IDE-Net: Extracting Interactive Driving Patterns from Human Data

Xiaosong Jia, Liting Sun✉, Masayoshi Tomizuka, and Wei Zhan[*]

Abstract

Autonomous vehicles (AVs) need to share the road with multiple, heterogeneous road users in a variety of driving scenarios. It is overwhelming and unnecessary to carefully interact with all observed agents, and AVs need to determine *whether* and *when* to interact with each surrounding agent. In order to facilitate the design and testing of prediction and planning modules of AVs, in-depth understanding of interactive behavior is expected with proper representation, and events in behavior data need to be extracted and categorized automatically. Answers to *what* are the essential patterns of interactions are also crucial for these motivations in addition to answering *whether* and *when*. Thus, learning to extract interactive driving events and patterns from human data for tackling the *whether-when-what* tasks is of critical importance for AVs. There is, however, no clear definition and taxonomy of interactive behavior, and most of the existing works are based on either manual labelling or hand-crafted rules and features. In this paper, we propose the **I**nteractive **D**riving event and pattern **E**xtraction **Net**work (IDE-Net), which is a deep learning framework to automatically extract interaction events and patterns directly from vehicle trajectories. In IDE-Net, we leverage the power of multi-task learning and proposed three auxiliary tasks to assist the pattern extraction in an unsupervised fashion. We also design a unique spatial-temporal block to encode the trajectory data. Experimental results on the INTERACTION dataset verified the effectiveness of such designs in terms of better generalizability and effective pattern extraction. We find three interpretable patterns of interactions, bringing insights for driver behavior representation, modeling and comprehension. Both objective and subjective evaluation metrics are adopted in our analysis of the learned patterns.

1 Introduction

1.1 Motivation

In various driving scenarios, autonomous vehicles need to interact with other road users such as vehicles, pedestrians and cyclists. It is computationally overwhelming and unnecessary for autonomous vehicles to pay equal attention to all detected entities simultaneously. Therefore, it is desired if we can identify *whether* each object may potentially interact with the ego vehicle, and *when* would be the starting and ending points of the interaction period, so that limited resources can be better allocated to tackle high-priority entities timely.

Meanwhile, predicting complex human driving behavior and designing human-like behaviors for autonomous vehicles are always challenging due to our insufficient understanding of interactive human behavior. Motions of interactive agents are intrinsically high dimensional. Desirable answers to *what* essential representation (with patterns) of such motions can be extracted or learned will

[*]X. Jia, L. Sun, M. Tomizuka and W. Zhan are with University of California, Berkeley. {jiaxiaosong, litingsun, wzhan, tomizuka}@berkeley.edu

Machine Learning for Autonomous Driving Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

significantly facilitate prediction and behavior modeling [1, 2], as well as imitation [3, 4], decision and planning [5] in terms of performances and computational efficiency with in-depth comprehension in a much lower dimensional space.

Moreover, with more and more trajectory data obtained from bird's-eye view [6] and ego-vehicle perspective [7], it is desired to learn to automatically extract interactive driving events (*whether* and *when*) and properly tag/categorize them (*what*) from large amounts of trajectory data in order to efficiently train and test behavior-related algorithms, as well as to generate scenarios and behavior for testing.

1.2 Related Works

1.2.1 Agent prioritization and scenario representation

In order to rank the priority of surrounding entities of the ego vehicle in a driving scene, [8] constructed a convolutional neural network to learn the ranking via supervised learning. Prioritization can also be implicitly learned via graph neural networks in prediction tasks [9]. However, the answers to *whether*, *when* and *what* for extracting interactions were not explicitly provided in these works. A comprehensive scene and motion representation framework was proposed in [9] to construct semantic graph for interactive vehicle prediction based on prior knowledge regarding the static and dynamic scenes. The representation is highly interpretable and reusable, but more extension is required to explicitly learn interactive patterns from data. Research on unsupervised representation learning was also presented recently to learn to identify driving modes in car following scenarios [10] or extract motion primitives for vehicle encountering [11]. The extracted patterns were expected to be more interpretable and reusable, and it is desired to train and test the models via driving data with complete information of surrounding entities in various highly interactive scenarios. In this work, we explicitly provide the answers to *whether*, *when* and *what* for extracting interactive events and patterns with better interpretability and reusability based on highly interactive driving data with complete surrounding information.

1.2.2 Interaction pattern extraction

Research efforts have been devoted to modeling interactions of agents. [12] constructed a relation network as an augmentation to other modules to handle relational reasoning. An attention-based method for multi-agent predictive modeling was presented in [13]. [14] proposed a hierarchical Bayesian model to perceive interactions from movements of shapes. [15] constructed a relation-aware framework to infer relational information from the interactions. The aforementioned works are based on either supervised learning of predefined interaction types or implicit modeling of interactions. In this paper, we explicitly extract interactive driving patterns with unsupervised learning to grant insight on how drivers tackle complicated driving scenarios and provide more interpretable representation for downstream modules.

Explicit inference over the potential interactions were performed in [16, 17]. Neural Relational Inference (NRI) [16] proposed a novel framework to output explicit interaction types of entities, and [17] further expanded the model to enable the combination of interactions types at the same time step. The interaction type between two objects is static, that is, given trajectories of two objects, NRI only provides one interaction type at the latest time step. The model has to be run over and over again by step to predict the interaction types in a period. Such independence among steps might yield frequent switches of interaction types, which is unrealistic according to [18]. Additionally, the framework determines the interaction type of a single time step based on the historical trajectories only, but the interaction types can be better inferred via utilizing the entire interaction trajectories (i.e., including both historical and future trajectories, similar as bi-LSTM outperforming LSTM). Our framework dynamically assigns interaction types for each time-step according to an overview of the complete trajectory to overcome the issues.

1.3 Contribution

Our contributions can be summarized as follows:

- We propose the **I**nteractive **D**riving event and pattern **E**xtraction **Net**work (IDE-Net) which can extract interactive events and patterns based on trajectory data of naturalistic human driving behavior.
- We propose a multi-task learning framework in IDE-Net to leverage the power of connected tasks in learning.
- We propose a spatial-temporal block with mixed LSTM and Transformer modules to encode trajectories, which achieves the best performance in experiments.
- We find three explainable interaction patterns by IDE-Net, which brings insight for driver behavior modeling. We did both objective and subjective evaluation to analyze the learned patterns. We also show that the proposed model could generalize well across scenarios with different road conditions and coordinate system.

