A Novel Network Based on Dynamic Graph Convolution and Multi-head Attention Mechanism for End-to-End Multimodal Trajectory Prediction

Hai-Sheng Li, Hui-Jia Ma, and Cong Hu

***Abstract*—Accurately predicting the future trajectories of agents in complex traffic scenarios is crucial for achieving intelligent transportation. However, existing trajectory prediction methods face challenges, such as modeling long-term dependencies and neglecting the impact of dynamic driving scenarios on trajectory prediction. We propose a novel trajectory prediction model named DG-Trajector to address these issues. This model leverages information from multimodal data through dynamic graph convolution networks and an end-to-end learning approach. The model employs dynamic graph convolution layers (DGConvLayer) to capture the correlations between trajectories flexibly. DGANet can adaptively focus on the importance of different parts by introducing a multi-head attention mechanism into the trajectory data. The features extracted by DGANet predict both endpoint goals and trajectories, achieving more accurate trajectory prediction by utilizing goal-driven prediction methods and considering information at different levels. Specifically, on the Argoverse dataset, The DG-Trajector achieves improvements over the benchmark network, while adding MR Indicators and expanding multimodal prediction capabilities. The experimental results of DG-Trajector show that the performance of multi-mode prediction is also very competitive. Its effectiveness and feasibility are verified in complex traffic scenarios.**

***Index Terms*—Autonomous Vehicles, Dynamic graph convolution, Multimodal motion prediction, End-to-end learning.**

I. Introduction

AUTONOMOUS driving agents must make safe and effective decisions in complex traffic scenarios, such as lane changes, overtaking, and deceleration. Existing tactical path planning algorithms [1] rely on reliable estimation of future trajectories of surrounding agents (e.g., vehicles and pedestrians). Consequently, predicting trajectories for adjacent vehicles has garnered increasing attention and application. For instance, location-based traffic recommendation services rely heavily on predicting vehicle behaviors in real time. This enables features like dynamic lane change suggestions and personalized risk assessments. Therefore, accurate and efficient vehicle trajectory prediction is crucial for deploying traffic control and management systems [2]. Effectively leveraging dynamic information in trajectories and integrating local and global information to enhance prediction accuracy and robustness pose significant challenges in trajectory prediction tasks. With the rapid development of graph neural networks, researchers have found that their application to trajectory prediction [3] has a good effect. As for how to model interaction and scene dynamics, research on interactive perception methods based on deep learning can be divided into two categories: one is non-graph-based trajectory prediction, Deo et al. [4] extract vehicle interaction features from gridded road scenes using convolutional networks. However, this method is limited by the accuracy of maneuver type classification and the inflexibility of fixed grid scenes. The second category is trajectory prediction based on graphs. Typically, existing methods represent input data as graph structures and predict trajectories by learning relationships between nodes in the graph through graph convolutional networks. VectorNet [5] proposed a new feature representation method by vectorizing trajectories, effectively capturing the interaction relationships between trajectories. Subsequently, a series of graph-based trajectory prediction models were proposed. Li et al. [6], [7] introduced the GRIP method, which combines graph operations with convolutional layers, treating driving scenes as graph models and representing vehicle interactions as correlations between nodes in the graph model. Although this method effectively captures spatial correlations between vehicles, it still lacks the extraction of time-related features. Chandra et al. [8] proposed a spectral clustering-based framework that can simultaneously predict vehicle trajectories and driving behaviors. However, this method constructs a global graph model of all historical driving scenes, resulting in huge and sparse corresponding adjacency matrices, leading to low computational efficiency. There is little research on dynamic graph processing in trajectory prediction [9]. To better extract time-related features, Malik et al. [10] constructed dynamic graphs as dynamic tensors and extracted time features through mathematical operations on three-dimensional tensors. Unfortunately, the authors used only single-layer graph convolution models to meet the needs of applications such as social networks and financial transactions. This design is unsuitable for traffic scenarios due to the complexity and realism of urban traffic environments.

Manuscript received April 20, 2025; revised September 1, 2025. This work was supported by a grant (Nos. BCIC-25-Y1 and BCIC-25-Y3) from Guangxi Key Laboratory of Brain-inspired Computing and Intelligent Chips, the Science and Technology Project of Guangxi under Grant Nos. 2024GXNSFAA010524, the Project of Guangxi Key Laboratory of Automatic Detection Technology and Instruments under Grant no. YQ24204.

Hai-Sheng Li is a Professor at Guangxi Key Laboratory of Brain-inspired Computing and Intelligent Chips, Guangxi Normal University, Guilin, Guangxi 541004, China (corresponding author, e-mail: lhs_ecjtu@126.com).

Hui-Jia Ma is a postgraduate at School of Electronic and Information Engineering, Guangxi Normal University, Guilin, Guangxi 541004, China (e-mail: 728518157@qq.com).

Cong Hu is a Professor of Guangxi Key Laboratory of Automatic Detection Technology and Instruments, Guilin University of Electronic Technology, Guilin, Guangxi 541004, China (e-mail: hucong@guet.edu.cn).

Despite some advanced methods, there are still many limitations. These methods often fail to fully exploit dynamic information in trajectory sequences and struggle to capture long-term dependencies between trajectories. We propose a novel dynamic graph network model called DG-Trajector to address these challenges. This model combines dynamic graph convolutional networks (GCNs) and multi-head attention mechanisms, effectively capturing dynamic information and long-term dependencies in trajectory sequences.

The core of the DG-Trajector model, DGANet, is a multi-layer dynamic graph convolutional network incorporating a multi-head attention mechanism. It dynamically constructs graph structures based on input data and extracts feature representations of trajectory sequences through multi-layer graph convolution operations. Unlike traditional static graph networks, DGANet dynamically adjusts the graph connections according to the dynamic characteristics of trajectory sequences, thereby better capturing the dynamic relationships between trajectories. Additionally, the multi-head attention mechanism introduced by DGANet effectively captures important information in trajectory sequences, focusing on significant trajectory features. Notably, DG-Trajector is not limited to dynamic graph convolutional network methods but also integrates endpoint generators and trajectory prediction modules. By predicting trajectories step by step, our method comprehensively addresses trajectory prediction tasks while maintaining model efficiency and interpretability. Our contributions can be summarized as follows:

- We propose a new method based on dynamic graph convolutional networks that effectively captures dynamic information and long-term dependencies in trajectory sequences.
- We utilize multi-layer dynamic graph convolution and multi-head attention mechanisms to extract and integrate both local and global trajectory information effectively, comprehensively modeling traffic scenes and improving prediction accuracy and robustness.
- Our method is evaluated on real datasets and demonstrates strong performance in the trajectory prediction task, validating its effectiveness and reliability.

II. Related work

Recent social interaction model methods for vehicle trajectory prediction based on machine learning can be divided into two categories: rasterization and vectorization.

A. Rasterized Representation

Rasterization relies on grid-based operations to simulate social interactions, transforming real-world scenes into discrete grid cells. Traditional trajectory prediction models mainly utilize rasterized scene representations [11], [12], [13] to depict context and interactions. Social-LSTM [14] is an extension based on LSTM networks, modeling social trajectory data to predict future trajectories; however, LSTM fails to capture interaction effects between vehicles. To address this limitation, Deo et al. [15] proposed a unified framework to identify the maneuvers of surrounding vehicles on highways and predict vehicle trajectories. They employ the convolutional social pooling mechanism to learn interdependencies between vehicles. These methods convert continuous scenes in high-precision maps into discrete grid representations, leading to a loss of accuracy and an inability to fully capture the details and continuity of the real world. Additionally, rasterization results in a large amount of data redundancy, increasing the burden of data storage and processing, and also incurs rendering losses. In comparison to rasterized representations, the Bird's Eye View (BEV) representation is a more advanced and flexible mapping method. BEV representation [16] converts maps from two-dimensional grids into more abstract topological structures to better describe objects and their relationships in the environment. However, generating BEV representations requires rasterization, inevitably resulting in information loss [17].

