heterogeneous graph.

All the models use vehicles positions ((x, y)), velocities ((v_x, v_y)), and yaw angles (ψ) for historical states.

The results are shown in Fig. 3.4. It can be observed that the interaction-aware models (GR and HGS) outperform the non-interaction-aware one (R). This observation is consistent with the results reported in previous works [58, 60], demonstrating the necessity of modeling interactions for trajectory prediction. In addition, the proposed HGS outperforms the homogeneous GR method. This result shows the advances of the proposed method: 1) modeling vehicle-infrastructure interactions as a heterogeneous graph, 2) obtaining features of different nodes with different encoders, and 3) leveraging graphic encoder to extract high-level interaction features. In this work, we consider two kinds of nodes for demonstration, while other types of nodes can be further considered with the proposed framework in the future.

3.4.2 Extendability and Renewability of HGS

The proposed scheme models interactions among vehicles and infrastructures as a heterogeneous graph. There is no limitation on input types. In this work, the heterogeneous graph contains two types of nodes, i.e., vehicle nodes and the road node. It can be further extended by introducing other kinds of nodes to take additional traffic infrastructures, such as traffic signals, and other agents, such as pedestrians and bicyclists, into consideration.

For the dynamics encoder, the proposed scheme does not impose any restriction on the information to be considered. It can be easily extended with the introduction of richer information if available. To illustrate this, the proposed model is implemented with three different inputs, (x, y), (x, y, v_x, v_y), and (x, y, v_x, v_y, ψ). The results are shown in Fig. 3.5. It is shown that the method, which considers

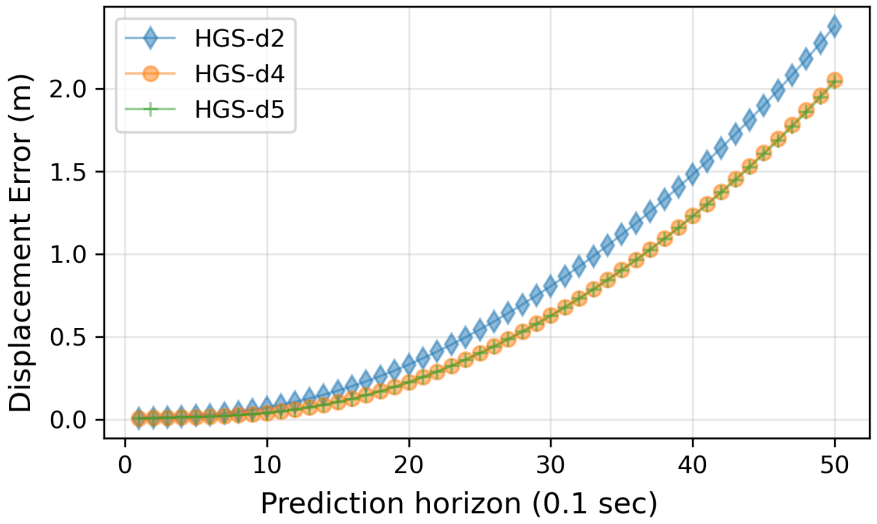

FIGURE 3.5: **Extendability of the proposed scheme.** HGS-d2: the proposed model HGS using (x, y) for historical states; HGS-d4: HGS using (x, y, v_x, v_y) for historical states; HGS-d5: HGS using (x, y, v_x, v_y, ψ) for historical states. Traceback horizon: $T_h = 30$ (3 sec). Prediction horizon $T_f = 50$ (5 sec). Graphic encoder: GAT. Dynamics encoder: GRU.

velocities in addition to the positions, further reduces the displacement error, but an additional consideration of the yaw angle does not show noticeable improvement. Please note that the scheme implemented in this section takes A Graph Attention Network (GAT) as the graphic encoder in the HGS block and GRU as the dynamics encoder, respectively.

In this work, a rotation-sensitive CNN is used to extract spatial features from a top-view image of the local map. The top view image can be replaced by a vectorized representation of an HD map to further reduce the model size, as stated in [84].

The main modules of the proposed framework are decoupled from each other so that they can be updated without affecting other modules. The proposed model has no assumption on what recurrent unit to be used for the dynamics encoder. It could be LSTM [66], GRU [110], or others. In this study, we apply a graphic encoder to extract the interactions among vehicles and infrastructures for the constructed graph with corresponding features. The graphic encoder could be Graph Convolutional Network (GCNs) [78], GATs [79], or other user-designed graphic encoders.

To show that HGS can be easily renewed, we implement the proposed scheme with different dynamics and graphic encoders. Besides, the performances of their

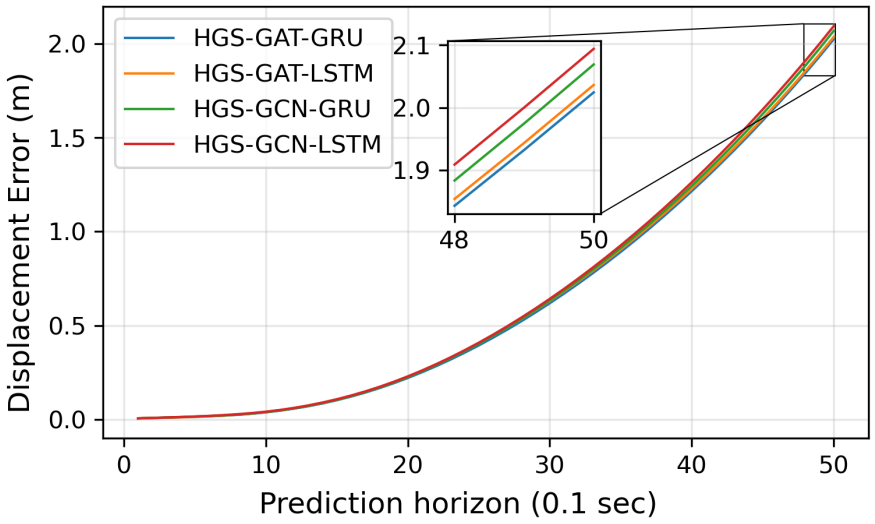

FIGURE 3.6: **Renewability of the proposed scheme.** HGS-GAT-GRU: the proposed HGS using GAT as the graphic encoder and GRU as the dynamics encoder. The same rule applies to HGS-GAT-LSTM, HGS-GCN-GRU, and HGS-GCN-LSTM. Traceback horizon: $T_h = 30$ (3 sec). Prediction horizon: $T_f = 50$ (5 sec). Input data: (x, y, v_x, v_y)

combinations are also reported. GCN is selected because of its simplicity and effectiveness, and GAT is selected as it treats neighboring nodes differently using an attention mechanism. The results shown in Fig. 3.6 show that GAT-based methods slightly outperform GCN-based methods, and the combination of GAT and GRU achieves the best performance on trajectory prediction. It is worth noting that results in Fig. 3.6 show that HGS can be updated with different encoders, but the difference in our experiments is not significant. We leave implementing HGS with other temporal and graphic operators for future work.

3.4.3 Impacts of Traceback Horizon

In this section, we further study how the traceback horizon affects the prediction accuracy of the proposed scheme and its ablations. The traceback horizons (T_h) are set to three values $(10, 30, 50)$ and studied with the input data (x, y, v_x, v_y). Fig. 3.7 shows that prolonging T_h from 10 to 30 leads to better performance for R and GR, while further prolonging T_h to 50 shows very limited improvement. For HGS, the differences among different traceback horizons are minor. Fig. 3.7 also shows that the effects of traceback horizons vary under different methods. Compared to its ablations, the proposed framework shows better performance with a very

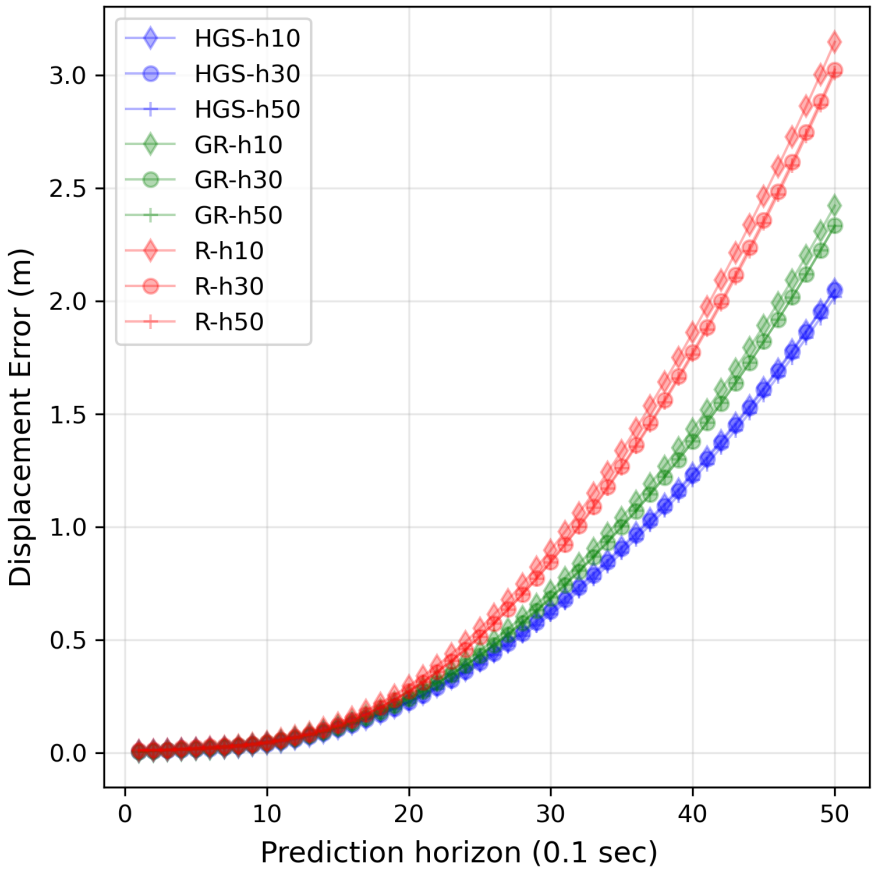

FIGURE 3.7: **Impacts of traceback horizon on prediction performance.** Traceback horizon: $T_h = 10, 30, 50$ (1, 3, 5 sec). Prediction horizon: $T_f = 50$ (5 sec). Input data: (x, y, v_x, v_y). Graphic encoder: GAT. Dynamics encoder: GRU.

short traceback horizon. It improves not only the accuracy but also the data and computation efficiency. This is because obtaining data with a longer traceback horizon is more difficult and needs more computing effort.

3.4.4 Computational Complexity

Computational complexity is an important aspect of prediction methods for on-board implementation, especially on platforms with limited computing resources (e.g., only CPU is available). We run HGS and its ablations with different batch sizes (1 or 100) and traceback horizons ($T_h = 10, 30, 50$) for 1,000 (1K) batches to see the inference time on a CPU (Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz). The input data consists of sequences of (x, y, v_x, v_y). The results are listed in Tab. 3.1. It can be seen that larger T_h always leads to a longer inference time for

TABLE 3.1: Computational complexity analysis

Model	No. data	T_h (ms)	Time (s)
R	1K×1	10	2.99
	1K×1	30	3.58
	1K×1	50	4.03
	1K×100	10	36.62
	1K×100	30	44.39
	1K×100	50	53.75
GR	1K×1	10	3.75
	1K×1	30	4.32
	1K×1	50	4.81
	1K×100	10	47.31
	1K×100	30	55.97
	1K×100	50	63.41
HGS	1K×1	10	4.28
	1K×1	30	4.87
	1K×1	50	5.33
	1K×100	10	52.89
	1K×100	30	61.19
	1K×100	50	70.59

all models. Compared to HGS with $T_h = 50$, HGS with $T_h = 10$ can save up to 25% of time without obvious accuracy loss.

3.4.5 Comparison with State-of-the-art Methods

To compare with existing works, we trained our scheme with data augmentation and submitted the results to the INTERACTION test set hosted online for competition. Finally, the proposed method (named GH_29_3 for submission) outperformed all participants and won first place in the INTERPRET Challenge (regular track).

We also compare our method with existing works on the Argoverse test set with an additional metric, miss rate (MR). MR is defined as the number of scenarios where none of the predicted trajectories are within 2.0 meters of ground truth according to endpoint error for up to $K = 6$ predicted trajectories. Since our method focuses on single trajectory prediction, we report only the results for $K = 1$. The test set is also hosted online and made invisible for easy and fair comparisons. It can be seen from Tab. 3.2 that our method outperforms all the baselines provided by the

TABLE 3.2: Comparison with state-of-the-art methods on the Argoverse test set

Methods	**Metrics (K=1)**		
	ADE (m)	FDE (m)	MR
Argo-(CV) [51]	3.53	7.89	0.83
Argo-(NN) [51]	3.45	7.88	0.87
Argo-(LSTM) [51]	2.15	4.97	0.75
Argo-(NN+map) [51]	3.65	8.12	0.94
Argo-(LSTM+map) [51]	2.92	6.45	0.98
Jean [89]	1.74	4.24	0.69
WIMP [106]	1.82	4.03	0.63
LaneGCN [21]	1.71	3.78	0.59
TNT [85]	2.17	4.96	0.71
mmTransformer [118]	1.77	4.00	0.62
PRIME [46]	1.91	3.82	0.59
Scene Transformer [90]	1.81	4.06	0.59
HGS	1.75	3.89	0.61

dataset holder [51] in all metrics. HGS also outperforms WIMP [106], TNT [85], mmTransformer [118] in terms of all three metrics. Compared to Jean [89] (winner of Argoverse motion challenge 2019), PRIME [46], LaneGCN [21] (winner of Argoverse motion challenge 2020), and Scene Transformer [90], HGS shows state-of-the-art performance. Results of Argoverse baselines are obtained from the supplementary file of [51]. Results of other methods are retrieved from the Argoverse challenges leaderboard on April 19(th), 2022.

3.5 Downgrading to Homogeneous Graph Social Pooling for Highway Driving

The proposed heterogeneous graph social (HGS) pooling can be downgraded to homogeneous graph social (HGS(homo)) pooling for highway driving with simple road structures. In this situation, all the nodes in the interaction graph are vehicles, i.e., they are homogeneous, so the interaction can be represented by a homogeneous graph. Fig. 3.8 shows the structure of HGS(homo). It considers two kinds of vehicles: the target vehicle and its neighboring vehicles. Neighboring vehicles considered are the target vehicle's preceding (#1) and following (#2) vehicles,

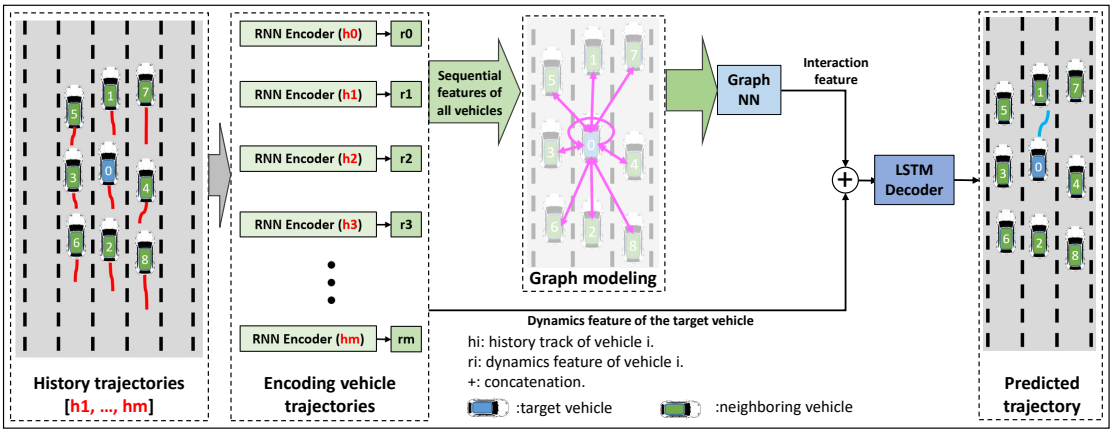

FIGURE 3.8: **Illustration of the homogeneous graph social (HGS(homo)) pooling method [1].** RNNs with shared weights are used to encode the dynamics features of vehicles individually. A GNN-based interaction encoder is applied to these dynamics features, which are contained in corresponding nodes in a directed interaction graph, to summarize the inter-vehicular interaction feature. Finally, an LSTM decoder predicts the trajectory by jointly considering the target vehicle's dynamics and interaction features.

its nearest neighbors in adjacent lanes (#3 and #4) in terms of longitudinal distance, and their preceding (#5 and #7) and following (#6 and #8) vehicles. The input to the model ($\mathcal{H}_t$) is a set of historical trajectories of all considered vehicles, including the target vehicle. The output is the predicted future trajectory of the target vehicle at time t. The HGS(homo) model contains the temporal dynamics encoder (3.2.2.1) and the trajectory decoder (3.2.2.4) as HGS. It replaces the heterogeneous graph social pooling (3.2.2.3 with a homogeneous graph social pooling. The map encoder (3.2.2.2) is excluded because of the highway's simple structure. The interaction graph is a directed homogeneous graph with nodes representing vehicles.

We process vehicle trajectories extracted from the publicly available NGSIM US-101 [47] dataset, collected from 7:50 a.m. to 8:35 a.m. on June 15, 2005, for training and validation. The study area is a 640 meters segment of U.S. Highway 101, consisting of five main lanes, one auxiliary lane, on-ramp, and off-ramp lanes. The vehicle trajectory data are recorded at 10 Hz using eight synchronized digital video cameras mounted on the top of a 36-story building. This work selects roughly balanced data so that the lane-keeping trajectories do not dominate the dataset. For details on data processing, please refer to [1]. The source codes for this section are released and can be found in Appendix-C.

We use the root-mean-square error (RMSE) in meters of the predicted trajectories against the ground truth future trajectories to evaluate different models. RMSE is calculated for each predictive time-step t_p within 5 seconds in the future. Previous works [58, 60, 81] also adopt this metric.

$$RMSE(t_p) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}((\hat{x}_{t_p}^i - x_{t_p}^i)^2 + (\hat{y}_{t_p}^i - y_{t_p}^i)^2)}, \tag{3.12}$$

where $n = 10000$ is the size of test set, and $(\hat{x}_{t_p}^i, \hat{y}_{t_p}^i)$ and $(x_{t_p}^i, y_{t_p}^i)$ are the predicted position of the target vehicle in data i at time t_p and the corresponding ground truth, respectively.

The following methods are implemented for comparison:

- **Dynamics-only**: this is the one-channel ablation of the proposed model considering the target vehicle's dynamics feature only for prediction.
- **Interaction-only**: this is another one-channel ablation using the interaction feature extracted by the GNN only.
- **Two-channel**: this is the proposed two-channel model.

The above implementations are trained and validated using the same dataset.

Prediction results of HGS(homo) are reported in Tab. 3.3, along with results reported in some related works. However, this section focuses on comparing results between the proposed HGS(homo) and its ablations, considering that different works are using different training and validation datasets. Tab. 3.3 shows that:

- Interaction-aware methods (2,3,4,5,6) outperform the dynamics-only method (1). This demonstrates the necessity of modeling interactions for trajectory prediction as stated in previous works [58, 60].
- The proposed two-channel model outperforms its interaction-only ablation. This shows that the target vehicle's dynamics feature should be emphasized in some way for trajectory prediction. This work sets an additional channel for it.

- The proposed method matches the CNN-LSTM method with advances in considering a variable number of surrounding agents and the potential for multi-trajectory prediction.
- The proposed method outperforms GRIP, SAMMP, and CS-LSTM in longer-term prediction. However, for the short-term prediction, GRIP shows better performance, possibly in that GRIP uses the whole dataset from NGSIM, where the lane-keeping trajectories are dominant and less challenging for trajectory prediction.

Please note that most existing works listed in Tab. 3.2 are not shown in Tab. 3.3 because they are heavily dependent on the road map, which is not available in the highway dataset used by methods in Tab. 3.3.

TABLE 3.3: Prediction performance comparison (RMSE in meters) [1]

	Methods	**Prediction horizon**				
		1 sec	***2 sec***	***3 sec***	***4 sec***	***5 sec***
1	Dynamics-only (Ours)	0.74	1.86	3.30	5.07	7.11
2	Interaction-only (Ours)	0.67	1.03	1.34	1.74	2.46
3	Two-channel (Ours)	0.68	0.99	1.21	1.53	2.14
4	CS-LSTM [58]	0.61	1.27	2.09	3.10	4.37
5	SAMMP [89]	0.51	1.13	1.88	2.81	3.98
6	GRIP [80]	0.37	0.86	1.45	2.21	3.16
7	CNN-LSTM [60]	0.64	0.96	1.22	1.53	2.09

Fig. 3.9 visualizes prediction results in situations with different numbers of surrounding vehicles from the validation set. It shows that the proposed model can predict the target vehicle is going to keep or change lanes in the next 5 seconds regardless of how many surrounding vehicles are in sight.

3.6 Discussions

In this study, a unified interaction-aware trajectory prediction scheme for autonomous driving is proposed and validated on real-world datasets collected from

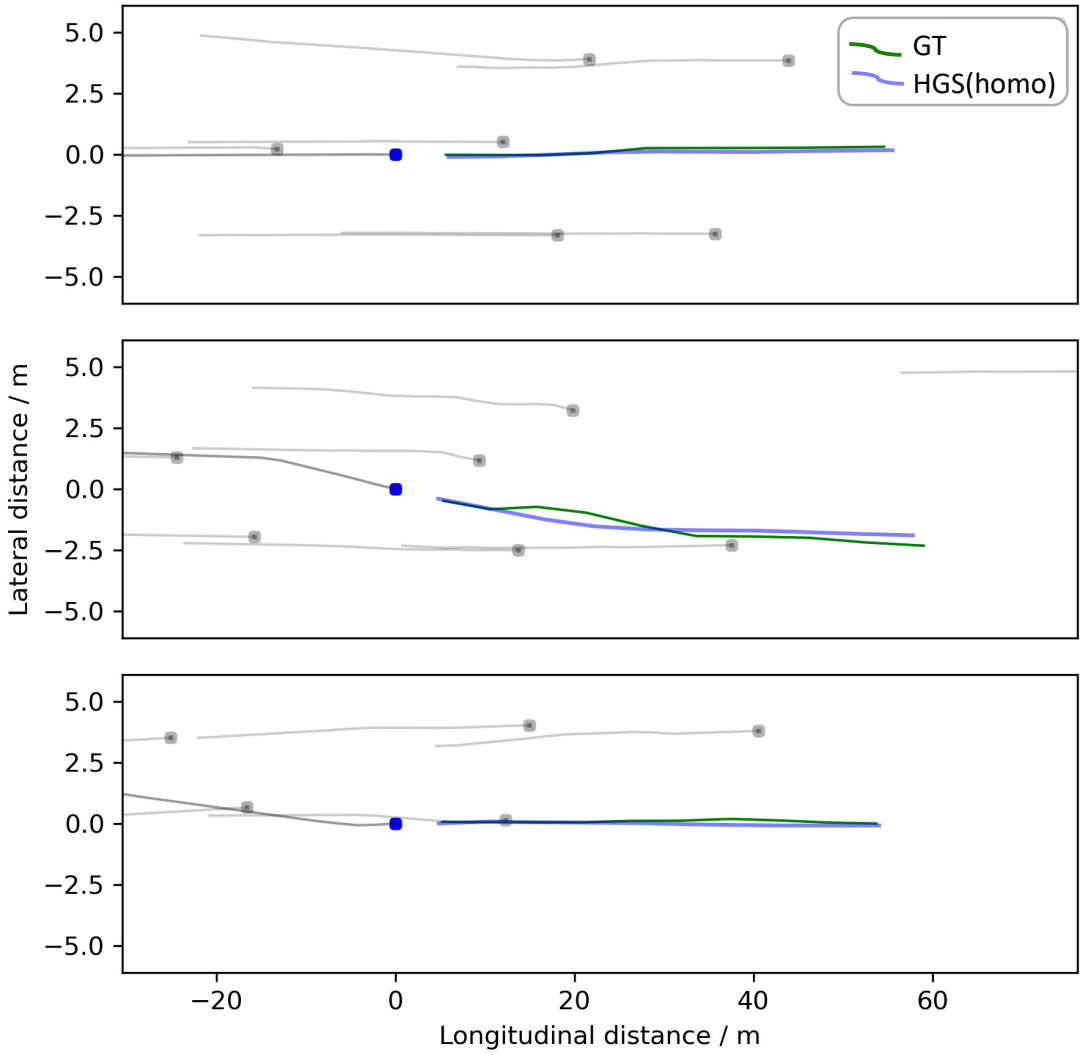

FIGURE 3.9: **Visualized predictions on highways [1].** Squares are considered vehicles (the target vehicle in blue and neighboring vehicles in gray). Gray lines are the vehicles' historical tracks over the last 3 seconds. The green line is the ground truth (GT) future trajectory of the target vehicle. The blue line is the prediction of the proposed two-channel model (HGS(homo)). All the vehicles move from left to right.

different cities. Experiments show that the proposed method achieves better performance by considering map information in heterogeneous graph social (HGS) pooling. The proposed HGS is an extension to convolutional social pooling (CS-LSTM) [58]. The extensions are twofold. First, HGS designs a rotation-sensitive map encoder to take into consideration the map structure, which is essential for urban driving under different situations, and models the interaction between traffic participants and the map. However, CS-LSTM ignores road structure because that

road is almost straight in highway scenarios. Second, HGS represents the interactions between traffic participants and the map in a unified heterogeneous graph that can accommodate an arbitrary number of traffic participants, while CS-LSTM represents inter-vehicular interactions with an occupancy grid with thirty-nine cells.

Ablation study shows that considering the interaction between traffic participants and the road map is essential for trajectory prediction. The extendability examination shows that considering richer information does not always improve prediction accuracy. This sheds light on designing a predictor based on enough information rather than redundant information that requires more effort to collect and compute. The renewability demonstration shows that HGS can be implemented with different sequential and graphic models. Even though the difference in our experiment is not significant enough to conclude what sequential and graphic models are better for HGS implementation, HGS is open to being implemented with more advanced encoders. Traceback horizon investigation shows that HGS achieves better performance with a shorter traceback horizon than its ablations, and a longer traceback horizon does not always contribute to prediction accuracy. Inference time comparison confirms that processing data with a longer traceback horizon leads to more inference time. Traceback horizon and inference time investigations together provide instructions to design efficient predictors that require less data collection and computation efforts and provide similar accuracy.

The driving scene is represented by a heterogeneous graph and modeled via the proposed HGS for trajectory prediction. We would like to point out that the scene representation and modeling with HGS can be applied to other autonomous driving tasks that require scene modeling, such as driving intention recognition [119], driving behavior modeling [120], decision-making [121], and path planning [122].

One of the limitations of this work is the limited factors considered in scene representations due to data availability. For example, traffic signals and rules are not considered in this work because the data are collected from unsignalized road junctions. Another limitation is the deterministic prediction that ignores the uncertainty of driving behaviors. Uncertainties can be included in two ways in the future. First, a two-dimensional Gaussian distribution can be predicted for each time step in place of an averaged position. Second, the multimodality of driving can be modeled by predicting a number of possible trajectories of the target vehicle with estimated possibilities. This thesis addresses the inherent multimodality

of driving in a map-adaptive way in Chapter 5. This twofold uncertainty awareness can be realized based on HGS by using corresponding decoders and training losses [58, 59, 76]. Besides, predicting the trajectory of a single target agent is another limitation that should be addressed to speed up inference time. The next chapter (Chapter 4) generalizes the single prediction method to simultaneous multi-agent prediction.

3.7 Conclusions

In this study, a novel framework is proposed with consideration of vehicle-infrastructure heterogeneous interactions for vehicle trajectory prediction. The proposed scheme constructs a heterogeneous graph to represent the interactions, where the nodes contain features extracted from corresponding encoders. Besides, a novel heterogeneous graph social (HGS) pooling block is designed to extract high-level interaction features. The framework can be easily downgraded for highway driving scenarios. Experimental results obtained using real-world driving datasets show that the proposed HGS method outperforms existing interaction-aware methods. Ablative studies also show that considering vehicle-infrastructure heterogeneous interactions effectively improves the trajectory prediction accuracy compared to those considering inter-vehicular interactions only.

