HeGA: Heterogeneous Graph Aggregation Network for Trajectory Prediction in High-Density Traffic

Shuncheng Liu*
University of Electronic Science and Technology of China
Chengdu, China
liushuncheng@std.uestc.edu.cn

Xu Chen*
University of Electronic Science and Technology of China
Chengdu, China
xuchen@std.uestc.edu.cn

Ziniu Wu
Massachusetts Institute of Technology
Cambridge, MA, USA
ziniuw@mit.edu

Liwei Deng
University of Electronic Science and Technology of China
Chengdu, China
deng_liwei@std.uestc.edu.cn

Han Su†
University of Electronic Science and Technology of China
Chengdu, China
hansu@uestc.edu.cn

Kai Zheng†
University of Electronic Science and Technology of China
Chengdu, China
zhengkai@uestc.edu.cn

ABSTRACT

Trajectory prediction enables the fast and accurate response of autonomous driving navigation in complex and dense traffics. In this paper, we present a novel trajectory prediction network called Heterogeneous Graph Aggregation (HeGA) for high-density heterogeneous traffic, where the traffic agents of various categories interact densely with each other. To predict the trajectory of a target agent, HeGA first automatically selects neighbors that interact with it by our proposed adaptive neighbor selector, and then aggregates their interactions based on a novel two-phase aggregation transformer block. At last, the historical residual connection LSTM enhances the historical information awareness and decodes the spatial coordinates as the prediction results. Extensive experiments on real data demonstrate that the proposed network significantly outperforms the existing state-of-the-art competitors by over 27% on average displacement error (ADE) and over 31% on final displacement error (FDE). We also deploy HeGA in a state-of-the-art framework for autonomous driving, demonstrating its superior applicability based on three simulated environments with different densities and complexities.

CCS CONCEPTS

• **Computer systems organization → Neural networks**; • **Information systems →** *Spatial-temporal systems*; *Traffic analysis.*

KEYWORDS

Heterogeneous Traffic; Trajectory Prediction; Autonomous Driving

*Both authors contribute equally to this paper.
†Corresponding authors: Kai Zheng and Han Su.

ACM Reference Format:
Shuncheng Liu, Xu Chen, Ziniu Wu, Liwei Deng, Han Su, and Kai Zheng. 2022. HeGA: Heterogeneous Graph Aggregation Network for Trajectory Prediction in High-Density Traffic. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557345

1 INTRODUCTION

Autonomous driving is one of the most active fields of research in artificial intelligence. The autonomous vehicle needs to explore the movement patterns of the surrounding agents and predict their future trajectories, in order to help make responsible navigation decisions. With the rapid growth of urbanisation, traffic nowadays exhibits two significant characteristics: high density and heterogeneity, which makes the trajectory prediction very difficult under such complex systems with various agents.

Existing methods [1, 7, 10, 25, 34] select the surrounding agents based on a fixed region (e.g., rectangular or elliptical region) or distance-based clustering (e.g., k-nearest neighbors). Then they use an encoder-decoder LSTM framework to predict the future trajectory of the target agent based on the historical movements of both the target agent and its surrounding agents. However, these methods show unsatisfactory in heterogeneous high-density traffics.

In high-density traffic systems, selecting the surrounding neighborhood via a fixed-size region or distance-based clustering often leads to either insufficient or redundant agents. In fact, the neighborhood region may depend on various traffic environments, and thus should be selected more adaptively. For instance, on the freeway, due to the high speed, the target vehicle needs to consider a much larger region than that on a local street. If the algorithm selects a small-sized region or small k parameter fitting a local street environment, it will not consider enough neighboring agents in a high-way environment, leading to an inaccurate prediction. Alternatively, if the algorithm selects a large-sized region or large k parameter fitting a high-way environment, the algorithm will consider excessive neighboring agents in dense local traffic. In this case, the redundant neighbors are falsely considered as useful input features, leading to an overfitting model that is not robust. Moreover, some latent features, such as velocity and direction, are completely ignored by these heuristic-based methods. These methods will be less socially aware of the abrupt interference.

The heterogeneous traffic systems involve various forms of interactions between different categories of traffic agents such as vehicles, pedestrians, and bicycles. Existing works use a single LSTM or CNN to extract the features of different categories of surrounding agents and model their interactions with the target agent. Such methods do not distinguish the features of different categories of agents, and over-simplify their interactions with the target agent, by forcing them to share parameters. However, different types of neighboring agents affect the target agent in different ways. For instance, a pedestrian is more cautious of a surrounding vehicle than a nearby bicycle. Therefore, ignoring the variability of the heterogeneous agents and over-simplifying their interactions (*a.k.a* 'wrong' parameter sharing) will hinder the target agent's ability to understand the complex traffics properly.

We summarize the existing challenges for trajectory prediction in high-density and heterogeneous traffics as the following two questions: ***(Q1)** how to adaptively select neighboring agents in various traffic environments; **(Q2)** how to effectively extract features for different categories of agents and model their interactions with the target agent.* To tackle these challenges, we propose a Heterogeneous Graph Aggregation network (HeGA). HeGA leverages the novel adaptive neighbor selector (ANS) to automatically choose neighbors around the target agent. The ANS learns sparse weights of interactions, corresponding to the importance of different neighbors. Based on the selected neighbors, the ANS generates a weighted heterogeneous graph, which not only represents agent features and interaction between agents but also filters redundant neighbors. This pruning on the heterogeneous graph enables model interpretation. For example, we observe that agents with different categories, distances, and speeds exhibit different interaction importance w.r.t. the target agent. This observation further justifies the necessity of ***Q2***. We treat the trajectory prediction as a sequence prediction task and then feed the heterogeneous graph to classical encoder-decoder architecture. We design the encoder to separately learn the features from different categories of agents and propose a novel two-phase aggregation transformer block to adaptively aggregate the features from them. Furthermore, the LSTM decoder with historical residual connection exploits and reinforces more history information from the encoder.

To the best of our knowledge, we are the first to propose an adaptive neighbor selector to learn the relative importance of neighbor interactions, and an aggregation transformer mechanism to handle heterogeneous interactions separately and adaptively. In summary, this work makes the following contributions:

- We propose an adaptive neighbor selector to automatically choose neighbors in diverse traffic environments and generate an interpretable heterogeneous graph.
- We propose a two-phase aggregation transformer block, targeting the heterogeneous graph structure, to adaptively combine information from different categories of agents.
- We conduct extensive experiments on two real-world datasets and observe that HeGA achieves state-of-the-art accuracy, an improvement over that of previous works by 27% on average displacement error (ADE) and 31% on final displacement error (FDE).
- We deploy different trajectory prediction networks (HeGA and baselines) in a prediction-and-search framework for autonomous driving, and investigate their applicability based on three simulated environments with increasing densities and complexities. Results show that HeGA achieves the fewest collisions and the shortest average driving time in all simulated environments, proving its superior applicability for autonomous driving.

2 PROBLEM DEFINITION AND WORKFLOW

In this section, we formally define the trajectory prediction problem under a heterogeneous traffic environment. We also present the high-level architecture of our proposed heterogeneous graph aggregation network and the workflow of trajectory prediction.

