Transferable and Adaptable Driving Behavior Prediction

Anonymous authors
Paper under double-blind review

Abstract

While autonomous vehicles still struggle to solve challenging situations during on-road driving, humans have long mastered the essence of driving with efficient, transferable, and adaptable driving capability. The obvious gap between humans and autonomous vehicles keeps us wondering about the essence of how human learns to drive. Inspired by humans' cognition model and semantic understanding during driving in a hierarchical learning procedure, we propose HATN, a hierarchical framework to generate high-quality, transferable, and adaptable predictions for driving behaviors in multi-agent dense-traffic environments. Our hierarchical method consists of a high-level intention identification policy and a low-level trajectory generation policy. We introduce a novel semantic definition for the two policies and generic state representation for each policy, so that the hierarchical framework is transferable across different driving scenarios. Besides, our model is able to capture variations of driving behaviors among individuals and scenarios by an online adaptation module. We demonstrate our algorithms in the task of trajectory prediction for real traffic data at intersections and roundabouts from the INTERACTION dataset. Through extensive numerical studies, it is evident that our method significantly outperformed other methods in terms of prediction accuracy, transferability, and adaptability. Pushing the performance by a considerable margin, we also provide a cognitive view of understanding the driving behavior behind such improvement. We highlight that in the future, more research attention and effort are deserved for the transferability and adaptability of autonomous driving planning and prediction algorithms. It is not only due to the promising performance elevation, but more fundamentally, they are crucial for the scalable and general deployment of autonomous vehicles.

1 Introduction

When autonomous vehicles are deployed on roads, they will encounter diverse scenarios varying in traffic density, road geometries, traffic rules, etc. Each scenario comes with different levels of difficulty in understanding and predicting future behaviors of other road participants. Even in a straight street with few road entities, the sensor system of AVs still confronts a daunting amount of information that may or may not be relevant to the behavior prediction task. Let alone in more complex scenarios like crowded, human-vehicle-mixed, complicated-road-geometry intersection or roundabouts, currently deployed AVs tend to timidly take conservative behaviors due to insufficient prediction capability. On the contrary, humans can drive through and across these environments easily, even while talking to friends or shaking to the music.

Moreover, most state-of-art behavior prediction algorithms for AVs, once trained for one scenario, are brittle due to overspecialization and tend to fail when transferred to similar or new scenarios. On the contrary, when a fresh human driver learns to understand and predict the behaviors of other drivers at one intersection, such an experience is omni-instructional, also helping to enhance behavior understanding and prediction capability in other intersections and roundabouts.

There is an obvious gap of capability between AVs and human drivers. We naturally wonder what is the secret in humans' brains, which allows us to understand and predict driving behavior so easily and efficiently. Evident from neuroscience, human's efficient shuttling in dense traffic flows and complex environments benefits from two cognition mechanisms: 1) hierarchy (Botvinick et al., 2009; Flet-Berliac, 2019) - cracking the entangling task into simpler sub-tasks; 2) selective attention (Niv, 2019; Radulescu et al., 2019) - identifying efficient and low-dimensional state representations among the huge information pool. Certainly, the two mechanisms are not mutually exclusive but are highly co-related. When dividing a complex task into easier sub-tasks, humans will choose a compact set of low-dimension states relevant to each sub-task respectively. An easy example can be found when a child learns to build a tower with blocks. Usually, the child would divide the overall task into a sub-task of searching for proper blocks and a sub-task of cautiously placing the blocks on the tower (Marcinowski et al., 2019; Spelke & Kinzler, 2007). In the high-level searching task, the child would care about the shape or weight of the blocks, but in the low-level placing task, the child would essentially pay attention to subtly adjusting the position and angle of the block. By choosing state features at different granularity (information hiding) and learning different skills separately (reward hiding) (Dayan & Hinton, 1993; Bacon et al., 2017), children are not only able to build the blocks efficiently and rapidly due to the simplified task and filtered state (efficient learning), but they are also capable of generalizing and reusing the two skills when they confront new scenarios or tasks (generalization).

The benefits of the two mechanisms in efficient learning and generalization are certainly fruitful for hierarchical methods, which end-to-end approaches (Salzmann et al., 2020; Codevilla et al., 2018) cannot enjoy. However, how much we can benefit from these two mechanisms significantly depends on how properly the hierarchies and relevant states are designed. To this point, there are some existing works (Zhao et al., 2020; Gao et al., 2020; Tang & Salakhutdinov, 2019) dividing the driving task into a high-level intention-determination task and a low-level action-execution task. An intention is usually defined as a goal point in the state space (Zhao et al., 2020; Ding et al., 2019; Sun et al., 2018a) or the latent space (Tang & Salakhutdinov, 2019; Rhinehart et al., 2019). Actions are then generated to reach that goal.

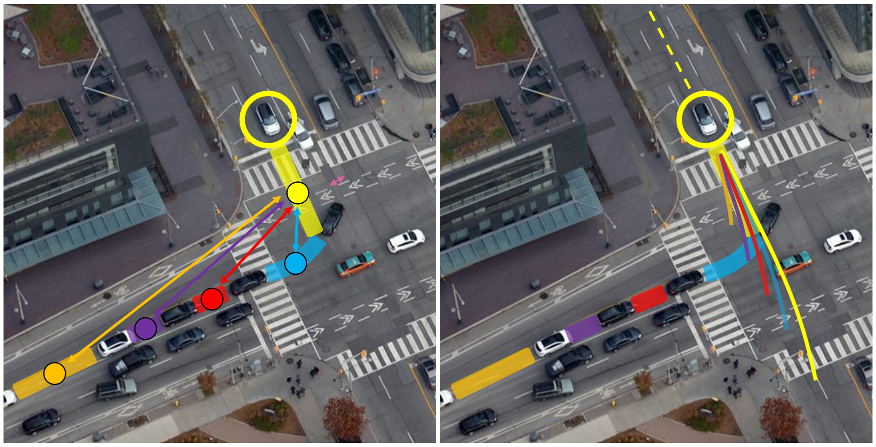
However, to gain human-level high-quality and transferable prediction capability, the definition of hierarchy should carry more semantics by referring to how humans think while driving (Shalev-Shwartz et al., 2017). When humans are shuttling through traffic flows, they first exhibit high-level intention to identify which "slot" is most spatially and temporally proper to insert into as shown in Figure 1(a). With the chosen slot to insert into and the map geometry, humans then will generate a desired reference line as a low-level action as shown in Figure 1(b). Then humans will polish their micro-action skills by optimizing how well they can track the reference line.

Such a hierarchical policy with more profound semantics enjoys many advantages. First, the policy is intrinsically scenario-transferable and reusable, because the representation of insertion slot and reference trajectory can be abstracted out and consistently defined across different scenarios. Second, the hierarchical design encourages efficient learning since each sub-task's state space is reduced where only relevant information for the sub-task is left, and each sub-policy's learns individually without information entanglement.

In addition, human behavior is naturally stochastic, heterogeneous, and time-varying. For instance, humans with different driving styles (Wang et al., 2021; Sun et al., 2018b; Schwarting et al., 2019) may result in distinct observed behaviors. Besides, though transferable, human behavior is still task-specific because there exist inevitable distribution shifts across scenarios, making the generalization harder. For instance, speed limits are set differently across different scenarios or cities, calling for driving customization on each scenario. Capturing such behavior variance can not only help to make more human-like customized behavior prediction for individuals, but also encourages better generalization across scenarios. As a result, an advanced prediction algorithm should also harness the power of online adaptation, to embrace the uncertainty in human behavior.

In summary, to generate high-quality, transferable and adaptable driving behavior prediction in multi-agent systems, we should not only design policies by leveraging human's intrinsic hierarchy and selective attention cognition model, but also capture diverse human behaviors with online adaptation methods. However, such a design is not trivial. Harmonious and natural divisions of hierarchies, along with compact and generic state representations, are crucial to achieve what we desire. Also, to seamlessly incorporate adaptation methods into the hierarchy policies, strict mathematical formulations and systematic analysis are required.

In this paper, we propose HATN (*Hierarchical Adaptable and Transferable Network*), a hierarchical framework for high-quality, transferable and adaptable behavior prediction in multi-agent traffic-dense driving environments. The framework consists of three parts: 1) a high-level semantic graph network (SGN) responsible for the slot-insertion task in multi-agent environments; 2) a low-level encoder decoder network (EDN)



(a) high-level intention policy (b) low-level trajectory policy

Figure 1: In dense-traffic multi-agent driving scenarios, human drivers have a hierarchical process in understanding and predicting driving behavior, which cares about different features in each hierarchy. Humans firstly predict which slot to insert into in the high-level intention hierarchy as in (a), paying high attention to features related to the slots on the scene. Then humans are predicted to execute actions to realize the intention in the low-level trajectory hierarchy as in (b), based on finer features related to vehicle dynamics. Such a hierarchical prediction process is 1) simplifying the learning as the whole task is divided into easier sub-tasks, which only takes in state relevant to its sub-goal; 2) intrinsically scenario-transferable as the representation for the “slots" and the “trajectory" can be defined consistently across different scenarios; 3) adaptable to individuals by leveraging historic behaviors to adjust model parameter.

which generates future trajectory according to historic dynamics and intention signals from the high-level policy; 3) an online adaptation module which applied modified extended Kalman filter (MEKF) algorithm to execute online adaptation for better individual customization and scenario transfer. To the best of our knowledge, this is the first method to explicitly and simultaneously take the driver’s nature of hierarchy, transferability, and adaptability into account.

In addition to being deployed for high-quality real-time transferable and adaptable trajectory predictions, the proposed method can have diverse practical applications. When the online adaptation module which requires the feedback signal is deactivated, the methods can also be applied across different scenarios as a planning system to: 1) generating more human-like, user-friendly and socially-compatible driving behaviors; 2) providing driving suggestions as an automatic driving assistance system, such as which slot to insert into; 3) reporting emergency alert when unsafe driving behavior happens. In the paper, we evaluate our method in the task of trajectory prediction since there are more abundant real data for numerical evaluation.

The key contributions of this paper are as follows:

1. Propose a hierarchical framework that takes novel sub-task definitions with compact and generic representations, and efficiently generates human-like driving behavior prediction in complex multi-agent intense-interaction environments, which is zero-shot transferable across scenarios.

2. Leverage online adaptation algorithms to capture behavior variance among individuals and scenarios for better customizability and transferability. A new set of metrics is proposed for the systematic evaluation of online adaptation performance.

3. Conduct extensive experiments on real human driving data, which include thorough ablation studies for each module of our method and show how our method outperforms other behavior forecasting methods, in terms of prediction accuracy, transferability and adaptability.

2 Related works

Our preliminary results were presented in non-archival workshop. The current version further provides: 1) comprehensive comparison with more state-of-the-art methods in Sec 7.4; 2) detailed description of the methodology in Sec 3 and semantic graph representation in Sec 4; 3) in-depth experiment for evaluation of the online adaptation in Appendix E; 4) discussion on the interacting agent density in Appendix F and algorithm running time Appendix G. In this section, we introduce related works on human behavior prediction categorized by methodology and property. The readers are referred to Rudenko et al. (2020) for a detailed survey.

2.1 Traditional prediction methods

The problem of predicting future motion for dynamic agents has been long studied in the literature. Classic physics-based methods include Intelligent Driver Model (Treiber et al., 2000), Kalman Filter (Elnagar, 2001), Rapidly Exploring Random Trees (Aoude et al., 2010), etc. These methods essentially analyze agents and propagate their historic and current state forward in time according to manually designed physical rules. Other classic optimization-based methods model humans as utility-maximization agents, whose future behavior can be predicted by assuming they are optimizing designed or learned reward (Fridovich-Keil et al., 2020; Wang et al., 2021). Other classic pattern-based methods classify driving motion into semantically interpretable maneuver classes, via Hidden Markov Model (Liu & Tomizuka, 2016; Deo et al., 2018), Gaussian Process (Zhang et al., 2021), and Bayesian Network (Schreier et al., 2014). Such classes are then used to facilitate intention-and-maneuver-aware prediction. These methods perform well in scenarios with simple road geometry and weak interaction like highways and straight streets. However, these methods struggle when confronting long-horizon prediction tasks or complex-road-geometry intense-interaction scenarios like crowded roundabouts and intersections. Such performance downgrade usually stems from the limited expressiveness of the model, insufficient interaction and context encoding, and laborious but uncomprehensive task-specific rule design.

2.2 Deep-learning-based prediction methods

The success of deep learning ushered in a variety of data-driven methods. Due to the temporal nature of the prediction task, these models often utilizes Recurrent Neural Network (RNN) variants to process temporal information (Park et al., 2018; Hu et al., 2018a; Zyner et al., 2019; Dequaire et al., 2018). To enhance interaction reasoning among agents, Convolution Neural Network (CNN) applies convolution operations on data (commonly in grid form) to model spatial and temporal relationship, such as 3D voxelization, rasterization in 2D bird's-eye view (BEV), or CNN feature maps (Radwan et al., 2020; Su et al., 2021). While these methods have shown great expressiveness with little human effort, the interaction among vehicles have been insufficiently reasoned. On the one hand, these methods lack flexibility as they usually only consider a fixed number of agents. On the other hand, These methods are also not order-invariant: processing agents in a different order would produce different results, while we would expect the same results for the same scene. Also, explicit relationship reasoning among agents is still missing. To tackle these drawbacks, Graph Neural Network (GNN) has been recently combined with RNN and CNN usually in an encoder decoder architecture to solve prediction tasks (Salzmann et al., 2020; Ding et al., 2019; Li et al., 2019; Ma et al., 2019; Choi et al., 2019; Li et al., 2021; Hu et al., 2020; 2018b). Due to the strong relational inductive bias of GNN (Battaglia et al., 2018), these GNN-based methods successfully achieve flexibility in agent number, ordering invariance, and explicit relationship reasoning. Consequently, in this paper, we also adopt a GNN-based architecture.

2.3 Hierarchical prediction

In the driving task, there inherently exists a high-level intention determination sub-task and a low-level motion execution sub-task. Thus the methods which predicting future motion in an end-to-end manner (Salzmann et al., 2020; Li et al., 2019; Ma et al., 2019) are usually hard to learn, explain, and verify. To this point, some existing methods exploit the hierarchies in the driving task (Gao et al., 2020; Choi et al., 2019; Li et al., 2021). An intention is usually defined as a goal point in the state space (Ding et al., 2019; Sun et al., 2018a) or the latent space (Tang & Salakhutdinov, 2019; Rhinehart et al., 2019). However, though equipped with hierarchies, most of these methods are still trained in an end-to-end manner. Consequently, the two benefits of hierarchies, task simplification and representation filtering, are hardly exploited since the model is still monolithically trained. There are a few works that not only utilizes a hierarchical design, but also trains the model hierarchically in two stages (Zhao et al., 2020) . However, the representation of intention in these works is far from generic, and the transferability of the algorithm is still ignored and not verified. In comparison, our method is not only able to monitor and refine the learning in each hierarchy, but also enhance transferability by designing generic and compact representation for each sub-policy so that they can be reused in different scenarios.

2.4 Transferable prediction

It is desired to have omnipotent prediction algorithms that can be applied to many different scenarios such as highways, intersections, and roundabouts. However, most existing methods either focus on certain scenarios (Ding et al., 2019; Choi et al., 2019), or train and apply their methods in data from various scenarios without assessing their scenario-wise performance (Salzmann et al., 2020; Li et al., 2019; Ma et al., 2019; Li et al., 2021). As a result, these methods tend to fail when transferred to novel scenarios without learning a new set of model parameters. Many factors contributes to such brittleness, while the representation may be the first to be blame. The input features of these methods are usually in Cartesian coordinate frame or scene images, which leads to two drawbacks: 1) scenario-specific information such as traffic regulations and road geometries are softly and insufficiently incorporated if not completely ignored. 2) these representations are not generic and will change each time a new scenario is encountered.