In the future, the proposed HGS method can be further improved from different aspects. One way is to consider a more informative HD map instead of a pictorial one. Other types of nodes, such as traffic signals, together with proper encoders, can also be involved in order to improve performance. Besides, the proposed method predicts the trajectory for only one target vehicle, which can be further generalized and expanded to simultaneous trajectory predictions for multiple vehicles to support the downstream decision-making and planning of connected automated vehicles. Other graph construction strategies can also be investigated to check their impacts on prediction performance.

Chapter 4

Multi-Agent Trajectory Prediction with Heterogeneous Edge Enhanced Graph Attention Networks

This chapter generalizes and expands the single-agent trajectory prediction framework developed in Chapter 3 to a multi-agent setting, where the predictor is expected to simultaneously predict future trajectories of a variable number of different traffic participants. A hybrid scene representation, which represents interactions among agents with a heterogeneous graph and the shared local map with an image, is adopted. A novel **H**eterogeneous **E**dge-enhanced graph **AT**tention network (HEAT) is proposed to model the interaction graph, and a map selector is designed to share the local map information across agents selectively[1].

4.1 Introduction

Most existing interaction-aware methods represent the motion of all agents in a shared coordinate system, which is sensitive to translation and rotation, and only aim at predicting the trajectory of a single agent [58, 60, 70, 83, 88]. However,

[1]The contents in this chapter are obtained from the published paper [71].

autonomous vehicles should simultaneously predict the future states of multiple surrounding agents, e.g., vehicles and pedestrians, to navigate in complex and highly dynamic urban driving scenarios.

This work focuses on simultaneously predicting future trajectories of multiple heterogeneous agents for both urban and highway driving by jointly considering agents' individual dynamics, their interactions, and the road structure. Agents' past states and a top view image of the area of interest are assumed to be available leveraging the vehicle-to-vehicle and vehicle-to-infrastructure communications [64].

Since a moving agent's future motion is affected by many factors, including but not limited to its own dynamics, social interactions, and the road structure, ideally, a trajectory predictor should consider as many of these associated factors as possible. Considering the availability of the datasets, we propose a three-channel framework for multi-agent trajectory prediction to handle these factors accordingly, enabling the modularized design and analysis. For agents' dynamics, we place each agent in its exclusive coordinate system to eliminate the impacts of coordinate shifting. This is because an agent's recorded states, no matter placed in which coordinate system, can always be converted to its own coordinate system without affecting other agents. For inter-agent interaction, we represent the interaction among agents as an edge-featured heterogeneous graph and design a novel heterogeneous edge-enhanced graph attention network to model the interaction among agents of different types. For the road structure, a pictorial map is shared across all agents with a gated map selector. Please see Fig. 4.1 for an overview of the proposed framework.

The main contributions of this work can be summarized as follows:

- A three-channel framework is proposed for multi-agent trajectory prediction. It jointly considers agents' individual dynamics, their interactions, and the road structure for trajectory prediction.
- A comprehensive and transformation-insensitive interaction representation is proposed based on the edge-featured heterogeneous graph, where the nodes and edges fall into different categories and contain corresponding attributes.

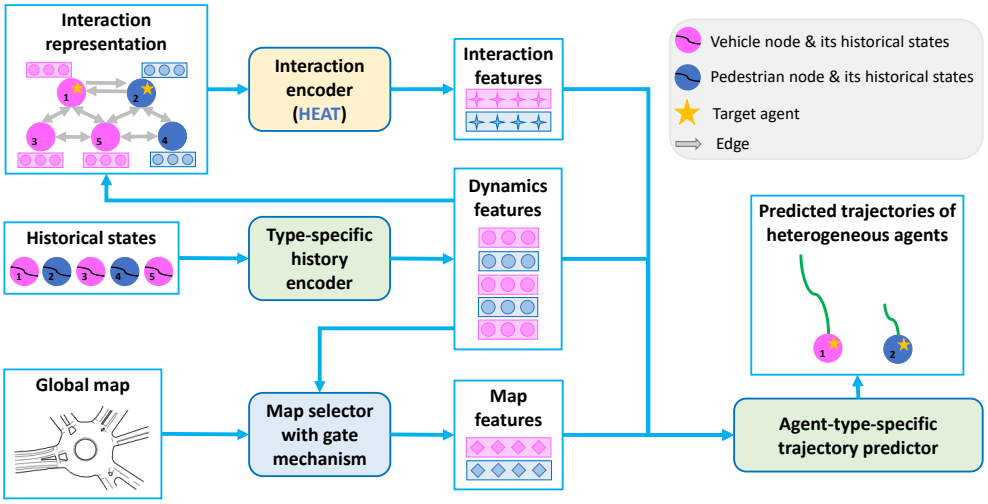

FIGURE 4.1: **Proposed multi-agent trajectory prediction framework.** Historical states of agents are encoded with type-specific history encoders to get their individual dynamics features. Inter-agent interaction is represented by an edge-featured heterogeneous graph with each node containing the dynamics feature of its corresponding agent. Then the proposed HEAT is applied to the interaction graph to extract interaction features for all agents in parallel. The map feature for an agent is processed with a gate mechanism by considering its dynamics on the map. Then features from these three channels are concatenated and fed to the agent-type-specific trajectory predictor to predict the future trajectories of all target agents.

- A novel heterogeneous edge-enhanced graph attention network (HEAT) is designed to model the inter-agent interaction for multi-agent trajectory prediction.
- A gate-based map selector is proposed to allow sharing the map information across all target agents in a selective manner rather than storing a local map for each agent or sharing the same map across all agents.

The remainder of this work is structured as follows: Sec. 4.2 introduces existing works most related to this work. Sec. 4.3 provides an overview of the proposed method. Sec. 4.4 elaborates on the proposed method and its key components. Sec. 4.5 validates the proposed method on real-world driving datasets collected from both urban and highway scenarios. Sec. 4.6 concludes this work and outlines possible future improvements.

4.2 Related Work

This section reviews interaction representation for interaction-aware trajectory prediction, various graph neural networks (GNNs) proposed for graph-based tasks, and how GNNs can be applied to trajectory prediction tasks. The distinction and advantages of the proposed interaction representation, graph neural network, and trajectory prediction framework are illustrated following each group of related works.

4.2.1 Interaction Representation

Interaction-aware trajectory prediction methods have employed many ways to represent inter-agent interactions recently. Convolutional social pooling designs an occupancy grid, where each cell contains the feature of the agent that falls in it, to model the interaction among agents in the grid [58]. The grid representation is modified in [60] to observe only the eight agents that mostly affect the target vehicle's behavior. The grid representation is applicable to highway driving since the highway is almost straight and can be easily divided into a grid. But this is not the case for urban driving. Therefore, to model interaction beyond highway driving, multi-agent tensor fusion (MATF) models the interaction by aligning agents' individual features to a top-view image of the driving scene [70]. However, it still ignores the relationships among agents. More and more recent works represent interactions as a graph, where each node represents an agent, and the edge represents the inter-agent relationship. Authors of [77] propose to represent the inter-vehicle interaction as a homogeneous directed graph for highway driving, where each vehicle is connected to up to eight of its neighbors. GRIP also uses a homogeneous graph to model the interaction [80]. The drawback of this kind of method is that the homogeneous graph ignores the type of agents. On the other hand, ReCoG proposes to represent the interaction as a heterogeneous graph, where a node represents either an agent or a map, and an agent is connected to other agents within a neighborhood [83]. ReCoG ignores the edge attributes between nodes. VectorNet [123] and TNT [124] both use a hierarchical heterogeneous graph to represent the interaction, where each object is represented by a sub-graph, and all the objects are then represented by a fully-connected graph. Nonetheless, these methods

fail to consider the edge attributes between nodes. SCALE-Net considers edge attributes and proposes to represent the interaction with an edge-featured homogeneous graph, where the edge feature contains relative states between two connected agents [81]. It ignores the heterogeneity of traffic participants. Social-WaGDAT proposes to generate a dynamic pair of history and future graphs for each time step, yet the nodes are assumed to be homogeneous and with a fixed number [88]. EvolveGraph learns an interaction graph that considers the heterogeneity of nodes and edges' types and directions [87]. However, the edge attribute is not considered.

Representing inter-agent interaction as a graph is more natural than using an image or a grid. However, most existing graph representations place all the agents on the same target-centered coordinate system, which is suitable for single-agent trajectory prediction but can hardly generalize to multi-agent situations because of the effects of coordinate translation and rotation. SCALE-Net places all agents in their own exclusive coordinate systems for generalization and uses edge attributes to preserve spatial relationships among agents [81]. However, the graph representation in SCALE-Net is not comprehensive to cover the heterogeneity of agents and their relationships for trajectory prediction. In this work, we propose to represent the inter-agent interaction in exclusive coordinate systems as a directed heterogeneous edge-featured graph, where different agents are represented by different nodes, and the edge between two agents is assigned with both attribute and type.

4.2.2 Graph Neural Networks

Neural networks have proven their powerful expression ability on tasks with well-structured data, e.g., image classification [54] with grid-like data and machine translation [55] with chain-like data. However, there are many interesting tasks with data represented in the form of graph [125]. More and more recent works are proposed to generalize neural networks to the graph domain. These works are either spectral [126–128] or non-spectral approaches [129–131]. Spectral methods, e.g., Graph Convolutional Networks (GCNs) [128], depend on the Laplacian eigenbasis of the graph, which is hard to calculate for a large graph, while non-spectral methods, e.g., Graph Attention Networks (GATs) [131], perform information aggregation only on the local neighborhood, avoiding heavy calculation of Laplacian eigenbasis. However, GATs are designed for homogeneous graphs [131].

Although it introduces an attention mechanism to aggregate features from neighboring nodes according to edge connections, the edge attribute is not considered. To address this issue, edge enhanced graph neural network (EGNN) considers continuous multi-dimensional edge features by using each dimension to guide an individual attention operation [82]. Convolution with Edge-Node Switching graph neural network (CensNet) utilizes the line graph of the original undirected graph and designs convolution operations on both graphs to explore edge features [132]. NENN incorporates node-level and edge-level attentions in a hierarchical manner and learns the node and edge embeddings in the corresponding level [133]. EGAT extends GAT with edge embedding to handle continuous edge features of undirected homogeneous graphs [134]. Nonetheless, the heterogeneity of nodes and edges in a graph is ignored in the above-mentioned works. Heterogeneous graph attention network (HAN) proposes to handle heterogeneous nodes in a graph with a hierarchical attention mechanism, where the node-level attends over meta-path-based neighbors, and the semantic-level attends over different meta-paths [135]. Heterogeneous Graph Transformer (HGT) proposes node- and edge-type dependent attention mechanism to handle both node and edge heterogeneity in a graph, followed by a heterogeneous message passing mechanism and target-specific aggregation for feature updating [136]. These GNNs can handle heterogeneity in a graph but ignore the edge features. For more information about GNNs, please refer to recent review articles [114, 137].

Most existing GNNs handle heterogeneity and edge features separately and cannot be directly used to model the interaction represented by a directed edge-featured heterogeneous graph. In this work, we extend GAT [131] to handle both heterogeneity and edge features for interaction modeling in multi-agent trajectory prediction.

4.2.3 Trajectory Prediction with GNNs

Graph-based interaction representation has attracted more and more interest in the field of trajectory prediction, which gives rise to the application of GNNs. Authors of [77] test two widely used GNNs (GCN [128] and GAT [131]) and their adaptions on the trajectory prediction task and find that adaptions of GNNs, which discern between the target and surrounding agents, outperform the GNNs

that treat them without distinction. They conceptually prove the effectiveness of graph-based interaction representation, but the agents' dynamics are not considered in their work. GRIP proposes a graph convolutional model, which comprises convolutional and graph operation layers alternatively, to summarize the temporal and spatial features of interactive agents [80]. The convolutional layer is applied to the temporal dimension, and the graph operation is applied to spatial relationships. Then an LSTM-based encoder-decoder is used for the final prediction. GRIP can predict the trajectories of multiple agents, but it ignores edge attributes. SCALE-Net constructs edge attributes with the relative measurements between two agents and employs Edge-enhanced Graph Convolutional Neural Network [82] to summarize interactions considering edge attributes [81]. One common issue of the above-mentioned models is that they ignore the heterogeneity of traffic participants. Social-WaGDAT designs Wasserstein Graph Double-Attention Network to learn the structure of the interaction graph dynamically and applies kinematic constraints on the predicted trajectory [88]. VectorNet applies GNNs to a fully-connected hierarchical graph, where a sub-graph contains the feature of an object (either an agent or a map component) represented by a sequence of vectors [123]. Heterogeneity is considered in the constructed graph, but the fully-connected graph ignores the spatial structure of interaction, and the number of edges increases exponentially with the number of nodes. TNT adopts VectorNet as an interaction feature extractor and further considers the multimodality of driving by predicting multiple trajectories conditional on selected target points in the map [124]. It shares the drawbacks of VectorNet. ReCoG constructs a heterogeneous graph to represent agent-agent and agent-infrastructure relationships, where the infrastructure (a top-view map) is a node in the graph, and applies GAT and GCN to extract interaction features [83]. However, edge attributes and types are ignored in ReCoG.

Existing trajectory prediction methods are proposed for specific interaction representations, and they can hardly be applied to new representations. In this work, we propose our three-channel framework for simultaneous multi-agent trajectory prediction along with our interaction representation and HEAT network.

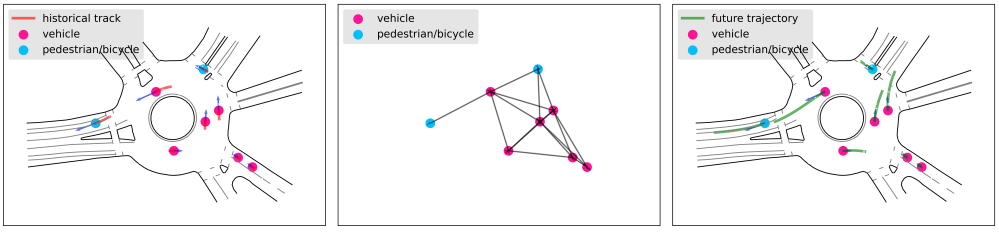

FIGURE 4.2: **Input, graph, and output.** *Left*, the one-second historical tracks of multiple agents of different types navigating in a roundabout scene. *Middle*, The structure of the constructed directed heterogeneous graph with neighboring connections. Self-loop is masked out for clarity. *Right*, the three-second future trajectories of multiple heterogeneous agents in the scene. The pink and blue dots show the current positions of vehicle and pedestrian/bicyclist agents, respectively. The solid red lines in the left figure are the historical trajectories of the agents over the last second. The solid green lines in the right figure are the corresponding future trajectories in three seconds. This figure is sampled from a roundabout scenario named *DR USA Roundabout FT* in the INTERACTION dataset.

4.3 Structure Overview

This section introduces the high-level structure of the proposed framework for heterogeneous multi-agent trajectory prediction and its key components, namely, agent-type-specific history encoder, HEAT-based heterogeneous interaction encoder, adaptive map selector, and agent-type-specific trajectory predictor. The proposed framework has three channels for dynamics, interaction, and map features, respectively, then jointly considers these features to predict future trajectories of heterogeneous agents. See Fig. 4.1 for an illustration of the three-channel framework.

Input and output. The task (see Fig. 4.2 for an illustration) of this work is to simultaneously predict multi-agent trajectories of a group of heterogeneous interactive agents considering their inter-agent interactions and the scene context (shown in the left of Fig. 4.2). At a time t, the input $\mathbf{X}_t$ contains each agent's historical states and the map of the scene (shown on the left of Fig. 4.2).

$$\mathbf{X}_t = [\mathcal{H}_t, \mathcal{M}], \tag{4.1}$$

where $\mathcal{H}_t = \{h_t^1, h_t^2, \ldots, h_t^n\}$ contains the historical states of n agents at time t, $\mathcal{M}$ is the scene context. Agent i's historical states at time t is represented by $h_t^i = [s_{t-T_h+1}^i, s_{t-T_h+2}^i, \ldots, s_t^i]$, with T_h as the traceback horizon. The state s_t^i, for

instance, can be agent i's position and velocity at t. The number of observed agents n is variable from case to case. The map $\mathcal{M}$ is to be shared by all the agents. The output contains predicted trajectories of $m \leq n$ heterogeneous agents (shown in right of Fig. 4.2):

$$\mathcal{F}_t = \{f_t^1, f_t^2, \ldots, f_t^m\}, \tag{4.2}$$

where $f_t^i = [(x_{t+1}^i, y_{t+1}^i), \ldots, (x_{t+T_f}^i, y_{t+T_f}^i)]$ is a sequence of the predicted 2D coordinates of agent i over a prediction horizon T_f, $\mathcal{F}_t$ is the set of predicted trajectories of m agents. Please note that the number of target agents m is not necessary to be equal to n and can vary from case to case. This is a typical situation in which an autonomous vehicle would need to predict the trajectories of the agents it is interacting with (the target agents), considering other non-target agents' information.

Agent-type-specific history encoder. To handle the heterogeneity of traffic participants, we propose to share a history encoder over a specific type of traffic participants. In this work, we assume that there are two types of traffic participants (i.e., vehicle and pedestrian/bicyclist), such that there will be two type-specific history encoders, one for each (see Fig. 4.1). History encoders are applied to individual agents' historical states to extract their dynamics features. The dynamics features are also used in the interaction channel as node features.

HEAT-based heterogeneous interaction encoder. In this work, we represent the interaction among heterogeneous traffic participants with a directed edge-featured heterogeneous graph (see the middle of Fig. 4.2 for an illustration and Sub.Sec. 4.4.2 for details) and propose a novel heterogeneous edge-enhanced graph attention network (HEAT) to extract interaction features (see Fig. 4.1). Nodes in the graph contain dynamics features of corresponding agents out from their history encoders.

Adaptive map selector. We design a convolutional neural network (CNN) to extract road features from a bird's eye view map of the driving scene and selectively share the map feature across all target agents according to their current positions, velocities, and yaw angles, by introducing a gate mechanism (see Fig. 4.1).

Agent-type-specific trajectory predictor. Similar to the history encoder, a trajectory predictor is shared over a specific type of target agents. The target agents in this work also fall into two categories (i.e., vehicle and pedestrian/bicyclist). To simultaneously predict trajectories, the predictor jointly considers the

target agents' dynamic features extracted from the history encoder, their interaction features obtained from the interaction encoder, and their corresponding map features received from the map selector. Please note that the input features of the trajectory predictors are hidden features (represented by high-dimensional vectors) from neural networks. See Fig. 4.1.

4.4 Method

This section first provides the architecture of the proposed multi-agent trajectory prediction framework (4.4.1), then elaborates on the proposed interaction representation (4.4.2), the proposed heterogeneous edge-enhanced graph attention network: HEAT (4.4.3), and gated map selector (4.4.4) for the framework.

4.4.1 Heterogeneous Multi-Agent Trajectory Prediction

The framework shown in Fig. 4.1 is proposed for multi-agent trajectory prediction, where there are two types of agents, leveraging both the historical states of agents and the infrastructure information. To handle the heterogeneity of agents, we design specific encoders (4.4.1.1) and decoders (4.4.1.4) for each type of agent. Considered agents are placed in their own exclusive coordinate system, and their interactions are represented by a directed edge-featured heterogeneous graph (4.4.2). A novel heterogeneous edge-enhanced graph attention network is proposed to extract interaction features from the constructed graph (4.4.1.2). To utilize the road structure and share it across all considered agents, we propose an adaptive map selector (4.4.1.3).

4.4.1.1 Agent-Type-Specific History Encoder

For an agent of type $\kappa, \kappa \in \{vehicle, pedestrian/bicyclist\}$, its historical states h_t^i is represented by a temporal sequence that can be passed to a type-specific encoder to extract its dynamics feature. Recurrent Neural Networks (RNNs), e.g., Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), are widely used for sequence modeling in machine translation [57, 138] and trajectory prediction [58,

83]. We adopt GRUs as history encoders in this work (Eq. 4.3) because of their effectiveness and simplicity:

$$r_t^i = \mathrm{GRU}_{\mathrm{hist}}^{\kappa}(h_t^i), \tag{4.3}$$

where $\mathrm{GRU}_{\mathrm{hist}}^{\kappa}$ is the historical encoder of agent type κ implemented using GRU, and r_t^i is the dynamics feature of vehicle i at time t. The output of this module is the dynamics features of all the agents:

$$R_t = \{r_t^1, r_t^2, \ldots, r_t^n\}, \tag{4.4}$$

where the dynamics features R_t also serve as the node features in the graph-based interaction representation.

4.4.1.2 Heterogeneous Interaction Modeling with HEAT

To comprehensively model the inter-agent interaction among heterogeneous agents, we represent the interaction as a directed edge-featured heterogeneous graph and propose a novel Heterogeneous Edge-enhanced graph ATtention network (HEAT) to extract interaction features from the graph representation. Details of the interaction representation and the proposed HEAT can be found in Sub.Sec. 4.4.2 and Sub.Sec. 4.4.3, respectively.

Agents' dynamics features R_t are put into their corresponding nodes in the graph. Then the proposed HEAT is applied to the graph to model the interaction features for all agents simultaneously.

$$G_t = \{g_t^1, g_t^2, \ldots, g_t^n\} = \mathrm{HEAT}_{\mathrm{enc}}(R_t, E_t), \tag{4.5}$$

where E_t is the edge set containing edge indexes, edge attributes, and edge types, g_t^i the interaction feature of agent i at time t, and G_t interaction features of all agents.

4.4.1.3 Map Selection with Gate Mechanism

Road structure shapes the motion of agents navigating within an urban scene, so it is necessary to take into consideration the road structure for trajectory prediction. Previous single trajectory prediction methods use a fixed-size local map centered at

the target vehicle's current position. But for simultaneous multi-agent trajectory prediction, this map representation has at least two drawbacks: 1) It needs to save multiple maps for multiple agents. 2) The fixed-size map can be either too large for a slow agent or too small for a fast agent. To handle the above-mentioned drawbacks, we propose an adaptive map selection method that allows sharing a global map across all the agents according to their current positions, velocities, and yaw angles:

$$m_t^i = \text{Selector}_{\text{map}}\left(\mathcal{M}, (x_t^i, y_t^i, v_{x_t}{}^i, v_{y_t}{}^i, \phi_t^i)\right), \tag{4.6}$$

where $\mathcal{M}$ is the global map and $(x_t^i, y_t^i, v_{x_t}{}^i, v_{y_t}{}^i, \phi_t^i)$ is the current position, velocity, and yaw angle of agent i in the map. The selector allows the method to selectively consider the global map and focus on the most relevant areas. The selected map feature of agent i is conditioned on its states.

4.4.1.4 Agent-Type-Specific Future Decoder

For an agent of type $\kappa, \kappa \in \{vehicle, pedestrian/bicyclist\}$, its future trajectory is predicted using an agent-type-specific future trajectory decoder by jointly considering its individual dynamics r_t^i, its interaction with other agents g_t^i, and the map feature regarding its current states m_t^i.

$$f_t^i = \text{LSTM}_{\text{fut}}^{\kappa}([r_t^i \| g_t^i \| m_t^i]), \tag{4.7}$$

where $\text{LSTM}_{\text{fut}}^{\kappa}$ is the future decoder shared across agents of type κ, $[r_t^i \| g_t^i \| m_t^i]$ the concatenation of features, and f_t^i the predicted future trajectory of agent i.

4.4.2 Interaction Representation with Directed Edge-Featured Heterogeneous Graph.

In this work, we place all agents in their own exclusive coordinate systems and represent their interaction as a directed edge-featured heterogeneous graph.

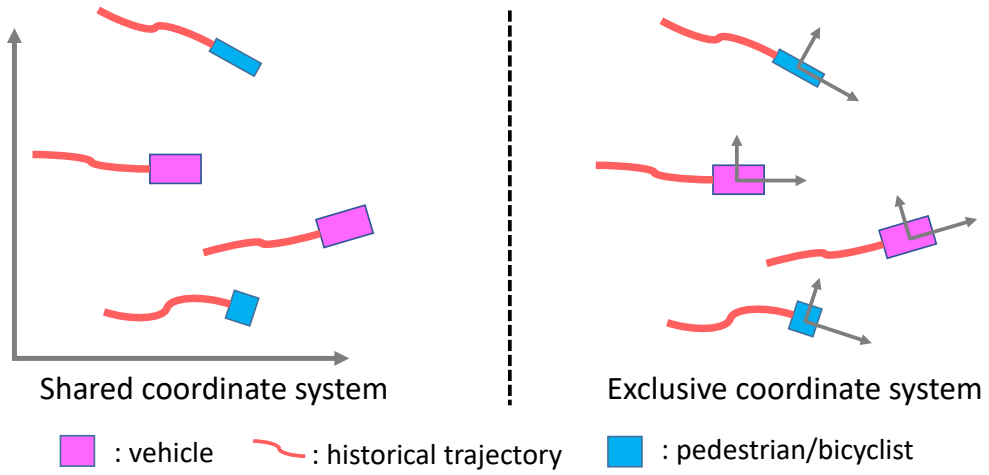

FIGURE 4.3: **Shared and exclusive coordinate systems.** *Left*, Heterogeneous agents in a shared coordinate system. *Right*, Heterogeneous agents with exclusive coordinate systems. An agent's exclusive coordinate system is with its origin fixed at the vehicle's current position and its horizontal axis pointing to the vehicle's moving direction.

4.4.2.1 Exclusive Coordinate System

Most existing interaction-aware trajectory prediction methods use either a shared coordinate system for all agents or an exclusive coordinate system for each to represent trajectories. A shared coordinate system preserves the spatial relationship among agents. However, it is sensitive to translation and rotation. The input to the model becomes totally different when a new shared coordinate system is applied. But in the case of the exclusive coordinate system, agents' states are represented locally and independent of other agents. The localized exclusive coordinate system standardizes the states of agents but omits spatial relationships among agents, which should be reserved with edge features to take advantage of the exclusive coordinate system.

4.4.2.2 Graph Represented Interaction

In this work, we propose to represent inter-agent interaction as a directed edge-featured heterogeneous graph for multi-agent trajectory prediction. Each node represents a traffic participant with a specific type and contains its feature extracted from a sequence recorded in its exclusive coordinate system. An edge from node j to node i means that node i's behavior is influenced by node j, and the edge

attribute is relative measurements of node j to node i, e.g., position, velocity, and yaw angle. The edge type is a concatenation of the type of node j and node i. For details of the edge construction in this work, please refer to Sub.Sec. 4.5.1, and the code will be released soon.

Definition 2 (Directed Edge-Featured Heterogeneous Graph)**.** A directed edge-featured heterogeneous graph can be represented by $\mathbb{G} = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is the set of n nodes, and $E \subset V \times V$ is the set of directed edges. Each node contains its node feature and belongs to a specific type. Each directed edge contains an edge attribute and falls into a specific edge type.

Compared to previous works that represent interaction as a homogeneous graph [77], edge-featured homogeneous graph [81], or heterogeneous graph without edge attributes [87], the proposed representation is more comprehensive. It covers the heterogeneity of traffic participants with heterogeneous nodes, preserves their individuality with exclusive coordinate systems, considers the difference of the mutual influence between two agents with directed edges, and maintains the spatial relationship of all the agents using edge attributes.

Transformation-insensitive interaction graph. To model the interaction among agents of different types, we construct a directed heterogeneous graph to represent these inter-agent relationships. An edge e_{ij} pointing from agent j to agent i is constructed if agent j is within a predefined neighborhood of agent i. Each valid edge e_{ij} is assigned with edge attribute and edge type. Then the edge set is:

$$E = \{e_{ij}\}_{(j \in \mathcal{N}_i)}, \quad i = 1, \ldots, n, \tag{4.8}$$

where i and j are the indexes of agents, and $\mathcal{N}_i$ is the neighborhood of agent i. Self-loops (e_{ii}) are included in the edge set. An example of the constructed graph is shown in the middle of Fig. 4.2

4.4.3 HEAT Layer

The above-mentioned interaction representation should be treated with a graph neural network that can handle the heterogeneity of nodes, directed edges, and continuous edge attributes. However, as shown in related works, existing GNNs do

not handle this in one fell swoop. In this work, we design a heterogeneous edge-featured graph attention network (HEAT), an extension of GAT, for the proposed comprehensive interaction representation. HEAT can be constructed by stacking HEAT layers. A HEAT layer updates node features by aggregating information from neighborhoods. It first transforms node and edge features accordingly, then aggregates node features via edge-enhanced masked attention (or optional multi-head attention) mechanism.