We assume that we can capture a set of heterogeneous traffic agents denoted as $\mathcal{A} = \{a_i\}_{i=1,2,\ldots,n}$ within the capture range of cameras, radars and other sensors. For any time t in a scene, the feature vector of the i-th agent a_i is represented as $f_i^t := [x_i^t \quad y_i^t \quad s_i \quad c_i]^\top$, where (x_i^t, y_i^t) is a spatial location in 2D coordinates; s_i refers to the size of the agent; c_i denotes the category of the agent. For simplicity, we consider three categories of heterogeneous traffic-agents throughout the paper, i.e., $c_i \in \{1, 2, 3\}$, where $1, 2, 3$ denotes vehicle, pedestrian and bicycle, respectively. Our approach can be naturally extended to traffic systems with more agent categories.

The problem can be defined as following: at time t, from the observed features of all captured agents $\mathcal{A}$ in the time interval $[t - r + 1, t]$, we predict the spatial coordinates of the target agent a_τ ($a_\tau \in \mathcal{A}$) in the time interval $[t + 1, t + z]$. r and z are the history and future window size respectively.

Figure 1 shows the architecture of our proposed heterogeneous graph aggregation network. The *inputs* of this end-to-end network are observed features of all agents $\mathcal{A}$ in historical time interval $[t - r + 1, t]$, i.e., a r-length sequence of f for each agent. The *outputs* are a sequence of predicted spatial coordinates $(\hat{x}_\tau, \hat{y}_\tau)$ in the future time interval $[t + 1, t + z]$ for the target agent. The heterogeneous graph aggregation network mainly consists of three components: (**1**) the novel adaptive neighbor selector (ANS) with global attention mechanism; (**2**) the encoder layer with a novel two-phase aggregation transformer block; (**3**) the decoder layer with historical residual connection LSTM.

As shown in Figure 1, to predict the trajectory of a bicycle agent a_τ, the input features of $\mathcal{A}$ are first fed into the ANS. The ANS selects the neighbor agents $\mathcal{N}_\tau$ ($\mathcal{N}_\tau \subset \mathcal{A}$) and forms a graph. The target and selected neighbor agents are the nodes. The weighted edges from each neighbor to a_τ indicate interaction where the learned weight corresponds to interaction importance. Then the heterogeneous graphs at all historical time steps are passed through the encoder layer. Specifically, the features of neighbor agents at each time step are encoded by the grouped structure of size embedding, space embedding, and time encoding. The aggregation transformer can aggregate all different groups of features into a high-dimensional feature vector via two-phase aggregation, i.e., agent and category aggregations. Using the encoded vector and historical information from the encoder, the decoder layer parses out the sequence of a_τ's spatial coordinates $(\hat{x}_\tau, \hat{y}_\tau)$ in the next z time steps using an LSTM model with historical residual connections.

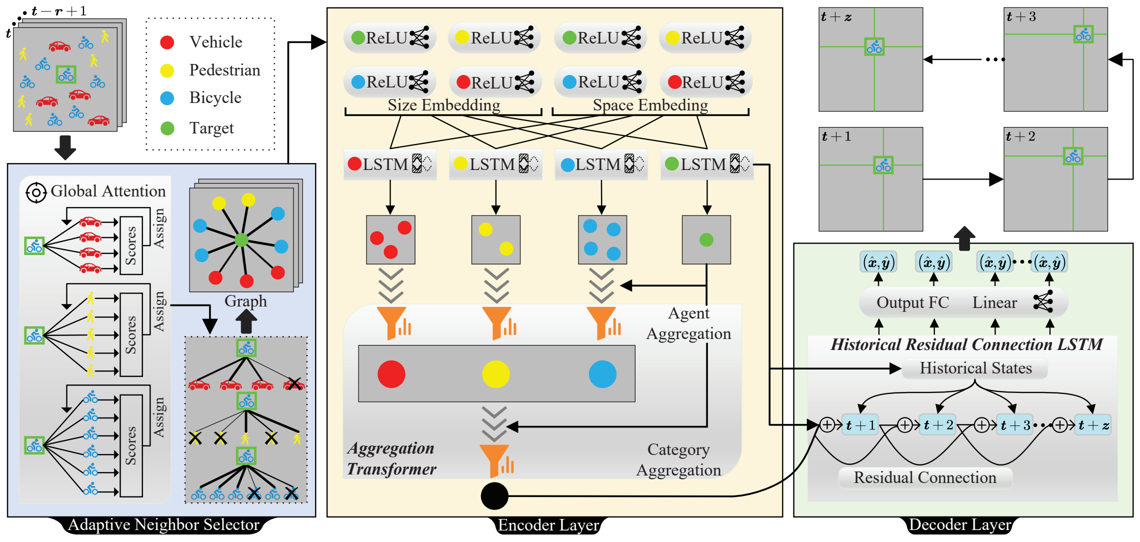


Figure 1: HeGA network overview

3 HETEROGENEOUS GRAPH AGGREGATION

3.1 Adaptive Neighbor Selector

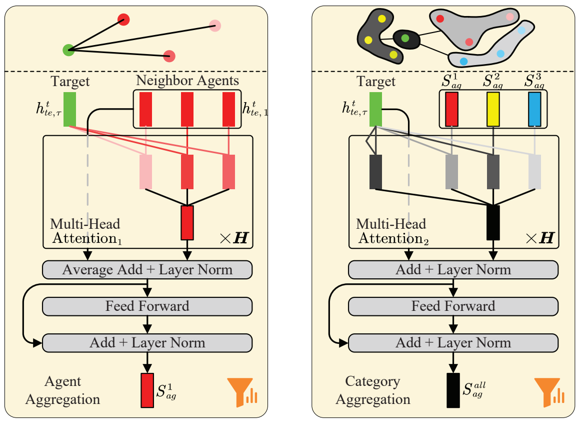
Eliminating the redundant neighbors helps the network focus on the effective interactions only, thus simultaneously increasing the model accuracy and reducing the complexity. Neighbor selection (NS) is a non-trivial problem, especially in the case of high-density heterogeneous traffic. Various neighbor interactions exist within such a complex system whereas simple heuristic-based NS (e.g., elliptical, rectangular, and k-NN) can hardly model them. To automatically select valuable neighbors around the target agent, we design an adaptive neighbor selector (ANS), whose workflow is shown on the left side of Figure 1. Taking the features of all agents as inputs, ANS uses a global attention layer to evaluate the interaction importance scores S of all non-target agents (i.e., $\mathcal{A} - \{a_\tau\}$). Specifically, we group the non-target agents based on categories, and calculate the interaction importance scores of agents in different groups separately. For each agents a_i in a category ($i \in [1, n_c]$, $c \in \{1,2,3\}$), its interaction importance score S_i is calculated as $attention((f_\tau, f_i); W_{ga})$, where $attention$ is the global attention function with Softmax activation [24], W_{ga} is the learnable attention parameter, and f_i is the features of an agent in time interval $[t-r+1, t]$. The global attention not only pays attention to the distance between agents, but also pays attention to the speed information due to the features of multiple time steps. To select neighbors by S_i, we add the $L1$ penalty to our final loss function to make the interaction importance scores S sparse and thus eliminate the redundant neighbors. Such constraints, controlled by a tunable parameter λ, can reduce the number of effective features and obtain sparsity. Naturally, we can remove the agents with zero interaction importance scores. We call the remaining agents as a_τ's neighbors (denoted by $\mathcal{N}_\tau$). Unlike previous NS methods that only distance information is considered, our ANS includes features extraction over multiple time steps and avoids parameter sharing for different categories. Although the workload in ANS has increased, the time efficiency of subsequent modules is improved since fewer neighbors are considered. The penalty parameter λ controls the number of neighbors, which further influences the trade-off between accuracy and efficiency (detailed in Section 4.3).