There have been effective representation definition for efficient and generalizable learning in some tasks, such as the visuomotor control of humanoids where the environment is divided into proprioceptive and exteroceptive state (Merel et al., 2018; Hasenclever et al., 2020), and the visual navigation task where image target is defined Zhu et al. (2017). However, in the driving context where complex road/traffic information and intense interaction exist, very few works can address the representation issues. One recent work (Hu et al., 2020) achieves transferable prediction by designing generic representations called Semantic Graph, where Dynamic Insertion Areas rather than road entities are regarded as the nodes. In such representations, scenario-specific information such as road geometry and traffic regulations is naturally and comprehensively incorporated in a generic manner. However, this work only considers the high-level intention prediction sub-task and cannot generate motion trajectory for practical use. In contrast, our paper adopts such generic representations and train the model hierarchically, which enables us to conduct both intention prediction and motion prediction simultaneously.

2.5 Adaptable prediction

Apart from hierarchies and transferability, adaptability is another desired property of the prediction algorithms since there inevitably exists behavior variance in different individuals and scenarios. Consequently, online adaptation has been recently studied in the human prediction research field. Cheng et al. (2020; 2019); Si et al. (2019) adopts recursive least square parameter adaptation algorithm (RLS-PAA) to adapt the parameter of the last layer in the neural network. To be able to conduct multi-step adaptation on random sets of parameters, Abuduweili & Liu (2021); Liu & Liu (2021); Abuduweili & Liu (2020) utilize modified extended Kalman filter (MEKF$_\lambda$) to conduct model linearization and empirically compare the results of adapting different sets of parameters with different steps. The recent work most related to ours (Abuduweili & Liu, 2021) adopts an encoder-decoder architecture for multi-task prediction, namely tasks of intention and trajectory prediction. However, compared to our method, the benefit of GNN, hierarchical training, and transferable generic representation is not considered.

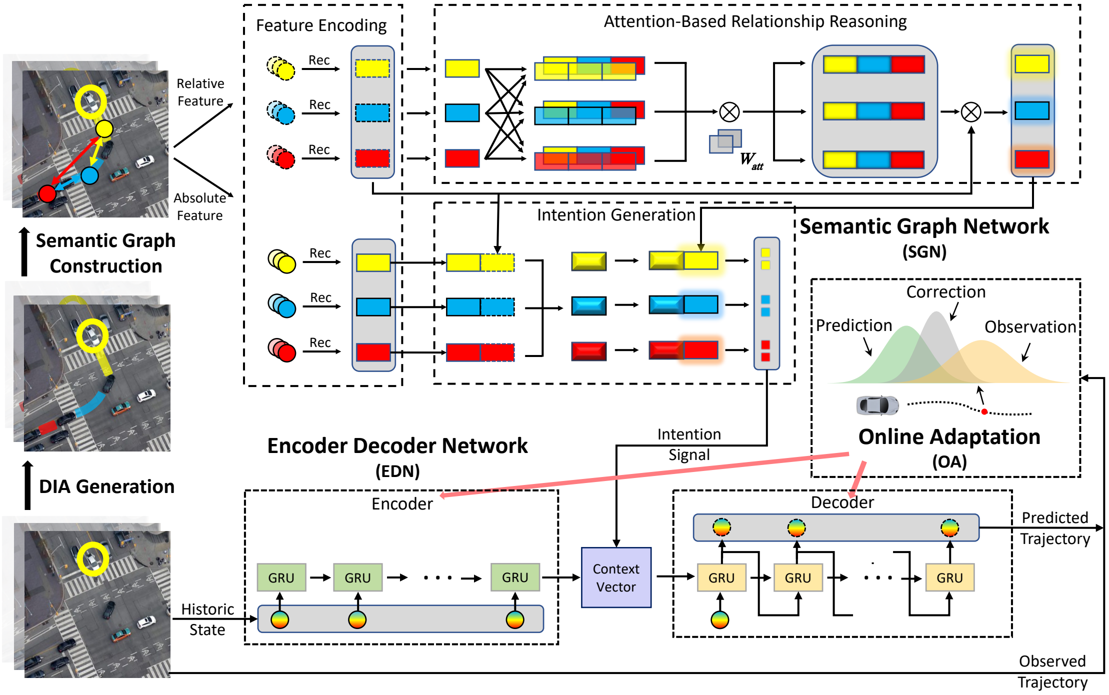


Figure 2: The proposed HATN framework consists of four parts: 1) on the left of the image, we extract ego vehicles' interacting cars and construct a Semantic Graph (SG). In the SG, Dynamic Insertion Areas (DIA) are defined as the nodes of the graph, which the ego vehicle can choose to insert into. 2) Taking the SG as the input, the high-level Semantic Graph Network (SGN) is responsible for reasoning about relationships among vehicles and predicting the intentions of individual vehicles, such as which area to insert into and the corresponding goal state. 3) The low-level Encoder Decoder Network (EDN) takes in each vehicle's historic dynamics and intention signal to predict their future trajectories. 4) The Online Adaptation (OA) module can online adapt EDN's parameter based on historic prediction errors, which captures behavior pattern of different individuals and scenarios. The input and output for each module are clarified formally in Table 1.

3 Problem formulation

With the proposed HATN framework, we aim at generating high-fidelity predictions of driving behaviors in multi-agent traffic-dense scenarios[1]. Specifically, with the proposed method shown in Figure 2, we focus on generating behavior predictions for any selected car (which will be called the ego car in the following discussion) and the cars interacting with the ego car in the next T_f seconds $\hat{\mathbf{Y}}_{t+1,t+T_f}$, based on observations of the last T_h seconds $\mathbf{O}_{t-T_h,t}$:

$$\hat{\mathbf{Y}}_{t+1,t+T_f} = f_{HATN}(\mathbf{O}_{t-T_h,t}). \tag{1}$$

High-fidelity trajectory prediction is challenging especially when the horizon extends, where the intention and inter-vehicle interaction have increasingly larger impacts on driving decisions. Evident from cognition science (as discussed in Sec 1), when humans drive in dense traffic flows, their decision-making policy naturally consist of hierarchies. Specifically, in the high-level hierarchy, human drivers intuitively search for the proper slot to insert into. Thus we first adopt a generic representation about the environment called the semantic graph (SG). In SG, dynamic insertion areas (DIA) are defined as the node of the graph, among which the vehicles can decide to insert into or not. Such a representation is compact, efficient, and generic, which captures sufficient information for intention determination and can be generically used across different driving scenarios. Illustrated in the left part of Figure 2, the process of extracting semantic graph

---

[1] This paper considers intersection and roundabout scenarios, which are relatively interaction-intense, while our method can be easily applied to other scenarios like highway and parking lot.

representation $\mathcal{G}_{t-T_h,t}$ from raw observations $\mathbf{O}_{t-T_h,t}$ can be formally described:

$$\mathcal{G}_{t-T_h,t} = f_{SG}(\mathbf{O}_{t-T_h,t}), \tag{2}$$

where $\mathcal{G}_{t-T_h,t} \in \mathbb{R}^{M \times T_h \times N_1}$ denotes the extracted semantic graph, consisting of M DIAs from the past T_h step, each with N_1 features. $\mathbf{O}_{t-T_h,t} \in \mathbb{R}^{M \times T_h \times N_2 + K}$ is the environment observation including K reference lines, and M vehicles in the past T_h steps, each with N_2 features. f_{SG} denotes the SG extraction function that selects cars interacting with ego car, extracts DIAs, and constructs SG.

With the semantic graph, we then propose a semantic graph network (SGN), which takes the semantic graph as input, inferences relationships and interactions among vehicles, and outputs the probability for ego car to insert into each of the M DIAs $w_t \in \mathbb{R}^M$ and the goal state distribution $g_t \in \mathbb{R}^{M \times J}$ for ego car and the interacting cars:

$$w_t, g_t = f_{SGN}(\mathcal{G}_{t-T_h,t}), \tag{3}$$

where for each of the M vehicles, J parameters are used to describe the distribution of the goal state.

In the low-level hierarchy, our insight is that when humans drive, conditional on the goal state as the intention signal, they conceptually track a reference trajectory via micro muscle actions. Thus we design a low-level trajectory-generation policy to imitate such behavior. Besides the intention signal, the trajectory generation procedure should also be subject to the instantaneous dynamics of the vehicles, which requires the encoding of the historic state $\mathbf{S}_{t-T_h,t}$. Furthermore, the policy should also be able to express various motion patterns, i.e. constant velocity and varying acceleration. To ensure dynamics continuity and motion diversity, we use the encoder decoder network (EDN) as the behavior-generation model, where the historic dynamics are processed by the encoder and then used by the decoder to generate diverse maneuvers. With model parameter θ, the task of EDN is illustrated as in the lower part of Figure 2 and can be summarized as:

$$\hat{\mathbf{Y}}_{t+1,t+T_f} = f_{EDN}(\mathbf{S}_{t-T_h,t}, g_t, \theta), \tag{4}$$

where $\mathbf{S}_{t-T_h,t} \in \mathbb{R}^{M \times T_h \times N_3}$ denotes the state of M vehicles in the past T_h time step, each with N_3 features. $\hat{\mathbf{Y}}_{t+1,t+T_f} \in \mathbb{R}^{M \times T_f \times N_4}$ denotes the prediction for the M vehicles in the future T_f time steps, each with N_4 features.

Up to this point, our model is capable of generating high-level intentions and low-level trajectories efficiently and transferably. However, the trained model can only capture the motion pattern in an average sense, while the nuances among individuals are hardly reflected. Besides, the behavior patterns also vary with different scenarios. To capture these behavior nuances across different individuals and scenarios, we set up an online adaptation module (OA), where a modified Extended Kalman Filter (MEFK$_\lambda$) algorithm is used to moderately adjust the model parameters for each agent based on its historic behaviors. Specifically, we regard EDN as a dynamic system and estimate its parameter θ by minimizing the error between ground-truth trajectory $\mathbf{Y}_{t-\tau,t}$ and predicted trajectory $\hat{\mathbf{Y}}_{t-\tau,t}$ in the past τ steps:

$$\theta_t = f_{MEKF_\lambda}(\theta_{t-1}, \mathbf{Y}_{t-\tau,t}, \hat{\mathbf{Y}}_{t-\tau,t}), \tag{5}$$

where $\hat{\mathbf{Y}}_{t-\tau,t} \in \mathbb{R}^{M \times \tau \times N_4}$ denotes the prediction for M vehicles from τ time step earlier to now, and $\hat{\mathbf{Y}}_{t-\tau,t} \in \mathbb{R}^{M \times \tau \times N_4}$ denotes the ground-truth observation of the M vehicles in the past τ time steps, each with N_4 features.

The rest of this paper is organized as follows. In Sec 4, we introduce the high-level intention-identification policy in detail, including the definition of semantic graph (SG) and the architecture of semantic graph network (SGN). In Sec. 5, we describe the low-level behavior-generation policy, including the design of encoder decoder network (EDN) and the method of integrating the intention signal into the EDN. In Sec. 6, we introduce the formulation and the utilized algorithm (MEKF$_\lambda$) of online adaptation module. Note that the possible design choices of each module are systematically discussed in each corresponding section. In Sec. 7, we conduct extensive empirical studies of our methods on real data, including a case study illustrating how our method works in Sec. 7.2, a brief summary of ablation studies on each module as a quick takeaway for the reader in Sec 7.3, and a comparison with other methods in Sec 7.4. More detailed ablation studies on each module to empirically find the optimal design choice can be found in Appendix B C D E.

Table 1: Input and output for each module in the proposed framework.

Module	Input	Output
SGN	Semantic graph (Dynamic insertion area)	Probability distribution of future inserting area and goal state
EDN	Most likely goal state, Historic dynamics	Most likely future trajectory
OA	Historic observation, Historic prediction, Prior EDN parameter	Updated EDN parameter, Adapted future trajectory

4 High-level intention-identification policy

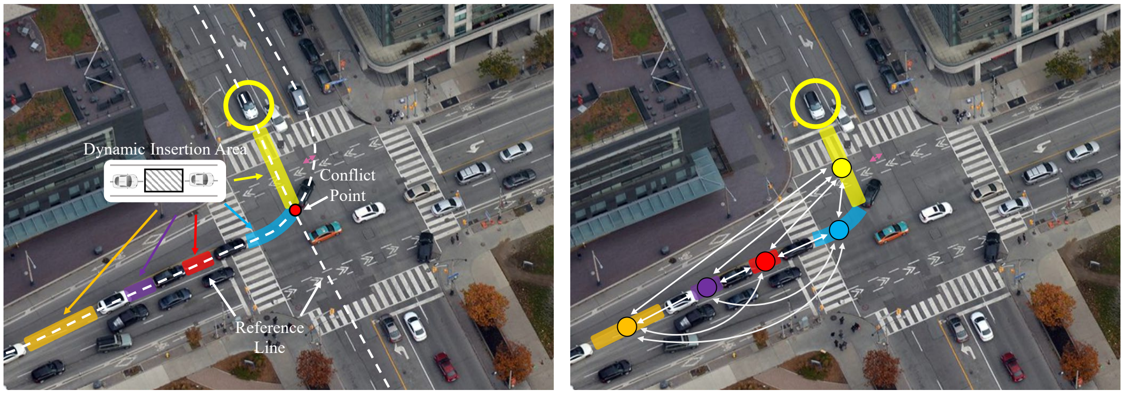
Human behaviors are usually hierarchically divided for better efficiency and generalizability. In this section, we introduce the high-level intention-identification policy in detail, including design insight, the definition of semantic graph (SG), and the architecture of semantic graph network (SGN).

In the high-level policy, humans usually take in low-dimension state feature to make decisions at a low resolution. Specifically in the driving task, human drivers first make a high-level decision on which area on the road is the most temporally and spatially suitable to insert into. Such areas are usually formed by the slots between cars, traffic signs, and road geometries. To imitate humans' intention of inserting into slots in dense traffic, we first adopt the dynamic insertion area (DIA) introduced in Hu et al. (2020) to define the slot formally. The extracted DIAs are then regarded as nodes to form a semantic graph (SG) to construct a generic and compact representation of the scenario. We then introduce the semantic graph network (SGN) which generates agents' intention by reasoning about their internal relationships. The advantages of adopting dynamic insertion area are threefold: (1) It explicitly describes humans' insertion behavior considering the map, traffic regulations, and interaction information. (2) It filters scene information and only extracts a compact set of vehicles and states crucial for the intention prediction task. (3) DIA is a generic representation, which can be used across different scenarios.

4.1 Semantic graph

The semantic graph (SG) utilizes dynamic insertion areas (DIA) as basic nodes for a generic spatial-temporal representation of the environment. As shown in Figure 3, when extracting DIAs from the scene, we first identify each agent's reference line by Dynamic Time Warping algorithm (Berndt & Clifford, 1994). Next we identify interacting cars whose lane reference crosses with the ego car's lane reference. Interaction essentially happens among these cars and the ego car as they are driving into a common area (the conflict point) (Markkula et al., 2020). Then DIAs are extracted by definition: *a dynamic area that can be inserted or entered by ego agent on the road.* Each DIA consists of a front boundary formed by a front agent, a rear boundary formed by a rear agent, and two side boundaries formed by reference lines. To capture each DIA's crucial information for humans' decision, we extract four high-level features under the Frenet coordinate: $d^{lon}_{f/r}$ denotes longitudinal distance to the conflict point of front or rear boundary; $v_{f/r}$ denotes the velocity of front or read boundary; $\phi_{f/r}$ denotes the angle of front or read boundary; $l = d^{lon}_{r} - d^{lon}_{f}$ measures the length of the DIA. To facilitate relationship inference among DIAs, we also define the relative feature for each DIA by aligning it with the reference DIA. Note that we choose the front DIA as the reference DIA because the ego vehicle is implicitly represented by the rear boundary of the front DIA.

