# Multi-Agent Trajectory Prediction by Combining Egocentric and Allocentric Views

**Anonymous Author(s)**
Affiliation
Address
email

**Abstract:** Trajectory prediction of road participants such as vehicles and pedestrians is crucial for autonomous driving. Recently, graph neural network (GNN) is widely adopted to capture the social interactions among the agents. Many GNN-based models formulate the prediction task as a single-agent prediction problem where multiple inference is needed for multi-agent prediction (which is common in practice), which leads to fundamental inconsistency in terms of homotopy as well as inefficiency for the memory and time. Moreover, even for models that do perform joint prediction, typically one centric agent is selected and all other agents' information is normalized based on that. Such centric-only normalization leads to asymmetric encoding of different agents in GNN, which might harm its performance. In this work, we propose a efficient multi-agent prediction framework that can predict all agents' trajectories jointly by normalizing and processing all agents' information symmetrically and homogeneously with combined egocentirc and allocentric views. Experiments are conducted on two interaction-rich behavior datasets: INTERACTION (vehicles) and TrajNet++ (pedestrian). The results show that the proposed framework can significantly boost the inference speed of the GNN-based model for multi-agent prediction and achieve better performance. In the INTERACTION dataset's challenge, the proposed model achieved the 1st place in the regular track and generalization track.

**Keywords:** Autonomous Driving, Joint Trajectory Prediction, Multi-Agent Interaction

## 1 Introduction

Trajectory prediction of road participants like vehicles and pedestrians is of great significance for planning and decision making of autonomous vehicles. Recently, lots of research efforts have been devoted to capture the social interactions among the agents with feature pooling [1, 2] or graph neural network [3, 4, 5]. However, most of the state-of-the-art (SOTA) models [5, 6, 7] formulate the problem as single-agent prediction, i.e., the models try to learn the marginal distribution of what a target agent may act in the future given the historical observations of all agents' trajectories in the scene. In cases of multiple agent prediction, a straightforward way is to run inference multiple times [8], that is, onece for each agent. Under such circumstances, the "joint" prediction results might not be consistent in terms of homotopy, not to mention the low efficiency in terms of memory and time. There are also models which perform joint prediction for multiple agents [9, 10]. However, such work adopts a uniform coordinate for all modeled agents, i.e., a centric agent is selected and all coordinates are normalized based on the centric agent. Therefore, only the selected agent is providing egocentric views, while all other agents are in allocentric views.[1] The disadvantage of all

[1] Egocentric and allocentric are terms usually used in the psychology and cognition area. According to [11], a study on human spatial cognition and animal navigation, they define the egocentric spatial coding system as the location of objects in space relative to the ego body and the allocentric spatial coding system as the location of one object is defined with respect to other objects. In this paper, we refer this two concepts. As for the coordinate system transformation, we set the reference's coordinate as the origin and yaw angle aligned with the negative x-axis.

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

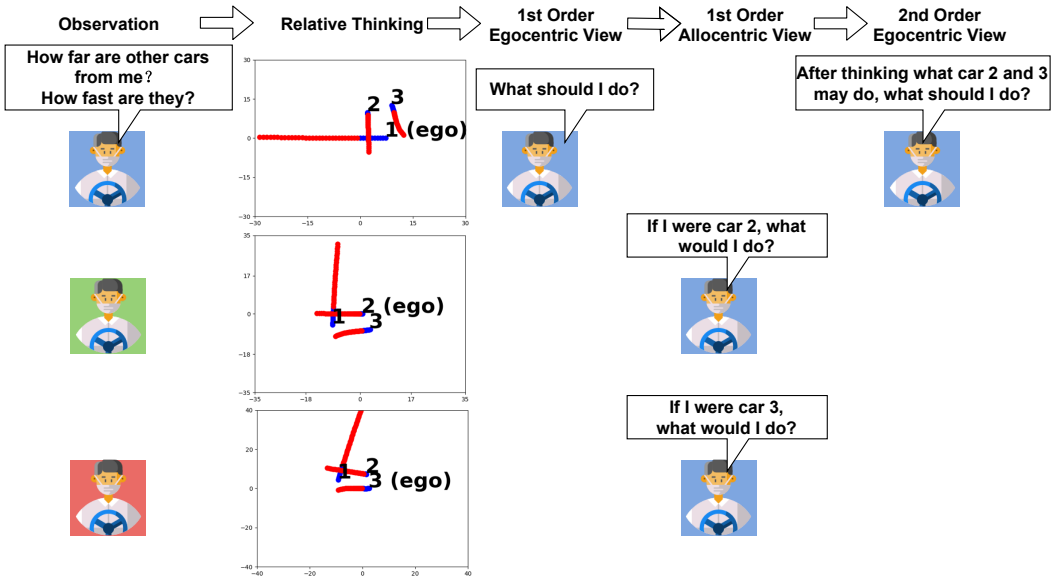



Figure 1: Illustration of the intuition for each stage of the proposed model - the thinking process of interacting agents. Here, we only show the first agent's (blue) thinking process since the other two are symmetrical. First, a human driver understands the world by estimating the relative distance and speed of other agents (Relative Thinking), which corresponds to the normalization stage of the proposed model. Then each agent considers how it should react (Egocentric View), which corresponds to the stage of updating the node features by aggregating neighbors' information in the proposed model. After that, an agent put itself in someone else's shoes for all neighbors (Allocentric View), which corresponds to the update of edge features by accessing source node's information in the proposed model. After alternatively repeating the Egocentric View stage and Allocentric View stage for several rounds, all agents make the decision according to the interacting situations.

data in a single egocentric view is two-fold. First, not all agents are treated in an unbiased way due to the lack of egocentric views from other agents, like missing the view of "putting in someone else's shoes". Second, the egocentric view of one agent makes the spatial relations between two other agents heterogeneous from the spatial relations between ego agent and other agents. The different distributions of input node feature might harm the GNN-based models' performance since it usually shares parameter for the aggregation and fusion function for all nodes.

In this work, we aim to design an efficient GNN-based multi-agent joint prediction model. Our key insight is that all agents should be treated symmetrically in the graph, providing both egocentric and allocentric views. Moreover, we decouple the update of the node and edge feature to improve the efficiency of the network. As a result, we propose to use the "Node update" step to collect and summarize other agents' information in a egocentric view, and then the "Edge update" step to collect information in allocentric views from each neighbor. Via such a combined egocentric-and-allocentric views, it is guaranteed that the graph model handles all agents homogeneously which will encourage the shared parameters to learn to extract the underlying interactions among agents. We illustrate the intuition behind the proposed model in Fig. 1

We conducted experiments on two interaction-rich datasets: INTERACTION (vehicles) [12] and TrajNet++ (pedestrian) [13]. The results show that compared to running multiple single prediction models, the efficiency is boosted 1.7-4.5 times higher while achieving better accuracy. In the INTERACTION dataset's challenge, the model achieved the $\mathbf{1^{st}}$ **place** in the regular track and generalization track [2] We have also conducted experiments on three efficient variants of the multi-inference single-prediction model under different view settings and the results show that the proposed model outperforms those variants, which demonstrates the advantage of combining the egocentric and allocentric views.

