Joint Interaction and Trajectory Prediction for Autonomous Driving using Graph Neural Networks

Donsuk Lee
School of Informatics, Computing, and Engineering
Indiana University, Bloomington, IN
donslee@iu.edu

Yiming Gu
Uber ATG
50 33rd St, Pittsburgh, PA
yiming@uber.com

Jerrick Hoang
Uber ATG
50 33rd St, Pittsburgh, PA
jhoang@uber.com

Micol Marchetti-Bowick
Uber ATG
50 33rd St, Pittsburgh, PA
mmarchettibowick@uber.com

Abstract

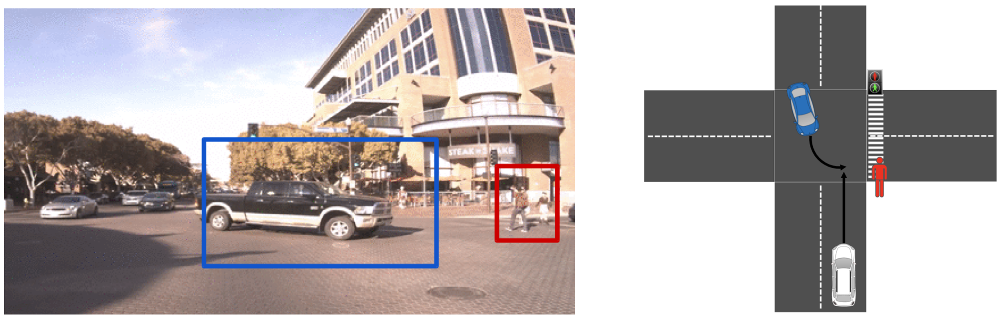
In this work, we aim to predict the future motion of vehicles in a traffic scene by explicitly modeling their pairwise interactions. Specifically, we propose a graph neural network that jointly predicts the discrete interaction modes and 5-second future trajectories for all agents in the scene. Our model infers an interaction graph whose nodes are agents and whose edges capture the long-term interaction intents among the agents. In order to train the model to recognize known modes of interaction, we introduce an auto-labeling function to generate ground truth interaction labels. Using a large-scale real-world driving dataset, we demonstrate that jointly predicting the trajectories along with the explicit interaction types leads to significantly lower trajectory error than baseline methods. Finally, we show through simulation studies that the learned interaction modes are semantically meaningful.

1 Introduction

Developing autonomous vehicles that can drive on public roads along with human drivers, pedestrians, cyclists, and other road users is a challenging task. Researchers have been attempting to solve this problem for many years, from the days of ALVINN [18] and the DARPA Urban Challenge [17, 23] to exploring a variety of approaches in recent years, including end-to-end learning [7] and traditional engineering stacks [15]. In order to drive both safely and comfortably in the real world, one of the most important and difficult tasks for self-driving vehicles (SDVs) is to predict the future behaviors of the surrounding road users.

There has been significant research dedicated to predicting the future states of traffic actors. One common line of research attempts to independently predict each actor's future trajectory from the scene [2, 8, 10, 13, 16, 24]. However, a limitation of many of these approaches is that they fail to capture the interactions among actors. For example, in the case shown in Figure 1, the future trajectory of the blue vehicle and the future trajectory of the pedestrian depend on one another. In this case, there are two possible outcomes of the interaction: the pedestrian yields to the vehicle, or, more likely, the vehicle yields to the pedestrian. Predicting the marginal future trajectory of the vehicle or the pedestrian will ignore the possible modes of interaction between the two actors, and will end up capturing an inaccurate distribution over the future trajectories.

Machine Learning for Autonomous Driving Workshop at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.



(a) Real image (b) Cartoon illustration

Figure 1: A traffic scene in which an SDV (white) approaches an intersection where another vehicle (blue) is interacting with a pedestrian (red). Failing to capture the interaction between the vehicle and the pedestrian when predicting their future behavior may lead the SDV to execute a poor motion plan.