3D Graph Formation. To address the limitation of the grid network, we view this task as graph-based supervised learning. Considering traffic agents as nodes and relationships as edges, we can construct a 3D graph, where each node has various feature vectors f_i^t. The 3D graph consists of the 2D plane graph propagating along the time dimension. To illustrate, ① adding all heterogeneous traffic agents including a_τ and $\mathcal{N}_\tau$ to the vertex set. So there are $1 + |\mathcal{N}_\tau|$ vertices in the beginning. ② Adding undirected edges with interaction importance scores from target agent a_τ to each neighbor in $\mathcal{N}_\tau$. Now a 2D star-like plane graph is completed. ③ Repeating steps ①-② for $r-1$ times. At last, we get a 3D graph, where each neighbor agent as node connects only with the target agent and the weight on each edge corresponds to each neighbor's interaction importance score. This heterogeneous 3D graph will be input to the encoder layer.

3.2 Encoder Layer

In heterogeneous traffic, different categories of agents have different forms of interactions. Representing these interactions with a single network as in previous works would result in 'wrong' parameter sharing and thus inaccurate prediction. Thereby, modeling them separately and adaptively is crucial in high-density heterogeneous traffic.

Taking the weighted heterogeneous 3D graph as input, HeGA treats the weight on each edge as the weighted importance for each corresponding neighbor node and multiplies each node's features with its weight, before feeding to the encoder. The encoder of HeGA consists of two feature embedding stages. At the first stage, the encoder embeds the size and space features at each time step t for each node. Then, the time-sequence of the embedded features will be fed into LSTMs to learn temporal information of spatial features, such as velocity and acceleration, for each agent. At the



(a) Agent Aggregation (Vehicle Example) (b) Category Aggregation

Figure 2: Aggregation transformer architecture

second stage, the encoder uses a novel two-phase aggregation transformer to aggregate neighbor interactions for the target agent. The aggregated vector will be the final output of the encoder.

Stage 1: Feature Embedding for Each Agent Node. In heterogeneous trajectory prediction, different categories of agents should be encoded separately to avoid sharing weights. Specifically, we leverage different embedding matrices to embed the size and spatial features for each category of heterogeneous agents, separately.

All neighboring nodes $\mathcal{N}_\tau$ will be first grouped into $\mathcal{N}^1_\tau$, $\mathcal{N}^2_\tau$ and $\mathcal{N}^3_\tau$ based on their categories. For space embedding at each time step t, the three groups $\mathcal{N}^{1,2,3}_\tau$ and the target agent node a_τ are embedded as $e^t_{sp,c}$ and $e^t_{sp,\tau}$ separately, which are defined as follows:

$$e^t_{sp,c} = \phi(s^t_i; W^c_{sp}),\ e^t_{sp,\tau} = \phi(s^t_\tau; W^\tau_{sp}), \tag{1}$$

where $c \in \{1, 2, 3\}$ refers to three categories; ϕ is a fully connected neural network with ReLU activation and parameters W; s_i and s_τ are the 2D space feature (x_i, y_i) of neighbor agent and target agent node at time t, respectively. For size embedding, four units are used to activate three groups of neighbors $\mathcal{N}^{1,2,3}_\tau$ and target agent a_τ to vectors e_{sz} respectively. The size embedding for each agent keeps fixed for all time steps, which is denoted as $e_{sz,c}$ or $e_{sz,\tau}$.

Next, we define four individual temporal LSTMs for each agent categories to learn temporal information of spatial features, such as velocity and acceleration, as follows:

$$\begin{aligned} h^t_{te,c} &= LSTM(Concat(e_{sz,c}, e^t_{sp,c}), h^{t-1}_{te,c}; W^c_{te}),\\ h^t_{te,\tau} &= LSTM(Concat(e_{sz,\tau}, e^t_{sp,\tau}), h^{t-1}_{te,\tau}; W^\tau_{te}), \end{aligned} \tag{2}$$

where $h^t_{te,c}$ and $h^t_{te,\tau}$ are the hidden states of $\mathcal{N}^c_\tau$ and a_τ respectively , W^c_{te} and W^τ_{te} denote the corresponding LSTM parameters. Corresponding temporal LSTM is executed for each agent node to obtain the final hidden state as the final feature for this node.

Stage 2: Aggregating Neighbor Interactions for Target Agent Node. To capture the different interactions of different neighbor agents, we use a novel two-phase aggregation transformer to fuse different neighbors i.e., agent and category aggregation. Unlike the simple self-attention-based aggregators in HAN [39], our improved transformer block aggregates feature hierarchically, from both agent-level and category-level.

The ***agent aggregation*** is designed for aggregating the interactions between the target agent node and all neighbors from a specific category. We use a *vehicle* example in Figure 2(a) to illustrate the agent aggregating.

First, the outputs of LSTMs ($h^t_{te,\tau}$ and $h^t_{te,1}$) will be the inputs to 'multi-head attention$_1$' and embedded to three vectors: query Q^τ, key K, and value V, as follows:

$$Q^\tau = \phi(h^t_{te,\tau}; W^Q), K = \phi(h^t_{te,1}; W^K), V = \phi(h^t_{te,1}; W^V). \tag{3}$$

Then we calculate an attention vector *score* for vehicle neighbor agents, following the standard procedure [37]. This procedure will be repeated H times and the final attention score S' can be calculated by concatenating H scores. The benefit of multi-head attention is that every separated attention will extract a feature from a different perspective. The final attention score S' is defined as:

$$S' = Concat(score_1, \ldots, score_H) \cdot W^s \tag{4}$$

where W^s are the parameters of a linear transformation.

Second, we find it beneficial to process the output S' with residual, layer normalization [5] and feed forward [37]. To increase the global awareness of all neighbors vehicles, we add the average value of all processed node features to the first residual connection, i.e., 'average add+layer norm' in Figure 2(a). We compute this residual connection as follows:

$$S' = LayerNorm(\overline{h^t_{te,1}} + S') \tag{5}$$

where $\overline{h^t_{te,1}}$ denotes the average of the processed feature (output from LSTM in Stage 1) for all vehicles.

Third, we can derive the final agent aggregation result S^1_{ag} by feeding S' into the second residual connection ('feed forward' and 'add+layer norm' in the Figure 2(a)), which are calculated as follows:

$$\begin{aligned} S_{feed} &= W^{f1} \cdot \max(0, W^{f2} \cdot S_{res} + b_1) + b_2,\\ S^1_{ag} &= LayerNorm(S_{res} + S_{feed}), \end{aligned} \tag{6}$$

where S_{feed} denotes the feed forward result, both W^{f1} and W^{f2} are the parameters of linear transformations biased by b_1 and b_2 respectively with a ReLU activation in between. Finally, we arrive at the output of this agent aggregation, which can be denoted as $S^1_{ag} \in \mathbb{R}^{1\times d_m}$. Similarly, for each category of neighbors, we can obtain corresponding aggregation results, i.e., S^1_{ag} for vehicles, S^2_{ag} for pedestrians and S^3_{ag} for bicycles.