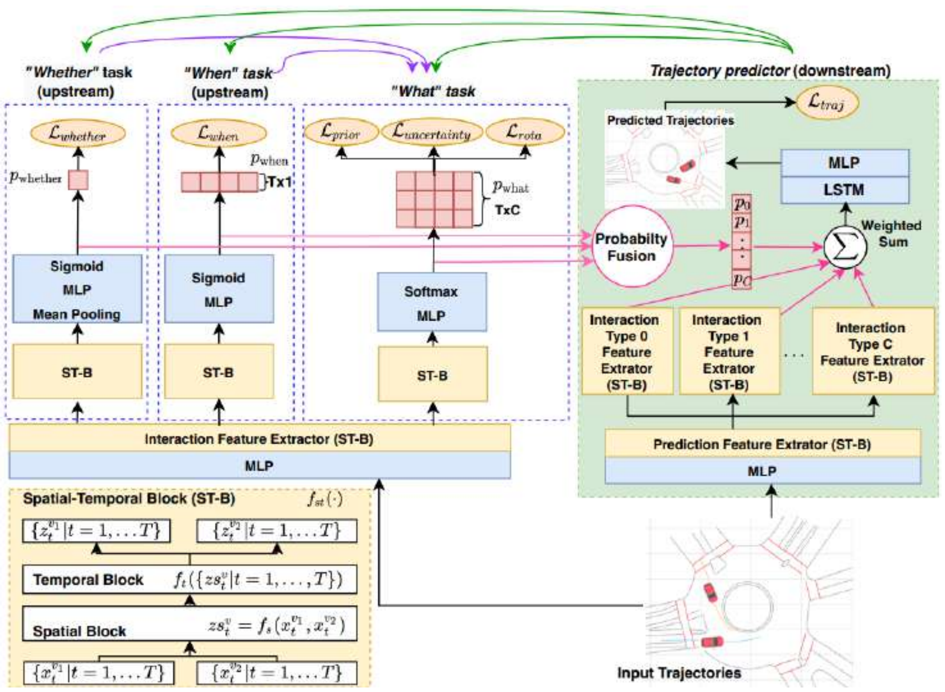

"Whether" task (upstream)
"When" task (upstream)
"What" task
Trajectory predictor (downstream)
$\mathcal{L}_{whether}$
$\mathcal{L}_{when}$
$\mathcal{L}_{prior}$
$\mathcal{L}_{uncertainty}$
$\mathcal{L}_{rota}$
$\mathcal{L}_{traj}$
Predicted Trajectories
p_{whether}
p_{when}
Tx1
p_{what}
TxC
Sigmoid
MLP
Mean Pooling
Softmax
ST-B
Probabilty Fusion
p_0
p_1
p_C
MLP
LSTM
Weighted Sum
Interaction Type 0 Feature Extrator (ST-B)
Interaction Type 1 Feature Extrator (ST-B)
Interaction Type C Feature Extrator (ST-B)
Interaction Feature Extractor (ST-B)
Prediction Feature Extrator (ST-B)
Spatial-Temporal Block (ST-B) $f_{st}(\cdot)$
$\{z_t^{v_1}|t=1,\dots T\}$
$\{z_t^{v_2}|t=1,\dots T\}$
Temporal Block $f_t(\{zs_t^v|t=1,\dots,T\})$
Spatial Block $zs_t^v = f_s(x_t^{v_1}, x_t^{v_2})$
$\{x_t^{v_1}|t=1,\dots T\}$
$\{x_t^{v_2}|t=1,\dots T\}$
Input Trajectories

Figure 1: The overview structure of the proposed IDE-Net. Its main task is to extract interactive driving patterns from human data, i.e., the unsupervised "*what*" task. To assist such an unsupervised learning task, we designed the IDE-Net to contain three more auxiliary tasks: a supervised "*whether*" task to predict interaction happens between the two vehicles, a supervised "*when*" task to predict the interacting time period, and finally a self-supervised trajectory prediction (TP) task (green box). The "*whether*" and "*when*" tasks are upstream tasks, thus they can utilize easy-to-obtain labels to extract low-level features that are shared with the "*what*" task. The TP task is a downstream task, which allows the IDE-Net to leverage the influence from the interaction types to the joint trajectories to better extract interaction patterns. For all the four tasks, we use several Spatial-Temporal Blocks (ST-B) $f_{st}(\cdot)$ to encode the spatial-temporal trajectories. MLP means multilayer perceptron.

2 Problem Formulation

In this work, we aim to design a learning framework which can automatically extract different interaction patterns in human driving behaviors by observing their joint trajectories. In a two-agent setting, a pair of joint trajectories from the two vehicles defines one sample. Suppose that the joint trajectories are of length T, i.e., T time steps. Then a sample contains (D^1, D^2) where $D^1 = \{\boldsymbol{d}_1^1, \boldsymbol{d}_2^1, ..., \boldsymbol{d}_T^1\}$ and $D^2 = \{\boldsymbol{d}_1^2, \boldsymbol{d}_2^2, ..., \boldsymbol{d}_T^2\}$ are, respectively, the trajectory data of vehicle 1 and vehicle 2. We also denote $D_t = \{\boldsymbol{d}_t^1, \boldsymbol{d}_t^2\}$ as the set of two vehicles' features at time step t. With these definitions, our goal is build a learning structure (such as a deep neural network) which takes in (D^1, D^2) and output interaction patterns at each time step, i.e., $l_{\text{type}} = \{l_{1,\text{type}}, l_{2,\text{type}}, ..., l_{T,\text{type}}\}$. No-interaction is also counted as one special interaction type.

Since there are no golden criteria regarding the interaction patterns, the proposed task is thus an unsupervised learning task without direct-accessible labels. In Section 3, we will introduce some unique network structures which helps us achieve such a goal.

3 Model Architecture

3.1 Trajectory Encoding via Mixed LSTM and Transformer

Note that interactive driving patterns emerge from the joint trajectories of two interacting vehicles. To encode such spatial-temporal signals, we propose a spatial-temporal block (SB-T), as shown in the yellow box in Fig. 1. Each ST-B contains one spatial block and one temporal block, sequentially connected. For the spatial block, Transformer layers [19] are introduced to obtain agent-permutation-invariant representation of the joint trajectories while fusing the relational information of two agents at each time step[1]. Such fused representation will be further encoded by the temporal block that contains several LSTM layers [22]. The relationship between the input and output of the ST-B can be represented by:

$$[Z^{v_1}, Z^{v_2}] = f_{\text{st}}([X^{v_1}, X^{v_2}]), \tag{1}$$

where $[X^{v_1}, X^{v_2}]$ is the input representation of the joint trajectories, and $[Z^{v_1}, Z^{v_2}]$ are correspondingly the output representation. Note that temporally, the lengths of the input and output equal. Therefore, several ST-Bs can be stacked to encode complex dynamics of the joint trajectories.

3.2 Learning via Auxiliary Tasks

To tackle the unsupervised "*what*" task, we design three auxiliary self/- supervised tasks in the IDE-Net to help extract interaction patterns. As shown in Fig. 1, the three tasks include: 1) an upstream "*whether*" task to predict whether interaction occurs between two vehicles, 2) another upstream "*when*" task to predict at which time steps such interaction occurs, and 3) a downstream trajectory prediction (TP) task to predict the joint future trajectories based on historical observations and the predicted interaction probabilities from the "*whether*", "*when*" and "*what*" tasks. We call the "*whether*" and "*when*" tasks as upstream tasks because interaction patterns will exist only when interaction happens, and the features used to encode the upstream tasks should be considered in the pattern extraction task. Therefore, we design a shared "Interaction Feature Extractor" for all the three "*whether*", "*when*" and "*what*" tasks in IDE-Net. Similarly, we call the TP task as a downstream task because different interaction patterns will influence the final trajectories of the vehicles, namely, the joint trajectories are effective indicators of interaction types. Via the "probability fusion" block and the weighted feature extractors in the TP task in the IDE-Net (Fig. 1), we allow such information flow. Hence, IDE-Net leverages the power of relatively easy but connected upstream and downstream tasks to extract unknown interaction patterns. Both the "*whether*" and "*when*" tasks are supervised tasks where the ground-truths are manually labeled via rules (see Appendix Section C for detailed rules). The TP task is self-supervised since all the ground-truth future joint trajectories can be directly observed from data itself.