[2]The generalization track is testing the model's performance on new maps/scenarios which have never been seen in the training time.

## 2 Related Works

Plenty of works have been done to capture the complex social interactions among multiple agents. There are pooling-based methods [1, 2] which use the mean/max/sum of all agents' hidden vectors to obtain a single vector to represent the social context. Social LSTM [1] utilized the social context of each time step as a part of the input for its next time step to the LSTM. Social GAN [2] obtained the social context by the pooling of output from the encoder RNN at the last time-step and fed the social context with each agents' hidden vector to the decoder to generate their future trajectories. Though pooling is fast, it loses too much information and lacks of expressive ability to output a specific context for each agent. There are also methods which renders the entire scene including the agents and HD Map onto images and captures the interactions implicitly by CNN. Hong et al. [14] adopted pure CNN to process the scene while Multipath [6] further set anchors to output multi-modal prediction and IntentNet [15] set a different motion predictor for different predefined intents. PRECOG [9] proposed a flow-based generative models conditioned on goals with CNN to process context. However, as pointed [4], those CNN-based methods' performances are significantly influenced by the receptive field, feature cropping strategy, and image resolution and it is very computationally expensive and inefficient compared to directly process the coordinate data. TPCN [7] conducted trajectory prediction from point-cloud perspective which is an interesting exploration across fields. Recently, GNN-based [3, 16, 17, 4, 18, 6, 19, 5] has gained great success and dominated most of the trajectory prediction datasets. Many of them [16, 6, 19, 5] set an ego agent and used it to normalize data. In the concurrently released Waymo Open Motion Dataset's paper [8], they implemented TNT [6] by the forward-N-times way to do multi-agent prediction. In ILVM[20], they proposed an end-to-end joint prediction framework, in which they first used CNN to extract features from LiDAR point clouds and raster map, used GNN to capture interactions, and used a novel one stage parallel sampling to do joint prediction. As for normalization, they used Rotated Region of Interest Align[21], which is specificlly designed for LiDAR point clouds/images to align. In the concurrent work Scene Transformer[22], they used Transformer for both spatial and temporal dimension, adopted mask strategies and done joint prediction. They normalized data simply by the ego agent.

## 3 Problem Formulation

Suppose there are $N$ agents in the same scene (they could be either vehicles or pedestrians), and our prediction task is to learn the joint distribution of their future trajectories conditioned on their historical states. Let us assume that the current time step is $t = 0$. For each agent $i = \{1, \cdots, N\}$, we have an observation of his/her states (such as coordinate, velocity, yaw angle, ...) from $t = -L$ to $t = -1$. We denote the joint observation as $\boldsymbol{X} = \{X_1, X_2, ..., X_N\}$ where $X_i = \{\boldsymbol{x}_i^{-L}, ..., \boldsymbol{x}_i^0\}$. Let us also define the future coordinates for all $N$ agents from $t = 1$ to $t = T$ as $\boldsymbol{Y} = \{Y_1, Y_2, ..., Y_N\}$ where $Y_i = \{\boldsymbol{y}_i^1, ..., \boldsymbol{y}_i^T\}$. Then, the joint prediction model is designed to learn the following conditional distribution:

$$p(Y_1, Y_2, ..., Y_N | X_1, X_2, ...X_N) \tag{1}$$

On the other hand, the single-prediction model is approximating a marginal distribution defined as follows:

$$p(Y_i | X_1, X_2, ...X_N) \quad \text{where agent i is the ego agent.} \tag{2}$$

In the cases of multiple-agent prediction, essentially, multiple-inference is using multiple marginal distributions to approximate the joint distribution by implicitly assuming the multiple marginal distributions are conditionally independent [8]:

$$p(Y_1, Y_2, ..., Y_N | X_1, X_2, ...X_N) = \prod_{i=1}^{N} p(Y_i | X_1, X_2, ...X_N) \tag{3}$$

Unfortunately, such assumption is not true in practice, particularly when there are intensive interactions among the $N$ agents.

## 4 The Model

[3]Note that the Edge MLP in the last layer could be removed to reduce computation (in this case - Edge MLP 2)

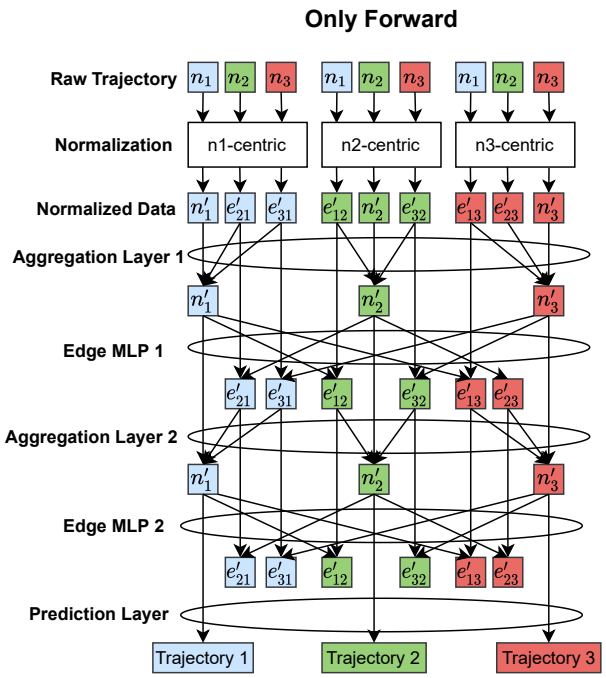



Figure 2: The structure of the proposed model: we use 3 agents as an example. $n_1, n_2, n_3$ means raw data. The superscript $\prime$ means the data after normalization. $e_{ij}$ means the normalized data of agent $i$ by the origin and orientation of agent $j$. First, all agents are encoded in their egocentric views in the normalization step; second, the node features (Aggregation Layer) are updated by the egocentric information separately for different agents; and finally, the edge features (Edge MLP) are updated using allocentric views, summarizing local information from all agents.[3]

An overview of the model is shown in Fig. 2. The model has the following three key modules:

**View Transformation (Normalization Layer):** All agents are encoded symmetrically in the graph. In the normalization layer, we set each agent in his/her own egocentric view instead of selecting one centric agent and normalize all other agents based on it. Specifically, for the coordinate system of each node (agent) and its in-edges' (neighbors), we use the agents' coordinate at $t_0$ (w.r.o.g) as the origin and let the yaw angle be aligned with the negative x-axis. Since the input node and edge features are temporal sequences, we use a temporal sequence encoder (ResNet-18 1D version+MLP) to encode the sequence into a feature vector.