B. Vectorized Representation

In vectorized map representation [5], [18], geographical spatial information is decomposed into a series of vector features, such as points, lines, and polygons. These features can represent the position, shape, attributes, and other information of geographical objects. Using a graph structure to describe the relationships between these geographical features, social interactions are modeled within the graph structure. The development of vectorized representation has greatly helped address information loss and data redundancy, while also significantly promoting the development of graph-based methods. Researchers have been exploring graph-based methods to capture spatial dependency relationships between vehicles. Compared to rasterized representations, graph neural networks support a more explicit interaction modeling, such as lane graphs or vectorized representations of scenes [5]. Building on their success in capturing hierarchical representations, recent works have employed GNNs for interaction modeling [19], [20]. However, due to the limited perception range of sensors in autonomous vehicles, once these models are determined, they cannot adapt to changes in vehicle composition in traffic scenes. In complex traffic scenarios, traffic graphs exhibit heterogeneity across different time periods. In our work, we also adopt vectorized representation; unlike them, we utilize dynamic graphs to model social interactions, which can more efficiently capture vehicle interaction information in highly dynamic traffic scenes.

C. Dynamic graph convolution

Dynamic Graph Convolution is a convolution operation used to process dynamic graph data. In traditional graph convolutional networks, the graph's structure is static, meaning that the nodes and edges of the graph do not change during training. In contrast, in dynamic graphs, the graph's structure changes over time, with nodes and edges able to change with each time step. Dynamic graph convolution [21], [22] models dynamic graph data by introducing a temporal dimension. It associates node features with time steps and utilizes node connections for information propagation and feature updates. Specifically, dynamic graph convolution first performs convolution operations on node features at

each time step, then updates the features of the current node by aggregating the features of neighboring nodes. This effectively captures temporal relationships and evolution patterns between nodes in dynamic graphs.

This technology has been widely applied in social networks, financial transactions, and traffic flow prediction [23], [24]. There are some similarities among these application scenarios, where the features of objects represented by graph models change dynamically, and the relationships between objects change dynamically. Therefore, effectively modeling dynamic graphs and extracting spatiotemporal features is one of the main challenges. Dynamically learned adjacency matrices from data exhibit more accurate effects on related traffic prediction problems, improving the accuracy and effectiveness of these tasks. Combining dynamic graph convolution, many excellent models have been successively proposed and applied to traffic prediction. Seo et al. [25] constructed a graph convolutional recurrent network for graphs, where features change over time but edges remain fixed. EdgeConv, proposed by Wang et al. [26], is a neural network that dynamically applies convolution operations on static graphs. Zhao et al. [27] developed a time GCN method called T-GCN for traffic prediction. To capture graph structures and handle temporal dynamics, Li et al. [28] combined a fixed adjacency matrix with a self-attention mechanism to generate a dynamic adjacency matrix to capture changes in spatio-temporal features and improve the accuracy of lane-level traffic prediction. Sankar et al. [29] proposed using a time self-attention layer to learn representations of dynamic graphs.

However, in trajectory prediction tasks, most GCN-based for trajectory prediction adopt static adjacency matrices, assuming that the correlation between nodes does not change over time. This adjacency matrix is often manually designed by researchers based on actual traffic conditions, with the most common being the 0-1 matrix representing the connection relationship between road nodes and the distance matrix representing the actual geographical distance between nodes. For example, the GRAPH-LSTM [30] and PRIME [31] models use binary tuples (A, D) as adjacency representations, where A is the adjacency matrix between road nodes (0-1 matrix), indicating whether there is a connection between nodes on the road, and D is a diagonal matrix indicating the degree of each node on the road. These matrices fail to capture the complex relationships and spatial dependencies between nodes. Because they treat the graph as a fixed-size matrix, they cannot adapt to changes in the graph structure over time. Therefore, traditional graph convolutional networks cannot handle dynamic graph data directly. Capturing vehicle interaction information in highly dynamic traffic scenarios remains a major challenge. Inspired by the successful application of dynamic graph convolution methods in the field of traffic prediction [28], [32], [33], we introduce dynamic graph convolution to deal with frequently changing graphs. To this end, we introduce dynamic graph convolution combined with self-attention mechanisms to deal with changing graphs. In this paper, we apply dynamic graph convolution to the trajectory prediction task and propose the DGANet module, which can effectively process dynamic graph data.

D. Multi-head Attention Mechanism

The Multi-Head Attention Mechanism, originally propsed by Vaswani et al. in [34], allows the model to compute attention across different positions in multiple representation subspaces. Specifically, it splits the input feature vector into multiple heads (typically 8 or more), each of which learns an independent set of attention weights. Then, the attention results of each head are concatenated together and passed through a linear transformation to generate the final output. Specifically, assuming the input features are X, after linear transformation, we obtain queries Q, keys K, and values V, where the parameters of the linear transformation are denoted as W_q, W_k and W_v, and the number of attention heads is h. The input first undergoes linear projection through linear layers, then attention scores are calculated for each query and key pair, and values are weighted summed, resulting in the output of each head, generating h attention values, where h is the number of heads. Finally, the outputs of each head are concatenated or weighted summed to obtain the final multi-head attention output. Attention score calculation adopts the scaled dot-product attention mechanism, and the computation of multi-head attention is as follows:

$$Attention(Q, K, V) = soft\max(\frac{QK^T}{\sqrt{d_k}})V \qquad (1)$$

where d_k is the dimension of keys, i.e., the input dimension of each head divided by the number of heads.

E. Trajectory prediction

In recent years, trajectory prediction has been studied extensively. Traditional methods typically rely on physics-based models or statistical methods. However, these methods often struggle to accurately describe target behaviors in complex environments and are sensitive to noise and other disturbances. To address the challenges posed by the limitations of traditional approaches, such as handling variable target behaviors, high uncertainty, and significant noise interference, target-driven prediction methods have emerged. Target-driven prediction involves predicting future trajectories based on given target points or regions through the analysis of historical data. For instance, Ziebart et al. [35] utilized reinforcement learning algorithms to predict future trajectories and learned from inferred reward functions. The inferred reward function provides destination-related information, and trajectory planning is conducted through Inverse Reinforcement Learning (IRL). Rehder et al. [36] introduced the concept of target-oriented pedestrian prediction, where machine learning algorithms first predict pedestrian targets and then guide trajectory formation based on these targets. Mangalam et al. [37] proposed an endpoint-conditioned trajectory prediction method, utilizing latent representations learned from Conditional Variational Autoencoder (CVAE) to generate endpoints, guiding predictions using multiple probabilistic anchor trajectory hypotheses [38]. R2P2 [39], [40] guided the path prediction process by introducing target point information, generating paths more aligned with practical requirements. PRIME [31] directly searched for a set of reachable paths and generated possible trajectory anchors in a planning manner. LTP [41] initially

