HAL
open science

TSGN: Temporal Scene Graph Neural Networks with Projected Vectorized Representation for Multi-Agent Motion Prediction

Yunong Wu, Thomas Gilles, Bogdan Stanciulescu, Fabien Moutarde

To cite this version:

Yunong Wu, Thomas Gilles, Bogdan Stanciulescu, Fabien Moutarde. TSGN: Temporal Scene Graph Neural Networks with Projected Vectorized Representation for Multi-Agent Motion Prediction. 2023 IEEE Intelligent Vehicles Symposium (IV), Jun 2023, Anchorage, France. pp.1-8, 10.1109/IV55152.2023.10186764 . hal-04375358

HAL Id: hal-04375358

https://minesparis-psl.hal.science/hal-04375358v1

Submitted on 5 Jan 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

TSGN: Temporal Scene Graph Neural Networks with Projected Vectorized Representation for Multi-Agent Motion Prediction

Yunong Wu[1], Thomas Gilles[2], Bogdan Stanciulescu[2], Fabien Moutarde[2]
[1] School of Computation, Information and Technology, Technical University of Munich
[2] MINES ParisTech, PSL University, Center for robotics
{wuyun}@in.tum.de , {thomas.gilles, bogdan.stanciulescu, Fabien.Moutarde}@minesparis.psl.eu

arXiv:2305.08190v1 [cs.CV] 14 May 2023

***Abstract*—Predicting future motions of nearby agents is essential for an autonomous vehicle to take safe and effective actions. In this paper, we propose TSGN, a framework using Temporal Scene Graph Neural Networks with projected vectorized representations for multi-agent trajectory prediction. Projected vectorized representation models the traffic scene as a graph which is constructed by a set of vectors. These vectors represent agents, road network, and their spatial relative relationships. All relative features under this representation are both translation- and rotation-invariant. Based on this representation, TSGN captures the spatial-temporal features across agents, road network, interactions among them, and temporal dependencies of temporal traffic scenes. TSGN can predict multimodal future trajectories for all agents simultaneously, plausibly, and accurately. Meanwhile, we propose a Hierarchical Lane Transformer for capturing interactions between agents and road network, which filters the surrounding road network and only keeps the most probable lane segments which could have an impact on the future behavior of the target agent. Without sacrificing the prediction performance, this greatly reduces the computational burden. Experiments show TSGN achieves state-of-the-art performance on the Argoverse motion forecasting benchmark.**

I. INTRODUCTION

Predicting the motion of multiple agents is an essential step in autonomous driving. It helps autonomous driving vehicles to plan their actions and prevent accidents. However, the traffic scene is highly complex. It contains agents, road network, and interactions among them. The prediction model needs to take these entities as inputs and outputs reasonable multimodal trajectories that the target agent could take in the future.

The traffic scene is viewed differently from the perspective of different agents. However, most exsiting methods select one target agent each time and take it as the central reference to perform the coordinate transformation on the entire scene, which is not symmetric for the other agents and results in only one prediction at a time. It is inefficient and not robust to the coordinate transformation. To alleviate these problems, HiVT [1] takes a symmetric way to encode the traffic scene. It divides the traffic scene into a set of local regions, each corresponding to one agent and centered on it, and uses a local-global mechanism to encode agents' features.

We adopt the symmetrical representation of HiVT and its local-global mechanisms. Based on them, we propose an extended scene representation method: Projected vectorized representation and an optimized prediction framework: TSGN. Our representation method introduces more features about agents and lane segments. It projects all relative spatial features between entities into the direction of the target entity for every timestamp, such that, all these relative features under this representation are also both translation- and rotation-invariant. TSGN consists of diverse interaction modules and temporal dependencies modules. These modules are constructed based on the transformer structure. TSGN treats the traffic scene in its entirety and encodes the agents' dynamics, surrounding lane segments, and interactions among them across time steps. In addition, we present a Hierarchical Lane Transformer module inside TSGN to capture the influence of the road network on the future motion of target agents. Unlike most approaches that use the full structural information of the surrounding road network, the Hierarchical Lane Transformer only selects the most probable lane segments which could have an impact on the future behavior of the target agent. It greatly reduces the computation burden and enables the model to do faster predictions without sacrificing the prediction performance.

Our contributions are summarized as follow:

- We extend the scene representation method of HiVT by adding new features about agents, lane segments, and their spatial relative relationships. We adopt a projected vectorized representation for all relative features of the traffic scene for every timestamp, which describes the relative spatial relationships in a more detailed way.
- We propose a Hierarchical Lane Transformer which only selects lane segments that affect the future behaviors of target agents the most for modeling the interactions, enabling much faster and equally accurate predictions.
- Our designed TSGN can make plausible and accurate predictions. We validate the performance of TSGN on Argoverse forecasting benchmark. It ranked 1st in terms of minADE on the leaderboard on September 13, 2022.

II. RELATED WORK

Recently deep learning-based models have brought great progress to the motion prediction. According to the different representation methods, these learning-based models can be divided into rasterized-representation-based models and vectorized-representation-based models.

Rasterized representation of traffic scenes has benefits of unitedly rendering various types of input information (e.g., HD map, agents' trajectories, spatial relationships) into an image with multiple channels. In [2], [3], different types of entities in traffic scenes are assigned different colors. HOME [4] rasterizes elements of HD map in different semantic channels and adds historical trajectories of the target agent and other agents on multiple channels to represent agents' physical states at each timestamp. In the same branch, [5], [6] adopt a similar rasterized representation method. However, rasterized representation methods are limited to restricted image size and CNNs have constrained receptive fields. Besides, they cannot capture the complex structural information of HD map and it is difficult for them to represent the agents' dynamics.