**Node update (Aggregation Layer):** Update each node feature according to their in-edges' features to incorporate neighbor's information $h'_i = f(\{h_{ji}, j \in \text{Neighbor(i)}\})$ where $h'_i$ is the updated node feature of agent i, $h_{ji}$ is the feature of in-edge from agent j to i, $f$ is the aggregate function and it is multi-head attention in Transformer. Note that for conciseness we added self-loop for each node which means each nodes' neighbors include itself. Since the input node feature and input in-edge features of each agent are both normalized in each agent' coordinate system, it is in the egocentric view.

**Edge update (Edge MLP):** Update each edge by their source node's feature to obtain updated neighbors' information $h'_{ji} = g(h_{ji}, h'_j)$ where we denote the concatenation+FeedForwardNet in Transformer as $g$. Since the new information is coming from the source node - the neighbor in the neighbor's view, it is in the allocentric view.

We would like to emphasize the benefit of decoupling the node update step and the edge update step. By decoupling, the node serve as the **storage** for the interaction information between each agent and its neighbors in the egocentric view. Via such design, the edge feature can be efficiently updated to collect and summarize the allocentric information. Hence, such a combined egocentric-

Table 1: Comparison with SOTA methods on the INTERACTION Dataset. - means the results are not publicly available. * means the model is trained by smaller gap between snippets which means more data, which is proposed by the challenge winner ReCog [25]. . The unit of ADE and FDE are both meters. ↓ means the lower the better.

| | Val ADE/FDE↓ | Test ADE/FDE↓ | Generalization ADE/FDE↓ |
|---|---|---|---|
| DESIRE [23] | 0.32/0.88 | - | - |
| MultiPath [24] | 0.30/0.99 | - | - |
| TNT [6] | 0.21/0.67 | - | - |
| MIFNet | - | 0.1973/0.6641 | 0.5339/1.4248 ($\mathbf{2^{nd}}$) |
| ReCog* [25] | 0.1919/0.6462 | 0.1878/0.6381 ($\mathbf{3^{rd}}$) | 0.5539/1.9187 |
| Mix | - | 0.1826($\mathbf{1^{st}}$)/0.6423 ($\mathbf{2^{nd}}$) | - |
| Ours | 0.1723/0.5988 | 0.1903/0.6563 | 0.3394/1.1983 |
| Ours* | 0.1700/0.5927 | 0.1840($\mathbf{2^{nd}}$)/0.6344 ($\mathbf{1^{st}}$) | 0.3263/1.1426 ($\mathbf{1^{st}}$) |

and-allocentric encoding and feature updating can help to 1) achieve an unbiased encoding of all agents for joint prediction, and 2) improve the efficiency of the computing process.

# 5 Experiements

## 5.1 Metrics

As for the metric, we adopt the widely used ADE and FDE. They are defined as: Average Displacement Error (ADE) represents the euclidean distance averaged over time and agents between the ground truth and the prediction. It is then averaged over all cases. The ADE of a single case is calculated as:

$$\text{ADE} = \frac{1}{NT}\sum_{n,t}\sqrt{(\hat{x}_{n,t} - x_{n,t})^2 + (\hat{y}_{n,t} - y_{n,t})^2} \tag{4}$$

where N is the number of target agents in the case, T ist the prediction horizon, $\hat{x}$ and $\hat{y}$ represent the groud truth coordinate, x and y represent the predicted coordinate. The unit is meter.

Final Displacement Error (FDE) represents the euclidean distance averaged over agents between the ground truth and the prediction at the last predicted time-step. It is then averaged over all cases. The FDE of a single case is calculated as:

$$\text{FDE} = \frac{1}{N}\sum_{n}\sqrt{(\hat{x}_{n,T} - x_{n,T})^2 + (\hat{y}_{n,T} - y_{n,T})^2} \tag{5}$$

where N is the number of target agents in the case, T ist the prediction horizon, $\hat{x}$ and $\hat{y}$ represent the groud truth coordinate, x and y represent the predicted coordinate. The unit is meter.

## 5.2 Results

We tested the performance of the proposed model against several SOTA models on two interaction-rich dataset: the INTERACTION dataset and the TrajNet++. Please refer to Appendix D for experimental setting details.

In INTERACTION dataset [12], we compared with DESIRE [23], MultiPath [24], TNT [6], and ReCog [25]. The former three are compared in [6]. We also listed the results of the top-rank methods on the leader board of the INTERACTION dataset. As we can see Tab. 1, our method outperforms the baseline methods, especially on the generalization ability track. We achieved the $\mathbf{1^{st}}$ **place** on both the regular test track and generalization ability track of the INTERACTION dataset.

In the TrajNet++ [13], existing results in this dataset are all measured only by the performance on the pre-defined 'interesting' primary pedestrian ignoring other agents. Thus, for fair comparisons, we re-implemented the two top-ranked methods under our multi-agent prediction setting: 1. SocialLSTM [1], a typical pooling-based method and the top-ranked method on TrajNet++'s leader board, is a good comparison for our GNN-based method. 2. D-LSTM [13], which was proposed in the TrajNet++ dataset paper and achieved SOTA performance in their paper, captures social interactions by concatenating top-4 nearest neighbors' hidden states. Note that [13] stated in their paper

that predicting pedestrian trajectories other than the primary pedestrian's would dampen their results which aligns well with ourfollowing for the single-view model. On the other hand, the GNN-based methods should be further improved to tackle high-density data like CFF dataset, which can be handled by pooling-based or top-k-neighbor-concatenation-based method due to the low computational complexity.

Table 2: Comparison with top-ranked methods on the TrajNet++ dataset under our multi-agent prediction setting. The unit of ADE and FDE are both meters. ↓ means the lower the better.

| | Val ADE/FDE ↓ |
|---|---|
| Social LSTM [1] | 0.2848/0.5604 |
| D-LSTM [13] | 0.2915/0.5738 |
| Ours | 0.2079/0.4270 |

# 6 Comparison with Variants of the Proposed Model

In this section, we compare the the proposed model against forwarding the single-prediction models N times (FNT) and its three variants in temrs of running speed and performance. By theoretical analysis and experimental results, we will show that the proposed model could achieve the best efficiency and accuracy among several settings of GNN for the multi-agent prediction.

**Basic Setting:** For simplicity, we use the fully-connected graph as an example in this section. Note that similar conclusions could be drawn when it comes to the non-fully-connected version. To make a fair comparison on the GNN setting, for other modules of the framework, we use the same configuration as the proposed model in all the following experiments. Specifically, we use Resnet-18 1D version+MLP as the temporal encoder for the input sequences of both node and edge. In the aggregation layers, we use Transformer (multi-head-attetnion+feed-forward-network) [26]. In the prediction layer, for each agent, we use a shared MLP to output their future trajectories based on each one's output node feature from GNN.