With the extracted DIAs as nodes, the 3D spatial-temporal semantic graph $\mathcal{G}^{t-T_h \to t} = (\mathcal{N}^{t-T_h \to t}, \mathcal{E}^{t-T_h \to t})$ can be constructed, where $t - T_h \to t$ denotes the time span from a previous time step $t - T_h$ to current time step t with T_h denoting the horizon. Such a representation differs from other methods in the criteria of choosing interacting cars and the definition of node in the graph, which is discussed in detail in Appendix F. The readers are referred to Hu et al. (2020) for more detailed description on the DIA properties and DIA extraction algorithm.



(a) Dynamic Insertion Area (DIA) Extraction (b) Semantic Graph (SG) Construction

Figure 3: Dynamic insertion area (DIA) extraction and semantic graph (SG) construction procedure: when other car's lane reference line crosses ego car's lane reference line, based on the conflict point, the DIAs are extracted and regarded as nodes to construct the SG.

4.2 Semantic graph network

As the high-level intention identification policy, the architecture of the SGN is shown in Figure 2. SGN takes the spatial-temporal 3D semantic graph from historic time step $t-T_h$ to the current time step t as the input, rather than only the spatial 2D semantic graph of the current time step t in previous work (Hu et al., 2020). Such a change aims at capturing more temporal dynamics and interactions among vehicles. SGN then decides which area to insert into and generate the associated goal state distribution. For simplicity, the mean of the goal state distribution is then delivered to a low-level policy for generating more human-like behaviors. The impact of sampling in the goal state distribution is left for future work to discuss.

4.2.1 Feature encoding layer

In this layer, we essentially encode the absolute and relative features for each node from historic time step $t-T_h$ to the current time step t:

$$h_i^t = f_{rec}^1(\mathbf{X}_i^t), \tag{6}$$

$$h'^t_i = f_{rec}^2(\mathbf{X}'^t_i), \tag{7}$$

where $\mathbf{X}_i^t = [x_i^{t-T_h}, ..., x_i^t]$ and $\mathbf{X}'^t_i = [x'^{t-T_h}_i, ..., x'^t_i]$ respectively denote the absolute features and relative features of node i from time step $t-T_f$ to current time t; h_i^t and h'^t_i denote the hidden states encoded from absolute and relative features respectively, namely the outputs of the recurrent function f_{rec}^1 and f_{rec}^2. h_i^t and h'^t_i are further embedded for later use.

$$\hat{h}_i^t = f_{enc}^1(h_i^t), \tag{8}$$

$$\hat{h}'^t_i = f_{enc}^2(h'^t_i). \tag{9}$$

4.2.2 Attention-based relationship reasoning layer

To infer relationships between any two nodes, inspired by Graph Attention Network (Veličković et al., 2017), we design an attention-based relationship reasoning layer. In this layer, we exploit the soft-attention mechanism (Luong et al., 2015; Bahdanau et al., 2014) to compute the node n_i's attention coefficients on node n_j:

$$a_{ji}^t = f_{att}(concat(\hat{h}'^t_j, \hat{h}'^t_i); \mathbf{W}_{att}), \tag{10}$$

where function f_{att} maps each concatenated two features into a scalar with the parameter $\mathbf{W}_{att}$. The attention coefficient is then normalized across all nodes $\mathcal{N}^t$ at time step t:

$$\alpha_{ji}^t = \frac{exp(a_{ji}^t)}{\sum_{n\in\mathcal{N}^t} exp(a_{ni}^t)}. \tag{11}$$

Eventually, node i's relationships with all nodes in the graph (including node i itself) are derived by the attention-weighted summation of all encoded relative features:

$$\bar{h}_i^t = \sum_{n \in \mathcal{N}^t} \alpha_{ni}^t \odot \hat{h}'^t_n, \tag{12}$$

where $\odot$ denote element-wise multiplication.

4.2.3 Intention generation layer

When predicting the intention, in addition to the node relationships, each node's own features are also required. Thus we first concatenate and encode each node's embedded absolute and relative feature:

$$\widetilde{h}_i^t = f_{enc}^3(concat(\hat{h}_i^t, \hat{h}'^t_i)). \tag{13}$$

Each DIA's future evolution in the latent space is then derived by combining its relationships with other nodes and its own features:

$$z_i^t = f_{enc}^4(concat(\bar{h}_i^t, \widetilde{h}_i^t)). \tag{14}$$

In this paper, the intention is defined as which DIA the ego car decides to insert into. Thus the latent vector representing each DIA's evolution is then used to generate the probability of being inserted by ego vehicle:

$$w_i^t = \frac{1}{1 + exp(f_{out}^1(z_i^t))}, \tag{15}$$

which is then normalized across all DIAs in current time step such that $\sum_{i \in \mathcal{N}^t} w_i^t = 1$. Moreover, to generate a practical intention signal for low-level policy to leverage, we further use a Gaussian Mixture Model (GMM) to generate a probabilistic distribution over each DIA's future goal state g in a certain horizon[2]:

$$f(g_i^t | z_i^t) = f(g_i^t | f_{out}^2(z_i^t)), \tag{16}$$

where the function f_{out}^2 maps the latent state z_i^t to the parameters of GMM (i.e. mixing coefficient α, mean μ, and covariance σ). The goal state g then can be retrieved by sampling in the GMM distribution.

4.2.4 Loss function

We not only expect the largest probability to be associated with the actual inserted area ($\mathcal{L}_{class}$), but also the ground-truth goal state to achieve the highest probability in the output distribution ($\mathcal{L}_{regress}$). Thus we define the loss function as:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{regress} + \beta \mathcal{L}_{class} \\ &= -\sum_{\mathcal{G}_s} \left(\sum_{i \in \mathcal{N}^s} log\Big(p\big(\breve{g}_i | f_{out}^2(z_i)\big)\Big) + \beta \sum_{i \in \mathcal{N}^s} \breve{w}_i log(w_i) \right), \end{aligned} \tag{17}$$

where $\mathcal{G}_s$ denotes all the training graph samples; $\mathcal{N}^s$ denotes all the nodes in one training graph sample; $\breve{g}_i$ and $\breve{w}_i$ denote the ground-truth label for goal state and insertion probability of node n_i. Though our goal is to predict the ego vehicle's future motion, we output the goal state for all interacting vehicles rather than only the ego vehicle (as done in Hu et al. (2020)) to encourage sufficient reasoning of interactions, and also realize data augmentation.

Note that though only the goal state g_t is delivered and used in the downstream low-level behavior prediction, the learning for insertion probability w_t serves as an auxiliary task to stabilize the goal state learning (Mirowski et al., 2016; Hasenclever et al., 2020). Defining goal state in the state space instead of the latent space also offers us accessible labels to monitor the high-level policy learning and provides more interpretability of the model. The detailed description of the layers can be found in Table 4 of the appendix, and detailed empirical evaluations on the high-level policy can be found in Appendix B.

---

[2] In this paper, we use the relative traveled distance in future 3 seconds as the goal state representation

5 Low-level behavior-generation policy

Once the high-level policy determines where to go, the low-level policy is then responsible to achieve that goal by processing information at a finer granularity. Thus in this section, we describe the low-level behavior-generation policy, including the architecture of encoder decoder network (EDN), the methods of integrating the intention signal into the EDN, and possible design choices which requires empirical studies at the end of the section.

In the driving task, humans will generate a sequence of micro actions like steering and acceleration based on vehicle dynamics to reach their goals. To generate future behaviors of arbitrary length and ensure sufficient expressiveness, we use the encoder-decoder network (EDN) (Cho et al., 2014; Neubig, 2017) as the low-level behavior-generation policy, given the historic information and intention signal from the high-level policy. The low-level policy enjoys two benefits from the hierarchical design: 1) the learning is simplified as the low-level policy only needs to care about vehicle's own dynamics, while the consideration for interactions, collision avoidance, road geometries are left to the high-level policy to take care (information hiding); 2) the policy is only optimized for reaching the goal (reward hiding), so that the effect of different design and training tricks can be better verified explicitly; 3) the learned policy is tranferable and reusable in different scenarios.

5.1 Encoder decoder network

The EDN consists of two GRU (graph recurrent unit) networks, namely the encoder and the decoder. At any time step t, the encoder takes in the sequence of historic and current vehicle states $\mathbf{S}_t = [\mathbf{s}_{t-T_h}, ...\mathbf{s}_t]$, and compresses all information into a context vector c_t. The context vector is then fed into the decoder as the initial hidden state to recursively generate future behaviors $\hat{\mathbf{Y}}_t = [\hat{\mathbf{y}}_{t+1}, ..., \hat{\mathbf{y}}_{t+T_f}]$. Specifically, the decoder takes the vehicle's current state as the initial input to generate the first-step behavior. In every following step, the decoder takes the output value of the last step as the input to generate a new output. Mathematically, the relationship among encoder, decoder, and context vector can be summarized:

$$c_t = f_{enc}(\mathbf{S}_t; \theta^E), \tag{18}$$

$$\hat{\mathbf{Y}}_t = f_{dec}(c_t, s_t; \theta^D), \tag{19}$$

where the context vector c_t is the last hidden state of the encoder and is also used as the decoder's initial hidden state; the current state $\mathbf{s}_t$ is fed as the decoder's first-step input. In this paper, we use a single-layer GRU and stack three dense layers on the decoder for stronger decoding capability.

The goal of EDN is to minimize the error between the ground-truth trajectory and the generated trajectory. Taking a deterministic approach, the loss function is simply designed to be:

$$\mathcal{L} = \sum_{i=0}^{N} ||\hat{\mathbf{Y}}_i - \mathbf{Y}_i||_p, \tag{20}$$

where N denotes the number of training trajectory samples. The objective can be measured in any l_p norm, while in this paper we consider l_2 norm.

5.2 Integrating the intention signal

The EDN can be regarded as a motion generator given the historic dynamics, while the encoding for interaction and map information is left to the high-level intention policy to handle. Such a hierarchical policy simplifies the learning burden for each sub-policy and offers better interpretability. However, it remains unknown *what* intention signals should be considered and *how* to integrate them into the low-level policy.

In our case, we aim at generating high-fidelity human-like predictions in a certain future horizon. So we naturally expect the intention signals to include the goal state g_t in the future horizon to guide the EDN's generation process. Besides, considering the fact that the same GRU cell is recursively utilized at each step, the GRU cell is unaware of whether the current decoding lies in the earlier horizon or the later horizon. Consequently, we introduce the current decoding step as another intention signal to help the decoder to better track the goal state g_t. Introducing these intention signals would then modify the decoder definition from Eq (19):

$$\hat{\mathbf{Y}}_t = f_{dec}(c_t, s_t, g_t; \theta^D). \tag{21}$$

There are various ways to incorporate the intention signal into the time series model. In general, when the additional feature is a temporal series, it is intuitive to append it to the end of the original input feature vector or output vector of the GRU (before the dense layers) as in Cheng et al. (2020; 2019). However, when we have a non-temporal-series additional feature, directly appending it to the original feature vector may create harder learning by polluting the temporal structure. A more delicate approach is to embed the additional feature with a dense layer and add it to the hidden state of RNN at the first-step decoding, so that the non-temporal signal is passed in the GRU cell state along the decoding sequence as in Karpathy & Fei-Fei (2015); Vinyals et al. (2015). Besides, in our case, the goal state intention signal is defined as the goal position in the physical world, so another approach is to directly transform the original input state to the state relative to the goal state, such that the model is implicitly told to reach origin at the last step of decoding.

Besides how to incorporate the intention signal, it also remains unclear what coordinate, features, and representation should be employed for the best performance? Thus we conducted systematic experiments to evaluate the effect of these factors. A brief summary of these experiments is provided in Sec 7.3, while detailed evaluations can be found in Appendix C D. We empirically found that in the test data distribution, the best performance goes: 1) in frenet coordinate, 2) including input features like velocity and yaw, 3) applying representation trick like incremental prediction and position alignment, 4) appending intention signal like goal state and decoding step into the input feature.

6 Online adaptation

In this section, we introduce the motivation, formulation and the utilized algorithm (MEKF$_\lambda$) of online adaptation module. Possible design choices are also discussed at the end of the section.

Though humans are usually assumed to be rational, the standards of "optimal plan" may still vary across different agents or circumstances (Baker et al., 2006; 2007). Consequently, human behaviors are naturally heterogeneous, stochastic, and time-varying. Different driving scenarios also inevitably create additional behavior shifts. We thus utilize online adaptation to inject customized individual and scenario patterns into the model. The key insight for online adaptation is that, though drivers cannot communicate directly, their historic behaviors can be a vital clue for their driving patterns, based on which we can adapt parameters of our model to better fit the individual or scenario.

6.1 Multi-step feedback adaptation formulation

The goal of online adaptation is to improve the quality of behavior prediction with feedback from the historic ground-truth information. In our case, the policy is hierarchically divided, with two sub-policies to be adapted. Due to the large delay in obtaining the long-term ground-truth intention label, we only consider the online adaptation for the low-level behavior-prediction policy, while keeping the high-level intention-identification policy intact. The intuition behind the online adaptation is thus that, though given the same goal state, drivers still have diverse ways to achieve it. Capturing such customized patterns can improve the human-likeness of generated behavior.

Formally, at time step t, online adaptation aims at exploiting local over-fitting to improve individual behavior prediction quality:

$$\min_{\theta} ||\hat{\mathbf{Y}}_t - \mathbf{Y}_t||_p, \tag{22}$$

where $\mathbf{Y}_t$ is the ground-truth trajectory; $\hat{\mathbf{Y}}_t$ is the predicted future trajectory by the EDN with the model parameter θ. Assume that the model parameter changes slowly, namely $\dot{\theta} \approx 0$. Then the model parameter that generates the best predictions in the future can be approximated by the model parameter that best fits the historic ground-truth observation. Also note that online adaptation can be iteratively executed for one agent once a new observation is received.

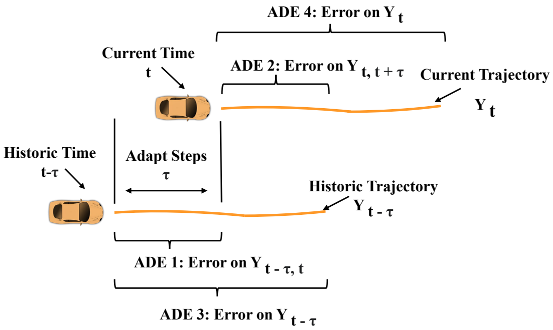


Figure 4: An illustration of online adaptation and 4 metrics for performance analysis. At time step t, the model parameters can be adapted by minimizing prediction error of the trajectory in past τ steps $\mathbf{Y}_{t-\tau,t}$. The adapted parameter is then used to generate prediction $\mathbf{Y}_t$ on current time t. A new set of 4 metrics is proposed to analyze the adaptation performance. ADE 1 can verify how the adaptation works on the source trajectory $\mathbf{Y}_{t-\tau,t}$. ADE 2 can verify whether adaptation can benefit short-term prediction in the presence of behavior gap between earlier time and current time. ADE 3 can verify whether we have obtained enough information from the source trajectory $\mathbf{Y}_{t-\tau,t}$. ADE 4 verifies whether adaptation can benefit the long-term prediction in the current time.

