sensors

MDPI

Article

Query-Informed Multi-Agent Motion Prediction

Chong Guo [1,2], Shouyi Fan [1], Chaoyi Chen [1], Wenbo Zhao [3], Jiawei Wang [1], Yao Zhang [1] and Yanhong Chen [1,*]

[1] College of Automotive Engineering, Jilin University, Changchun 130025, China; guochong@jlu.edu.cn (C.G.); syfan22@mails.jlu.edu.cn (S.F.); chency21@mails.jlu.edu.cn (C.C.); wjw20@mails.jlu.edu.cn (J.W.); zyao18@mails.jlu.edu.cn (Y.Z.)
[2] Changsha Automobile Innovation Research Institute, Changsha 410005, China
[3] FAW Car Co., Ltd., Changchun 130015, China; zhaowenbo@fawcar.com.cn
* Correspondence: yhchen@jlu.edu.cn

Abstract: In a dynamic environment, autonomous driving vehicles require accurate decision-making and trajectory planning. To achieve this, autonomous vehicles need to understand their surrounding environment and predict the behavior and future trajectories of other traffic participants. In recent years, vectorization methods have dominated the field of motion prediction due to their ability to capture complex interactions in traffic scenes. However, existing research using vectorization methods for scene encoding often overlooks important physical information about vehicles, such as speed and heading angle, relying solely on displacement to represent the physical attributes of agents. This approach is insufficient for accurate trajectory prediction models. Additionally, agents' future trajectories can be diverse, such as proceeding straight or making left or right turns at intersections. Therefore, the output of trajectory prediction models should be multimodal to account for these variations. Existing research has used multiple regression heads to output future trajectories and confidence, but the results have been suboptimal. To address these issues, we propose QINET, a method for accurate multimodal trajectory prediction for all agents in a scene. In the scene encoding part, we enhance the feature attributes of agent vehicles to better represent the physical information of agents in the scene. Our scene representation also possesses rotational and spatial invariance. In the decoder part, we use cross-attention and induce the generation of multimodal future trajectories by employing a self-learned query matrix. Experimental results demonstrate that QINET achieves state-of-the-art performance on the Argoverse motion prediction benchmark and is capable of fast multimodal trajectory prediction for multiple agents.

Keywords: autonomous vehicles; trajectory prediction; query-informed; multimodal

Citation: Guo, C.; Fan, S.; Chen, C.; Zhao, W.; Wang, J.; Zhang, Y.; Chen, Y. Query-Informed Multi-Agent Motion Prediction. *Sensors* **2024**, *24*, 9. https://doi.org/10.3390/s24010009

Academic Editors: Peter Han Joo Chong and Gregor Klančar

Received: 31 October 2023
Revised: 7 December 2023
Accepted: 17 December 2023
Published: 19 December 2023

1. Introduction

Multi-object motion prediction is an essential step in autonomous driving. It aids autonomous vehicles in making informed decisions and prevents accidents. However, traffic scenes are highly complex, involving targets, road networks, and their interactions. Prediction models need to take these entities as inputs and output reasonable multimodal trajectories that intelligent agents may take in the future.

Recently, deep learning-based methods have shown promising results in motion prediction tasks [1–4]. Some studies rasterize scenes into top–down views and employ CNNs for prediction [1,5,6]. While these methods are easily implementable with off-the-shelf image models, they have limited applicability and come with a high cost. Given these constraints, recent research [2,4] has adopted a vectorization approach for a more efficient scene representation, extracting a set of vector nodes from the trajectories of traffic participants and map elements. Subsequently, to learn relationships between vectorized entities, such as trajectory waypoints and lane segments, some studies [7–9] use graph neural networks to process scenes, some studies [10] use transformers to process scenes, others [11] use point cloud models to process scenes. In addition, some research [12–14]

has focused on the vulnerability of data-driven algorithms, Autoencoder, and GAN for trajectory prediction.

However, during the scene encoding process, the perspective of predicting the scene varies for each target. Most existing methods transform the entire scene with respect to one target agent at a time, which leads to asymmetry for other agents. This approach has lower prediction efficiency and is less robust to co-ordinate transformations. To address these issues, HiVT [15] uses rotation invariant and shift invariant scene representations, which greatly enhances the robustness of the prediction model. Additionally, many studies simply use displacement to encode vectors without considering other physical information of scene entities, such as speed, heading angle, and spatial relationships between entities. This oversight may lead to suboptimal trajectory predictions. In the decoder section, the predicted trajectories should be multimodal. However, current research [16] mostly employs non-maximum suppression (NMS) methods, setting a certain threshold, and filtering out multimodal trajectories based on L2 distances between future trajectory endpoints, which yields unsatisfactory results.

We employ a symmetric scene representation using HiVT, in which all relative features possess translational and rotational invariance. Building upon this, we introduce a query-based framework for inducing the generation of multimodal trajectories: QINET. In terms of scene encoding, our representation method incorporates additional relevant features about the target and lane segments to convey their physical significance. In the trajectory prediction decoder section, to enhance the multimodality of output trajectories, we introduce self-learnable parameter matrices for cross-attention over the agent vehicle's history, which we refer to as the query mechanism.

QINET consists of an encoder module and a decoder module. In the encoder section, we expand the scene representation of vectorized entities, incorporating new features related to targets and lane segments. We use projected vector representations for all relevant features of the traffic scene at each timestamp, providing a more detailed description of relative spatial relationships. In the decoder section, we develop a query-informed architecture that combines features from agent history, agent relationships, and the road graph as a target-centric scene representation. This is achieved through querying and self-attention learning to extract and combine scene features. The combined feature representation is used to induce the output of multimodal trajectories. Some feature representation comes from the agent's historical trajectory features, while another part comes from the output of the encoder section, which we refer to as anchor points. In this way, the network first learns to generate scene modal features with maximum diversity without environmental constraints, and then decodes future trajectories after absorbing anchor points carrying rich target-oriented contextual information.

Our contributions can be summarized as follows: first, we extend HiVT's scene representation method by incorporating new features related to targets and lane segments, providing a more detailed description of relative spatial relationships; second, we propose a queries-informed decoder based on DETR [17], which combines historical query information and anchor point information end-to-end, promoting the multimodality of output trajectories; and, third, the designed QINET is capable of making accurate and reasonable predictions.

2. Related Work

Traffic Scene Representation. Solving the motion prediction problem requires learning rich representations from elements in the traffic scene, including high-definition maps and the agent's past trajectories. A significant amount of research employs grid-based scene representations as model inputs [18–20] and utilizes standard image models [21,22] for learning. Specifically, these methods extract map elements (such as lane boundaries, stop lines, and crosswalks) from high-definition maps and render the scene into a top–down image using different colors. The agent's past trajectories are either rasterized into additional image channels [1,5] or processed by temporal models like RNNs [23,24].

Rasterization methods require mature computer technology support, but their drawback lies in being inefficient and costly. Recently, vectorization methods [2,4,25] have gained popularity due to their efficient sparse encoding and ability to capture complex structural information. Unlike rasterization methods, these approaches consider the scene as a set of entities associated with semantic and geometric attributes, and learn the relationships between these entities. VectorNet [25] employs graph neural networks to model interactions between lanes and trajectory polylines. It has also served as a backbone in some subsequent works [26,27]. LaneGCN [2] constructs lane graphs from lane segments and employs multi-scale graph convolutional networks to capture long-range dependencies by learning features of graph nodes. TPCN [4] extends point cloud models to learn from a spatial-temporal set composed of trajectory waypoints and lane points. Our scene representation falls under this category as well, but what sets it apart is that all vectorized entities are characterized by relative positions, enhancing model robustness as relative displacements remain invariant to translation. The approach most related to this paper is HiVT [15], which uses a translation-invariant scene representation method that avoids using absolute positions and characterizes the geometric entities with relative positions and constructs rotation-invariant transformers to model the different interactions between the vectorized entities locally and globally.