Recently, there has been an increasing amount of work on modeling interaction in multi-agent systems with neural networks [1, 3, 9, 12, 14, 19, 20, 21, 22, 5]. For example, in CommNet [20], communication protocols (interactions) between smart agents are learned in conjunction with the final prediction outcomes through a Graph Neural Network (GNN), where the communication is learned implicitly. Similarly, in SocialLSTM [1], interaction between pedestrians is captured by the *social pooling* operation on the hidden states of the LSTMs. As opposed to modeling interaction implicitly, Neural Relational Inference [14] models the interactions in dynamical systems as latent edge types of an interaction graph, which are learned in an unsupervised manner.

Another related approach in the autonomous driving domain is IntentNet [6]. In this work, the model learns discrete actions, such as “keep lane” and “left lane change” using supervision. One limitation of predicting actions instead of interactions is that it is unnatural to pose constraints or priors on a pair of actor actions, but much easier to do so with interactions. An example of such a prior is illustrated in Figure 1, where we believe that if the pedestrian goes first, then the vehicle will yield, and vice versa. By introducing the concept of pair-wise interaction, we are able to capture the fact that in this interaction pair, it is unlikely that both the vehicle and the pedestrian will go at the same time and it is instead more likely that one actor will yield to the other. Importantly, by using the future observations of the scenario to categorize interaction types during training, we can learn these pair-wise interactions, instead of independent agent-wise actions, *explicitly* in a supervised manner.

In this paper, we propose a supervised learning framework for the joint prediction of interactions and trajectories. Specifically, we model interactions as intermediate discrete variables, which are learned from labeled examples, in order to capture the long-term relative intents of the actors. To achieve this, we introduce an interaction labeling function which uses simple heuristics to programmatically generate the labels from the future trajectories. This enables us to build a large dataset of vehicle interactions without relying on human experts for manual labeling. In addition to improving the accuracy of trajectory predictions, we show that explicitly modeling the interaction types helps capture the modes of vehicles’ future behaviors in an explainable manner. Our approach is empirically verified by experiments conducted on a large-scale dataset collected by real-world autonomous vehicles.

2 Problem Formulation

We are mainly interested in predicting the future dynamics of multi-agent systems consisting of vehicles. Our goal is to predict the future trajectories of vehicles in traffic scenes, given their observed states and some additional features describing the traffic conditions around them. In order to jointly model the dynamics of all agents in the system in a structured way, we introduce an auxiliary task of learning discrete interaction types between agents.

We first define the state $\mathbf{s}_t^i$ to be the 2D position and velocity of agent i at time t, and let $\mathbf{s}_{t_1:t_2}^i$ denote the sequence of agent i’s states from t_1 to t_2. For compactness, we further define $S_{t_1:t_2} = \{\mathbf{s}_{t_1:t_2}^i\}_{i=1:N}$ to be the state sequences of all N agents in the scene. Then, the trajectory prediction task is to predict the future states $S_{t:t+T}$ of all agents given observations of their past states $S_{t-T:t}$.

Next, we assume that there exist discrete types that summarize the modes of interaction between each pair of agents. Under this assumption, we introduce a secondary task of learning the interaction types from labeled examples. In traffic scenarios, it is often difficult to capture interactions based solely on the agents' dynamics, and additional contextual features about the the agents in the scene can be very informative. Let $\mathbf{a}_i \in \mathbb{R}^{F_a}$ and $\mathbf{p}_{ij} \in \mathbb{R}^{F_p}$ be agent-wise and pair-wise features that describe an individual agent (capturing basic traffic context) and the relationship between a pair of agents (capturing their relative dynamics via a compact set of high-level features), respectively. Then, the interaction prediction task is to predict the interaction label l_{ij} for each ordered pair of agents (i, j) given the agent-wise and pair-wise features along with the observed dynamics of the pair.

Since our primary goal is still trajectory prediction, we combine the predicted interaction types $\hat{l_{ij}}$ with information about the agents' past states and provide these as inputs to the trajectory prediction module. Explicitly capturing these interaction types guides the trajectory prediction module on how to aggregate information from agents $j \neq i$ when predicting the future behavior of agent i, which ultimately leads to more accurate predictions.