Practically, since online adaptation can be conducted as soon as at least one-step new observation is available, the length of ground-truth observation τ dose not necessarily have to match the behavior generation horizon T_f. Thus the online adaptation is indeed a multistep feedback strategy (Abuduweili & Liu, 2021). By definition, at time step t, we have the recent τ step ground-truth observation $\mathbf{Y}_{t-\tau,t} = [y_{t-\tau+1}, y_{t-\tau+2}, ..., y_t]$. From the memory buffer we also have the generated behavior at $t-\tau$ steps earlier $\hat{\mathbf{Y}}_{t-\tau,t} = [\hat{y}_{t-\tau+1}, \hat{y}_{t-\tau+2}, ..., \hat{y}_t]$. Then the model parameter can be adapted based on the recent τ-step error:

$$\hat{\theta}_t = f_{adapt}(\hat{\theta}_{t-1}, \hat{\mathbf{Y}}_{t-\tau,t}, \mathbf{Y}_{t-\tau,t}), \tag{23}$$

where f_{adapt} denotes the adaptation algorithm to be discussed in detail in Sec 6.2. The adapted model is then used to generate behaviors in the future T_f steps from the current time t. It is worth noting that here only τ-steps errors are utilized and we expect better performance in T_f steps. Nevertheless, behavior prediction error usually grows exponentially as the horizon extends. When τ is too small, we may not obtain enough information for the online adaptation to benefit behavior prediction in the whole future T_f step horizon. Intuitively, the problem can be mitigated by using errors of more steps. However, there exists a τ-step time lag in τ-step adaptation strategy. Too many steps may also create a big gap between historic behavior and current behavior, so that the model adapted at an earlier time may be outdated and incapable of tracking the current behavior pattern. Thus there is indeed a trade-off between obtaining more information and maintaining behavior continuity when we increase observation steps τ. Consequently, we propose a new set of metrics shown in Figure 4 to investigate the such a trade-off in detail:

1. ADE 1: This metric evaluates the prediction error of the adapted steps on the historic trajectory $Y_{t-\tau,t}$. Because these steps are the observation source used to conduct online adaptation, the metric can verify whether the algorithm is working or not.
2. ADE 2: This metric evaluates the prediction error of the adaptation steps on the current trajectory, which aims at verifying how the time lag is influencing the adaptation. Also, this method can be used to verify whether adaptation could improve short-term behavior prediction.
3. ADE 3: This metric evaluates the prediction error of the whole historic trajectory, which shows if we have gotten enough information on the behavior pattern.
4. ADE 4: This metric evaluates the prediction performance of the whole current trajectory, which shows whether or not the adaptation based on historic information can help current long-term behavior prediction.

The effect of utilizing different steps' observation is briefly discussed in 7.3 and analysed in detail in Appendix E, where we empirically found that in the tested data distribution, the best performance goes when 2 or 3 steps of observation are used.

Algorithm 1 τ step online adaptation with MEKF$_\lambda$

Input: Offline trained EDN network with parameter θ, initial variance $\mathbf{Q}_t$ and $\mathbf{R}_t$ for measurement noise and process noise respectively, forgetting factor λ

Output: A sequence of generated future behavior $\{\hat{\mathbf{Y}}_t\}_{t=1}^{T}$

```
for $t = 1, 2, ..., T$ do
    if $t \geq \tau$ then
        stack recent $\tau$-step observations:
            $\mathbf{Y}_{t-\tau,t} = [y_{t-\tau+1}, ..., y_t]$
        stack recent $\tau$-step generated trajectory:
            $\hat{\mathbf{Y}}_{t-\tau,t} = [\hat{y}_{t-\tau+1}, ..., \hat{y}_t]$
        adapt model parameter via MEKF$_\lambda$:
            $\hat{\theta}_t = f_{MEKF_\lambda}(\theta_{t-1}, \hat{\mathbf{Y}}_{t-\tau,t}, \mathbf{Y}_{t-\tau,t})$
    else
        initialization: $\hat{\theta}_t = \theta$
    end if
    collect goal state $g_t$ from SGN and input features $\mathbf{S}_t$.
    generate future behavior:
        $\hat{\mathbf{Y}}_t = [\hat{y}_{t+1}, ...\hat{y}_{t+T_f}] = f_{EDN}(\mathbf{S}_t, g_t, \hat{\theta}_t)$.
end for
return sequence of behaviors $\{\hat{\mathbf{Y}}_t\}_{t=1}^{T}$
```

6.2 Robust nonlinear adaptation algorithms

There are many online adaptation approaches, such as stochastic gradient descent (SGD) (Bhasin et al., 2012), recursive least square parameter adaptation algorithm (RLS-PAA) (Ljung & Priouret, 1991). In this paper, we choose the modified extended Kalman filter with forgetting factors (MEKF$_\lambda$) (Abuduweili & Liu, 2021) as the adaptation algorithm due to its robustness to data noises and efficient use of second-order information. Compared to the previous work (Abuduweili & Liu, 2021), we use the method for driving behavior prediction, a more complex problem.

The MEKF$_\lambda$ regards the adaptation of a neural network as a parameter estimation process of a nonlinear system with noise:

$$\mathbf{Y}_t = f_{EDN}(\hat{\theta}_t, \mathbf{S}_t) + \mathbf{u}_t \tag{24}$$

$$\hat{\theta}_t = \hat{\theta}_{t-1} + \omega_t \tag{25}$$

where $\mathbf{Y}_t$ is the observation of the ground-truth trajectory; $\hat{\mathbf{Y}}_t = f_{EDN}(\hat{\theta}_t, \mathbf{S}_t)$ is the generated behavior by the EDN policy f_{EDN} with the input $\mathbf{S}_t$ at time step t; $\hat{\theta}_t$ is the estimate of the model parameter of the EDN; the measurement noise $u_t \sim \mathcal{N}(0, \mathbf{R}_t)$ and the process noise $\omega_t \sim \mathcal{N}(0, \mathbf{Q}_t)$ are assumed to be Gaussian with zero mean and white noise. Since the correlation among noises are unknown, it is reasonable to assume noises are identical and independent of each other. For simplicity, we assume $\mathbf{Q}_t = \sigma_q \mathbf{I}$ and $\mathbf{R}_t = \sigma_r \mathbf{I}$ where $\sigma_q > 0$ and $\sigma_r > 0$. Applying MEKF$_\lambda$ on the above dynamic equations, we obtain the following equations to update the estimate of the model parameter:

$$\hat{\theta}_t = \hat{\theta}_{t-1} + \mathbf{K}_t \cdot (\mathbf{Y}_t - \hat{\mathbf{Y}}_t) \tag{26}$$

$$\mathbf{K}_t = \mathbf{P}_{t-1} \cdot \mathbf{H}_t^T \cdot (\mathbf{H}_t \cdot \mathbf{P}_{t-1} \cdot \mathbf{H}_t^T + \mathbf{R}_t)^{-1} \tag{27}$$

$$\mathbf{P}_t = \lambda^{-1}(\mathbf{P}_t - \mathbf{K}_t \cdot \mathbf{H}_t \cdot \mathbf{P}_{t-1} + \mathbf{Q}_t) \tag{28}$$

where $\mathbf{K}_t$ is the Kalman gain. $\mathbf{P}_t$ is a matrix representing the uncertainty in the estimates of the parameter θ of the model; λ is the forgetting factor to discount old measurements; $\mathbf{H}_t$ is the gradient matrix by linearizing the network:

$$\mathbf{H}_t = \frac{\partial f_{EDN}(\hat{\theta}_{t-1}, \mathbf{S}_{t-1})}{\partial \hat{\theta}_{t-1}} = \frac{\partial \hat{\mathbf{Y}}_{t-1}}{\partial \hat{\theta}_{t-1}} \tag{29}$$

In implementation, we need to specify initial conditions θ_0 and $\mathbf{P}_0$. θ_0 is initialized as the offline trained model parameter. For $\mathbf{P}_0$, due to absence of prior knowledge on the initial model parameter uncertainty, we simply set it as an identity matrix $\mathbf{P}_0 = p_i\mathbf{I}$ with $p_i > 0$. The whole process of the online adaptation is summarized in Algorithm 1, which enables us to adapt the parameter of any layer in the model. The performance of adapting different layers is briefly discussed in Sec 7.3 and analysed in detail in Appendix E, where we empirically found that in the tested data distribution, the best performance goes when the last layer of the network is adapted.

7 Experiment

By detailed experiments on real data, we aim at answering the following key questions:

1. In the high-level intention identification policy (SGN), what features in the input and output should be considered? What graph network architecture works the best?

2. In the low-level behavior prediction policy (EDN), what coordinates and features in the input and output work the best? Whether commonly used representation tricks and mechanisms in the encoder decoder architecture would improve performance?

3. How to integrate the intention signal into the encoder decoder? How much can the intention signal improve the prediction accuracy?

4. How to systematically evaluate the performance of online adaptation (OA)? How many steps of observation are the best to adapt? What is the best layer in the network to adapt?

5. How does the whole proposed method perform compared to other methods in terms of prediction accuracy, transferability and adaptability?

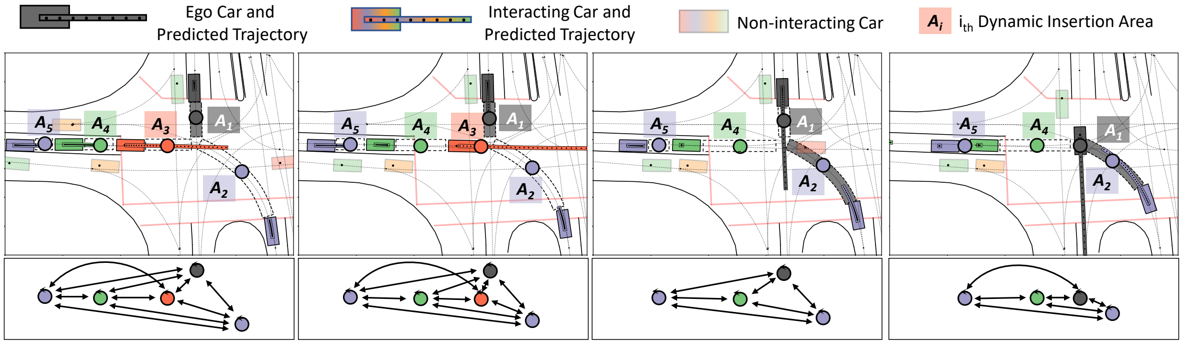
In this section, we first provide a case study illustrating how our method works. We then provide a brief summary of evaluations on each module (SGN, EDN, OA) addressing the first 4 questions, and a comparison with other methods for the 5-th question. For comprehensive evaluation on each module to answer the first 4 questions in detail, we refer interested readers to the Appendix B C D E respectively.

7.1 Experiment setting

We verified our proposed method with real human driving data from the INTERACTION dataset (Zhan et al., 2019). Two different scenarios were utilized: a 5-way unsignalized intersection (Figure 6) and an 8-way roundabout (Figure 7). Road reference paths and traffic regulations were extracted from the provided high-definition map. The intersection scenario was used to train our policy and evaluate the performance of behavior prediction. The roundabout scenario was used to evaluate the transferability of our method. In the intersection scenario, we had 19084 data points, which were split to 80% for training data and 20 % for testing data. In the roundabout scenario, there were 9711 data points to evaluate the transferability. Adaptability is evaluated on both scenarios.

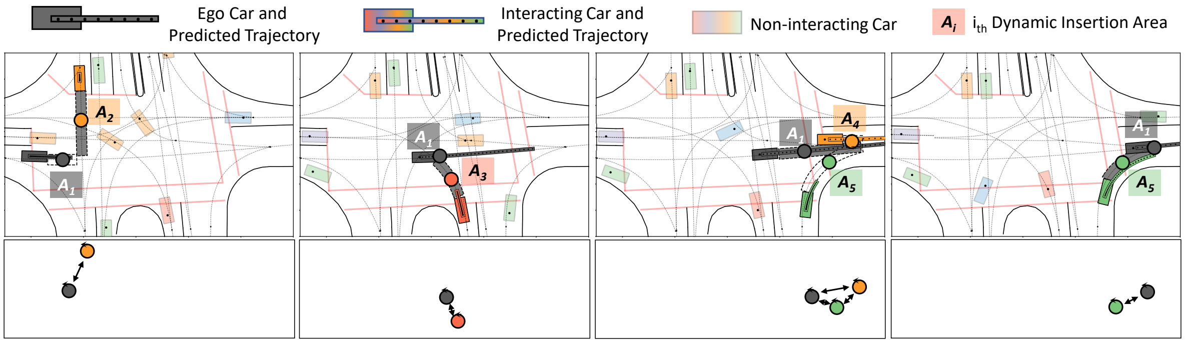
In our experiments, we choose the historic time steps T_h as 10 and future time step T_f as 30, which means we utilized historic information in the past 1 second to generate future behavior in the next 3 seconds. In addition to the long-term behavior prediction evaluation in the whole future 30 steps, we also evaluated short-term behavior prediction in the future 3 steps, as short-term behavior is safety-critical especially in close-distance interactions.

The method was implemented in Pytorch on a desktop computer with an Intel Core i7 9th Gen CPU and a NVIDIA RTX 2060 GPU. For each model, we performed optimizations with Adam and sweep over more than 20 combinations of hyperparameters to select the best one including batch size, hidden dimension, learning rate, dropout rate, etc.


Ego Car and Predicted Trajectory
Interacting Car and Predicted Trajectory
Non-interacting Car
A_i i_th Dynamic Insertion Area
A_1
A_2
A_3
A_4
A_5

(a) (b) (c) (d)

Figure 5: Case 1 - an illustration of how our method works during one interaction. Our methods can predict which DIA ego car will insert into, and the future trajectories of ego car and its interacting cars. Black car denotes the ego car; transparent-color cars denote non-interacting cars; bright-color cars denote vehicles interacting with ego car, based on which the DIAs are extracted. Each DIA is marked with dashed lines and one node. The same color is used for one DIA's node, notation, and rear-bound vehicle. The graphical relationships at one scene is displayed underneath each scene figure. The darker the DIA is, the more likely the ego car is going to insert into that area. The predicted most likely future trajectory of the ego vehicle and its interacting vehicles are displayed in each vehicle's color. In this case, DIA A_1 denotes the slot between ego car and the conflict point; DIA A_2 denotes the slot between purple car and the conflict point; DIA A_3 denotes the slot between orange car and the conflict point; DIA A_4 denotes the slot between green car and orange car; DIA A_5 denotes the slot between purple car and green car. During this interaction, the ego car first yielded the red car in (a)(b), and then passed before other cars (c)(d).