The ***category aggregation*** combines the information from all categories of neighbors (S^1_{ag},S^2_{ag},S^3_{ag}) and target agent ($h^t_{te,\tau}$ from Stage 1) itself, into one vector S^{all}_{ag}, as in Figure 2(b). Since the target agent needs to consider the importance of itself, the multi-head attention and residual connection are different from that in the agent aggregation.Specifically, the outputs of temporal LSTM and agent aggregation will first be embedded as follows:

$$\begin{aligned} Q^\tau_{ca} &= \phi(h^t_{te,\tau}; W^Q_{ca}),\\ K_{ca} &= \phi(Concat(h^t_{te,\tau}, S^1_{ag}, S^2_{ag}, S^3_{ag}); W^K_{ca}),\\ V_{ca} &= \phi(Concat(h^t_{te,\tau}, S^1_{ag}, S^2_{ag}, S^3_{ag}); W^V_{ca}). \end{aligned} \tag{7}$$

We calculate the attention value S'' as the output of 'multi-head attention$_2$', following the standard procedure [37]. Category aggregation not only aims to trade off the attention between neighbors but also focuses on the information of the target agent. Therefore, we add residual connections of target agent features, i.e. *LayerNorm* ($h^t_{te,\tau} + S''$). Thereafter, the rest calculation of category aggregation is the same as agent aggregation, and the category aggregation outputs the encoder layer result S^{all}_{ag}.

3.3 Decoder Layer

Inspired by the recent advance in language [6] and vision [15] models, we propose a historical residual connection LSTM as the decoder layer to take advantage of more useful history information and leverage the residual connection. This decoder has two technological advantages over the traditional vanilla LSTM, used by existing works [7, 10, 30]. First, for each decoding time step, instead of using the output from the previous one-time step, we combine outputs of the previous two-time steps as the input. Second, for each decoding time step, instead of using the hidden state from the previous time step, we take the final hidden state in encoder LSTM as the state input.

The right side in Figure 1 shows the workflow of HeGA's decoder. The hidden states $h^{t}_{te,\tau}$, $h^{t-1}_{te,\tau}$ from encoder and the encoder result S^{all}_{ag} are fed into the decoder. The historical residual connection LSTM can be defined as follows for different time steps l:

$$h^{t+l}_{de} = \begin{cases} LSTM(\alpha(h^{t-1}_{te,\tau}) + S^{all}_{ag}, h^{t}_{te,\tau}), l = 1, \\ LSTM(\alpha(S^{all}_{ag}) + h^{t+l-1}_{de}, h^{t}_{te,\tau}), l = 2, \\ LSTM(\alpha(h^{t+l-2}_{de}) + h^{t+l-1}_{de}, h^{t}_{te,\tau}), l \in [3, z], \end{cases} \tag{8}$$

where α is a learnable factor of residual connection [40] which can determine the importance of historical information, and the $LSTM(input, state\ input)$ function processes two types of inputs using the standard procedure [17]. In each decoding step, we fix the passed hidden state input as $h^{t}_{te,\tau}$, which can reinforce history information from the encoder.

Thereafter, at each decoding time step l, the output hidden state h^{t+l}_{de} will be linearly transformed into a 2 dimensional vector, i.e., predicted spatial location $(\hat{x}^{t+l}_{\tau}, \hat{y}^{t+l}_{\tau})$. The coordinates in all future time interval $[t+1, t+z]$ jointly form the target agent trajectory prediction results.

Loss Function. We combine the $L2$ distance and and $L1$ penalty as the loss function, which can be defined as follows:

$$Loss = \frac{1}{z} \sum_{l=1}^{z} [(x^{t+l}_{\tau} - \hat{x}^{t+l}_{\tau})^2 + (y^{t+l}_{\tau} - \hat{y}^{t+l}_{\tau})^2] + \lambda \sum_{i=1}^{n-1} |S_i| \tag{9}$$

where x^{t+l}_{τ} and y^{t+l}_{τ} are the true values of the spatial coordinates at t+l, λ is a tuneable parameter for $L1$ penalty and S_i is the interaction importance score of agent a_i. We have tried both to sample from bivariate Gaussian distribution and to use $L2$ distance. We find using $L2$ distance is much beneficial to gradient descent since it boosts the velocity of gradient descent and achieves better results.

4 EXPERIMENTS

4.1 Experimental Settings

Datasets. We evaluate the proposed framework on two publicly available datasets: BaiduApollo [25] and NGSIM (US highway 101) [35]. The BaiduApollo dataset is collected in urban areas with high-density traffics, which contains real-world trajectories of heterogeneous traffic agents, including vehicles, bicycles, and pedestrians. The raw NGSIM dataset only contains vehicles on the highway, which is captured in $10Hz$ (10 frames per second) over a period of 45 minutes. We preprocess the NGSIM dataset with two steps, ① removing the frames with less than 10 vehicles, ② down-sampling the data to a rate of $2Hz$ (setting the time step to 0.5s). Unless otherwise stated, all experiments are conducted on the BaiduApollo dataset.

Table 1: Hyperparameters setting of HeGA

Hyperparameters	Values
Penalty Parameter λ in ANS	0.01
Embedding Size of x_i, y_i and s_i	32
Dimension of LSTM Hidden State	128
Dimension of Feed-Forward Layer in Agent Aggregation	128
Dimension of Feed-Forward Layer in Category Aggregation	64
Number of Heads H	8

Evaluation Metrics. Two evaluation metrics are used to measure the prediction error (a lower metric stands for a better model performance):

• *Average Displacement Error (ADE)*: The Euclidean distance between the ground truth trajectories and the predicted trajectories averaged overall predicted time steps [31].

• *Final Displacement Error (FDE)*: The Euclidean distance between the ground truth destination and the predicted destination at the last predicted time step [1].

Evaluation Baselines. We compare our network with four state-of-the-art models:

• *LSTM Encoder-Decoder (LSTM-ED)*: A traditional architecture widely used in trajectory prediction [30].

• *Social LSTM (S-LSTM)*: An LSTM-based model equipped with grid-based social pooling layers that is used to predict pedestrian trajectories in crowds [1].

• *TrafficPredict*: An LSTM-based architecture based on a 4D graph to learn interactions between different agents [25].

• *TraPHic*: An LSTM-based network using horizon state-space convolution and neighbor state-space convolution to learn interactions between agents in different ranges[7].

We also perform the ablation study with the following variants of our approach:

• *HeGA-NoANS*: Our proposed network without the adaptive neighbor selector. We use all non-target agents ($\mathcal{A} - \{a_{\tau}\}$) as neighbors.

• *HeGA-NoAF*: Our proposed network without the aggregation transformer module. We substitute the two-phase aggregation transformer with the one-phase multi-head attention.

• *HeGA-NoHRC*: Our proposed network without the historical residual connections in the decoder layer. We use a vanilla LSTM as the decoder layer.

• *HeGA-NoAF&HRC*: Our proposed network without both the aggregation transformer module and the historical residual connections. We substitute them with the same settings above.

• *HeGA-PS*: Our proposed network with parameter sharing for different categories of agents. We set all the category features as 1, thus canceling the separate calculation in the network.