## 6.1 Baseline - forwarding $N$ times (FNT)

For a fully connected graphs with $N$ agents, there are $\frac{N(N-1)}{2}$ edges, which means each GNN layer has $\frac{N(N-1)}{2}$ message passing and $N$ updating. Thus, the computational complexity of $L$ layers of GNN in one forward is $L * (O(\frac{N(N-1)}{2}) + O(N)) = O(N^2)$ since L is a hyperparameter and we could treat it as a constant. As a result, in Equ. 6, we derive the computational complexity of FNT with GNN-based model on the fully-connected graph.

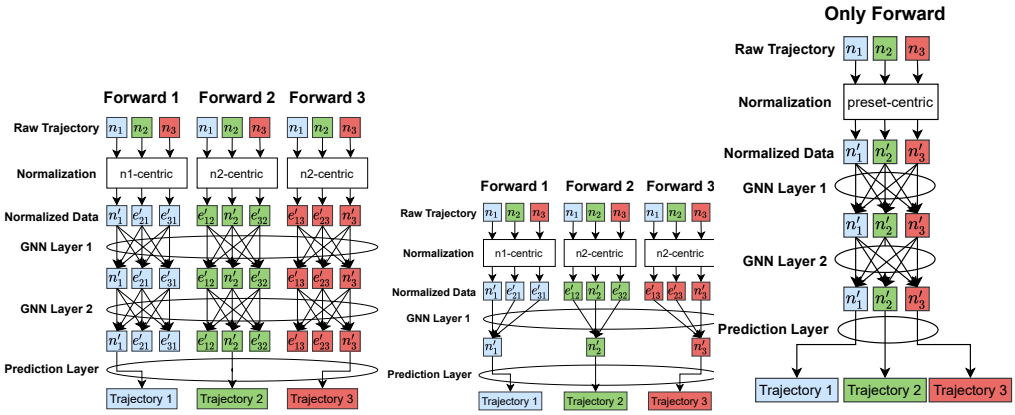
$$\underbrace{O(N)}_{\text{Number of Forwards}} * \underbrace{O(N^2)}_{\text{Complexity of GNN on Fully Connected Graph}} = O(N^3) \tag{6}$$

In contrast, as shown in Fig. 2, for the proposed model, the computational complexity of the two consecutive steps (Aggregation Layer and Edge MLP) both are $O(N^2)$ which makes the overall complexity $O(N^2)$.

**Represent FNT by a single graph:** The incorporation of neighbors' information in GNN essentially is: each target node obtains source nodes' information through their in-edges. Thus, FNT is equivalent to: each agent transforms its own states and others' states into its ego-centric coordinate system by normalization. Its own transformed states could be seen as a node while others' transformed states could be seen as its in-edges. Then, for each agent, it maintains and updates both its node feature and in-edge features in GNN. Finally, it uses its node feature to do the trajectory prediction. Fig. 3a gives the illustration of above description.

Table 3: Comparison between the proposed method and forward-N-times method. The unit of ADE and FDE are both the meter. The unit of Train/Val time is the second. ↓ means the lower the better.

| | INTERACTION | | TrajNet++ | |
|---|---|---|---|---|
| | Val ADE/FDE↓ | Train/Val Time↓ | Val ADE/FDE↓ | Train/Val Time↓ |
| FNT | 0.1840/0.6231 | 1261s/50s | 0.2328/0.4659 | 762s/23s |
| Ours | 0.1723/0.5988 | 713S/27s | 0.2079/0.4270 | 175s/10s |




(a) Illustration of computational process of forward-N-times (FNT).

(b) Illustration of computational process of one-layer GNN (OL).

(c) Illustration of computational process of vairant II. The preset-centric normalization could be either average of all agents (AA) or randomly selection (RA).

Figure 3:

After analyzing the computational process, we could find that **the redundant computing part in the FNT is: for each agent, it needs to maintain the information between any of two agents instead of only keeping the directly related information.**

Here, we conduct comparison experiments between FNT and the proposed methods. Experiments results are in Tab. 3. As we can see, compared to FNT, our method could significantly reduce the computational time while achieving better results.

### 6.2 Variant I - stop maintaining the edge feature

After finding out the abundance, we explore some other variants which could reduce the computational complexity of multi-agent prediction to $O(n^2)$ like the the proposed model and evaluate their precision.

Since the abundance is on the edges, the first intuitive variant is: **how about we stopping maintaining the edge feature and only update the node feature?** In this variant, there is no multi-hop or multi-order information included. For example, for agent 1, it could not obtain the interaction information between agent 2 and 3. Thus, this variant is equivalent to the one-layer GNN which only updates the node. We denote it as OL (one-layer). The illustration is in Fig. 3b. From the figure, we can obtain the computational complexity of OL in Equ. 7.

$$\underbrace{O(N)}_{\text{Number of Forwards}} * \underbrace{O(N)}_{\text{Ego Agnet Uses Its Neighbors' Features}} = O(N^2) \tag{7}$$

However, there might be two problems in one-layer GNN: 1: limited expressive power and hard to capture complex interactions. 2: for non-fully connected graph, lack of multi-hop neighbors' information.

**Validation Experiments** Here, we compare the results of GNN with one layer and two layers to evaluate the OL's influence on the models' performance. We compare the following settings:

- FNT: 2 layers GNN + fully-connected graph
- OL: 1 layer GNN + fully-connected graph
- FNT-Graph: 2 layers GNN + connected by distance threshold graph
- OL-Graph: 1 layer GNN + connected by distance threshold graph

For more experiment details, please refer to Appendix D.

The results are in Tab. 4. We can draw the conclusions that 1. using fully-connected graph could obtain better results. Thus, we use fully-connected graph for the rest of experiments in the paper. 2. OL consistently performs worse than FNT.

Table 4: Validation experiments for the one-layer GNN(OL) refinement. The unit of ADE and FDE are both the meter. The unit of Train/Val time is the second. ↓ means the lower the better.

| ADE/FDE | INTERACTION Val↓ | TrajNet++ Val↓ |
|---|---|---|
| FNT | 0.1840/0.6231 | 0.2328/0.4659 |
| OL | 0.1908/0.6379 | 0.2337/0.4753 |
| FTN-Graph | 0.1909/0.6704 | 0.2334/0.4693 |
| OL-Graph | 0.1991/0.6864 | 0.2343/0.4805 |

Table 5: Validation experiments for the Average of Agents(AA) and Random Agent (RA) plan. The unit of ADE and FDE are both the meter. The unit of Train/Val time is the second. ↓ means the lower the better.

| | INTERACTION | | TrajNet++ | |
|---|---|---|---|---|
| | Val ADE/FDE↓ | Train/Val Time↓ | Val ADE/FDE↓ | Train/Val Time↓ |
| FNT | 0.1840/0.6231 | 1261s/50s | 0.2328/0.4659 | 762s/23s |
| RA | 0.1938/0.6474 | 1308s/51s | 0.2483/0.4833 | 776s/24s |
| AA | 0.2187/0.7189 | 642s/21s | 0.2382/0.4762 | 141s/6s |

In conclusion, **though not maintaining the edge feature could reduce the computational complexity, it loses much expressive power and high-order information which makes it impractical**.