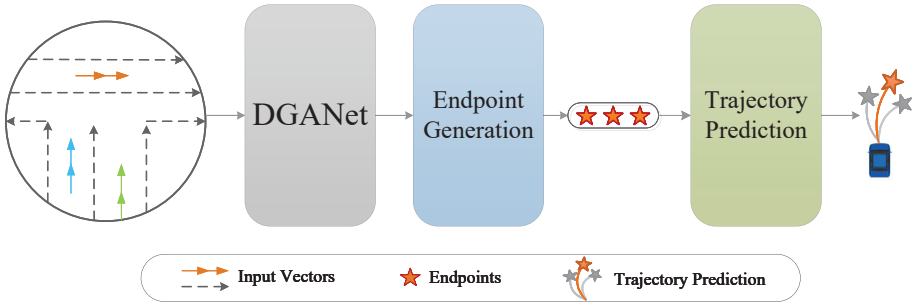

Fig. 1: Overview of DG-Trajector.

predicted possible target lanes and positions, and after obtaining target information, utilized conditional generation models to generate conditional trajectories. Furthermore, to consider uncertainty and multimodality in prediction space, generative adversarial networks (GAN) [42] and variational autoencoders (VAE) [43] were used to sample the latent space to generate multiple trajectory predictions. TNT [44] employs a goal-driven approach to predict future trajectory. The decoder receives the information from the encoding stage as input and calculates the probability distribution of the future position through forward propagation, which can better capture the time dependence and dynamic change characteristics.

Inspired by target-driven prediction methods [40], [41], [42], [43], [44], similarly, we use a goal-driven prediction method to predict trajectories, thereby improving prediction accuracy.

III. Method

This section outlines the overall framework of our proposed model, the DG-Trajector network architecture, as illustrated in Figure 1. In the vectorized scene representation, DGANet encodes and extracts features from vectorized polyline features. The endpoint generation module generates a set of candidate endpoints. In the input trajectory prediction module, a set of candidate trajectories is generated and scores are computed for each trajectory. Finally, K trajectories are selected based on trajectory scoring, where orange represents the optimal trajectories. Specific design details will be elaborated in the subsequent subsections of this section.

A. Feature encoding

Polyline Encoding: We use a structured vector representation to represent maps and agents. The vector representation proposed by VectorNet [5] creates a connected graph for each scene element independently. Given the agent's past trajectory, $A = \{a_i\}$, where $a_i \in R^{T\times 2}$, T is the number of time steps in the trajectory, and each time step contains the agent's position (x, y) on the HD map. For lane data, represented as $M = \{m_i\}$, where $m_i \in R^{l_i\times 2}$, l_i denotes the number of continuous points included in lane i For each scene element, we convert its trajectory data into polylines. The polyline encoding for each agent and lane is independent, and therefore, each is represented using separate subgraphs (see Figure 2). The length of each polyline is d, and its feature vector is F_i^d. Based on the

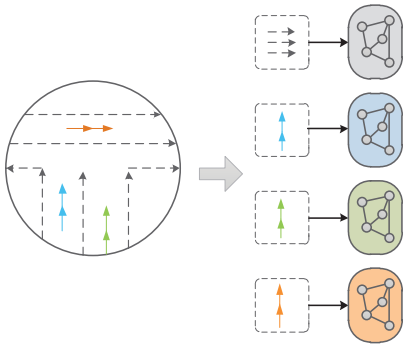

Fig. 2: Polyline encoding module.

continuous points of the trajectory data, we calculate the center point and direction of the polyline, considering them as part of the polyline's feature vector. Additionally, we include the corresponding type identifier (agent or lane) as part of the feature vector.

B. Interaction Module

DGANet presented in Figure 3, an essential component of the DG-Trajector, is meticulously designed to capture local structural information among points in polylines effectively. It consists of key multi-layer DGConvLayers and a multi-head attention mechanism. The multi-layer DGConvLayers propagate and aggregate information layer by layer through dynamic graph convolution, gradually extracting higher-level feature representations. Normalization operations are applied to the features via linear layers to enhance their representational capability. Figure 4 illustrates the specific details of DGConvLayer. Since trajectory prediction tasks require considering temporal dependencies, dynamic graph convolution can adaptively

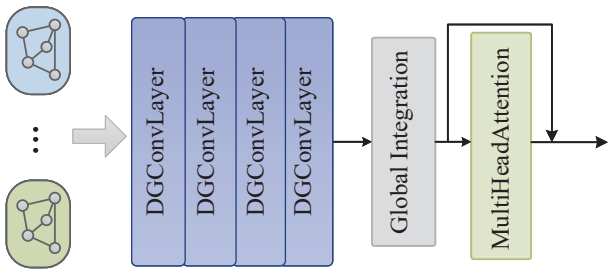

Fig. 3: DGANet.

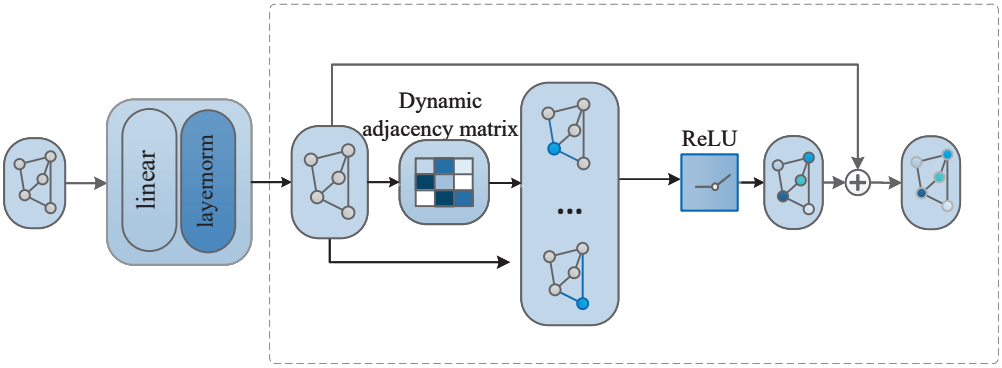

Fig. 4: DGConvLayer.

generate dynamic adjacency matrices based on dynamic changes in the input data, thereby better modeling the dynamic relationships and spatiotemporal dependencies within the data. In the DGConvLayer, original features are transformed into higher-level abstract representations via linear layers to enable the network to learn richer feature information. Meanwhile, normalization operations are applied to the features to enhance their representational capability, eliminate differences in scales among features. According to the length d of each polyline and the feature dimension $\dot{d}$ of each polyline point, an adjacency matrix $\hat{A}$ of the shape $(d, \dot{d})$ is initialized to preliminarily represent the relationships between points in the polyline. The softmax operation is performed on matrix $\hat{A}$ to generate attention scores, determining the degree of association between each node and other nodes, so that weighted consideration can be given during information propagation to better utilize the information of the graph structure. After transposing the node feature tensor F and multiplying it with the attention scores, the information of each node is propagated and aggregated, effectively integrating information among nodes and generating node feature representations F_{wei}, which are modulated by attention mechanisms. At each time step T, the weights of the dynamic adjacency matrix $\hat{A}$ are adjusted based on the input feature tensor to better capture time-series information. The formulas are as follows:

$$\hat{A} = f(F, T) \tag{2}$$

$$\hat{A}_{att} = \{soft\max(\hat{A})\} \tag{3}$$

$$F_{wei} = \hat{A}_{att}(F^T) \tag{4}$$

where F represents the feature tensor of trajectory points, the function f denotes the process of generating the dynamic adjacency matrix.

