HDGT: Heterogeneous Driving Graph Transformer for Multi-Agent Trajectory Prediction via Scene Encoding

Xiaosong Jia, Penghao Wu, Li Chen, Hongyang Li, Yu Liu, Junchi Yan, *Senior Member, IEEE*

Abstract—One essential task for autonomous driving is to encode the information of a driving scene into vector representations so that the downstream task such as trajectory prediction could perform well. The driving scene is complicated, and there exists heterogeneity within elements, where they own diverse types of information *i.e.*, agent dynamics, map routing, road lines, *etc*. Meanwhile, there also exist relativity across elements - meaning they have spatial relations with each other; such relations should be canonically represented regarding the relative measurements since the absolute value of the coordinate is meaningless. Taking these two observations into consideration, we propose a novel backbone, namely Heterogeneous Driving Graph Transformer (HDGT), which models the driving scene as a heterogeneous graph with different types of nodes and edges. For graph construction, each node represents either an agent or a road element and each edge represents their semantics relations such as Pedestrian-To-Crosswalk, Lane-To-Left-Lane. As for spatial relation encoding, instead of setting a fixed global reference, the coordinate information of the node as well as its in-edges is transformed to the local node-centric coordinate system. For the aggregation module in the graph neural network (GNN), we adopt the transformer structure in a hierarchical way to fit the heterogeneous nature of inputs. Experimental results show that the proposed method achieves new state-of-the-art on INTERACTION Prediction Challenge and Waymo Open Motion Challenge, in which we rank 1^{st} and 2^{nd} respectively regarding the minADE/minFDE metric.

Index Terms—Autonomous Driving, Trajectory Prediction, Heterogeneous Graph Neural Network, Scene Understanding

1 Introduction

Autonomous driving is one of the most influential domains for AI applications in recent years. For a system design in autonomous driving, there are a few key modules to name: perception/localization, prediction, planning, and control. The perception/localization module is responsible to extract visual and semantic information from raw sensor data and HD-Map. The outcomes of preceding functionalities are in a compact but highly unstructured form, *e.g.*, coordinates, states, and relations. For the downstream tasks such as trajectory prediction and planning, It is essential to obtain comprehensive semantic representations from those unstructured inputs. The conventional pipeline for the task is to first acquire a representation vector for each target agent and the corresponding vector should exhibit both the ego and environmental information. Then given the vector as input, the subsequent decoder could conduct prediction/planning in a desired form (raw trajectory/control signal/etc), combined with some prior assumptions or physical constraints. However, even deeply inspired by the success of deep learning literature in recent years, the encoding process needs to be investigated nonetheless. There is no de facto paradigm to encode a representative vector due to the following observations.

- X. Jia, P. Wu, and J. Yan are with Department of Computer Science and Engineering, and MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University, and Shanghai AI Lab, Shanghai, China.
L. Chen, H. Li, and Y. Liu are with Shanghai AI Lab, Shanghai, China.
This work was partly done when Xiaosong Jia and Penghao Wu were interns at the Shanghai AI Lab.
E-mail: {jiaxiaosong,wupenghaocraig,yanjunchi}@sjtu.edu.cn
{chenli1, lihongyang, liuyu}@pjlab.org.cn,
Correspondence author: Junchi Yan

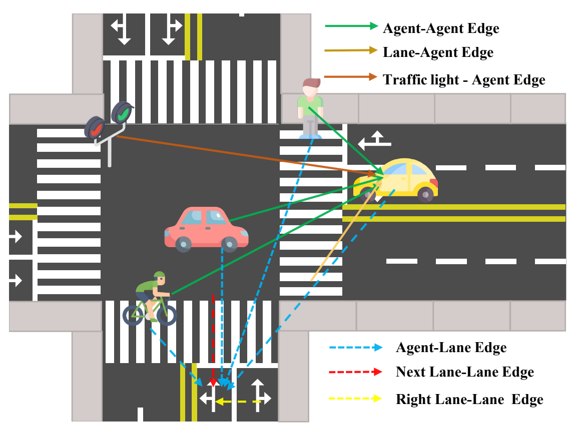


Fig. 1: Illustration of the heterogeneous driving graph. In the proposed framework, different types of agents and map elements are modeled as different types of nodes. The diverse semantic relations among them are modeled as different types of edges.

One inherent property that lies in the data is the *heterogeneity*, where we imply objects and their relations to have different types. Specifically, the input data may include agents' motion states, traffic lights with position, lane polylines with feasible movement connections, road-edge polygons with various types, *etc*. Fig. 1 illustrates an heterogeneous driving graph. It is non-trivial to encode those information comprehensively since they are highly unstructured and it is intuitive to believe that a good network design should consider a wide span of the input type and their semantic relations *separately and explicitly*. Another observation is the *relativity* among data, where we indicate each agent in the driving scene should process surrounding elements in spirit of its local, relative measurement system. This is especially true for the driving scenarios. If we adopt the global, absolute measurement, the vehicle would get confused to take action (turn left, go straight, *etc.*) since the information it receives is not under its local reference. From a machine learning perspective, if we adopt global reference, models would simply overfit the value of the absolute coordinates on the training scenes, and perform poorly on the unseen scenes. Hence, instead of processing the absolute coordinates, one should unanimously encode the relative relations among elements in a local view. Such a view shift from other elements to the agent should be incorporated into the network explicitly.

We have witnessed some promising attempts towards a good representation in the encoding process for trajectory prediction, following the two intuitions aforementioned. VectorNet based approaches [1]–[4] treat all agents and lanes as the same type in a fully-connected graph, which ignore the different types of elements and semantic relations such as lane connection. LaneGCN based solutions [5]–[9] introduce several graphs according to distance heuristic and connective information of the lane. This is a good attempt to investigate data heterogeneity. However, the sequential order of those four different relation aggregation functions in LaneGCN is designed manually, which may not be optimal and not flexible to extend. SceneTransformer [10] builds the fully connected graph and use cross-attention to update agents' feature; it lacks the lane connectivity information and does not update the lane nodes. Furthermore, most approaches aforementioned select a fixed agent as the global reference in the coordinate system, which is non-symmetric for the reference agent and other agents. As suggested in [11], to obtain the best results for multi-agent prediction, those methods have to forward multiple times with each agent as the reference agent. The concurrent work MultiPath++ [12] designs gated mechanism to update information and the features are in each agents' local coordinate system. The lane connectivity information is still missed and the updating of different nodes is still sequential in MultiPath++ nonetheless.