More recently, vectorized representation methods are proposed to capture the structural information of traffic scenes. They use graph neural networks (GNNs) to capture the spatial features of agents and HD map and to learn the relationships between them. VectorNet [7] is the first to use the vectorized representation method to encode both HD map and agents' dynamics. TNT [8] and DenseTNT [9] adopt the same representation method and use VectorNet as their backbone network. LaneGCN [10] treats lane segments as nodes and constructs a lane graph from the lane segments. [11]–[14] adopt similar lane graph representation methods as LaneGCN. TPCN [15] and its extension DCMS [16] treat trajectory prediction problems as joint temporal point cloud learning and use PointNet++ [17] in the point representation learning. HDGT [18] models the traffic scene as a heterogeneous graph with different types of nodes and edges. The approach most related to this paper is HiVT [1], 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.

III. TEMPORAL TRAFFIC SCENE REPRESENTATION

We adopt the feature representation of HiVT and extend it. In this section, we introduce our feature representation in details. Our introduced extended features of the traffic scene and the original features of HiVT in their definitions are presented in blue and black respectively.

A. Node Feature Representation

For traffic agents, we extract their trajectory segments which are in a form of directed splines with respect to time. We consider each agent at each timestamp as a node and we characterize these temporal geometric nodes' attributes as $\mathbf{n}_a = \{\mathbf{R}_i^{T\top}\mathbf{d}_i^t, \mathbf{R}_i^{T\top}\mathbf{v}_i^t, s_i^t, \mathbf{R}_i^{T\top}\mathbf{a}_i^t, \Delta t^t, \mathbf{b}_i | i = 1, ..., N_t, t = 0, ..., 19\}$ with

$$\mathbf{p}_i^t = [x_i^t, y_i^t] \qquad s_i^t = \|\mathbf{v}_i^t\| \tag{1}$$

$$\mathbf{d}_i^t = \mathbf{p}_i^t - \mathbf{p}_i^{t-1} \qquad \alpha_i^t = \arctan \frac{{d_{y_i}}^t}{{d_{x_i}}^t} \tag{2}$$

$${d_{x_i}}^t = x_i^t - x_i^{t-1} \qquad \mathbf{a}_i^t = [\cos(\alpha_i^t), \sin(\alpha_i^t)] \tag{3}$$

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

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

where N_t represents the total number of agents at timestamp t, $\mathbf{p}_i^t$ is the location vector which consists of the lateral and longitudinal position of agent i at timestamp t, $\mathbf{d}_i^t$ is the displacement vector from timestamp $t-1$ to timestamp t, $\mathbf{v}_i^t$ is the velocity, s_i^t is the speed, $\mathbf{a}_i^t$ is the heading vector which consists of the cosine and sine of the agent's heading α_i^t, Δt^t is the timestamp difference, $\mathbf{R}_i^T$ is the rotation matrix parameterized by the heading $\alpha_i^\top$ at the current timestamp T (19), and $\mathbf{b}_i$ corresponds to semantic attribute. The sampling rate is 10Hz, but we found it deviates in different scenes, thus we include the timestamp difference into the node feature.

For road network, we extract coordinates of lane segments and their semantic attributes, such as turn directions. We similarly vectorize lane segments as the vectorization for agents. The geometric node attributes of lane segments $\mathbf{n}_l = \{\mathbf{d}_\xi, \mathbf{a}_\xi, l_\xi, \mathbf{b}_\xi | \xi = 1, ..., M\}$ consist of displacement vector $\mathbf{d}_\xi$ of each lane segment, heading vector $\mathbf{a}_\xi$, length of the lane segment l_ξ, and also the semantic attribute $\mathbf{b}_\xi$, where

$$\mathbf{d}_\xi = \mathbf{p}_\xi^1 - \mathbf{p}_\xi^0 \qquad l_\xi = \|\mathbf{d}_\xi\| \tag{6}$$

$$\mathbf{a}_\xi = [\cos(\alpha_\xi), \sin(\alpha_\xi)] \tag{7}$$

M is the total number of lane segments, $\mathbf{p}_\xi^0$ and $\mathbf{p}_\xi^1$ represent the start point and end point of the lane segment ξ respectively. Instead of using absolute representations, using relative representations ensure these state quantities are not affected by coordinate translation transformations.

B. Edge Feature Representation

The node features do not preserve relative spatial relationships. Thus, we further introduce the geometric edge attributes between entities. We characterize the edge attributes between agents as $\mathbf{e}_{aa} = \{\mathbf{R}_i^{T\top}\mathbf{d}_{ij}^t, \mathbf{R}_i^{T\top}\mathbf{v}_{ij}^t, l_{ij}, s_{ij}, \mathbf{d}_{j2i}^t, \mathbf{v}_{j2i}^t, \mathbf{a}_{j2i}^t | t = 0, ..., 19; i, j = 1, ..., N_t, i \neq j\}$, where

$$\mathbf{d}_{ij}^t = \mathbf{p}_j^t - \mathbf{p}_i^t \qquad \alpha_{ij}^t = \alpha_j^t - \alpha_i^t \tag{8}$$

$$l_{ij}^t = \|\mathbf{d}_{ij}^t\| \qquad \mathbf{a}_{j2i}^t = [\cos(\alpha_{ji}^t), \sin(\alpha_{ji}^t)] \tag{9}$$

$$\mathbf{v}_{ij}^t = \mathbf{v}_j^t - \mathbf{v}_i^t \qquad \mathbf{d}_{j2i}^t = \mathbf{R}_i^{t\top}\mathbf{d}_{ji}^t \tag{10}$$

$$s_{ij}^t = \|\mathbf{v}_{ij}^t\| \qquad \mathbf{v}_{j2i}^t = \mathbf{R}_i^{t\top}\mathbf{v}_{ji}^t \tag{11}$$

where $\mathbf{d}_{ij}^t$ is the relative position vector between the target agent i and agent j at timestamp t, l_{ij}^t is the distance, $\mathbf{v}_{ij}^t$ is the relative velocity and s_{ij} is the relative speed. $\mathbf{R}_i^t$ is the rotation matrix parameterized by the heading α_i^t of the target agent i at the timestamp t. $\mathbf{d}_{j2i}^t$ and $\mathbf{v}_{j2i}^t$ are