The dynamic adjacency matrix $\hat{A}$ is passed to the GCN module for graph convolution. Weighted propagation based on $\hat{A}$ helps each node better utilize information from neighboring nodes, thereby enhancing the feature representation capability. Features generated at each layer are concatenated with the input features of the current layer, effectively incorporating residual connections, aggregating features to retain important information, and mitigating the vanishing gradient problem. To enhance the model's representational power, multiple dynamic graph convolution layers are stacked. The output of each layer serves as the input for the next layer, gradually extracting higher-level feature information. Thus, the concatenated polyline features proceed to the next DGConvLayer, and this process is repeated four times to extract higher-level semantic features. The operation formula for the DGConvLayer is as follows:

$$F' = F + GCN(F_{wei}, \hat{A}) \tag{5}$$

By adjusting the adjacency matrix with a self-attention mechanism, different trajectory data can be adaptively modeled accordingly, and the spatiotemporal dynamic characteristics of trajectory data can be better understood and predicted. At the same time, high-level abstract features of trajectory data are extracted step by step through multi-layer dynamic graph convolution to represent the complexity and diversity of trajectory data, thus improving the representation capability and prediction accuracy of the model.

After multiple layers of feature extraction and temporal information integration, the feature representation possesses richer semantic information and better temporal modeling capabilities. The features of all polylines are aggregated into a tensor through batch encoding and aggregation operations, which is then input input as a whole into the global layer (see Figure 5).

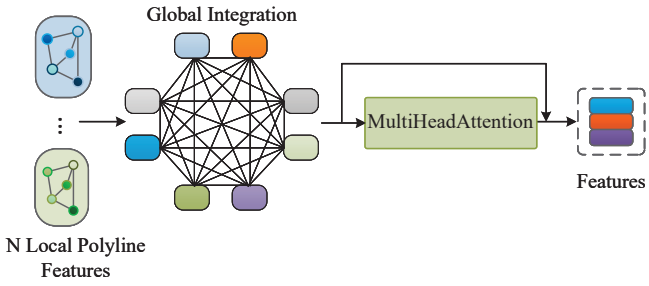

Fig. 5: Global layer.

In the global layer, this tensor represents the entire graph and applies the multi-head attention mechanism to capture global relationships and contextual information between polylines. The calculation formula for the multi-head

attention mechanism is as follows:

$$head(h) = soft\max(\frac{VQ_h(L_g) \times VK_h(L_g)}{\sqrt{d}})VV_h(L_g) \tag{6}$$

where $VQ_h(L_g)$, $VK_h(L_g)$ and $VV_h(L_g)$ are the linear projections of the node feature matrix $V(L_g)$ for head h. The global features generated after the multi-head attention mechanism form a tensor that contains the global contextual information and relationships between the input polylines. These global features are concatenated with the input polyline features along the feature dimension. The final output is a feature tensor of shape (64, 128), where 128 is the output feature dimension of the linear layer. This concatenation operation is similar to residual connections but not identical. In residual connections, the original input is combined with the processed output through simple element-wise addition or concatenation to better preserve the original input information. Here, the concatenation operation connects the two along the feature dimension. The global features provide overall information for all polylines, while the original polyline features contain local features for each polyline. By concatenating them along the feature dimension, the model can consider both global and local information, leading to a more comprehensive understanding of the data.

Compared to the standard multi-head attention mechanism, the multi-head attention in dynamic graph convolution considers the influence of the dynamic adjacency matrix when calculating attention weights. This allows attention calculation to adapt to the dynamic changes in the input data, better capturing the temporal correlations. DGANet, combining dynamic graph convolution and multi-head attention, can fully leverage local polyline features and global polyline relationships, achieving multi-level information fusion. This results in richer and more expressive feature representations.

C. Endpoint and Trajectory prediction

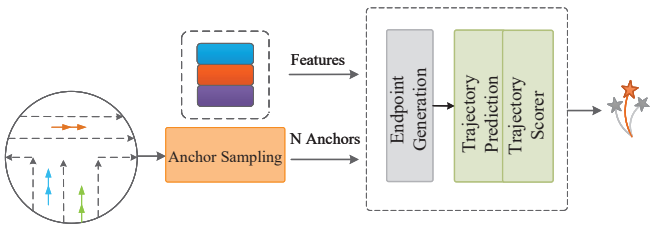

Fig. 6: Endpoint and Trajectory prediction.

We use endpoint generation and trajectory prediction modules to improve trajectory prediction, integrating information from different levels. The endpoint generation module takes the features extracted by DGANet and sampled anchors as input, generates a set of candidate endpoints, and selects the top N. The trajectory prediction module receives these N endpoints and combines them with the input features to create a set of candidate trajectories. These trajectories are scored by a trajectory scorer, and the top K trajectories with the highest scores are selected as the final prediction results. Figure 6 shows the structure of the endpoint generation and trajectory prediction modules. The detailed design of these modules will be elaborated in the following subsections.

D. Endpoint generation

Firstly, lane centerline points are sampled as candidate anchor positions (see Figure 7). These centerlines represent the paths along which vehicles might travel on the road. For each candidate anchor, the vehicle's driving path is approximated using polyline or curve methods based on nearby lane centerline points. When using the polyline sampling algorithm (e.g., straight driving), straight segments are generated between lane centerline points to approximate the vehicle's path. In contrast, when using the curve sampling algorithm (e.g., turning), polynomial fitting methods are employed to generate curves. Sampling algorithms are used to determine the specific positions of the sampled points. The polyline sampling algorithm samples a fixed number of points at equal intervals along each polyline as anchors. The curve sampling algorithm fits curves to the polylines and then samples a fixed number of points at equal intervals along the fitted curves as anchors. The points sampled from each polyline are combined to form a set of anchor points. These anchors represent possible vehicle trajectories, with the final anchor points representing the possible final positions of the vehicle. In practice, the differences between each candidate anchor and others are calculated to ensure no duplicates are generated. The resulting anchor points are then used in the subsequent endpoint generation process.

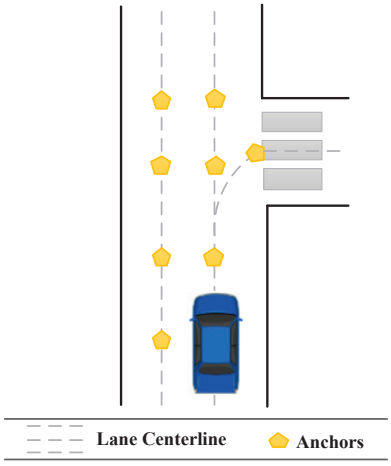

Fig. 7: Anchor sampling.