3.3 Unsupervised "*What*" Task - Interaction Pattern Extraction

In the "*What*" task, we assume that there are C types of interaction patterns between two agents, and let network to predict the probability of each type at each time step. Although the three auxiliary tasks help reduce the difficulty of the "*What*" task, we cannot rely on those tasks due to the mode collapse problem. Specifically, the mode collapse problem means that the model would always tend to output only one kind of interaction type because the neural network's easiest strategy to minimize the trajectory prediction loss is to output only one type and optimize its corresponding parameters in the Trajectory Predictor, which is not what we want. Therefore, in IDE-Net, we introduce three additional loss to encourage it to extract meaningful interaction patterns, namely, the prior loss $\mathcal{L}_{\text{prior}}$, uncertainty loss $\mathcal{L}_{\text{uncertainty}}$, and rotation-invariant loss $\mathcal{L}_{\text{rota}}$, as shown in Fig. 1.

[1]Transformer is an attention-based order-independent set operation in deep learning. It can effectively fuse information in set in a pairwise way and has obtained huge success in the NLP field [20, 21].

Prior Loss. To alleviate mode collapse, we make an assumption that different interaction types approximately distribute uniformly in the dataset. Therefore, we would punish prediction outputs that are significantly deviating from such priors, i.e., a prior loss $\mathcal{L}_{\text{prior}}$ is given as the entropy of the interaction type distribution:

$$\mathcal{L}_{\text{prior}} = \sum^{C} p_c \log p_c, \text{ with } p_c = \frac{1}{NT} \sum^{N} \sum^{T} p^t_{\text{what},n,c}. \tag{2}$$

Uncertainty Loss. If only with prior loss, the model finds another way to cheat: it tends to output equal confidences for all kinds of interactions at all time-steps of all samples, which can minimize the prior loss but is not what we want as well. Therefore, to encourage the IDE-Net to output certain prediction at each time step for some interaction type c, an uncertainty loss is introduced to punish outputs with high entropy at each time step:

$$\mathcal{L}_{\text{uncertainty}} = \frac{1}{NT} \sum^{N} \sum^{T} \sum^{C} -p^t_{\text{what},n,c} \log p^t_{\text{what},n,c}. \tag{3}$$

By using the uncertainty term $\mathcal{L}_{\text{uncertainty}}$, the model would tend to output $\{[1,0,0],[0,1,0],[0,0,1]\}$ instead of simply $\{[0.333,0.333,0.333],[0.333,0.333,0.333],[0.333,0.333,0.333]\}$.

Rotation-Invariant Loss. Inspired by recent success of contrasting learning [23, 24], we would like to make the prediction of interaction types rotation-invariant. For each sample, we randomly rotate it twice with different angles and punish inconsistent outputs:

$$\mathcal{L}_{\text{rota-Invariant}} = \frac{1}{NTC} \sum^{N} \sum^{T} \sum^{C} (p^{t,1}_{\text{what},n,c} - p^{t,2}_{\text{what},n,c})^2, \tag{4}$$

where $p^{t,1}_{\text{what},n,c}$ and $p^{t,2}_{\text{what},n,c}$ represent the confidences of interaction type c at time-step t predicted based on the same sample n with two rotation angles respectively.

4 Experiment

4.1 Datasets and experiment settings

Trajectory datasets facilitating meaningful interaction extraction research should contain the following aspects: 1) a variety of driving scenarios with complex driving behaviors, such as roundabouts, unsignalized intersections, merging and lane change, etc.; 2) densely interactive driving behavior with considerable numbers of interaction pairs; 3) HD map with semantic information; 4) complete information (without occlusions) of the surrounding entities which may impact the behavior of the investigated objects. The INTERACTION dataset [6] meets all the aforementioned requirements, and our experiments were performed on it. We randomly selected 15657 pairs of vehicles from five different driving scenarios as the training set, and another 1740 pairs as the test set. We empirically set the number of interaction types as three[2], and it serves as a hyper-parameter of our model.

4.2 Data Preprocessing

Note that in structured driving environments, trajectories of vehicles differ significantly among different scenarios with maps. Our goal is to extract the interactive patterns between agents that are generic across all such scenarios, i.e., excluding the influence of different road structures. Therefore, we propose three data preprocessing methods to reduce such influence from road structures (Appendix H gives the ablation study for the three methods):

- Coordinate Invariance (CI): CI converts global coordinates to relative coordinates by setting the origin of coordinate system as the center position of the two agents' initial positions.
- Orientation Invariance (OI): OI helps to make features not sensitive to absolute orientations by random rotation data augmentation at each batch.
- Rotation Normalization (RN): RN normalizes the coordinates in different scenarios according to the mean and covariance of the augmented data in OI. Details regarding the RN step is given in the Appendix Section F.

[2]We performed user studies on the results of different number of the interaction types. Three is the number with the best performance.

4.3 Performance of the "When" and "Whether" Tasks

Method	Setting	FT		GL		MA		SR		EP	
		When	Whether	When	Whether	When	Whether	When	Whether	When	Whether
LSTM	Single	0.818	0.926	0.914	0.923	0.783	0.862	0.500	0.891	0.904	0.889
	Transfer	0.491	0.798	0.599	0.724	0.635	0.793	0.615	0.864	0.735	0.865
	Finetune	$\underline{0.837}$	$\underline{\mathbf{0.912}}$	$\underline{0.937}$	$\underline{0.924}$	$\underline{0.815}$	$\underline{\mathbf{0.871}}$	$\underline{0.750}$	$\underline{0.923}$	$\underline{0.923}$	$\underline{\mathbf{0.944}}$
Transformer	Single	0.740	0.877	0.862	0.884	0.727	0.839	0.147	0.897	0.333	0.846
	Transfer	0.433	0.777	0.528	0.718	0.535	0.760	0.473	0.884	0.612	0.841
	Finetune	$\underline{0.751}$	$\underline{0.882}$	$\underline{0.883}$	$\underline{0.901}$	$\underline{0.741}$	$\underline{0.876}$	$\underline{0.559}$	$\underline{0.910}$	$\underline{0.778}$	$\underline{0.923}$
Mixed	Single	0.848	$\underline{0.911}$	0.918	0.922	0.839	0.859	0.706	0.936	0.907	0.892
	Transfer	0.542	0.812	0.681	0.692	0.634	0.802	0.637	0.879	0.653	0.833
	Finetune	$\underline{\mathbf{0.878}}$	0.909	$\underline{\mathbf{0.946}}$	$\underline{\mathbf{0.929}}$	$\underline{\mathbf{0.853}}$	$\underline{0.869}$	$\underline{\mathbf{0.882}}$	$\underline{\mathbf{0.949}}$	$\underline{\mathbf{0.930}}$	$\underline{\mathbf{0.944}}$

Table 1: Prediction performance comparison of three ST-B settings. FT, GL, MA, SR and EP are different scenarios in the INTERACTION dataset.

Manipulated Factors. We manipulated a single factor: the structure of the ST-B in IDE-Net. Three conditions were compared: 1) LSTM layers only, 2) Transformer layers only, and 3) the proposed mixed LSTM and Transformer in Section 3.1. In setting 2), we add the position embedding similar to [19] to capture the temporal signals.

Dependent Variables. For the "*whether*" task, we measured the prediction accuracy. For the "*when*" task, we measured the intersection-over-union (IoU) accuracy and adopt **IoU 0.6 accuracy** as correct prediction, namely, if IoU of a prediction is larger than 0.6, it counts as an accurate prediction. IoU is defined as:

$$\text{IoU} = \frac{[\hat{l}_{\text{start}}, \hat{l}_{\text{end}}] \cap [l_{\text{start}}, l_{\text{end}}]}{[\hat{l}_{\text{start}}, \hat{l}_{\text{end}}] \cup [l_{\text{start}}, l_{\text{end}}]}, \tag{5}$$

where $\hat{l}_{\text{start}}$ and $\hat{l}_{\text{end}}$ are, respectively, the predicted starting and ending frames of interactions.