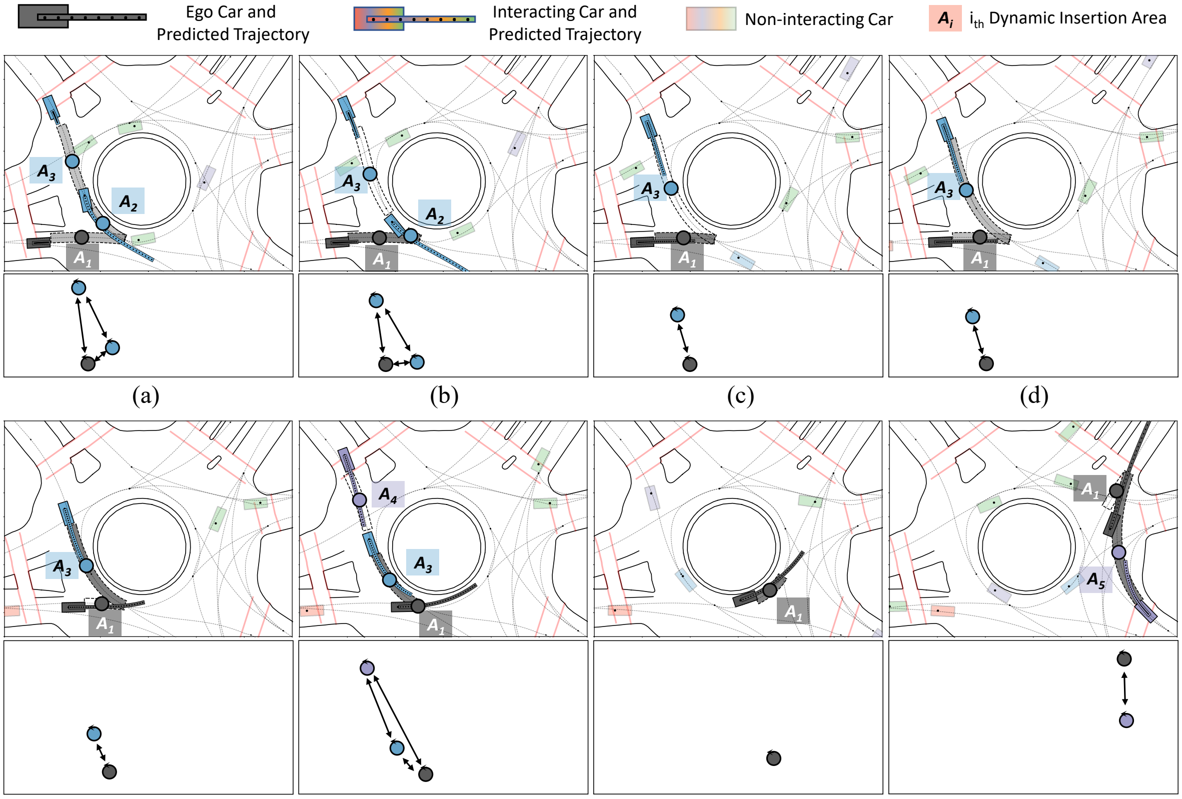

Ego Car and Predicted Trajectory
Interacting Car and Predicted Trajectory
Non-interacting Car
A_i i_th Dynamic Insertion Area
A_1
A_2
A_3
A_4
A_5

(a) (b) (c) (d)

Figure 6: Case 2 - an illustration of how our method works during a sequence of interactions. When ego car crosses the whole scene, our method can constantly identify interactions and extract interacting cars. The ego car first interacted with the yellow car in (a), then with the orange car in (b), then the green and orange car in (c)(d)

7.2 Case study

We first illustrate how our method works with three examples in Figure 5-7: 1) how ego vehicle interacted with other vehicles to pass a common conflict point (one interaction); 2) how the ego vehicle interacted with other vehicles to pass a sequence of conflict points (a sequence of interaction); 3) how ego vehicle interacted with other vehicles when it is zero-shot transferred to the roundabout scenario without retraining (scenario-transferable interactions).


Ego Car and Predicted Trajectory
Interacting Car and Predicted Trajectory
Non-interacting Car
A_i i_{th} Dynamic Insertion Area
A_1
A_2
A_3
A_4
A_5

(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 7: Case 3 - an illustration of how our method is directly transferred to the roundabout scenario without learning a new set of parameters, after it is trained in the intersection scenario. The ego car first yielded the first blue car in (a)(b), then passed before the second blue car (c)(d)(e)(f). Note that humans' hesitating and intention switch in this period was captured by our method, where the ego vehicle first decided to yield the blue car in (c), then changed it mind in (d), and finally passed before the blue car in (e)(f). The ego car then continued to run in (g), and left the roundabout before the purple car in (h).

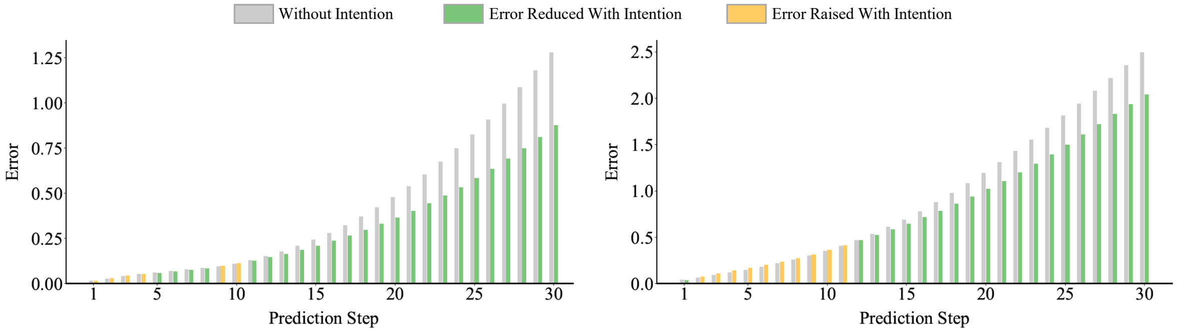
7.2.1 Case 1: one interaction

As in Figure 5, we first show how our method works in one interaction. Once we chose the ego vehicle, we extracted cars whose reference lines conflict with the ego car's reference line. These cars were regarded as the interacting vehicles and corresponding DIAs were extracted. Our method would then predict the intention and future trajectory of the ego vehicle and its interacting vehicles. In the figure, we drew the ego vehicle with black color, the interacting vehicles with bright colors, and the non-interacting vehicles with transparent colors. The darker a DIA is, the more likely the ego vehicle would insert into that DIA. The predicted future trajectories of the ego vehicle and the interacting vehicles are also displayed with the corresponding color.

As in Figure 5(a)(b), the black ego car initially had 5 areas to choose to insert into: A_1, A_2, A_3, A_4, A_5. The ego vehicle braked so our method predicted that it would insert into its front DIA A_1, which means yielding to other vehicles. In Figure 5(c), the orange vehicle behind DIA A_3 ran away and the ego vehicle accelerated, so our method predicted ego vehicle would insert into DIA A_2, which means passing before other vehicles. In Figure 5 (d) the ego vehicle crossed the conflict point and finished this interaction.

7.2.2 Case 2: A sequence of interactions

We illustrate how one vehicle crossed the intersection with a sequence of interactions in Figure 6. Specifically, the ego vehicle initially interacted with the upper yellow vehicle as in Figure 6(a). As the black ego vehicle was running at a high speed, our method predicted it would insert into DIA A_2 and pass before the red



(a) On Intersection Scenario (b) On Roundabout Scenario

Figure 8: Prediction error (m) of each step. The prediction error grew exponentially as horizon extends. The intention signal effectively suppressed the error growth, especially in the long term.

Table 2: A brief summary of ablation studies conducted to develop our method. We incrementally investigated: the effect of the feature and representation in EDN; the benefit of intention signal to prediction; the performance of online adaptation. We evaluated the performance by calculating ADE (m) and FDE (m) between predicted trajectory and ground-truth trajectory under different horizons and scenarios.

				EDN Exploration		Intention Integration		Online Adaptation
Scenario	Metric	Horizon	Baseline	Feature	Representation	Goal State Signal	Time Signal	Ours
Intersection	ADE	3s	0.884 ± 0.594	0.629 ± 0.397	0.407 ± 0.328	0.302 ± 0.251	0.305 ± 0.253	**0.301 ± 0.250**
		0.3s	0.409 ± 0.245	0.319 ± 0.224	0.027 ± 0.020	**0.021 ± 0.017**	0.029 ± 0.020	0.023 ± 0.014
	FDE	3s	1.850 ± 1.684	1.416 ± 1.175	1.279 ± 1.130	0.890 ± 0.831	**0.876 ± 0.835**	0.877 ± 0.830
		0.3s	0.423 ± 0.245	0.324 ± 0.229	0.040 ± 0.034	0.036 ± 0.025	0.043 ± 0.032	**0.032 ± 0.024**
Roundabout (Transfer)	ADE	3s	2.924 ± 4.695	2.201 ± 3.999	0.941 ± 0.778	0.845 ± 0.564	0.815 ± 0.526	**0.815 ± 0.526**
		0.3s	1.572 ± 4.029	1.543 ± 3.957	0.062 ± 0.071	0.060 ± 0.060	0.073 ± 0.065	**0.052 ± 0.068**
	FDE	3s	5.446 ± 7.157	4.123 ± 5.227	2.494 ± 2.070	2.081 ± 1.440	2.038 ± 1.409	**2.041 ± 1.409**
		0.3s	1.546 ± 3.894	1.544 ± 3.944	0.088 ± 0.104	0.091 ± 0.101	0.108 ± 0.107	**0.079 ± 0.114**

vehicle. After finishing the first interaction, the ego vehicle then interacted with the orange vehicle below as in Figure 6(b). Our method predicted the ego vehicle would continue to pass and insert into the DIA A_3. Later in Figure 6(c), the ego vehicle's reference path conflicted with that of the green and orange car. The ego vehicle first decelerated and our method predicted it would insert into its front DIA A_1 and yield other cars. In Figure 6(d), after the yellow car ran away, the ego vehicle then accelerated and inserted into DIA A_5 to pass ahead of the green car.

7.2.3 Case 3: Scenario-transferable interactions

In Figure 7, we show that after our policy was trained in the intersection scenario, it can be zero-shot transferred to the roundabout scenario. In Figure 7(a), the ego vehicle just entered the roundabout with some speed, so it was predicted to be equally likely to insert into the three DIAs A_1, A_2, A_3. In Figure 7(b), the ego vehicle decelerated, so it was predicted to yield other vehicles and insert into its front DIA A_1. In Figure 7(c), the first blue car moved away but the ego vehicle still remained at a low speed, leading to the prediction that the ego car would continue to yield other vehicles. But later we witnessed a change of plan. In Figure 7(d), the ego vehicle accelerated so it became equally likely to insert into either DIA A_1 or DIA A_3. In Figure 7(e) the ego vehicle gained a high speed, and it was predicted to pass before the blue car by inserting into DIA A_3. After finishing the first interaction in Figure 7(f), the ego vehicle continued to run while there were no other interacting vehicles as in Figure 7(g). In Figure 7(h), one purple car entered the roundabout and the ego vehicle decided to pass before it by inserting into DIA A_5.

7.3 A summary of ablation study results

AS shown in Table 2, we briefly summarize all ablation studies conducted to evaluate each module (SGN, EDN, OA) as a quick takeaway for the reader. A detailed version of these ablation studies can be found in Appendix B C D E

Starting from a baseline EDN which takes in the historic position and predicts future trajectory, we first explored the effect of features and representations in EDN. By adding features such as speed and yaw, we reduced the ADE in 3 seconds by 28% and 24% in the two scenarios. In the representation aspect, we conducted incremental prediction and position alignment. Such design not only effectively reduced the ADE in 3 seconds by 35% and 57% in the two scenarios, but also significantly reduced the ADE in 0.3 seconds by 91% and 95%. We can consequently conclude that the information of speed and yaw, along with the incremental prediction and position alignment, are vital for the prediction performance in encoder decoder architecture.

Next, we integrated intention signals into the EDN. According to the Table 2, integrating the goal state signal significantly reduced the ADE in 3 seconds by 25% and 10% in the two scenarios. The time signal barely benefitted the prediction in the intersection scenario, but it reduced the ADE in 3 seconds of the roundabout scenario by 3.5%. In Figure 8, we also displayed the error by step in the future 30 prediction steps. We can clearly see that as the prediction horizon extended, the prediction error grew exponentially. But the intention signal effectively suppressed the error growth, especially in the long horizon. Such results effectively demonstrated the necessity of intention integration.

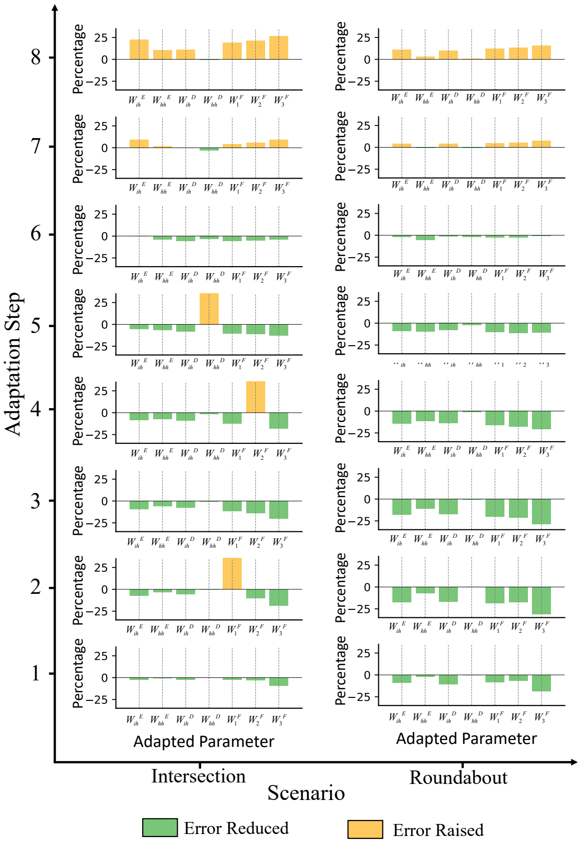


Figure 9: Percentage of short-term prediction error (ADE 2) reduced or raised after online adaptation, under 1) different adaptation step τ; 2) different layer of parameters adapted; 2) different scenarios. Three conclusions from the results: 1) online adaptation works the best around the adaptation step of 2 or 3; 2) prediction accuracy was improved by a higher percentage in the roundabout scenario (transferred) than in the intersection scenario (trained); 3) the best adaptation performance is obtained usually by adapting W_3^F, the last layer of the FC network in the decoder.

Finally, the online adaptation was implemented. As shown in Table 2, the online adaptation barely improved the long-term prediction in the next 3 seconds, but the short-term prediction in 0.3 seconds was improved by 20% and 28% in the intersection and roundabout scenarios. As a result, a conclusion is that though the online adaptation may not be able to help with long-term prediction, it can help to refine short-term prediction. Such improvement is valuable especially in close-distance prediction, where a small prediction shift can make a big difference in terms of safety. Besides, as in Figure 9, we show the percentage of short-term prediction error (ADE 2) reduced or raised after adaptation, under different adaptation step τ, different adpated parameters, and different scenarios. We have several observations: 1) as the adaptation step τ increased, the percentage of error reduction increased and reached the peak at 2 or 3 steps. But after that the help of adaptation decayed. After 7 steps, the predictions became even worse due to a big behavior gap; 2) Intuitively, the adaptation worked better in the roundabout scenario, compared to the intersection scenario, as the model was trained on the intersection scenario and directly transferred to the roundabout scenario. 3) The best adaptation performance is usually achieved by adapting the layer W_3^F, which is the last layer of the FC network in the decoder.

Table 3: Performance comparison with other methods. We evaluated the performance by calculating ADE (m) and FDE (m) between predicted trajectory and ground-truth trajectory in different horizons and scenarios.