Detailed Settings of HeGA and Baselines. Following the commonly used evaluation methodology in [25], we set the length of the input historical trajectories to 4 frames (r=4), and the trajectory length to predict to 6 frames (z=6), which correspond to 2 and 3 seconds respectively. The hyperparameters setting of our network is listed in Table 1. For both the BaiduApollo and NGSIM datasets, we divide them into three parts (60%, 20%, 20%) for training, validation, and testing respectively. We add paddings into inputs to

Table 2: ADE and FDE of baselines and ours (HeGA & variants) for heterogeneous trajectory prediction

ADE/FDE	Baselines				Ours					
Target Categories	LSTM-ED	S-LSTM	TrafficPredict	TraPHic	**HeGA**	HeGA -NoANS	HeGA -NoAF	HeGA -NoHRC	HeGA -NoAF&HRC	HeGA -PS
Vehicle	8.49/14.65	8.37/13.75	7.94/10.58	6.85/10.71	**2.57/4.85**	2.89/5.04	2.86/5.07	3.31/5.33	4.49/7.98	5.22/8.37
Pedestrian	5.44/7.65	5.02/6.23	5.33/7.42	3.17/5.43	**2.29/3.56**	2.41/3.68	2.58/3.96	2.45/3.61	2.43/3.30	2.64/3.85
Bicycle	8.21/12.96	7.68/11.42	7.26/12.86	5.22/7.12	**2.73/4.88**	5.44/8.06	5.51/8.74	4.98/7.76	5.87/9.26	6.43/10.16
Average	7.52/12.22	7.23/11.04	7.08/10.07	5.42/8.42	**2.51/4.46**	3.26/5.22	3.29/5.45	3.37/5.28	4.14/6.82	4.68/7.36

Table 3: ADE and FDE of baselines and HeGA for homogeneous trajectory prediction

Methods	LSTM-ED	S-LSTM	TrafficPredict	TraPHic	**HeGA**
ADE	0.49	0.36	0.33	0.26	**0.18**
FDE	0.84	0.66	0.49	0.45	**0.38**

ensure every training sample has the same dimension for efficient batch training. The maximum padding number is chosen according to the training dataset. We train the network using the Adam optimizer [18] for 200 epochs with a scheduled learning rate of 0.001 and a batch size of 64. The hyperparameters are tuned by the grid search method on the validation part, and we set the best one as the default value. Finally, we test our network on the testing part as the final result. For a fair comparison, we add the size embedding as an extra input to improve the baseline models and try our best to grid search on the validation set for the best hyperparameters. For all the compared models, we report the improved results.

Implementations. All models and algorithms are implemented in Python on Linux, and we conduct the experiments on a machine with an Intel(R) CPU i7-4770@3.4GHz with 32G RAM and NVIDIA TITAN Xp with 12GB GPU memory.

4.2 End-to-End Evaluation of HeGA

HeGA VS Baselines. To give a thorough comparison on heterogeneous trajectory prediction, we first test HeGA and baselines on the BaiduApollo dataset in which there are three types of traffic agents (see Table 2). As expected, LSTM-ED performs the worst in all methods since it can only encode very limited features. It performs especially worse on predicting long trajectories, such as the trajectories of vehicles. S-LSTM and TrafficPredict have similar performances because they both try to jointly predict all agents at the same time. Since they do not have target agents and can not focus on every interaction, the predictions tend to have a larger deviation. In addition, the heterogeneous influence can not be learned completely in these two models. TraPHic outperforms previous methods due to its ability to learn interactions in different ranges. However, it still can not distinguish interactions with different agents. Our framework is noticeably better than other methods since we take every traffic agents under consideration and allocate them with adaptive weights to learn heterogeneous influence. Our framework reduces at least 27% ADE and 31% FDE compared to all previous models.

Furthermore, to show the homogeneous trajectory prediction performance, we test HeGA and baselines on the NGSIM dataset in which there are only vehicle traffic agents. As shown in Table 3, LSTM-ED has decent performance since there are much fewer

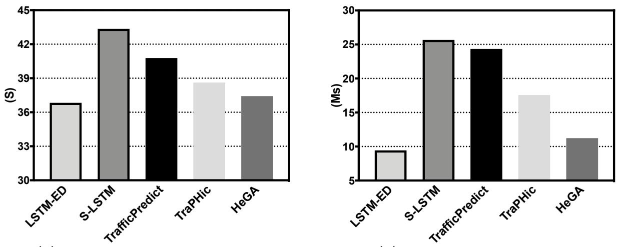


Figure 3: Efficiency of baselines and HeGA

interactions on the highway and the density of traffic agents is much lower. In the homogeneous trajectory prediction situation, our two-phase aggregation transformer module degenerates to the one-phase *agent aggregation* module, but we still find that HeGA performs better than baselines.

Ablation Study of HeGA. As shown in Table 2, we also conduct an ablation study to demonstrate the effectiveness of the novel modules in HeGA. HeGA-NoANS performs worse than HeGA in all three categories, which proves that not all non-target agents are useful for prediction in high-density traffic situation. Excess information brings varying degrees of damage to different categories.

HeGA-NoAF performs uniformly worse than HeGA on all types of target agents. Interestingly, the performance degradation is especially significant for bicycles. The reason is that compared to the other two types of agents, bicycles have stronger interactions with both vehicles and pedestrians. Therefore, the experiment demonstrates that the AF module can better extract features for different agent categories, and model their interactions more precisely.

HeGA-NoHRC has decent prediction results for short trajectories from pedestrians and bicycles. However, it performs noticeably worse than networks with HRC, including HeGA and HeGA-NoAF. This clearly shows that HRC is helpful for long-range features memorization by linking every current frame to the past two frames directly. Accordingly, it is expected that HeGA-NoAF&HRC performs worse than HeGA-NoAF and HeGA-NoHRC in most cases.

At last, HeGA-PS performs worst among all experiments, which proves the adverse impact of parameter sharing. In HeGA-PS, the problem is regarded as a homogeneous trajectory prediction situation where the embedding and encoding layers are all in one category. Experiments show that wrong parameter sharing brings even more than 40% performance degradation.

Efficiency of HeGA. To evaluate the computation efficiency of HeGA, we measure the mean training time per epoch and mean inference time of HeGA and baselines detailed in Figure 3. For mean training time per epoch (see Figure 3(a)), all results are in the same order of magnitude (36 ~ 45 *seconds*). LSTM-ED takes the

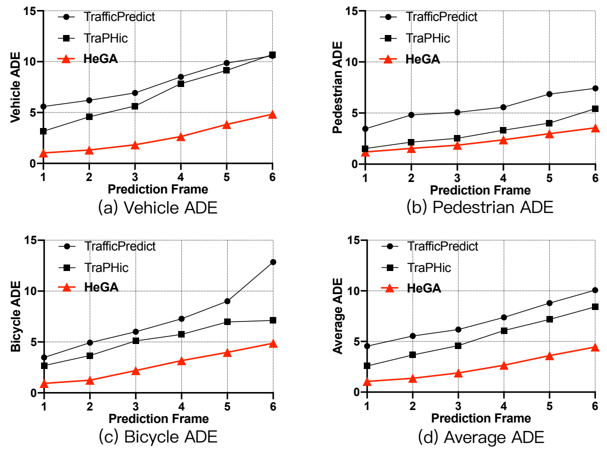


Figure 4: Variations of ADE of each frame

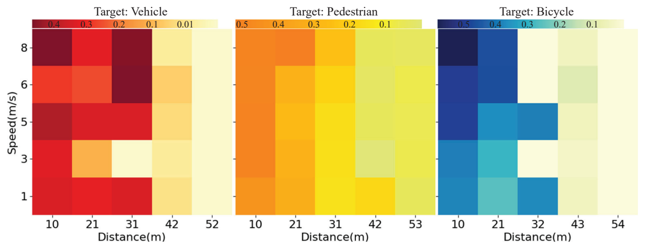


Figure 5: Visualization of interaction importance scores

least amount of time to train an epoch while Social LSTM takes the longest time. Since our novel ANS module eliminates many redundant agents, the time consumption of our network is close to that of LSTM-ED. On the contrary, the time consumption of social LSTM is almost twice as much as that of HeGA because it involves grid search. It proves that our network achieves solid results without sacrificing much time. For mean inference time (see Figure 3(b)), all results are also in the same order of magnitude ($9 \sim 26$ *milliseconds*). The tendency of the two graphs is consistent. It takes about $11ms$ for our network to inference 6 frames, proving that HeGA is applicable for real-time application.