### 6.3 Variant II - forwarding only once

Another option to reduce the computation is to only forward once, which means the model uses all nodes in the graph to do prediction instead of just using the ego node. As a result, the total computational complexity is reduced to the computational complexity of GNN - $O(N^2)$.

However, another problem arises: **how to normalize data without setting an ego agent?** The following are two intuitive variants:

- Average of Agents (AA): we take the average coordinate and yaw angle of all agents at $t_0$ as the origin and the orientation of the positive x-axis respectively.
- Random Agent (RA): we randomly select an agent as the ego agent to do the normalization. Note that though it is faster in inference in theory, in training/validation, we still need to input all possible ego vehicle choices, which means no speed up. In fact, compared to FNT, it is equivalent to adding an auxiliary task of trajectory prediction for non-ego agents.

The illustration of AA/RA is in Fig. 3c. We compare the performance of RA and AA to the original forward-N-times (FNT) in Tab. 5. We also record the Train/Val time which is measured by the time to run one epoch on the train/val set on one RTX 2080 Ti with all GPU memory.

From the results, we can conclude that: 1. Forward only one time could significantly reduce the computational time. 2. However, the two alternative normalization methods dampen the performance. 3. AA performs the worst. We conjecture that AA is too sensitive since it is influenced by all agents in the scene and it cannot effectively reduce the input and output space. 4. TrajNet++ (pedestrian) dataset was less influenced. We conjecture that it is due to the smaller range of scenes for pedestrians.

In conclusion, **forward only once is effective for decreasing the computational complexity while a more stable normalization strategy is needed to maintain the performance.** Thus, as demonstrated before, in the proposed model, all agents to be treated symmetrically and homogeneously is significant for the performance of the model.

## 7 Conclusion

In this paper, we presented a multi-agent prediction model which combines the egocentric and allocentric views. It achieves SOTA performances on the two interaction-rich behavior datasets: INTERACTION (vehicles) and TrajNet++ (pedestrian). In the INTERACTION dataset's challenge, the model is ranked in the first place in both the regular track and generalization track. Besides, compared to the common multi-inference single-prediction model with similar architecture, the model could increase the running speed by 1.7-4.5 times. For future work, we would extend it to incorporate the map information and do multi-modal prediction with combining egocentric and allocentric views.

# 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[2] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018. doi:10.1109/CVPR.2018.00240.

[3] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14424–14432, 2020.

[4] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020.

[5] W. Zeng, M. Liang, R. Liao, and R. Urtasun. Lanercnn: Distributed representations for graph-centric motion forecasting. *arXiv preprint arXiv:2101.06653*, 2021.

[6] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, C. Li, and D. Anguelov. TNT: target-driven trajectory prediction. *CoRR*, abs/2008.08294, 2020. URL https://arxiv.org/abs/2008.08294.

[7] M. Ye, T. Cao, and Q. Chen. Tpcn: Temporal point cloud networks for motion forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11318–11327, June 2021.

[8] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. Qi, Y. Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. *arXiv preprint arXiv:2104.10133*, 2021.

[9] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2821–2830, 2019.

[10] Y. C. Tang and R. Salakhutdinov. Multiple futures prediction. 2019.

[11] R. L. Klatzky. Allocentric and egocentric spatial representations: Definitions, distinctions, and interconnections. In *Spatial cognition*, pages 1–17. Springer, 1998.

[12] 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.

[13] P. Kothari, S. Kreiss, and A. Alahi. Human trajectory forecasting in crowds: A deep learning perspective. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[14] J. Hong, B. Sapp, and J. Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8454–8462, 2019.

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

[16] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control. *ArXiv*, abs/2001.03093, 2020.

[17] J. Li, H. Ma, Z. Zhang, and M. Tomizuka. Social-wagdat: Interaction-aware trajectory prediction via wasserstein graph double-attention network. *CoRR*, abs/2002.06241, 2020. URL https://arxiv.org/abs/2002.06241.

[18] V. Kosaraju, A. Sadeghian, R. Martin-Martin, I. Reid, H. Rezatofighi, and S. Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlche Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/d09bf41544a3365a46c9077ebb5e35c3-Paper.pdf.

[19] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun. Learning lane graph representations for motion forecasting. In *European Conference on Computer Vision*, pages 541–556. Springer, 2020.

[20] S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pages 624–641. Springer, 2020.

[21] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue. Arbitrary-oriented scene text detection via rotation proposals. *IEEE Transactions on Multimedia*, 20(11):3111–3122, 2018.

[22] J. Ngiam, B. Caine, V. Vasudevan, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, et al. Scene transformer: A unified multi-task model for behavior prediction and planning. *arXiv preprint arXiv:2106.08417*, 2021.

[23] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017.

[24] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 86–99. PMLR, 30 Oct–01 Nov 2020. URL http://proceedings.mlr.press/v100/chai20a.html.

[25] A. Scibior, V. Lioutas, D. Reda, P. Bateni, and F. Wood. Imagining the road ahead: Multi-agent trajectory prediction via differentiable simulation. *arXiv preprint arXiv:2104.11212*, 2021.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017.

[27] 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*, 10(1):225–241, 2018.

[28] S. Pellegrini, A. Ess, and L. Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *European conference on computer vision*, pages 452–465. Springer, 2010.

[29] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. In *Computer graphics forum*, volume 26, pages 655–664. Wiley Online Library, 2007.

[30] T. Chavdarova, P. Baqué, S. Bouquet, A. Maksai, C. Jose, T. Bagautdinov, L. Lettry, P. Fua, L. Van Gool, and F. Fleuret. Wildtrack: A multi-camera hd dataset for dense unscripted pedestrian detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5030–5039, 2018.

[31] L. Sun, Z. Yan, S. M. Mellado, M. Hanheide, and T. Duckett. 3dof pedestrian trajectory prediction learned from long-term autonomous mobile robot deployment data. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5942–5948. IEEE, 2018.

[32] A. Alahi, V. Ramanathan, and L. Fei-Fei. Socially-aware large-scale crowd forecasting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2203–2210, 2014.

# Appendices

## A Qualitative Results

In this section, we give visualizations of the prediction results of the proposed model against baseline methods in Fig. 4 and Fig.5. From the visualizations, we could find that the proposed model could better capture the complex interactions among agents and make accurate predictions compared to baseline methods.