Motion Prediction. Since social interactions are ubiquitous in traffic scenes and significantly influence the future motions of traffic agents, many motion prediction methods consider dependencies between agent behaviors and rational agent–agent interactions. They employ social pooling [28,29], graph neural networks [30,31], or attention mechanisms [20,25,32–34]. Inspired by the success of transformer models [10] in various fields, some recent works utilize transformers in motion prediction tasks to model spatial relationships, temporal dependencies, and relationships between agents and map elements [23,32,34,35].

In contrast, our transformer architecture differs from existing architectures by incorporating hierarchical learning of local and global representations. We encode each timestep in the local encoder, breaking down the time. In addition, we decompose the space by modeling multiple agents with a goal-centered representation that is invariant to translation and rotation in the scene. In the global encoder, we interact with all cars in the scene to obtain remote dependency information. The combination of a hierarchical structure and symmetric design allows our approach to achieve state-of-the-art prediction performance with fewer parameters and lower computational costs.

3. Method

3.1. Overall Framework

This section proposes a query-informed vehicle trajectory prediction method, QINET, to induce the model to generate multi-modal predicted trajectory to the maximum extent. The overall process is shown in Figure 1. The prediction model consists of three parts: scene vectorization representation, encoder, and decoder. Firstly, in the scene vectorization representation part, we extend the feature representation in HiVT [15]. For the vector node representation of agent, we use the displacement of agent and its absolute value, velocity, and its absolute value, sine–cosine value of heading angle, and timestamp length information. For lane vector node representation, we use lane segment displacement and heading angle to represent. Such scenarios represent a more detailed description of the relative spatial relationships between vector nodes. Then, our feature extraction encoder makes use of HiVT's encoder architecture to design subgraphs for local encoding. Transformer encoder is used to pay attention to time information, and global graph is used to extract and interact remote dependent information. In addition, we propose a query-informed multi-scene modality for an end-to-end learning approach to induce output multimodal prediction trajectories. In our method, the proposed generation is obtained by querying the output current encoding of transformer encoder by cross-attention method, which is the information extraction of agent historical trajectory. The multi-modality of

the predicted trajectory can be promoted to the maximum extent without environmental constraints. In addition, we obtain the anchor feature representation that absorbs and carries rich goal-oriented context information through self-attentional learning of global graph output. A portion of the input to the multi-modal trajectory prediction decoder comes from the query-informed multi-scene modality features, while another portion comes from anchor features.

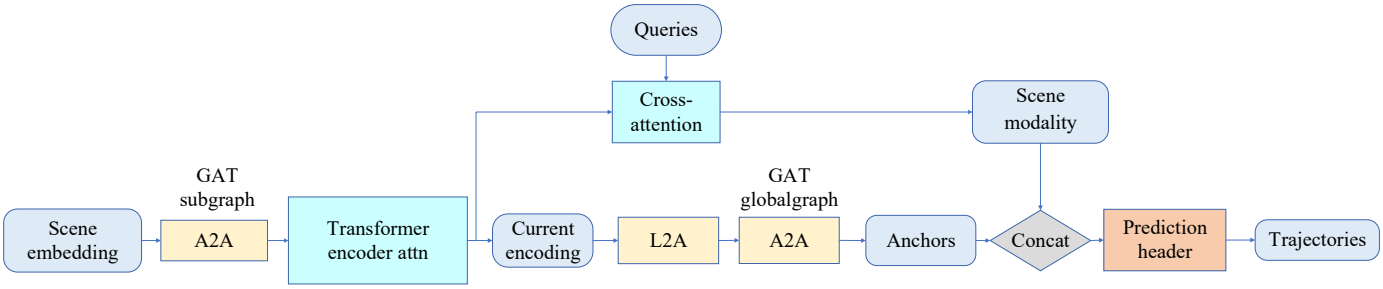

Figure 1. This is the overall framework of QINET. We utilize the graph attention mechanism (GAT) to establish A2A and L2A for extracting environmental features around participants. We set query matrices (queries) to query the historical trajectory features of the agent, obtaining diverse scene-modal features. These scene-modal features, combined with anchor features learned from the global graph output, are used to output multimodal trajectories.

3.2. Complexity Analysis

In the local encoder part, we utilized the model architecture of HiVT to decompose time and spatial dimensions, learning spatial relationships locally at each timestep. This approach reduces complexity from $O\left((NT+L)^2\right)$ to $O\left(NT^2+TN^2+NL\right)$, where N, T, and L represent the number of agents, historical timesteps, and the number of lane segments, respectively. Although we expanded the feature dimension of nodes in HiVT, this expansion has minimal impact on the overall complexity. In addition, in the decoder part, the complexity added by the method of designing query matrices to obtain multimodal scene feature representations can be calculated as $O(NT)$, which is lightweight compared to the local encoder.

3.3. Scene Representation

3.3.1. Node Feature Representation

For the traffic agent, we extract trajectory segments at each timestamp, which take the form of directed splines. These trajectory segments, referred to as vector nodes, are characterized by their feature attributes as: $\mathbf{n}_a = \left\{ \mathbf{R}_i^{T^\top} \mathbf{d}_i^t, \mathbf{R}_i^{T^\top} \mathbf{v}_i^t, \mathbf{R}_i^{T^\top} \mathbf{a}_i^t, \Delta t^t, \mathbf{b}_i \mid i = 1, \ldots, N_t, \right.$ $\left. t = 0, \ldots, 19 \right\}$. The main feature attributes are highlighted in red as follows:

$$\mathbf{l}_i^t = \left[x_i^t, y_i^t\right] \tag{1}$$

$$\mathbf{d}_i^t = \mathbf{l}_i^t - \mathbf{l}_i^{t-1} \tag{2}$$

$$d_{x_i}^t = x_i^t - x_i^{t-1} \tag{3}$$

$$d_{y_i}^t = y_i^t - y_i^{t-1} \tag{4}$$

$$\mathbf{v}_i^t = \frac{\mathbf{d}_i^t}{\Delta t^t} \tag{5}$$

$$\alpha_i^t = \arctan \frac{d_{y_i}^t}{d_{x_i}^t}. \tag{6}$$

$$\mathbf{a}_i^t = \left[\cos\left(\alpha_i^t\right), \sin\left(\alpha_i^t\right)\right] \tag{7}$$

$$\Delta t^t = t^t - t^{t-1} \tag{8}$$

where N_t represents the total number of agent vehicles appearing at timestamp t. $\mathbf{l}_i^t$ denotes the co-ordinates of the i-th agent in the scene at timestamp t. $\mathbf{d}_i^t$ is the displacement vector of agent vehicle i from timestamp $t-1$ to t. $\mathbf{v}_i^t$ represents the speed. α_i^t indicates the heading angle of the i-th agent at timestamp t. $\mathbf{a}_i^t$ is the heading vector composed of the cosine and sine of the agent's heading angle. Δt^t represents the duration of the timestamp. Including this in the node features is considered due to the non-uniform sampling frequency of the Argoverse dataset, which is not consistently 0.1 s. $\mathbf{R}_i^T$ is the rotation matrix defined by the heading angle of the i-th agent at the current timestep (t = 19). $\mathbf{b}_i$ represents the semantic feature.