Variations of ADE of Each Frame. Figure 4 show the variations of ADE along the timeline. Generally, our network has the lowest ADE of all frames compared to other models. All chart shows an increasing tendency of ADE along the timeline. It is worth noting that in Figure 4(d), the average accuracy of our network degrades slower than other models when predicting farther into the future. Figure 4(a) indicates that vehicle trajectories are the most difficult to predict among the three, as they tend to be longer and the speed variation of vehicles is larger than pedestrians and bicycles. Our network shows a noticeable improvement in trajectory prediction for vehicles, due to the combination of the HRC module and the AF module. We follow the convention of using short observations to predict long future trajectories in previous works. Note that the first two prediction frames are the most important. The $75th$ percentile error of our network in the first two frames is less than 1m (see Figure 4), which is relatively precise.

Table 4: Average ADE of neighbor selection methods

Methods	Full	Elliptical	Rectangle	k-NN	**ANS**
Average ADE	3.26	3.97	4.05	3.13	**2.51**

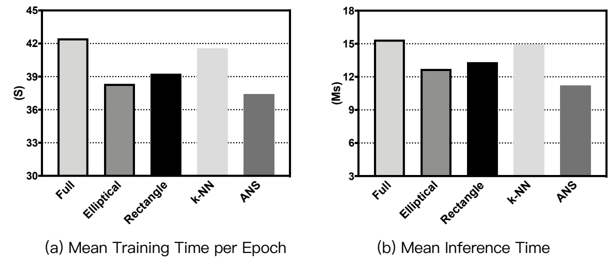


Figure 6: Efficiency of neighbor selection methods

4.3 Effect of Adaptive Neighbor Selector

Interpretability Analysis. To analyze the generated graph in HeGA, we visualize the interaction importance scores of neighbor agents outputted by ANS with respect to the category, distance, and speed using heat maps in Figure 5 where the x axes denote the distances between the target agent and neighbor agents, and the y axes denote the speeds of neighbor agents. For the vehicles, the ANS learns to focus on a short distance (10 meters) and a middle distance (31 meters), because drivers in vehicles are likely to see farther to make timely reactions. In contrast, for the pedestrians and bicycles, the ANS focuses on a short distance (10 meters). For all three categories, ANS pays attention to neighbor agents with higher speed, which is consistent with reality. Three heat maps prove that our ANS can extract useful latent features and select reasonable regions.

Effectiveness of Neighbor Selection Methods. To study the effectiveness of NS methods, including heuristic-based methods (i.e., full-coverage, elliptical [7], rectangle [10], k-NN [34]) and our ANS method, we compare the average ADE of HeGA under these NS methods. The results are shown in Table 4. We can see that ANS is more accurate than other methods. The full-coverage method is not as good as our ANS method because it contains redundant and misleading neighbors that bring extra loss to outputs. The elliptical and rectangle methods perform worse than the former one because the range and shape of the region depend on traffic and road conditions. The k-NN methods perform worse than our ANS because it ignore some latent features, e.g., velocity and direction.

Efficiency of Neighbor Selection Methods. We also record the mean training time per epoch and mean inference time of HeGA under different NS methods. As shown in Figure 6, ANS is more efficient than other methods. The full-coverage method takes the longest time since it uses all non-target agents ($\mathcal{A} - \{a_\tau\}$) as neighbors. The k-NN method is better than the full-coverage method because it removes some distant agents. The elliptical and rectangle methods perform better than the k-NN method since they have no sorting calculation. Compared with heuristic-based NS methods, our ANS method eliminates many redundant agents, the time-efficiency of subsequent modules (i.e., encoder and decoder) is improved since fewer neighbors are considered.

Table 5: Effect of penalty parameter λ

λ	10^{-5}	10^{-4}	10^{-3}	$\mathbf{10^{-2}}$	10^{-1}	1
Average ADE	3.23	3.02	2.74	**2.51**	2.97	3.15
MTTE(S)	42.15	40.35	38.62	**37.43**	35.82	34.57
MIT(Ms)	15.09	14.26	12.75	**11.23**	10.48	8.86

Effect of Penalty Parameter λ. λ is a penalty parameter in ANS, it controls the sparsity of the interaction importance scores S, and further influences the number of neighbors. The more the number of neighbors, the lower the efficiency of the network. Conversely, the fewer the number of neighbors, the higher the efficiency of the network. However, too many or too few neighbors will affect the accuracy. To balance the accuracy and efficiency of HeGA, we tune λ from 10^{-5} to 1, and record the average ADE, mean training time per epoch (MTTE) and mean inference time (MIT) of HeGA under different λ. As shown in Table 5, when λ=10^{-2}, the average ADE is the lowest. The mean training time per epoch and mean inference time are decent when λ=10^{-2}. Due to the above observation, we choose λ=10^{-2} to balance accuracy and efficiency.

4.4 Applicability of HeGA

We further deploy different trajectory prediction networks (HeGA and baselines) in a framework for autonomous driving, and investigate their applicability based on three simulated environments. In the following, we introduce the deployment and simulated environments, and finally present the applicability analysis.

Deployment. In order to make trajectory prediction practical for autonomous driving, we deploy HeGA and baselines (LSTM-ED, S-LSTM, TrafficPredict, TraPHIc) in Cheetah framework [22], respectively. Cheetah is a prediction-and-search framework that enables autonomous vehicle to perform lane change and/or speed change safely and efficiently. In particular, Cheetah first predicts the near future trajectories for surrounding agents, then searches for the optimal maneuver with the maximum speed and minimum impact on surrounding agents, and finally performs the optimal maneuver. In order for the framework to function properly, HeGA and baselines follow the trajectory prediction settings used in Cheetah.

Simulated Environments. As it requires interaction between an autonomous vehicle and other traffic agents, we utilize Cheetah with different trajectory prediction networks to control the autonomous vehicle, and use the microscopic traffic simulator [19, 22, 32, 41] to simulates the behaviors of other traffic agents. In this work, we focus on the complex and dense traffics, so we predefine three simulated environments with increasing traffic densities and complexities as follows:

• *Highway*: An autonomous vehicle and 600 conventional vehicles (with human drivers) traveling on a straight three-lane road of length *3km* with periodic boundary conditions [32]. Figure 7(a) presents a screenshot of the Highway simulated environment.