## B Discussion About Normalization of Trajectory Data

The widely used representation format for the motions of multiple agents is Cartesian coordinates. However, the origin and orientation of the coordinate system could be arbitrary to describe the same pattern. As a result, to make the neural network based model generalize well instead of overfitting the usually limited data, the input space and output space should be regularized cautiously.

Generally speaking, there are three kinds of design philosophy:

- P1: Input all possible representations of the same pattern to train the neural network
- P2: Preprocess the pattern into a canonical form. As a result, different forms of the same pattern would be the same input for the neural network.
- P3: Represent in a relative form to avoid using the absolute origin and orientation.

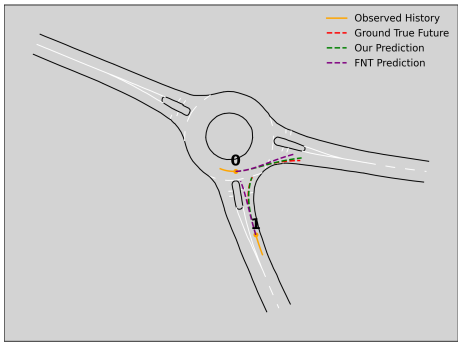
Only adopt P1 is infeasible since the range of origins is infinite. Many prediction models adopt P2 by 1. using the ego agent's coordinate at a certain time-step as the origin. 2. making the direction of the x-axis align with the ego agent's yaw angle at a certain time-step. In this paper, we adopt this kind of normalization as default and use $t_0$ since the choice of that certain time-step is arbitrary. We explore the effects of other normalization ways including relative features and random rotation of axis in Appendix C.

## C Exploration of Different Normalization Strategies

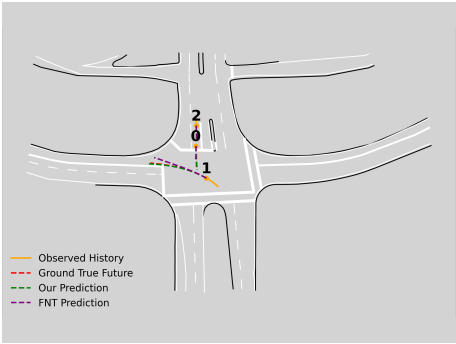
Besides the most widely adopted normalization method, we further conducted experiments with our method under other normalization strategies including:

- Temporally Relative Movements (Rel): in the original setting, we select the states of ego agent at a certain time step to normalize. In Rel, others' features are normalized at each time-step by ego agent's states at the corresponding time-step similar to [13, 27]. Take the coordinate as an example, the input edge is the relative coordinates of other agents under the coordinate system where the ego agent is at the origin all the time. Note that the node input is processed as the original normalization so that the absolute motion information is preserved. In this setting, we use different $W_k, W_v$ in the multi-head attention for self-loop edge and other edges so that the absolute and relative information could be processed respectively.
- Random Rotation (Rot): some SOTA methods [5, 7, 13, 16].use random rotation to augment the data instead of aligning the positive x-axis. This is feasible since the range of orientation is limited ($0 \sim 2\pi$) and it is possible to let the neural network learn to handle all possible orientations. Its benefit is that it can avoid the sensitivity to the yaw angle which might be inaccurate for the noisy data with lots of fluctuations and thus make the model much robust. However, its downside is that it needs much longer training epochs considering the input and output space are larger.

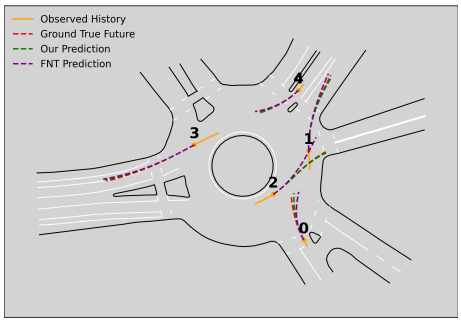
With the two aforementioned normalization strategies, we compare the following 5 settings: original, Rel, original+Rel (concatenation of the two kinds of input features), Rot, original+Rel+Rot. Considering normalization methods have a large influence on the generalization ability of the model, we conduct extra experiments on the Test Track (similar scenes with train/val) and Generalization



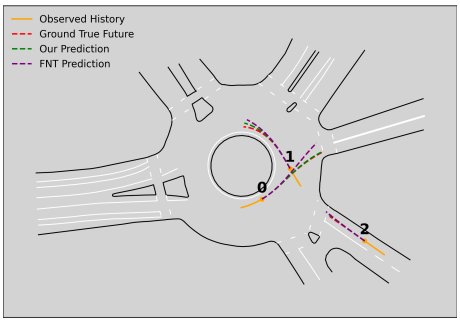

(a) The proposed model captured the interactive relations of v0-v1 and successfully predicted that v0 would like to go first and made a curve to occupy the road and v1 had to go after v1 while the baseline FNT model simply predicted lane-following behaviors.



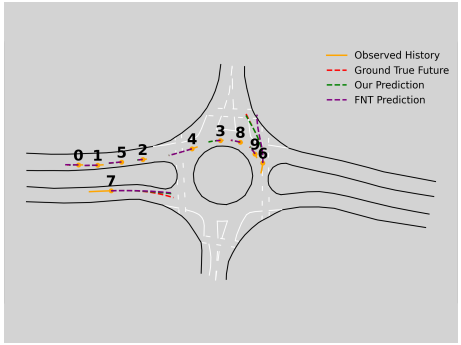

(b) The proposed model captured the interactions between v0 and v1 and successfully predicted that v1 had to make turn with less cure considering v0 made aggressive forward moving while the FNT model failed to predict the situation.




(c) The baseline FNT model failed to predicted the goal of v2 and thus its predictions about v0 and v2 deviated from the ground truth. On the other side, the proposed model correctly predicted v2's future motion and correctly predicted that v0 could not directly move toward its goal and had to follow v2.



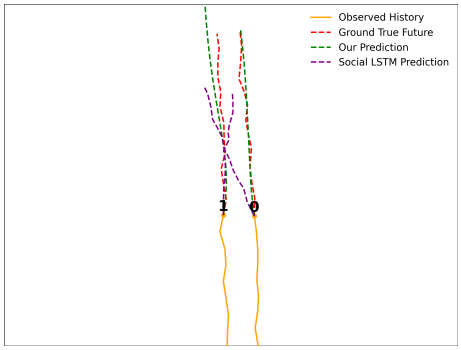

(d) The proposed model found the influence from v1 to v0 and correctly predicted v0's avoiding tendency while the FNT model failed to do so.