The endpoint generation module in Figure 8 consists of an endpoint corrector, a confidence evaluator, and a series of linear layers and activation functions. The endpoint corrector and the confidence evaluator are responsible for generating endpoint corrections and confidence scores, respectively. The linear layers and activation functions merge the feature representation with anchor point positions through a series of linear and nonlinear transformations, producing an intermediate representation for the endpoint generation module. This intermediate representation contains preliminary prediction information about the endpoint positions, providing a foundation for subsequent endpoint correction and confidence evaluation. The endpoint corrector adjusts the endpoint positions based on the intermediate representation. It maps the intermediate representation to the correction space through a series of linear transformations and activation functions to obtain the final endpoint corrections. These corrections are used to refine

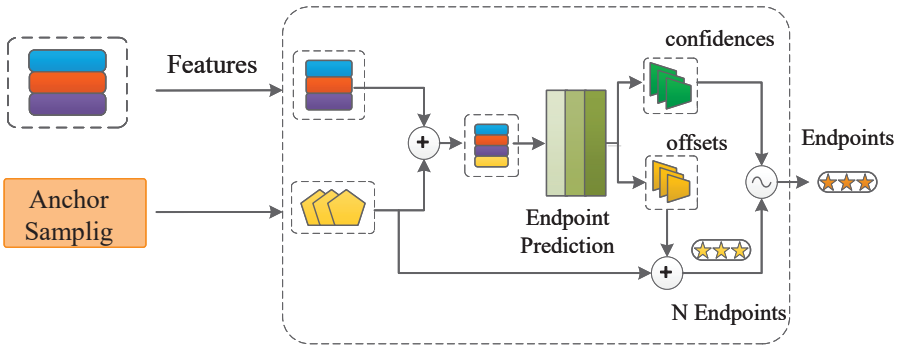

Fig. 8: Endpoint generator.

the preliminary endpoint predictions, making them more accurate and reliable.

$$EndPoint = anchor + offset \tag{7}$$

The confidence evaluator is used to assess the confidence level of each endpoint position. It maps the intermediate representation to the confidence score space through a series of linear transformations and activation functions, obtaining the confidence score for each endpoint position. This confidence score represents the model's confidence in predicting the endpoint position, where higher scores indicating greater confidence in the prediction at that position. Endpoints are then filtered based on their confidence scores, selecting the top-N endpoints with the highest confidence. Together, these two components constitute the endpoint generation module, which completes the prediction and evaluation of endpoint positions through a series of interactive operations and information propagation, providing crucial support and foundation for target-driven prediction tasks.

1) Trajectory prediction: The trajectory prediction module, as presented in Figure 9, uses the given endpoints to forecast trajectories and comprises a trajectory predictor and a trajectory scorer.

The trajectory predictor forecasts trajectories based on the provided global features and endpoints. Initially, it concatenates the global features with the target position to effectively integrate the features. Then, it transforms the features through a series of linear layers and activation functions to produce the predicted trajectories. During prediction, the trajectory predictor also utilizes skip connections between endpoint positions and features to enhance information propagation and prediction performance. The Trajectory Scorer evaluates the quality of predicted trajectories by receiving global features and predicted trajectories as input and outputting trajectory scores after a series of processing. These scores represent the model's assessment of the quality of each predicted trajectory, with higher scores indicating better trajectory quality. The design of the trajectory scorer aims to improve the accuracy and robustness of the model's trajectory predictions. Together, these components constitute the endpoint-driven prediction module, playing a crucial role in model training and inference. Through a series of interactive operations and information exchanges, they jointly accomplish trajectory prediction and evaluation, providing an effective solution for trajectory prediction tasks.

E. Loss Functions

In our DG-Trajector approach, three key loss functions are employed: endpoint loss, prediction loss, and prediction score loss. These loss functions play a crucial role during training, aiding the model in learning and optimization. The endpoint loss is used to quantify the accuracy of endpoint position estimation. It consists of two parts: confidence loss L_{conf} and offset loss L_{offset}.The specific formulas is as follows:

$$L_{conf} = BCE(P_{closest_{onehot}}, \hat{P}_{confs}) \tag{8}$$

$$L_{offset} = H_{\delta}(P_{closest_{offset}}, \hat{P}_{closest_{offset}}) \tag{9}$$

$$L_{EndPoint} = L_{conf} + L_{offset} \tag{10}$$

where $P_{closest}$ represents the element extracted from the endpoint set P_p that is closest to the true endpoint P_{gt}, H_{δ} is the Huber loss parameterized by δ. $P_{closest_{offset}}$=P_{gt}−$\hat{P}_{closest_{offset}}$, is the predicted endpoint

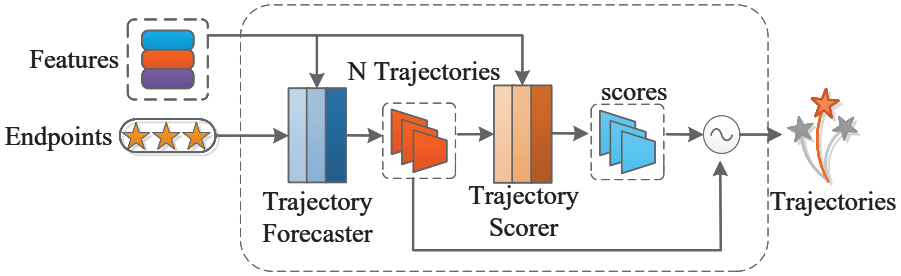

Fig. 9: Trajectory Prediction.

offset. BCE denotes binary cross-entropy loss, $P_{closest_{onehot}}$ is the index of the $P_{closest}$ element represented in one-hot format, and $\hat{P}_{confs}$ is the confidence of each predicted endpoint, ranging between 0 and 1. The endpoint loss $L_{EndPoint}$ helps the model accurately locate the endpoints by comparing the model's predicted endpoint positions with the true positions. It encourages the model to output high-confidence endpoint positions.

The prediction loss measures the distance between the trajectory predicted by the model and the true trajectory. It utilizes the Huber loss to balance the accuracy and robustness of the predicted trajectory, aiding the model in better predicting the motion trajectory of endpoints.The calculation formula is as follows:

$$L_{trajde} = H_{\delta}(T_{traj}, \hat{T}_{traj}) \tag{11}$$

where T_{traj} is the true trajectory, and $\hat{T}_{traj}$ is the predicted trajectory, during trajectory evaluation training, $\hat{T}_{traj}$ adopts the true endpoints of the trajectory to stabilize learning.

The prediction scoring loss is used to evaluate the model's confidence in predicting trajectories. It first represents the similarity between trajectories by calculating the maximum distance between all paired points of the predicted trajectory and the true trajectory, i.e.:

$$D(T_{traj}, \hat{T}_{traj}) = \max(\left\| T_k^{traj} - \hat{T}_k^{traj} \right\|_2^2) \tag{12}$$

The magnitude of the negative value is usually related to the degree of error. Therefore, by computing the softmax of the negative values and using it as the true distribution $P_{traj_{confs}}$, reliable trajectory confidence is trained as:

$$L_{sorce} = BCE(P_{traj_{confs}}, \hat{P}_{traj_{confs}}) \tag{13}$$

The prediction scoring loss helps the model better estimate the quality of the predicted trajectories, improving the model's robustness in complex scenarios. By considering the endpoint loss, prediction loss, and prediction scoring loss comprehensively, the DG-Trajector model can optimize target prediction and trajectory prediction tasks comprehensively, enhancing the model's performance and generalization ability, thus better adapting to various complex real-world scenarios.

IV. Experiment

A. Datasets

The Argoverse prediction dataset [45] is provided for future trajectory prediction. This dataset contains 333,000 sequences, each 5 seconds long, with a trajectory sampling frequency of 10Hz. In this dataset, the observation time is 0.2 seconds, and the future prediction time is 2.5 seconds. Argoverse comprises 323,557 scenes and is commonly used as a benchmark for single-agent motion prediction. Considering high-definition maps and 20 seconds of historical information, the objective is to predict the position of targets within the next 3 seconds. Additionally, the dataset includes 62,022 multi-agent scenes, with up to 40 agents per scene. In these scenarios, the goal is to predict the trajectories of agents in the future.