4.4.3.1 Input and Output

The input to the HEAT layer contains a set of node features: $\mathbf{h} = \{\vec{h}_i | i \in [1, n]\}$, where $\vec{h}_i \in \mathbb{R}^{F_h}$ is the feature vector of node i; and a set of edge attributes: $\mathbf{e}^{attr} = \{e_{ij}^{attr} | i, j \in [1, n]\}$, where $e_{ij}^{attr} \in \mathbb{R}^{F_e^{attr}}$ is the attribute of the edge pointing from node j to node i. A set of edge types is represented as $\mathbf{e}^{type} = \{e_{ij}^{type} | i, j \in [1, n]\}$, where $e_{ij}^{type} \in \mathbb{R}^{F_e^{type}}$ is the type of the edge pointing from node j to node i. The output of the HEAT layer is a new set of node features: $\mathbf{h}' = \{\vec{h}'_i | i \in [1, n]\}, \vec{h}'_i \in \mathbb{R}^{F'_h}$.

4.4.3.2 Heterogeneous Transformation

Different kinds of nodes in a heterogeneous graph have different feature spaces and should be projected to a shared feature space. We adopt the node-type-specific transformation matrix $\mathbf{M}_{\kappa i} (\kappa \in \{vehicle, pedestrian/bicyclist\})$ introduced in [135] to handle two kinds of nodes in this work. The transformation can be expressed as $\vec{h}_i = \mathbf{M}_{\kappa i} \cdot \vec{h}_i$. Please note that we will simply re-use symbols including $\vec{h}_i, \mathbf{e}^{attr}, \mathbf{e}^{type}$ to indicate the transformed feature in the attention (4.4.3.3) and aggregation (4.4.3.4) parts.

Existing works either consider edge attributes or edge types as edge features, whereas this is not the case for multi-agent trajectory prediction. We argue that, for trajectory prediction, edge feature and type are two different attributes. The edge features are usually some measurements in a continuous space, such as the distance between two nodes. However, the edge type is always a discrete indicator. Thus, we separately consider the edge features and types by introducing the edge

attribute transformation: $\mathbf{e}^{attr} = \mathbf{M}_\phi \cdot \mathbf{e}^{attr}$, where $\mathbf{M}_\phi$ is the edge attribute transformation matrix, and the edge type transformation: $\mathbf{e}^{type} = \mathbf{M}_\chi \cdot \mathbf{e}^{type}$, where $\mathbf{M}_\chi$ is the edge type transformation matrix.

4.4.3.3 Edge-Enhanced Masked Attention

For an edge pointing from node j to node i, its edge feature: $e_{ij} = [e_{ij}^{attr} \| e_{ij}^{type}]$, is a concatenation of its transformed edge attribute and type. For node i, a concatenated feature vector $e_{ij}^{+} = [e_{ij} \| \vec{h}_j]$ represents the feature of node j from node i's point of view considering the edge attribute and type. e_{ij}^{+} is then sent to a shared attention mechanism [55], which is a single-layer feed-forward neural network, $\vec{\mathbf{a}}$, followed by LeakyReLU non-linearity and softmax normalization. The attention coefficient α_{ij} indicates the importance of the node j to node i jointly considering node and edge features. A GAT layer performs masked attention, which attends over the neighborhood of node i only, to utilize the structural information of the graph while casting away the edges' feature and type [131]. In this work, an edge-enhanced masked attention in Eq. 4.9 is performed to fully consider the graph attributes:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakeyReLU}\left(\vec{\mathbf{a}}^T [\vec{h}_i \| e_{ij}^{+}]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakeyReLU}\left(\vec{\mathbf{a}}^T [\vec{h}_i \| e_{ik}^{+}]\right)\right)}, \tag{4.9}$$

where $\mathcal{N}_i$ is the neighborhood of node i in the graph. The attention coefficients are then used to update the feature of node i with a linear combination over its neighborhood.

4.4.3.4 Node Feature Aggregation

The feature of node i is updated by calculating a weighted sum of edge-integrated node features over its neighborhood, followed by a Sigmoid function:

$$\vec{h}_i' = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}_h [e_{ij}^{attr} \| \vec{h}_j]\right). \tag{4.10}$$

Edge types are not included in the edge-integrated feature since it is discrete and already considered in the previous attention mechanism (Eq. 4.9). Similar

TABLE 4.1: Notations of HEAT layers

$\vec{h}_i$	Feature vector of node i
$\mathbf{h}$	The set of node features in a graph
e_{ij}^{attr}	The attribute of edge from j to i
$\mathbf{e}^{attr}$	The set of edge attributes in a graph
e_{ij}^{type}	The type of edge from j to i
$\mathbf{e}^{type}$	The set of edge types in a graph
e_{ij}	Concatenation of projected attribute and type
e_{ij}^{+}	Concatenation of e_{ij} and $\vec{h}_j$
α_{ij}	Node j's attention coefficient for node i
$\mathcal{N}_i$	Neighborhood of node i
$\|$	Concatenation
$\vec{\mathbf{a}}^T$	Attention mechanism
σ	Sigmoid function
$\mathbf{h}'$	The set of updated node features in a graph

to GAT [131], HEAT allows the running of several independent attention mechanisms to stabilize the self-attention mechanism.

The elements of the proposed HEAT layer are listed in Tab. 4.1 for convenience.

4.4.4 Gated Map Selection

The road structure highly affects the motions of traffic participants, so trajectory prediction cannot ignore this information. Single-agent trajectory prediction methods, such as ReCoG [83], use a fixed-size local map centered at the target vehicle's current position. However, a fixed-size local map ignores the dynamics of agents. A small map is enough to predict a slow agent, while a larger map is needed for a fast-moving agent. Multi-agent prediction methods, such as MATF [70], share the same map feature across all target vehicles ignoring the fact that different target agents are affected by different parts of the map. To enable selective map sharing, we propose to apply the gate mechanism on the CNN-extracted map feature for map selection, which has been widely used in sequence modeling [66, 110, 139]. For example, LSTM has three gates (input gate, forget gate, and output gate) to

manage the information flows along the sequence data [139]. The input gate is designed to select what information about the current step is to be added to the memory of the network. In this work, we design the selection gate z_t^i to select a map feature for an agent i according to its current state in the map:

$$z_t^i = \sigma(W_z[\vec{\mathcal{M}} \| s_t^i] + b_z), \tag{4.11}$$

where $\vec{\mathcal{M}}$ is the map $\mathcal{M}$'s feature vector extracted using a CNN, s_t^i is agent i's current state in the map's coordinate system, W_z is a projection weight matrix, b_z is a bias, σ is a Sigmoid function. Thus, z_t^i is a vector with each element containing a number between 0 and 1. Then the map feature of agent i at time t is selected with this gate:

$$m_t^i = z_t^i \circ \vec{\mathcal{M}}, \tag{4.12}$$

where $\circ$ is element-wise production and m_t^i is the selected map feature.

4.5 Real-World Dataset Validation

This section first compares the proposed trajectory prediction method with state-of-the-art models on the recently published INTERACTION dataset [20] for urban driving, then on the NGSIM US-101 [47] dataset for highway driving. The INTERACTION dataset is provided by the Mechanical Systems Control (MSC) Lab of the University of California, Berkeley, the Centre for Robotics of MINES ParisTech, and the Institute for Measurement and Control Technology (MRT) of FZI Research Center for Information Technology and Karlsruhe Institute of Technology. The NGSIM dataset is provided by the U.S. Federal Highway Administration.

4.5.1 Validation on Heterogeneous Dataset

The proposed heterogeneous multi-agent trajectory prediction method is trained and validated on the INTERACTION [20]. The full name of the dataset is INTERnational, Adversarial and Cooperative moTION Dataset. It contains naturalistic trajectories of different traffic participants, e.g., vehicles and pedestrians, in highly interactive urban scenarios worldwide. The recorded scenarios fall into three categories: roundabout, intersection, and merging.

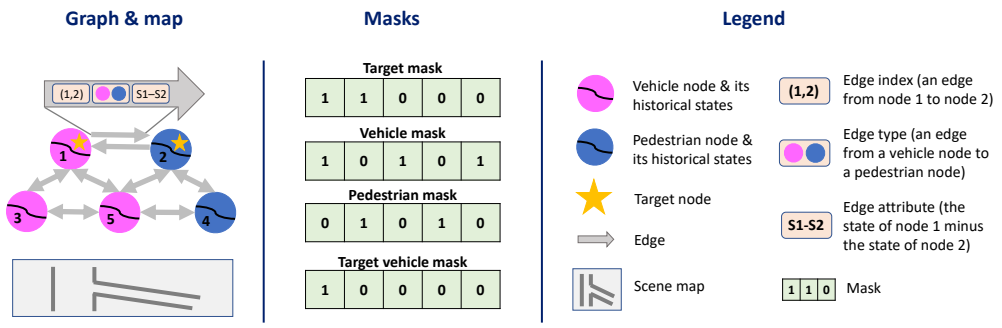

FIGURE 4.4: **Processed data.** *Left*, the directed edge-featured heterogeneous interaction graph (top) obtained after processing and the map (bottom). *Middle*, the mask vectors. *Right*, the legend of this figure. The interaction graph is constructed via a close connection strategy, where each node is only connected to its neighboring nodes via directed edges. A directed edge is identified via its edge index and contains edge type and edge attribute.

4.5.1.1 Heterogeneous Dataset

INTERACTION dataset provides states of agents at each timestamp along with a high-definition (HD) map. The state of a vehicle at a timestamp includes its position, velocity, yaw angle, and shape, while the state of a pedestrian/bicyclist includes only its position, velocity, and yaw angle. Since this work aims at simultaneously predicting trajectories of multiple heterogeneous agents and proposes to represent inter-agent interaction as a heterogeneous edge-featured graph, the raw dataset is processed accordingly. Please see Fig. 4.4 for an illustration of the processed data and the appendix for a detailed description. The processed dataset is split to train and validation set following the split suggested by the authors of the INTERACTION dataset. The processed dataset contains $425,192$ data pieces for training and $104,627$ for validation.

4.5.1.2 Comparison with State-of-the-art Methods

In this work, we evaluate prediction performance using Average Displacement Error (ADE) and Final Displacement Error (FDE) in meters adopted by previous works [83, 124]. The proposed method is compared with the following methods on the INTERACTION dataset.

- DESIRE: DESIRE predicts multimodal trajectories by jointly considering motion history, static scene context, and inter-agent interactions. It first

TABLE 4.2: Comparison with state-of-the-art methods on the INTERACTION dataset

Methods	MM	ADE@3sec (m)	FDE@3sec (m)
DESIRE [103]	✓	0.32 ($\mathbf{min}_6$)	0.88 ($\mathbf{min}_6$)
MultiPath [59]	✓	0.30 ($\mathbf{min}_6$)	0.99 ($\mathbf{min}_6$)
TNT [124]	✓	0.21 ($\mathbf{min}_6$)	0.67 ($\mathbf{min}_6$)
ReCoG [83]		**0.19**	**0.65**
HEAT-I-R (Ours)		**0.19**	0.66

generates diverse hypothetical future prediction samples using a conditional variational auto-encoder, then ranks and refines the samples in an inverse optimal control framework with regression [103].

- MultiPath: MultiPath handles driving uncertainty by hierarchically modeling intent and control uncertainties. It first produces a set of K anchor trajectories as intents, then predicts future trajectories conditioned on anchors, where the uncertainty is modeled as a Gaussian distribution given an intent [59].

- TNT: TNT utilizes VectorNet [123] to encode the target agent's interaction with surrounding agents and the environment. It first predicts an agent's target states within a prediction horizon, then generates trajectories for each target. Finally, a set of predictions is selected according to estimated likelihoods for multimodality [124].

- ReCoG: ReCoG represents vehicle-vehicle and vehicle-infrastructure interactions as a heterogeneous graph and applies state-of-the-art GNNs for interaction encoding. It predicts a single trajectory for a single target vehicle [83].

Tab. 4.2 compares the proposed three-channel model HEAT-I-R with existing methods. It shows that: 1) The proposed HEAT-I-R outperforms DESIRE [103] and MultiPath [59], even though these two methods predict multimodal (MM) trajectories for a single agent and the ADE and FDE are reported with the minimum values among the multiple predictions [124]; 2) HEAT-I-R matches the performance of TNT [124] and ReCoG [60]. Please note that TNT [124] predicts six-modal trajectories for a single agent and reports the minimum ADE and FDE over all predictions, and ReCoG [83], the winner solution of the INTERPRET Challenge

TABLE 4.3: Ablative comparison on the INTERACTION dataset's *DR USA Roundabout FT* scenario

Methods	ADE@8sec (m)	FDE@8sec (m)
R	3.99	11.64
GAT	3.98	11.59
GAT-R	3.5	10.62
HEAT	3.10	8.83
HEAT-R	3.09	8.84
HEAT-I-R	**2.97**	**8.56**

(NeurIPS 2020) [140], predicts a single trajectory for a single target. Compared to TNT [124] and ReCoG [83], the proposed HEAT-I-R is able to predict trajectories of multiple agents (MA) simultaneously. The inference time satisfies the real-time requirements. It uses 0.06 seconds to predict the trajectories of a batch of 128 data pieces, including hundreds of target agents.

4.5.1.3 Ablative Study

In this work, we conduct ablative studies on the INTERACTION dataset's *DR USA Roundabout FT* scenario for a longer prediction horizon (eight seconds). The eight-second horizon is selected to challenge our method since a longer-term prediction is intrinsically more difficult than a short-term one [53]. The following settings are trained and validated on the same dataset.

- R: One-channel model considering the target vehicle's RNN-encoded dynamics feature only for future trajectory prediction.
- GAT: One-channel model predicting the future trajectory of the target vehicle considering its graph-modeled interaction feature extracted using GAT.
- GAT-R: Two-channel model jointly considering the GAT-extracted interaction and RNN-encoded dynamics features for trajectory prediction.
- HEAT: One-channel model predicting the future trajectory of the target vehicle considering its graph-modeled interaction feature extracted using HEAT.
- HEAT-R: Two-channel model jointly considering the HEAT-extracted interaction and RNN-encoded dynamics features for trajectory prediction.

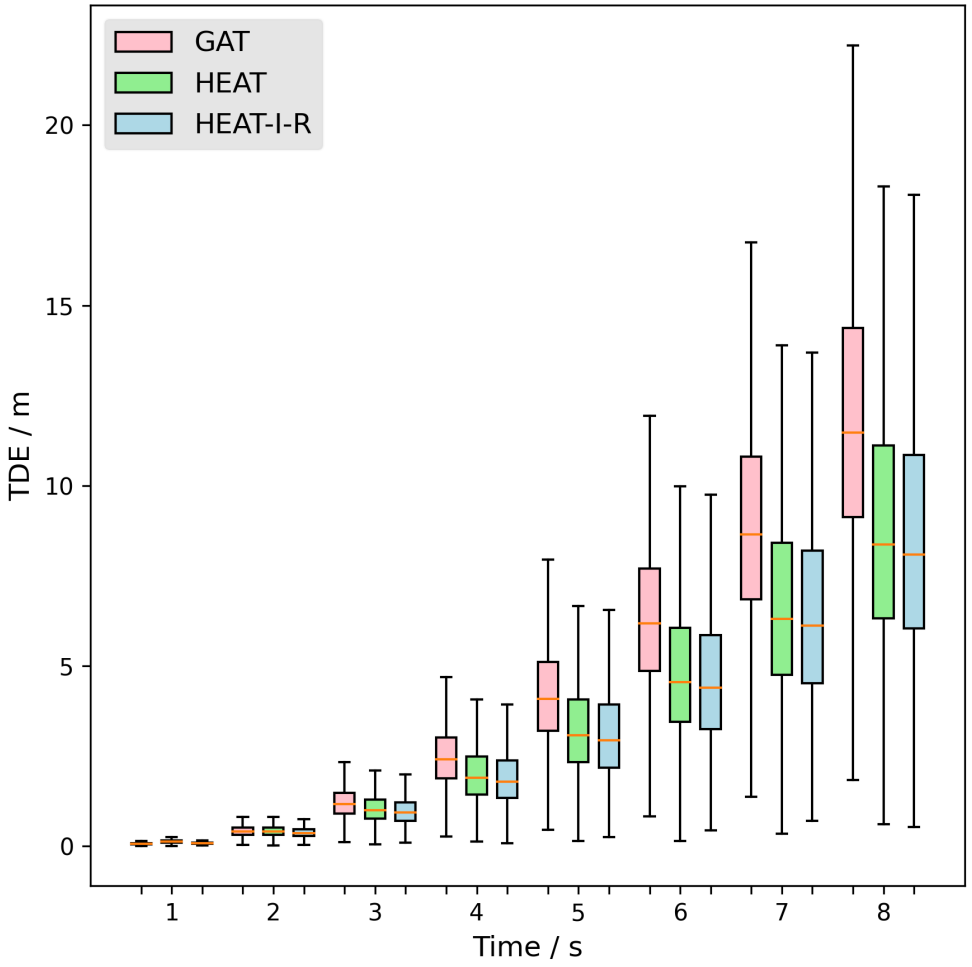

FIGURE 4.5: **Box plots of the TDEs of ablative models.** GAT, HEAT, and HEAT-I-R are selected for the clarity of the box plots.

- HEAT-I-R: The proposed three-channel framework, which combines the target vehicle's individual dynamics feature, its interaction feature, and the selected map feature for trajectory prediction.

The source codes for this section are available online and can be found in Appendix-C.

Tab. 4.3 shows the ADE@8sec and FDE@8sec of the above-listed implementations. It is observed that the proposed three-channel framework (HEAT-I-R) outperforms its two-channel (HEAT-R) and one-channel (HEAT) ablations, which supports the intuition that individual dynamics, inter-agent interactions, and road information all benefit trajectory prediction. It is also noticed that the one-channel GAT-based method (GAT in Tab. 4.3) performs as poorly as the non-interaction-aware

method (R in Tab. 4.3.), but their combination (GAT-R) improves prediction accuracy. GAT-based methods are not suitable for trajectory prediction with exclusive coordinate systems, where the spatial relationships of agents are stored in edge features because GATs ignore edge features.

Fig. 4.5 shows box plots of the TDE (displacement error over time) of three ablatives models (GAT, HEAT, and HEAT-I-R) over an eight-second prediction horizon. The red boxes show the results of GAT, the green boxes the results of HEAT, and the blue boxes the result of the proposed HEAT-I-R. Outliers and the results of other ablative models are not plotted for clarity. It can be seen that the proposed HEAT-based model shows more stable performance (with a shorter interquartile range (IQR)) than the GAT-based model, and the proposed three-channel framework (HEAT-I-R) further improves accuracy and stability.

4.5.2 Validation On Homogeneous Dataset

The proposed HEAT is an immediate extension to GAT [131] while the GAT is designed for homogeneous graphs. To compare the proposed HEAT with GAT on the single-agent trajectory prediction task, we construct two homogeneous datasets using vehicle trajectories provided by the publicly accessible NGSIM US-101 dataset [47]. The trajectories in NGSIM US-101 dataset are recorded from a segment of U.S. Highway 101 at 10 Hz. Since most of the trajectories did not change lanes throughout the study area, we reprocess the dataset to build a roughly balanced dataset where lane-keeping trajectories do not dominate the dataset. The dataset is constructed by first selecting target vehicles that have changed their lanes only once during recording, then selecting trajectory segments for the target vehicle and its neighboring vehicles. The data processing procedure is similar to that in CNN-LSTM [60], a previous work focusing on the NGSIM dataset, except that an arbitrary number of neighboring vehicles is allowed in this work. After the above processing, a total of $63,176$ data pieces are selected for training $(53,176)$ and validation $(10,000)$. The selected data is further processed to formulate two different datasets, one with exclusive coordinate systems and the other one with a shared coordinate system. In the former dataset, each agent is placed in its own coordinate system, in which the agent's current position and yaw angle are zeros, while in the latter one, all the agents share the coordinate system whose origin

is fixed at the current position of the target vehicle and horizontal axis points to the direction of the target vehicle's current velocity. The former dataset has edge attributes containing the relative position of the agent in the source node to the agent in the target node, while the latter one does not have edge attributes because of the shared stationary frame of reference.

Similar to the ablative study on the heterogeneous dataset, we implement five models, namely R, GAT, GAT-R, HEAT, and HEAT-R, to show the effectiveness of the proposed HEAT layer. Descriptions of these models can be found in 4.5.1.3. All these models are trained and validated on the dataset with the exclusive coordinate system, and the GAT-based baselines are further implemented in the dataset with a shared coordinate system.

We compare the proposed model with state-of-the-art methods and report the results in Tab. 4.4. The prediction results in this section are evaluated using root-mean-square error (RMSE) in meters to compare with the existing works [58, 60, 81].

Tab. 4.4 compares the proposed method with existing works. It shows that the proposed HEAT-R method outperforms existing models at longer prediction horizons (3–5 sec) and matches the state-of-the-art methods in short-term prediction (1–2 sec). It can be seen that the accuracy of the proposed HEAT-R is quite close to that of CNN-LSTM [60], which uses a share coordinate system and accepts exactly eight close neighboring vehicles. However, HEAT-R is able to handle an arbitrary number of neighboring vehicles and use the exclusive coordinate system that standardizes the input sequence and narrows down the search space for the input encoder.

Tab. 4.5 compares HEAT with GAT-based and RNN-based methods, where the first five rows (#1–5) show the results on the dataset with the exclusive coordinate system, and the last two rows (#6–7) show the results of GAT-based models on the dataset with the shared coordinate system. It can be seen that the graph-based interaction-aware methods (#2–7) outperform the non-interaction-aware RNN-based one (#1). This observation is consistent with previous works [58, 60, 70], which shows again the necessity to model interaction for trajectory prediction. The GAT-based methods show better performance on the second dataset (shared, #6,7)

compared to the results on the first dataset (exclusive, #2,3). This is quite reasonable in that GAT ignores the spatial relationship among agents contained in the edge features in the first dataset, while the spatial relationship is preserved by the shared coordinate system in the second dataset. The proposed HEAT-based methods (#4–5) for the dataset with the exclusive coordinate system outperform all GAT-based methods. This shows the advantage of using the exclusive coordinate system and applying HEAT for interaction extraction.

It can also be found that the results reported in this section seem poorer than that of the previous section. But we avoid comparing methods across datasets for the following reasons: 1) in this section, the main model HEAT-R does not consider road structure since the highways are usually relatively straight; 2) the average speed of highway driving is usually higher than that of the urban driving, which makes the prediction task more challenging; 3) the proposed method is designed for heterogeneous multi-agent prediction, so it is more suitable for multiple agents' prediction.

TABLE 4.4: Prediction performance comparison with existing works (RMSE in meters) on the NGSIM dataset

	Methods	**Prediction horizon**				
		1 sec	***2 sec***	***3 sec***	***4 sec***	***5 sec***
1	HEAT-R (Ours)	0.68	0.92	**1.15**	**1.45**	**2.05**
2	CS-LSTM [58]	0.61	1.27	2.09	3.10	4.37
3	GRIP [80]	**0.37**	**0.86**	1.45	2.21	3.16
4	CNN-LSTM [60]	0.64	0.96	1.22	1.53	2.09
5	CS-LSTM(M) [58]	0.62	1.29	2.13	3.20	4.52
6	GAIL-GRU [141]	0.69	1.51	2.55	3.65	4.71
7	Scale-Net [81]	0.46	1.16	1.97	2.91	-
8	MATF-GAN [70]	0.66	1.34	2.08	2.97	4.13

4.5.3 Implications and Limitations

It can be seen from the validation results that the proposed method can simultaneously predict multiple heterogeneous objects' trajectories with state-of-the-art performance. The assumption and concept of this method are in line with the real-world implementation scenarios, where autonomous vehicles interact with different types of other road users and need to predict their future motions for better

TABLE 4.5: Prediction performance comparison over ablative implementations (RMSE in meters) on the NGSIM dataset

	Methods	Prediction horizon				
		1 sec	*2 sec*	*3 sec*	*4 sec*	*5 sec*
1	R-1	0.6931	1.7275	3.0850	4.7735	6.7855
2	GAT-1	0.7808	1.4916	2.3914	3.4981	4.8713
3	GAT-R-1	0.8228	1.6987	2.7385	3.9672	5.4129
4	HEAT-1	0.6590	0.8556	1.0469	1.3216	1.8894
5	HEAT-R-1	0.6794	0.9212	1.1528	1.4457,	2.0518
6	GAT-2	0.7685	1.2115	1.7343	2.4533	3.5466
7	GAT-R-2	0.6439	0.9592	1.2643	1.6489	2.3124

decision-making. The prediction results can be used by the downstream decision-making and planning modules to improve safety and efficiency. In addition, not limited to autonomous driving, this method can be expanded and applied to many other robotic application domains.

Despite the performance and scalability, the proposed method still has limitations. Currently, it can only make deterministic predictions, while the motion of moving agents would have inherent multimodality. This thesis addresses the inherent multimodality of driving behaviors in a map-adaptive way in Chapter 5. In this work, we assume that the input data is ready to use, but the quality and availability of the data will be affected by the hardware settings in real-world applications, and the integration of the proposed method with other sub-modules and algorithms still needs to be configured in vehicle implementations.

4.6 Conclusions

In this work, we propose a three-channel framework for simultaneous heterogeneous multi-agent trajectory prediction. We represent the inter-agent interaction in traffic with a directed edge-featured heterogeneous graph, design a novel heterogeneous edge-enhanced graph attention network for inter-agent interaction modeling, and introduce a gate mechanism for selective map sharing across all target agents. Validations on both urban (INTERACTION) and highway (NGSIM) driving datasets show that the proposed method achieves state-of-the-art performance and is able

to simultaneously predict multi-agent trajectories of an arbitrary number of heterogeneous agents.

For future works, one promising direction is to handle the multimodality of traffic participants' behaviors by introducing multimodal prediction. This will largely reduce the minimum ADE. Another direction is to incorporate rich infrastructure information, such as traffic lights, into our framework, to enhance the prediction accuracy.

Chapter 5

Map-Adaptive Multimodal Trajectory Prediction Using Hierarchical Graph Neural Networks

Both Chapter 3 and Chapter 4 focus on deterministic trajectory prediction ignoring the multimodality of driving behaviors. This chapter further extends the deterministic prediction to map-adaptive multimodal prediction. A heterogeneous hierarchical graph containing agents and their candidate centerlines (CCLs) is developed to represent the driving scene. A hierarchical graph operator (HGO) is proposed to regulate the information flow in graph operations and obtain the encoded scene feature for the trajectory decoder. The trajectory decoder is able to predict a set of possible trajectories of the target vehicle according to its CCLs, thus making the predictor map-adaptive.

5.1 Introduction

Most recent works have proposed to jointly consider the target agent's own dynamics, its interaction with surrounding agents, and the impacts of infrastructures. Surrounding agents are the agents that can potentially impact the target vehicle's future motion [11]. The selection of criteria varies from study to study. They

represent the agents and the map either separately or integrally and try to predict multimodal future motions of target agents [21, 59, 61, 76, 86, 124]. However, scene representation and multimodality for trajectory prediction are far from being well explored. Most existing works predict a predefined number of possible future motions of a target agent. A prediction set with a fixed number of options limits the model to generalize to complex map geometries, and few studies have been done for map-adaptive prediction. Goal-Net [61] is the map-adaptive method that is most related to our work. It predicts a set of lane-following trajectories according to lane centerlines, making it adaptive to different road structures. It also generates a motion-based trajectory to cover critical corner cases where a vehicle drives in an uncooperative way.

In this work, we attempt to represent the complex driving scene and predict the multimodal motions of a target vehicle in an integrated manner. The following challenges are to be tackled: First, the number of traffic participants can vary from case to case. Second, the road structure is quite complex and shapes the motion of vehicles to a great extent. Third, the interdependencies among vehicles and infrastructures are complicated. Forth, driving behavior is multimodal, meaning that given the same situation, there may exist different future motions. For example, a vehicle approaching an intersection can either go straight forward or turn left/right. Fifth, driving behavior is not always rational in some critical corner cases. We represent the driving scene with a heterogeneous hierarchical graph, wherein a node is either an agent or one of its CCLs and contains the corresponding feature. This representation can accommodate an arbitrary number of agents and their centerlines, thus tackling the first two challenges. We represent interdependencies among nodes with directed edges and propose a three-stage graph operator to encode the scene graph. An edge-masking technology is used to regulate information flow in different stages. The hierarchical graph operator is designed to cope with the third challenge. To handle the fourth and fifth challenges, we design an integrated multimodal predictor via graph operation and edge-masking that can simultaneously predict single-CCL, cross-CCL, and motion-based future trajectories of a target agent. The graph operation allows our predictor to output a variable number of trajectories according to the target agent's CCLs. Thus our method is map-adaptive.