For lane vector nodes, we opt to extract the co-ordinates of lane points along with their associated semantic attributes, such as dashed or solid lines, and turning directions. We vectorize lane segments into nodes similar to agent vectors, and represent them as $\mathbf{n}_l = \{\mathbf{d}_k, \mathbf{a}_k, \mathbf{b}_k \mid k = 1, \ldots, N_l\}$, where N_l denotes the total number of lane segments, $\mathbf{d}_k$ represents the displacement vector of the lane segment, $\mathbf{a}_k$ is composed of the sine and cosine values of the heading angle of the lane segment, indicating the direction of the displacement vector, and $\mathbf{b}_k$ denotes the semantic attribute. The specific expressions for $\mathbf{d}_k$ and $\mathbf{a}_k$ are as follows:

$$\mathbf{d}_k = \mathbf{l}_k^1 - \mathbf{l}_k^0 \tag{9}$$

$$\mathbf{a}_k = [\cos(\alpha_k), \sin(\alpha_k)] \tag{10}$$

where $\mathbf{l}_k^1$ and $\mathbf{l}_k^0$ represent the endpoint and starting point of the lane segment, respectively. In the vectorized node representation, we abstain from using any absolute positions and instead utilize relative positions. This ensures that the node feature attributes possess translational invariance.

3.3.2. Edge Feature Representation

Node features only represent the characteristics of agent vehicles and lane segments. The graph attention mechanism requires specifying attention targets in the scene, and encoding the features of edges between the target agent vehicle and the attention targets. Therefore, we introduce attributes for the edges between entities. For the edge attributes between agents, we describe them as follows: $\mathbf{e}_{aa} = \left\{ \mathbf{R}_i^T \mathbf{d}_{ij}^t, \mathbf{R}_i^T \mathbf{v}_{ij}^t, \mathbf{d}_{j2i}^t, \mathbf{v}_{j2i}^t, \mathbf{a}_{j2i}^t \mid t = 0, \ldots, 19; i, j = 1, i \neq j \right\}$, the details are as follows:

$$\mathbf{d}_{ij}^t = \mathbf{l}_j^t - \mathbf{l}_i^t \tag{11}$$

$$\mathbf{v}_{ij}^t = \mathbf{v}_j^t - \mathbf{v}_i^t \tag{12}$$

$$\alpha_{ij}^t = \alpha_j^t - \alpha_i^t \tag{13}$$

$$\mathbf{a}_{j2i}^t = \left[\cos\left(\alpha_{ji}^t\right), \sin\left(\alpha_{ji}^t\right)\right] \tag{14}$$

$$\mathbf{d}_{j2i}^t = \mathbf{R}_i^{t^\top} \mathbf{d}_{ji}^t = \left[\mathbf{d}_{j2ix}^t, \mathbf{d}_{j2iy}^t\right] \tag{15}$$

$$\mathbf{v}_{j2i}^t = \mathbf{R}_i^{t^\top} \mathbf{v}_{ji}^t = \left[\mathbf{v}_{j2ix}^t, \mathbf{v}_{j2iy}^t\right] \tag{16}$$

where $\mathbf{d}_{ij}^t$ is the relative displacement vector between agent i and agent j, $\mathbf{v}_{ij}^t$ is the velocity vector between them, $\mathbf{R}_i^t$ is the rotation matrix parameterized by the heading angle of center agent i at timestamp t, $\mathbf{a}_{j2i}^t$ represents the relative heading angle vector, $\mathbf{d}_{j2i}^t$ expresses the lateral and longitudinal distance of agent j relative to agent i at timestamp t, and $\mathbf{v}_{j2i}^t$ represents the lateral and longitudinal velocity of agent j relative to agent i at timestamp t, where x denotes lateral relative to the center node agent i, and y denotes longitudinal relative to the center node agent i.

For the edge attributes between agent nodes and lane nodes, we describe them as follows: $\mathbf{e}_{al} = \left\{ \mathbf{R}_i^T \mathbf{d}_{ik}^t, \mathbf{d}_{i2k}^t, \mathbf{v}_{i2k}^t, \mathbf{a}_{i2k}^t \mid t = 0, \ldots, 19; i = 1, \ldots, N_t; k == 1, \ldots, N_l \right\}$; the details are as follows:

$$\mathbf{d}_{ik}^t = \mathbf{l}_k^0 - \mathbf{l}_i^t \tag{17}$$

$$\mathbf{d}_{i2k}^t = \mathbf{R}_k^{t^\top} \mathbf{d}_{ik}^t = \left[d_{i2kx}^t, d_{i2ky}^t \right] \tag{18}$$

$$\mathbf{v}_{i2k}^t = \mathbf{R}_k^{t^\top} \mathbf{v}_i^t = \left[v_{i2kx}^t, v_{i2ky}^t \right] \tag{19}$$

$$\alpha_{ik}^t = \alpha_k^t - \alpha_i^t \tag{20}$$

$$\mathbf{a}_{i2k}^t = \left[\cos(\alpha_{ik}^t), \sin(\alpha_{ik}^t) \right] \tag{21}$$

where $\mathbf{d}_{ik}^t$ is the relative position vector between agent I and lane segment k at timestamp t, $\mathbf{R}_k^t$ is the rotation matrix parameterized by the heading angle α_k^t of lane node k, $\mathbf{d}_{i2k}^t$ represents the lateral and longitudinal distance from agent i to lane segment k, where x is lateral relative to lane segment k and y is longitudinal relative to lane segment k, $\mathbf{v}_{i2k}^t$ represents the relative velocity vector, v_{i2kx}^t and v_{i2ky}^t, respectively, denote the lateral and longitudinal velocity of agent i relative to lane segment k, and $\mathbf{a}_{i2k}^t$ is the relative heading angle vector.

The relative positions and velocities we propose describe the distance between two independent nodes, as well as the speed at which one node moves laterally and longitudinally towards another node. Compared to absolute representations, this relative representation provides a more detailed description of the interactions between entities, allowing downstream networks to better understand their behaviors. Additionally, this lateral and longitudinal relative representation naturally ensures translational and rotational invariance.

3.4. Encoder

3.4.1. Local Encoder

The local encoder processes the temporal scene graph in two stages, as shown in Figure 2. In the first stage, it models the agent–agent interactions for each timestep, which we refer to as A2A. For A2A, we perform local interactions centered around each agent within a limited range. After the interaction between entities, we use a time transformer encoder module to capture temporal dependencies across the traffic scene. In the second stage, we extract the features from the last timestep of the output of the transformer encoder. These features contain information about the central agent's vehicle at the current timestep, as well as interaction information with nearby other agent vehicles in both spatial and temporal dimensions. We use these features for modeling lane–agent interactions, which we refer to as L2A. With this, after the local encoder is completed, we obtain agent features that are enriched with rich contextual information.

Agent–Agent Interaction. During the A2A step, we utilize a graph neural network. We employ multi-head cross-attention to understand the influence of different surrounding agent vehicles within each local range on the central agent vehicle. Specifically, we first apply multi-layer perceptions (MLPs) to the node attributes of the central agent vehicle and the corresponding edge attributes. This allows us to obtain a time-variant encoding $\boldsymbol{Z}_i = \left\{ \boldsymbol{z}_i^t \middle| t = 1, \ldots, T \right\}$ for the central agent node i, along with time-variant encodings for the surrounding neighboring nodes associated with it:

$$\mathbf{z}_i^t = \phi_{\text{center}} \left(\mathbf{R}_i^{T^\top} \mathbf{d}_i^t, \mathbf{R}_i^{T^\top} \mathbf{v}_i^t, \mathbf{R}_i^{T^\top} \mathrm{a}_i^\top, \Delta t^t, \mathbf{b}_i \right] \Big) \tag{22}$$

$$\mathbf{z}_{ij}^t = \phi_{\text{nbr}} \Big(\left[\mathbf{R}_i^{T^\top} \mathbf{d}_j^t, \mathbf{R}_i^{T^\top} \mathbf{d}_{ij}^t, \mathbf{R}_i^{T^\top} \mathbf{v}_{ij}^t, \mathbf{d}_{j2i}^t, \mathbf{v}_{j2i}^t, \mathbf{a}_{j2i}^t, \mathbf{b}_j \right] \Big) \tag{23}$$

where ϕ_{center} and ϕ_{nbr} represent MLP modules. Due to the use of relative vectors and the presence of rotation matrices, both the node attributes of the central node and its associated

edge attributes possess translational and rotational invariance. Next, we use cross-attention to fuse the central node features and its edge features. The query part of the cross-attention is derived from the central node attribute $\mathbf{z}_i^t$, while the key and value parts come from the edge attributes $\mathbf{z}_{ij}^t$. Subsequently, we perform dot product [10] and gating operations [15], resulting in the output $\hat{\mathbf{Z}}_\mathbf{i} = \left\{\hat{\mathbf{z}}_i^t | t = 1, \ldots, T\right\}$. We then further apply an MLP to $\hat{\mathbf{Z}}_\mathbf{i}$ and use residual connections to obtain the merged feature encoding $\mathbf{S}_\mathbf{i} = \left\{\mathbf{s}_i^t | t = 1, \ldots, T\right\}$, which contains information about agent interactions and updates after the interaction.