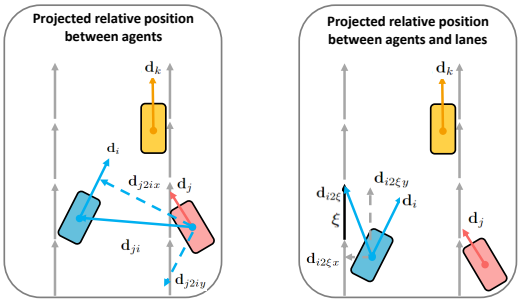

Fig. 1. In the figure on the left, the blue agent i is defined as the target agent, $\mathbf{d}_{ji}$ is the relative position vector between agent i and agent j, $\mathbf{d}_{j2ix}$ and $\mathbf{d}_{j2iy}$ is the lateral and longitudinal projected relative position. In the figure on the right, $\mathbf{d}_{i2\xi}$ is the relative position vector between agent i and the black lane segment ξ, $\mathbf{d}_{i2\xi x}$ and $\mathbf{d}_{i2\xi y}$ is the lateral and longitudinal projected relative position.

the projected relative position vector and projected relative velocity respectively, and $\mathbf{a}^t_{j2i}$ is the relative heading vector. The projected relative position is defined as projecting relative position in the direction of the heading of the target agent. An example is shown in the left side of Fig. 1. Similarly, we obtain the projected relative velocity by projecting relative velocity in the direction of the heading of the target agent.

We characterize the edge features between agents and lane segments as $\mathbf{e}_{al} = \{\mathbf{R}_i^{T^\top}\mathbf{d}^t_{i\xi}, \mathbf{d}^t_{i2\xi}, l^t_{i\xi}, \mathbf{v}^t_{i2\xi}, \mathbf{a}^t_{i2\xi} | t = 0, ..., 19; i = 1, ..., N_t; \xi == 1, ..., M\}$, where

$$\mathbf{d}^t_{i\xi} = \mathbf{p}^0_\xi - \mathbf{p}^t_i \qquad \mathbf{v}^t_{i2\xi} = \mathbf{R}_\xi^{t^\top}\mathbf{v}^t_i \tag{12}$$

$$\mathbf{d}^t_{i2\xi} = \mathbf{R}_\xi^{t^\top}\mathbf{d}^t_{i\xi} \qquad \alpha^t_{i\xi} = \alpha^t_\xi - \alpha^t_i \tag{13}$$

$$l^t_{ij} = \|\mathbf{d}^t_{i\xi}\| \qquad \mathbf{a}^t_{i2\xi} = [\cos(\alpha^t_{i\xi}), \sin(\alpha^t_{i\xi})] \tag{14}$$

where $\mathbf{d}^t_{i\xi}$ is the relative position vector between agent i and lane segment ξ at timestamp t, $l^t_{i\xi}$ is the distance and $\mathbf{R}_\xi$ is the rotation matrix parameterized by the heading α_ξ of the target lane. $\mathbf{d}^t_{i2\xi}$ and $\mathbf{v}^t_{i2\xi}$ represent the projected relative position vector and projected relative velocity respectively, and $\mathbf{a}^t_{i2\xi}$ is the relative heading vector. The projected relative position between lane segment ξ and agent i is defined as projecting relative position in the direction of the heading of the target lane segment ξ. An example is shown in the right side of the Fig. 1. Similarly, we obtain the projected relative velocity $\mathbf{v}^t_{i2\xi}$ by projecting the agent's velocity in the direction of the heading of the target lane segment.

The projected relative position and the projected relative velocity depict how close two individual nodes are and how fast one node moves towards another node in both lateral and longitudinal directions. Compared to relative representation, projected relative representation describes the interaction between entities in a more detailed way, allowing downstream networks to better understand their behavior. Moreover, such a projected relative representation guarantees translation invariance and rotation invariance naturally.

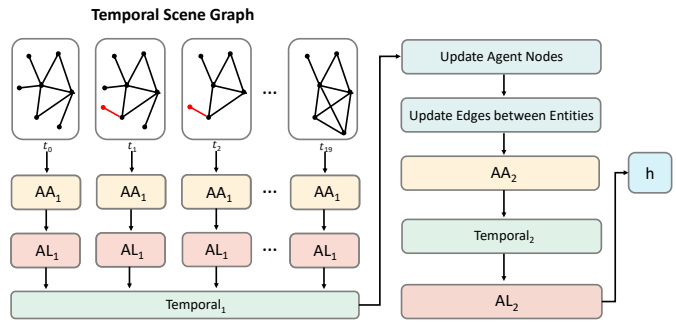

Fig. 2. Local Encoder of TSGN. The local encoder processes the temporal traffic scene in two stages. In the first stage, it models the agent-agent interactions (A-A) and agent-lane interactions (A-L) per time step and uses a temporal transformer module to capture the temporal dependencies across traffic scenes. In the second stage, the refreshed agents' features are used to update the relative spatial relationships between entities, and the agent-agent interaction is modeled again per time step. Then a temporal transformer with a classification token for summarizing the temporal information and an agent-lane transformer focusing only on the current timestamp with a larger receptive field are applied. $\mathbf{h}$ is the final local representation for all agents.