The main contributions of this work can be summarized as follows:

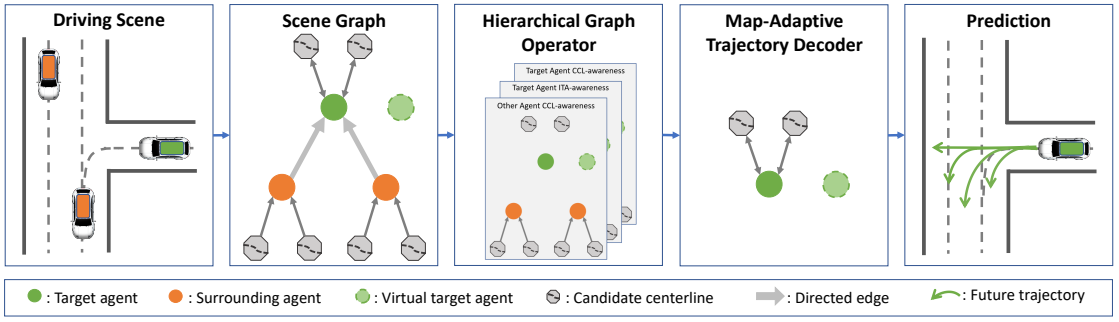

FIGURE 5.1: **Overview of the proposed scheme.** Given a driving scene consisting of agents and the HD map, we first assign a variable number of CCLs to each agent according to its dynamics and the road structure. Then we represent the driving scene with a heterogeneous hierarchical graph (scene graph) with an additional virtual target agent node. Next, we process the scene graph using our proposed hierarchical graph operator. Finally, we apply a map-adaptive prediction decoder to predict a variable number of trajectories. These predictions of the target agent fall into three categories, namely single-CCL, cross-CCL, and motion-based predictions.

- At the high level, this work designs a map-adaptive multimodal trajectory prediction framework that jointly considers the target agent's own dynamics, its interaction with other agents, and the road structure.

- For scene representation and encoding, this work represents the complex driving scene with a single heterogeneous hierarchical graph. It proposes a hierarchical graph operator augmented with an edge-masking technology to encode the scene graph for trajectory prediction.

- For map-adaptive trajectory decoding, this work proposes a comprehensive map-adaptive multimodal predictor implemented with a graph operation and edge-masking technology. It produces three kinds of predictions: 1) a set of CCL-guided trajectories adaptive to the road topology and can generalize to unseen road structures; 2) a cross-CCL trajectory conditioned on the overall topology, considering that a driver will not always follow a single centerline; 3) a motion-based trajectory to cover the corner case where the vehicle is not following the map topology.

The remaining part of the paper proceeds as follows: Sec. 5.2 introduces two clusters of related works. Sec. 5.3 outlines the main parts of the proposed framework.

Sec. 5.4 describes the proposed method in detail. Sec. 5.5 shows the validation results on a large-scale dataset. Sec. 5.6 provides discussions on the results. Sec. 5.7 concludes this work and lists some possible future works.

5.2 Related Work

This section reviews scene representations for interaction-aware trajectory prediction and multimodal trajectory prediction methods. The distinction and advantage of the proposed scene representation and multimodal predictor are illustrated following each group of related work.

5.2.1 Scene Representations

Most recent works jointly consider agent and map information for trajectory prediction. The map (with or without agents) information can be represented via rasterization or vectorization. Rasterization methods render the map (with or without agents) information into a multi-channel image. Each layer of the image contains some semantic information, such as the traffic signals, the road structure, and the agents' past trajectories. Multiple-Trajectory Prediction (MTP) renders the actor's surrounding map and other actors into an actor-specific RGB (red, green, and blue) image and distinguishes objects with shapes and colors [76]. MultiPath uses a single 3-dimensional array to represent the scene, with the first two dimensions representing spatial locations in a Bird's Eye View (BEV) image and the depth dimension holding a rich dynamic and semantic information [59]. ReCoG represents a target-centered local map as a 2D image, where only road structure is rendered, then takes the map as a node in a heterogeneous graph to comprehensively represent the scene [83]. HEAT renders the global map into a 2D image and shares the map information across all target agents via a gate-based map selector. The agents and the map are represented separately in HEAT [71]. Rasterization is adopted in previous works because of its simplicity. However, it inevitably losses information provided by the high-definition map (HD map), like the connectivity of the lane segments.

A graph is a natural way to represent the relationships (e.g., either or both of inter-agent interaction and lanelet connectivity) among objects [125, 142]. ReCoG represents the relationships among agents and the map via a star-like heterogeneous graph, wherein the map node contains features extracted from a raster image [83]. HEAT represents the relationships among different kinds of agents using a heterogeneous edge-featured graph. Map information is considered via another channel [71]. VectorNet represents the driving scene with a hierarchical heterogeneous graph, where a higher-level node contains either an agent's trajectory or a map feature, and a lower-level node is a vector in a sequence (polyline). The graph is constructed with full connection ignoring the connectivity among vectors. The fully-connected graph structure is inefficient since the number of edges in the graph grows quadratically with the number of nodes [84]. LaneGCN represents actors and the map separately. It constructs a lane graph from the HD map preserving connectivity between lane nodes, and saves four types of connections for downstream encoders [21]. Rather than representing agents as single nodes, LaneRCNN constructs a dynamics-related lane graph for each agent, with each node containing geometric, semantic, and agent information [86].

We represent the driving scene with a heterogeneous hierarchical graph containing both agent and CCL nodes. An agent's CCLs are only connected to the agent itself, and all the surrounding agents are connected only to the target agent. This star-like connection makes the number of edges grows linearly, rather than quadratically, with the number of nodes. Each edge in the graph is assigned a type label, and the connection of the graph can be stored in a sparse manner. Rather than saving multiple edge sets and applying a specific graph neural network (GNN) for each, we utilize an edge-masking strategy to regulate information flow in the graph and apply a GNN for a subset of edge types hierarchically.

5.2.2 Multimodal Trajectory Predictions

For capturing the inherent multimodality of a moving agent's future motion, many works have been proposed to output multiple possible future trajectories of a target agent. MTP uses a convolutional neural network (CNN) to produce a fixed number of trajectories for a target agent [76] and shows that setting the model

to produce three modalities gives the best performance on their metrics. MultiPath first clusters trajectories in the training set via K-means to obtain a set of anchor trajectories then reformulates the task into anchor selection and regression [59]. CoverNet [143] dynamically generates a trajectory set according to the target agent's current dynamic state, then follows the anchor selection and regression operations introduced in MultiPath [59]. TNT first predicts a fixed number of targets (final points in a fixed prediction horizon) for a given agent, then estimates trajectories conditioned on the targets, and finally scores and selects a fixed number of trajectories as the final prediction [124]. Heatmaps are also used in some recent works for goal generation [22, 144].

There are also some generative methods for multimodal prediction using conditional variational auto-encoders (CVAE) or Generative Adversarial Networks (GAN) [103, 145, 146]. However, they are not quite related to our method.

Rather than selecting a fixed number of anchors or targets for all driving scenarios, Goal-Net [61] predicts a variable number of trajectories that are adaptive to the road structure. Specifically, it first generates goal paths for a target agent, where the number of goal paths is determined by the number of centerlines of the mapped lanes, then constructs a graph to represent the relationship between the target agent and its goal paths and applies a GNN to predict goal-based trajectories. Besides, it also has a separate motion-based prediction channel to capture non-map-compliant behaviors. The map-adaptive nature of the Goal-Net allows it to handle different mapped lane topologies. However, GoalNet ignores the vehicles behind the target and those in other lanes, which can have a critical impact on the target vehicle's behavior. For example, the behavior of a target vehicle operating an unprotected left turn will be affected by the vehicles in the lane on its left side. However, GoalNet does not consider this information. In addition, the goal-free prediction, although also named as motion-based, is not purely based on the target vehicle's past motion, thus not able to cover the critical corner case where a vehicle just maintain its motion, which can be in either normal (e.g., cruising) or critical situations (e.g., brake failure). However, the assumption that a vehicle will always follow a given lane ignores the fact that a vehicle sometimes navigates through references not covered by a single centerline, as stated in [147].

To address the above-mentioned limitations of GoalNet, we propose a map-adaptive

multimodal trajectory predictor that can predict single-CCL, cross-CCL, and motion-based trajectories of a target agent simultaneously based on a comprehensive representation of all nearby agents and CCLs. CCL-based predictions condition the target agent's future motions on individual CCLs, while the cross-CCL prediction tries to produce an overall prediction considering all the information in case a specific CCL cannot explain the target agent's behavior or there is no CCL can be retrieved from the lane graph. The motion-based prediction extrapolates the actor's current motion into the future to capture non-map-compliant behaviors, which is a corner case critical for safety.

5.3 Structure Overview

This section introduces the high-level architecture of the proposed map-adaptive multimodal trajectory predictor. The predictor takes as input both the historical states of multiple agents and their CCLs retrieved from the HD map and then outputs a variable number of possible future trajectories of a target agent. The number of predictions depends on the number of the target agent's CCLs. Given the input, our framework first represents the input as a heterogeneous hierarchical graph (scene graph). Then it encodes the scene graph with a hierarchical graph operator. Next, it applies a map-adaptive graph operator for multimodality. Finally, a decoder is applied to each modality to produce the final trajectories. See Fig. 5.1 for an illustration of the pipeline.

Input and output. This work aims to predict a set of multimodal trajectories of a target agent given the agents' dynamics and the local map. At a time t, the input $\mathbf{X}_t$ contains historical states of considered agents and their CCLs:

$$\mathbf{X}_t = [\mathcal{H}_t, \mathcal{C}_t], \tag{5.1}$$

where $\mathcal{H}_t = \{h_t^1, h_t^2, \ldots, h_t^n\}$ contains the historical states of n agents at time t and $\mathcal{C}_t = \{c_t^{1,1}, c_t^{1,2}, \ldots, c_t^{2,1}, c_t^{2,2}, \ldots, c_t^{n,m-1}, c_t^{n,m}\}$ contains the CCLs of each agent. $h_t^i = [s_{t-T_h+1}^i, s_{t-T_h+2}^i, \ldots, s_t^i]$ is the historical states of agent i over a traceback horizon T_h, where $s_t^i = [x_t^i, y_t^i, v_{xt}{}^i, v_{yt}{}^i]$ is the states (position and velocity) of agent i at time t. $c_t^{i,j} = [(x,y)_1^{i,j}, (x,y)_2^{i,j}, \ldots, (x,y)_{20}^{i,j}]$ is the j_{th} CCL of agent i at time t

that contains 20 way-points. Note that the number of considered agents n and the number of CCLs of an agent m vary from case to case.

Without loss of generality, we number the target agent with one. Then the output is a set of its future trajectories:

$$\mathcal{F}_t = \{f_t^{1,1}, f_t^{1,2}, \ldots, f_t^{1,m}, f_t^{1,m+1}, f_t^{1,m+2}\}, \tag{5.2}$$

where $f_t^{1,j} = [(x_{t+1}^{1,j}, y_{t+1}^{1,j}), \ldots, (x_{t+T_f}^{1,j}, y_{t+T_f}^{1,j})]$ is j_{th} sequence of predicted XY coordinates of the target agent over a prediction horizon T_f. The first m predictions are based on the target agent's m CCLs. $f_t^{1,m+1}$ is the cross-CCL prediction and $f_t^{1,m+2}$ is the motion-based prediction. The predicted trajectories are further described in Sub.Sec. 5.4.4.

Heterogeous hierarchical scene graph. Given the agents and their CCLs, we represent their relationships with a heterogeneous hierarchical graph. A node in the graph is either an agent or a CCL of an agent. To keep the connection sparse, the CCL nodes of an agent are only connected to the agent node itself, and all the surrounding agents are only connected to the target agent node. We adopt a distance threshold [80] to select surrounding agents. That is, only agents whose distances to the target vehicle are below a threshold are considered surrounding agents. Specifically, the threshold is set to 50 meters in implementation. Each raw node feature is first processed by a corresponding Recurrent Neural Network (RNN). Then an agent node contains its dynamics feature, and a CCL node contains its sequential feature accordingly. We also introduce a virtual target node into the graph to preserve the dynamics feature of the target agent from graph operation for motion-based prediction.

Hierarchical graph operator. We design a three-stage graph operator with information flow regulation to encode the scene graph. The information flow is regulated by an edge-masking technology that masks out certain edges in the graph before graph operation. The first stage lets information flows from surrounding agents' CCLs to the surrounding agents. Then the second stage lets information flows from surrounding agents to the target agent. The third stage lets the target agent collect information about its CCLs. These stages are implemented by applying a graph operator on the graph with masked edge indexes. After this operation,

the target agent node collects information about its surrounding agents and its own options.

Map-adaptive trajectory predictor. Rather than implicitly account for multimodal behaviors by generating a fixed number of future trajectories via multimodal regression, we explicitly link driving modalities with CCL options and propose to predict a set of future trajectories of a target agent according to the number of its CCLs. This is realized via graph representation and operation. In addition to predictions relying on CCLs, the predictor also produces a motion-based prediction concurrently to cover corner cases. The motion-based prediction is integrated into the graph representation and operation by introducing a virtual target node into the graph. Besides adding a virtual target node to the graph, no further operation is needed for motion-based prediction. This is because that graph neural networks (GNNs) process all nodes at the same time, no matter whether the node is connected to others or not.

5.4 Method

In this section, we describe our map-adaptive trajectory prediction method in detail. We first show how we represent the driving context as a heterogeneous hierarchical graph (5.4.1) and how we process raw node features for the usage of the downstream hierarchical graph operator (5.4.2). Then we describe the hierarchical graph operator (5.4.3) and the prediction decoder (5.4.4), followed by a short introduction to the loss used for training (5.4.5).

5.4.1 Heterogeneous Hierarchical Scene Graph

In this work, we represent the driving scene as a heterogeneous hierarchical graph, where the nodes and edges fall into different categories. The hierarchical graph contains two layers, where the lower layer is the agent-CCL graph and the upper layer is the inter-agent interaction graph. The agent-CCL graph is a star-like graph with the agent at the center and all its CCLs linked to it (indicated by deep gray arrows in the second block of Fig. 5.1). The interaction graph is another star-like graph with the target agent at the center and all neighboring nodes linked to it

(indicated by light gray arrows in the second block of Fig. 5.1). In addition to the objects in the driving scene, we introduce a virtual target agent node (light green node with a dashed border in the second block of Fig. 5.1) for motion-based prediction. The virtual node is isolated in the graph and has no CCL nodes to form a sub-graph. We also assume that each node in the graph has a self-loop for information preservation. But, for clarity, these self-loops are not plotted. The graph contains five kinds of nodes and many kinds of edges.

There are many advantages of this representation: 1) the graph representation can accommodate an arbitrary number of objects; 2) the heterogeneous graph can comprehensively represent different kinds of objects; 3) the star-like graph structure is sparse so that it is more efficient compared to graphs with dense connectivity [124]; 4) the hierarchical structure allows information flow from local to global; 5) the introduced virtual node preserves the target agent's dynamics for motion-based prediction.

Candidate centerlines selection. The Argoverse dataset provides centerline segments and their connectivity. It also provides a map API (Application Programming Interface) to interact with the HD map. We can get a vehicle's CCLs given its latest trajectory with this API. For more details of this map API, please refer to their released code.

Graph construction. We construct a heterogeneous hierarchical graph to represent the interdependencies among agents and CCLs. The graph contains two types of objects (agents and CCLs), and they are further divided into four types of nodes (target agent, surrounding agents, target agent's CCLs, and surrounding agents' CCLs). In addition to these nodes, we introduce a virtual target node in the constructed graph to integrate motion-based prediction. For an agent node, the raw node feature is the agent's historical states. For a CCL node, the raw node feature is a sequence of XY coordinates of this CCL. A directed edge pointing from node j to node i means that node j has an impact on node i, and there will be information flow from node j to node i. Each edge is associated with an edge type determined by the edge's source and target nodes. The edge set is represented as:

$$E = \{e_{ij}\}_{(j \in \mathcal{N}_i)}, \quad i = 1, \ldots, N, \tag{5.3}$$

TABLE 5.1: Nodes and edges in the scene graph

TarAg	Target agent node
VirTarAg	Virtual target agent node
SurAg	Surrounding agent node
TarCCL	Target agent's CCL node
SurCCL	Surrounding agent's CCL node
TarAg-Loop	Self-loop of the *TarAg* node
VirTarAg-Loop	Self-loop of the *VirTarAg* node
SurAg-Loop	Self-loop of the *SurAg* node
TarCCL-Loop	Self-loop of the *TarCCL* node
SurCCL-Loop	Self-loop of the *SurCCL* node
SurCCL→*SurAg*	Edge from *SurCCL* node to *SurAg* node
SurAg→*TarAg*	Edge from *SurAg* node to *TarAg* node
TarCCL→*TarAg*	Edge from *TarCCL* node to *TarAg* node
TarAg→*TarCCL*	Edge from *TarAg* node to *TarCCL* node

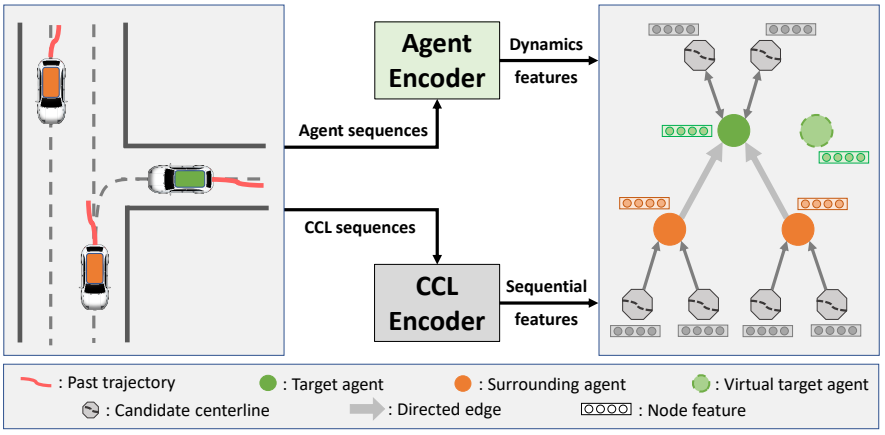

FIGURE 5.2: **Agent and CCL encoders.** Given the agents' historical states and CCLs, we apply agent and CCL encoders to extract sequential dependence in corresponding sequences. Then we take the extracted features as node features of the scene graph.

where e_{ij} is a directed edge from node j (the source node) to node i (the target node), $\mathcal{N}_i$ is the neighborhood of node i, and N is the number of nodes in the graph. Self-loops (e_{ii}) are included in the edge set since a node is also counted as a neighbor of itself. An example of the constructed graph is shown in the second block of Fig. 5.1. Tab. 5.1 shows the node and edge types in this heterogeneous hierarchical graph.

5.4.2 Agent and CCL Encoders

Since there are two kinds of objects in our scene graph, i.e., vehicles and their CCLs, we introduce one shared encoder for each type. We assume that the CCLs are sequences of XY coordinates and the historical states of vehicles are sequences of their positions and velocities over the last two seconds. All coordinates are defined in the target-centered coordinate framework, whose origin is fixed at the target agent's current position and horizontal axis aligned to the target agent's current heading direction.

5.4.2.1 Agent Dynamics Encoder

An agent is represented by a sequence of its historical states. We use a Gated Recurrent Unit (GRU [110]) network to model its dynamics from its historical states:

$$r_t^i = \mathrm{GRU}_{\mathrm{agn}}(h_t^i), \tag{5.4}$$

where h_t^i is the historical sequence of vehicle node i at time t, $\mathrm{GRU}_{\mathrm{agn}}$ is the GRU network for agents dynamics encoding, and r_t^i is is the extracted temporal feature.

5.4.2.2 Candidate Centerline Encoder

A CCL is represented by a sequence of XY coordinates. We use another GRU network to model the sequential dependencies in a centerline sequence:

$$r_t^j = \mathrm{GRU}_{\mathrm{ccl}}(c_t^j), \tag{5.5}$$

where c_t^j is the way-point sequence of CCL j at time t, $\mathrm{GRU}_{\mathrm{ccl}}$ is the GRU network for centerline encoding, and r_t^j is is the extracted sequential feature. Then the extracted features $R_t = \{r_t^1, r_t^2, \ldots, r_t^i, r_t^j, \ldots, r_t^{N-1}, r_t^N\}$ are taken as node features of the scene graph. Please see Fig. 5.2 for an illustration of the sequence encoding.

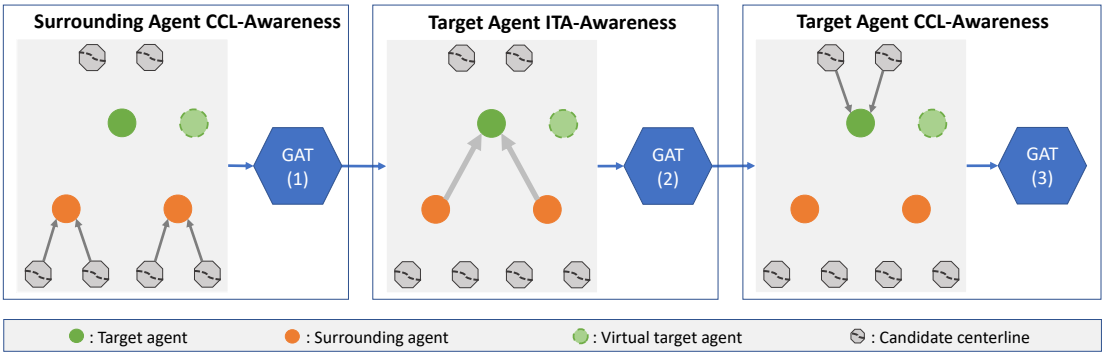

FIGURE 5.3: **Information flow in the hierarchical graph operator (HGO).** The information flow is regulated by edge-masking. The first stage of HGO is for surrounding agents' CCL-awareness. The second stage is for the target agent's interaction-awareness. The third stage is for the target agent's CCL-awareness. Each stage is implemented with a graph operation with edge-masking.

5.4.3 Hierarchical Graph Operator

In this subsection, we introduce the hierarchical graph operator (HGO) with the edge-masking technique (5.4.3.1) designed to encode the scene graph. HGO consists of three stages, namely 1) surrounding agents' CCL-awareness, 2) the target agent's interaction-awareness, and 3) the target agent's CCL-awareness. The first stage (5.4.3.2) allows the surrounding agents to gather information from their CCLs. The second stage (5.4.3.3) then allows the target agent to model its interaction with the surrounding agents. The third stage (5.4.3.4) then brings CCL-awareness to the target agent. Each stage is implemented with a separate Graph Attention Network (GAT [131]) with information flow regulated by the edge-masking technology. GAT is selected because it is widely used and powerful in even handling heterogeneous graphs. Authors of [93] prove that GAT with proper inputs can generally match or outperform existing heterogeneous GNNs across various scenarios. It is worth noting that HGO is open to being implemented with other GNNs (e.g., Graph Convolutional Networks (GCNs)). The information flow in HGO is shown in Fig. 5.3. Since GAT is utilized to implement the graph operators throughout this work, we first briefly introduce it before diving into the details.

Graph Attention Network. In this work, we want to model the effects of a target vehicle's surrounding agents and CCLs on its future motion and represent this relationship as a graph. Many GNNs are designed to apply neural networks to graph learning tasks. We select GAT [131] considering that: 1) its effectiveness has

been proven in many graph learning tasks; 2) it operates in the local neighborhood; 3) its attention mechanism allows us to model the importance of different factors.

For a node i, a GAT layer first computes attention coefficients over its neighborhood:

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

where $\vec{h}_i$ is the node feature of node i, $\vec{h}_j$ the node feature of node i's neighboring node j, $\mathbf{W}$ a shared linear transformation applied to every node, $\vec{\mathbf{a}}^T$ an attention mechanism implemented with a single-layer fully-connected network, LeakeyReLU the used nonlinearity, and $\mathcal{N}_i$ the neighborhood of node i. Then it updates the feature of node i via a linear combination of features of neighboring nodes according to the normalized attention coefficients:

$$\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}_h\vec{h}_i\right), \tag{5.7}$$

where $\mathbf{W}_h$ is the linear transformation matrix, and σ is the sigmoid function. Please note that GAT also supports multi-head attention for learning stabilization. In a GAT operation, information only flows along directed edges so that it can be regulated by manipulating the edge set. For details of the GAT layer, please refer to [131].

5.4.3.1 Edge-Masking

Data masking is a technique to hide irrelevant information from a model so that it can be trained on more relevant data for a specific task. This technique is widely used in language and vision models and shows promising improvements [148, 149]. Inspired by data masking, we propose edge-masking, a technique that hides irrelevant edges of a graph before processing it with a GNN, to train task-specific GNNs. In the setting of GNNs, information flows from one node to another through the edge between them. The information in node j can only be passed to node i through GNN operations if there is an edge from j to i. So we can regulate information flow from nodes to nodes (which can be of different types) by hiding irrelevant edges via edge-masking to train GNNs for specific sub-tasks. This is different from HetGNN [150], which applies a GNN for each type of edge connection.

Using the edge-masking technique, we only need to save one edge set with several edge masks for each graph operator.

5.4.3.2 Surrounding Vehicles' CCL-Awareness

Before modeling interactions between the target and its surrounding agents, we first let these surrounding agents gather information from their own CCLs. This operation, when modeling inter-agent interactions in the following stage, gives the target agent a broader view of the road structure and possible motions of its surrounding agents. We apply a GAT to the entire graph with edge-masking to regulate information flow in this graph operation so that the information only flows from surrounding agents' CCL nodes to themselves:

$$G_t^1 = \mathrm{GAT}_1(R_t, E_1), \tag{5.8}$$

where R_t contains node features for both agent and CLL nodes, and E_1 is the edge set retrieved via masking for this stage. GAT_1 is the GAT for this stage, and G_t^1 is the output of this stage. Each surrounding agent node in G_t^1 is with CCL-awareness. All the other nodes, i.e., the target, the virtual target, and all the centerline nodes, remain isolated. The information flow regulated by edge-masking is shown in the first block of Fig. 5.3. Specifically, the edges of the following types are used in this graph operator: {*SurCCL→SurAg*, *TarAg-Loop*, *SurAg-Loop*, *TarCCL-Loop*, *VirTarAg-Loop*}.

5.4.3.3 Target Vehicle's Interaction-Awareness

In the second stage, we allow the target agent to gather information from its neighborhood. Since its neighboring agents are aware of their corresponding CCLs, this stage provides interaction-awareness to the target vehicle along with further road-awareness from its neighbors:

$$G_t^2 = \mathrm{GAT}_2(G_t^1, E_2), \tag{5.9}$$

where G_t^1 is the output of Eq. 5.8, E_2 the edge set retrieved via masking for this stage, GAT_2 the GAT for this stage, and G_t^2 the output of this stage. This stage brings interaction-awareness (ITA-awareness) to the target agent node. All the

other nodes, i.e., the surrounding agents, the virtual target, and all the CCL nodes, remain isolated. The information flow regulated by edge-masking is shown in the second block of Fig. 5.3. Specifically, the edges of the following types are used in this stage: {*SurAg→TarAg*, *TarAg-Loop*, *SurAg-Loop*, *TarCCL-Loop*, *VirTarAg-Loop*}.

5.4.3.4 Target Vehicle's CCL-Awareness

The third stage is to make the target agent aware of its options, that is, its CCLs:

$$G_t^3 = \mathrm{GAT}_3(G_t^2, E_3), \tag{5.10}$$

where G_t^2 is the output of Eq. 5.9, E_3 the edge set retrieved via masking for this stage, GAT_3 the GAT for this stage, and G_t^3 the output of this stage. This stage lets the target agent look at its CCLs with knowledge of surrounding agents' options and interactions. All the other nodes, i.e., the surrounding agents, the virtual target, and all the CCL nodes, remain isolated. The information flow regulated by edge-masking is shown in the third block of Fig. 5.3. Specifically, the edges of the following types are used in this stage: {*TarCCL→TarAg*, *TarAg-Loop*, *TarCCL-Loop*, *VirTarAg-Loop*}.