Hypothesis. We hypothesize that the proposed mixed LSTM and Transformer ST-B structure can achieve better prediction accuracy in "*whether*" and "*when*" tasks compared to the other two settings.

Results. We tested the prediction performances on five different scenarios (FT, GL, MA, SR, EP) in the INTERACTION dataset [6] with diverse number of samples ranging from 9000 to 100. We tested the performance under three different training settings: 1) *Single* means training and testing both with the current one, 2) *Transfer* means training with all the other scenarios and testing on the current one, and 3) *Finetune* means training with all the other scenarios and then finetuning with the current one.

The results are shown in Table 1. We can conclude that: 1) "*whether*" task was much simpler than "*when*" task since even training without samples of the current scenario in the dataset, it was still able to achieve decent results (> 0.7); 2) regarding the prediction performance, Mixed $>$ Pure LSTM $>$ Pure Transformer, which shows the effectiveness of Transformer for set of data and LSTM for sequence of data; and 3) finetuning always had the best performance among the three settings. It proved that better generalization ability can be achieved by taking into account more samples from other driving scenarios which may have completely different road structures. Therefore, for all the following experiments, we used the proposed mixed ST-B as it achieved the best performance among the three compared structures.

4.4 Results of the "*What*" Task

4.4.1 Learned Patterns

From the interaction types given by the IDE-Net, we conclude the three following patterns (interaction type):

- **Noticing**: two vehicles observe each other and realize that there would be overlaps between their paths. They slow down to avoid collisions, although the speeds may still be relatively high.
- **Yielding**: one vehicle tends or decides to yield.

- **Caution**: two vehicles are relatively close to each other; one moving and the other may also be moving but cautiously.

Visualizations of the learned patterns are in Appendix Section A

4.4.2 Quantitative Evaluation

To quantify the performance of the pattern extraction task (*what* task), we adopted two metrics: 1) a cross-validation metric which compares the distribution difference between the extracted pattern types in the training and test sets; and 2) a user study which compares the human labeled interaction patterns with the extracted ones via the IDE-Net.

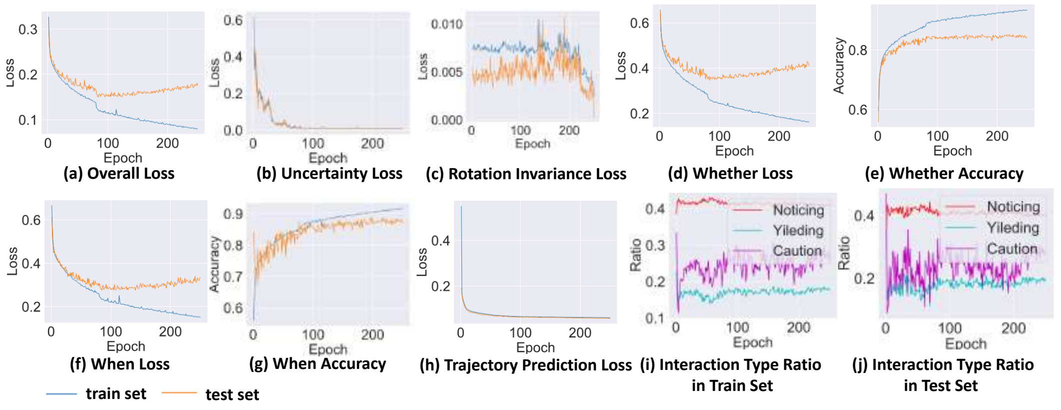


Figure 2: An overview of the performance of IDE-Net

Type	Noticing	Yielding	Caution
Train	0.469	0.208	0.322
Test	0.452	0.229	0.322

(a) Ratios of each interaction type on training and testing sets

VTA	MTA	VAA	MAA
92.5%	76.0%	100.0%	90.5%

(b) Results of user study.

Table 2: Results with both the cross-validation metric and the user study

Results with the Cross-Validation Metric. The percentages of the extracted interaction patterns are shown in Fig. 2 (i)-(j) and Table 2 (a). The distribution of the three interaction patterns on the training and test sets were quite similar, which indicates good generalization capability of the proposed IDE-Net, i.e., it did not overfit the training set but indeed learned to extract different interaction patterns of vehicles.

User Study We also conducted an user study to provide a subjective metric to evaluate the performance of IDE-Net. We recruited 10 users with driving experiences, 5 males and 5 females. We visualized the interactive trajectories as well as the extracted interaction types together as videos and showed each user 12 videos (each scenario has at least 2 videos). We asked the users to label 1) one of the three interaction types by themselves, and 2) whether they agree with the predicted interaction types. The procedure of the user study is as follows:

1. Initial setting: We told users the meaning of the three interaction types: Noticing (N), Yielding(Y), and Caution(C). Each user watched all 12 videos in a random order.
2. Experiment I: We showed users videos only with marks indicating whether the two vehicles were interacting at each time step. We asked users to provide the interaction types and their order.
3. Experiment II: We showed them videos with marks indicating the interaction types at each time step. We asked users whether they agree with the types labeled by IDE-Net.

Metric: In Experiment I, we considered the interactions types labeled by the model were aligned with user inputs if both the interaction types and their order were the same. We calculate two metrics based on it: 1) for each video, we averaged the human labels by voting (Voting Type Accuracy - VTA); and 2) we took the mean accuracy over all videos and all users (Mean Type Accuracy - MTA). In Experiment II, similarly, we defined two metrics: *Voting Agreed Accuracy - VAA* and *Mean Agreed Accuracy - MAA*, which were obtained via the similar process as the *Voting Type Accuracy* and *Mean Type Accuracy* in Experiment I.

The results are shown in Table 2(b). The interaction types labeled by the users aligned quite well with the results obtained by the IDE-Net. This shows that IDE-Net was able to extract the generic interaction patterns and the extracted patterns are quite interpretable.

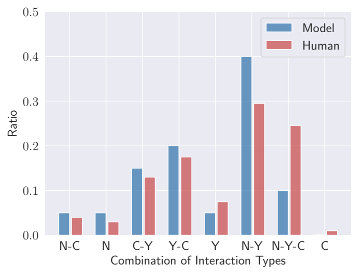


Figure 3: The ratio of permutations labeled by the model and by human.

	Noticing	Yielding	Caution
Model	0.60	0.90	0.50
Human	0.61	0.92	0.60

Table 3: The frequency of interaction types.

The performance of IDE-Net over all losses is summarized in Fig. 2(a)-(h). We can see that although the supervised "*whether*" and "*when*" tasks show a sign of over-fitting, the uncertain loss kept decreasing and no over-fitting occurred for the trajectory prediction task. Moreover, the consistent decreasing in uncertain loss means that the model became more and more certain about the interaction types of each time step. We also counted the time steps when the confidence over a specific interaction type was significantly larger than the other two, i.e., the confidence was greater than 0.9 and divided it by the overall length of the interaction window. We find that the ratio was more than **99.5%**, which indicates that the proposed model was able to find the significant differences among interaction types.

We also calculated the frequency of different interaction types. As shown in Table 3, the results given by IDE-Net were quite close to that given by human, although human tended to label more interaction types. It means that IDE-Net was relatively conservative.