IV. PREDICTION FRAMEWORK

In this section, we introduce our prediction framework TSGN which consists of a local encoder, a global encoder, and a multimodal decoder as well as the components of the local encoder. Following that, we cover details about our proposed Hierarchical Lane Transformer.

A. Local Encoder

The local encoder operates the temporal scene graph in two stages. In the first stage, it models agent-agent interactions and agent-lane interactions per time step and uses a temporal transformer module to capture temporal dependencies across traffic scenes. Since the number of agent-lane edges is large, we use a reduced receptive field of view in this stage for efficiency. In the second stage, it uses the refreshed agents' features to update relative spatial relationships between entities and models agent-agent interactions per time step again, followed by a temporal transformer with a classification token for summarizing the temporal information and an agent-lane transformer only focusing on the current timestamp with a larger receptive field, we obtain the final updated features for each agent. In summary, the local encoder of TSGN contains the following modules: Agent-Agent Transformer in the first stage (AA_1) and second stage (AA_2), Agent-Lane Transformer in the first stage (AL_1 for all timestamps) and second stage (AL_2 only for the current timestamp), and Temporal Transformer in the first stage ($\mathrm{Temporal}_1$) and second stage ($\mathrm{Temporal}_2$). An overview of the local encoder is shown in Fig. 2. In the remainder of this subsection, we first describe the Agent-Agent Transformer, Agent-Lane Transformer, and Temporal Transformer separately, and then we detail how these transformers are going to be used in the two different stages.

Agent-Agent Transformer

We use a multi-head cross-attention block to learn the interaction relationships and the importance of the influence

of different surrounding agents on the center agent for each local region at each time step. Specifically, first, we apply Multi-Layer Perceptrons (MLPs) on its node attributes and edge attributes, then we obtain the central agent i's temporal node embedding $\mathbf{z}_i^t$ and any of its associated temporal edge embedding $\mathbf{z}_{ij}^t$ which is connected with neighboring agent j:

$$\mathbf{z}_i^t = \phi_{\text{center}}\Big([\mathbf{R}_i^{T\top}\mathbf{d}_i^t, \mathbf{R}_i^{T\top}\mathbf{v}_i^t, \mathbf{R}_i^{T\top}\mathbf{a}_i^t, s_i^t, \Delta t^t, \mathbf{b}_i]\Big) \tag{15}$$

$$\mathbf{z}_{ij}^t = \phi_{\text{nbr}}\Big([\mathbf{R}_i^{T\top}\mathbf{d}_j^t, \mathbf{R}_i^{T\top}\mathbf{d}_{ij}^t, \mathbf{R}_i^{T\top}\mathbf{v}_{ij}^t, l_{ij}^t, s_{ij}^t, \mathbf{d}_{j2i}^t, \mathbf{v}_{j2i}^t, \mathbf{a}_{j2i}^t, \mathbf{b}_j]\Big) \tag{16}$$

where ϕ_{center} and ϕ_{nbr} are MLP blocks. Both node and edge attributes are translation and rotation invariant, their embeddings also inherit these properties. Then, a cross attention block takes $\mathbf{z}_i^t$ as query, $\mathbf{z}_{ij}^t$ as key, $\mathbf{z}_{ij}^t$ as value, performs the scaled dot-product attention [19] with a gating function [1] , and outputs $\hat{\mathbf{z}}_i^t$. We further apply an MLP block in conjunction with residual connections [20] and takes $\hat{\mathbf{z}}_i^t$ as input to obtain the first updated node embedding $\mathbf{s}_i^t$ of agent i at timestamp t which incorporates interactions between agents.

Agent-Lane Transformer

To capture implicit interactions between agents and lane segments, we employ another multi-head cross-attention block. First, we encode the local lane segments, and their associated edge attributes which are connected with the central agent i by an MLP block ϕ_{lane} to obtain the edge embedding:

$$\mathbf{z}_{i\xi} = \phi_{\text{lane}}\Big([\mathbf{R}_i^{T\top}\mathbf{d}_\xi, \mathbf{R}_i^{T\top}\mathbf{d}_{i\xi}, \mathbf{d}_{i2\xi}, l_{i\xi}, \mathbf{v}_{i2\xi}, \mathbf{a}_{i2\xi}, \mathbf{b}_\xi]\Big) \tag{17}$$

Second, we employ a multi-head cross-attention block that takes agent i's node embedding $\mathbf{s}_i^t$ as query, the edge embedding $\mathbf{z}_{i\xi}$ as key and value, and performs the scaled dot-product attention the same as the Agent-Agent Transformer, followed by an MLP block in conjunction with residual connections. We then obtain the updated node embedding $\widetilde{\mathbf{s}}_i^t$ which incorporates interactions between agents and interactions between agents and lane segments.

Temporal Transformer

We treat the encoded traffic scene at each timestamp as a whole and use a temporal transformer to capture temporal dependencies of all historical encoded traffic scenes. First, we add the learnable positional embeddings [19] to the central agent i's node embedding $\{\widetilde{\mathbf{s}}_i^t\}_{t=1}^T$ at each timestamp and stack them into a matrix $\widetilde{\mathbf{S}}_i$. Then, we use a self-attention block to perform the scaled dot-product attention, in which $\widetilde{\mathbf{S}}_i$ is taken as query, key and value. We obtain the output of the temporal transformer $\hat{\mathbf{S}}_i$ which summarizes the spatial-temporal features of agent i for every timestamp.