3 Method

We tackle the trajectory prediction problem by jointly learning to predict both interaction types and future trajectories. The key insight is that by learning interaction types and future dynamics jointly, a model can learn to make better and more explainable predictions.

Labeling Function. Supervised learning of interaction types requires labeled examples. Instead of obtaining interaction labels from human experts, we use simple heuristics to programmatically generate the labels, similar in philosophy to [25]. We extract an interaction label l_{ij} for each ordered pair of agents (i, j) in the scene at each timestep t. The label is determined by the future trajectories of the agents. Given the trajectories, the labeling function outputs: a) $l_{ij} = \texttt{IGNORING}$ if trajectories do not intersect, b) $l_{ij} = \texttt{GOING}$ if trajectories intersect and i arrives at the intersection point *before* j, and c) $l_{ij} = \texttt{YIELDING}$ if trajectories intersect and i arrives at the intersection point *after* j.

Graph Representation of Agent States. It is undesirable to use a global coordinate system in trajectory prediction tasks because of the high variability of the input and output coordinates [4]. Instead, we transform the trajectory waypoints of each agent to their individual reference frame. The current position of an agent at time t is set to be the origin of the agent's reference frame, and the coordinates of past and future trajectory points are offset by the agent's current position and rotated by the current heading of the agent in a global frame.

When the coordinates of the past and current states of each agent are transformed from global frame to that individual agent's coordinate frame, information about the relative positions and velocities among agents is lost. However, a model needs to be informed with the relative configuration of agents in order to reason about their interaction. To preserve the relationships among agents, we represent the configuration of agents at a given timestep as a state graph, in which a node represents the state of an individual agent in its reference frame, and a directed edge represents the relative state of the destination node in the source node's reference frame.

Modeling Interaction with Graph Network. We use a variant of Graph Network (GN) layers [3] to process the state graphs and model the interactions between agents. Our GN consists of two components: an edge model which combines the representations of each edge and its terminal nodes to output an updated edge representation, and a node model which operates on each node to aggregate the representations of incident edges and outputs an updated node representation. We model different types of interaction using a separate learnable function f_m for each type m in the edge model (see Appendix for details). Given the predicted scores of the interaction types between a pair of actors, the edge model computes the sum of the outputs from each f_m weighted by these scores.

Figure 2 describes the overall schematics of our joint prediction model. Our model consists of three components: 1) trajectory encoder network, 2) interaction prediction network, and 3) trajectory decoder network. First, the encoder network encodes the observed past states $\mathbf{s}^i_{\tau-T:\tau}$ of each agent into a hidden state h^i. Then, the interaction prediction network takes in the encoded states of all agents $H = \{h^i\}_{i=1:N}$, along with the agent-wise features $V_f = \{\mathbf{a}_i\}_{i=1:N}$ and pair-wise features $E_f = \{\mathbf{p}_{ij}\}_{i,j=1:N,j=1:N,i\neq j}$, and predicts the interaction type scores $\hat{L} = \{\hat{l}_{ij}\}_{i=1:N,j=1:N,i\neq j}$ for every ordered pair of agents, using a stack of two vanilla (untyped) GN layers. Finally, given the

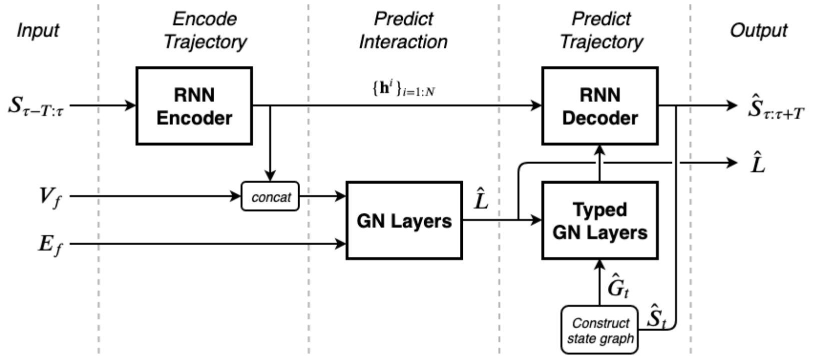


Figure 2: A joint model for interaction and trajectory prediction. See text in Section 3 for details.