5.4.4 Map-Adaptive Trajectory Predictor

Our approach utilizes a variable number of CCLs to predict three kinds of future trajectories of a vehicle of interest. The number of CCLs depends on the lane geometry of the driving scene, and the predicted trajectories include single-CCL, cross-CCL, and motion-based predictions. This design is based on the following observations: 1) the road structure mainly shapes the motion of vehicles, and the vehicles tend to follow centerlines when driving to keep a safe distance from each other; 2) there are some situations where a vehicle will drive along with a combination of two or more centerlines; 3) the motion of a vehicle can purely depend on its own dynamics in some corner-cases.

To handle the variable number of CCLs, we adopt the graph representation and a GNN in this predictor. Fig. 5.4 shows an illustration of this predictor. The graph

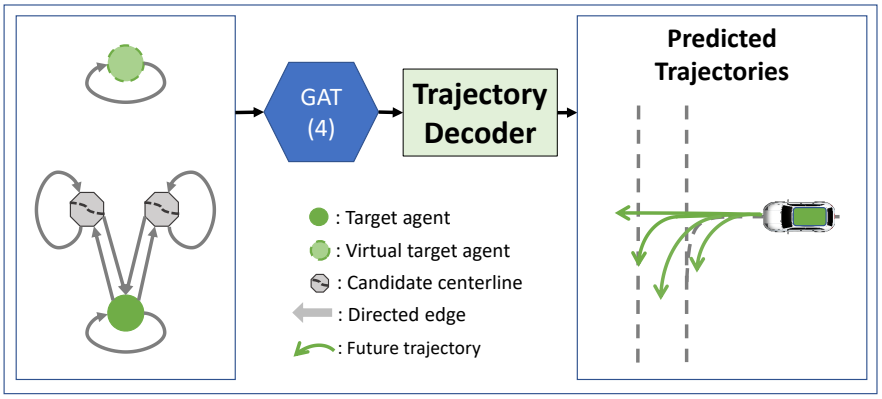

FIGURE 5.4: **Map-adaptive trajectory predictor.** After encoding, we apply another GAT on the graph with edges shown in the left block of this figure. This is to distribute the target agent's feature to the CCL nodes and let the target agent node reconsider its options (CCLs). Then we apply a trajectory decoder to output the final multimodal prediction.

structure used by this predictor is shown in the left block of Fig. 5.4, which is a heterogeneous graph containing three types of nodes: a target node, a virtual target node, and a set of CCL nodes of the target vehicle. The graph structure is also obtained via edge-masking. Throughout all the previous encoding stages with our information flow regulation strategy, the node features are updated and contain corresponding features for three types of predictions. The target node contains the overall information of the scene; the virtual target node contains its own dynamics; the target vehicle's CCL nodes contain corresponding CCL features. Since we are focusing on the target agent, all surrounding agents and their CCL nodes are ignored in this part. Given the number of the target vehicle's CCLs, our predictor will output $m + 2$ predictions:

$$\mathcal{F}_t = \mathrm{MLP}_{\mathrm{pred}}\left(\mathrm{GAT}_{\mathrm{pred}}(G_t^3, E_4), Mask_{tar}\right), \tag{5.11}$$

where G_t^3 is the output of Eq. 5.10, E_4 the edge set retrieved for this stage via masking, $\mathrm{GAT}_{\mathrm{pred}}$ the GAT used for prediction. $Mask_{tar}$ is used to select the target agent node and the target CCL nodes from the output of $\mathrm{GAT}_{\mathrm{pred}}$. $\mathrm{MLP}_{\mathrm{pred}}$ is the trajectory decoder implemented with a multilayer perceptron (MLP), and $\mathcal{F}_t$ is the predicted future trajectories of the target agent. $\mathcal{F}_t$ contains m single-CCL predictions, one cross-CCL prediction, and one motion-based prediction. Specifically, the edges of the following types are used in this graph operator: $\{TarCCL \rightarrow TarAg,$ $TarAg \rightarrow TarCCL,$ $TarAg\text{-}Loop,$ $TarCCL\text{-}Loop,$ $VirTarAg\text{-}Loop\}$.

5.4.5 Modified MTP Loss for This Work

We modify the Multiple-Trajectory Prediction (MTP) loss proposed in [76] to train our map-adaptive prediction framework in an end-to-end way. The modified MTP loss takes as input a set of predicted trajectories and one ground truth trajectory of the target agent. Unlike the original MTP loss, the modified MTP loss minimizes the regression loss only. It first selects the predicted trajectory with the smallest average L2 distance to the ground truth as the best mode, then calculates the smoothed L1 loss between the best prediction and the ground truth trajectory. For more details about the MTP loss, please refer to [76].

5.5 Real-World Dataset Validation

In this section, we first introduce the large-scale real-world dataset (5.5.1) and metrics (5.5.2) used for validation. Then we compare our method with existing methods on the Argoverse motion forecasting benchmark (5.5.3). Next, we show the results of our ablation studies (5.5.4) followed by visualized predictions (5.5.5) and the implementation details of this work (5.5.6).

5.5.1 Argoverse Motion Forecasting Dataset

The proposed scheme is trained and validated using the recently published Argoverse motion forecasting dataset [51]. The dataset consists of 327,790 interesting driving scenarios extracted from over 1,000 driving hours. Each scenario is a 5-second long segment and contains the 2D, birds-eye-view centroid of each tracked object sampled at 10 Hz. The first 2 seconds are observation and the next 3 seconds are to be predicted. The dataset also provides an HD map for each scenario along with a map API for easily using the map information (e.g., get the CCLs of an agent given its past trajectory). The dataset is split into 208,272 training sequences, 40,127 validation sequences, and 79,391 test sequences.

Since this work focuses on trajectory prediction for a single target agent, we use a shared coordinate system for all the trajectories and waypoints. The origin of the shared frame of reference is fixed at the target agent's current position,

and the horizontal axis is aligned with the target agent's heading direction. This makes our method robust to coordinate transformation since data represented in other coordinate systems can always be transformed to be target-centric. For an agent node, its raw node feature is the historical states (a 20×4 array containing $[x, y, v_x, v_y]$ at each time step) of the agent over the past two seconds. Note that the velocity ($[v_x, v_y]$) is calculated rather than provided by the dataset. For a CCL node, its raw node feature is a sequence of waypoints sampled from the CCL. All the CCLs are reshaped into a 20×2 array.

5.5.2 The Evaluation Metrics

We adopt Average Displacement Error (ADE) and Final Displacement Error (FDE) in meters to evaluate prediction performances. ADE and FDE are widely used in previous works [83, 124], and they are calculated as below:

$$ADE = \frac{1}{T_f} \sum_{\tau=1}^{T_f} \sqrt{(\hat{x}_\tau - x_\tau)^2 + (\hat{y}_\tau - y_\tau)^2}, \tag{5.12}$$

$$FDE = \sqrt{(\hat{x}_{T_f} - x_{T_f})^2 + (\hat{y}_{T_f} - y_{T_f})^2}, \tag{5.13}$$

where $(\hat{x}_\tau, \hat{y}_\tau)$ and (x_τ, y_τ) are the predicted and ground truth positions in XY coordinates at time τ, respectively, and T_f is the prediction horizon. We evaluate the multimodal prediction with minimum ADE (minADE) and minimum FDE (minFDE) over multiple modalities.

5.5.3 Comparison with existing Methods

We compare our model with existing models proposed in recent years and two official baselines provided by the Argoverse dataset on the Argoverse motion forecasting benchmark. The benchmark allows up to six predictions so that we replace our map-adaptive decoder with a decoder that predicts six possible future trajectories of a target agent for a fair comparison. We report the results in terms of minADE, minFDE, and miss rate (MR). Miss rate is the number of scenarios where none of the predicted trajectories of a target agent are within 2.0 meters of the ground truth according to endpoint error [152]. The MR is listed, but the comparison is mainly

TABLE 5.2: Performance on Argoverse motion forecasting benchmark (test set)

	Methods	K=1			K=6		
		minADE	***minFDE***	***MR***	***minADE***	***minFDE***	***MR***
1	Argoverse Baseline [51]	2.96	6.81	0.81	2.34	5.44	0.69
2	Argoverse Baseline (NN) [51]	3.45	7.88	0.87	1.71	3.29	0.54
3	Holmes [151]	2.91	6.54	0.82	1.38	2.66	0.42
4	cxx [19]	1.91	4.31	0.66	0.99	1.71	0.19
5	Jean [19, 89]	1.86	4.18	0.63	0.93	1.49	0.19
6	LaneGCN [21]	1.71	3.78	0.59	0.87	1.36	0.16
7	LaneRCNN [86]	1.68	3.69	0.59	0.90	1.45	0.12
8	DenseTNT [22]	-	-	-	0.94	1.49	0.11
9	TNT [124]	2.17	4.96	0.71	0.91	1.44	0.17
10	GOHOME [144]	-	3.65	0.57	0.94	1.45	0.11
11	**Ours**	1.88	4.18	0.64	0.89	1.40	0.17

on minADE and minFDE. It can be seen from Tab. 5.2 that our model significantly outperforms both baselines and the Holmes method. Our model also outperforms cxx (the third-place winner of the Argovese motion forecasting challenge 2019), Jean (the first-place winner of the Argovese motion forecasting challenge 2019), LaneRCNN, TNT, DenseTNT, and GOHOME in terms of minADE and minFDE. The performance of our model matches that of LaneGCN, which is the first-place winner of the Argoverse motion forecasting challenge 2020. These results show the effectiveness of the proposed scene representation and encoding approach.

5.5.4 Ablative Studies

We conduct ablative studies to show the importance of each module (g1, g2, g3) and the superiority of the proposed approach compared to a single GAT (G). In Tab. 5.2, r, if checked, means that the model uses RNN (GRU) to encode sequences; each of the graph operators (g1, g2, and g3), if checked, means that the model uses the checked encoder(s); G, if checked, means that the model uses a single GNN to encode the entire hierarchical heterogeneous graph. All the ablative models are implemented with the proposed map-adaptive predictor (5.4.4). The number of predicted trajectories adapts to the number of CCLs of the target agent. They differ from each other mainly in the encoder part:

- **cl-r:** The simplest model that encodes only the target agent's past dynamics using an RNN for prediction. No interaction and lane information is considered.

TABLE 5.3: Ablative study results

	Methods	r	g1	g2	g3	G	minADE	minFDE
1	cl-r	✓					0.883 ± 0.008	1.716 ± 0.017
2	cl-r-g2	✓		✓			0.827 ± 0.01	1.561 ± 0.019
3	cl-r-g3	✓			✓		0.838 ± 0.015	1.602 ± 0.033
4	cl-r-g1g2	✓	✓	✓			0.817 ± 0.017	1.533 ± 0.042
5	cl-r-g2g3	✓		✓	✓		0.798 ± 0.014	1.487 ± 0.036
6	cl-r-G	✓				✓	0.817 ± 0.011	1.549 ± 0.029
7	cl-r-g1g2g3	✓	✓	✓	✓		0.783 ± 0.003	1.448 ± 0.006

- **cl-r-g2:** The model that encodes all vehicles' dynamics and their interactions using RNN and GAT (g2), respectively. No lane information is considered in this model.
- **cl-r-g3:** The model that encodes the target agent's past dynamics and its CCLs using different RNNs and realizes the target vehicle's CCL-awareness using a graph operation (g3). No interaction is considered.
- **cl-r-g1g2:** The model that encodes all vehicles' dynamics and the surrounding agents' CCLs, then updates surrounding agents' features with their CCLs information using a GAT (g1), next models their interaction with the target vehicle using another GAT(g2). The target agent's CCLs are not considered in the encoding part.
- **cl-r-g2g3:** The model that first encodes all vehicles' dynamics and the target agent's CCLs using RNNs (r), then models inter-agent interaction using a GAT (g2), next updates the target node with its CCLs information using another GAT (g3). Surrounding agents' CCLs are not considered.
- **cl-r-G:** The model that encodes all vehicles' dynamics using RNNs and models the complex relationship, represented by a heterogeneous hierarchical graph, using a one-layer GAT (G). This model has access to the target vehicle's CCLs and surrounding vehicles.
- **cl-r-g1g2g3:** The proposed model.

The results of the implemented models are shown in Tab. 5.3, where we run each model five times and report the mean and standard deviation of their minADE and minFDE. It can be observed that as we progressively add more graph operators,

both minADE and minFDE decrease and the proposed method shows the most stable performance. This demonstrates the effectiveness of our framework. Besides, there are some more observations we can draw from Tab. 5.3: 1) comparing row 2 and row 3 with row 1, we can see that both the target agent's interaction-awareness (g2) and overall CCL-awareness (g3) improve the performance, respectively; 2) comparing row 4 with row 2, we can observe that the surrounding agents' CCL awareness (g1) is useful for trajectory prediction; 3) comparing row 5 with row 2 and row 3, we can see that combining the target agent's interaction awareness (g2) and its overall CCL awareness (g3) improves the performance; 4) comparing row 5 with row 6, we can observe that our framework with hierarchical graph operator (g2g3) outperforms the model using a single GAT (G) to model a heterogeneous hierarchical graph, despite both having access to the same data in the graph. 5) Comparing row 7 with the rest of the rows, we can tell that the proposed hierarchical graph operator outperforms all its ablations. This shows the effectiveness of the proposed HGO.

We also show the minADE and minFDE results of all ablative models in Fig. 5.5 for an intuitive comparison. The results of one ablative model are shown in the same color. The top of Fig. 5.5 shows the minADE results, and the bottom shows the minFDE results. The average values over five experiments are shown in colored bars, and the standard deviations are shown at the top of the corresponding bars. It can be seen that the proposed model outperforms all the ablative models and achieves the best minADE and minFDE in a more stable way.

5.5.5 The Visualized Results

In Fig. 5.6, we show that our method is able to predict a variable number of trajectories of a target agent according to its CCLs. It can be seen that our method is able to predict three types of trajectories. The left part of Fig. 5.6 shows the case where the target vehicle has only one CCL. It can be seen that motion-based prediction extrapolates its historical track, and the cross-CCL prediction is the closest to the ground truth. The middle part shows the case where the target vehicle has two CCLs. It can be seen that the proposed model is able to generate four predictions according to the number of CCLs, and one of the single-CCL predictions is closest to the ground truth. The right part shows the case where

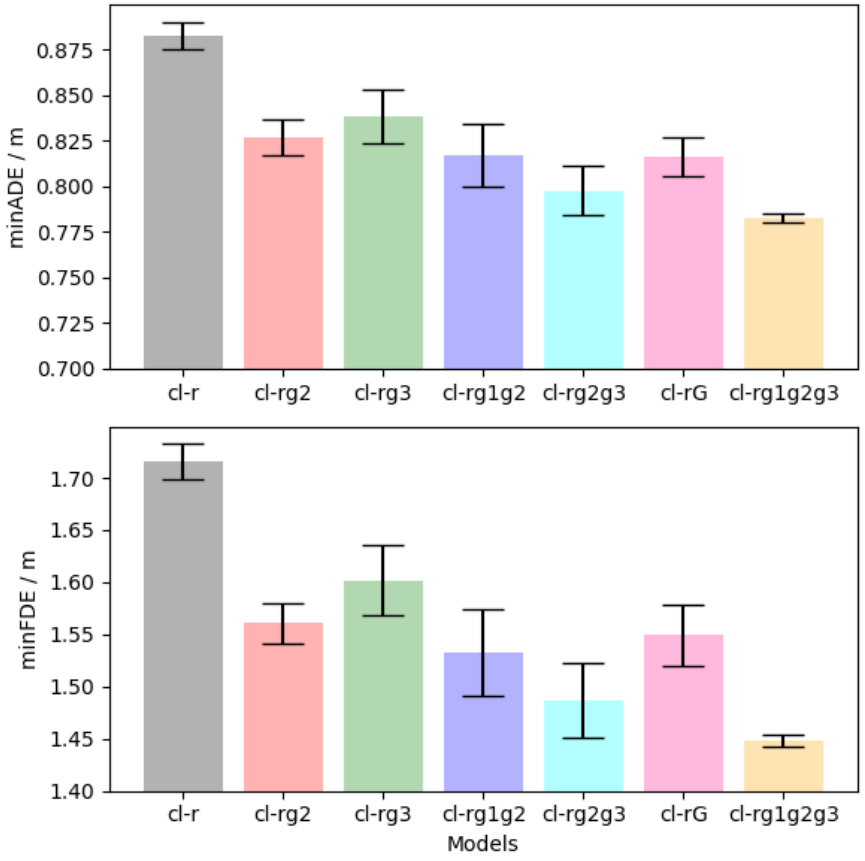

FIGURE 5.5: **Minimum ADE and FDE of ablative models.** *Top*, the minADE results of all ablative models over five experiments. *Bottom*, the minFDE results of all ablative models over five experiments. Standard deviations are shown at the top of the bars.

the target vehicle has three CCLs. It can be seen that single-CCL predictions are aligned with CCLs, and the motion-based prediction is closest to the ground truth. Three cases in Fig. 5.6 together show that our method is able to predict a variable number of trajectories according to different situations (different numbers of CCLs). Fig. 5.6 also demonstrates the necessity to produce three kinds of predictions (e.g., single-CCL, cross-CCL, and motion-based predictions) since the best performance (the closest to the ground truth) is achieved by different kinds of predictions in different situations.

5.5.6 The Implementation Details

The proposed model and its ablations are implemented using PyTorch [115] and PyTorch Geometric [116]. All the models can be trained end-to-end, and they are

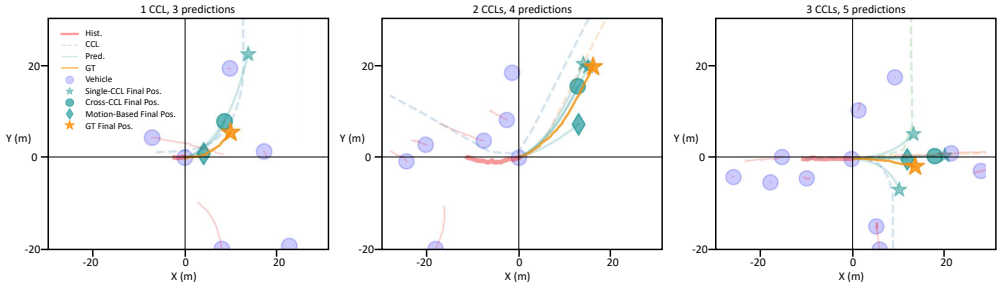

FIGURE 5.6: **Visualized prediction results.** This figure showcases some scenarios with different numbers of CCLs selected from the validation set of the Argoverse motion forecasting dataset. Light blue dots (Vehicle) show the current positions of all the agents, and red curves show their corresponding historical trajectories (Hist.). Dashed curves (CCL) show the CCLs of the target agent. Yellow curves (GT) and stars (GT Final Pos.) show the target agent's ground truth trajectory and final position over the prediction horizon, respectively. Green stars (Single-CCL Final Pos.) show the final positions of the single-CCL predictions. Green dots (Cross-CCL Final Pos.) show that of the cross-CCL prediction. Green diamonds (Motion-Based Final Pos.) show that of the motion-based predictions. Green curves (Pred.) followed by predicted final positions are their corresponding trajectories.

trained for 50 epochs using Adam as the optimizer. The learning rate of Adam starts from 0.002 and decreases by half at the ends of epochs: 1, 6, 12, 18, 24, and 30. For the proposed model, the raw agent and CCL features are first embedded into a 32-dimensional space separately before being sent to their corresponding GRU encoders. The hidden sizes of both GRU encoders are 64. The GATs in the hierarchical graph operator are one-layer GATs with three attention heads, and their hidden sizes are all set to 128. The GAT in the map-adaptive trajectory decoder uses only a single attention head. Finally, a two-layer MLP is used to produce the trajectories. LeakyReLU is used as the activation function between layers.

5.6 Discussions

Very little was found in the literature on the question of map-adaptive trajectory prediction [61], despite the fact that map-adaptability is essential to the generalization of these methods. This study aims to design a map-adaptive multimodal trajectory predictor that can represent driving scenes in a unified form and account for map-compliant and non-map-compliant driving behaviors in a single graph operation. Numerical results show that the proposed method achieves competitive

performance on a large-scale real-world dataset and outperforms all its ablations. Visualization results show that the proposed method is map-adaptive. We find that all three factors (surrounding vehicles' CCL-awareness, the target vehicle's interaction-awareness, and its CCL-awareness) positively affect the prediction performance. However, an inappropriate encoder can impair performance. For example, cl-r-G performs much worse than cl-r-g2g3 despite using the same input. This is because that cl-r-G does not distinguish node and edge types in the heterogeneous interaction graph.

We provide more descriptions of existing methods listed in Tab. 5.2. Jean [19, 89], in their presentation, uses an attention mechanism to allow the vehicle to attend to centerline segments and then uses another attention mechanism to model inter-vehicular interactions. It can be seen from the slides that Jean is not using CCLs, and all the vehicles share the same centerline segments. In contrast, our model utilizes CCLs provided by Argoverse so that each vehicle has its CCLs retrieved according to its past trajectory. The method called cxx [19] encodes trajectories and velocity sequences using standard Long Short-Term Memory (LSTM [66]) and encodes lanes using 1-D CNNs. Then it uses an attention mechanism to fuse the dynamics and lane features. No inter-vehicular interaction is considered explicitly. LaneRCNN [86] constructs a lane graph for each actor according to their dynamics and embeds each node feature with geometric, semantic, and actor information. It then scores the lane nodes, selects K lane nodes as target waypoints, and produces the final prediction via interpolation and refinement. Since there could be many lanes in the lane graph, scoring and selecting target nodes is still challenging. LaneGCN [21] constructs a lane graph from the HD map, designs a special GCN to consider dilated connections in the graph, and exploits a fusion network to capture four types of interactions. The actor and map are represented separately in LaneGCN, which is not comprehensive and can hardly generalize to accommodate new types of information, such as traffic signals. LaneGCN introduces multiple adjacency matrices for the lane graph, which is memory-consuming. We comprehensively represent the actor and map information with a heterogeneous hierarchical graph, and regulate the information flow via an edge-masking strategy. Instead of saving a graph for each edge type, our framework allows selecting a subset of edge types for message passing. It is more flexible for heterogeneous graphs than saving multiple adjacency matrices. TNT [124] uses VectorNet [84] to encode the driving scene and proposes a three-stage prediction framework. It first predicts a set

of target waypoints sampled from the centerlines, then estimates a trajectory for each target, next scores and selects trajectories with the highest scores as the final multimodal prediction. The three-stage framework of TNT is reasonable, but the context is represented by a fully-connected hierarchical graph, which is inefficient and makes it hard to generalize to large-scale graphs. DenseTNT [22] also uses VectorNet as the context encoder, so it shares the inefficiency of TNT caused by the fully-connected graph. GOHOME [144] constructs a road graph with lanelets utilizing geometric and connectivity information provided by the HD map. It then scores and selects lanelets with the highest probability of containing the endpoint of the target agent. Next, it generates partial heatmaps for these lanelets and then combines them into a global heatmap. Finally, it sparsely samples endpoints from the heatmap and produces a trajectory for each. Our method avoids time-consuming sampling and scoring by linking CCLs with driving modalities since CCLs provide real-time references for vehicles.

One limitation of the proposed method is that it only accounts for the spatial multimodality of driving. Temporal multimodality can be included by subdividing spatial modes as did in [58, 61] in the future. Another limitation is that the proposed method relies heavily on the CCLs. We believe that more advanced CCL identification methods can improve performance and flexibility. If we can provide CCLs that can represent not only spatial but also temporal multimodalities, we can directly extend the current method for spatio-temporal multimodal prediction.

5.7 Conclusions

In this work, we propose a map-adaptive multimodal trajectory prediction framework that can predict an agent's single-CCL, cross-CCL, and motion-based trajectories in an integrated manner. We represent the driving scene using a heterogeneous hierarchical graph and design a hierarchical graph operator with an edge-masking technology to encode the driving scene. Validation on the Argoverse motion forecasting benchmark shows that our method matches the performance of recent works with the merit of map-adaptability. In addition to map-compliant predictions, our method also considers the corner case where a vehicle's future

motion purely depends on its own motion. Considering this crucial corner case is essential for the safety of an autonomous vehicle.

In the future, we plan to incorporate probability prediction for each possible motion and temporal multimodality prediction into this framework.

Chapter 6

Predictive Motion Planning Using Neural Networks

A series of trajectory prediction methods have been developed from Chapter 3 to Chapter 5. However, the ultimate goal of trajectory prediction is to improve the downstream decision-making, planning, and control of autonomous vehicles. In this chapter, the impacts of trajectory prediction on motion planning performance are investigated. A predictive motion planner based on neural networks is proposed and leveraged for the study.

6.1 Introduction

Most works on trajectory prediction state that prediction is crucial for an autonomous vehicle to navigate in complex scenarios via planning and focus on improving prediction accuracy in different metrics. However, prediction and planning are treated separately in most recent works, even though these two modules are complexly coupled, and combing optimal modules may lead to sub-optimal overall performance. Recently, authors of [95] propose a neural motion planner trained with an additional perception/prediction loss to infuse perception and prediction information into intermediate representations. Authors of [62, 98] propose to utilize predictions explicitly to calculate collision costs of sampled trajectories. A contingency planner is proposed in [153] to generate a set of contingent trajectories for diverse predicted future scenes. These optimization-based predictive

planners [62, 98] require handcrafted parameter tuning to design a suitable cost function, which is time-consuming. Despite these studies on neural motion planning and predictive planning, the best performance a predictive planner can achieve and the relationship between prediction and planning are not well studied. In this chapter, we follow [95] and propose a predictive planner that is trained to utilize future information to produce trajectories similar to human demonstrations. We define the best performance of this planner as its performance achieved with access to ground truth future motions of other agents and study how prediction accuracy can affect its performance. The proposed predictive planner can be trained end-to-end without hand-engineered cost functions.

The remaining sections of this chapter are structured as follows. Sec. 6.2 elaborates on the proposed predictive planner and its components. Sec. 6.3 describes the experiments conducted on a real-world dataset. Sec. 6.4 reports the experimental results. Sec. 6.5 provides further discussions on the results and the proposed method. Sec. 6.6 concludes this study and outlines directions for future studies.

6.2 Method

We propose a predictive motion planner that can generate a trajectory over a predefined planning horizon. Both observed information and predicted trajectories of other vehicles are considered for planning. Our planner takes as input the observed historical states of all considered traffic participants and a local map shared by them and produces target agents' future trajectories predicted over the planning horizon as intermediate representations. The predicted trajectories improve interpretability and are explicitly taken as input to the planner. The final output of our planner is a trajectory of the ego vehicle over the prediction horizon. We train our planner with a loss encouraging it to produce trajectories similar to human demonstrations. The planner is agnostic to the choice of predictor as long as predicted trajectories of target vehicles are available. In this section, we first formulate the problem of predictive planning (6.2.1). Then we introduce how we represent and encode the driving scene (6.2.2). Next, we describe the multi-agent trajectory predictor (6.2.3) and oracle planner (6.2.4) that together makes our predictive planner (6.2.5).