1) First Stage

The local encoder models agent-agent interactions and agent-lane interactions per time step. These two interaction modules aggregate the spatial information of the traffic scene for every timestamp. As explained before, due to the great computational complexity, we use a reduced receptive field of view in this stage for efficiency. Taking the encoded temporal traffic scene as input, the temporal transformer further summarizes the temporal information. Unlike most approaches which only capture temporal dependencies of agents' historical dynamics, we propose to incorporate agent-agent interactions and agent-lane interactions to update agents' node embedding for every timestamp, since the interactions in the past will affect the agents' pattern in the future.

2) Second Stage

We use refreshed node embeddings of agents in the first stage to update the edge embeddings between agents as:

$$\mathbf{z}_{ij_2}^t = \phi_{\text{nbr}_2}\Big([\hat{\mathbf{s}}_i^t, \mathbf{z}_{ij}^t]\Big) \tag{18}$$

where $\{\hat{\mathbf{s}}_i^t\}_{t=1}^T$ is the unstacked result of $\hat{\mathbf{S}}_i$ and ϕ_{nbr_2} is an MLP block. We use refreshed agents' nodes embeddings $\{\hat{\mathbf{s}}_i^t\}_{t=1}^T$ and edges embeddings $\mathbf{z}_{ij_2}^t$ to repeat an agent-agent interaction encoding for every timestamp. In the first modeling of agent-agent interactions, the agents lacked information about road network, which could in turn affect their behaviors and interactions. Thus, we apply it again, after all agents are incorporated with the road network information. Afterwards, we apply another temporal transformer with an extra learnable classification token [21] and a temporal mask that enforces the tokens only focus on the preceding time steps to summarize the whole temporal sequence into one single token $\bar{\mathbf{s}}_i$. Then, the edge embeddings between agents and lane segments are calculated as:

$$\mathbf{z}_{i\xi_2} = \phi_{\text{lane}_2}\Big([\bar{\mathbf{s}}_i, \mathbf{z}_{i\xi}]\Big) \tag{19}$$

where ϕ_{lane_2} is an MLP block. We input the updated extra classification token $\bar{\mathbf{s}}_i$ as query and updated edge embeddings $\mathbf{z}_{i\xi_2}$ between agent i and lane segment ξ as key and value to another same agent-lane interaction module with a larger receptive field. We obtain the final node embedding $\mathbf{h}_i$ of the central agent i. $\mathbf{h}_i$ summarizes the fused rich spatial-temporal features of the agent i. It incorporates agent i's dynamics and the iterative interactions between agent i and its surrounding environment. The final local representation for all agents $\mathbf{h}$ is defined as $\mathbf{h} = \{\mathbf{h}_i | i = 1, ..., N\}$, where N is the number of agents.

B. Global Encoder

The local encoder output represents the features of the central agent within its local region, but it does not incorporate the pairwise interactions between local regions and lack the long-range dependencies in the scene. Thus, we apply a global encoder [1] for high-order interactions.

Similar to the Agent-Agent Transformer, we use an MLP ϕ_{rel} to obtain the edge embedding $\mathbf{g}_{ij}$ between agent i's and agent j's local regions:

$$\mathbf{g}_{ij} = \phi_{\text{rel}}\Big([\mathbf{R}_i^{T\top}\mathbf{d}_{ij}^T, \mathbf{R}_i^{T\top}\mathbf{v}_{ij}^T, l_{ij}^T, s_{ij}^T, \mathbf{d}_{j2i}^T, \mathbf{v}_{j2i}^T, \mathbf{a}_{j2i}^T]\Big) \tag{20}$$

where T represents the current timestep. We take $\mathbf{h}_i$ as query, $[\mathbf{h}_j, \mathbf{g}_{ij}]$ as key and value. A multi-head cross-attention

block receives them as input and outputs agent i's updated node embedding which incorporates global scene information. Afterwards, we employ an MLP block to map the updated node embedding of each agent to the final global representation $\widetilde{\mathbf{h}}$ with shape $[F, N, D]$, where F is the number of mixture components, representing the multimodality of agents' trajectories in the future, and D is the number of agents' features.

C. Multimodal Decoder

In complex traffic scenes, agents can behave multimodally. To characterize this phenomenon, we apply a mixture density network [22] with GRU [23] as the decoder to output multimodal trajectories for each agent. Specifically, we tile the global representation $\widetilde{\mathbf{h}}$ with shape $[F, N, D]$ and the local representation $\mathbf{h}$ with shape $[N, D]$ H and F times, then we obtain the tiled global representation with shape $[H, F \times N, D]$ and the tiled local representation with shape $[1, F \times N, D]$, where H is the number of future steps. The GRU receives the tiled global representation as inputs and the tiled local representation as the initial hidden state and outputs $\mathbf{o}$ with shape $[H, F \times N, D]$. We transpose the first dimension and second dimension of $\mathbf{o}$ and apply two MLP blocks ϕ_{loc}, ϕ_{unc} which take the GRU output $\mathbf{o}$ as input and output locations $\boldsymbol{\mu}$ with shape $[F, N, H, 2]$ and associated uncertainties $\mathbf{b}$ with shape $[F, N, H, 2]$. We apply another MLP block ϕ_{prob} with the concatenation of $\mathbf{h}$ and $\widetilde{\mathbf{h}}$ as input to obtain the probability $\boldsymbol{\pi}$ of possible future trajectories of all agents:

$$\mathbf{o} = \text{GRU}([\widetilde{\mathbf{h}}, \mathbf{h}]) \qquad \boldsymbol{\mu} = \phi_{\text{loc}}(\mathbf{o}) \tag{21}$$

$$\mathbf{b} = \phi_{\text{unc}}(\mathbf{o}) \qquad \boldsymbol{\pi} = \phi_{\text{prob}}([\widetilde{\mathbf{h}}, \mathbf{h}]) \tag{22}$$