To alleviate the problems mentioned above, we propose the **H**eterogeneous **D**riving **G**raph **T**ransformer, namely **HDGT**. The driving scene is modeled as a heterogeneous graph in which each node could either be an agent or a road element. The spatial features of nodes as well as their in-edges[1] are in each node's local, ego-centric system. We devise the network structure in a spirit from the recent success of Transformer [13]. For the aggregation function in the graph, different sets of Transformer parameters are adopted to attend to different kinds of in-edge types. At each GNN layer, all nodes and edges are updated simultaneously - making the model easy to stack and scale.

To summarize, the main highlights of this paper are:

1) Instead of choosing one reference agent, we propose a symmetric way of encoding the spatial relationship between elements in the scene, which significantly improves the generalization ability and enables the model to predict multiple agents' future in one forward without performance drop.

2) We devise an unified heterogeneous GNN for trajectory forecasting with HD-map and propose a hierarchical way of adopting Transformer to fit the heterogeneous nature of the inputs. The explicit modeling of all semantics and relations in the scene could improve the prediction accuracy by a large margin while the unified and simultaneous updating of all element makes it easy to implement and stack.

3) We examine the proposed HDGT on two challenging large-scale datasets and it achieves state-of-the-art-performance on the corresponding online leaderboards. We do thorough ablation studies to evaluate the effectiveness of each design.

In particular, our extensive experimental results show that the proposed HDGT achieves new records on two challenging benchmarks, namely INTERACTION Prediction Challenge [14] and Waymo Open Motion Challenge [11]. Throughout 11/3/2021 (our final submission on INTERACTION) to 4/21/2022 (we arxiv this report), we still ranked the $\mathbf{1^{st}}$ and $\mathbf{2^{nd}}$ respectively in terms of minADE/minFDE metric. Source code and trained models will be made public available.

2 Related Work

Heterogeneous Graph Neural Network Graph neural network (GNN) [15] aims for extracting information from graph structure. For graphs with different types of nodes and edges, heterogeneous graph neural network is further introduced. R-GCN [16] uses GCN like aggregating functions but with different parameters for different types. HetGNN [17] use distinct RNNs for different nodes. Transformer [13], as a set function to aggregate information of elements, have gain hude success in multiple fields. As for its application in heterogeneous graph, HGT [18] apply different Transformers for different relations on the academic graph and [19] applies it on social media graph.

Raster-based Trajectory Prediction Early deep learning based methods for trajectory prediction focus on the interaction among agents and do not take the HD-Map into consideration. Social LSTM [20] combines LSTM with social pooling while Social GAN [21] further introduces GAN to improve the diversity of prediction. [22] considers the edge relations to better capture the interaction. [22] designs a GNN for agents detected from raw sensor data to capture their interactions. Trajectron [23] combines recurrent sequence modeling and variational deep generative modeling with graph-structured model to predicts distributions of multi-agent trajectory. IDE-Net [24] explicitly extracts the interaction between agents by Transformer. ST-GCN [25] adopts GCN on both spatial and temporal dimension. However, such approaches ignore road information which may lead to sub-optimal performance.

To extract information from HD-Map, one popular choice is to use the rasterized images [26]–[31]. Specifically, they project map elements into a top-down view image according to the 2D coordinates; different types of elements may be painted in different channels. Following this branch, [32] further formulates the output on the rasterized images as well. Trajectron++ [33] improves Trajectron with the rasterized HD-Map, heterogeneous agents type, and future-conditional predictions. Prank [34] produces the conditional distribution of agent's trajectories plausible in the given scene. [35] aims to model the joint distribution over future trajectories via an implicit latent variable model with GNN for agents only. [36] builds a agent-only heterogeneous graph neural network whose edges including the relative information. [37] designs a memory network to store the scene knowledge. [38] uses multiple sensors' data for trajectory prediction. With the blossom of CNN structure in vision, this branch of methods could outperform those without map information. However, as

1. In this paper, the heterogeneous graph we build is directional, which means the edge feature from A to B is different from B to A. We denote in-edges of A as all the edges going into A.