• *Urban Road*: Based on the Highway simulated environment, we add 300 pedestrians and 300 bicycles on the road. The 300 pedestrians and 300 bicycles are required to move in the rightmost lane of the road, which is common in urban traffic system. Figure 7(b) presents a screenshot of the Urban Road simulated environment.

• *Downtown Street*: Based on the Urban Road simulated environment, we further add 300 pedestrians and 300 bicycles on the road

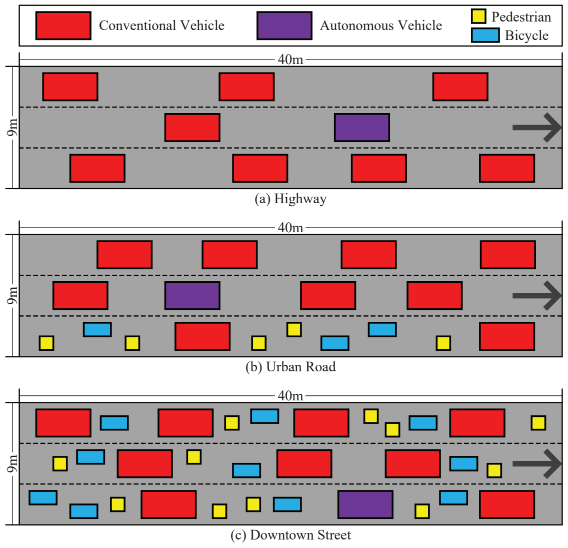


Figure 7: Screenshots of simulated environments

and relax restrictions for all pedestrians and bicycles. The 600 pedestrians and 600 bicycles can move in all lanes of the road, which is common in downtown traffic system. Figure 7(c) presents a screenshot of the Downtown Street simulated environment.

Overall, the densities and complexities of all simulated environments can be ranked as: *Highway*>*Urban Road*>*Downtown Street*, and thus we can study the applicability of different trajectory prediction networks comprehensively. The detailed settings of simulated environments are listed in Table 7, following the settings used in previous works [19, 22, 32, 41].

Applicability Analysis. To investigate the applicability of HeGA and baselines based on three simulated environments, we conducted 100 tests on each simulated environment wherein each test the autonomous vehicle is initialized at a random position and drives through *3km*, and measure the applicability from both safety and efficiency aspects:

• *Safety*: We record the number of collisions caused by the autonomous vehicle in each test. This metric directly reflects the safety of the autonomous vehicle.

• *Efficiency*: We further record the driving time of the autonomous vehicle in each test. It should be noted that we add extra 5 minutes as a penalty time when the autonomous vehicle collides with any traffic agent. This metric directly reflects the efficiency of the autonomous vehicle.

Accordingly, if the autonomous vehicle is safer (i.e., fewer collisions) and more efficient (i.e., shorter driving time), the corresponding trajectory prediction network has better applicability. We report the average number of collisions and average driving time of the autonomous vehicle controlled by Cheetah with HeGA and baselines in Table 6. We can see that Cheetah with HeGA causes the fewest collisions in all simulated environments, demonstrating the safety of our network for autonomous driving. Moreover, Cheetah with HeGA achieves the shortest average driving time in all simulated environments, which demonstrates the efficiency of HeGA for autonomous driving. The main reasons are two-fold. Firstly, HeGA can adaptively choose neighbors in diverse traffic

Table 6: Applicability (safety and efficiency) of baselines and HeGA for autonomous driving

Metrics	**Average Number of Collisions**					**Average Driving Time(Min)**				
Simulated Environments	LSTM-ED	S-LSTM	TrafficPredict	TraPHic	**HeGA**	LSTM-ED	S-LSTM	TrafficPredict	TraPHic	**HeGA**
Highway	0.85	0.79	0.72	0.67	**0.43**	8.14	7.45	6.87	6.49	**5.38**
Urban Road	1.14	0.97	0.86	0.92	**0.77**	9.63	8.85	7.95	8.03	**6.54**
Downtown Street	1.59	1.35	1.12	1.08	**0.84**	12.37	11.85	10.53	10.24	**8.62**

Table 7: Detailed settings of Simulated Environments

Description		**Settings**
Basics	Size of Vehicle	$4m \times 2m$
	Size of Pedestrian	$1m \times 1m$
	Size of Bicycle	$2m \times 1m$
	Road Shape	Straight
	Road Length	$3km$
	Road Boundary	Periodic
	Number of Lane	3
	Number of Autonomous Vehicle	1
	Number of Conventional Vehicle	600
	Traffic Flow	Unidirectional
Highway	Number of Conventional Vehicle	600
	Speed Limit for Vehicle	$0 \sim 115\ km/h$
	Traffic Density	200 $agents/km$
	Traffic Complexity	Low
Urban Road	Number of Pedestrian	300
	Number of Bicycle	300
	Speed Limit for Vehicle	$0 \sim 50\ km/h$
	Speed Limit for Pedestrian	$0 \sim 10\ km/h$
	Speed Limit for Bicycle	$0 \sim 25\ km/h$
	Traffic Density	400 $agents/km$
	Traffic Complexity	Medium
Downtown Street	Number of Pedestrian	600
	Number of Bicycle	600
	Speed Limit for Vehicle	$0 \sim 40\ km/h$
	Speed Limit for Pedestrian	$0 \sim 8\ km/h$
	Speed Limit for Bicycle	$0 \sim 20\ km/h$
	Traffic Density	600 $agents/km$
	Traffic Complexity	High

environments. Secondly, HeGA can effectively combine information from different categories of traffic agents. Therefore, HeGA provides more accurate trajectory prediction results for Cheetah, thus making the autonomous vehicle safer and more efficient.

5 RELATED WORK

Traffic trajectory prediction tasks can be classified into homogeneous and heterogeneous problems.

Homogeneous Trajectory Prediction. Homogeneous trajectory prediction methods are designed for scenarios with a single category of traffic agents, which corresponds to vehicles on highways or pedestrians in crowd space. Traditionally, homogeneous trajectory prediction is solved by statistical learning method, including Hidden Markov Models (HMMs) [12, 36], Bayesian Network [20, 33], Support Vector Machines [4], etc. These methods only consider single trajectory pattern recognition thus ignore the social interactions, which have an important role in the real world. Social interaction is the influence received by the ego agent from its neighbors. It is an abstract problem to model and simulate. More works that model traffic-agent interactions achieve better results. Helbing et al. [16] take social interaction and inner motivation into consideration called social force. The simulation is based on hand-craft rules but achieves surprisingly good results in some simple scenes. However, it can not be generalized on modern datasets.

Recently, more researchers utilize the power of deep learning methods to tackle this problem. The majority of them use RNN based encoder-decoder paradigm. [3, 11, 23, 26, 27] used LSTM encoder-decoder architecture to predict vehicle trajectories. However, the simple vanilla LSTM in their methods omits a large amount of historical information, which is critical in traffic systems. For the pedestrian trajectory prediction, Social LSTM [1], Social GAN [14] and Social Attention [38] are pioneering methods modeling social interaction with grid maps. Generative Adversarial Networks (GANs) based LSTM can generate multiple possible results of trajectories. However, these grid-based methods suffer from inherent inaccuracies due to the discretization of continuous 2D space. Furthermore, the sparse grids occupy large storage and require excessive computational power to traverse through. The aforementioned methods can not capture the interaction between different categories of agents in heterogeneous scenarios.