B. Evaluation Metrics

In trajectory prediction tasks, we adopt widely used metrics such as minimum average displacement error ($mADE_K$), final displacement error ($mFDE_K$), and miss rate (MR_K). These metrics are computed based on the endpoints closest to the true trajectories among K trajectory predictions. Specifically, $mADE_K$ measures the average L2 distance between complete predictions and ground truth values. It calculates the average distance between each predicted trajectory and the true trajectory and then takes the average over all predicted trajectories. $mFDE_K$ is used to measure the difference between predicted endpoints and ground truth values. It computes the distance between the endpoint of each predicted trajectory and the endpoint of the true trajectory and then takes the minimum over all predicted trajectories. MR_K measures the proportion of cases, based on FDE, where all predictions fall outside a 2-meter range of the true ground. This metric can be used to evaluate the model's performance under large error conditions.

C. Experimental details

We trained our DG-Trajector model on an NVIDIA 4090 GPU with 24GB of memory. Adam optimizer was used for training, PyCharm Community Edition 2024.1 served as the development tool, and the experiments were conducted using the Python programming language. We set batch to 8 and epoch to 50 in the initial training, using a learning rate of 0.0001.

D. Performance

In this section, we compare the proposed DG-Trajector in detail with other methods on the Argoverse validation set. The results are shown in detail in Table I and Table II.

TABLE I: Comparisons with advanced algorithms in Argoverse leaderboard

Method	$mADE_{K=1}$	$mFDE_{K=1}$	$MR_{K=1}$
Argoverse Baseline[45]	3.45	7.88	0.87
TPNet[40]	1.75	3.88	-
Vector Net[5]	**1.66**	3.67	-
Ours	1.68	**3.62**	**0.61**

TABLE II: Comparisons with advanced algorithms in Argoverse leaderboard

Method	$mADE_{K=6}$	$mFDE_{K=6}$	$MR_{K=6}$
Argoverse Baseline[45]	1.71	3.29	0.54
SMART[49]	1.44	2.47	-
R2P2[38]	1.40	2.35	-
Luo[48]	1.35	2.68	-
WIMP[44]	1.30	2.46	0.33
TPNet[40]	1.28	1.91	-
DiversityGAN[41]	1.13	1.78	-
TNT-Henry1iu[31]	1.11	2.12	0.31
Half-normal[46]	1.09	2.07	0.38
DESIRE[47]	1.09	1.89	-
MTPLA[11]	1.05	**1.56**	-
BVN[46]	1.04	1.98	0.35
Ours	**1.02**	1.71	**0.28**

We compared the DG-Trajector with existing methods having similar configurations. Results in Table I and Table II, presented on the validation set for $K = 1$ and

$K = 6$, include the Argoverse official baseline and other candidate-based trajectory prediction methods similar to the proposed architecture. To ensure fairness in comparison, we take all data from the original papers and highlight the best results for each metric in bold. In Table I, the DG-Trajector model indicators were on par with VectorNet in terms of accuracy with $mADE_{K=1}$=1.68 and $mFDE_{K=1}$=3.62. Compared with the traditional LSTM Baseline (Argoverse Baseline), $mADE$ (3.45) and $mFDE$ (7.88) were reduced by 51% and 54% respectively, verifying the robustness of the dynamic graph convolutional network in capturing the kinematic characteristics of vehicles. It was worth noting that in terms of the loss rate (MR) index, DG-Trajector reached an MR of 0.61, indicating that the model effectively reduced the probability of the trajectory deviating from the feasible area. Compared with TPNet (with an $mFDE$ of 3.88 in Table 1), DG-Trajector achieved a 6.7% improvement in the end point accuracy ($mFDE$) through the implicit multimodal coding mechanism (3.62 vs. 3.88), demonstrating the generalization ability of the dynamic graph structure in complex interaction scenarios.

In the multimodal prediction tasks in Table II (K=6), the advantages of DG-Trajector were more significant. Its $mADE$=1.02 was 1.9% lower than 1.04 of the current Bayesian variational network BVN, while its $mFDE$=1.71 was 3.9% optimized compared with 1.78 of DiversityGAN. This indicated that the model could not only cover diverse motion patterns in multi-candidate trajectory generation, but also accurately capture the physical laws of the main modes. In terms of the loss rate index, the MR of DG-Trajector was 0.28, which was 26.3% lower than 0.38 of Half-normal, highlighting the synergy effect of its spatio-temporal joint modeling and uncertainty quantification module. Compared with SMART and DESIRE, the DG-Trajector reduced the fusion ability of multi-scale spatiotemporal features through the multi-head attention mechanism by 29.1% and 6.4% respectively on the $mADE$ index. Furthermore, the DG-Trajector demonstrated outstanding stability in long-term prediction tasks. Its $mFDE$ index was 17.0% lower than that of MTPLA, verifying the advantages of the dynamic graph structure in time series modeling. It was worth noting that the the $mADE$ and MR of DG-Trajector was the best among all the comparison methods, which indicated that DG-Trajector had the competitive ability to capture the main mode while considering the multimodality of the predicted trajectory.

E. Ablation Study

We conducted ablation studies on the DGANet module on the Argoverse validation set to explore the impact of the module on performance.

As shown in Table III, the introduction of dynamic graph convolution and multi-head attention mechanism significantly improves the model performance. When DGANet is not enabled, the model performs weakly in indicators such as the minimum average displacement error in orbit, the final displacement error and the loss rate, and the parameter scale is small. After introducing Dynamic graph convolution (DGConv), the model performance has been substantially improved. $mADE_{K=6}$ decreased from 1.15 to 1.12 (a decrease of 2.6%), and $mFDE_{K=6}$ decreased from 1.95 to 1.85 (a decrease of 5.13%), verifying the adaptive modeling ability of the dynamic adjacency matrix for spatial interaction relationships. Further analysis shows that DGConv can effectively identify the game intentions of conflicting vehicles at intersections by adjusting the interaction weights among agents in real time. However, at this time, MR only slightly decreased from 0.40 to 0.38, reflecting that there are still bottlenecks in the modeling of the long-term spatiotemporal dependence of a single dynamic graph convolution. After superimposing the multi-head Attention mechanism (MHAT), the model has made breakthrough progress in trajectory multimodal modeling. MR dropped sharply to 0.30 (a decrease of 21.05%), and $mFDE_{K=6}$ was further optimized to 1.78 (a decrease of 3.8%). This was attributed to the focusing ability of the attention mechanism on key spatiotemporal features, confirming the resource efficiency advantage of the attention module. Finally, after introducing the residual connection and normalization operation, the comprehensive performance of the complete DGANet model reaches the optimum. $mADE_{K=6}$ was further reduced to 1.02 (a decrease of 4.67%), and MR was further reduced to 0.28 (a decrease of 6.67%), proving that the residual structure effectively alleviated the gradient decay problem of deep networks. The computational overhead brought by increasing the number of parameters to 496K was still within an acceptable range.