Method	mean DPE	mean ATE	mean CTE
Baseline, no interaction	2.051	1.818	0.558
Graph, untyped, yielding/going edges only	1.725	1.511	0.512
Graph, untyped, all edges	1.713	1.491	0.523
Graph, oracle, yielding/going edges only	1.709	1.435	0.519
Graph, oracle, all edges	1.638	1.435	0.489
Graph, joint, supervised interaction	1.611	1.397	0.500
Graph, joint, unsupervised interaction	**1.579**	**1.378**	0.477
Trajectory prediction with map and scene context [11]	1.643	1.533	**0.334**

Table 1: Quantitative evaluation. We report the displacement error (DPE), along-track error (ATE), and cross-track error (CTE) in meters averaged over a 5-second horizon.

hidden states H, interaction type scores $\hat{L}$, and the initial states S_τ, the decoder network rolls out the future states of the agents $\hat{S}_{\tau:\tau+T}$. This module aggregates information from all actors using a stack of two typed GN layers, which employ an MLP for each learned function f_m in the edge model.

Loss Function. The loss function we use is the combination of a classification loss over the edges for predicting the discrete interaction types (edge loss) and a regression loss over the nodes for predicting the continuous future trajectories (node loss). Our complete loss function is given by

$$L = \alpha\,\mathrm{CE}(L, \hat{L}) + \mathrm{MSE}(S_{\tau:\tau+T}, \hat{S}_{\tau:\tau+T})$$

where CE() is the Cross-Entropy loss and MSE() is the Mean-Squared-Error loss. For the supervised interaction model, we set $\alpha = 5$, and for the unsupervised interaction model, we set $\alpha = 0$.

4 Experiments

We present experiments on real-world traffic data collected by autonomous vehicles which operated in numerous locations across North America. The dataset contains trajectories of 68,878 unique vehicles in various traffic scenarios. The vehicles were tracked continuously with a sampling frequency of 2 Hz (0.5s interval). We sample trajectories in sliding time windows of 10 seconds, and use the first 5 seconds as inputs and the last 5 seconds as prediction targets. Using the extracted trajectories, we run our labeling function to obtain the labels for every pair of agents that are less than 100 meters apart from each other. To evaluate the performance of our models, we report the mean displacement error, cross-track error, and along-track error between the estimated and ground-truth trajectories. We present an ablation study to analyze the capability of our proposed method to capture interactions between agents. The quantitative results are summarized in Table 1.

Our baseline model is an RNN encoder and decoder, which treats trajectories independently without modeling interactions between agents. In addition, we introduce two variants of our joint model. The first variant (*untyped*) has a single edge function to learn interaction without differentiating between types. The second variant (*oracle*) is modified to use ground-truth interaction types, instead of its own predictions, to predict trajectories. We also modify each variant to exclude the edges with `IGNORING` labels from the graph in order to see if these edges can be ignored as the name suggests.

We first demonstrate the power of graph networks to model interaction by comparing our joint model and its variants against the baseline. We observe that all of the graph models significantly outperform the baseline. This suggests that the motion of vehicles is highly interdependent, and graph models can effectively capture their interactions. Next, we showcase the effect of interaction labels on trajectory prediction. The typed variants outperform all of the untyped variants, which suggests that our graph model benefits from the discrete modeling of interaction types. Furthermore, we can see that the typed model benefits from having information shared along the `IGNORING` edges.

Finally, we present the results of our full fledged joint prediction model. Even without rich map context, our model shows comparable performance with [11], particularly on along-track error, which captures the temporal accuracy of the predicted trajectories. Notably, another version of the joint model trained without supervision on interaction labels (simply by zeroing out the interaction classification loss) achieves better performance than the supervised model. This implies that the heuristics used in our labeling function are not optimal, and could be improved for better trajectory prediction. Nonetheless, we observe via simulation experiments that the supervised model predicts trajectories that are consistent with the meanings of the interaction labels (see Appendix and supplementary video for details). This interpretability helps provide key insights into the model's behavior, which is a crucial step towards building safe prediction systems for autonomous vehicles.