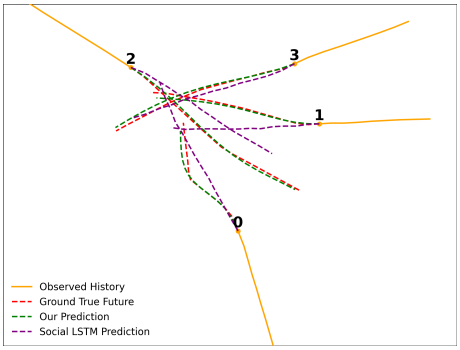

(e) v6 and v9 were close to each other which means that v6 should move cautiously toward its goal to avoid collision. The proposed correctly gave the path v6 had taken to cross while the FNT model gave a path which has collision with curbstone.

(f) The FNT model failed to predict the slow movement of v1 and thus wrongly predicted v2 as making aggressive and dangerous movement while the proposed model made the correct prediction for both v1 and v2.

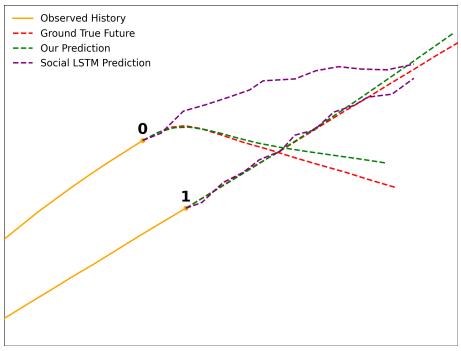
Figure 4: Qualitative results on validation set of INTERACTION Dataset. The observed history trejctories of agents are in yellow, ground truth of future trajectories are red, the proposed model's predictions are are in green, and the FNT model's predictions are in purple. The number are the ID of agents and we will refer the vehicle with ID i as vi.



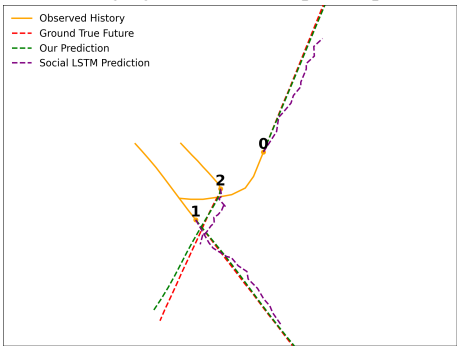

(a) This is the group moving behavior of pedestrians. The proposed model successfully predicted how two pedestrians walk together in a group while the Social LSTM's prediction caused a collision.



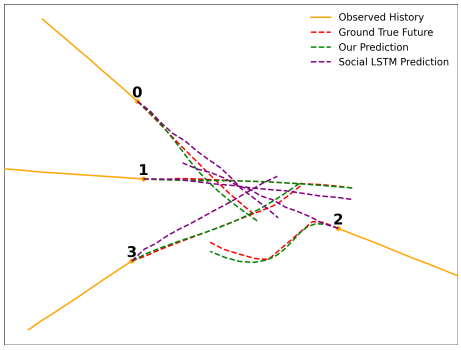

(b) p1, p2, p3, and p4 all have their own goal. Social LSTM made simple predictions of moving forward and caused very closed distance at some moments in the pairs p0-p3 and p1-p2 which is what pedestrians try to avoid in reality. The proposed model successfully predicted the changing of direction of p0 and p1.



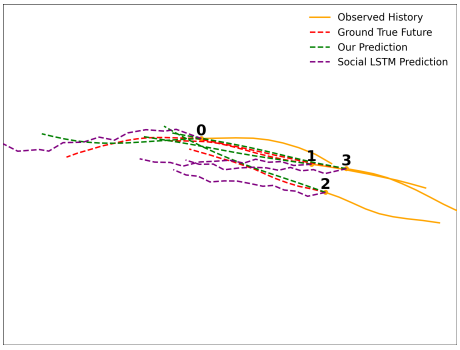

(c) In this case, p0 needed to make a right turn in front of p1. Social LSTM predicted v0's goal wrongly while the proposed model correctly predicted the path of v0 and v1.




(d) In this case, Social LSTM made too conservative predictions for p0 and p2. The proposed model correctly predicted the fast and efficient motion of the pedestrians.




(e) This is also a Collision-Avoidance case in which p0, p1, p2 and p3 had different goals and overlapped paths. The proposed captured the complex interactions and correctly predicted the bypassing of all agents while Social LSTM made a more straightforward predictions which may not often happen in reality.




(f) This is a Leader-Follower case where p0 was the leader and p1, p2, and p3 are the followers. The proposed correctly predicted their behaviors and intentions while the Social LSTM failed to predict the closely following pattern of p1, p2, and p3.

Figure 5: Qualitative results on validation set of TrajNet++ Dataset. The observed history trejctories of agents are in yellow, ground truth of future trajectories are red, the proposed model's predictions are are in green, and the Social LSTM's predictions are in purple. The number are the ID of agents and we will refer the pedestrian with ID i as pi.

Table 6: Comparison of different normalization strategies. The unit of ADE and FDE are both meters. ↓ means the lower the better.

| | INTERACTION | | | TrajNet++ |
|---|---|---|---|---|
| | Val ADE/FDE↓ | Test ADE/FDE↓ | Generalization ADE/FDE↓ | Val ADE/FDE↓ |
| origianl | 0.1723/0.5988 | 0.1903/0.6563 | 0.3394/1.1983 | 0.2079/0.4270 |
| Rel | 0.1773/0.6091 | 0.1925/0.6573 | 0.3412/1.1772 | 0.2207/0.4509 |
| original+Rel | 0.1768/0.6072 | 0.2115/0.6884 | 0.3395/1.1123 | 0.2203/0.4535 |
| Rot | 0.1732/0.6047 | 0.1906/0.6592 | 0.3297/1.1589 | 0.2573/0.5145 |
| original+Rel+Rot | 0.1857/0.6273 | 0.2031/0.6798 | 0.3283/1.1489 | 0.2200/0.4323 |

Track (different scenes of train/val) of INTERACTION dataset whose labels are both held by the organizer. From the results in Tab. 6, we can draw the following conclusion: 1. It seems that there is no gain to use the relative feature of each time-step (Rel). We conjecture that it is because this mathematical transformation is trivial for the neural network to learn. Additionally, the extra parameters $W_K, W_V$ make it easier to overfit. 2. Random rotation (Rot) also does not have advantages in these two datasets, which makes the longer training not worth it.

# D Experimental Details

## D.1 Datasets

**INTERACTION Datset** [12] consists of various highly interactive driving situations, including highway ramps, roundabouts, and intersections, recorded using drones or fixed cameras worldwide. For sample generation and train/val split, we use the official code to generate samples. The only difference is that we divide the snippets by time while the official code further chooses each agent in the scene as the ego agent and generates multiple samples. As a result, we have 43668 training snippets and 10751 validation snippets. The average number of agents in each sample is 10.7. Same as the default setting, the frames are sampled at 10Hz and we use 10 frames as input to predict 30 frames in the future. Results reported in the existing papers are all based on the single agent prediction. Considering the only difference is that we treat a scene with N examples as one sample while the default setting treats as N samples, the val ADE/FDE could be directly compared since it is a metric calculated by average over agents. In fact, all of their reported results are equivalent to being obtained from forward-N-times (FNT) of their model. Additionally, INTERACTION has an online leader board for two tracks whose labels are held: the test track which is in similar scenes with train/val data, and the generalization track which is in very different scenes. Since the competition is for single agent prediction, we only upload the target agent's prediction from our model.