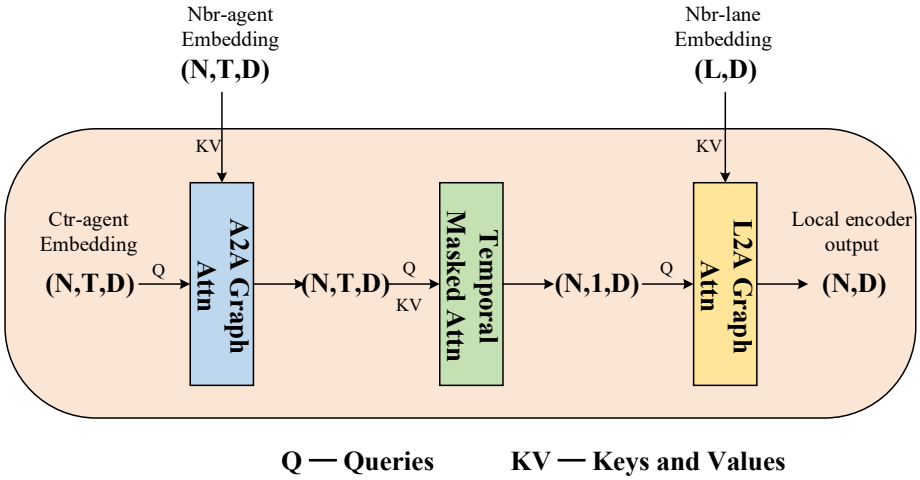

Figure 2. Overview of the local encoder diagram: the query always originates from the central agent node feature. For A2A, we extract key and value from neighboring agent nodes. Self-attention is employed in the temporal encoder. For L2A, key and value are sourced from neighboring lane nodes. In the diagram, N represents the number of agents in the scene, T denotes the number of timesteps, and L represents the number of lane nodes.

Temporal Dependency. To further capture temporal dependencies, we apply a transformer encoder to the output $\mathbf{S}_\mathbf{i}$ of the A2A step. Following the approach of BERT [36], we introduce learnable position embeddings at each timestamp and stack them onto $\mathbf{S}_\mathbf{i}$ to obtain the new matrix $\hat{\mathbf{S}}_\mathbf{i} \in \mathbb{R}^{T \times d_h}$. Unlike previous studies [15], we do not add an extra learnable token at the end position, resulting in $\hat{\mathbf{S}}_\mathbf{i} \in \mathbb{R}^{(T+1) \times d_h}$. Instead, we directly process $\hat{\mathbf{S}}_\mathbf{i}$ through the transformer encoder to obtain the updated sequence features $\mathbf{H}_i = \left\{\mathbf{h}_i^t \mid t = 1, \ldots, T\right\}$ and extract the final node feature $\mathbf{h}_i^T$ belonging to the current timestep. This feature is then fed into the subsequent L2A module, as we have observed improved performance with this approach. During the transformer encoding process, a time mask is applied to enforce tokens to only attend to preceding timesteps.

Agent–Lane Interaction. To facilitate information interaction between agents and lane segments, we apply another multi-head cross-attention module. First, we use a multi-layer perceptron to encode the edge features between the central agent node i and nearby lane nodes:

$$\mathbf{z}_{ik} = \phi_{\text{lane}}\left(\left[\mathbf{R}_i^{T^\top}\mathbf{d}_k, \mathbf{R}_i^{T^\top}\mathbf{d}_{ik}, \mathbf{d}_{i2k}, \mathbf{v}_{i2k}, \mathbf{a}_{i2k}, \mathbf{b}_k\right]\right) \tag{24}$$

where ϕ_{lane} represents an MLP module. We use the current timestep's agent node feature $\mathbf{h}_i^T$ from the transformer encoder output as the query, and the edge attributes $\mathbf{z}_{ik}$ between the agent and the lane segment as the key and value. The field of view is an adjustable threshold used to limit the lane nodes that need to be interactively fused with the central agent node. We obtain the final node embedding $\mathbf{h}_i$ for central agent i. It encapsulates a rich spatiotemporal representation fused by agent i, combining the dynamic characteristics of agent i with its iterative interactions with the surrounding environment. The final local representation for all agents is defined as $\mathbf{H} = \{\mathbf{h}_i \mid i = 1, \ldots, N\}$, where N is the number of agents.

3.4.2. Global Encoder

The local encoder only achieves information interaction within a local scope, lacking remote dependency relationships within the scene. Therefore, we designed a global encoder. Similar to the A2A module in the local encoder, we employ an MLP to encode the edge attributes between agent i and agent j.

$$\mathbf{g}_{ij} = \phi_{\text{rel}}\left(\left[\mathbf{R}_i^{T^\top}\mathbf{d}_{ij}^T, \mathbf{R}_i^{T^\top}\mathbf{v}_{ij}^T, \mathbf{d}_{j2i}^T, \mathbf{v}_{j2i}^T, \mathbf{a}_{j2i}^T\right]\right) \tag{25}$$

Here, T represents the current timestep. Afterwards, we use $\mathbf{h}_i$ as the query, $\left[\mathbf{h}_j, \mathbf{g}_{ij}\right]$ as the key and value:

$$\mathbf{q}_i = \mathbf{W}^{Q^{\text{global}}}\mathbf{h}_i, \tag{26}$$

$$\mathbf{k}_{ij} = \mathbf{W}^{K^{\text{global}}}\left[\mathbf{h}_j, \mathbf{g}_{ij}\right], \tag{27}$$

$$\mathbf{v}_{ij} = \mathbf{W}^{V^{\text{global}}}\left[\mathbf{h}_j, \mathbf{g}_{ij}\right], \tag{28}$$

where $\mathbf{W}^{Q^{\text{global}}}$, $\mathbf{W}^{K^{\text{global}}}$, and $\mathbf{W}^{V^{\text{global}}}$ are linear transformation matrices. Then, we apply a multi-head cross-attention module to update the features of agent i:

$$\boldsymbol{\alpha}_i = \text{softmax}\left(\frac{\mathbf{q}_i^T}{\sqrt{d_k}} \cdot \left[\left\{\mathbf{k}_{ij}\right\}_{j\in\mathcal{N}_i}\right]\right) \tag{29}$$

$$\hat{\mathbf{h}}_i = \sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{v}_{ij} \tag{30}$$

where $\mathcal{N}_i$ contains the neighboring agents that central agent i needs to interact with, $\boldsymbol{\alpha}_{ij}$ represents the score weight of neighbor agent j relative to agent i. Furthermore, we pass the updated neighbor agent feature $\hat{\mathbf{h}}_i$ and the central agent feature $\mathbf{h}_i$ through a gating step [15] before inputting them into the MLP module to obtain the output of the global graph $\widetilde{\mathbf{h}}_i$, with feature dimensions denoted as [K, N, and D]. Here, K is the number of heads in the multi-head cross-attention module, representing the number of modes in the output trajectory, N denotes the number of agents, and D is the feature dimension.