5 Conclusion

In this paper, we propose a graph-based model for multi-agent trajectory and interaction prediction, which explicitly models discrete interaction types using programmatically generated weak labels and typed edge models. The main advantages of our approach are: i) we can gain a boost in performance without additional labeling costs when compared to the baseline, and ii) our model can effectively capture the multi-modal behavior of interacting agents while learning semantically meaningful interaction modes.

References

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] F. Altché and A. de La Fortelle. An lstm network for highway trajectory prediction. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 353–359. IEEE, 2017.

[3] P. Battaglia, J. B. C. Hamrick, V. Bapst, A. Sanchez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. Allen, C. Nash, V. J. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv*, 2018.

[4] S. Becker, R. Hug, W. Hübner, and M. Arens. Red: A simple but effective baseline predictor for the TrajNet benchmark. In *Computer Vision – ECCV 2018 Workshops*, 2019.

[5] S. Casas, C. Gulino, R. Liao, and R. Urtasun. Spatially-aware graph neural networks for relational behavior forecasting from sensor data. *arXiv preprint arXiv:1910.08233*, 2019.

[6] S. Casas, W. Luo, and R. Urtasun. IntentNet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956, 2018.

[7] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. DeepDriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

[8] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019.

[9] N. Deo and M. M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1549–15498, 2018.

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

[11] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, and J. Schneider. Motion prediction of traffic actors for autonomous driving using deep convolutional networks. *arXiv preprint arXiv:1808.05819*, 2018.

[12] Y. Hoshen. Vain: Attentional multi-agent predictive modeling. In *Neural Information Processing Systems*, 2017.

[13] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 399–404. IEEE, 2017.

[14] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel. Neural relational inference for interacting systems. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[15] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 163–168. IEEE, 2011.

[16] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.

[17] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, et al. Junior: The Stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[18] D. A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[19] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. *arXiv preprint arXiv:1905.01296*, 2019.

[20] S. Sukhbaatar, R. Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016.

[21] C. Sun, P. Karlsson, J. Wu, J. B. Tenenbaum, and K. Murphy. Predicting the present and future states of multi-agent systems from partially-observed visual data. In *International Conference on Learning Representations*, 2019.

[22] A. Tacchetti, H. F. Song, P. A. M. Mediano, V. Zambaldi, J. Kramár, N. C. Rabinowitz, T. Graepel, M. Botvinick, and P. W. Battaglia. Relational forward models for multi-agent learning. In *International Conference on Learning Representations*, 2019.

[23] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[24] G. Xie, H. Gao, L. Qian, B. Huang, K. Li, and J. Wang. Vehicle trajectory prediction by integrating physics-and maneuver-based approaches using interactive multiple models. *IEEE Transactions on Industrial Electronics*, 65(7):5999–6008, 2017.

[25] E. Zhan, S. Zheng, Y. Yue, L. Sha, and P. Lucey. Generating multi-agent trajectories using programmatic weak supervision. In *International Conference on Learning Representations*, 2019.

6 Appendix

6.1 Qualitative Evaluation on Simulated Data

In order to understand the influence of the edge types on the final trajectory predictions generated by our model, we simulated several very simple dynamic interactions between two actors. We generated simulated historical states for the actors and then injected fixed values for the edge scores to analyze how the predicted trajectories change as a function of the injected edge types. We visualized the predicted trajectories from the baseline model, oracle model with all edges, supervised joint model, and unsupervised joint model while injecting several different combinations of interaction types. We compiled the results of these simulations into a short video, which is available at `https://www.youtube.com/watch?v=n5RNRDdoPoQ`.

In these simulations, we find that all of the graph models learn to rely heavily on the edge type in order to predict the future trajectories of the two actors. We can effectively control how the interaction plays out simply by injecting different edge types (e.g., yielding/going vs. going/yielding). The interaction modes of the oracle and supervised models correspond directly to the labeled categories that we provide. Interestingly, the interaction modes learned by the unsupervised model encode a similar set of categories, but seem to capture the leading/following relationship separately from the going/yielding relationship.