To explore the co-appearance relationship of different interaction types, we define the interaction types and the order of their appearance as a permutation. For instance, in a video, if the two vehicles are labeled as first Noticing, then Yielding, and finally Caution, the permutation of this prediction is then "N-Y-C". We calculated the ratio of permutations labeled by the model and by users as shown in Fig. 3. Human behaved quite similar to the model except that human tend to give more "N-Y-C". "N-Y-C" was preferred probably due to the biased default in human, i.e., all three interaction types should appear and show up in sequence.

5 Conclusion

In this paper, we proposed a framework to automatically extract interactive driving events and patterns from driving data without manual labelling. Experiments on the INTERACTION dataset across different driving scenarios were performed. The results showed that through the introduction of the Spatial-Temporal Block, the mutual attention and influence between trajectories of agents were effectively extracted, as evaluated via both objective and subjective metrics. Moreover, we found that the extracted three interaction types are quite interpretable with significantly different motion patterns. Such a framework can greatly boost research related to interactive behaviors in autonomous driving, e.g., interactive behavior analysis and interactive prediction/planning, since it can be used to as a tool to extract not only the interactive pairs but also the interaction patterns from enormous of detected trajectories.

There are still limitations about this work. Currently, the proposed framework focuses on two-agent settings, and for multi-agent settings, we have to enumerate all possible combinations. In real world, however, interactions may not be just between two vehicles, but can be potentially among multi-agents (more that two). Additionally, road structure information is also important for the capturing of interactions. In our future work, we will extend towards these two directions by further designing ST-B blocks for multiple entities (including both vehicles and road structures).

References

[1] K. Brown, K. Driggs-Campbell, and M. J. Kochenderfer, “Modeling and prediction of human driver behavior: A survey,” *arXiv preprint arXiv:2006.08832*, 2020.

[2] L. Sun, W. Zhan, Y. Hu, and M. Tomizuka, “Interpretable modelling of driving behaviors in interactive driving scenarios based on cumulative prospect theory,” in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 4329–4335.

[3] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, “End-to-end driving via conditional imitation learning,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[4] N. Rhinehart, R. McAllister, and S. Levine, “Deep Imitative Models for Flexible Inference, Planning, and Control,” in *2019 International Conference on Learning Representations (ICLR)*, 2019.

[5] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, “End-to-end interpretable neural motion planner,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[6] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, and M. Tomizuka, “INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps,” *arXiv:1910.03088 [cs, eess]*, 2019.

[7] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, “Scalability in perception for autonomous driving: Waymo open dataset,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[8] K. S. Refaat, K. Ding, N. Ponomareva, and S. Ross, “Agent prioritization for autonomous navigation,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2060–2067.

[9] Y. Hu, W. Zhan, and M. Tomizuka, “Scenario-transferable semantic graph reasoning for interaction-aware probabilistic prediction,” *arXiv preprint arXiv:2004.03053*, 2020.

[10] Q. Lin, Y. Zhang, S. Verwer, and J. Wang, “MOHA: A Multi-Mode Hybrid Automaton Model for Learning Car-Following Behaviors,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–8, 2018.

[11] W. Wang, W. Zhang, and D. Zhao, “Understanding V2V Driving Scenarios through Traffic Primitives,” *arXiv:1807.10422 [cs, stat]*, Jul. 2018, arXiv: 1807.10422.

[12] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, “A simple neural network module for relational reasoning,” in *Advances in neural information processing systems*, 2017, pp. 4967–4976.

[13] Y. Hoshen, “Vain: Attentional multi-agent predictive modeling,” in *Advances in Neural Information Processing Systems*, 2017, pp. 2701–2711.

[14] T. Shu, Y. Peng, L. Fan, H. Lu, and S.-C. Zhu, “Perception of human interaction based on motion trajectories: From aerial videos to decontextualized animations,” *Topics in cognitive science*, vol. 10, no. 1, pp. 225–241, 2018.

[15] C. Choi and B. Dariush, “Learning to infer relations for future trajectory forecast,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[16] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, “Neural relational inference for interacting systems,” *arXiv preprint arXiv:1802.04687*, 2018.

[17] E. Webb, B. Day, H. Andres-Terre, and P. Lió, “Factorised neural relational inference for multi-interaction systems,” *arXiv preprint arXiv:1905.08721*, 2019.

[18] L. Sun*, M. Cai*, W. Zhan, and M. Tomizuka, “A game-theoretic policy-aware interaction strategy with validation on real traffic data,” in *2020 IEEE Conference on Intelligence Robots and Systems (IROS)*. IEEE, 2020.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

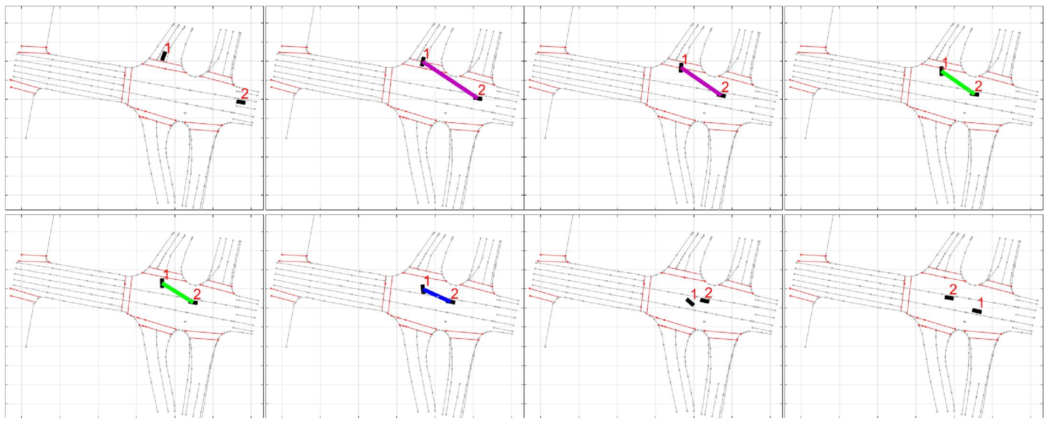
[21] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

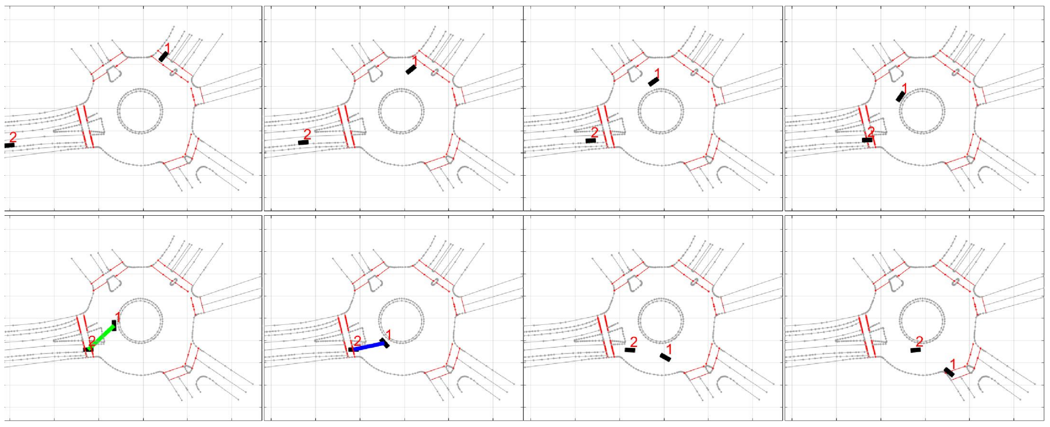
[23] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *arXiv preprint arXiv:2003.04297*, 2020.