3.5. Decoder

3.5.1. Query-Informed Multi-Scene Modality Creation

As shown in Figure 3, we relinquish the constraints of specific driving scenarios and aim to maximize the diversity of future trajectory candidates by first creating multi-modal scenes through querying from the motion history of the target agent. Specifically, inspired by the approach of setting object queries in DETR [17], we define a set of learnable parameters forming a query matrix $Q_{\text{scene}} \in \mathbb{R}^{K\times \text{D}}$ to attend to the output $\mathbf{H_i} = \left\{h_i^t \middle| t = 1, \ldots T\right\} \in \mathbb{R}^{T\times \text{D}}$ from the temporal dependency module. This is achieved by generating K scene modal features $E_{\text{scene}} = \left\{E_{\text{scene}}^k \middle| k = 1, \ldots, K\right\} \in \mathbb{R}^{K\times D_E}$ through a cross-attention mechanism:

$$E_{scene} = Softmax\left(\frac{\left(Q_{scene}\mathbf{W}_q\right)\left(\mathbf{H}_i\mathbf{W}_k\right)^T}{\sqrt{D_E}}\right)\left(\mathbf{H}_i\mathbf{W}_v\right), \tag{31}$$

$$score_k = softmax(\alpha_k) \tag{32}$$

$$E_{scene}^k = \sum_{t=1,\ldots,T} score_{kt}\left(\mathbf{W}_q^{\text{T}}h_i^t\right) \tag{33}$$

where $\mathbf{W}_q$, $\mathbf{W}_k$, and $\mathbf{W}_v \in R^{D\times D_E}$ are linear transformation matrices. α_{kt} represents the contribution of the agent's feature at the t-th historical timestamp to the k-th scene mode. We apply a softmax operation to the contributions of all timestamps corresponding to the

k-th scene mode to obtain the vector $score_k$. The contribution scores corresponding to each timestamp in $score_k$ are multiplied with the feature of the agent at that timestamp and then summed, resulting in the queried k-th scene modal feature. Since the scene features are derived from the agent's historical trajectory encoding and do not contain semantic features of the surrounding environment, this ensures maximum multimodality of the scene. D_E represents the dimension after linear transformation. The denominator $\sqrt{D_E}$ is used for normalization and to prevent the dot product from becoming too large, which might lead to saturation in the softmax operation.

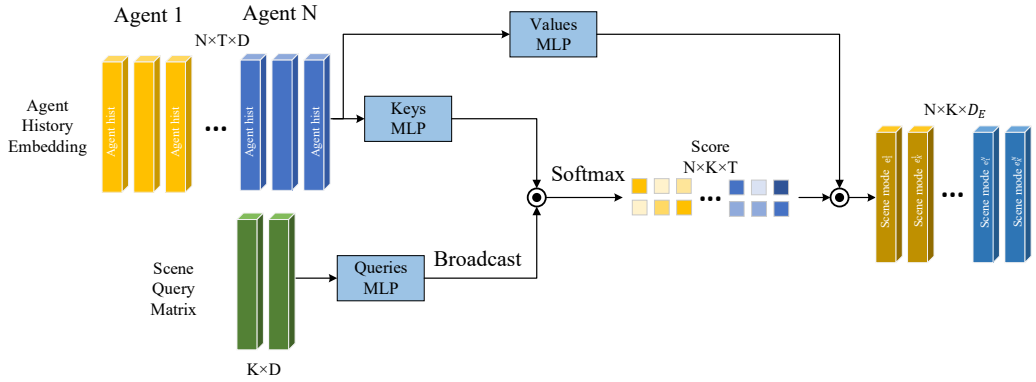

Figure 3. Construction diagram of multi-scene modality. We establish a learnable scene query matrix to query the historical trajectory features of agents, obtaining modality features for multiple scenes for decoding future trajectories.

3.5.2. Anchor Learning

The anchors are learned end-to-end in the network to convey target-oriented environmental information while preserving diversity. We set the number of anchors to K, which is equal to the number of environmental modes, so that each anchor corresponds to one scene mode. We apply an MLP to the output of the global graph $\widetilde{\mathbf{h}}_i$ to generate anchor features corresponding to each scene mode:

$$E_{\text{anchor}} = \phi_{\text{anchor}}\left(\widetilde{\mathbf{h}}_i\right) \tag{34}$$

where ϕ_{scene} represents an MLP module. Our model does not directly utilize predicted endpoints, but rather leverages their embeddings (i.e., pre-output features) as anchor points $E_{\text{anch}} = \left\{E^i_{\text{anch}} \middle| i = 1, \ldots, K\right\}$. These anchor points are used to inject target-oriented scene context into the generated multimodal scene features E_{scene} . This is a new type of approach compared to previous research. In TNT [16], anchor points are manually sampled uniformly from the map. In MultiPath [1], anchor points are predefined trajectories clustered from training data. In MultiPath++ [37], anchor points are learnable model parameters that are fixed after training and independent of the input. In contrast, we propose using anchor embeddings to facilitate trajectory learning. Compared to TNT and MultiPath, our anchors are more adaptive and convenient to obtain through end-to-end learning. Compared to MultiPath++, our anchors correspond to individual samples, thus carrying specific sample-specific information.

3.5.3. Trajectory Prediction Head

As described above, the multimodal scene encoding E_{scene} can be seen as unconstrained future trajectories inferred solely from the agent's history, while anchors E_{anchor} convey target-based contextual information. Here, we combine both to allow the network to make further selections and refinements:

$$E_{\text{final}} = W_1 E_{\text{scene}} + W_2 E_{\text{anchor}} \tag{35}$$

where W_1 and W_2 are linear transformation matrices. Taking the fused features as input, the multimodal prediction head outputs the final motion predictions. For each participant, it predicts K possible future trajectories along with their confidences. The head has two branches: one regression branch predicting the trajectories for each mode, and one classification branch predicting the confidence scores for each mode. For the i-th participant, we apply residual blocks and linear layers to regress the K sequences of relative co-ordinates in the regression branch:

$$O_{i,\text{ reg}} = \left\{ \left(\mathbf{p}_{i,1}^k, \mathbf{p}_{i,2}^k, \ldots, \mathbf{p}_{i,T}^k \right) \right\}_{k \in [0,K-1]} \tag{36}$$

where, $\mathbf{p}_{i,t}^k$ represents the predicted relative co-ordinates of the i-th participant in the k-th mode at timestep t, i.e., co-ordinates in the local co-ordinate system with the historical endpoint of the i-th participant as the origin. For the classification branch, we apply an MLP to $\mathbf{p}_{i,T}^k - \mathbf{p}_{i,0}$ to obtain K distance embeddings, where $\mathbf{p}_{i,0}$ is the last point of the historical trajectory of the i-th agent. Then, we concatenate each distance embedding with the agent features, apply residual blocks and linear layers to output K confidence scores, $O_{i,\text{cls}} = (c_{i,0}, c_{i,1}, \ldots, c_{i,K-1})$.

4. Simulation Results

4.1. Experimental Settings

4.1.1. Dataset

We utilize the Argoverse motion forecasting dataset [38], which comprises real-world traffic scenarios with agent trajectories and high-definition maps. The dataset encompasses 324,557 authentic traffic scenes. The training, validation, and test sets include 205,942, 39,472, and 78,143 scenes, respectively. Each scene is a 5 s sequence sampled at 10 Hz, containing the positions of all agents in the past 2 s. In the Argoverse motion forecasting challenge, the task is to predict the future positions of a target agent for the next 3 s based on an initial observation of the first 2 s of the scene.

4.1.2. Metrics

We adhere to the Argoverse benchmark and evaluate our model using metrics including minimum average displacement error (minADE), minimum final displacement error (minFDE), and miss rate (MR). These metrics allow the model to predict up to 6 trajectories for each agent.