6.2 Qualitative Evaluation on Real Data

Next, we looked at some examples of real-world scenes and qualitatively evaluated our model's performance on these cases. Table 2 illustrates some examples of trajectory predictions for actors driving in three-way and four-way intersections. The trajectories are predicted from time t to $t + H$, where H is the prediction horizon. The different colors indicate trajectory predictions for different actors. Each dot shows a single predicted waypoint (at 0.5-second intervals), and the more transparent the dot is, the further away it is in the future (i.e., the further its timestamp is from the current time, t).

We observe several interesting patterns in these examples. First, note that the map in the figures is purely for illustration – we do not provide map information directly to the model. Nevertheless, the graph models are able to learn the lane directions and drivable surfaces to some degree by observing the histories of the other vehicles. Second, we notice that the model that uses only the `YIELDING/GOING` edges (second column) is substantially worse at capturing lane-following behavior than all other models (see rows (a), (b), (c), and (e) for examples), suggesting that the `IGNORING` edges are useful for transmitting implicit map information from actor to actor. Third, if we compare the supervised and unsupervised models (last two columns), we observe that the unsupervised model is slightly worse at predicting lane-following behavior than the supervised model (see rows (c), (d), and (e) for examples). We also notice that the supervised model appears to predict fewer conflicts between trajectories than the unsupervised model (an example can be seen in row (b), where the supervised model clearly predicts the yellow actor to yield to the blue actor, but the unsupervised model predicts that both will go at the same time). Lastly, we see that in case (e), the red merging actor may be equally likely to turn left or right, and because the current model is uni-modal (i.e., it only predicts the single most likely future trajectory), it is not able to model such discrete modes. This suggests two future works: (1) incorporating map elements (traffic signals, traffic signs, lane segments, sidewalks) as nodes in the graph; (2) adding multi-modality to the model.

6.3 Typed Graph Network Architecture

Here, we describe a variant of Graph Networks [3] used in our model architecture. A Graph Network (GN) layer propagates information between the nodes and edges to output a new graph with updated representations for each node and each edge. Following the notation in [3], a graph $G = (V, E)$ is defined by nodes $V = \{\mathbf{v_i}\}_{i=1:N^v}$ and directed edges $E = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1:N^e}$, where $\mathbf{v}_i \in \mathbb{R}^{d^v}$ and $\mathbf{e}_k \in \mathbb{R}^{d^e}$ are node and edge attributes. We extend the original formulation by defining edges with discrete types and an update function for the typed edges. Assuming M distinct edge types, let $l_k = (l_{k,1}, ..., l_{k,M})$ be the one-hot encoding or the scores of the types for edge k. Then, the typed edge update function outputs updated edge attribute $\mathbf{e}'_k = \sum_{m=1}^{M} l_{k,m} \cdot f_m(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k})$, where f_m is a learnable function for each edge type m. Additional details can be found in Figure 3.