The complete results of the ablation experiment show that dynamic graph convolution enhances the spatial relationship modeling ability through the adaptive adjustment of interaction weights, while the multi-head attention mechanism solves the generation problem of multimodal trajectories through feature focusing. The synergistic effect of the two enables the model to achieve the optimal balance between accuracy and scene coverage ability. DGANet effectively improves the accuracy of trajectory prediction and scene coverage ability through the adaptive modeling of interaction relationships by dynamic graph convolution and the focusing of key spatiotemporal features by the multi-head attention mechanism. Although the increase in the number of parameters may bring computational overhead, the marginal benefit of its performance improvement is significant. Especially in complex dynamic scenes, the model's modeling ability for long-term dependencies and multimodal interactions has been substantially enhanced,

TABLE III: DGANet ablation on the argoverse dataset

Component	$mADE_{K=6}$	$mFDE_{K=6}$	$MR_{K=6}$	$Param(K)$
Baseline	1.15	1.95	0.40	162
DGANet(DGConv)	1.12	1.85	0.38	284
DGANet(DGConv+MHAT)	1.07	1.78	0.30	438
DGANet(DGConv+MHAT+Resconn)	1.02	1.71	0.28	496

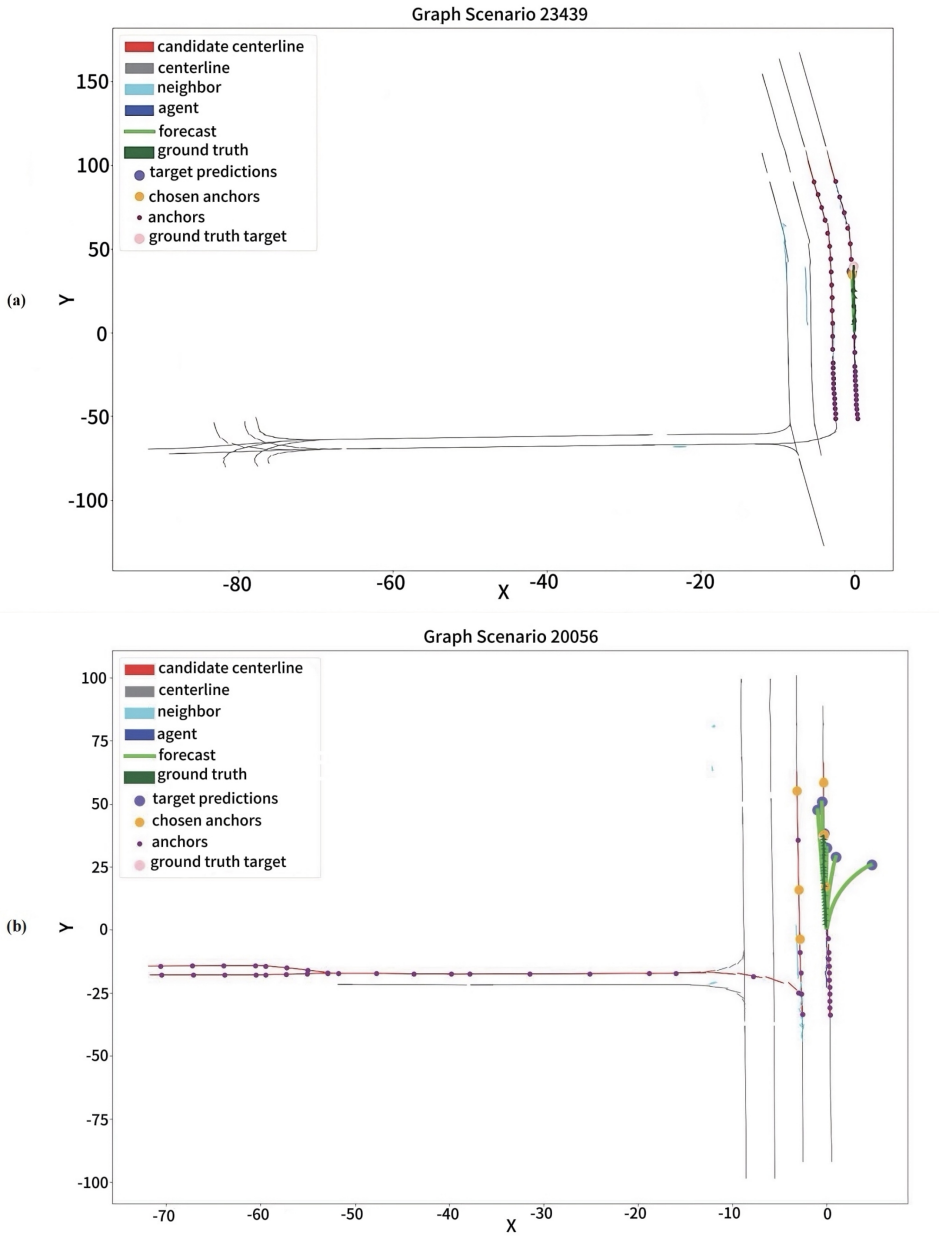

Fig. 10: Qualitative results of DG Trajector on Argoverse validation set.

verifying the rationality and necessity of the module design.

F. Visualizations

In Figure 10, panels (a) and (b) show the prediction results for $K = 1$ and $K = 6$, respectively. Gray lines represent lanes around the target vehicle; dark blue represents observed historical trajectories of the agents; green lines represent multiple predictions; and dark green represents the true future trajectory. These visualization results clearly demonstrate that our DG-Trajector can effectively predict intelligent agents in both single-modal and multimodal scenarios. They confirm the effectiveness of DGANet in accurately predicting vehicle trajectories in different scenarios while capturing dynamic motion characteristics, indicating its effectiveness for trajectory prediction tasks.

V. Conclusion

In this work, we propose a novel and effective trajectory prediction model, DG-Trajector, which achieves outstanding performance in complex interaction scenarios. The DGANet module combines dynamic adjacency matrix generation and multi-head attention mechanisms to extract features and model dynamic relationships in the input trajectory data, effectively capturing its spatiotemporal correlations. Experiments show that compared with the baseline model, DG-Trajector improves performance at mFD$E_{K=1}$ and reduces the MR index. More importantly, DG-Trajector achieves multimodal prediction, and even compared with many advanced methods, DG-Trajector's performance is competitive, with low mAD$E_{K=1}$ and MR, It is particularly worth noting that its dynamic adjacency matrix generation mechanism, through robust modeling of vehicle kinematic characteristics, further reduces the MR Index to 0.28 in the multimodal prediction task (K=6), indicating that our method has higher accuracy and lower error. However, at the same time, it also brings challenges. The model complexity and environmental adaptability still need to be optimized,

and it will take a long time for repeated training and adjustment. One promising direction for future work could be to further optimize the model. For example, the efficiency and accuracy of the model can be improved by introducing more complex attention mechanisms or incorporating more prior information. Another promising direction is to explore the application of trajectory prediction models in multimodal data, better understand the relationships between different modalities, and improve the accuracy and robustness of trajectory prediction.

Data availability

The datasets generated during or analyzed during the current study are available from the corresponding author upon reasonable request.

References

[1] S. Sivaraman and M. M. Trivedi, "Dynamic Probabilistic Drivability Maps for Lane Change and Merge Driver Assistance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, pp. 2063-2073, 2014.

[2] X. Quan, A. Halik, and Y. Ma, "Regularized Multi-Scale Spatial-Temporal Attention with Dual Decoders for Traffic Flow Prediction," *IAENG International Journal of Computer Science*, vol. 52, no. 5, pp. 1570-1584, 2025.

[3] L. Geng, W. Yang, and Jiao, "A Multilayer Human Motion Prediction Perceptron by Aggregating Repetitive Motion," *Machine Vision and Applications*, vol. 34, 2023.

[4] N. Deo and M. M. Trivedi, "Convolutional Social Pooling for Vehicle Trajectory Prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1468-1476.

[5] J. Gao, C. Sun, H. Zhao, Y. Shen, and D. Anguelov, "VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11525-11533.

[6] X. Li, X. Ying, and M. C. Chuah, "GRIP: Graph-Based Interaction-Aware Trajectory Prediction," in *2019 IEEE Intelligent Transportation Systems Conference*, 2019, pp. 3960-3966.