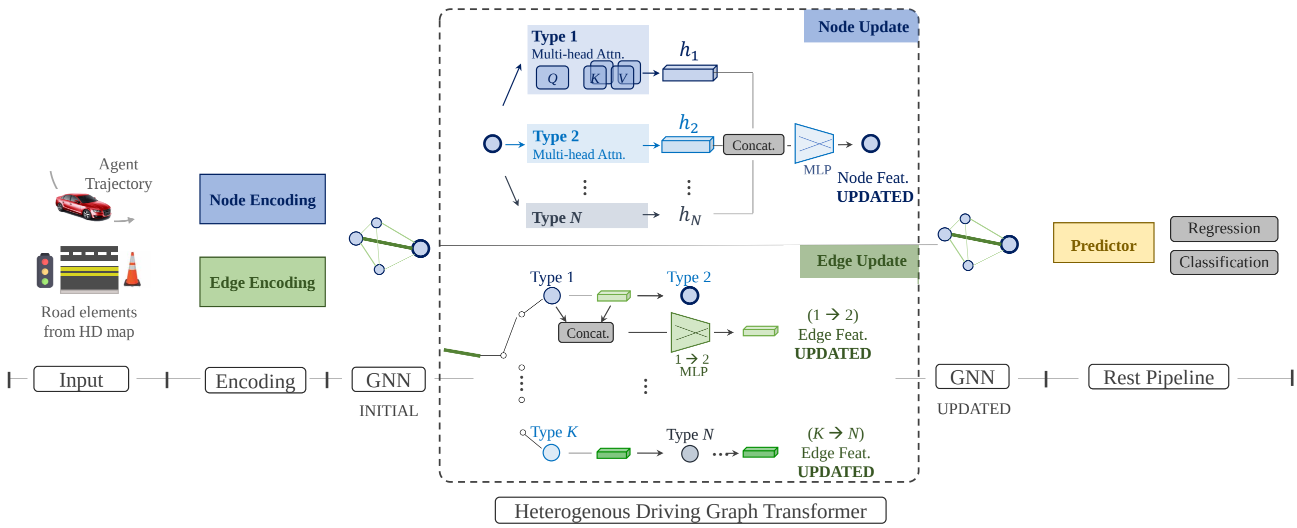


Fig. 2: The proposed HDGT. Inputs of agent trajectory and road semantics are fed into the Encoding module, where the preliminary features of node and edge are constructed as input to GNN. The core lies in the Heterogeneous Transformer which treats different types of agents and edges separately to obtain a representation. The updated graph is then fed to the predictor and generate final trajectories.

demonstrated in [1], the raster-based methods lose too much information during painting and are inefficient to generate good representation, since the road elements are sparse in image space. Besides, it is difficult for CNN to capture global information.

Coordinate-based Scene Encoding. Encoding the input in a coordinate-based spirit [1], [5] has proven great performance in multiple competitions [11], [14], [39], [40]. Works in this branch typically employ 1D CNN or LSTM [41] to process the temporal data, adopt the PointNet [42] to process polylines, and use graph neural network (GNN) to handle the relation among elements. An interesting exception is TPCN [43] and its extension DCMS [44] which treat all coordinate data as point clouds. VectorNet [1] uses sub-graph for lane and agent encoding and a fully connected global graph to capture their relations. Its extensions TNT [2] introduces goal-based predictions for diversity and DenseTNT [3] further designs dense goal sets and a corresponding optimization pipeline to further improve its performance. LaneGCN [5] introduces a series of four different information aggregations modules for the driving scene. LaneRCNN [6] proposes local LaneROI and based on it goal-based prediction is used. [7] propose to predict multi-modal future conditioned on lane connecting topology. [9] proposes the graph-based heatmap output to optimize the missing rate while [8] designs a hierarchical heatmap paradigm and a learnable trajectory recombination method for joint multi-agent trajectory prediction. SceneTransformer proposes a factorized spatial-temporal network which applies Transformer on the fully-connected spatial/temporal graph alternatively. The concurrent work multipath++ [12] designs gated mechanism to update information and the features are in each agents' local coordinate system.

Comparing to the existing works, we are the first to comprehensively model all node types (agent types such as pedestrians/cyclist/vehicle and map element types such as lanes and crossWalk) and all semantic and geometric relations (such as Pedestrian-CrossWalk and Lane-LeftLane) into one single heterogeneous graph and update the node and edge features in a unanimous and parallel way but with distinct parameters for different node or edge types. The benefits of each design is studied thoroughly in the experiments section.

3 Methodology

We formally introduce our proposed method. The pipeline of HDGT is shown in Fig. 2. The past states of agents and HD-Map are fed as input to the system. The scene is modeled as a heterogeneous graph where the graph construction is described in Sec. 3.1. The spatial feature encoding module converts raw sensor data into vector representations of the local coordinate system (Sec. 3.2). The core module of our approach is to update the node features of elements and the edge relations in the graph, which is depicted in details in Sec. 3.3. Finally, the prediction heads and loss are provided in Sec. 3.4.

Problem Formulation. Given N agents' states, referring to coordinates, velocity, heading, *etc*., in the past L time-steps, we are tasked to predict the future trajectories of agents in the next T time-steps. The scene information is provided as well, including road line (polyline coordinates and type), lane (polyline coordinates, type, and connection with other lanes), traffic lights (coordinate with past L states), *etc*.

3.1 Construction of the Heterogeneous Graph

A directed graph is represented as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of edges. A heterogeneous graph has a node type mapping function of $\tau(v) : \mathcal{V} \rightarrow \mathcal{A}$ and an edge type mapping function $\phi(e) : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{A}$ and $\mathcal{R}$ denote the set of node and edge type respectively.

The type of an edge $e : u \rightarrow v$ is determined by the type of source node u and destination node v and their relations. When building the heterogeneous graph, all possible combinations of semantics relations are considered. For example, there are the following edge types: A-A (agent to agent), A-L (agent to lane), L-leftL (lane to left lane). For an agent node, instead of connecting it to all the other counterparts [1], [5] constrains each node only connecting to those whose distance is smaller than a threshold. However, in some scenarios, vehicles traversed a long

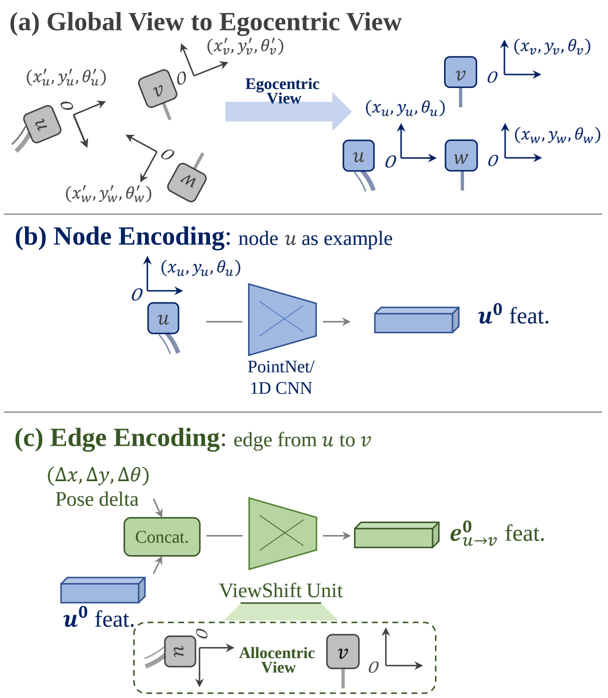


Fig. 3: Detailed elaboration of the Relative Spatial Relation Encoding module. View shift from a global reference to the egocentric coordinate system (a) and spatial information encoding of the node (b) and edge (c).

way in the prediction horizon[2]. It is computationally expensive and unnecessary to set a fixed large threshold for *all* agents. For example, for pedestrians, vehicles very far away would not influence their future behaviors. Thus, similar to [6], for each agent, we set the distance threshold flexible to be its speed times the prediction horizon with a buffer value. For a lane node, besides connecting to agent nodes, it connects to the surrounding lane nodes (left, right, entry, and exit) only.