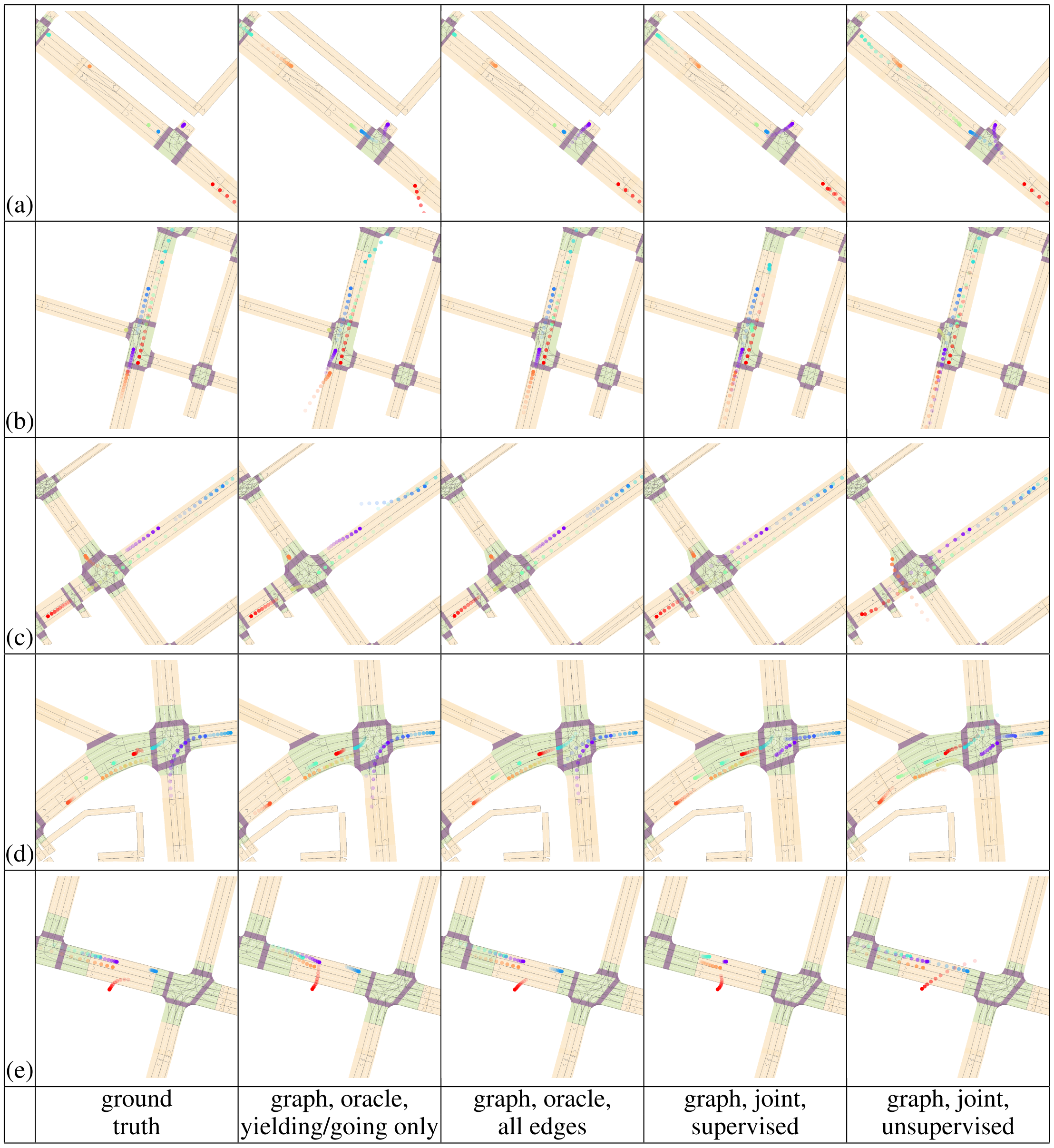
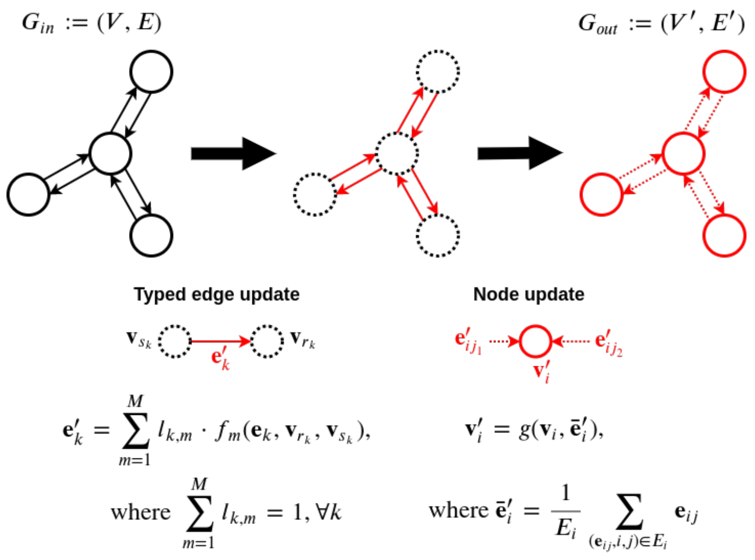


Table 2: Examples of trajectory predictions on real-world driving scenarios, from time t to $t + 5s$: (a) 3-way intersection; (b) unconventional 4-way intersection; (c) canonical 4-way intersection; (d) 5-way intersection; (e) actor merging from outside the road network. The different colors indicate trajectory predictions for different actors. Each dot shows a single predicted waypoint (spaced at 0.5-second intervals), and the more transparent the dot is, the further away it is in the future (i.e., the further its timestamp is from the current time, t).