D. Loss Function

The tracking data of Argoverse dataset 1 contains noisy samples. Compared to Gaussian negative log-likelihood loss, Laplace negative log-likelihood loss is more robust to outliers, thus we design the multimodal decoder in that way so that it predicts a parameterization of the location and its associated uncertainty corresponding to a Laplace distribution [24]. We use the winner-take-all strategy, and only apply the loss to the best-predicted trajectory which is defined as the one that has the minimum endpoint error. Our final loss consists of the regression loss $\mathcal{L}_{\text{reg}}$ and the classification loss $\mathcal{L}_{\text{cls}}$:

$$\mathcal{L}_{reg} = -\frac{1}{NH} \sum_{i=1}^{N} \sum_{t=T+1}^{T+H} \log P\Big(\mathbf{R}_i^{T\top}(\mathbf{p}_i^t - \mathbf{p}_i^T) | \hat{\boldsymbol{\mu}}_i^t, \hat{\mathbf{b}}_i^t\Big) \tag{23}$$

$$\overline{\boldsymbol{\pi}}_i = \text{softmax}\Big(- \sum_{t=T+1}^{T+H} \|\boldsymbol{\mu}_i^t, \mathbf{R}_i^{T\top}(\mathbf{p}_i^t - \mathbf{p}_i^T)\|\Big) \tag{24}$$

$$\mathcal{L}_{cls} = \sum_{i=1}^{N} \text{CE}(\overline{\boldsymbol{\pi}}_i, \hat{\boldsymbol{\pi}}_i) \qquad \mathcal{L} = \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{cls}} \tag{25}$$

where$\hat{\boldsymbol{\mu}}_i^t$ and $\hat{\mathbf{b}}_i^t$ are the locations and their associated uncertainty of the best-predicted trajectory of agent i at timestamp t. $\boldsymbol{\mu}_i^t$ is the locations of all predicted trajectories of agent i at timestamp t. CE is the cross-entropy loss.

E. Hierarchical Lane Transformer

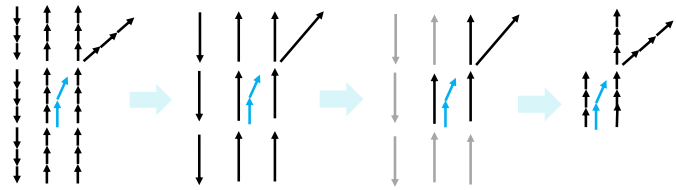

Fig. 3. Hierarchical process. First, the road network is constructed by lanelets instead of lane segments. Second, Agent-Lanelet Transformer selects the lanelets which score are higher than threshold. Finally, only lane segments that belong to the selected lanelets are taken into account in calculating agent-lane interactions. In this example, a lanelet consists of 3 lane segments. The selected lanelets are shown in black, the unselected lanelets are shown in gray, and the past trajectory of the target agent is shown in blue.

The number of agent-lane edges in the local region is still very large, even if we limit the radius of the local region. When the model performs the agent-lane interaction for every timestamp, it suffers from great time complexity and space complexity. In addition, not all lane segments in the local region have an impact on the agents' future behavior. Since a lanelet [25] can be viewed as a set of lane segments, we can filter out the unuseful lanelets and lane segments belonging to them to reduce the computational complexity. In our case, a lanelet is defined as a sequence of 10 centerline points. We propose the Hierarchical Lane Transformer and apply it in the agent-lane interaction module to only select lanelets that are most important to the target agent. As the other unselected lanelets would have scarcely any influence on the target agent, this does not decrease performance. We will demonstrate it in the ablation studies.

Specifically, the Hierarchical Lane Transformer consists of an Agent-Lanelet Transformer and an Agent-Lane Transformer. The Agent-Lanelet Transformer has the same architecture as the Agent-Lane Transformer. We take the average position and the average position vector of all lane segments of lanelet Ξ as the position $\mathbf{p}_\Xi$ and position vector $\mathbf{d}_\Xi$ of the lanelet Ξ. We create the edge embedding $\mathbf{z}_{i\Xi}$ between agent i and lanelet Ξ as follows:

$$\mathbf{z}_{i\Xi} = \phi_{\text{lanelet}}\Big([\mathbf{R}_i^{T\top}\mathbf{d}_\Xi, \mathbf{R}_i^{T\top}\mathbf{d}_{i\Xi}, \mathbf{d}_{i2\Xi}, l_{i\Xi}, \mathbf{v}_{i2\Xi}, \mathbf{h}_{i2\Xi}, \mathbf{a}_\Xi]\Big) \tag{26}$$

In the first step, the Agent-Lanelet Transformer performs the scaled dot-product attention and yields a score $\boldsymbol{\alpha}_{i\Xi}^t$ for each lanelet to identify the most probably lanelets that can affect the future motion of the target agent i. In the second step, only lane segments whose score are greater than threshold are taken into account in calculating agent-lane interactions. To enhance the influence of the lanelet's score $\boldsymbol{\alpha}_{i\Xi}^t$, we multiply the scaled dot-product in Agent-Lane Transformer with $\boldsymbol{\alpha}_{i\Xi}^t$ before softmax function:

$$\boldsymbol{\alpha}_{i\xi}^t = \boldsymbol{\alpha}_{i\Xi}^t \cdot \text{softmax}\Big(\frac{\mathbf{q}_i^{t\top}}{\sqrt{d_k}} \cdot \mathbf{k}_{i\xi}^t\Big) \tag{27}$$

TABLE I
RESULTS ON ARGOVERSE MOTION FORECASTING LEADERBOARD.