3.2 Relative Spatial Relation Encoding Module

The Relative Spatial Relation Encoding aodule as described in Fig. 3 is designed to transfer raw data into hidden representative vectors. The outcome of this module serves as the initial node and edge features for the graph neural network.

Note that one should not directly feed raw data to the neural network since the raw spatial data are usually in a global coordinate system defined by datasets. Considering the same set of spatial relations could be represented differently by the coordinate system with different origins and orientations, the spatial data should be normalized into a canonical form. Non-normalized inputs would inappropriately force the network to overfit the absolute value of training data and thus unfavorably cause poor generalization. To normalize the data, one popular choice is to select one agent as a global reference [1], [5]. However, recent works [12], [36], [45] suggested that it is beneficial to aggregate relative spatial features for each node instead of having a global reference. To this end, we follow the intuition as discussed above. Specifically, for each node, we transform the raw spatial feature $\mathbf{f}_{global}$ into ego-centric coordinate system $\mathbf{f}_{ego}$. Fig. 3(a) depicts the view change from a global view to the ego-centric view.

2. Taking Waymo Open Motion as an example, some vehicles moved more than 300 meters in the 8 seconds prediction horizon.

To generate node features, we use (a) 1D CNN to process agents' feature, (b) the simplified PointNet with multi-layer perception (MLP) to process polyline features, and (c) the learnable embedding to deal with type information; all of which are commonly used in previous literature [1], [5]. We denote the initial node feature set as $\{\mathbf{v}^0\}$ for all entries in $\mathcal{V}$. Fig. 3(b) describes the node feature encoding process.

To encode the relative spatial relation between nodes, we let the in-edge feature from node u to v be the concatenation of the source node's feature $\mathbf{u}^0$ with the relative pose change between the source and destination nodes' local reference. As shown in Fig. 3(c), the `ViewShift` unit is responsible for shifting the view from node u's pose to that of v's. In this way, u's characteristics is encoded in v's local coordinate system. It is implemented by the MLP layer which has different parameters for different source node types. For node u and v, there exists a directed edge $e: u \rightarrow v$, and the initial edge feature $e^0_{u\rightarrow v}$ is obtained by:

$$\mathbf{e}^0_{u\rightarrow v} = \mathrm{MLP}_{\tau(u)}\left(\texttt{concat}\left[\mathbf{u}^0, [\Delta x, \Delta y, \sin(\Delta\theta), \cos(\Delta\theta)]\right]\right) \tag{1}$$

where $\Delta x, \Delta y$ are the relative coordinates between u, v, and $\Delta\theta$ is the relative orientation.

In this way, we decouple the encoding process of the motion/geometric characteristics of a single object and the spatial relation between two objects. The inputs for the node feature encoder (1D CNN/PointNet) are all in the ego-centric coordinate system, which satisfies the i.i.d input assumption and is less prone to overfitting absolute coordinate value. The `ViewShift` unit focuses on projecting other objects' features to the target nodes' coordinate system to capture the relative spatial relation.

3.3 Heterogeneous Driving Graph Transformer

The core module of our proposed method is shown in the dashed box of Fig. 2. It aggregates and updates node and edge features with multiple Transformers. This module has K GNN layers in total hence the update process is performed K times accordingly. As in [45], we decouple the feature update for each layer into two parts: aggregate each nodes' in-edge information to update node features based on its neighborhood and update edge features in light of the corresponding source nodes' features.

Node Update. The Transformer structure [13] is adopted as the aggregation function, where we use each node's feature as query and its in-edges' features as keys and values to update the node feature. However, Transformer is initially designed for language data, meaning all elements are homogeneous words. This assumption does not hold true for the driving scene data since in-edges could either be agents (dynamic states) or lanes (polylines). Therefore, we propose a hierarchical manner to adopt Transformer and this is in unanimous spirit with the heterogeneous property of elements. For the aggregation of in-edges' features, each type of in-edges shares a set of parameters of Transformer and does cross-attention with the node. Note that there would be a set of attended features - each for a specific edge type. We concatenate the outcome from different Transformers (shown as h_N in Fig. 2) and use an MLP layer to generate the final updated node features.

Mathematically, for a node v with a set of in-edges $\{e_i\}_{i=0}^{i=M}$ in the k^{th} layer, we update its feature from $\mathbf{v}^{k-1}$ to $\mathbf{v}^k$. Denote the features of all in-edges of v with a specific edge type j as $\mathbf{Y}_j$, we calculate the attention matrix as:

$$\mathbf{A}_j = \texttt{softmax}\left(\frac{\mathbf{Q}_j\mathbf{K}_j^\top}{\sqrt{D}}\right) \tag{2}$$

where $\mathbf{Q}_j = \mathbf{v}^{k-1}\mathbf{W}_{j,qry}$, $\mathbf{K}_j = \mathbf{Y}_j\mathbf{W}_{j,key}$, D is the dimension of the query and key embeddings, $\mathbf{W}_{j,qry}$ and $\mathbf{W}_{j,key}$ are the query and key matrix for edge type j.

The multi-head cross attention is performed as:

$$h_j = \underset{h\in[i,H]}{\texttt{concat}} \left[\mathbf{A}_j^h \mathbf{V}_j^h\right] \mathbf{W}_{j,out} + \mathbf{b}_{j,out} \tag{3}$$

where $\mathbf{V}_j = \mathbf{Y}_j\mathbf{W}_{j,val}$, $\mathbf{W}_{j,val}$, $\mathbf{W}_{j,out}$, $\mathbf{b}_{j,out}$ are the value matrix, output matrix, and output bias for the edge type j respectively, the superscript h denotes the head index, and H is the total number of heads.