6.4 Additional Quantitative Comparison

In Figure 4, we further compare our supervised joint model to the baseline from [11] by measuring the trajectory errors at different time horizons (1 second, 3 seconds, 5 seconds). The results indicate that our approach is worse than the baseline in terms of cross-track error, which is expected because [11] provides a rasterized bird's eye view of the map as an input to the model, and we don't have the same map and scene context. However, we also see that our method is better than the baseline in terms of along-track error, which highlights the value of explicitly capturing temporal interactions such as going and yielding for the trajectory prediction problem.



$G_{in} := (V, E)$
$G_{out} := (V', E')$
Typed edge update
$\mathbf{v}_{s_k}$
$\mathbf{e}'_k$
$\mathbf{v}_{r_k}$
$\mathbf{e}'_k = \sum_{m=1}^{M} l_{k,m} \cdot f_m(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}),$
where $\sum_{m=1}^{M} l_{k,m} = 1, \forall k$
Node update
$\mathbf{e}'_{ij_1}$
$\mathbf{e}'_{ij_2}$
$\mathbf{v}'_i$
$\mathbf{v}'_i = g(\mathbf{v}_i, \bar{\mathbf{e}}'_i),$
where $\bar{\mathbf{e}}'_i = \frac{1}{E_i} \sum_{(\mathbf{e}_{ij}, i, j) \in E_i} \mathbf{e}_{ij}$

Figure 3: Details of the GN layer updates with typed edges. The update function takes in $G_{in} = (V, E)$, which is an input representation for each node and each edge, and outputs $G_{out} = (V', E')$, which is the updated representation for each node and each edge. First, each edge representation is updated by processing the node attributes of its endpoints with a type-specific function f_m. Then, each node representation is updated by aggregating the attributes of the incoming edges.

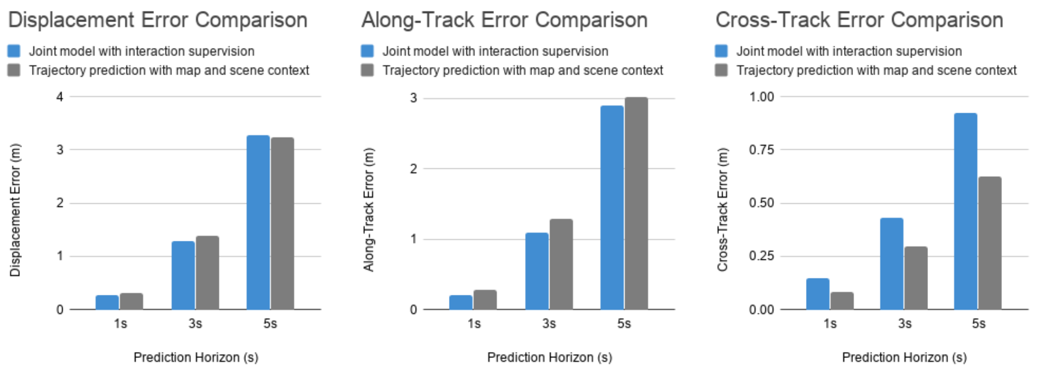


Figure 4: A comparison of the trajectory prediction error between our proposed supervised joint model (shown in blue) and the baseline from [11] (shown in gray). From left to right, the plots show the displacement error, along-track error, and cross-track error at multiple different time horizons. All errors are reported in meters. We see that our approach is comparable in terms of displacement error, better in terms of along-track error, and worse in terms of cross-track error.