			Rule-Based Method			Learning-Based Method					
Scenario	Metric	Horizon	IDM	FSM-D	FSM-T	V-LSTM	S-LSTM	S-GAN	Grip++	Trajectron++	HATN (Ours)
Intersection	ADE	3s	2.847 ± 1.963	3.181 ± 2.403	3.372 ± 2.495	1.315 ± 1.177	1.277 ± 1.295	1.372 ± 1.221	0.949 ± 0.670	0.510 ± 0.440	**0.301 ± 0.250**
		0.3s	0.042 ± 0.014	0.051 ± 0.034	0.054 ± 0.034	0.073 ± 0.006	0.047 ± 0.003	0.073 ± 0.004	0.024 ± 0.002	**0.009 ± 0.008**	0.023 ± 0.014
	FDE	3s	7.655 ± 5.536	8.709 ± 7.081	9.011 ± 7.192	3.386 ± 2.229	3.160 ± 2.387	3.469 ± 2.265	2.646 ± 5.773	1.617 ± 1.517	**0.877 ± 0.830**
		0.3s	0.091 ± 0.033	0.103 ± 0.059	0.111 ± 0.059	0.140 ± 0.003	0.112 ± 0.001	0.153 ± 0.002	0.037 ± 0.007	**0.014 ± 0.013**	0.032 ± 0.024
Roundabout (Transfer)	ADE	3s	5.271 ± 1.950	4.637 ± 1.448	4.824 ± 1.509	2.202 ± 4.295	2.459 ± 4.675	2.273 ± 4.448	1.543 ± 1.021	1.250 ± 0.849	**0.815 ± 0.526**
		0.3s	0.126 ± 0.062	0.093 ± 0.070	0.101 ± 0.004	0.061 ± 0.001	0.099 ± 0.002	0.090 ± 0.003	0.034 ± 0.004	**0.015 ± 0.011**	0.052 ± 0.068
	FDE	3s	13.891 ± 5.845	13.133 ± 4.208	13.505 ± 4.407	6.136 ± 8.445	6.668 ± 9.081	6.354 ± 9.499	4.352 ± 8.420	4.063 ± 2.706	**2.041 ± 1.409**
		0.3s	0.206 ± 0.096	0.157 ± 0.105	0.162 ± 0.102	0.189 ± 0.008	0.162 ± 0.003	0.211 ± 0.006	0.055 ± 0.001	**0.025 ± 0.020**	0.079 ± 0.114

7.4 Comparison with other methods

In this section, as in Table 3, we compared our method with other methods in terms of behavior prediction accuracy, transferability, and adaptability in different horizons and scenarios.

We first considered three rule-based methods. The IDM (Treiber et al., 2000) method basically follows its front car on the same reference path. The FSM-based (Zhang et al., 2017) method additionally considers cars on the other reference paths that have conflicts with the ego car, and follows the closest front car using the IDM model. When deciding the closest front car, the FSM-D method calculates each car's distance to the conflict point and chooses the closest one. The FSM-T method first calculates each vehicle's time needed to reach the conflict point by assuming they are running at a constant speed, and then chooses the closest one. We set the parameter of the IDM model as the values identified in urban driving situations (Liebner et al., 2012). As shown in Table 3, the rule-based method did not work well in the prediction task. Several reasons may be possible. First is that intersections and roundabouts are really complicated with intense multi-agent interactions. Simple rules can hardly capture such complex behaviors while systematic rules are hard to manually design. A second reason is that, the parameter in the driving model is hard to specify as it is also scenario-and-individual-specific.

We later evaluated several learning-based methods. Vallina LSTM (V-LSTM) adopts encoder decoder architecture with LSTM cells, which processes historic trajectories and generate future trajectory for each agent. Note that the representation tricks of position alighment and incremental prediction is also applied. To consider interaction between agents, Social LSTM (S-LSTM) (Alahi et al., 2016) additionally pools nearby agents' hidden states at every step using social pooling operation. Social-GAN (Gupta et al., 2018) modelled each agent as a LSTM-GAN, where a generator generates trajectory, which is then evaluated against the ground-truth trajectory by a discriminator. As shown in Table 3, both the three methods achieved much higher accuracy than traditional methods, distinguishing the power of deep learning. Though equipped with social pooling, S-LSTM only performed closely to V-LSTM, similar to experiment in Gupta et al. (2018); Salzmann et al. (2020). S-GAN suffered from unstable convergence during training and achieved slightly worse performance. In practice, the long running time caused by pooling operations render these methods intractable in real-time deployment. In comparison, benefit by GNN operation, our method achieved real-time computation and scaled pretty well with the agent number as shown in Appendix G.

We then evaluated other graph-based learning methods. Grip++ (Li et al., 2019) represents the interactions of close agents with a graph, applies graph convolution operations to extract spatial and temporal features, and subsequently uses an encoder-decoder LSTM model to make future predictions. We also implemented Trajectron++ (Salzmann et al., 2020), a GNN-based method. Trajectron++ takes vehicles as nodes of a graph and utilizes a graph neural network to conduct relationship reasoning. The map information is integrated by embedding the image of the map. A generative model is then used to predict future actions. The future trajectory is then generated by propagating the vehicle dynamics with the predicted actions. According to Table 3, we found these methods effectively surpassed previous three learning-based methods, benefit from the representation and relational inductive bias of graph. Among these methods, Trajectron++ and our method performed much better than Grip++, which could be showing the better reasoning capability of graph operations compared to convolution operations. Among the two best methods, our method significantly outperformed Trajectron++, with ADE in 3 seconds lower by 41% and 34% in the intersection and roundabout scenarios respectively. Such results demonstrated that our method's great capability in the long-term prediction, benefiting from our design of semantic hierarchy and transferable representation. Nevertheless, Trajectron++ performed better than ours in the short-term prediction. One important reason is that the predicted trajectory of Trajectron++ is strictly dynamics-feasible, as it essentially predicts future actions and propagates them through the dynamics to retrieve future trajectories. Such results motivate us to include dynamic constrain in our future work. Besides, our method also differs from the two methods in the criteria to selecting interacting vehicles, which is analyzed in detail in Appendix F.

8 Conclusion

In this paper, inspired by humans' cognition model and semantic understanding during driving, we proposed a hierarchical framework to generate high-quality driving behavior prediction in a multi-agent dense-traffic environment. The proposed method consists of: 1) constructing semantic graph as a generic representation for the environment, which is transferable across different scenarios; 2) a semantic graph network for high-level intention prediction; 3) an encoder decoder network for low-level trajectory prediction; 4) an online adaptation module to adapt to different individuals and scenarios. The proposed method hierarchically divided the driving task into two sub-tasks with profound semantics, which simplifies the learning and provides more interpretability. Due to the generic representation for each hierarchy/sub-task, our method can be directly transferred to new scenarios after it is trained in one scenario. The online adaptation module can also adapt our method to different individual and scenarios for high-fidelity predictions.

In the experiments on real human data, we empirically investigated 1) what features and graph network architecture should be utilized in the high-level intention prediction; 2) whether commonly used features, representation tricks, and mechanisms would benefit low-level trajectory prediction; 3) how to integrate intention signal into the low-level trajectory prediction policy; 4) systematically evaluation of the online adaptation with a new set of metrics, including how much the prediction accuracy can be improved, what is the best step of observation and layer to adapt; 5) how our method outperforms other methods.

In the future, we emphasize that the transferability and adaptability of prediction and planning algorithms are critical and worth more research efforts for the general and wide deployment of autonomous vehicles on the roads. It is also interesting to explore the interaction between high-level policy and low-level policy to accommodate for each other, simultaneous online adaptation for intention and action.

References

Abulikemu Abuduweili and Changliu Liu. Robust online model adaptation by extended kalman filter with exponential moving average and dynamic multi-epoch strategy. In *Learning for Dynamics and Control*, pp. 65–74. PMLR, 2020.

Abulikemu Abuduweili and Changliu Liu. Robust nonlinear adaptation algorithms for multitask prediction networks. *International Journal of Adaptive Control and Signal Processing*, 35(3):314–341, 2021.

Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 961–971, 2016.

Georges S Aoude, Brandon D Luders, Kenneth KH Lee, Daniel S Levine, and Jonathan P How. Threat assessment design for driver assistance system at intersections. In *13th international ieee conference on intelligent transportation systems*, pp. 1855–1862. IEEE, 2010.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

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

Chris L Baker, Joshua B Tenenbaum, and Rebecca Saxe. Bayesian models of human action understanding. *Advances in neural information processing systems*, 18:99, 2006.

Chris L Baker, Joshua B Tenenbaum, and Rebecca R Saxe. Goal inference as inverse planning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 29, pp. 29, 2007.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pp. 359–370. Seattle, WA, USA:, 1994.

Shubhendu Bhasin, Rushikesh Kamalapurkar, Huyen T Dinh, and Warren E Dixon. Robust identification-based state derivative estimation for nonlinear systems. *IEEE Transactions on Automatic Control*, 58(1): 187–192, 2012.

Matthew M Botvinick, Yael Niv, and Andew G Barto. Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. *Cognition*, 113(3):262–280, 2009.

Yujiao Cheng, Weiye Zhao, Changliu Liu, and Masayoshi Tomizuka. Human motion prediction using semi-adaptable neural networks. In *2019 American Control Conference (ACC)*, pp. 4884–4890. IEEE, 2019.

Yujiao Cheng, Liting Sun, Changliu Liu, and Masayoshi Tomizuka. Towards efficient human-robot collaboration with robust plan recognition and trajectory prediction. *IEEE Robotics and Automation Letters*, 5 (2):2602–2609, 2020.

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.

Chiho Choi, Srikanth Malla, Abhishek Patil, and Joon Hee Choi. Drogon: A trajectory prediction model based on intention-conditioned behavior reasoning. *arXiv preprint arXiv:1908.00024*, 2019.

Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4693–4700. IEEE, 2018.

P Dayan and GE Hinton. Feudal reinforcement learning., 1993.

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.

Julie Dequaire, Peter Ondrúška, Dushyant Rao, Dominic Wang, and Ingmar Posner. Deep tracking in the wild: End-to-end tracking using recurrent neural networks. *The International Journal of Robotics Research*, 37(4-5):492–512, 2018.

Wenchao Ding, Jing Chen, and Shaojie Shen. Predicting vehicle behaviors over an extended horizon using behavior interaction network. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8634–8640. IEEE, 2019.

Ashraf Elnagar. Prediction of moving objects in dynamic environments using kalman filters. In *Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Cat. No. 01EX515)*, pp. 414–419. IEEE, 2001.

Yannis Flet-Berliac. The promise of hierarchical reinforcement learning. *The Gradient*, 2019.

David Fridovich-Keil, Andrea Bajcsy, Jaime F Fisac, Sylvia L Herbert, Steven Wang, Anca D Dragan, and Claire J Tomlin. Confidence-aware motion prediction for real-time collision avoidance1. *The International Journal of Robotics Research*, 39(2-3):250–265, 2020.

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*, pp. 11525–11533, 2020.

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*, pp. 2255–2264, 2018.

Leonard Hasenclever, Fabio Pardo, Raia Hadsell, Nicolas Heess, and Josh Merel. Comic: Complementary task learning & mimicry for reusable skills. In *International Conference on Machine Learning*, pp. 4105–4115. PMLR, 2020.

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

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

Yeping Hu, Wei Zhan, and Masayoshi Tomizuka. Scenario-transferable semantic graph reasoning for interaction-aware probabilistic prediction. *arXiv preprint arXiv:2004.03053*, 2020.

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137, 2015.

Xin Li, Xiaowen Ying, and Mooi Choo Chuah. Grip++: Enhanced graph-based interaction-aware trajectory prediction for autonomous driving. *arXiv preprint arXiv:1907.07792*, 2019.

Zirui Li, Chao Lu, Yangtian Yi, and Jianwei Gong. A hierarchical framework for interactive behaviour prediction of heterogeneous traffic participants based on graph neural network. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

Martin Liebner, Michael Baumann, Felix Klanner, and Christoph Stiller. Driver intent inference at urban intersections using the intelligent driver model. In *2012 IEEE Intelligent Vehicles Symposium*, pp. 1162–1167. IEEE, 2012.

Changliu Liu and Masayoshi Tomizuka. Enabling safe freeway driving for automated vehicles. In *Proceedings of the American Control Conference (ACC)*, pp. 3461 – 3467. IEEE, 2016.

Ruixuan Liu and Changliu Liu. Human motion prediction using adaptable recurrent neural networks and inverse kinematics. *IEEE Control Systems Letters*, 5(5):1651–1656, 2021. doi: 10.1109/LCSYS.2020.3042609.

Lennart Ljung and Pierre Priouret. A result on the mean square error obtained using general tracking algorithms. *International journal of adaptive control and signal processing*, 5(4):231–248, 1991.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

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, pp. 6120–6127, 2019.

Emily C Marcinowski, Eliza Nelson, Julie M Campbell, and George F Michel. The development of object construction from infancy through toddlerhood. *Infancy*, 24(3):368–391, 2019.

Gustav Markkula, Ruth Madigan, Dimitris Nathanael, Evangelia Portouli, Yee Mun Lee, André Dietrich, Jac Billington, Anna Schieben, and Natasha Merat. Defining interactions: A conceptual framework for understanding interactive behaviour in human and automated road traffic. *Theoretical Issues in Ergonomics Science*, 21(6):728–752, 2020.

Josh Merel, Arun Ahuja, Vu Pham, Saran Tunyasuvunakool, Siqi Liu, Dhruva Tirumala, Nicolas Heess, and Greg Wayne. Hierarchical visuomotor control of humanoids. *arXiv preprint arXiv:1811.09656*, 2018.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

Graham Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*, 2017.

Yael Niv. Learning task-state representations. *Nature neuroscience*, 22(10):1544–1553, 2019.

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)*, pp. 1672–1678. IEEE, 2018.

Angela Radulescu, Yael Niv, and Ian Ballard. Holistic reinforcement learning: the role of structure and attention. *Trends in cognitive sciences*, 23(4):278–292, 2019.

Noha Radwan, Wolfram Burgard, and Abhinav Valada. Multimodal interaction-aware motion prediction for autonomous street crossing. *The International Journal of Robotics Research*, 39(13):1567–1598, 2020.

Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2821–2830, 2019.

Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: a survey. *The International Journal of Robotics Research*, 39(8): 895–935, 2020.

Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pp. 683–700. Springer, 2020.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pp. 4470–4479. PMLR, 2018.

Matthias Schreier, Volker Willert, and Jürgen Adamy. Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems. In *17th international ieee conference on intelligent transportation systems (ITSC)*, pp. 334–341. IEEE, 2014.

Wilko Schwarting, Alyssa Pierson, Javier Alonso-Mora, Sertac Karaman, and Daniela Rus. Social behavior for autonomous vehicles. *Proceedings of the National Academy of Sciences*, 116(50):24972–24978, 2019.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

Wenwen Si, Tianhao Wei, and Changliu Liu. Agen: Adaptable generative prediction networks for autonomous driving. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 281–286, 2019. doi: 10.1109/IVS.2019.8814238.

Elizabeth S Spelke and Katherine D Kinzler. Core knowledge. *Developmental science*, 10(1):89–96, 2007.

Zhaoen Su, Chao Wang, David Bradley, Carlos Vallespi-Gonzalez, Carl Wellington, and Nemanja Djuric. Convolutions for spatial interaction modeling, 2021.

Liting Sun, Wei Zhan, and Masayoshi Tomizuka. Probabilistic prediction of interactive driving behavior via hierarchical inverse reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2111–2117. IEEE, 2018a.

Liting Sun, Wei Zhan, Masayoshi Tomizuka, and Anca D Dragan. Courteous autonomous cars. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 663–670. IEEE, 2018b.

Charlie Tang and Russ R Salakhutdinov. Multiple futures prediction. *Advances in Neural Information Processing Systems*, 32:15424–15434, 2019.

Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

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

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3156–3164, 2015.