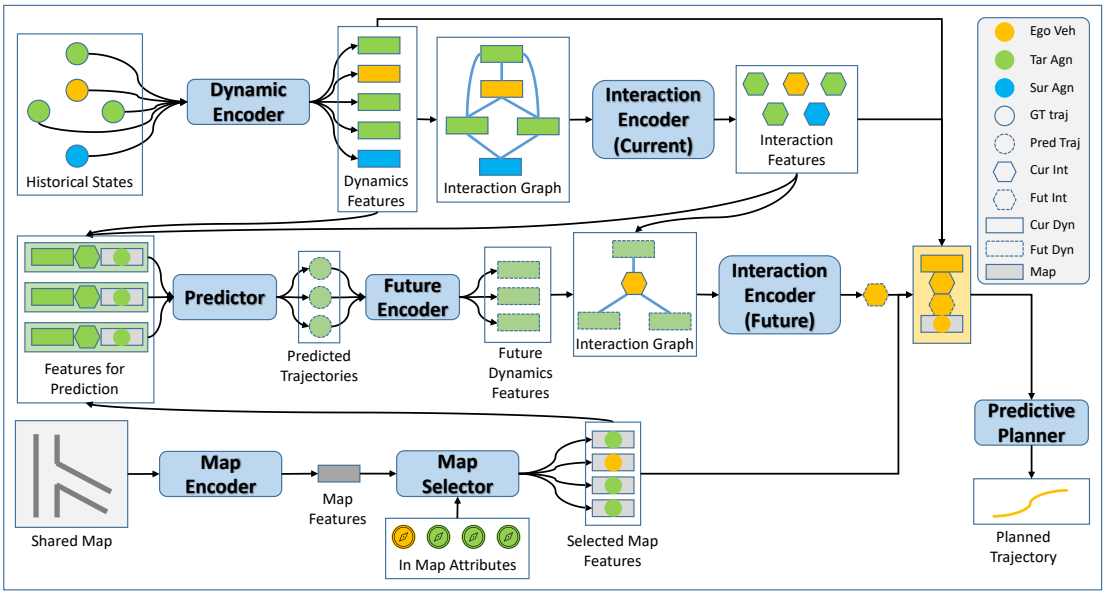

FIGURE 6.1: **The proposed predictive motion planner.** The dynamics encoder takes as input agents' historical states and outputs their individual dynamics features. An interaction graph is then constructed with nodes containing the dynamics features of corresponding agents. The interaction encoder takes as input the graph and produces interaction features of all agents. The map encoder extracts spatial features of the shared map, and then the gate-based map selector generates specified map features for agents according to their relative states on the map. Target agents' dynamics, interaction, and map features are combined as inputs to the predictor that predicts future trajectories of them. Then a future encoder, similar to a dynamics encoder, captures dynamics features from predictions. Next, an interaction graph containing the ego agent's interaction and target agents' future dynamics features is constructed and processed by another interaction encoder. Finally, the predictive planner takes as inputs the ego agent's dynamics, interaction, future interaction, and selected map features and generates a planned trajectory for the next seconds. (Ego Agn: ego agent. Tar Agn: target agent. Sur Agn: surrounding agent. GT Traj: ground truth trajectory. Pred Traj: predicted trajectory. Cur Int: current interaction feature. Fut Int: future interaction feature. Cur Dyn: current dynamics feature. Fut Dyn: future dynamics feature. Map: map feature.).

6.2.1 Problem Formulation

6.2.1.1 Agents Terminology

To define the problem of predictive planning, we divide traffic participants in a local area into three categories:

- **Ego Vehicle (EV)** is the autonomous vehicle that navigates in a local area through predictive planning.

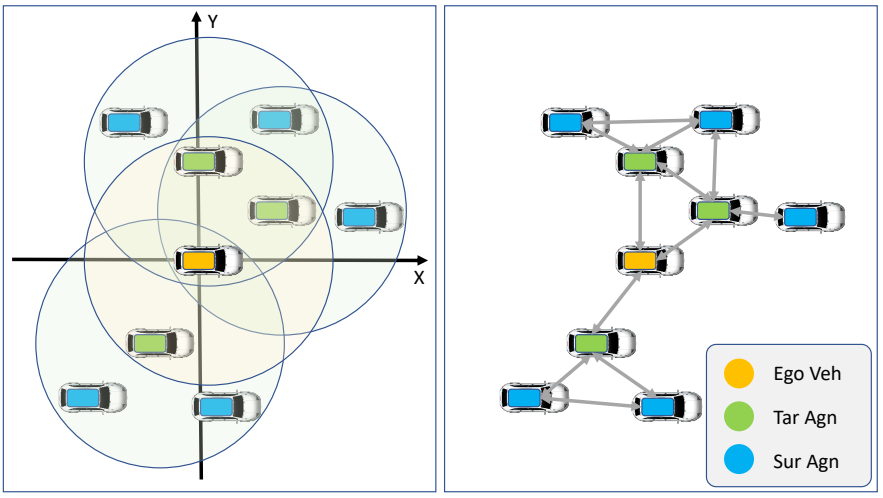

FIGURE 6.2: **Terminology and interaction graph.** *Left*, the distance-based terminology indicated by colors. The vehicles within a certain distance to the EV are considered TAs whose future trajectories need to be predicted for the EV's navigation. Other agents that fall into the neighborhood of target agents are considered to have impacts on TAs' behaviors. These agents are SAs that will be used for prediction. *Right*, the interaction graph with distance-based connections. Two vehicles are connected if the distance between them is within a threshold. Self-loops are included but not plotted for clarity.

- **Target Agents (TAs)** are the traffic participants of interest whose future trajectories are to be predicted by the EV.
- **Surrounding Agents (SAs)** are other traffic participants that have potential impacts on the TAs' future motions.

They are divided according to distance-based criteria. The left part of Fig. 6.2 shows an example of this terminology.

6.2.1.2 Frame of Reference

Considering that planning is ego-centric, we choose to use an ego-centric frame of reference and place traffic participants and the local map into the shared frame of reference for both prediction and planning. The spatial relationships among traffic participants and the local map are preserved in their states. From the EV's perspective, the frame of reference is designed with its origin fixed at the EV's current position and its horizontal axis aligned with the EV's current heading direction. This frame of reference is independent of the observation coordinate system and

can be easily applied to autonomous vehicles using onboard sensors. In addition, it normalizes the observations to a certain extent and makes the algorithms more robust to the translation and rotation of coordinate systems. An illustration of this frame of reference can be found in the left part of Fig. 6.2.

6.2.1.3 Predictive Planning

A modularized autonomous driving system can be implemented in-order with the following modules, sensing, perception, prediction, planning/decision-making, and control. This pipeline works in a periodic way. Predictive planning is to explicitly consider prediction results in the planning module. We take H_t to represent the historical states of all traffic participants (including EV, TAs, and SAs) and M_t the shared local map. Specifically,

$$H_t = \{h_t^0, h_t^1, \ldots, h_t^m, h_t^{m+1}, \ldots, h_t^n\},$$

where h_t^0 is the historical states of the EV, $h_t^1, \ldots, h_t^m$ are that of TAs, and $h_t^{m+1}, \ldots, h_t^n$ are that of SAs. Then the input to the predictive planner is $X_t = [H_t, M_t]$. The predictive planner is expected to generate a trajectory y_t^0 for the EV to navigate in the next seconds. Given the input X_t, a planner tries to capture the probabilistic distribution $P(y_t^0|X_t)$, a predictor tries to capture $P(Z_t|X_t)$, where $Z_t = [z_t^1, \ldots, z_t^m]$ represents future trajectories of the TAs. Based on the planner and predictor, we formulate our predictive planner (PrePlan) as an approximation of:

$$P(y_t^0|X_t) = P(y_t^0|Z_t, X_t)P(Z_t|X_t). \tag{6.1}$$

We also define a planner taking as input the ground truth future trajectories of TAs (Z^{gt}) for approximating $P(y_t^0|Z_t^{gt}, X_t)$. Since Z_t^{gt} is not accessible in inference, we name this planner an oracle planner (OrcPlan). We assume that the trajectories are unimodal and use deterministic mappings $y_t^0 = f_{pl}(X_t)$, $Z_t = f_{pr}(X_t)$, $y_t^0 = f_{pp}(Z_t, X_t)$, and $y_t^0 = f_{op}(Z_t^{gt}, X_t)$ for planner, predictor, predictive planner, and oracle planner, respectively. All the mappings (f_{pl}, f_{pr}, f_{pp}, and f_{op}) are approximated with neural networks in implementation. We can move on to our scene representation and encoding with the formulated problem.

6.2.2 Scene Representation and Encoding

In this work, the driving scene is represented by a shared local map and an interaction graph. The local map is a Bird's Eye View (BEV) image of the local area, and the interaction graph is a directed graph with nodes representing traffic participants. Two traffic participants are linked if their distance is below a threshold. An illustration of the interaction graph can be found in the right part of Fig. 6.2. This representation contains temporal and spatial information in different forms and needs to be processed separately with different encoders before an overall scene encoding. We design the following modules: dynamics encoder, map encoder, map distributor, and interaction encoder to obtain scene features. These modules are similar to previous works[71, 83].

6.2.2.1 Temporal Encoder

The temporal encoder is expected to capture an agent's dynamics feature from a temporal sequence of its historical states. A simple way is to flatten the sequence into a one-dimensional vector and apply a multilayer perceptron (MLP) as the encoder. MLP is easy but ignores the temporal dependence between time steps, which is essential for sequence modeling. So Recurrent Neural Networks (RNNs) are often adopted for sequence modeling in previous works. Between two widely-used RNNs, namely Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU) networks, we adopt a GRU as our temporal encoder. To achieve performance on par with LSTMs, GRUs usually need fewer parameters and simpler computation, thus providing faster training and inference [110]. The dynamics encoder is described as follows:

$$D_t = \mathrm{Dyn}_{\mathrm{enc}}(H_t), \tag{6.2}$$

where $\mathrm{Dyn}_{\mathrm{enc}}$ is the temporal encoder implemented with a two-layer GRU, and D_t represents the dynamics features of all traffic participants.

6.2.2.2 Spatial Encoder

The spatial encoder is designed to extract spatial features from the local map. Considering that direction in the map is essential information for navigation, we adopt a rotation-sensitive map feature extractor as the spatial encoder:

$$m_t = \mathrm{Spt}_{\mathrm{enc}}(M_t), \tag{6.3}$$

where $\mathrm{Spt}_{\mathrm{enc}}$ is the rotation-sensitive spatial encoder, and m_t is the spatial feature of the shared local map. The spatial encoder is implemented with a four-layer convolutional block without max-pooling layers.

6.2.2.3 Map Distributor

To share the map information between all agents, we employ a map distributor that prepares a customized spatial feature vector m_t^i for agent i. The spatial feature is customized according to agent i's state (position, velocity, and yaw angle) in the local map via a gate mechanism [71, 110]:

$$m_t^i = \mathrm{Spt}_{\mathrm{gate}}(m_t, s_t^i), \tag{6.4}$$

where s_t^i is agent i's state at time t, and m_t^i is its customized spatial feature. $\mathrm{Spt}_{\mathrm{gate}}$ is the map distributor based on a gate mechanism.

6.2.2.4 Current Interaction Encoder

With all agents' dynamics features placed in corresponding nodes in the constructed interaction graph (see the right part of Fig. 6.2), we can model inter-agent interactions by processing the graph via graph neural networks (GNNs):

$$G_t = \mathrm{Int}_{\mathrm{enc}}^{\mathrm{cur}}(D_t, E_t), \tag{6.5}$$

where E_t is the edge set of the interaction graph, and G_t contains the interaction features of all the agents. Considering that all traffic participants and the local map are placed in the ego-centric frame of reference, where no edge features are needed for spatial information, $\mathrm{Int}_{\mathrm{enc}}^{\mathrm{cur}}$ is implemented with a one-layer graph attention

network (GAT) [131] and can be implemented with multilayer GATs easily for higher-order interaction features.

6.2.2.5 Future Interaction Encoder

To utilize information about the future, we need a future information encoder. As our oracle and predictive planners rely on the future trajectories of other agents, we adopt another GNN-based interaction encoder for future trajectories. Similar to the above current interaction encoder, the future interaction encoder is expressed by the following equation:

$$F_t = \mathrm{Int}_{\mathrm{enc}}^{\mathrm{fut}}(T_t, D_t^0, E_t), \tag{6.6}$$

where T_t is the future trajectories of TAs, D_t^0 the dynamics feature of the EV, and F_t the future interaction feature. For the oracle planner, T_t is T_t^{GT}, the ground truth future trajectories of TAs, while for the predictive planner, T_t is replaced with TAs' predicted trajectories T_t^{PT}.

6.2.3 Multi-Agent Trajectory Predictor

The trajectory predictor shares the same encoder with the planners. It predicts TAs' trajectories over the next 5 seconds. After encoding, the prediction decoder combines target vehicles' dynamics, interaction, and selected map features and predicts their future trajectories simultaneously:

$$T_t^{PT} = \mathrm{MATP}(G_t^{TA}, D_t^{TA}, m_t^{TA}), \tag{6.7}$$

where G_t^{TA}, D_t^{TA}, and m_t^{TA} are the interaction, dynamics, and spatial features of all TAs, respectively. T_t^{PT} contains the predicted trajectories. We empirically select LSTMs as decoders since the choice depend heavily on the dataset [110]. Despite our implementation, decoders in this work can be replaced with other sequential models for performance comparison. The predictor is trained independently of the planner and can be replaced with other predictors that may contribute to better performance. Note that our oracle planner relies on the unimodal ground-truth future motions of TAs. We implement the predictor for unimodal modal prediction,

whose outputs can be directly used by the predictive planner. Predictive planning considering multimodal prediction of multiple agents is left for future research.

6.2.4 Oracle Planner

The oracle planner is designed to take as input the ground truth future trajectories of TAs for planning. The future interaction features F_t are obtained via Eq. 6.6. Then the oracle planner can be expressed as:

$$y_t^0 = \text{Plan}_{\text{orc}}(F_t^0, G_t^0, D_t^0, m_t^0), \tag{6.8}$$

where Plan_{orc} is the oracle planner, and y_t^0 is the planned trajectory. The oracle planner is implemented for training.

6.2.5 Predictive Planner

For inference, we replace TAs' ground truth future trajectories T_t^{GT} with predictions T_t^{PT} and apply Eq. 6.6 again to extract future interaction features F_t^0. Then we get a predictive planner:

$$y_t^0 = \text{Plan}_{\text{pre}}(F_t^0, G_t^0, D_t^0, m_t^0), \tag{6.9}$$

where Plan_{pre} is the predictive planner, and y_t^0 is the planned trajectory. The predictive planner is implemented for inference. The predictive planner is a combination of a predictor and our oracle planner. It is independent of the predictor.

6.3 Experiments with Real World Driving Data

6.3.1 The Real World Driving Dataset

As a representative dataset in the area of intelligent transportation systems, the INTERACTION dataset provides naturalistic interactive motions of different traffic participants collected from different countries. The dataset is dedicated to supporting many behavior-related tasks, including but not limited to motion prediction,

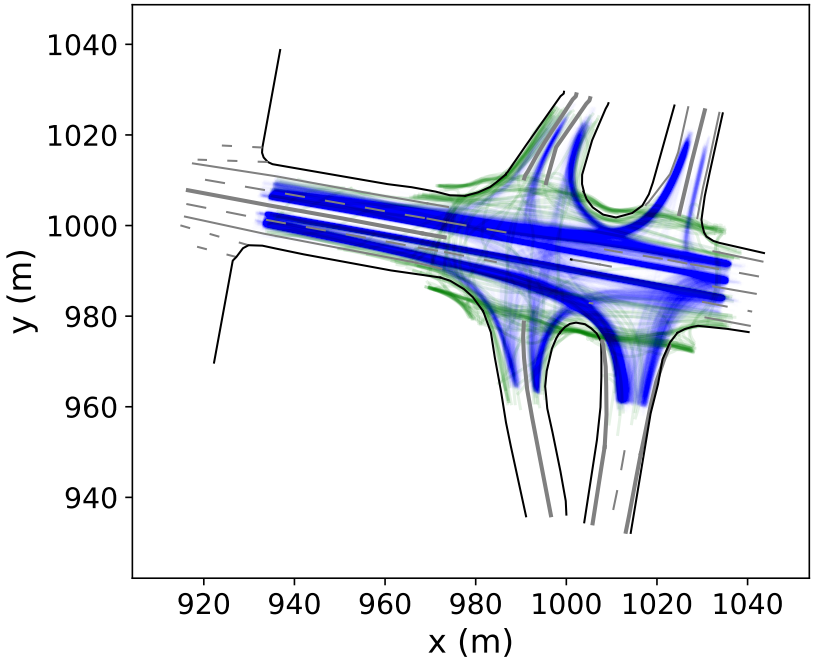

FIGURE 6.3: **Map and traffic.** Map and recorded traffic in the study area. Blue lines are trajectories of vehicles, and green lines are that of pedestrian/bicyclist agents.

behavior analysis and modeling, decision-making, planning, interactive behavior recognition, and driving behavior generation. We use part of this dataset collected from an unsignalized intersection for research on predictive planning. This part is named *DR_USA_Intersection_GL* in the dataset. The road map and recorded traffic of the study area are plotted in Fig. 6.3, where blue and green lines show trajectories of 8,434 vehicles and 184 pedestrian/bicyclist agents, respectively.

6.3.2 Data Processing

To process data for training and validation, we first determine the frames to be considered, and then we select EVs for each frame. For each frame-ego pair, we obtain a piece of data. The steps are summarized below: 1) Initialize an empty directed graph (DG). 2) Get the EV's historical states (h_t^0) and add EV to the DG with h_t^0 as node feature. 3) Find agents within 30 meters of the EV as TAs and add them to the DG with $h_t^1, \ldots, h_t^m$ as node features. 4) Add future trajectories of the EV and TAs as ground truths for planning and prediction, respectively. 5) Find neighboring agents for each TA and add a SA node with historical states if it is not

in the DG. 6) Add self-loops for each node. 7) Add directed edges between each pair of neighboring nodes. 8) Record in-map attributes for the EV and TAs. We also save a shared map of the study area. Two agents are defined as each other's neighbors if the distance between them is less than 30 meters.

6.3.3 Baselines

We implement the following baselines for comparison:

- **Ego-motion (EM).** This is a neural motion planner that extrapolates the EV's historical motions to the next seconds. This baseline is implemented as a sequence-to-sequence model that takes as input only the EV's historical states and outputs its future trajectory. This is different from the Ego-motion baseline in [95], where a four-layer MLP is implemented as the Ego-motion planner. The Ego-motion planner in [95] outperforms Adaptive Cruise Control (ACC) based and manual cost based planners.
- **Planner-c (Pc).** This is a neural motion planner that takes as input the current states of traffic participants and the local map, trying to approximate traditional planners. Using this baseline, we would like to investigate how considering the historical states can affect the planner and how the future trajectory of other agents can affect the planner.
- **Planner-h (Ph).** This is a neural motion planner that combines EM and Pc. It takes as input the EV's historical states, other agents' current states, and the local map.
- **Planner-H (PH).** This is an extension to Ph. It takes the historical states of the EV and other agents and the local map as input.

All these baselines can generate a trajectory for the ego to follow in the next five seconds. The above baselines can be used to investigate how knowing vehicles' historical states can improve the neural motion planner.

6.3.4 Predictors

We implement two trajectory predictors to investigate how different predictors can affect the performance of the predictive planner.

- **Predictor-D.** This is a non-interaction-aware predictor that takes as input only the TV's historical states as input.
- **Predictor-IMD.** This is the interaction-aware trajectory predictor introduced in 6.2.3.

These two predictors are pre-trained with the same dataset. They are implemented to investigate how prediction accuracy can affect downstream planners.

6.3.5 Our Planners

We use TAs' ground truth future trajectories to train an oracle planner (**OrcPlan**) and investigate how knowing other agents' future motions can improve planning performance. However, ground truths are not accessible in inference. This is also the reason we call it an oracle planner. For inference, we replace the ground truth trajectories used by the oracle planner with predicted trajectories and call the oracle planner for inference our predictive planner (**PrePlan**).

6.3.6 Metrics

We adopt three metrics (Average Displacement Error (ADE), Final Displacement Error (FDE), and Miss Rate (MR)) from trajectory prediction tasks to evaluate how the implemented planners can imitate human driving behaviors. Eq. 6.10 and Eq. 6.11 shows the calculation of ADE and FDE:

$$ADE = \frac{1}{T_f} \sum_{\tau=1}^{T_f} \sqrt{(\hat{x}_\tau - x_\tau)^2 + (\hat{y}_\tau - y_\tau)^2}, \tag{6.10}$$

$$FDE = \sqrt{(\hat{x}_{T_f} - x_{T_f})^2 + (\hat{y}_{T_f} - y_{T_f})^2}, \tag{6.11}$$

where $(\hat{x}_\tau, \hat{y}_\tau)$ is the predicted position in the XY coordinates at time τ, (x_τ, y_τ) is the ground truth of position at time τ, and T_f is the prediction horizon. MR is defined as the number of scenarios where the predicted trajectories are two meters away from ground truths according to endpoint error. Collision rate (CR), jerk, acceleration (Acc.), and deceleration (Decel.) are also adopted to evaluate planners.

6.4 Results

This section reports the results of our experiments. We first compare the performances of our baselines to show how historical states can improve planners. Then we present the planning performances of our oracle and predictive planners to illustrate how prediction can help planners. Finally, we provide an overall comparison between all the implemented planners.

6.4.1 Impacts of Historical States

TABLE 6.1: Planning performances of baselines

	$T_h = 1$	$T_h = 3$	$T_h = 5$	$T_h = 8$	$T_h = 10$
			ADE ↓		
Ego-motion					0.945±0.002
Planner-c	1.235±0.01				
Planner-h		0.963±0.012	0.911±0.011	0.892±0.015	**0.874±0.023**
Planner-H		0.954±0.011	0.909±0.016	0.879±0.02	**0.864±0.013**
			FDE ↓		
Ego-motion					3.044±0.012
Planner-c	3.327±0.034				
Planner-h		2.829±0.035	2.721±0.032	2.69±0.034	**2.64±0.048**
Planner-H		2.809±0.023	2.709±0.028	2.658±0.047	**2.612±0.026**
			MR ↓		
Ego-motion					0.502±0.006
Planner-c	0.504±0.008				
Planner-h		0.456±0.007	0.443±0.006	0.443±0.008	**0.437±0.007**
Planner-H		0.449±0.006	0.437±0.01	0.431±0.011	**0.427±0.004**

Tab. 6.1 compares the performances of our baselines (6.3.3) in terms of ADE, FDE, and MR. The Ego-motion planner is trained with $T_h = 10$. Planner-c is trained with $T_h = 1$. Planner-h and Planner-H are trained with T_h range from 3

to 10. All the results are obtained by calculating the mean and standard deviation of five experiments. Comparing Ego-motion and Planner-c, we can see that Ego-motion shows better performance, indicating the importance of ego's motion in planning. We can also observe that the performances of both Planner-h and Planner-H increase with the length of the traceback horizon (T_h), and Planner-H shows consistent but marginal advances.

6.4.2 Performances of Predictive Planners

TABLE 6.2: Prediction performances

	$T_f = 10$	$T_f = 20$	$T_f = 30$	$T_f = 40$	$T_f = 50$
	DE_τ				
Predictor-D	0.3058	0.5423	1.1333	1.9154	2.773
Predictor-IMD	0.2958	0.5066	1.0198	1.6677	2.3611

We train two predictors and show their performances in terms of Displacement Error in meters at time τ (DE_τ) in Tab. 6.2, where T_f is the prediction horizon. We can see that Predictor-IMD always shows better performance compared to Predictor-D, especially for long-term predictions, indicating the necessity of considering interactions for predictors. Combining these two predictors with our oracle planners, we obtain PrePlan (PT-D) and PrePlan (PT-IMD), respectively. Performances of our oracle and predictive planners are presented in Tab. 6.3 along with that of Planner-H ($T_h = 10$). Planner-H ($T_h = 10$) is compared here because it uses the same inputs as our predictive planner (PrePlan (PT-D)). We can see that OrcPlan shows better performance as T_f increases and outperforms other planners with $T_f \geq 20$, indicating that knowing the future motion of surrounding agents can help neural motion planners to imitate human drivers. However, the predictive planners achieve the best performance at $T_f = 30$ (rather than $T_f = 50$), which is better than Planner-H ($T_h = 10$). Comparing OrcPlan with $T_f = 10$ and Planner-H with $T_h = 10$, we observe a counterintuitive result that using ground truth of surrounding agents' motions in the next one second raises counteractive for planning. Even though OrcPlan outperforms all the planners, the achieved displacement error and miss rate remain higher than expected. Another interesting observation is that PrePlan (PT-IMD) does not show significantly better performance than PrePlan (PT-D) despite the better predictor used. This indicates that

improvement in the prediction module does not contribute to proportional gain in the predictive planning module.

TABLE 6.3: Planning performances of oracle and predictive planners

	$T_f = 10$	$T_f = 20$	$T_f = 30$	$T_f = 40$	$T_f = 50$	Planner-H ($T_h = 10$)
			ADE			
OrcPlan	0.871±0.015	0.835±0.007	0.822±0.011	0.783±0.022	**0.777±0.007**	0.864±0.013
PrePlan (PT-IMD)	0.871±0.014	0.85±0.004	**0.847±0.011**	0.864±0.019	0.878±0.007	
PrePlan (PT-D)	0.872±0.014	0.853±0.004	**0.846±0.009**	0.883±0.018	0.907±0.009	
			FDE			
OrcPlan	2.632±0.027	2.542±0.027	2.476±0.03	2.339±0.06	**2.303±0.02**	2.612±0.026
PrePlan (PT-IMD)	2.632±0.027	2.591±0.017	**2.576±0.03**	2.632±0.042	2.671±0.019	
PrePlan (PT-D)	2.639±0.025	2.596±0.014	**2.579±0.026**	2.697±0.04	2.759±0.017	
			MR			
OrcPlan	0.433±0.005	0.411±0.006	0.398±0.008	0.375±0.013	**0.366±0.006**	0.427±0.004
PrePlan (PT-IMD)	0.434±0.005	0.424±0.005	**0.423±0.009**	0.434±0.012	0.434±0.006	
PrePlan (PT-D)	0.433±0.005	0.425±0.005	**0.424±0.008**	0.445±0.013	0.45±0.005	

6.4.3 Comparison with Human Drivers

TABLE 6.4: Planning performance comparison with human demonstrations

Planner	CR ↓	Jerk	Acc.	Decel.
Ego-Motion	0.06	2.511	0.764	0.547
Planner-c	0.057	2.001	**0.36**	0.614
Planner-h ($T_h = 10$)	0.047	**1.83**	**0.517**	**0.464**
Planner-H ($T_h = 10$)	0.046	2.712	0.465	0.789
PrePlan (PT-D, $T_f = 30$)	0.037	2.265	0.586	0.582
PrePlan (PT-IMD, $T_f = 30$)	0.036	2.274	0.591	0.593
OrcPlan	**0.025**	2.291	0.595	0.595
Human	**0.009**	**1.366**	**0.509**	**0.485**

We also compare all the implemented planners with human driving data in terms of collision rate (CR), jerk, acceleration (Acc.), and deceleration (Decel.). As shown in Tab 6.4, the predictive planner (PrePlan (PT-IMD, $T_f = 30$)) achieves the lowest CR among all planners except the oracle planner. This demonstrates the effectiveness of incorporating prediction into planning. We can see there is still a gap between predictive planners and the oracle planner. This indicates that the performance of the predictive planner can be improved with a better trajectory predictor that can approximate the ground truth closely enough. Ideally, if the predictor can predict future motions of TAs without displacement error, it

will achieve the same performance as the oracle planner. If we compare Ego-motion and Planner-c over results in Tab. 6.1 and Tab. 6.4, we can see that even though Ego-Motion achieved much better performance in terms of ADE, FDE, and MR, it fails to outperform Planner-c in terms of CR, the most critical metric for motion planning. We can also see that even human drivers present a CR close to one percent. This observation is consistent to [20], where it is clearly stated that the INTERACTION dataset contains near-collision and collision situations. Comparing planners with human drivers in terms of Jerk, Acc., and Decel., we can see that Planner-h ($T_h = 10$) surprisingly presents the closest records to that of human drivers. It also shows the lowest Jerk and Decel. among all planners and its Decel. is even lower than that of humans. However, the lowest Acc. is shown by Planner-c.

6.4.4 Inference Time

The inference time of a learning-based planner is important in practical applications, and our planner can satisfy the real-time requirements with limited computational resources. We run our planner to do inference on about 15,000 data and observe that it takes 135 seconds to finish inference on these data with batch size set to 1. The average time for a single inference is 0.009 seconds. When we set the batch size to 32, the planner takes 16 seconds to do inference on the same amount of data. The average batch inference time is 0.033 seconds. Both the single and batch inference of the proposed neural motion planners satisfy the real-time requirement. The real-time inference can is achieved with a single GPU (using only 15% memory and 25% GPU utilization of an NVIDIA GeForce RTX 2080 SUPER) plus a single CPU (a single core of the Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz).