4.1.3. Implementation Details

We trained all models using the AdamW optimizer [39] with an initial learning rate of 0.001 for 64 epochs. We employed a cosine annealing scheduler for learning rate decay. The number of layers for the agent–agent transformer, agent–lane transformer, temporal transformer, and global encoder was set to 1, 1, 4, and 3, respectively. The number of hidden units was 128, and there were 8 heads in all multi-head attention blocks. The local region radius for A2A was set to 20 m, and for L2A it was set to 50 m. We did not predict agents that appeared for less than two steps, unless it was the target agent.

4.1.4. Comparison with State-of-the-Art

In Table 1, we present the results of QINET on the Argoverse motion prediction test set, comparing it with other state-of-the-art models. The data in Table 1 are sourced from the Argoverse leaderboard. QINET outperforms all other methods in terms of minADE and minFDE, and maintains a competitive ranking in MR, verifying the superior predictive performance of our method. The sacrifice in the MR metric stems from the decoder's multimodal influence on the generated trajectories, but this influence improves the accuracy of trajectory prediction in some scenarios.

Table 1. Results on Argoverse motion forecasting leaderboard.

Models	minFDE	minADE	MR
THOMAS [40]	1.4388	0.9423	0.1038
GOHOME [41]	1.4503	0.9425	0.1048
DenseTNT [26]	1.2815	0.8817	0.1258
mmTransformer [27]	1.3383	0.8436	0.1540
TNT [16]	1.4457	0.9097	0.1656
LaneRCNN [42]	1.4526	0.9038	0.1232
QINET	1.2643	0.8140	0.1436

In the validation set section, we compared our results with HiVT. We found that before model ensemble, our model performs better on the Argoverse validation set compared to HIVT, as shown in Table 2 with specific metrics.

Table 2. Comparison of QINET with HIVT on the Argoverse validation set without model ensembling.

Models (No Model Ensemble)	minFDE	minADE	MR
HiVT-128	0.6612	0.9691	0.0921
QINET	0.6514	0.9481	0.0892

4.1.5. Ablation Studies

Our ablation study consists of four parts: the importance of each module in QINET, the importance of expanded scene representation, and the importance of layered lane transformers. We conducted these experiments on the Argoverse validation set.

Importance of Each Module.

To investigate the importance of each module for the overall network, we individually removed each module and tested its contribution on the Argoverse test set, as shown in Table 3. Each module contributes to the improvement of network performance.

Table 3. Importance of each component of our framework.

Extended A2A	Temporal Masked Attn	Scene Modality Creation	minADE	minFDE	MR
	✓	✓	0.6804	0.9843	0.0936
✓		✓	0.9628	1.4686	1.1977
✓	✓		0.6902	1.0376	0.1101
✓	✓	✓	0.6514	0.9481	0.0891

Firstly, without A2A, the model lacks local interactions within the prediction scenes, resulting in a decrease in model metrics.

Secondly, the absence of the temporal dependency module prevents the network from addressing temporal dependencies. Since inferring future trajectories of agents in highly dynamic environments heavily relies on historical information, the lack of the transformer encoder module significantly impairs the model's performance metrics.

Thirdly, lane information plays a crucial role in motion prediction, as road environment information constrains the trajectories of vehicles to some extent. Under such constraints, vehicles generally move along the lanes. Moreover, global graph A2A also contributes to the model's effectiveness, as global interactions can capture long-range dependency relationships, enhancing the accuracy of predictions.

4.1.6. Qualitative Results of QINET

In the visualizations in Figure 4, we selected representative scenes to demonstrate the qualitative results of the QINET network. The visualizations confirm that QINET is capable of performing multimodal predictions for all agents, and the predicted trajectories are

reasonable and close to the ground truth. For clarity, we display only the agent's historical trajectory in yellow, the ground truth future trajectory in red, and the predicted trajectory in green. It can be observed that due to the presence of local A2A, temporal, L2A, and global A2A modules, our network effectively extracts agent features and predicts their future trajectories. Additionally, the query-informed multimodal scene encoding effectively promotes the multimodality of agent vehicle future trajectories.

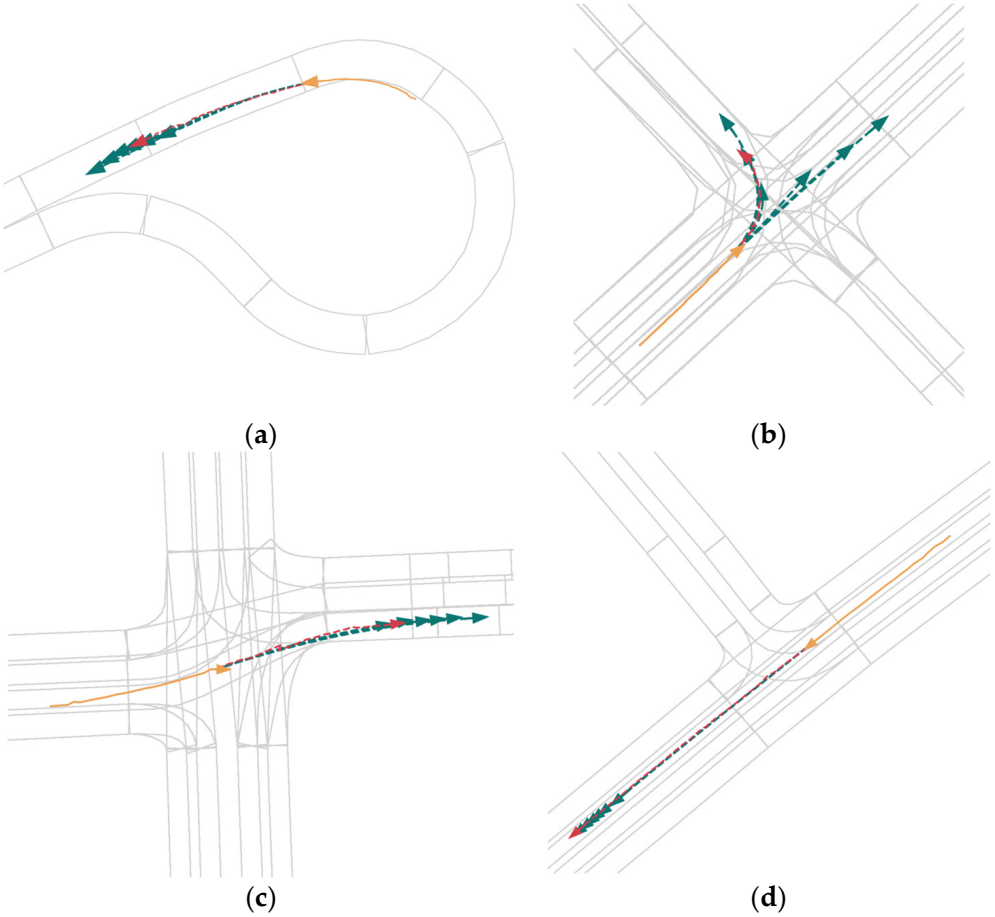

Figure 4. Qualitative Results of QINET. We selected several classical scenario prediction results as shown in (**a**–**d**). For clarity, we visualize individual agents separately. We use orange to depict past trajectories, red for actual trajectories, and green for predicted trajectories.

4.1.7. Comparison with HiVT in Bad Case

In this part, we compared the output results of QINET with those of HiVT, as shown in the following Figure 5. It can be seen that some bad cases predicted in the HiVT model show better prediction results in the QINET model. In addition, in some intersection scenarios, the QINET network is capable of demonstrating awareness of turning. This is attributed to our decoder design that enhances the multimodality of the trajectory predictions.

4.1.8. Failed Cases

In this section, we present some scenarios where QINET predictions failed, as shown in Figure 6. Compared to HiVT, QINET's prediction results have improved, with trajectories becoming more multimodal. This improvement is due to the presence of multimodal scene query features in the decoder.