Heterogeneous Trajectory Prediction. Recently, heterogeneous methods have attracted more attention. TrafficPredict [25] used a complete graph with LSTMs to learn different agent interactions. However, their complete graph is uninterpretable due to the LSTM-based edge encoding, as well as their complete graph can not choose neighbors adaptively. Convolutional Neural Networks (CNN), achieving great results and used as a paradigm in computer vision, are also applied in trajectory prediction problems [2, 9, 13, 21, 28, 29]. [7, 8] propose the prediction methods with horizon and neighbor state-spaces convolutional neural network. They combine CNN and LSTM as a hybrid model and use grid-based pooling to measure agent interactions. However, the drawbacks of these state-of-the-art methods include: (1) the 'wrong' parameter sharing in the representation of agents interaction; (2) the heuristic-based methods in the neighbor selection.

6 CONCLUSION

Trajectory prediction in dense and heterogeneous scenarios is a challenging problem for autonomous driving that lacks effective solutions. In this work, we propose HeGA, a novel heterogeneous graph aggregation network that predicts the trajectories for different categories of agents in high-density heterogeneous traffics. Extensive empirical studies based on real datasets confirm the superiority of our network over the state-of-the-art approaches in terms of ADE and FDE. We also deploy HeGA in a prediction-and-search framework for autonomous driving, proving its superior applicability based on three simulated environments.

ACKNOWLEDGMENTS

This work is partially supported by NSFC (No. 61972069, 61836007 and 61832017), and Shenzhen Municipal Science and Technology R&D Funding Basic Research Program (JCYJ20210324133607021).

REFERENCES

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social lstm: Human trajectory prediction in crowded spaces. In *CVPR*. 961–971.

[2] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. 2017. Understanding of a convolutional neural network. In *ICET*. 1–6.

[3] F. Altché and A. de La Fortelle. 2017. An LSTM network for highway trajectory prediction. In *ITSC*. 353–359.

[4] Georges S Aoude, Vishnu R Desaraju, Lauren H Stephens, and Jonathan P How. 2012. Driver behavior classification at intersections and validation on large naturalistic data set. *IEEE Transactions on Intelligent Transportation Systems* (2012), 724–736.

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[6] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *ICASSP*. 4960–4964.

[7] Rohan Chandra, Uttaran Bhattacharya, Aniket Bera, and Dinesh Manocha. 2019. Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions. In *CVPR*. 8483–8492.

[8] Rohan Chandra, Uttaran Bhattacharya, Christian Roncal, Aniket Bera, and Dinesh Manocha. 2019. RobustTP: End-to-End Trajectory Prediction for Heterogeneous Road-Agents in Dense Traffic with Noisy Sensor Inputs. In *ACM Computer Science in Cars Symposium*. 1–9.

[9] Liwei Deng, Hao Sun, Rui Sun, Yan Zhao, and Han Su. 2022. Efficient and Effective Similar Subtrajectory Search: A Spatial-aware Comprehension Approach. *TIST* 13, 3 (2022), 35:1–35:22.

[10] Nachiket Deo and Mohan M Trivedi. 2018. Convolutional social pooling for vehicle trajectory prediction. In *Workshops on CVPR*. 1468–1476.

[11] N. Deo and M. M. Trivedi. 2018. Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs. In *IEEE Intelligent Vehicles Symposium (IV)*. 1179–1184.

[12] Jonas Firl, Hagen Stübing, Sorin A Huss, and Christoph Stiller. 2012. Predictive maneuver evaluation for enhancement of car-to-x mobility data. In *IEEE Intelligent Vehicles Symposium (IV)*. 558–564.

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

[14] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. 2018. Social gan: Socially acceptable trajectories with generative adversarial networks. In *CVPR*. 2255–2264.

[15] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. 770–778.

[16] Dirk Helbing and Peter Molnar. 1995. Social force model for pedestrian dynamics. *Physical review E* (1995), 4282.

[17] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation* (1997), 1735–1780.

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. 2012. Recent development and applications of SUMO-Simulation of Urban MObility. *International journal on advances in systems and measurements* (2012).

[20] Stéphanie Lefèvre, Christian Laugier, and Javier Ibañez-Guzmán. 2011. Exploiting map information for driver intention estimation at road intersections. In *IEEE Intelligent Vehicles Symposium (IV)*. 583–588.

[21] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. 2020. Learning lane graph representations for motion forecasting. In *European Conference on Computer Vision*. Springer, 541–556.

[22] Shuncheng Liu, Han Su, Yan Zhao, Kai Zeng, and Kai Zheng. 2021. Lane Change Scheduling for Autonomous Vehicle: A Prediction-and-Search Framework. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3343–3353.

[23] Shuncheng Liu, Zhi Xu, Huimin Ren, Tianfu He, Boyang Han, Jie Bao, Kai Zheng, and Yu Zheng. 2022. Detecting Loaded Trajectories for Hazardous Chemicals Transportation. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 3294–3306.

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

[25] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. 2019. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *AAAI*. 6120–6127.

[26] K. Messaoud, I. Yahiaoui, A. Verroust, and F. Nashashibi. 2020. Attention Based Vehicle Trajectory Prediction. *IEEE Transactions on Intelligent Vehicles* (2020), 175–185.

[27] Kaouther Messaoud, Itheri Yahiaoui, Anne Verroust-Blondet, and Fawzi Nashashibi. 2019. Non-local social pooling for vehicle trajectory prediction. In *IEEE Intelligent Vehicles Symposium (IV)*. 975–980.

[28] Sriram Narayanan, Ramin Moslemi, Francesco Pittaluga, Buyu Liu, and Manmohan Chandraker. 2021. Divide-and-conquer for lane-aware diverse trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15799–15808.

[29] Nishant Nikhil and Brendan Tran Morris. 2018. Convolutional neural network for trajectory prediction. In *Workshops on ECCV*.

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

[31] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. 2009. You'll never walk alone: Modeling social behavior for multi-target tracking. In *ICCV*. 261–268.

[32] Marcus Rickert, Kai Nagel, Michael Schreckenberg, and Andreas Latour. 1996. Two lane traffic simulations using cellular automata. *Physica A: Statistical Mechanics and its Applications* (1996), 534–550.

[33] Matthias Schreier, Volker Willert, and Jürgen Adamy. 2014. Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems. In *ITSC*. 334–341.

[34] Aditya Shrivastava, Jai Prakash V Verma, Swati Jain, and Sanjay Garg. 2021. A deep learning based approach for trajectory estimation using geographically clustered data. *SN Applied Sciences* (2021), 1–17.

[35] The Next Generation SIMulation. 2007. US Highway 101 Dataset. (2007).

[36] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. 2020. A survey of trajectory distance measures and performance evaluation. *The VLDB Journal* 29, 1 (2020), 3–32.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.

[38] Anirudh Vemula, Katharina Muelling, and Jean Oh. 2018. Social attention: Modeling attention in human crowds. In *ICRA*. 1–7.

[39] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*. 2022–2032.

[40] Yi Xu, Jing Yang, and Shaoyi Du. 2020. CF-LSTM: Cascaded Feature-Based Long Short-Term Networks for Predicting Pedestrian Trajectory. In *AAAI*.

[41] Hwasoo Yeo, Alexander Skabardonis, John Halkias, James Colyar, and Vassili Alexiadis. 2008. Oversaturated freeway flow algorithm for use in next generation simulation. *Transportation Research Record* (2008), 68–79.