[7] X. Li, X. Ying, and M. C. Chuah, "GRIP++: Enhanced Graph-Based Interaction-Aware Trajectory Prediction for Autonomous Driving," *arXiv Preprint arXiv:1907.07792*, 2019.

[8] R. Chandra, T. Guan, S. Panuganti, and T. Mittal, "Forecasting Trajectory and Behavior of Road-Agents Using Spectral Clustering in Graph-LSTMs," *IEEE Robotics and Automation Letters*, 2020.

[9] F. Manessi, A. Rozza, and M. Manzo, "Dynamic Graph Convolutional Networks," *Pattern Recognition*, 2020.

[10] O. A. Malik, S. Ubaru, L. Horesh, M. E. Kilmer, and H. Avron, "Dynamic Graph Convolutional Networks Using the Tensor M-Product," *arXiv:1910.07643*, 2019.

[11] C. Luo, L. Sun, D. Dabiri, and A. Yuille, "Probabilistic Multi-Modal Trajectory Prediction with Lane Attention for Autonomous Vehicles," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 2370-2376.

[12] H. Cui, V. Radosavljevic, and F. C. Chou, "Multimodal Trajectory Predictions for Autonomous Driving Using Deep Convolutional Networks," in *International Conference on Robotics and Automation*, 2019, pp. 2090-2096.

[13] Z. He, Y. Li, H. Li, and N. Xu, "Multi-Objective Optimization for Online Train Trajectory Planning with Moving Window Method," *IAENG International Journal of Computer Science*, vol. 50, no. 3, pp. 1074-1082, 2023.

[14] A. Alahi, K. Goel, and V. Ramanathan, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 961-971.

[15] N. Deo and M. M. Trivedi, "Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver Based LSTMs," in *IEEE Intelligent Vehicles Symposium*, 2018, pp. 1179-1184.

[16] M. S. Shirazi and B. T. Morris, "Trajectory Prediction of Vehicles Turning at Intersections Using Deep Neural Networks," *Machine Vision and Applications*, vol. 30, pp. 1097-1109, 2019.

[17] M. Liang, B. Yang, and R. Hu, "Learning Lane Graph Representations for Motion Forecasting," in *Computer Vision-ECCV*, Springer International Publishing, 2020, pp. 541-556.

[18] J. Schmidt, J. Jordan, and Gritschneder, "CRAT-Pred: Vehicle Trajectory Prediction with Crystal Graph Convolutional Neural Networks and Multi-Head Self-Attention," in *International Conference on Robotics and Automation*, 2022.

[19] B. Liu, J. Wang, and Z. Wang, "Combined Multilevel Detection and Trajectory Prediction with FairMOT for Pedestrian Multi-Object Tracking," *IAENG International Journal of Computer Science*, vol. 52, no. 4, pp. 1137-1147, 2025.

[20] H. Sen, "Time Series Prediction Based on Improved Deep Learning," *IAENG International Journal of Computer Science*, vol. 49, no. 4, pp. 1133-1138, 2022.

[21] F. Manessi, A. Rozza, and M. Manzo, "Dynamic Graph Convolutional Networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp. 3552-3559.

[22] Z. Wu, S. Pan, F. Chen, and G. Long, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 4-24, 2024.

[23] L. Zhao, Y. Song, and Zhang, "T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, 2019.

[24] X. Huang, Y. Ye, and X. Yang, "Multi-View Dynamic Graph Convolution Neural Network for Traffic Flow Prediction," *Expert Systems with Applications*, 2023.

[25] Y. Seo, M. Defferrard, and P. Vandergheynst, "Structured Sequence Modeling with Graph Convolutional Recurrent Networks," in *Neural Information Processing 25th International Conference*, 2018, pp. 362-373.

[26] Y. Wang, Y. Sun, and Z. Liu, "Dynamic Graph CNN for Learning on Point Clouds," *ACM Transactions on Graphics*, vol. 38, 2019.

[27] L. Zhao, Y. Song, and C. Zhang, "T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, 2019.

[28] B. Li, Q. Yang, J. Chen, D. Yu, D. Wang, and F. Wan, "A Dynamic Spatio-Temporal Deep Learning Model for Lane-Level Traffic Prediction," *Journal of Advanced Transportation*, vol. 2023, no. 1, 2023.

[29] A. Sankar, Y. Wu, L. Gou, and Zhang, "Dysat: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 519-527.

[30] H. Cheng and W. Liao, "Exploring Dynamic Context for Multi-Path Trajectory Prediction," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 12795-12801.

[31] J. Liu, X. Mao, Y. Fang, D. Zhu, and M. Q. H. Meng, "A Survey on Deep-Learning Approaches for Vehicle Trajectory Prediction in Autonomous Driving," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, 2021, pp. 978-985.

[32] K. G. Quach, P. Nguyen, and Le, "DyGLIP: A Dynamic Graph Model with Link Prediction for Accurate Multi-Camera Multiple Object Tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13784-13793.

[33] C. Gao, J. Zhu, and F. Zhang, "A Novel Representation Learning for Dynamic Graphs Based on Graph Convolutional Networks," *IEEE Transactions on Cybernetics*, vol. 53, 2022.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[35] B. D. Ziebart and N. Ratliff, "Planning-Based Prediction for Pedestrians," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[36] E. Rehder and H. Kloeden, "Goal-Directed Pedestrian Prediction," in *IEEE International Conference on Computer Vision Workshops*, 2015.

[37] K. Mangalam, H. Girase, H. Agarwal, K. Lee, E. Adeli, J. Malik, and A. Gaidon, "It Is Not the Journey but the Destination: Endpoint Conditioned Trajectory Prediction," *arXiv:2004.02025*, 2020.

[38] C. Choi, A. Patil, and S. Malla, "Drogon: A Causal Reasoning Framework for Future Trajectory Forecast," *arXiv:1908.00024*, 2019.

[39] N. Rhinehart and K. M. Kitani, "R2P2: A Reparameterized Pushforward Policy for Diverse, Precise Generative Path Forecasting," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 772-788.

[40] L. Fang, Q. Jiang, and J. Shi, "TPNet: Trajectory Proposal Network for Motion Prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6796-6805.

[41] J. Wang, T. Ye, and Y. Gu, "LTP: Lane-Based Trajectory Prediction for Autonomous Driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[42] X. Huang and S. G. McGill, "DiversityGAN: Diversity-Aware Vehicle Motion Prediction via Latent Semantic Sampling," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5089-5096, 2020.

[43] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker, "DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 336-345.
[44] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, A. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, C. Li, and D. Anguelov, "TNT: Target-Driven Trajectory Prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.
[45] M. F. 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
[46] J. Strohbeck, J. Müller, M. Herrmann, J. Büchner, and M. Buchholz, "Deep Kernel Learning for Uncertainty Estimation in Multiple Trajectory Prediction Networks," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 11396-11402.
[47] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker, "DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 336-345.
[48] C. Luo, L. Sun, D. Dabiri, and A. Yuille, "Probabilistic Multi-Modal Trajectory Prediction with Lane Attention for Autonomous Vehicles," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 2370-2376.
[49] N. Sriram, B. Liu, F. Pittaluga, and M. Chandraker, "SMART: Simultaneous Multi-Agent Recurrent Trajectory Prediction," in *Proceedings of the European Conference on Computer Vision*, 2020, pp. 463-479.