Models	minFDE	minADE	MR
LaneGCN [10]	1.3640	0.8679	0.1634
Scene Transformer [28]	1.2321	0.8026	0.1255
HOME+GOME [12]	1.2919	0.8904	**0.0846**
DenseTNT [9]	1.2815	0.8817	0.1258
TPCN [15]	1.2442	0.8153	0.1333
MultiPath++ [29]	1.214	0.790	0.13
HiVT [1]	1.1693	0.7735	0.1267
DCMS [16]	**1.1350**	0.7659	0.1094
Wayformer [30]	1.1615	0.7675	0.1186
TSGN	1.1370	**0.7537**	0.1223

where the lane segment ξ belongs to the lanelet Ξ. An example of this hierarchical process is shown in Fig. 3.

V. EXPERIMENTS

A. Experimental Settings

1) Dataset

We use the Argoverse Motion Forecasting Dataset [26]. It is a dataset with agent trajectories and HD map which contains 324557 real traffic scenarios. The training, validation, and test sets contain 205942, 39472, and 78143 scenarios respectively. Each scenario is a 5-second sequence long sampled at 10 Hz and contains the position of all agents in the past 2 seconds. In Argoverse Motion Forecasting Challenge, we need to predict the 3-second future positions of one target agent given the initial 2-second observations of the scene.

2) Metrics

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

3) Implementation Details

We train all models for 64 epochs using AdamW optimizer [27] initialized with a learning rate of 0.0005. We employ the cosine annealing scheduler for the learning rate decay. We set the number of layers for Agent-Agent Transformer, Agent-Lane Transformer, Agent-Lanelet Transformer, Temporal Transformer, and Global Encoder to 1,1,1,4,3. The number of hidden units is 128, the number of heads in all multi-head attention blocks is 8. The radius of local regions in the first stage is set to 20 meters, while in the second stage is set to 50 meters. We drop the agent which moved less than 6 meters unless it is the target agent [28].

B. Comparison with State-of-the-art

We show in Tab. I the results of TSGN compared to other state-of-the-art models on the Argoverse motion forecasting test set. The data in Tab. 1 is collected from the Argoverse leaderboard on 13/09/2022. TSGN outperforms all the other methods in terms of minADE, and remains competitive ranking on minFDE, which verifies the superior prediction performance of our method.

TABLE II
IMPORTANCE OF EACH COMPONENT OF OUR FRAMEWORK.

AA_1	AL_2	$Temporal_1$	AL_1	AA_2	$Temporal_2$	minFDE	minADE	MR
✓	✓	✓				0.959	0.652	0.090
✓	✓	✓	✓			0.930	0.644	0.085
✓	✓	✓	✓	✓	✓	**0.916**	**0.636**	**0.084**

TABLE III
ABLATION STUDIES ON THE EXTENDED SCENE REPRESENTATION.

Representation method	minFDE	minADE	MR
raw scene representation	0.967	0.662	0.092
extended scene representation	**0.959**	**0.652**	**0.090**

C. Ablation Studies

Our ablation studies consist of four parts: the importance of each module of TSGN, the importance of the extended scene reresentation and the importance of the Hierarchical Lane Transformer. We conduct these experiments on the Argoverse validation set.

1) Importance of Each Module

HiVT consists of AA_1, AL_2, $Temporal_1$, and Global modules. The functions of these 4 modules can be seen in [1]. As shown in Tab. II, AL_1, AA_2, and $Temporal_2$ modules can improve minADE, minFDE, and MR to a certain degree. First, every moment of interaction will affect future behavior. AL_1 captures the agent-lane interaction at each timestamp, it noticeably improves the prediction performance. Second, the components of the second stage of the local encoder (AA_2, $Temporal_2$) have a significant impact on the performance. As discussed before, AA_1 does not consider the influence that the agent-lane interaction has on the agent-agent interaction. Based on AA_1, AA_2 further captures the agent-agent interaction incorporated with the map information.

2) Importance of the Extended Scene Reresentation

In this ablation study, we use HiVT [1] as the prediction framework. As shown in Tab. III, compared to the original representation, the extended scene representation yields better performance. Our proposed representation provides more information about the entities and more finely describes the relative relationships between these entities, allowing downstream networks to better understand the interactions between individuals in a complex scene.

3) Importance of the Hierarchical Lane Transformer

In this ablation study, we use TSGN as the prediction framework. We present the performance evaluation of the Hierarchical Lane Transformer in Tab. IV. In practice, we select the lanelets whose score $\boldsymbol{\alpha}_{i\Xi}^{t}$ is greater than $0.75 \cdot \overline{\boldsymbol{\alpha}_{i\Xi}^{t}}$,

TABLE IV
ABLATION STUDIES ON THE LANELETS SELECTION MODULE.