An MLP layer with residual (for its simplicity) is adopted to process the attended features from all edge types to get the final updated node feature $\mathbf{v}^k$ as:

$$\mathbf{v}^k = \mathrm{MLP}\left(\underset{j\in[1,J]}{\texttt{concat}}[h_j]\right) + \mathbf{v}^{k-1} \tag{4}$$

where J is the number of edge types for all in-edges of node v.

Edge Update. We concatenate the edge feature with its source node feature and use an MLP layer with residual to obtain the updated edge feature. Note that for different edge types, we adopt different MLPs hence the heterogeneity is ensured. Specifically, at k^{th} layer, for an edge $e : u \rightarrow v$, we update its feature $\mathbf{e}^{k-1}$ to $\mathbf{e}^k$:

$$\mathbf{e}^k = \mathrm{MLP}_{\phi(e)}\left(\texttt{concat}[\mathbf{u}^{k-1}, \mathbf{e}^{k-1}]\right) + \mathbf{e}^{k-1}. \tag{5}$$

Note that the residual shortcut path for both node and edge update is omitted for brevity in Fig. 2.

INTERACTION Dataset consists of various highly interactive driving situations, including highway ramps, roundabouts, and intersections, recorded using drones or fixed cameras worldwide [14]. We use its most recent 1.2 version which provides the official train/val/test split and the labels of test set are held out for the online competition. In this dataset, targets are only vehicles. The input data is 1 second history in 10 Hz and the required output is 3 seconds future in 10Hz. At most 6 modes are allowed.

3.4 Output Head and Training Objective

For the output head, we adopt the commonly used regression - MLP plue classification-MLP combination [5], [6], [10], [36]. Specifically, for each target agent, we use the node feature at HDGT's last layer as its hidden representation. An MLP module is used for regression; it outputs a tensor of size $N \times K \times T \times 2$, where N is the number of target agents, K is the number of required mode, T is the prediction length, and 2 corresponds to (x, y). Another MLP is used to generate the confidence of each mode for each agent - a tensor of size $N \times K$. For different types of agents (vehicles/pedestrians/cyclists), different parameters for the two heads are used to capture the diverse moving behaviors. Denote an agent's outputs as:

$$O_{\mathrm{reg}} = \{(\mathbf{p}_1^k, \mathbf{p}_2^k, \cdots \mathbf{p}_T^k)\}_{k\in[1,K]}, \tag{6}$$

$$O_{\mathrm{cls}} = \{c^k\}_{k\in[1,K]}, \tag{7}$$

where $\mathbf{p}_i^k = (x_i^k, y_i^k)$ is the predicted coordinate of this agent at time step i in the k^{th} mode, and c_k is the confidence.

As for the training objective, for each agent, we only backpropagate loss for the regression head with the lowest regression loss to avoid mode collapse and encourage diversity. For the classification head, we use the cross-entropy loss. The mode with the lowest regression loss is set as the positive sample, and the rest modes are set as negative samples consequently.

Mathematically, for the n^{th} agent, we find the index k_n of predicted mode with lowest regression loss:

$$k_n = \underset{k\in[1,K]}{\mathrm{argmin}} \frac{1}{2T}\sum_{t=1}^{T}\left[d(x_{n,t}^k, x_{n,t}^*) + d(y_{n,t}^k, y_{n,t}^*)\right], \tag{8}$$

where $x_{n,t}^*, y_{n,t}^*$ is the ground truth coordinate at time step t of the n^{th} agent, and we adopt the widely used smooth L1 loss as [49]:

$$d(x_1, x_2) = \begin{cases} 0.5(x_1 - x_2)^2 & \text{if } \|x_1 - x_2\|_1 < 1, \\ \|x_1 - x_2\|_1 - 0.5 & \text{otherwise.} \end{cases} \tag{9}$$

The final training objective can be calculated as:

$$\begin{aligned} \mathcal{L} &= \lambda\mathcal{L}_{\mathrm{cls}} + \mathcal{L}_{\mathrm{reg}} \\ &= \lambda\frac{1}{N}\sum_{n=1}^{N}\mathrm{CrossEntropy}\left(\{c^k\}_n, I(k_n)\right) \\ &\quad + \frac{1}{2NT}\sum_{n=1}^{N}\sum_{t=1}^{T}\left[d(x_{n,t}^{k_n}, x_{n,t}^*) + d(y_{n,t}^{k_n}, y_{n,t}^*)\right], \end{aligned} \tag{10}$$

where $I(k_n)$ is an one-hot vector of length K whose k_n^{th} entry is 1 and λ is the hyperparameter set to be 0.1 throughout the paper.

4 Experiments

We depict the experimental results in details. Sec. 4.1 provides a description of datasets and evaluation metrics and the implementation details on key hyperparameters; the numerical analysis is provided in Sec. 4.2 where state-of-the-art works are compared. The qualitative visualization is presented subsequently in Sec. 4.5 to give the audience a glimpse of what the model learns in various complicated scenarios. Ablation studies follow next in Sec. 4.3 and Sec. 4.4 investigates the model capacity of HDGT.

4.1 Protocols and Implementation Details

Waymo Open Motion Dataset provides a large scale dataset with annotations for objects with interacting behaviors over a wide range of road geometries [11]. We use its most recent 1.2 version with road connection information. It also provides the official train/val/test split and has an online leaderboard for test set. The target agents are divided into 3 types: vehicles, pedestrians, and cyclists. The input is 1.1 seconds history in 10 Hz and required output is 8 seconds future in 2Hz. At most 6 modes are allowed.