Overall, the results in this chapter provide essential insights into the design of neural motion planners. It is shown that both the historical states of the ego and the current states of other agents are important for motion planners, and combining these two kinds of information leads to better performance. The effectiveness of incorporating prediction into planning is verified, while there is still a gap between

predictive planners and oracle planners. Both prediction horizon and accuracy can affect the performance of a predictive planner.

6.5 Discussions

This study aims to design a predictive planner that takes as input the predicted trajectories of several target agents and to investigate how prediction can improve planning performance. The results show that our oracle and predictive planners outperform all the baselines and demonstrate the effectiveness of the designed predictive planner and the necessity of incorporating prediction into planning. One interesting finding is that the improvement in the prediction module does not lead to a corresponding improvement in the planning module. A possible explanation for this might be that the current predictive planner cannot take full advantage of the improved prediction. Another possible explanation is that the relationship between prediction and planning modules is inherently complex and needs more investigation. Another finding is that knowing the ground truth motion of others in a longer horizon leads to better performance. This is what we expected. However, incorporating short-term (one-second) prediction, even ground truth, into planning shows a negative effect on planning. This is unanticipated because information about the future is additional and does not cause loss of other information. It is difficult to explain this, but it may be that the current predictive planner can not handle short-term prediction (or ground truth) properly. This observation should be interpreted with caution because it is counterintuitive. Contrary to the finding in the oracle planner, a longer prediction horizon does not always result in better planning performance. The best performance is achieved when the prediction horizon is three seconds rather than five. One possible reason is that the prediction error goes unacceptable after three seconds, so the prediction misleads the planner that is trained with ground truth. It is also observed that the Ego-motion planner outperforms Planner-c in terms of ADE, FDE, and MR but shows higher CR. This can be explained by the lack of interaction information in Ego-motion.

The proposed predictive planner shows a way to incorporate prediction and planning by training an oracle planner and implementing it with pre-trained predictors. The results of this study raise intriguing questions regarding the nature of human

driving behaviors and the design of predictive planners. The oracle planner's performance shows a top ceiling of the planner that is almost impossible for the current planner to break through by endlessly improving prediction accuracy, so we should focus on not only improving prediction accuracy but also designing planners that can take full advantage of predictions.

In the future, if the planning method is used for multiple vehicles that interact with each other, there may exist conflicts between the planned trajectories of different vehicles at a time t. Then these vehicles will be involved in a game process that ends up with some vehicles yield or occasional collision. In this process, the planning method will update the planned trajectory every time a new observation is received. The game process is common in human driving scenarios, where two interacting vehicles can have the same goal. They would then play a game against that goal. The potential conflicts among either or both of self-driving and human-driving vehicles should be addressed carefully in the future for the safety of autonomous driving.

One of the limitations of the proposed predictive planner is that it does not consider prediction uncertainties. This can be addressed in a two-step manner. The first step is to estimate the uncertainties of predictions and the second step is to design a predictive planner that can cover these uncertainties. Another limitation is that the oracle planner cannot take full advantage of the ground truth.

6.6 Conclusions

An interaction-aware predictive planner is designed in this work to imitate human driving behaviors. The predictive planner is obtained via training an oracle planner aware of target agents' ground truth future trajectories and replacing the ground truth with predicted trajectories for inference. Experimental results on a real-world dataset show that the proposed predictive planner achieves the best performance over many baselines in terms of displacement error, miss rate, and collision rate. The gap between the predictive planner and the oracle planner shows that it is promising to improve prediction accuracy for a better planner since the oracle planner can be deemed as a predictive planner with a zero-error predictor.

Future works are suggested to design predictive planners considering prediction uncertainty since prediction can never be exactly the same as ground truth. We also suggest further investigations on the relationship between prediction and planning based on our observation that an improvement in prediction does not contribute to proportional improvement in planning.

Chapter 7

Conclusion & Future Work

7.1 Conclusion

In this thesis, a series of interaction-aware trajectory prediction methods, including single-agent trajectory prediction, multi-agent trajectory prediction, and multimodal trajectory prediction, are developed for autonomous driving. Besides, the impacts of trajectory prediction on trajectory planning are also investigated.

In Chapter 3, a novel framework with consideration of vehicle-infrastructure heterogeneous interactions is proposed for trajectory prediction of a single target vehicle. In the proposed scheme, a heterogeneous graph is developed to represent the interactions, where the nodes contain features extracted from corresponding encoders. Besides, a novel heterogeneous graph social pooling (HGS) module is designed to extract high-level interaction features. The framework can be easily expanded for highway driving scenarios. Experimental results obtained using real-world driving datasets show that the proposed HGS method outperforms existing interaction-aware methods in terms of prediction accuracy. Besides, ablative studies demonstrate that the consideration of vehicle-infrastructure heterogeneous interactions effectively improves the prediction accuracy compared to those methods only considering inter-vehicle interactions.

Then, the above prediction method for single-agent is generalized and expanded for heterogeneous multi-agent trajectory prediction in Chapter 4. To do this, a novel three-channel framework is designed to jointly consider traffic participants'

dynamics, interaction, and map features. The driving scene is represented in a hybrid way, where the inter-agent interaction in the traffic system is represented with an edge-featured heterogeneous graph, and the shared local map is represented with a Bird's Eye View (BEV) image. Two shared Recurrent Neural Networks (RNNs) are adopted to capture vehicles' and pedestrians' dynamics features from their historical states, respectively. A novel heterogeneous edge-enhanced graph attention network (HEAT) is proposed to model the inter-agent interactions, and a map-sharing technique based on the gate mechanism is also leveraged to share the local map across all target agents. Experimental validations on real-world driving datasets of both urban and highway scenarios show that the proposed method not only achieves state-of-the-art performance but also can provide simultaneous predictions of multi-agent trajectories for a variable number of heterogeneous agents.

Besides the unimodal predictions for single and multiple agents, this thesis also tackles the inherent multimodality problem of driving behaviors for prediction in Chapter 5. A novel map-adaptive multimodal trajectory prediction framework is proposed. Within this framework, through a single graph operation, a variable number of map-compliant trajectories and a non-map-compliant trajectory can be generated. Map-compliant predictions are conditioned on either a single candidate centerline (CCL) or a bunch of all CCLs, making the predictor adaptive to different road structures. The non-map-compliant prediction captures the irrational driving behavior for safety concerns. The driving scene is represented with a heterogeneous hierarchical graph containing both agents and their CCLs. A hierarchical graph operator (HGO) with an edge-masking technology is proposed to encode the driving scene. Validation on the Argoverse motion forecasting benchmark shows that the proposed method achieves state-of-the-art performance with the advantage of map-adaptive capacity.

Beyond pure prediction, in Chapter 6, predictive planning and the impacts of prediction on downstream trajectory planning are also investigated. An interaction-aware predictive planner, which is trained to imitate human driving behaviors, is designed to investigate the problem of how prediction would affect the performance of motion planning. The predictive planner is obtained by training an oracle planner, which is aware of target agents' ground truth future trajectories, and replacing the ground truth with the predicted trajectories for inference during

implementation. Experimental results on a real-world dataset show that the proposed predictive planner achieves better performance over other baselines in terms of displacement error, miss rate, and collision rate. The gap between the predictive planner and the oracle planner shows that it is promising to further enhance the planning performance by improving the prediction accuracy.

To be implemented in real-world self-driving systems, the proposed methods require upstream localization, perception, and tracking results, since the historical states of other traffic participants are needed as the input of the proposed methods. Perception can be realized using either or both of camera and LiDAR. Other sensors, such as radar, can also be used for better perception via sensor fusion. For the single-agent and multi-agent prediction methods in Chapter 3 and Chapter 4, we are using BEV maps, where only the map is needed. We do not need a BEV image to show the real-time traffic. The map can be obtained from main-stream map providers and converted into images for the usage of our method. The map-adaptive multimodal method in Chapter 5, however, requires a high-definition map (HD map) of the local area since we need the candidate centerlines of vehicles of interest. The methods can run on both CPUs or GPUs, and using GPUs is suggested for faster inference.

7.2 Future Work

Although many studies have been done in trajectory prediction and path planning in the past years, there are still many aspects that need to be further investigated in the future.

Scene representation and encoding. Researchers have proposed many methods to encode driving scenes with different representations. However, there is no unified representation of various driving scenes so far. The lack of a universal representation limits the generalizability of prediction methods with large-scale deployment in autonomous vehicles in the real world because a method can hardly be applied to a situation that cannot be described. Among many representation approaches proposed so far, graph-based representations are promising because a graph can accommodate an arbitrary number of heterogeneous objects and represent their interdependencies via directed edges. For example, when modeling a driving scene

in the context of traffic systems, a node can represent a vehicle, a pedestrian, a lanelet, a junction, a traffic signal, etc. A new object can always be added to the existing graph. There are three important steps that need to be done to further improve the graph-based scene representation and encoding in the future. The first aspect is to construct the graph with proper connections, that is, to determine the edge set of the graph. This step needs to identify interdependencies between pairs of nodes and connect nodes with directed edges for information flow in the graph. Once the graph structure is settled, the second step is to assign the node, and edge features properly. This requires researchers to select or design proper encoders for different kinds of nodes and edges. Then the third step is required to design graph operators to handle the heterogeneity in the scene graphs. In this step, the advances in heterogeneous graph neural networks can be leveraged.

Trajectory decoding. For future work, an immediate step is to generalize the map-adaptive multimodal prediction method proposed in Chapter 5 with uncertainty estimations. The uncertainty includes both motion and mode uncertainties. The motion uncertainty captures the distribution of agents' position over a planar map at each time step, and the mode uncertainty captures the possibility of driving modalities. The former can be modeled via bivariate Gaussian distributions, and the latter can be treated as a multi-class classification problem over a variable number of modalities. Then the next step can focus on generalizing uncertainty-aware multimodal predictions to multi-agent settings by modeling the joint distribution of multiple agents' behaviors. Social consistency should be considered in this step such that there is no conflict between any pair of trajectories in a joint modality in normal cases. Besides, trajectory predictors should be designed from the ego vehicle's point of view. One possible way is to design a decoder that can output trajectories upon the ego's request. For example, the predictor can focus on a small set of target agents requested by the ego vehicle rather than all the agents in sight. For a specific target agent, the predictor can focus on predicting its driving options that may affect the ego's planned trajectories. This attentive approach can reduce computation efforts for real-time implementations.

Predictive planning. The ultimate goal of trajectory prediction is to further improve the performance of decision-making and motion control of autonomous vehicles with respect to safety, smartness, and efficiency. So prediction must be integrated into the planning module, and therefore predictive planning is worthwhile

exploring. There are many problems that should be addressed for the development of predictive planners. First, predictive planners should be able to address prediction uncertainty since prediction can never be exactly the same as the ground truth. Second, the relationship between prediction and planning needs to be further studied in order to answer the following questions: 1) How would the improvement in prediction affect the downstream planning performance? 2) Is there a floor of prediction error below which improving prediction accuracy leads to no improvement or even a negative effect on planning? 3) Can we design a predictive planner that is scalable to predictors of different uncertainties as long as these uncertainties are known? Third, learning-based motion planners should be further investigated since they have great potential to be incorporated with data-driven predictions. However, the current limitations in explainability and reliability need to be addressed. In general, plenty of effort is needed in these research areas.

Appendix A

Visualization and Detailed Results for Chapter 3

A.1 Visualization

Visualization of the HGS predictions on the INTERACTION validation set is presented in Fig. A.1, where GT is the ground truth for future trajectory, and HGS is the corresponding predicted trajectory. Fig. A.1 shows that the proposed scheme HGS is able to accurately predict the future trajectory of the target vehicle under various driving situations.

A.2 Detailed Results

Tab. A.1 shows the detailed ADE and FDE values of the implemented models (Method), trained and validated with the dataset described in Sub.Sec. 3.3.1, on different driving scenarios (Scene). The scenarios are numbered consistent with those in Fig. 3.3. It can be seen that the proposed method outperforms its ablations throughout all the scenarios.

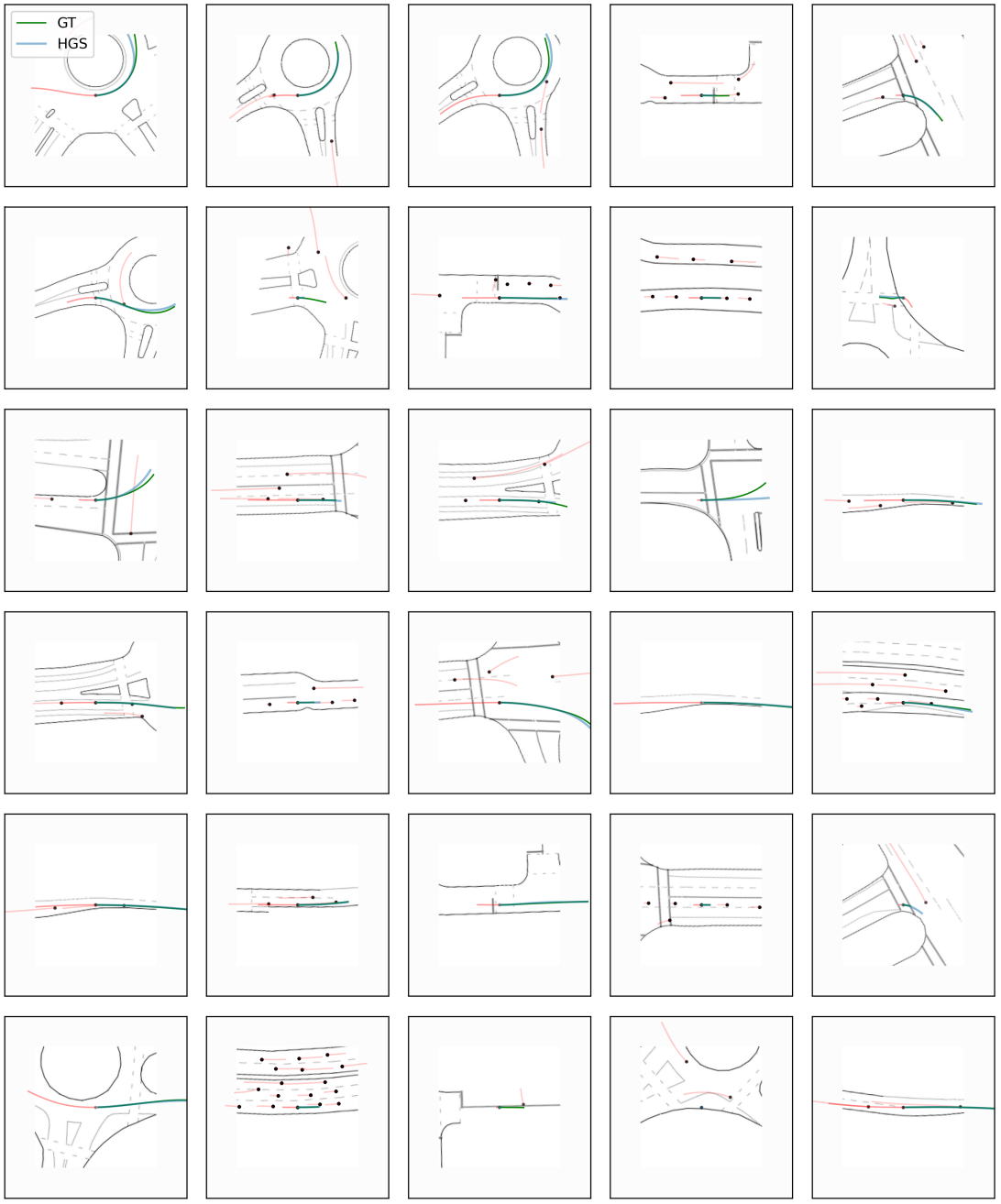

FIGURE A.1: **Visualized results of HGS using the INTERACTION validation dataset.** Traceback horizon: $T_h = 30$. Prediction horizon: $T_f = 50$. Input data: (x, y, v_x, v_y). Graphic encoder: GAT. Dynamics encoder: GRU. GT: Ground truth trajectory. HGS: HGS predicted trajectory. Black dots and red lines are current positions and historical tracks of vehicles, respectively.

TABLE A.1: Detailed quantitative results in ADE / FDE (m) for $T_h = 30$ and $s_t = (x, y, v_x, v_y)$

Scene	**Method**	**Prediction horizon**				
		1.0s	2.0s	3.0s	4.0s	5.0s
1	R	0.0121 / 0.0276	0.0589 / 0.2034	0.1763 / 0.615	0.366 / 1.2225	0.6183 / 1.9784
	GR	0.0115 / 0.0263	0.0487 / 0.1568	0.1288 / 0.4121	0.2447 / 0.7527	0.3926 / 1.1912
	HGS	**0.0101** / **0.023**	**0.0451** / **0.1481**	**0.1222** / **0.3958**	**0.2345** / **0.7279**	**0.3788** / **1.1578**
2	R	0.0195 / 0.0437	0.0863 / 0.2879	0.2507 / 0.8727	0.5301 / 1.8304	0.9304 / 3.1607
	GR	0.02 / 0.0428	0.0761 / 0.2362	0.1977 / 0.64	0.3887 / 1.2578	0.6527 / 2.117
	HGS	**0.0185** / **0.0378**	**0.0678** / **0.2089**	**0.1724** / **0.5467**	**0.3297** / **1.0358**	**0.5422** / **1.7086**
3	R	0.0429 / 0.0938	0.1681 / 0.536	0.4662 / 1.5894	0.9801 / 3.4339	1.7768 / 6.3827
	GR	**0.036** / 0.0758	0.147 / 0.4929	0.4319 / 1.5193	0.9331 / 3.3296	1.7024 / 6.1131
	HGS	0.0428 / **0.0816**	**0.1298** / **0.3783**	**0.3256** / **1.0632**	**0.6667** / **2.297**	**1.2017** / **4.3426**
4	R	0.0254 / 0.0542	0.0994 / 0.3201	0.2789 / 0.9565	0.579 / 1.9574	0.9927 / 3.2482
	GR	0.0253 / 0.0545	0.0941 / 0.2946	0.2519 / 0.8372	0.508 / 1.6797	0.8618 / 2.8047
	HGS	**0.0234** / **0.0502**	**0.0865** / **0.2685**	**0.2269** / **0.7403**	**0.4465** / **1.4326**	**0.7374** / **2.3117**
5	R	0.0275 / 0.0597	0.1069 / 0.3386	0.2922 / 0.9885	0.6025 / 2.0382	1.0393 / 3.4463
	GR	**0.027** / 0.0582	0.1017 / 0.3179	0.2719 / 0.9053	0.5509 / 1.8362	0.9416 / 3.0947
	HGS	**0.027** / **0.0566**	**0.0971** / **0.2972**	**0.2515** / **0.8184**	**0.4959** / **1.6023**	**0.8264** / **2.6341**
6	R	0.0212 / 0.0473	0.09 / 0.2975	0.2626 / 0.9244	0.564 / 1.9771	1.0026 / 3.463
	GR	0.0218 / 0.0471	0.086 / 0.2771	0.2415 / 0.8286	0.506 / 1.7409	0.8881 / 3.0291
	HGS	**0.0203** / **0.0441**	**0.0803** / **0.2579**	**0.223** / **0.7585**	**0.4604** / **1.5564**	**0.7963** / **2.666**
7	R	0.024 / 0.0559	0.1159 / 0.4038	0.366 / 1.3454	0.819 / 2.9557	1.4837 / 5.2434
	GR	0.0259 / 0.0581	0.1114 / 0.3704	0.3263 / 1.147	0.6987 / 2.441	1.2403 / 4.3134
	HGS	**0.0235** / **0.052**	**0.0982** / **0.3234**	**0.2818** / **0.974**	**0.5891** / **2.0049**	**1.0212** / **3.4444**
8	R	0.0251 / 0.057	0.1132 / 0.3837	0.3396 / 1.2063	0.7299 / 2.5401	1.2775 / 4.2798
	GR	0.025 / 0.055	0.1022 / 0.3352	0.2924 / 1.0126	0.6144 / 2.1038	1.0698 / 3.5969
	HGS	**0.0237** / **0.0506**	**0.0927** / **0.2972**	**0.2534** / **0.8447**	**0.5075** / **1.655**	**0.8503** / **2.7286**
9	R	0.025 / 0.0545	0.1088 / 0.3703	0.3322 / 1.1984	0.7311 / 2.613	1.318 / 4.6179
	GR	0.0264 / 0.0557	0.1047 / 0.3438	0.304 / 1.0683	0.6516 / 2.2798	1.1558 / 3.9752
	HGS	**0.0238** / **0.0512**	**0.094** / **0.3052**	**0.2667** / **0.92**	**0.5582** / **1.907**	**0.9704** / **3.2597**
10	R	0.0246 / 0.0526	0.0947 / 0.2984	0.2567 / 0.864	0.5267 / 1.7711	0.9022 / 2.9615
	GR	0.0249 / 0.0531	0.0886 / 0.2683	0.2257 / 0.7256	0.4426 / 1.4318	0.7405 / 2.3781
	HGS	**0.0223** / **0.0473**	**0.0795** / **0.242**	**0.2021** / **0.6449**	**0.3908** / **1.2415**	**0.6454** / **2.0413**

Appendix B

Processed Data and Visualization for Chapter 4

B.1 Processed Data

A processed data are stored as:

- Historical states: The historical states within a traceback horizon of all agents. The historical states of agent i (position, velocity, and orientation) are stored in its own exclusive coordinate system with the origin fixed at its current position and the horizontal axis pointing to its current direction. See Fig. 4.3 for the illustration of the exclusive coordinate system.
- Edge indexes: The graph connectivity represented as a set of directed edges. A directed edge from node j to node i means that agent j is within the neighborhood of agent i and affects the behavior of agent i. In this work, if agent j is within 30 meters to agent i, then it is treated as a neighbor of agent i.
- Edge attributes: The attributes of all edges. The attribute of an edge from agent j to agent i contains agent j's relative states to that of agent i. In this work, the relative states contain $(\Delta x, \Delta y, \Delta v_x, \Delta v_y, \Delta\psi)$.
- Edge types: The types of all edges. The type of an edge from agent j to agent i contains a concatenation of the types of agent j and agent i. In this

work, the edge type of a directed edge from node j of type $[1, 0, 0]$ to node i of type $[0, 0, 1]$ is set to $[1, 0, 0, 0, 0, 1]$.

- Target masks: The mask of the agents to be predicted. If agent i's future trajectory is to be predicted, its mask is set to 1, and else it's set to 0.
- Vehicle masks: The mask of the vehicles in a scene with 1 represents a vehicle, and 0 represents a non-vehicle.
- Pedestrian masks: The mask of the pedestrians in a scene with 1 represents a pedestrian, and 0 represents a non-pedestrian.
- Target vehicle masks: The mask of the vehicles to be predicted with 1 means that the vehicle's future trajectory is to be predicted.
- Scene map: The map of the scene represented by a top-view image. Since we propose a learned map selector to share the map across all agents, the map can be stored outside each piece of data just once. That saves a great amount of disk space. A corresponding map is saved for each of the scenarios in the INTERACTION dataset. For examples of the scenario maps, please refer to Fig. B.1. The image map of a scenario is shared across all the agents in this scenario via the designed map selector.
- Vehicle-to-map attributes: The states of all agents relative to the map's center at the current time t.
- Ground truth future trajectories: The recorded future trajectories of all target agents over the prediction horizon.

B.2 Visualization

Prediction results of the proposed framework on several scenarios in the INTERACTION dataset are shown in Fig. B.1.

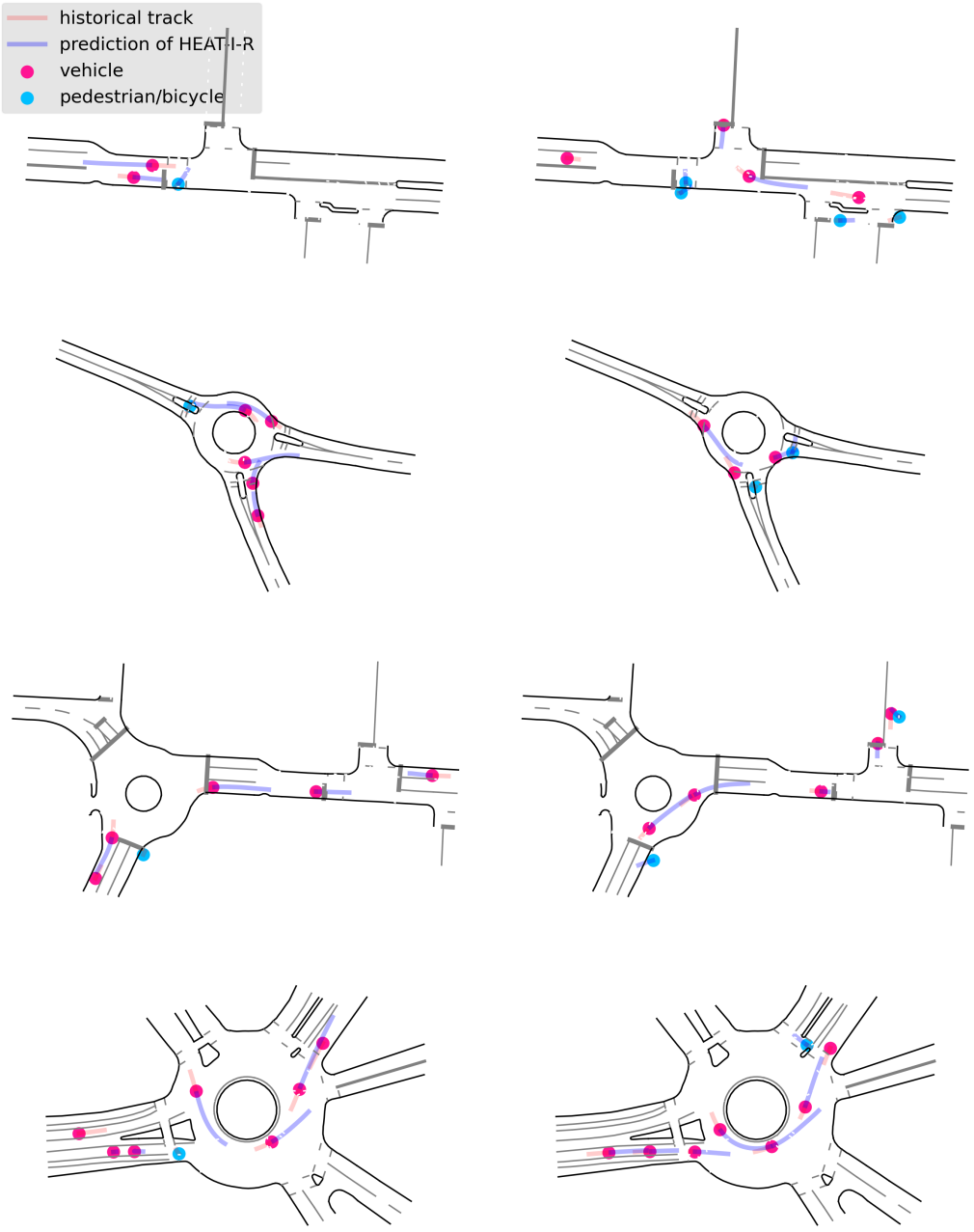

FIGURE B.1: **Visualized prediction results.** This figure visualizes the prediction results of the proposed HEAT-based multi-agent trajectory prediction method on various driving scenarios on the INTERACTION dataset. Deep pink dots (vehicle in the legend) are the vehicles' current positions. Deep sky blue dots (pedestrian/bicyclist in the legend) are the pedestrian/bicyclist agents' current positions. Red lines (historical track in the legend) are their one-second historical trajectories, and blue lines (prediction of HEAT-I-R in the legend) are the trajectories predicted by the proposed framework with HEAT. It shows that the proposed method is able to simultaneously predict trajectories of a variable number of heterogeneous agents (vehicle and pedestrian/bicyclist) in different scenarios.

Appendix C

Source Codes

C.1 Source Codes for Chapter 3

The source codes for single-agent trajectory prediction in highway scenarios are released at `https://github.com/Xiaoyu006/SATP-with-GNN`.

C.2 Source Codes for Chapter 4

The source codes for multi-agent trajectory prediction are released at `https://github.com/Xiaoyu006/MATP-with-HEAT`.

Appendix D

Datasets

The data-driven methods proposed in the thesis are mainly developed on two datasets, the INTERACTION dataset and the Argoverse motion forecasting dataset. The sizes of datasets used in each chapter are listed in Tab. D.1.