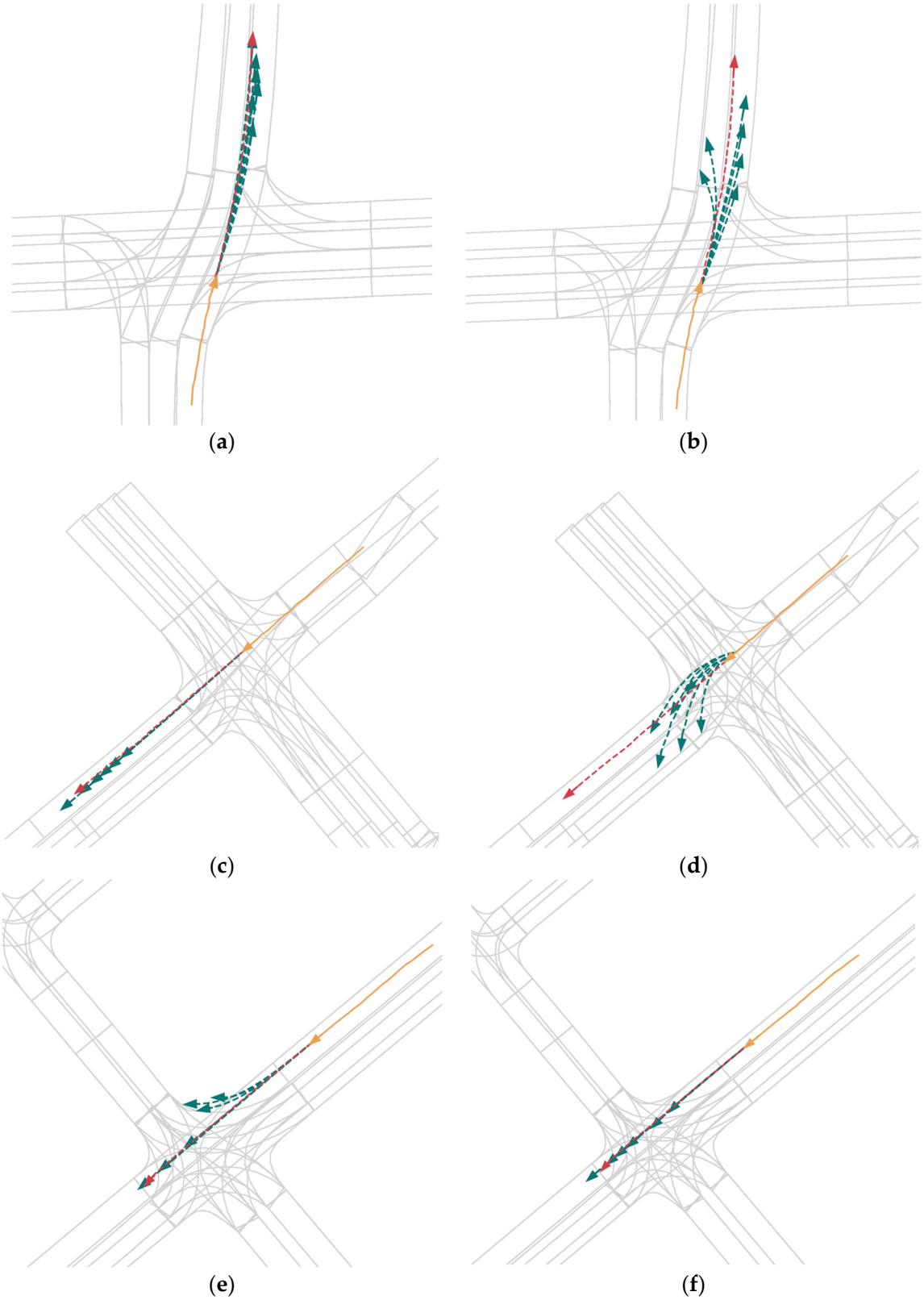

Figure 5. Visualize the comparative results. (**a**,**c**,**e**) come from QINET, while (**b**,**d**,**f**) come from HiVT.

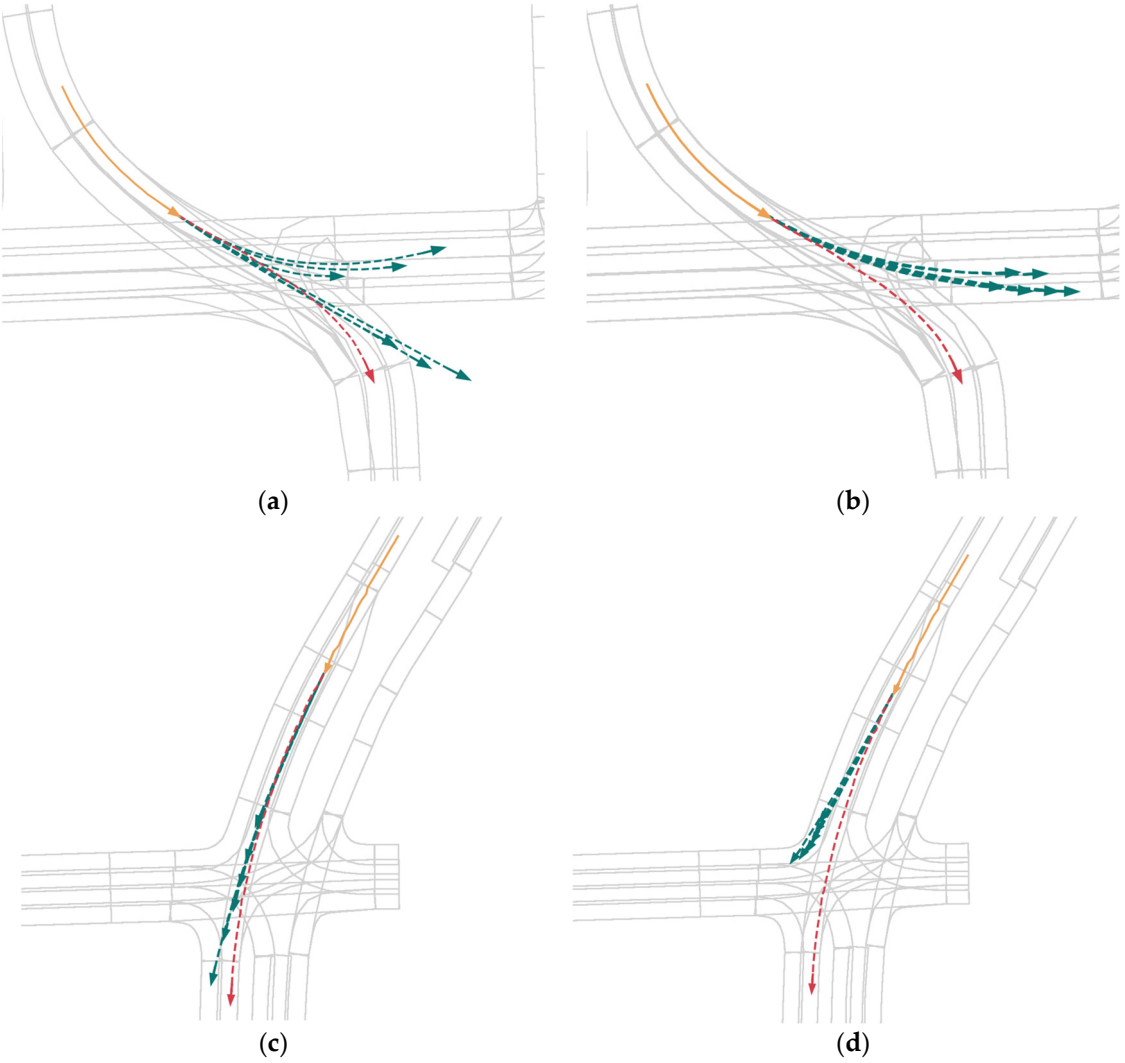

Figure 6. Visualize the comparative results. (**a**,**c**) come from QINET, while (**b**,**d**) come from HiVT.