We follow the official train/val/test split of the datasets above and report the test set results on the online leaderboards. Similar to [1], [5], we use minADE, minFDE, and MR as the validation metric. Note that both benchmarks have slightly different definitions of the aforementioned metrics. Some other metrics such as minJointADE (minSADE), minJointFDE (minSFDE), SMR [14], [35], and mAP [11] are also used. Therefore, we use the following metrics in the validation set and report the leaderboard results in spirit of the official metrics respectively.

a) **minADE** (Minimum Average Distance Error): the minimum value of the Euclidean distance between the prediction and ground truth averaged by the prediction length T, for K required predictions. b) **minFDE** (Minimum Final Distance Error): similar to minADE, despite that it only calculates the error at the final timestep T. c) **MR** (Missing Rate): the ratio of whether the Euclidean

TABLE 1
Results on the INTERACTION Leaderboard, *test* set. [3]

Method	minADE↓	minFDE↓	MR↓
DenseTNT [3]	0.4342	0.7952	0.0596
MultiModalTransformer [4]	0.2130	0.5511	0.0511
GOHOME [9]	0.2005	0.5988	**0.0491**
HDGT (Ours)	**0.1676**	**0.4776**	0.0556

(a) Single Agent Track

Method	minADE↓	minFDE↓	MR↓
DenseTNT [3]	0.5452	0.6375	0.0348
GOHOME [9]	0.2020	0.5902	0.0284
MultiModalTransformer [4]	0.1989	0.5799	0.0429
HDGT (Ours)	**0.1553**	**0.4577**	**0.0276**

(b) Conditional Single Agent Track

Method	minSADE↓	minSFDE↓	SMR↓
ReCoG2 [46]	0.4668	1.1597	0.2377
DenseTNT [3]	0.4195	1.1288	0.2240
THOMAS [9]	0.4164	0.9679	**0.1791**
HDGT (Ours)	**0.3030**	**0.9580**	0.1938

(c) Multi-Agent Track

Method	minSADE↓	minSFDE↓	SMR↓
ReCoG2 [46]	0.3295	0.8693	0.1498
THOMAS [9]	0.3148	**0.7162**	**0.1067**
DenseTNT [3]	0.2786	0.8916	0.1502
HDGT (Ours)	**0.2255**	0.7875	0.1322

(d) Conditional Multi-Agent Track

TABLE 2
Results on the Waymo Open Motion Leaderboard, *test* set.
*Note that the concurrent MultiPath++ involves ensemble with results shown in *italic*; the proposed HDGT method ranks the first in the single model in terms of minADE/minFDE/MR and is shown in **bold**.

	minADE↓	minFDE↓	MR↓	mAP↑
SimpleCNNOnRaster [47]	0.7400	1.4936	0.2091	0.2136
ReCoAt [48]	0.7703	1.6668	0.2437	0.2711
DenseTNT [3]	1.0387	1.5514	0.1573	0.3281
Kraken-NMS (Yandex SDG)	0.6732	1.3947	0.1850	**0.3646**
SceneTransformer [10]	0.6117	1.2116	0.1564	0.2788
*MultiPath++ [12]	*0.5557*	*1.1577*	*0.1340*	*0.4092*
HDGT (ours)	**0.5933**	**1.2055**	**0.1511**	0.2854

TABLE 3
Results of Model Component Ablation Study. *Relative* means using local coordinate system for each node and its in-edges, otherwise we set the autonomous vehicle as the global reference. *HD-Map* means using the HD-Map information otherwise only agents' information. *Semantic* means building graph according to distance threshold and lane connection information otherwise building a fully-connected graph. *Heterogeneous* means using different parameters for different node and edge types otherwise sharing the parameters. HET: Heterogeneous.

Relative	HD-Map	Semantic	HET	minADE ↓	minFDE ↓	MR ↓
				0.2287	0.5338	0.0153
✓				0.1889	0.4572	0.0093
✓	✓			0.1526	0.3726	0.0044
✓	✓	✓		0.1328	0.3379	0.0038
✓	✓	✓	✓	**0.1071**	**0.2945**	**0.0023**

distance between the prediction and ground truth at the final timestep T for all K predictions is larger than 2m.

Implementation Details. The model is implemented with Pytorch and DGL. We use a hidden dimension of size 128. We use three layers of GNN on INTERACTION, which is similar to [4], [46]. On Waymo Open Motion, we have six layers of GNN (12.1M) containing the same magnitude of parameters with [10] (15.3M). Following the popular recipe for Transformer-based model, we use the AdamW optimizer with an initial learning rate 5e-4, wight decay 1e-4, and batch size of 64. For both datasets, the number of training epochs is 30 with 1 epoch warmup and then linearly decay to 0. All the experiments are conducted on NVIDIA Tesla V100.

4.2 Comparison to State-of-the-arts

Throughout 11/3/2021 (since our final submission to INTERACTION public evaluation) to 4/21/2022 (as of we submit this draft), in terms of minADE/minFDE metric, we rank the **best** result in all four tracks on the INTERACTION Leaderboard[4] and **second** on the Waymo Open Motion Leaderboard[5]. Table 1 and Table 2 report the detailed results and comparison to previous state-of-the-arts.