Letian Wang, Liting Sun, Masayoshi Tomizuka, and Wei Zhan. Socially-compatible behavior design of autonomous vehicles with verification on real human data. *IEEE Robotics and Automation Letters*, 6(2): 3421–3428, 2021.

Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

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]*, 2019.

Chengyuan Zhang, Jiacheng Zhu, Wenshuo Wang, and Junqiang Xi. Spatiotemporal learning of multivehicle interaction patterns in lane-change scenarios. *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2021. doi: 10.1109/TITS.2021.3057645.

Mengxuan Zhang, Nan Li, Anouck Girard, and Ilya Kolmanovsky. A finite state machine based automated driving controller and its stochastic optimization. In *Dynamic Systems and Control Conference*, volume 58288, pp. V002T07A002. American Society of Mechanical Engineers, 2017.

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.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3357–3364. IEEE, 2017.

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.

A Network architecture detail

The detailed design of the network architecture can be found in Tabel 4.

B Semantic graph network evaluation

In the high-level intention-identification task, we evaluated how well out method is able to predict future intentions by comparing the performance of our semantic graph network (SGN) with that of the other six approaches/variants. Three of them are set to explore the effect of different features and representations in the input and output. And the rest of them are the variants of the proposed network, which explored the effect of frequently used network architectures and tricks.

Table 4: Architecture detail for SGN and EDN.

SGN Architecture Detail	
f_{rec}^1	GRU cells
f_{rec}^2	GRU cells
f_{enc}^1	Dense layer with tanh activation function
f_{enc}^2	Dense layer with tanh activation function
f_{att}	Dense layer with leaky relu activation function
f_{enc}^3	Dense layer without activation function
f_{enc}^4	Dense layer with tanh activation function

EDN Architecture Detail	
Encoder	GRU cells
Decoder	GRU cell stacked with 3 dense layers, each with tanh activation function and dropout

1. No-Temporal: This method does not take historic information into account, namely only considering the information of the current time step t.
2. GAT: This method uses the absolute feature to calculate relationships among nodes instead of using the relative feature. This method corresponds to the original graph attention network (Veličković et al., 2017).
3. Single-Agent: This method only considers the loss of ego vehicle's intention prediction, and does not consider the intention prediction for other interactive vehicles.
4. Two-Layer-Graph: This method has a two-layer graph to conduct information embedding, namely exploits the graph aggregations twice (Sanchez-Gonzalez et al., 2018).
5. Multi-Head: This method employs the multi-head attention mechanism to stabilize learning (Veličković et al., 2017), namely operating the relationship reasoning multiple times in parallel independently, and concatenates all features as the final aggregated feature. In our case, we set the head number as 3.
6. Seq-Graph: This method first conducts relationship reasoning for the graph at each time step and then feeds the sequence of aggregated graphs into RNN for temporal processing. As a comparison, our method first embeds each node's sequence of historic features with RNN and then conduct relationship reasoning using each node's hidden state from RNN at the current time step.

The models were trained and tested on the intersection scenario. The trained models were also directly tested on the roundabout scenario to evaluate the zero-shot transferability. The inserted area prediction accuracy was evaluated by the multi-class classification accuracy. The performance of goal state prediction was evaluated by the absolute distance error (ADE) between the generated goal state and ground-truth state.

B.1 Inserted area prediction accuracy

On the Inserted area prediction accuracy shown in Table 5, we first provide some overview analysis. All models achieved close accuracy of around 90%, generally benefiting from the representation of the semantic graph. Another overall observation is that most models' transferability performance in the roundabout scenario surprisingly surpassed the performance on the intersection scenario on which the models were originally trained. This is because the intersection is a harder scenario than the roundabout, as the vehicles need to interact with many vehicles from different directions simultaneously when they are entering the intersection, while vehicles in the roundabout only need to interact with the cars from nearby branches.

We then show some detailed analysis. The GAT had the highest performance while our method followed as the second. The No-Temporal method had the lowest accuracy and largest variance in the intersection

Table 5: The statistical evaluation of the high-level intention prediction policy. We compared our method with the other six approaches/variants to explore the effect of different representations, features, and architectures. Note that all the models are trained in the intersection scenario and then directly transferred and tested on the roundabout scenario without further training. Our method outperformed all other methods in the goal state prediction (ADE).

		Representation/Feature Ablation Study			Architecture Ablation Study			
Scenario	Measure	No-Temporal	GAT	Single-Agent	Two-Layer-Graph	Multi-Head	Seq-Graph	Ours
Intersection	Acc (%)	87.44 ± 33.13	**91.93 ± 27.05**	88.8 ± 31.53	90.15 ± 29.79	90.00 ± 30.00	89.8 ± 30.24	90.50 ± 28.70
	ADE (m)	1.59 ± 1.67	1.18 ± 1.51	1.04 ± 0.90	0.98 ± 0.93	0.97 ± 0.75	1.33 ± 1.91	**0.94 ± 0.73**
Roundabout (Transfer)	Acc (%)	**93.92 ± 23.88**	91.21 ± 28.12	92.20 ± 26.75	90.54 ± 29.26	92.10 ± 26.96	91.60 ± 27.62	90.70 ± 29.08
	ADE (m)	3.62 ± 6.72	2.70 ± 5.16	1.88 ± 2.48	1.87 ± 2.51	2.79 ± 20.78	3.10 ± 3.10	**1.70 ± 1.99**

scenario (87.44±22.13%). This is because it lacks temporal information, which could otherwise efficiently help to identify which DIA to insert into by considering historic speed and acceleration. What is interesting is that the No-Temporal method contrarily achieved the highest insertion accuracy (93.92±23.88) in the roundabout scenario. One possible explanation is that the absence of temporal information constrained the model's capability and thus avoided over-fit, so the No-Temporal method has the best transferability performance. Such hypothesis also helps to explain why the Two-Layer-Graph method had the lowest insertion accuracy in the roundabout scenario (90.54±29.26%), as twice aggregations make the model brittle to over-specification.

B.2 Goal state prediction error

The goal state, on the one hand, is practically more important as it is directly delivered to low-level policy to guide the behavior generation process. On the other hand, it is much harder than the insertion identification task as it requires more delicate information extraction and inference. Consequently, we can see that the performance of different models varied a lot as shown in Table 5. Also, the models' performance significantly down-graded when they were directly transferred to the roundabout scenario, as the two scenarios have different geometries and different driving patterns such as speed and steering.

Specifically, we have several observations: 1) our method achieved the lowest error in both intersection and roundabout scenarios; 2) the No-Temporal method was the worst in both intersection and roundabout scenarios, due to the lack of temporal information; 3) the GAT method generated much higher errors than our method especially in the roundabout scenario (58%), which shows the necessity of using the relative features in relationship reasoning. 4) Our method outperformed the Single-Agent method, which implies the advantages of data augmentation and encouraging interaction inference by taking all vehicles' generated goal state into the loss function. 5) The Two-Layer-Graph method was the closest one to our method, though it came with serious over-fitting, for which we conducted more hyperparameter tuning to find a proper dropout value. 6) The Multi-Head method achieved the second-best accuracy in the intersection scenario but much worse performance in the roundabout scenario, which could be possibly improved by more delicate searching for a proper head number. 7) The Seq-Graph method was the second-worst in both the intersection and the roundabout scenario, which may imply that the complex encoding for past interactions could hardly help prediction but indeed makes the learning harder.

With the results above, a summary of conclusions on the intention prediction policy is that it is necessary to consider temporal information, use the relative feature for relationship reasoning, and predict the intention of all agents in the scene. On the contrary, the architectures like Two-Layer-Graph, Multi-Head mechanism, and Seq-Graph do not help here.

C Encoder decoder network evaluation

There are many existing works exploiting the encoder decoder architecture for the driving behavior generation (Park et al., 2018; Tang & Salakhutdinov, 2019; Zyner et al., 2019), but several questions still remain unclear: what coordinate should be employed? what features should be considered? what representation

Table 6: The statistical evaluation of the low-level behavior generation policy. We conducted experiments incrementally to explore the performance under different coordinates, features, representations, and mechanisms. The best performance is achieved when taking the Frenet coordinate, the speed and yaw feature in the input, and representation of incremental prediction and position alignment.

(a) Coordinate	Intersection		Roundabout (Transfer)	
	ADE	FDE	ADE	FDE
Cartesian	1.53 ± 1.22	2.90 ± 2.77	12.57 ± 5.68	19.77 ± 7.43
Frenet	**0.91 ± 0.59**	**1.87 ± 1.48**	**2.96 ± 4.65**	**5.52 ± 7.20**

(b) Speed Ablation		Intersection		Roundabout (Transfer)	
In	Out	ADE	FDE	ADE	FDE
×	×	0.91 ± 0.59	1.87 ± 1.48	2.96 ± 4.65	5.52 ± 7.20
✓	×	0.71 ± 0.54	1.53 ± 1.27	**2.33 ± 3.84**	**4.30 ± 5.44**
×	✓	0.78 ± 0.53	1.74 ± 1.41	2.51 ± 4.16	4.97 ± 6.35
✓	✓	**0.70 ± 0.49**	**1.46 ± 1.26**	2.40 ± 4.10	4.67 ± 6.03

(c) Yaw Ablation		Intersection		Roundabout (Transfer)	
In	Out	ADE	FDE	ADE	FDE
×	×	0.71 ± 0.54	1.53 ± 1.27	2.33 ± 3.84	4.30 ± 5.44
✓	×	**0.67 ± 0.46**	**1.45 ± 1.19**	**2.23 ± 3.95**	**4.14 ± 5.19**
×	✓	0.73 ± 0.50	1.59 ± 1.36	2.34 ± 3.96	4.49 ± 6.46
✓	✓	0.67 ± 0.48	1.46 ± 1.24	2.51 ± 4.60	4.77 ± 6.43

(d) Repre Ablation		Intersection		Roundabout (Transfer)	
Inc	Ali	ADE	FDE	ADE	FDE
×	×	0.67 ± 0.46	1.45 ± 1.19	2.23 ± 3.95	4.14 ± 5.19
×	✓	0.48 ± 0.44	1.32 ± 1.32	1.26 ± 0.95	3.04 ± 2.44
✓	×	0.43 ± 0.35	1.36 ± 1.19	1.07 ± 1.10	2.66 ± 2.31
✓	✓	**0.41 ± 0.33**	**1.29 ± 1.14**	**0.96 ± 0.80**	**2.53 ± 2.13**

(e) Mech Ablation		Intersection		Roundabout (Transfer)	
TF	Att	ADE	FDE	ADE	FDE
×	×	**0.41 ± 0.33**	**1.29 ± 1.14**	**0.96 ± 0.80**	**2.53 ± 2.13**
✓	×	0.41 ± 0.34	1.32 ± 1.16	0.97 ± 0.83	2.54 ± 2.14
×	✓	0.43 ± 0.34	1.32 ± 1.13	1.10 ± 1.32	2.57 ± 2.50

performs better? and whether commonly-used mechanisms in encoder decoder architecture can improve the performance in the driving task? To answer these questions, we conducted extensive ablation studies on the encoder decdoer network (EDN), where the intention signal is removed to avoid additional variance from the high-level policy. As shown in Table 6, starting from a naive encoder decoder that simply takes in position features and predicts positions, we incrementally added more tricks to test their effectiveness. Two metrics are set, absolute distance error (ADE) and final distance error (FDE).

C.1 Coordinate study

We first investigated which coordinate we should employ between Frenet and Cartesian coordinate. As shown in Table 6(a), the EDN with Frenet coordinate performed 40% (ADE) and 35% (FDE) better than the EDN with Cartesian coordinate in the intersection scenario. In the zero-transferred roundabout scenario, though the performance of both two methods downgraded, the performance of the method on Cartesian coordinate decayed more significantly, with ADE higher by 324% and FDE higher by 258% compared to the method on Frenet coordinate. This is because the Frenet coordinate implicitly incorporates the map information into the model. Compared to running in any direction in the Cartesian coordinate, in the Frenet coordinate the vehicles only need to follow the direction of references paths, which constrains its behavior in a more predictable pattern. Note that in the frenet coordinate, we use Dynamic Time Warping algorithm (Berndt & Clifford, 1994) to determine the most likely reference line that each agent lies on.

C.2 Feature study

Second, we explored the effect of features, specifically, the speed feature and yaw feature. For each feature, we consider two circumstances. The first is to incorporate the feature into the input of the encoder to provide more information. The second is to set the feature as additional desired outputs of the decoder, which could possibly help to stabilize the learning for position prediction. Thus for each feature, we explored 4 settings in terms of whether or not to add the feature into the input or output.

As in Table 6(b), incorporating speed feature in either the input or output could both effectively improve performance in the two scenarios. When incorporating it into the input and output simultaneously, compared to only considering it in the input, the performance was slightly improved in the intersection scenario and slightly degraded in the roundabout scenario. Concluding from the average performance in the two scenarios, we chose to only take the speed into the input of the encoder.

For the yaw feature, as shown in Table 6(c), taking it into input could slightly benefit the performance, while incorporating it into the output made the performance worse. One possible reason for such performance decay is that the yaw information has been already implicitly covered in the longitudinal and lateral speed information. Adding the yaw information into the output of the decoder could provide little additional information but indeed made the learning harder. We thus decided to incorporate the yaw feature only into the input of the encoder.

C.3 Representation study

There are two commonly-used representation techniques to shape the data distribution. The first technique is called incremental prediction (Li et al., 2019), which predicts the relative position compared to the position of the last step, rather than directly predicting the absolute position. The second technique is called position alignment, which aligns the positions of each step to the vehicle's current position (Park et al., 2018). According to Table 6(d), both two techniques could significantly improve the prediction accuracy, and applying both of them worked the best, improving the ADE by 38% in the intersection and by 56% in the roundabout.

C.4 Mechanism study

There are two frequently used mechanisms in the encoder decoder architecture: teacher forcing (Williams & Zipser, 1989) and attention mechanism (Bahdanau et al., 2014). Teacher forcing aims at facilitating the learning of complex tasks while the attention is designed to attend differently to different historic input. From the results in Table 6(c), we can see neither of the two mechanisms could benefit the performance. Considering that the encoder decoder is used as a dynamics approximator, which is a relatively simple task, the teacher forcing has pretty limited performance improvement since the EDN itself can already learn well enough. The attention mechanism indeed made the performance worse as the vehicle dynamics are most related to the recent state so previous states may not be necessarily informative.

As a summary of the ablation studies above, the best performance goes when taking the Frenet coordinate, the speed and yaw feature in the input, and representation of incremental prediction and position alignment. The teacher forcing and attention mechanism could not benefit the performance.

D Intention signal integration evaluation

As mentioned in Sec 5.2, we have two intention signals, namely the goal state and the decoding step. The goal state signal refers to the goal position in the future horizon. The decoding step refers to the which step the decoding cell lies in the whole deocding horizon. The two signals can be integrated into the low-level policy EDN in several ways, such as appending it into the input or output of the decoder (note as Input or Output), embedding it into the hidden state at the first step (note as Hidden). For the goal state intention signal, we can additionally choose to introduce it by transforming the origin state of the vehicle into the state relative to the goal state (note as Transform). In this section, we evaluate the performance when the different intention signals are incorporated in different ways, as shown in Tabel 7.

Table 7: The statistical evaluation of different methods to introduce intention signals (the goal state and the decoding step) into the low-level trajectory prediction policy. The best performance is achieved when both the two intention signals are appended into the input feature.