[24] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton, "Big self-supervised models are strong semi-supervised learners," *arXiv preprint arXiv:2006.10029*, 2020.

Appendix A Some Illustrative Examples of the Learned Patterns

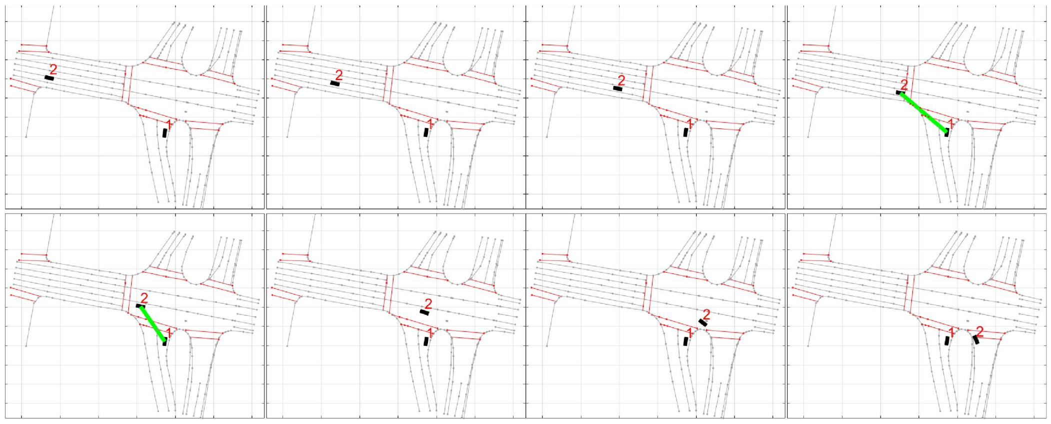


(a) Noticing (Purple)-Yielding (Green)-Caution (Blue). For simplicity, we denote one vehicle as $v1$ and the other as $v2$. The interaction began when the two vehicles noticed each other (Purple). Then, $v2$ yielded and the $v1$ kept going (Green). Finally, $v2$ started driving slowly after $v1$ passed and the interaction ended.

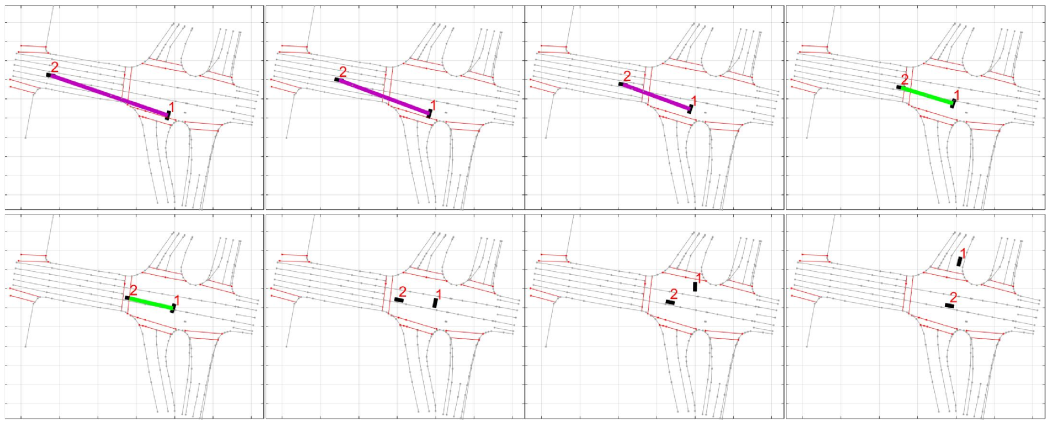


(b) Yielding (Green)-Caution (Blue). Before $v1$ came, the $v2$ was waiting at stop-sign and thus there was no Noticing part and $v1$ just passed while $v2$ kept yielding (Green). Then, once vehicle passed the front of $v2$, $v2$ started immediately and followed the vehicle closely (Blue). Finally, the two vehicles went to their goals and the interaction ended.

Figure 4: Examples of interaction patterns.



(a) Yielding (Green). $v1$ stopped before $v2$ came and thus no noticing part. Long after $v2$, $v1$ started. There was no potential collisions at all (no caution part). This sample shows that the model could capture patterns of vehicles' motions separately.



(b) Noticing (Purple)-Yielding (Green). Both vehicles noticed each other very early and slowed down. $V2$ yielded until the $v1$ had passed. There is caution part because their directions are perpendicular and once one of the vehicle passed, it was impossible to collide and thus there was no need to be too cautious.

Figure 5: Examples of interaction patterns.

Figures 4 and 5 demonstrate cases of the learned interaction types at different time steps of interactive trajectory pairs. We uniformly sampled 8 frames to represent each example video. In each subfigure, the line between the two vehicles represents that they are interacting and different colors represent different interaction types (Noticing-Purple, Yielding-Green, Caution-Blue).

Appendix B Differences among Learned Interaction Patterns

There may be concerns that the interaction types and their corresponding features for trajectory prediction were very similar to each other. Thus, in this section, we checked the differences among the three learned interaction patterns. We visualized the the hidden layer before the output MLP layer for the *What* task, as well as the hidden layer before the LSTM module for the Trajectory Prediction task. As shown in Fig. 6, clusters of interaction patterns are linearly separable in both figures. The results show that the model can classify interaction types with high confidences and predict trajectory in a interaction-type-specific way.

Additionally, we visualize the interaction types of each time-step in a feature space including the relative distance and relative speed. In Fig. 7 (left), their distributions were different. The characteristics of three interaction types are as follows: "Noticing" (Orange) happens when the two vehicles are driving towards a potentially conflicting area but still at relatively larger distances

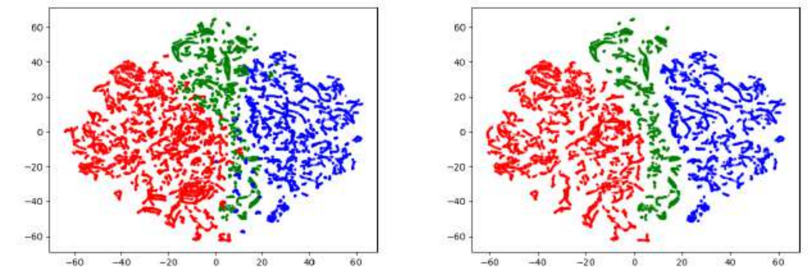


Figure 6: Visualization for features of time-steps by t-sne. Each color represents a type of interaction. (Left) Features before fed into the output MLP for the *What* task. (Right) Features before fed into the LSTM for the Trajectory Prediction task..

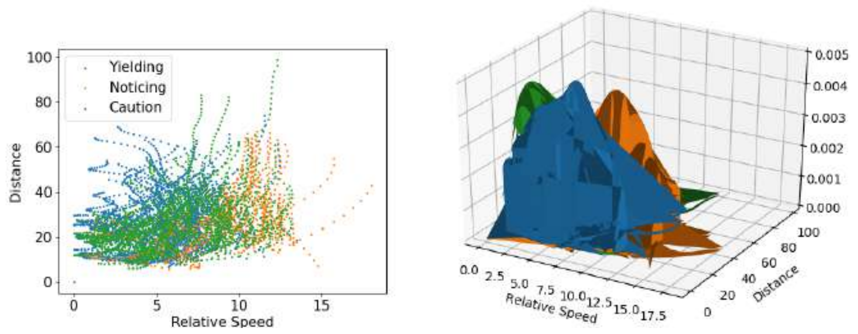


Figure 7: Orange-Noticing, Blue-Yielding, Green-Caution. (Left) Visualization on relative distance - relative speed space with each color representing a interaction type. (Right) Its probability density distribution estimated by Gaussian kernel density estimation.

compared to "Yielding" (Blue) and relatively larger speeds compared to "Caution" (Green); "Yielding" features smaller relative distances; and "Caution" features the smallest relative speed, indicating negotiation.