D.1 INTERACTION Dataset

The INTERACTION dataset is released by the Mechanical Systems Control (MSC) Lab (University of California, Berkeley) in collaboration with researchers from Centre of Robotics (MINES ParisTech) and FZI Research Center for Information Technology and Karlsruhe Institute of Technology.

The INTERACTION dataset contains trajectories of around 40,000 traffic participants recorded in different countries. The length of the recorded video is around 1,000 minutes.

This dataset can be found at `https://interaction-dataset.com/`.

D.2 Argoverse motion forecasting dataset

The Argoverse motion forecasting dataset is released by Argo AI, LLC.

The Argoverse motion forecasting dataset contains 333,441 (about 330K) 5-second sequences of highly interactive driving scenarios mined from 1,000 hours of videos, which are collected from two districts in Miami and Pittsburgh, USA.

This dataset can be found at `https://www.argoverse.org/av1.html#forecasting-link`.

TABLE D.1: Datasets used in this thesis

Chapter	Dataset	Data Size
Ch3, HGS-SATP	INTERACTION Argoverse	410K 330K
Ch4, HEAT-MATP	INTERACTION	530K
Ch5, HGO-MMTP	Argoverse	330K
Ch6, PrePlan	DR_USA_Intersection_GL of INTERACTION dataset	80K

Appendix E

Summary of trajectory prediction methods

The methods introduced in the literature review (Chapter 2) are summarized in Tab. E.1 for a clear comparison.

TABLE E.1: Summary of trajectory prediction methods

Classes	Methods			
PTP	State Retention	Reachable Set	Monte Carlo Simulation	State Estimation
	Pros: straightforward, simple, efficient	Pros: uncertainty-aware, safe	Pros: model analytically intractable distribution	Pros: uncertainty-aware
	Cons: ignore uncertainty, ignore interaction	Cons: conservative, ignore interaction	Cons: ignore uncertainty, ignore interaction, susceptible to sample size	Cons: ignore interaction
	Refs: [26, 27]	Refs: [28–33]	Refs: [34–36]	Refs: [38–45]
LTP inputs	Raw	Sequence	Grid	Graph
	Pros: suitable in rural areas	Pros: simple, efficient in computation	Pros: interaction modeling, map inclusive	Pros: general representation
	Cons: expensive in computation	Cons: ignore interaction	Cons: ignore connections of road elements, need resolution tuning	Cons: susceptible to graph structure, need new graph neural networkds
	Refs: [62, 63]	Refs: [65, 67–69]	Refs: [58–60, 70, 72, 73, 75, 76]	Refs: [1, 21, 77, 80, 81, 84–88]
Unimodal LTP	Direct regression		Intention-aware	
	Pros: straightforward		Pros: mitigate mode-collapse	
	Cons: mode collapse		Cons: need intention labeling, susceptible to wrong intention classification	
	Refs: [1, 63, 70, 71, 75, 77, 80, 99, 100]		Refs: [68, 94, 101]	
Multimodal LTP	Intention-aware	Sampling & Scoring	Multimodal regression	Map-adaptive
	Pros: explainable modes	Pros: fine-grained driving behavior, good generalizability	Pros: straightforward	Pros: good generalizability, explainable
	Cons: restricted by intention set, need intention labeling, coarse-grained driving behavior	Cons: time-consuming redundant sampling	Cons: unexplainable modes, less generalizability	Cons: rely on HD maps
	Refs: [58, 59, 102]	Refs: [67, 69, 85, 97, 103]	Refs: [21, 76, 106]	Refs: [61]

List of Author's Awards, Patents, and Publications[1]

Awards

- 2021 Winner (1st place) of the Interaction Prediction track of the Waymo Open Dataset Challenges at CVPR 2021, USA. Link: Waymo Open Dataset Challenges 2021.
- 2021 Winner (2nd place) of the Motion Prediction track of the Waymo Open Dataset Challenges at CVPR 2021, USA. Link: Waymo Open Dataset Challenges 2021.
- 2020 Winner (1st place) of the regular track in the INTERACTION-Dataset-Based PREdicTion (INTERPRET) Challenge at NeurIPS 2020, USA. Link: INTERPRET Challenge, NeurIPS 2020.

Patents

- Trajectory Predicting Methods and Systems. International Publication Number: WO 2022/231519 A1, 03 November 2022.

Journal Articles

- **Mo, X.**, Huang, Z., Xing, Y., & Lv, C. (2022). Multi-Agent Trajectory Prediction With Heterogeneous Edge-Enhanced Graph Attention Network. IEEE Transactions on Intelligent Transportation Systems, Accepted, In Press.

[1]The superscript * indicates corresponding authors

- **Mo, X.**, Xing, Y., & Lv, C. (2022). Heterogeneous Graph Social Pooling for Interaction-Aware Vehicle Trajectory Prediction. IEEE Transactions on Intelligent Vehicles, T-IV-22-10-1214, Under Review.
- **Mo, X.**, Xing, Y., & Lv, C. (2022). Interaction-Aware Vehicle Trajectory Prediction With Graph Neural Networks for Smart Mobility. IEEE/CAA Journal of Automatica Sinica, JAS-2022-0853, Under Review.
- Xing, Y., Lv, C., **Mo, X.**, Hu, Z., Huang, C., & Hang, P. (2021). Toward Safe and Smart Mobility: Energy-Aware Deep Learning for Driving Behavior Analysis and Prediction of Connected Vehicles. IEEE Transactions on Intelligent Transportation Systems, 22(7), 4267-4280.
- **Mo, X.**, Chen, Z., & Zhang, H. T. (2019). Effects of adding a reverse edge across a stem in a directed acyclic graph. Automatica, 103, 254-260.
- Zhang, H. T., Chen, Z., & **Mo, X.*** (2017). Effect of adding edges to consensus networks with directed acyclic graphs. IEEE Transactions on Automatic Control, 62(9), 4891-4897.

Conference Proceedings

- **Mo, X.**, Huang, Z., & Lv, C. (2022, October). Stochastic Multimodal Interaction Prediction for Urban Driving. In 2022 IEEE International Intelligent Transportation Systems Conference (ITSC). IEEE.
- **Mo, X.**, Xing, Y., & Lv, C. (2021, September). Graph and Recurrent Neural Network-based Vehicle Trajectory Prediction For Highway Driving. In 2021 IEEE International Intelligent Transportation Systems Conference (ITSC) (pp. 1934-1939). IEEE.
- **Mo, X.**, Xing, Y., & Lv, C. (2020, October). Interaction-aware trajectory prediction of connected vehicles using cnn-lstm networks. In IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society (pp. 5057-5062). IEEE.
- Liu, Q., Mo, Y., **Mo, X.**, Lv, C., Mihankhah, E., & Wang, D. (2019, June). Secure pose estimation for autonomous vehicles under cyber attacks. In 2019 IEEE Intelligent Vehicles Symposium (IV) (pp. 1583-1588). IEEE.

Bibliography

[1] Xiaoyu Mo, Yang Xing, and Chen Lv. Graph and recurrent neural network-based vehicle trajectory prediction for highway driving. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1934–1939. IEEE, 2021. xxv, 18, 20, 26, 44, 46, 47, 146

[2] Junping Zhang, Fei-Yue Wang, Kunfeng Wang, Wei-Hua Lin, Xin Xu, and Cheng Chen. Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1624–1639, 2011. 1

[3] Li Zhu, Fei Richard Yu, Yige Wang, Bin Ning, and Tao Tang. Big data analytics in intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 20(1):383–398, 2018.

[4] Bin Shuai, Quan Zhou, Ji Li, Yinglong He, Ziyang Li, Huw Williams, Hongming Xu, and Shijin Shuai. Heuristic action execution for energy efficient charge-sustaining control of connected hybrid vehicles with model-free double q-learning. *Applied Energy*, 267:114900, 2020.

[5] Faezeh Farivar, Mohammad Sayad Haghighi, Alireza Jolfaei, and Sheng Wen. On the security of networked control systems in smart vehicle and its adaptive cruise control. *IEEE Transactions on Intelligent Transportation Systems*, 22 (6):3824–3831, 2021.

[6] Soheila Ghane, Alireza Jolfaei, Lars Kulik, Kotagiri Ramamohanarao, and Deepak Puthal. Preserving privacy in the internet of connected vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[7] Muhammad Usman, Mian Ahmad Jan, and Alireza Jolfaei. Speed: A deep learning assisted privacy-preserved framework for intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[8] Quan Zhou, Dezong Zhao, Bin Shuai, Yanfei Li, Huw Williams, and Hongming Xu. Knowledge implementation and transfer with an adaptive learning network for real-time power management of the plug-in hybrid vehicle. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 1

[9] Jianyu Chen, Shengbo Eben Li, and Masayoshi Tomizuka. Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 1

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

[11] Sajjad Mozaffari, Omar Y Al-Jarrah, Mehrdad Dianati, Paul Jennings, and Alexandros Mouzakitis. Deep learning-based vehicle behavior prediction for autonomous driving applications: A review. *IEEE Transactions on Intelligent Transportation Systems*, 2020. 1, 9, 14, 20, 24, 79

[12] Lian Hou, Long Xin, Shengbo Eben Li, Bo Cheng, and Wenjun Wang. Interactive trajectory prediction of surrounding road users for autonomous driving using structural-lstm network. *IEEE Transactions on Intelligent Transportation Systems*, 21(11):4615–4625, 2019. 1

[13] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9): 3848–3858, 2019. 4

[14] Ryan Kortvelesy and Amanda Prorok. Modgnn: Expert policy approximation in multi-agent systems with a modular graph neural network architecture. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9161–9167. IEEE, 2021. 4

[15] Yulei Wu, Hong-Ning Dai, and Haina Tang. Graph neural networks for anomaly detection in industrial internet of things. *IEEE Internet of Things Journal*, 2021. 4

[16] Zhiwei Guo and Heng Wang. A deep graph neural network-based mechanism for social recommendations. *IEEE Transactions on Industrial Informatics*, 17(4):2776–2783, 2020. 4

[17] Pritam Saha, Debadyuti Mukherjee, Pawan Kumar Singh, Ali Ahmadian, Massimiliano Ferrara, and Ram Sarkar. Retracted article: Graphcovidnet: A graph neural network based model for detecting covid-19 from ct scans and x-rays of chest. *Scientific reports*, 11(1):1–16, 2021. 4

[18] Yuyang Wang, Jianren Wang, Zhonglin Cao, and Amir Barati Farimani. Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*, 4(3):279–287, 2022. 4

[19] Argoai challenge. `https://slideslive.com/38923162/argoai-challenge`, 2019. 6, 98, 103

[20] Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kümmerle, Hendrik Königshof, Christoph Stiller, Arnaud de La Fortelle, and Masayoshi Tomizuka. INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps. *arXiv:1910.03088 [cs, eess]*, September 2019. 6, 13, 34, 35, 68, 122

[21] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *European Conference on Computer Vision*, pages 541–556. Springer, 2020. 7, 18, 23, 43, 80, 83, 98, 103, 146

[22] Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15303–15312, 2021. 7, 84, 98, 104

[23] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal*, 1(1):1–14, 2014. 9, 10, 25

[24] Phillip Karle, Maximilian Geisslinger, Johannes Betz, and Markus Lienkamp. Scenario understanding and motion prediction for autonomous vehicles-review and comparison. *IEEE Transactions on Intelligent Transportation Systems*, 2022. 9, 10, 24

[25] Robin Schubert, Eric Richter, and Gerd Wanielik. Comparison and evaluation of advanced motion models for vehicle tracking. In *2008 11th international conference on information fusion*, pages 1–6. IEEE, 2008. 10

[26] Jrg Hillenbrand, Andreas M Spieker, and Kristian Kroschel. A multilevel collision mitigation approach—its situation assessment, decision making, and performance tradeoffs. *IEEE Transactions on intelligent transportation systems*, 7(4):528–540, 2006. 10, 146

[27] Panagiotis Lytrivis, George Thomaidis, and Angelos Amditis. Cooperative path prediction in vehicular environments. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pages 803–808. IEEE, 2008. 10, 146

[28] Matthias Althoff. *Reachability analysis and its application to the safety assessment of autonomous cars*. PhD thesis, Technische Universität München, 2010. 11, 146

[29] Markus Koschi and Matthias Althoff. Spot: A tool for set-based prediction of traffic participants. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1686–1693. IEEE, 2017. 11

[30] Michael Hartmann and Daniel Watzenig. Optimal motion planning with reachable sets of vulnerable road users. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 891–898. IEEE, 2019. 11

[31] Ming-Yuan Yu, Ram Vasudevan, and Matthew Johnson-Roberson. Risk assessment and planning with bidirectional reachability for autonomous driving. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5363–5369. IEEE, 2020. 11

[32] Matthias Althoff and John M Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4): 903–918, 2014. 11

[33] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006. 11, 146

[34] Matthias Althoff and Alexander Mergel. Comparison of markov chain abstraction and monte carlo simulation for the safety assessment of autonomous cars. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1237–1247, 2011. 11, 146

[35] Adrian Broadhurst, Simon Baker, and Takeo Kanade. Monte carlo road safety reasoning. In *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pages 319–324. IEEE, 2005. 12

[36] Andreas Eidehall and Lars Petersson. Statistical threat assessment for general road scenes using monte carlo sampling. *IEEE Transactions on intelligent transportation systems*, 9(1):137–147, 2008. 12, 146

[37] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960. 12

[38] Samer Ammoun and Fawzi Nashashibi. Real time trajectory prediction for collision risk estimation between vehicles. In *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing*, pages 417–422. IEEE, 2009. 12, 25, 146

[39] Guotao Xie, Hongbo Gao, Lijun Qian, Bin Huang, Keqiang Li, and Jianqiang Wang. Vehicle trajectory prediction by integrating physics-and maneuver-based approaches using interactive multiple models. *IEEE Transactions on Industrial Electronics*, 65(7):5999–6008, 2017. 12

[40] Beomjun Kim, Kwanwoo Park, and Kyongsu Yi. Probabilistic threat assessment with environment description and rule-based multi-traffic prediction for integrated risk management system. *IEEE Intelligent Transportation Systems Magazine*, 9(3):8–22, 2017. 12

[41] Stefan Hoermann, Daniel Stumper, and Klaus Dietmayer. Probabilistic long-term prediction for autonomous vehicles. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 237–243. IEEE, 2017. 12

[42] Aris Polychronopoulos, Manolis Tsogas, Angelos J Amditis, and Luisa Andreone. Sensor fusion for predicting vehicles' path for collision avoidance systems. *IEEE Transactions on Intelligent Transportation Systems*, 8(3): 549–562, 2007. 12

[43] Helgo Dyckmanns, Richard Matthaei, Markus Maurer, Bernd Lichte, Jan Effertz, and Dirk Stüker. Object tracking in urban intersections based on active use of a priori knowledge: Active interacting multi model filter. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 625–630. IEEE, 2011. 12

[44] Cesar Barrios and Yuichi Motai. Improving estimation of vehicle's trajectory using the latest global positioning system with kalman filtering. *IEEE Transactions on Instrumentation and Measurement*, 60(12):3747–3755, 2011. 12

[45] Nachiket Deo, Akshay Rangesh, and Mohan M Trivedi. How would surround vehicles move? a unified framework for maneuver classification and motion prediction. *IEEE Transactions on Intelligent Vehicles*, 3(2):129–140, 2018. 12, 146

[46] Haoran Song, Di Luan, Wenchao Ding, Michael Y Wang, and Qifeng Chen. Learning to predict vehicle trajectories with model-based planning. In *Conference on Robot Learning*, pages 1035–1045. PMLR, 2022. 13, 43

[47] John Halkias James Colyar. Us highway 101 dataset. `https://www.fhwa.dot.gov/publications/research/operations/07030/index.cfm`, 2007. 13, 44, 68, 73

[48] John Halkias James Colyar. Interstate 80 freeway dataset. `https://www.fhwa.dot.gov/publications/research/operations/06137/`, 2006.

[49] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125. IEEE, 2018.

[50] Xinyu Huang, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2702–2719, 2019.

[51] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 43, 96, 98

[52] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[53] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9710–9719, 2021. 13, 71

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 13, 55

[55] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 13, 55, 66

[56] Iago Gomes and Denis Wolf. A review on intention-aware and interaction-aware trajectory prediction for autonomous vehicles. 2022. 13

[57] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 13, 60

[58] 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*, pages 1468–1476, 2018. 13, 15, 16, 21, 26, 27, 38, 45, 46, 47, 49, 51, 54, 60, 74, 75, 104, 146

[59] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 17, 21, 49, 70, 80, 82, 84, 146

[60] Xiaoyu Mo, Yang Xing, and Chen Lv. Interaction-aware trajectory prediction of connected vehicles using cnn-lstm networks. In *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, pages 5057–5062. IEEE, 2020. 13, 15, 16, 26, 38, 45, 46, 51, 54, 70, 73, 74, 75, 146

[61] Lingyao Zhang, Po-Hsun Su, Jerrick Hoang, Galen Clark Haynes, and Micol Marchetti-Bowick. Map-adaptive goal-based trajectory prediction. *arXiv preprint arXiv:2009.04450*, 2020. 13, 23, 26, 80, 84, 102, 104, 146

[62] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14403–14412, 2021. 14, 107, 108, 146

[63] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018. 14, 19, 21, 146

[64] Alireza Talebpour and Hani S Mahmassani. Influence of connected and autonomous vehicles on traffic flow stability and throughput. *Transportation Research Part C: Emerging Technologies*, 71:143–163, 2016. 15, 52

[65] Alex Zyner, Stewart Worrall, and Eduardo Nebot. A recurrent neural network solution for predicting driver intention at unsignalized intersections. *IEEE Robotics and Automation Letters*, 3(3):1759–1764, 2018. 15, 146

[66] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 15, 30, 39, 67, 103

[67] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678. IEEE, 2018. 15, 22, 146

[68] Long Xin, Pin Wang, Ching-Yao Chan, Jianyu Chen, Shengbo Eben Li, and Bo Cheng. Intention-aware long horizon trajectory prediction of surrounding vehicles using dual lstm networks. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1441–1446. IEEE, 2018. 15, 21, 146

[69] Alex Zyner, Stewart Worrall, and Eduardo Nebot. Naturalistic driver intention and path prediction using recurrent neural networks. *IEEE transactions on intelligent transportation systems*, 21(4):1584–1594, 2019. 15, 22, 146

[70] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12126–12134, 2019. 15, 16, 20, 26, 27, 51, 54, 67, 74, 75, 146

[71] Xiaoyu Mo, Zhiyu Huang, Yang Xing, and Chen Lv. Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network. *IEEE Transactions on Intelligent Transportation Systems*, 2022. 15, 19, 20, 26, 51, 82, 83, 112, 113, 146

[72] Stefan Hoermann, Martin Bach, and Klaus Dietmayer. Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2056–2063. IEEE, 2018. 16, 146

[73] Marcel Schreiber, Stefan Hoermann, and Klaus Dietmayer. Long-term occupancy grid prediction using recurrent neural networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9299–9305. IEEE, 2019. 16, 146

[74] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015. 16

[75] Florent Altché and Arnaud de La Fortelle. An lstm network for highway trajectory prediction. In *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*, pages 353–359. IEEE, 2017. 16, 21, 146

[76] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019. 16, 23, 49, 80, 82, 83, 96, 146

[77] Frederik Diehl, Thomas Brunner, Michael Truong Le, and Alois Knoll. Graph neural networks for modelling traffic participant interaction. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 695–701. IEEE, 2019. 18, 20, 26, 54, 56, 64, 146

[78] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 18, 39

[79] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 18, 33, 39

[80] Xin Li, Xiaowen Ying, and Mooi Choo Chuah. Grip: Graph-based interaction-aware trajectory prediction. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3960–3966. IEEE, 2019. 18, 20, 26, 46, 54, 57, 75, 86, 146

[81] Hyeongseok Jeon, Dongsuk Kum, and Junwon Choi. Scale-net: Scalable vehicle trajectory prediction network under random number of interacting vehicles via edge-enhanced graph convolutional neural network. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2020. 18, 26, 45, 55, 57, 64, 74, 75, 146

[82] Liyu Gong and Qiang Cheng. Exploiting edge features for graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9211–9219, 2019. 18, 56, 57

[83] Xiaoyu Mo, Yang Xing, and Chen Lv. Recog: A deep learning framework with heterogeneous graph for interaction-aware trajectory prediction. *arXiv preprint arXiv:2012.05032*, 2020. 18, 26, 51, 54, 57, 61, 67, 69, 70, 71, 82, 83, 97, 112

[84] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics

from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020. 18, 19, 26, 39, 83, 103, 146

[85] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. In *Conference on Robot Learning*, pages 895–904. PMLR, 2021. 18, 22, 43, 146

[86] Wenyuan Zeng, Ming Liang, Renjie Liao, and Raquel Urtasun. Lanercnn: Distributed representations for graph-centric motion forecasting. *arXiv preprint arXiv:2101.06653*, 2021. 18, 80, 83, 98, 103

[87] Jiachen Li, Fan Yang, Masayoshi Tomizuka, and Chiho Choi. Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning. *Proceedings of the Neural Information Processing Systems (NeurIPS)*, 2020. 19, 26, 55, 64

[88] Jiachen Li, Hengbo Ma, Zhihao Zhang, and Masayoshi Tomizuka. Social-wagdat: Interaction-aware trajectory prediction via wasserstein graph double-attention network. *arXiv preprint arXiv:2002.06241*, 2020. 19, 26, 51, 55, 57, 146

[89] Jean Mercat, Thomas Gilles, Nicole El Zoghby, Guillaume Sandou, Dominique Beauvois, and Guillermo Pita Gil. Multi-head attention for multi-modal joint vehicle motion forecasting. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9638–9644. IEEE, 2020. 19, 43, 46, 98, 103

[90] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene transformer: A unified multi-task model for behavior prediction and planning. *arXiv e-prints*, pages arXiv–2106, 2021. 43

[91] Zhiyu Huang, Xiaoyu Mo, and Chen Lv. Multi-modal motion prediction with transformer-based neural network for autonomous driving. *arXiv preprint arXiv:2109.06446*, 2021. 19

[92] Ekaterina Tolstaya, Reza Mahjourian, Carlton Downey, Balakrishnan Vadarajan, Benjamin Sapp, and Dragomir Anguelov. Identifying driver interactions via conditional behavior prediction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3473–3479. IEEE, 2021. 19

[93] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1150–1160, 2021. 19, 91

[94] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956. PMLR, 2018. 19, 21, 146

[95] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019. 20, 107, 108, 117

[96] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision*, pages 683–700. Springer, 2020. 20

[97] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Home: Heatmap output for future motion estimation. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 500–507. IEEE, 2021. 19, 20, 22, 146

[98] Wenyuan Zeng, Shenlong Wang, Renjie Liao, Yun Chen, Bin Yang, and Raquel Urtasun. Dsdnet: Deep structured self-driving network. In *European conference on computer vision*, pages 156–172. Springer, 2020. 20, 107, 108

[99] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6120–6127, 2019. 21, 146

[100] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, Nitin Singh, and Jeff Schneider. Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2095–2104, 2020. 21, 146

[101] Wenchao Ding and Shaojie Shen. Online vehicle trajectory prediction using policy anticipation network and optimization-based context reasoning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9610–9616. IEEE, 2019. 21, 146

[102] Nachiket Deo and Mohan M Trivedi. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1179–1184. IEEE, 2018. 21, 146

[103] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017. 22, 70, 84, 146

[104] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015. 22

[105] Christopher M Bishop. Mixture density networks. 1994. 22

[106] Siddhesh Khandelwal, William Qi, Jagjeet Singh, Andrew Hartnett, and Deva Ramanan. What-if motion prediction for autonomous driving. *arXiv preprint arXiv:2008.10587*, 2020. 23, 43, 146

[107] Yanjun Huang, Jiatong Du, Ziru Yang, Zewei Zhou, Lin Zhang, and Hong Chen. A survey on trajectory-prediction methods for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2022. 24

[108] Matthias Althoff, Olaf Stursberg, and Martin Buss. Model-based probabilistic collision detection in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 10(2):299–310, 2009. 25

[109] Kaouther Messaoud, Nachiket Deo, Mohan M Trivedi, and Fawzi Nashashibi. Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 165–170. IEEE, 2021. 27

[110] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 30, 39, 67, 90, 112, 113, 114

[111] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 31

[112] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 31

[113] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018. 33

[114] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020. 33, 56

[115] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle,

A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`. 36, 101

[116] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 36, 101

[117] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 36

[118] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal motion prediction with stacked transformers. *Computer Vision and Pattern Recognition*, 2021. 43

[119] Qian Shi and Hui Zhang. An improved learning-based lstm approach for lane change intention prediction subject to imbalanced data. *Transportation research part C: emerging technologies*, 133:103414, 2021. 48

[120] Yang Xing, Chen Lv, Dongpu Cao, and Efstathios Velenis. Multi-scale driver behavior modeling based on deep spatial-temporal representation for intelligent vehicles. *Transportation research part C: emerging technologies*, 130: 103288, 2021. 48

[121] Guofa Li, Yifan Yang, Shen Li, Xingda Qu, Nengchao Lyu, and Shengbo Eben Li. Decision making of autonomous vehicles in lane change scenarios: Deep reinforcement learning approaches with risk awareness. *Transportation research part C: emerging technologies*, 134:103452, 2022. 48

[122] Chengyuan Ma, Chunhui Yu, and Xiaoguang Yang. Trajectory planning for connected and automated vehicles at isolated signalized intersections under mixed traffic environment. *Transportation research part C: emerging technologies*, 130:103309, 2021. 48

[123] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 54, 57, 70

[124] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. *arXiv preprint arXiv:2008.08294*, 2020. 54, 57, 69, 70, 71, 80, 84, 88, 97, 98, 103

[125] Xiaoyu Mo, Zhiyong Chen, and Hai-Tao Zhang. Effects of adding a reverse edge across a stem in a directed acyclic graph. *Automatica*, 103:254–260, 2019. 55, 83

[126] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 55

[127] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*, 2016.

[128] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017. 55, 56

[129] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017. 55

[130] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[131] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`. 55, 56, 66, 67, 73, 91, 92, 114

[132] Xiaodong Jiang, Ronghang Zhu, Sheng Li, and Pengsheng Ji. Co-embedding of nodes and edges with graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 56

[133] Yulei Yang and Dongsheng Li. Nenn: Incorporate node and edge features in graph neural networks. In *Asian Conference on Machine Learning*, pages 593–608. PMLR, 2020. 56

[134] Jun Chen and Haopeng Chen. Edge-featured graph attention network. *arXiv preprint arXiv:2101.07671*, 2021. 56

[135] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, pages 2022–2032, 2019. 56, 65

[136] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, pages 2704–2710, 2020. 56

[137] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. 56

[138] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 60

[139] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002. 67, 68

[140] Interaction-dataset-based prediction challenge. `http://challenge.interaction-dataset.com/prediction-challenge/intro`, 2020. 71

[141] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 204–211. IEEE, 2017. 75

[142] Hai-Tao Zhang, Zhiyong Chen, and Xiaoyu Mo. Effect of adding edges to consensus networks with directed acyclic graphs. *IEEE Transactions on Automatic Control*, 62(9):4891–4897, 2017. 83

[143] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14074–14083, 2020. 84

[144] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Gohome: Graph-oriented heatmap output forfuture motion estimation. *arXiv preprint arXiv:2109.01827*, 2021. 84, 98, 104

[145] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018. 84

[146] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1349–1358, 2019. 84

[147] Fang Da and Yu Zhang. Path-aware graph attention for hd maps in motion prediction. *arXiv preprint arXiv:2202.13772*, 2022. 84

[148] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 92

[149] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021. 92

[150] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 793–803, 2019. 92

[151] Xin Huang, Stephen G McGill, Jonathan A DeCastro, Brian C Williams, Luke Fletcher, John J Leonard, and Guy Rosman. Diversity-aware vehicle motion prediction via latent semantic sampling. *arXiv preprint arXiv:1911.12736*, 2019. 98

[152] Evaluation criteria of the argoverse motion forecasting competition. `https://eval.ai/web/challenges/challenge-page/454/evaluation`, 2020. 97

[153] Alexander Cui, Sergio Casas, Abbas Sadat, Renjie Liao, and Raquel Urtasun. Lookout: Diverse multi-future prediction and planning for self-driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16107–16116, 2021. 107