(a) Ground-truth Goal State Introduction

Scenario	Metric	No-Intention	With-Intention			
			Transform	Input	Output	Hidden
Intersection	ADE (m)	0.41 ± 0.11	**0.10 ± 0.10**	0.15 ± 0.13	0.17 ± 0.16	0.13 ± 0.14
	FDE (m)	1.29 ± 1.30	**0.15 ± 0.25**	0.29 ± 0.40	0.38 ± 0.41	0.26 ± 0.39
Roundabout (Transfer)	ADE (m)	0.96 ± 0.64	**0.42 ± 0.37**	0.51 ± 0.46	0.60 ± 0.54	0.48 ± 0.41
	FDE (m)	2.53 ± 4.54	**0.72 ± 0.66**	0.94 ± 0.73	1.03 ± 0.74	0.84 ± 0.67

(a) Predicted Goal State Introduction

Scenario	Metric	No-Intention	With-Intention			
			Transform	Input	Output	Hidden
Intersection	ADE (m)	0.41 ± 0.11	0.31 ± 0.25	**0.30 ± 0.25**	0.32 ± 0.25	0.31 ± 0.25
	FDE (m)	1.29 ± 1.30	0.92 ± 0.85	0.89 ± 0.83	0.89 ± 0.83	**0.89 ± 0.82**
Roundabout (Transfer)	ADE (m)	0.96 ± 0.64	0.89 ± 0.56	**0.86 ± 0.61**	0.92 ± 0.75	0.87 ± 0.54
	FDE (m)	2.53 ± 4.54	2.14 ± 1.44	**2.12 ± 1.51**	2.19 ± 1.69	2.22 ± 1.46

(a) Decoding step Introduction

Scenario	Metric	No-Intention	With-Intention			
			Transform	Input	Output	Hidden
Intersection	ADE (m)	0.41 ± 0.11	0.30 ± 0.25	**0.30 ± 0.25**	0.30 ± 0.25	0.30 ± 0.24
	FDE (m)	1.29 ± 1.30	0.89 ± 0.83	0.88 ± 0.83	0.89 ± 0.83	**0.88 ± 0.81**
Roundabout (Transfer)	ADE (m)	0.96 ± 0.64	0.86 ± 0.61	**0.82 ± 0.53**	0.87 ± 0.59	0.84 ± 0.53
	FDE (m)	2.53 ± 4.54	2.12 ± 1.51	**2.06 ± 1.43**	2.15 ± 1.51	2.11 ± 0.23

D.1 Integrating ground-truth goal state

First, we introduced the ground-truth goal state into the EDN to measure the most performance improvement we can get from the ground-truth intention. According to Table 7(a), the Transform method had the best performance and reduced the ADE by 75% and 56% in the two scenarios, which represents the most benefit we can get with ground-truth intention but is also impossible as there exist inevitable errors in the predicted goal state.

D.2 Integrating predicted goal state

When integrating the predicted goal state into EDN, the error in the goal state prediction would perturb the performance. As in Table 7(b), while the performance of these goal state integration methods was close, appending the goal state into the input feature list performed the best, reducing the ADE by 26% and 10% in the two scenarios.

D.3 Integrating decoding step

After introducing the goal state into the input feature, we further investigate the performance when decoding step signal is introduce as in Table 7(c). Similarly, appending it into the input performed the best, reducing the error in the roundabout scenario by 5%.

D.4 Visualizing the effect of intention signal

In Figure 8, we illustrated the effect of introducing the intention signal (the predicted goal state and the decoding step), by calculating the prediction error of each step in the future 30 steps. Obviously, as the prediction horizon extended, the prediction became more difficult and the error grew exponentially. After introducing the intention signal, the error growth was effectively suppressed, especially in the long horizon.

E Online adaptation evaluation

The online adaptation aims at capturing nuances in different individuals and scenarios by exploiting historic observations to subtly adjust model parameters. In this section, we address two questions by empirical evaluation 1) How many steps of observation are the best to adapt? 2) What is the best layer in the network to adapt? The new set of metrics mentioned in Sec 6 and shown in Figure 4 are used for the evaluation.

E.1 Trade-off in adaptation step

The adaptation step τ is an important parameter in the multi-step online adaptation algorithm. On the one hand, we can obtain more information by increasing τ. On the other hand, the behavior gap between the current time and historic time also increases. As a result, there is a performance trade-off when we increase the adaptation step τ. To empirically answer the question that how many steps are the best, We run the online adaptation on both the intersection and roundabout scenarios, and collected statistical results of the absolute distance error (ADE) between the ground-truth and the predicted trajectory. Note that here when evaluating one adaptation step τ, we adapted parameters in different layers and chose the best performance as the performance of that adaptation step τ.

Figure 10(a) shows the online adaptation results in the intersection scenario. According to ADE 1 and ADE 2 in the first two images, as the adaptation step τ increased, the prediction error of the first τ step on both the historic trajectory (ADE 1) and the current trajectory (ADE 2) increased. But the adaptation can depress such error growth. For ADE 1, the error was reduced by higher percentage when more steps τ of observation are used for adaptation, because more information was gained. For the ADE 2, as the adaptation steps τ increased, the error reduction percentage first increased and reached a peak of 20.5% at 3 adaptation steps. After that, the error reduction percentage decreased as the behavior gap had come into effect due to a longer time lag τ. The last two images show the error in the whole historic trajectory (ADE 3) and current trajectory (ADE 4). For the ADE 3, longer τ led to a higher percentage of error reduction due to more information gained. However, for the ADE 4 evaluating the performance of long-term prediction, due to the insufficient information and behavior gap, the improvement was limited. Similar results can be found in the roundabout scenario in Figure 10(b). But in the ADE 2, more improvement (28%) was achieved in the short-term prediction, due to the fact that the model was not trained on the roundabout scenario and there was more space for adaptation.

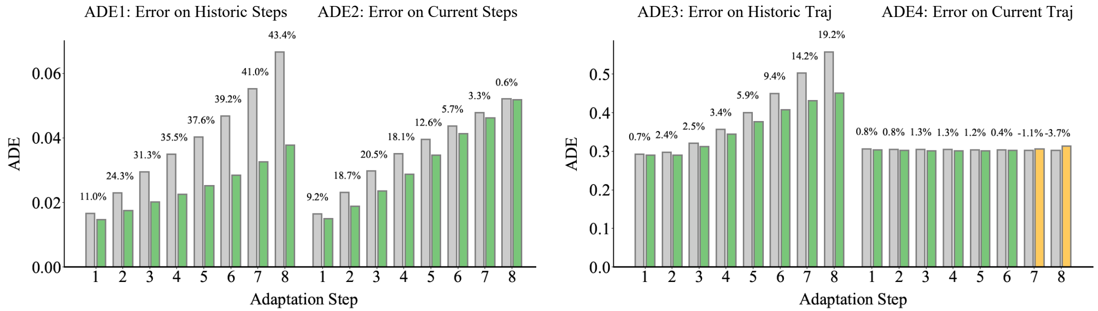
With these analyses, a conclusion is that though the adaptation does not help with the long-term behavior prediction in the next 3 seconds, the short-term behavior prediction in the next 0.3 seconds is effectively improved by 20.5% and 28.7% in the two scenarios. Such improvement in short-term prediction is valuable as it can effectively enhance safety in close-distance interactions.

E.2 Adaptation layer choice

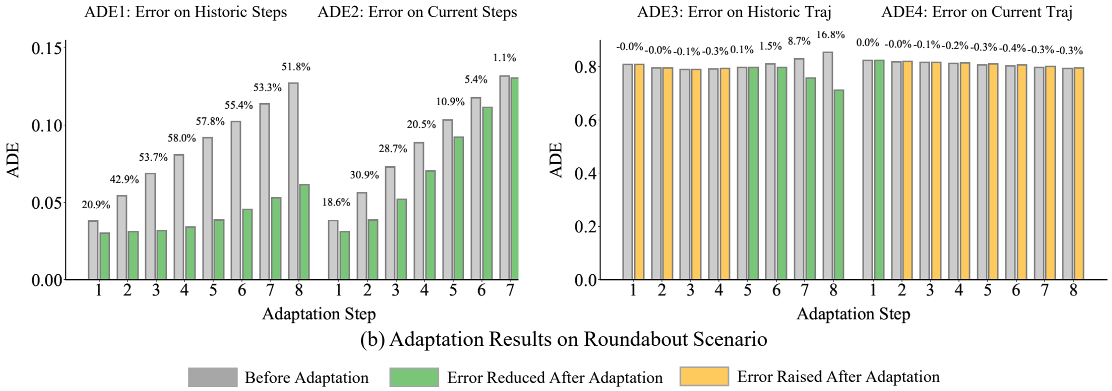
The neural network consists of many layers of parameters and it remains a question which layer shall we adapt in order to get the best adaptation performance. Thus in the section, we empirically analyze the performance of adapting different layers.

We first claim and denote all the layers. In the Encoder Decoder Network (EDN), both the encoder and decoder consist of single-layer gated recurrent units (GRU), and the decoder is additionally stacked with three layers of fully connected (FC) networks. We denote the encoder GRU's input-hidden weights as W_{ih}^E, the encoder GRU's hidden-hidden weights as W_{hh}^E, the decoder GRU's input-hidden weights as W_{ih}^D, the decoder GRU's hidden-hidden weights as W_{hh}^D, and the weights of the three-layer FC as W_1^F, W_2^F, W_3^F respectively.

As in Figure 9, we show the percentage of the change of ADE 2 after adaptation, under different adaptation step τ, different parameters, and different scenarios. We have several observations: 1) as the adaptation step τ increased, the percentage of error reduction increased and reached the peak at 2 or 3 steps. But after that the help of adaptation decayed, and after 7 steps, the predictions became even worse due to a big behavior gap; 2) Intuitively, the adaptation worked better in the roundabout scenario, compared to the intersection



(a) Adaptation Results on Intersection Scenario



(b) Adaptation Results on Roundabout Scenario

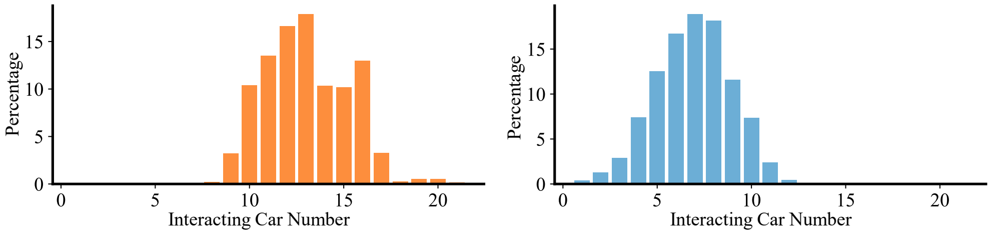
Figure 10: Online adaptation performance analysis. According to ADE 1 and ADE 3 in (a)(b), as adaptation step τ increased, the online adaptation could get more information and improve prediction accuracy by a higher percentage. In ADE 2, as the adaptation step τ increases, the improved percentage first grew higher due to more information obtained, but then declined due to the behavior gap between earlier time and current time. When τ was 3, the short-term prediction was improved by 20% and 28% on the two scenarios respectively. In ADE 4, we can see online adaptation can barely benefit long-term prediction.

scenario, as the model was trained on the intersection scenario and directly transferred to the roundabout scenario. 3) The best adaptation performance is usually achieved by adapting the layer W_3^F, which is the last layer of the FC network in the decoder.

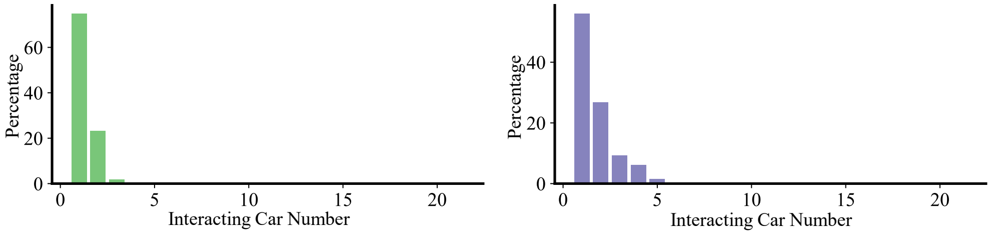
F Interacting car density

In multi-agent systems, it is important to answer the question which vehicles should be considered as the interacting vehicles. In some works, all the vehicles in the scene are considered as the interacting vehicles; while in some works, the interacting vehicles are defined as the vehicles within a certain range of the ego vehicle (Salzmann et al. (2020); Li et al. (2019)). However, we argue that distance may not necessarily determine interaction. For instance, cars that are close but driving in opposite direction may not be interacting at all. Essentially, interactions will happen among cars which are driving into a common area. We consider the interacting vehicles as the vehicles whose reference lines conflict with ego vehicles' reference line, and regard them as the node in our graph.

In Figure 11, we show the interacting density distribution as we choose interacting vehicles by different criteria. These results are collected from the real human data in the intersection scenario of the Interaction Dataset. In Figure 11(a), when we considered all the vehicles in the scene as the interacting vehicles, each ego car would be assumed to interact with 10 to 17 cars for most of the time. In Figure 11(b), when we took the vehicles within the range of 30 meters as the interacting car, there would be 5 to 10 interacting vehicles. Such results are counter-intuitive as it would be tough for humans to attend to so many cars at the same time. But when we considered cars within the range of 10 meters, as shown in Figure 11(c), the



(a) Car Density in the Scene (b) Car Density Within 30 Meters' Range



(c) Car Density Within 10 Meters' Range (d) Car Density In Semantic Graph

Figure 11: Percentage of interacting car number, when different criteria is used to choose interacting vehicles.

number of interacting cars was further reduced to be no more than 2. Nevertheless, such assumption also has its drawback, as people may still care about vehicles far from them as long as they are intervening each other. In our method, we care about vehicles whose reference lines conflict with ego car's reference line. As in Figure 11(d), the percentage of interacting vehicles number gradually decayed untill 5, which may be closer to real driving situations.

G Running time

A key consideration in robotics is the runtime complexity. In particular, we care about how the number of agents would affect the model's running time. Consequently, we evaluated the time it took our method to perform forward inference in different agent number. For points with insufficient number of agents, we imputed values by copying existing agents. As shown in Figure 12, both the SGN and EDN scaled well to the number of agents. For SGN, the running time was always near 0.001 seconds as the agent number increased to 100. The primary reason of such rapid computation is that in the SGN, the calculation for agents is conducted simultaneously in the form of matrix operation. In the EDN, the computation time was near 0.018 seconds, which is slower than the SGN due to the iterative decoding process in the decoder. The running time scaled well as all the agents can be batched and calculated simultaneously in one forward inference. As for the online adaptation, the running time correlates with the adaptation steps. For instance, the adaptation took an average of 0.03 seconds per sample when set the adaptation step as τ 1, but the average time for adaptation step of 3 was 0.1 seconds for per sample. According to these results, our method successfully meets the real-time computation requirement. Note that such a real-time computation speed is achieved via code in python, which, in practical deployment, can be rewritten in C++ and paralleled for further acceleration.

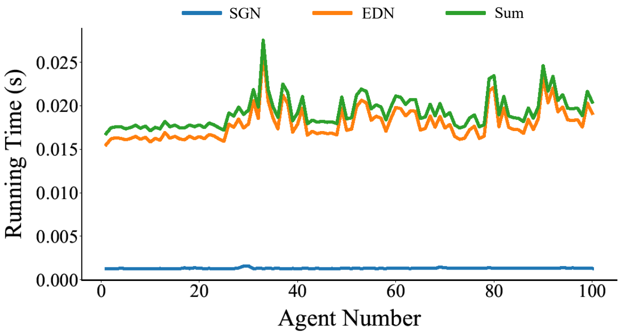


Figure 12: Running time of our method under different agent numbers.