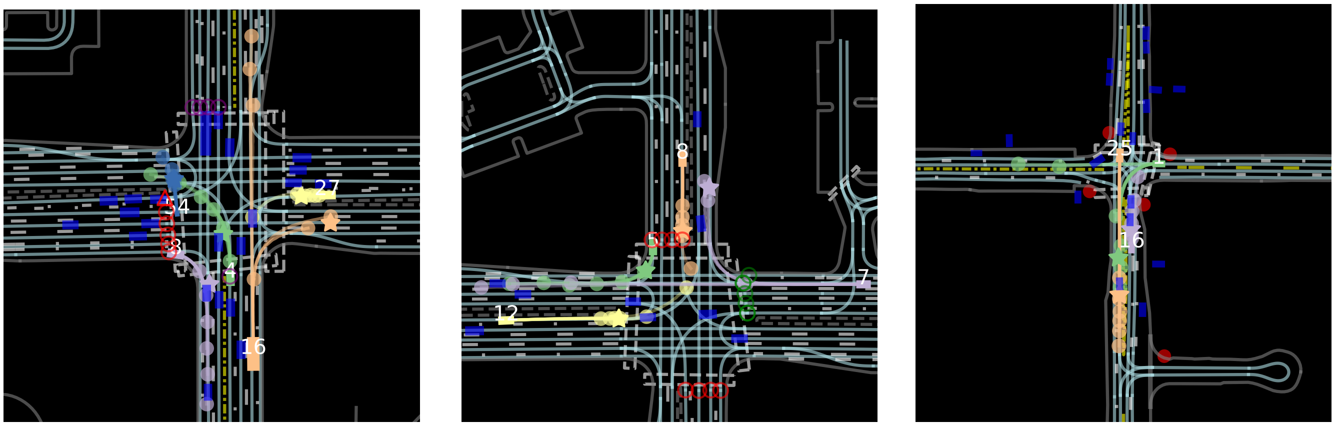
On INTERACTION, one can observe that HDGT outperforms other candidates by a large margin in terms of minADE/minFDE. As for the MR metric, the model still has satisfying results. Note that HDGT's output head is the simple regression and classification scheme, which differs from DenseTNT and GOHOME having specific designs to optimize MR or THOMAS for the joint multi-agent prediction. The experimental results demonstrate the advantage of our HDGT, which utilizes the heterogeneous information as well as the relative coordinate encoding.

On Waymo Open Motion, HDGT ranked 2^{nd} in terms of minADE/minFDE while the 1^{st} solution Multipath++ is an ensemble result. On computational efficiency, compared to the recent SceneTransformer with 15.3M parameters operating on a fully connected graph, HDGT outperforms it with 12.1M parameters. Furthermore, in terms of the MR and mAP [11] metric, HDGT performs on par with previous light head design methods, including ReCoAt, SimpleCNNOnRaster, and SceneTransformer; while it performs worse than DenseTNT, Kraken-NMS, and MultiPath++, all of which have optimization steps for mAP improvement.

4.3 Ablation Study

We conduct the ablation study to validate the effectiveness of each module in HDGT and the results are shown in Table 3. The experiments are conducted on the validation set of INTERACTION dataset and we keep all the other settings the same. Note that all models have enough capacities to overfit the training set and we report their best performance on the validation set. From the results, we can conclude that:

3. Note that there are four tracks in the INTERACTION challenges with different data and settings. We adopt their settings by: for the two conditional tracks which additionally give the future trajectory of the ego agent, we ignore the additional information for the consistency with other datasets and experiments. For the two joint tracks which require joint multi-agent prediction, we let all predictions with the same index as one modality.
4. http://challenge.interaction-dataset.com/prediction-challenge/intro
5. https://waymo.com/open/challenges/2021/motion-prediction/



(a) **P54** was passing the road through the crosswalk while **V4** desired to turn left. The model outputs their different potential futures. For **V16**, since its target is unclear by observation, the model outputs diverse futures: go straight, yield, or do right turn.

(b) **P7** could either go straight through the green light or turn right. Thus, the prediction of **P5** is influenced. For **V8** and **V12**, they were far away from the traffic light and thus their predictions include both passing and yielding.

(c) **V1**'s future had high uncertainty (going straight or left turn) and it was also influenced by either **V25** or **V16** under different situations. Also, the existing of stop sign (red double ring) results in conservative predictions for **V16**.

Fig. 4: **Visualization of Prediction Results on Waymo Opem Motion Dataset.** ⋆ represents the ground-truth final position and ○ denotes predicted final positions of modes (lower confidence, more transparent). Abbreviation: V-vehicle, P-pedestrian, and C-cyclist.

TABLE 4
Performance by different numbers of GNN layers. For Waymo, the minADE/minFDE is calculated in their official way.

Dataset	INTERACTION		Waymo	
Layer #	minADE ↓	minFDE ↓	minADE ↓	minFDE ↓
1	0.1430	0.3997	0.6462	1.3099
2	0.1149	0.3243	0.6013	1.2004
3	**0.1071**	**0.2945**	0.5835	1.1736
6	0.1082	0.3035	**0.5673**	**1.1507**

- *Relative representation in the allocentric view is better.* In model 1, we set the autonomous vehicle as the global reference while in model 2 we adopt the proposed node-centric local coordinate system. The improvement of the model 2 shows the necessity of learning relative spatial relations instead of absolute coordinates, which aligns well with the conclusions in [12], [36], [45].
- *Road semantics from HD-map are important.* In model 3, we add HD-Map and construct a homogeneous fully-connected graph similar to VectorNet [1]. It has better performance compared to model 2, which proves the importance of HD-map information for the trajectory prediction[6].
- *Semantic relations are helpful.* In model 4, instead of a fully-connected graph, we build the semantic graph as proposed in HDGT but the GNN parameters for different types of nodes are still shared. As a result, it brings improvement compared to model 3 while worse than model 5.
- *Heterogeneous graph network parameters are vital.* Model 5 is the full version of HDGT and we can find that specific parameters for different types of nodes and edges could boost the performance significantly compared to model 4.

6. For example, drivers tend to drive following the lane centerline. When it comes to parking, they would stop nearby the curb.

4.4 Exploration on Model Capacity

We examine the scalability of HDGT. We conducted experiments on the validation set of INTERACTION and Waymo Open Motion. From Table 4, we can observe that for INTERACTION (40K tracks), the performance saturates with around 3 layers of HDGT. When it comes to the larger Waymo Open Motion (7.64M tracks), the performance could be still improved even with 6 layers. This shows that HDGT enjoys the strong scalability of Transformer structure, which brings huge success in the vision [50] and language [51] fields. The simple and yet unified structure of HDGT makes it easy to extend its capacity when there is more data, which fits perfectly for the autonomous driving industry.

4.5 Visualization

To give an overview, we visualize the prediction results of HDGT in some scenarios on the Waymo Open Motion dataset in Fig. 4. One can conclude that in the HDGT's multi-mode predictions, the effect from a range of different elements such as agents, lanes, stop-signs, traffic lights, and *etc* are considered. Thus, the explicit modeling of the heterogeneous nature of the driving scene is significant and beneficial for the trajectory prediction.