To have a better illustration of the differences of their distributions, we conducted the kernel density estimation to estimate the probability density function As shown in Fig. 7 (right), the three interaction types has different peaks, which shows that our model can capture the difference of different interactions patterns considering both relative speed and distance.

Appendix C Labeling Rules for Supervised Whether and When Task

In the experiments, we use the INTERACTION Dataset. There are no labels in the original dataset. Thus, for the *whether* task, we assign labels l_{whether} where 0 represents no-interaction, 1 represents having interaction, and -100 represents not-sure. For the *when* task, we assign labels l_{start} and l_{end} to represent, respectively, the start time index and end time index of the interaction period. All the time steps between l_{start} and l_{end} are labeled as 1 and otherwise 0. We design two rules to label the samples:

1. For each vehicle which stops at the stop sign, if it stops for more than 3 seconds (a hyper-parameter), then for all the vehicles passing through its front zone while it is stopping, we label the pair of vehicles as an interaction pair, i.e., $l_{\text{whether}} = 1$. The intuition here is: if a vehicle stops longer than what is required by traffic law, it is highly likely that it is waiting for other vehicles, which should count as an interaction process.

As for $(l_{\text{start}}, l_{\text{end}})$, we label the first time step when the passing vehicle is less than 20 meters (another hyper-parameter) away from the stopping vehicle's front zone as l_{start} and the time step that moving vehicle passes the front zone as l_{end}. The reason is that when vehicles are close, there might be highly interactive behaviors to avoid collisions and keep efficiency.

To give some negative samples, when a stopping vehicle stops for less than 1 second (the third hyper-parameter), we label the pair of vehicles as $l_{\text{whether}} = 0$ because the stopping vehicle stops so shortly that it might not wait for other vehicles, i.e., no interaction exists.

For those samples with a stopping vehicle stopping between 1-3 seconds, we label them as not-sure samples. They are not used to calculate the supervised *whether* and *when* loss.

2. For each pair of vehicles, if their minimal time-to-collision (TTC)[1] interval is less than 3 seconds, we label them as $l_{\text{whether}} = 1$.

   We also set the first time step when both vehicles are less than 20 meters (the fourth hyper-parameter) away from their potential collision points as l_{start} and the first time step when one of them passes the potential collision points as l_{end}. The intuition of this rule is similar: when two vehicles are close to each other, there might be interactions.

   Also, we set negative samples as the pairs of vehicles whose minimal time-to-collision time is larger than 8 seconds and not-sure samples as between 3-8 seconds.

Additionally, for pairs of vehicles which have no time-to-collision and none of which stops at a stop sign, they are labelled as $l_{\text{intera}} = 0$. For each scenario, we randomly sample those negative samples until the number of positive samples and negative samples are approximately equal.

Appendix D Comparison with Neural Relational Inference

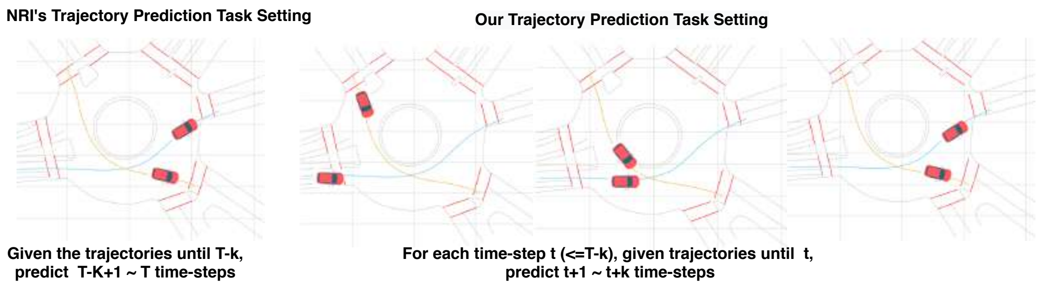

NRI's Trajectory Prediction Task Setting
Our Trajectory Prediction Task Setting
Given the trajectories until T-k, predict T-K+1 ~ T time-steps
For each time-step t (<=T-k), given trajectories until t, predict t+1 ~ t+k time-steps

Figure 8: Comparison with NRI[16] about trajectory prediction auxiliary setting. Under their setting, for each sample, they only do trajectory prediction for the time-step T-k. While in IDE-Net, we do trajectory prediction for each time-step t ($t \leq T - k$). The advantage of doing in the latter way is that: at different time-steps, there may be different interaction types which may have different influence in the future motion of vehicles. By predicting at each time-step, the influence of interaction type could be fully considered. In contrast, if doing prediction in the former way, what happened before T-k may not have much influence on their future motions. As a result, the influence of interaction may be weaken or even lost.

Fig. Fig. 8 compares the setting for the Trajectory Prediction Task of NRI and the IDE-Net. Additionally, to avoid data leakage, we use causal mask (mask for future time-steps) in the downstream tasks. However, in the upstream tasks and *what* task, all blocks do not have masks for time-steps in the future (causal mask), which means each time-step has access to features of all the other time-steps. There are two reasons not to have causal mask. First, interactions are about a period of time and it is hard to decide the interaction type of one time-step without knowing the entire trajectory. Secondly, since we have the auxiliary task to predict vehicles' future coordinates in each time-step, there may be concerns about data leakage. However, in the proposed structure, only interaction types of each time-step are passed to the trajectory predictor. So, if this leakage can be very helpful for the trajectory prediction, it means that the predicted interaction types can capture their influences on vehicles motions, which is the goal of our model.

[1]The TTC is defined as the estimated time for the vehicle to arrive at the collision point assuming that the vehicle is running at the constant speed.

Appendix E Statistics of the Dataset

Scenario	# Sample	# Positive Sample	# Negative Sample	# Not Sure Sample
USA_Roundabout_SR	481	182	207	92
USA_Roundabout_FT	9445	3965	3978	1402
USA_Roundabout_EP	189	62	90	27
USA_Intersection_MA	1702	734	752	216
USA_Intersection_GL	5580	2281	2355	944

Table 4: Statistics of obtained samples..

We have 17397 samples from 5 scenarios of the INTERACTION dataset. The statistics is given in Tab. Table 4. Note that we choose the five scenarios since 1. the number of samples is diverse so that we can evaluate the generalization ability and transferring ability of the model. 2. they have different road condition - FT, SR, EP are Roundabouts while GL and MA are intersections.