**TrajNet++ Dataset** [13] comprises several popular public pedestrian datasets including ETH [28]/UCY [29]/WildTrack [30]/L-CAS [31]/CFF [32] and data generated by ORCA simulators. Their authors selected those highly interactive samples from the original data and made sure different kinds of interactions have a proper ratio. Here, we do not use the CFF data in it since this scene has too many agents in one sample (100+) which makes it impossible to run the simplest GNN for FNT setting under batch-size 1 on RTX 2080 Ti. Note that existing results reported in this dataset are based on the results of the primary pedestrian of each scene where the primary pedestrian is defined by the author as 'the interesting agent' in the scene. Thus, we re-divide the scene by time and predict all agents' future trajectories in the same snippet. We split the train/val in 8:2 and make sure there are no overlaps in time between train and validation set. As a result, we have obtained 60860 training samples and 15219 validation samples. The average number of agents in each sample is 5.3. Same as the default setting, the frames are sampled at 2.5Hz and we use 9 frames as input to predict 12 frames in the future. Since prediction for agents other than primary pedestrians are not considered in the ADE/FDE calcuating of the original dataset, we re-implement two top-rank methods on it under our multi-agent prediction setting.

## D.2 Loss

For all models, we set an MLP prediction head to output all the future coordinates of each agent based on their corresponding output node feature from GNN. We use the widely adopted smooth L1 loss as in Equ. 8 where B is the batch size and T is the number of steps predicted.

$$\mathcal{L} = \frac{1}{2BT}\sum_{b=1}^{B}\sum_{t=1}^{T} l(x_{bt} - \hat{x}_{bt}) + l(y_{bt} - \hat{y}_{bt}) \\ \text{where} \quad l(x) = \begin{cases} 0.5x^2 & ||x|| < 1 \\ ||x|| - 0.5 & ||x|| \geq 1 \end{cases} \tag{8}$$

## D.3 Implementation Details

We train all models with AdamW, lr 5e-4, weight-decay 1e-2, dropout 0.05, hidden dim 128, head number 4, GNN layer 2 if not specified. We train each model for 300 epochs and report the best results on the val set. The raw feature used in INTERACTION Dataset includes the coordinate, velocity, yaw angle, width, and length of vehicles. As for the TrajNet++ dataset, we used coordinate and calculated the velocity and yaw angle and take the three features as raw input. For the relative feature, we use relative coordinate, relative velocity, and relative yaw angle. As for all the angle data, we further use its corresponding cos and sin value as the input for ease of learning.

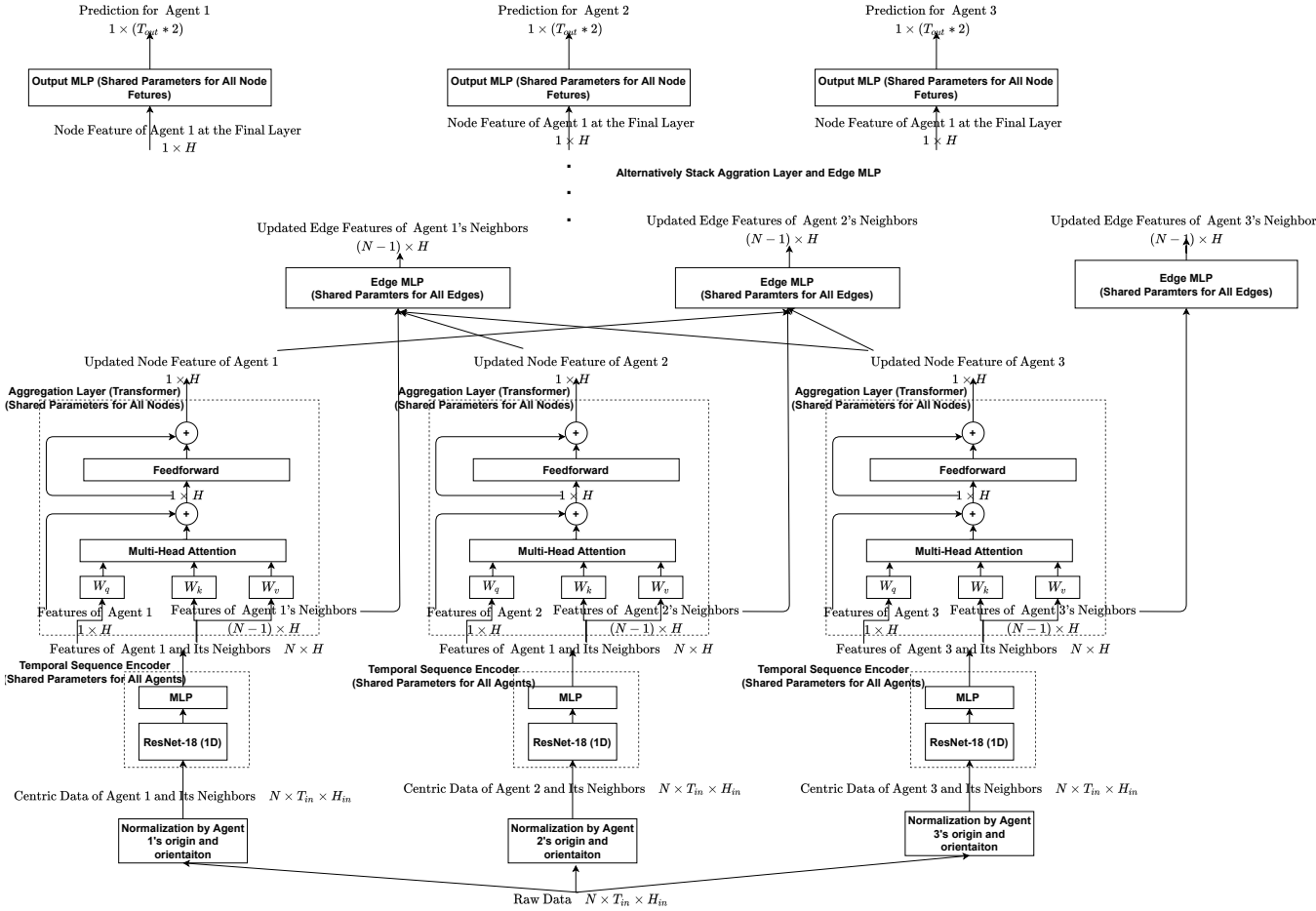



Figure 6: Implementation Details of the proposed method.