5 Conclusion

In this paper, we have proposed a principled approach to extract better representation for trajectory prediction. It models the driving scene as a heterogeneous graph and adapt transformer as the graph aggregation function. The spatial features are normalized into the local coordinate system when being aggregated. The proposed approach has achieved state-of-the-art results on two recent large-scale and competitive benchmarks. Ablation studies validate the effectiveness of each module in the proposed method.

Acknowledgement

This work was in part supported by National Key Research and Development Program of China (2020AAA0107600), NSFC

(U19B2035, 61972250, 72061127003), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102). The authors are thankful to the anonymous reviewers.

References

[1] 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 *CVPR*, 2020.

[2] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid *et al.*, "Tnt: Target-driven trajectory prediction," *arXiv preprint arXiv:2008.08294*, 2020.

[3] J. Gu, C. Sun, and H. Zhao, "Densetnt: End-to-end trajectory prediction from dense goal sets," in *ICCV*, 2021.

[4] Z. Huang, X. Mo, and C. Lv, "Multi-modal motion prediction with transformer-based neural network for autonomous driving," *arXiv preprint arXiv:2109.06446*.

[5] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," in *ECCV*, 2020.

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

[7] N. Deo, E. Wolff, and O. Beijbom, "Multimodal trajectory prediction conditioned on lane-graph traversals," in *CoRL*, 2021.

[8] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Thomas: Trajectory heatmap output with learned multi-agent sampling," *arXiv preprint arXiv:2110.06607*, 2021.

[9] ——, "GOHOME: graph-oriented heatmap output for future motion estimation," *arXiv preprint arXiv:2109.01827*, 2021.

[10] 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.

[11] 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.

[12] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov *et al.*, "Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction," *arXiv preprint arXiv:2111.14973*, 2021.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[14] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof, C. Stiller, A. de La Fortelle *et al.*, "Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps," *arXiv preprint arXiv:1910.03088*, 2019.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[16] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web*, A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, and M. Alam, Eds., 2018.

[17] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *SIGKDD*, 2019.

[18] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *WWW*, 2020.

[19] Y. Cao, H. Peng, J. Wu, Y. Dou, J. Li, and P. S. Yu, "Knowledge-preserving incremental social event detection via heterogeneous gnns," in *The Web Conference*, 2021.

[20] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *CVPR*, 2016.

[21] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *CVPR*, 2018.

[22] B. Ivanovic, E. Schmerling, K. Leung, and M. Pavone, "Generative modeling of multimodal multi-human behavior," in *IROS*, 2018.

[23] B. Ivanovic and M. Pavone, "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs," in *ICCV*, 2019.

[24] X. Jia, L. Sun, M. Tomizuka, and W. Zhan, "Ide-net: Interactive driving event and pattern extraction from human data," *arXiv preprint arXiv:2011.02403*, 2020.

[25] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, "Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction," in *CVPR*, 2020, pp. 14 424–14 432.

[26] C. Tang and R. R. Salakhutdinov, "Multiple futures prediction," in *NeurIPS*, 2019.

[27] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. K. Chandraker, "DESIRE: distant future prediction in dynamic scenes with interacting agents," in *CVPR*, 2017.

[28] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, "Multi-agent tensor fusion for contextual trajectory prediction," in *CVPR*, 2019.

[29] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," in *CoRL*, 2019.

[30] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *ICRA*, 2019.

[31] J. Hong, B. Sapp, and J. Philbin, "Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions," in *CVPR*, 2019.

[32] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[33] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control," *arXiv preprint arXiv:2001.03093*, 2020.

[34] Y. Biktairov, M. Stebelev, I. Rudenko, O. Shliazhko, and B. Yangel, "PRANK: motion prediction based on ranking," *arXiv preprint arXiv:2010.12007*, 2020.

[35] S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun, "Implicit latent variable model for scene-consistent motion forecasting," in *ECCV*, 2020.

[36] X. Mo, Y. Xing, and C. Lv, "Heterogeneous edge-enhanced graph attention network for multi-agent trajectory prediction," *arXiv preprint arXiv:2106.07161*, 2021.

[37] F. Marchetti, F. Becattini, L. Seidenari, and A. D. Bimbo, "Mantra: Memory augmented networks for multiple trajectory prediction," in *CVPR*, 2020.

[38] C. Choi, J. H. Choi, J. Li, and S. Malla, "Shared cross-modal trajectory prediction for autonomous driving," in *CVPR*, 2021.

[39] M. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *CVPR*, 2019.

[40] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, 1997.

[42] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

[43] M. Ye, T. Cao, and Q. Chen, "Tpcn: Temporal point cloud networks for motion forecasting," in *CVPR*, 2021.

[44] M. Ye, J. Xu, X. Xu, T. Cao, and Q. Chen, "Dcms: Motion forecasting with dual consistency and multi-pseudo-target supervision," *arXiv preprint arXiv:2204.05859*, 2022.

[45] X. Jia, L. Sun, H. Zhao, M. Tomizuka, and W. Zhan, "Multi-agent trajectory prediction by combining egocentric and allocentric views," in *CoRL*, 2021.

[46] X. Mo, Y. Xing, and C. Lv, "Recog: A deep learning framework with heterogeneous graph for interaction-aware trajectory prediction," *arXiv preprint arXiv:2012.05032*, 2020.

[47] S. Konev, K. Brodt, and A. Sanakoyeu, "Motioncnn: A strong baseline for motion prediction in autonomous driving," *Workshop on Autonomous Driving, CVPR*, 2021.

[48] Z. Huang, X. Mo, and C. Lv, "Recoat: A deep learning framework with attention mechanism for multi-modal motion prediction." *Workshop on Autonomous Driving, CVPR*, 2021.

[49] R. Girshick, "Fast r-cnn," in *ICCV*, 2015.

[50] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, "Scaling vision transformers," *arXiv preprint arXiv:2106.04560*, 2021.

[51] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.