Appendix F Details about Rotation Normalization

Since coordinates in different scenarios may have different scales of coordinate, we need to do normalization properly to make the performance of the model generalize well across scenarios. Before that, we would like to first find out the expectation and variance of coordinates after the random rotation in the Orientation Invariance (OI) step. Without loss of generality, we take the x coordinate as an example and it is easy to extend to y. After randomly rotation, the rotated coordinate $x' = x * \cos\theta - y * \sin\theta$ where $\theta \sim U(0, \pi)$ is randomly sampled rotation angle for the coordinate system. The goal of RN is to let $Ex' = 0$ and $Dx' = 1$ where $E(\cdot)$ and $D(\cdot)$, respectively, represent the expectation and variance among all samples and all time steps. Since x, y, and θ are independent and $\theta \sim U(0, 2\pi)$, we have:

$$\begin{aligned} Ex' &= E(x * \cos\theta - y * \sin\theta) \\ &= \frac{1}{2\pi} E \int_0^{2\pi} x * \cos\theta - y * \sin\theta \mathrm{d}\theta \\ &= 0 \end{aligned} \tag{6}$$

which means we do not need to do transnational movement of the coordinate axis in the normalization step. Its expectation is already 0 after random rotation. Then, for variance, we have:

$$\begin{aligned} Dx'_t &= E(x')^2 - (Ex')^2 \\ &= E(x * \cos\theta - y * \sin\theta)^2 \\ &= \frac{1}{2\pi} E \int_0^{2\pi} (x * \cos\theta - y * \sin\theta)^2 \mathrm{d}\theta \\ &= E\frac{x^2 + y^2}{2} \end{aligned} \tag{7}$$

To let $Dx'_t = 1$, we divide raw coordinate x by $\sqrt{E\frac{(x)^2+(y)^2}{2}}$. Similarly, since variance equation is symmetric for x and y, we could divide raw coordinate y by the same value.

Appendix G Ablation Study for the Multi-Task Learning Paradigm

IDE-Net aims to extract interpretable interaction types between vehicles. Towards the goal, the proposed "*whether-when-what*" paradigm has two benefits. First, the three tasks are intrinsically hierarchical with semantic meanings and increasing complexity. The existence of upstream tasks can improve the performance of downstream tasks. Second, the feature-sharing structure leverages the power of multi-task learning to boost both the learning performance and efficiency. We conducted ablation studies to support these two statements.

We conducted several comparison experiments: 1) only *whether* task, 2) only *when* task, 3) only *whether* and *when* tasks, and 4) our approach. Under each experiment setting, we calculate the

Whether	When	What	Whether Acc	When Acc	Best Epoch
✓			0.893	-	264
	✓		0.181	0.843	279
✓	✓		0.898	0.867	263
✓	✓	✓	0.909	0.878	291

Table 5: Ablation study for *whether* and *when* tasks under different multi-task settings. Note that when only having the *when* task, if the output sequence contains no "1" labels at all, the output of "whether" task is assigned as "0".

accuracy (Acc) of the *whether* and *when* tasks. The goal of this set of comparison experiment is to demonstrate the power of feature-sharing structure in improving the learning efficiency and performance.

Results are shown in Tab. 5. Under the multi-task learning framework, performances of both *whether* and *when* tasks were improved remarkably. Note that the Whether Acc was bad with *when* task only. The reason is that compared to the binary classification in "whether" task, it is much harder for *when* task to output all 0s for the non-interactive trajectories due to the probabilistic nature of deep learning. Additionally, the parallel training of multiple tasks did not make a significant difference in terms of converging time (best epoch) compared to training those tasks separately. Therefore, the multi-task learning framework, which was able to avoid the burden of learning different low-layer features for each task, is efficient.

We have also conducted another user study for the results with *what* task only. The users cannot distinguish the extracted interaction patterns. This proved that the proposed multi-task structure can improve the interpretability of the learned patterns.

Appendix H Ablation Study for the Three Proposed Data Preprocessing Methods

Method	Setting	Zero Norm		No Rotation		No Local Scale		Our Method	
		When	Whether	When	Whether	When	Whether	When	Whether
LSTM	Single	0.808	0.879	0.822	0.889	0.820	0.888	0.818	0.926
	Transfer	0.237	0.769	0.453	0.763	0.494	0.801	0.491	0.798
	Finetune	0.796	0.890	0.818	0.881	0.788	0.860	0.837	**0.912**
Transformer	Single	0.776	0.872	0.740	0.874	0.741	0.869	0.740	0.877
	Transfer	0.300	0.741	0.358	0.736	0.437	0.782	0.433	0.777
	Finetune	0.723	0.748	0.728	0.865	0.710	0.840	0.751	0.882
Mixed	Single	0.841	0.901	0.847	0.885	0.837	0.895	0.848	0.911
	Transfer	0.242	0.781	0.253	0.835	0.546	0.805	0.542	0.812
	Finetune	0.852	0.901	0.833	0.877	0.847	0.892	**0.878**	0.909

Table 6: Ablation Study on FT dataset. *Zero Norm* uses the common zero-score normalization instead of the proposed rotation normalization (RN); *No Rotation* does not use the proposed Orientation Invariance (OI), instead, it uses the average orientation at the final observed time step as the positive axis; *No Local Scale* does not use the proposed Coordinate Invariance (CI). **Bold** represents the best under the current metric among all and underline represents the best under the current metric with the current model.

Method	Zero Norm		No Rotation		No Local Scale		Our Method	
	When	Whether	When	Whether	When	Whether	When	Whether
LSTM	0.835	0.886	0.851	0.889	0.858	0.887	0.878	0.891
Transformer	0.800	0.879	0.785	0.871	0.799	0.872	0.812	0.887
Mixed	0.856	0.898	0.865	0.895	0.878	0.885	**0.882**	**0.910**

Table 7: Ablation Study on all scenarios. *Zero Norm* uses the common zero-score normalization instead of the proposed rotation normalization (RN); *No Rotation* does not use the proposed Orientation Invariance (OI), instead, it uses the average orientation of the two vehicles at the final observed time-step as the positive axis of the coordinate system; *No Local Scale* does not use the proposed Coordinate Invariance (CI). **Bold** represents the best under the current metric among all and underline represents the best under the current metric with the current model.

We also did an ablation study to show the effectiveness of our proposed three data preprocessing methods. First, we did an ablation study on the FT dataset (which has the largest number of samples). We tested the performance under three different training settings: 1) *Single* means training and testing with the only one scenario, 2) *Transfer* means a training with all the other scenarios and test on different one, and 3) *Fine-tune* means training with all the other scenarios and then fine-tuning with the current one. We show the results under three different Spatial-Temporal Blocks: 1) only LSTM layers, 2) only Transformer layers, and 3) the proposed mixed LSTM and Transformer. The results are in Tab. 6.

From Tab. 6, we can conclude that: 1. Under the Fine-tune setting which has the best performance, all three data pre-processing methods could improve the model's performance. 2. Under the Single Setting, not all the pre-processing methods help improve the performance. We think it may be because removing one of the pre-processing method could make the model much easier to overfit the scenarios. However, the generalization ability is poor. To further verify this hypothesis, we trained and tested with all the scenarios to see how the data pre-processing methods could influence the results when the train and test data both contain diverse driving scenarios. The results is in Tab. 7. We can see in Tab. 7, removing any of the three data pre-processing methods would significantly reduce the performance in terms of generalizability.

Appendix I Hyper-Parameter Setting

In the experiment, we use GELU as the activatation function, AdamW as optimizer, and warm-up learning rate schedule. We set weight_decay as 1e-7, learning rate as 3e-6, epoch as 250, dropout as 0.01, ratio of warm-up as 0.01, and gradient-clip as 10. As for model, we use hidden-dimension as 384, head-num as 16. All module has 2 Spatial-Temporal Blocks (ST-B). For the trajectory prediction, we predict 5 future steps. The weights of each loss terms: Supervised Whether Loss - 0.233, Supervised When Loss - 0.233, Trajectory Prediction Loss - 0.007, Prior Loss - 0.233, Uncertainty Loss - 0.007, Rotation-invaraince Loss - 0.023.