5. Conclusions

This paper presents a new multi-agent prediction framework that enhances trajectory prediction accuracy by constructing extended node features and edge features. It utilizes the query mechanism in cross-attention to obtain multi-scenario modal encodings, thereby maximally promoting the multimodality of generated trajectories. Experiments demonstrate that our method achieves good results in both prediction accuracy and the multimodality of generated trajectories on the Argoverse motion prediction benchmark. Future research will focus on how to conduct more efficient lane-to-agent (L2A) processing to improve the model's inference speed. This is because considering all lane nodes within a certain radius can sometimes lead to resource wastage in some scenarios. For example, in scenarios where a vehicle is predicted to drive in the far-left lane, L2A would include nodes from the opposite lane, which may not be meaningful.

Author Contributions: Conceptualization, C.G.; Methodology, C.G. and S.F.; Software, C.G. and S.F.; Validation, C.G. and S.F.; Formal analysis, C.G. and S.F.; Resources, C.G. and W.Z.; Writing—original draft, C.G. and S.F.; Writing—review & editing, C.C., W.Z. and J.W.; Visualization, C.G., S.F. and W.Z.; Supervision, W.Z., Y.Z. and Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest. The funding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Chai, Y.; Sapp, B.; Bansal, M. MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction. In Proceedings of the Robot Learning (CoRL), New Orleans, LA, USA, 6–9 May 2019.
2. Liang, M.; Yang, B.; Hu, R. Learning lane graph representations for motion forecasting. In Proceedings of the European Conference on Computer Vision (ECCV), Tel Aviv, Israel, 15–16 August 2020.
3. Mercat, J.; Gilles, T.; El Zoghby, N. Multi-head attention for multi-modal joint vehicle motion forecasting. In Proceedings of the International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–4 June 2020.
4. Ye, M.; Cao, T.; Chen, Q. Tpcn: Temporal point cloud networks for motion forecasting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 19–25 June 2021.
5. Cui, H.; Radosavljevic, V.; Chou, F.-C. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In Proceedings of the International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019.
6. Hong, J.; Sapp, B.; Philbin, J. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019.
7. Bruna, J.; Zaremba, W.; Szlam, A. Spectral networks and locally connected networks on graphs. In Proceedings of the International Conference on Learning Representations (ICLR), Banff, AB, Canada, 14–16 April 2014.
8. Gilmer, J.; Schoenholz, S.S.; Riley, P.F. Neural message passing for quantum chemistry. In Proceedings of the International Conference on Machine Learning (ICML), Sydney, Australia, 6–11 August 2017.
9. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
10. Vaswani, A.; Shazeer, N.; Parmar, N. Attention is all you need. In Proceedings of the Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017.
11. Qi, C.R.; Su, H.; Mo, K. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
12. Hashemi, S.M.; Botez, R.M.; Grigorie, T.L. New Reliability Studies of Data-Driven Aircraft Trajectory Prediction. *Aerospace* **2020**, *7*, 145. [CrossRef]
13. Hashemi, S.M.; Hashemi, S.A.; Botez, R.M.; Ghazi, G. A Novel Fault-Tolerant Air Traffic Management Methodology Using Autoencoder and P2P Blockchain Consensus Protocol. *Aerospace* **2023**, *10*, 357. [CrossRef]
14. Hashemi, S.M.; Hashemi, S.A.; Botez, R.M.; Ghazi, G. Aircraft Trajectory Prediction Enhanced through Resilient Generative Adversarial Networks Secured by Blockchain: Application to UAS-S4 Ehécatl. *Appl. Sci.* **2023**, *13*, 9503. [CrossRef]
15. Zhou, Z.; Ye, L.; Wang, J. Hivt: Hierarchical vector transformer for multi-agent motion prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 19–24 June 2022.
16. Zhao, H.; Gao, J.; Lan, T. TNT: Target-driven trajectory prediction. In Proceedings of the Conference on Robot Learning (CoRL), Auckland, New Zealand, 14–18 December 2020.
17. Carion, N.; Massa, F.; Synnaeve, G. End-to-end object detection with transformers. In Proceedings of the European conference on computer vision (ECCV), Tel Aviv, Israel, 15–16 August 2020.
18. Djuric, N.; Radosavljevic, V.; Cui, H. Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Snowmass, CO, USA, 2–5 May 2020.
19. Gilles, T.; Sabatini, S.; Tsishkou, D. Home: Heatmap output for future motion estimation. In Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC), Indianapolis, IN, USA, 19–22 September 2021.
20. Salzmann, T.; Ivanovic, B.; Chakravarty, P. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, SC, USA, 23–28 August 2020.
21. Huang, G.; Liu, Z.; Van Der Maaten, L. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
22. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning (ICML), Long Beach, CA, USA, 10–15 June 2019.
23. Rhinehart, N.; Kitani, K.M.; Vernaza, P. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
24. Rhinehart, N.; McAllister, R.; Kitani, K. Precog: Prediction conditioned on goals in visual multi-agent settings. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019.

25. Gao, J.; Sun, C.; Zhao, H. VectorNet: Encoding HD Maps and Agent Dynamics From Vectorized Representation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–20 June 2020.
26. Gu, J.; Sun, C.; Zhao, H. Densetnt: End-to-end trajectory prediction from dense goal sets. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 11–17 October 2021.
27. Liu, Y.; Zhang, J.; Fang, L. Multimodal motion prediction with stacked transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 19–25 June 2021.
28. Alahi, A.; Goel, K.; Ramanathan, V. Social lstm: Human trajectory prediction in crowded spaces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, CA, USA, 26 June–1 July 2016.
29. Gupta, A.; Johnson, J.; Fei-Fei, L. Social gan: Socially acceptable trajectories with generative adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018.
30. Casas, S.; Gulino, C.; Liao, R. Spagnn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–4 June 2020.
31. Huang, Y.; Bi, H.; Li, Z. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019.
32. Ngiam, J.; Caine, B.; Vasudevan, V. Scene transformer: A unified architecture for predicting multiple agent trajectories. In Proceedings of the International Conference on Learning Representations (ICLR), Kigali, Rwanda, 25–29 April 2022.
33. Ye, L.; Wang, Z. GSAN: Graph Self-Attention Network for Learning Spatial–Temporal Interaction Representation in Autonomous Driving. *IEEE Internet Things J.* **2022**, *9*, 9190–9204. [CrossRef]
34. Yu, C.; Ma, X.; Ren, J. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, SC, USA, 23–28 August 2020.
35. Yuan, Y.; Weng, X.; Ou, Y. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 11–17 October 2021.
36. Devlin, J.; Chang, M.-W.; Lee, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the Conference of the North-American-Chapter of the Association-for-Computational-Linguistics—Human Language Technologies (NAACL-HLT), Minneapolis, MN, USA, 2–7 June 2019.
37. Varadarajan, B.; Hefny, A.; Srivastava, A. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In Proceedings of the International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022.
38. Chang, M.-F.; Lambert, J.; Sangkloy, P. Argoverse: 3d tracking and forecasting with rich maps. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019.
39. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv* **2017**, arXiv:1711.05101.
40. Gilles, T.; Sabatini, S. Thomas: Trajectory heatmap output with learned multi-agent sampling. *arXiv* **2021**, arXiv:2110.06607v3.
41. Gilles, T.; Sabatini, S.; Tsishkou, D. Gohome: Graph-oriented heatmap output for future motion estimation. In Proceedings of the International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022.
42. Zeng, W.; Liang, M.; Liao, R. Lanercnn: Distributed representations for graph-centric motion forecasting. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021.