A-L module	minFDE	minADE	MR	Lane usage rate	Speed (ms)
w/o Lanelet selection	**0.916**	**0.636**	**0.084**	100%	331
w/ Lanelet selection	0.922	0.638	0.085	**45.8%**	**226**

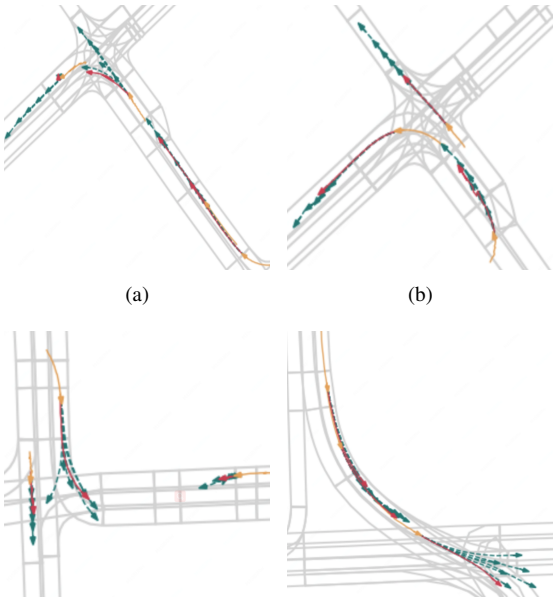

Fig. 4. Qualitative Results of TSGN. For clarity, we visualize only agents which are observed at current timestamp and move more than 6 meters. We present the past trajectories in orange, the ground-truth trajectories in red and the predicted trajectories in green.

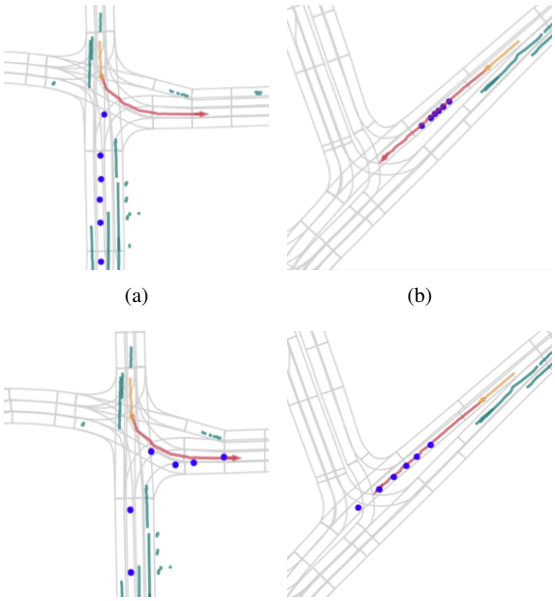

Fig. 5. Qualitative results of HiVT and TSGN are shown in the first row and second row respectively. The central agent's past trajectory is shown in orange, the ground-truth trajectory is shown in red, and the endpoints of predicted trajectories are shown in blue. The past trajectories of other agents are shown in green.

where $\overline{\boldsymbol{\alpha}_{i\Xi}^{t}}$ represents the scatter mean score of all lanelets which have a shared edge with agent i. We measure the resulting average proportion of selected lanes to be 45.8%, dividing the number of processed lane features by more than 2. As a consequence, the computational complexity of agent-lane interaction is significantly reduced and the measured inference speed improves greatly. Moreover, the Hierarchical Lane Transformer also keeps the prediction performance. As shown in Tab. IV, the minFDE, minADE and MR are barely significantly worse.

D. Qualitative Results of TSGN

From Fig. 4(a) to Fig. 4(d), we visualize the qualitative results of TSGN in 4 traffic scenes. One can conclude that TSGN can make multimodal predictions for all agents simultaneously and the prediction in each maneuver is reasonable. Comparing the best-predicted trajectory with the ground truth, we see that TSGN can make accurate predictions.

E. Comparison with HiVT in Worse Cases

To compare with HiVT, we present the qualitative results of TSGN and HiVT in some worse cases of HiVT, in which the minFDE of HiVT is greater than 10 meters. For clarity, we only visualize end points of the 6 predicted trajectories of the central agent. We compare these two models in two different traffic scenes. Fig. 5(a) and Fig. 5(b) show the results of HiVT, while Fig. 5(c) and Fig. 5(d) show the results of TSGN. From Fig. 5(a) and Fig. 5(c), we observe that TSGN can more successfully predict reasonable multimodal future trajectories for agents. In addition, as shown in Fig. 5(b) and Fig. 5(d), TSGN is more sensitive to speed-related information. When the agent is accelerating or decelerating, it can better perceive the current motion state of the agents.

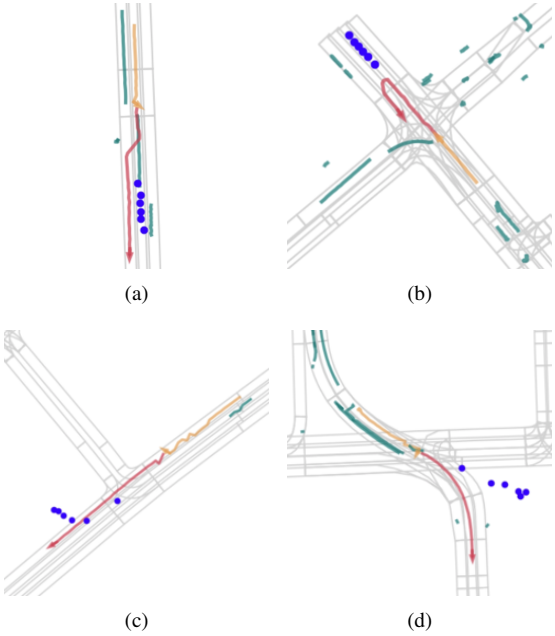

Fig. 6. Failed cases of TSGN. The visualization method is the same as Fig.5.

F. Failed Cases

From Fig. 6(a) to Fig. 6(d), we present some classic failed cases of TSGN. First, as shown in Fig. 6(a) and Fig. 6(b), we

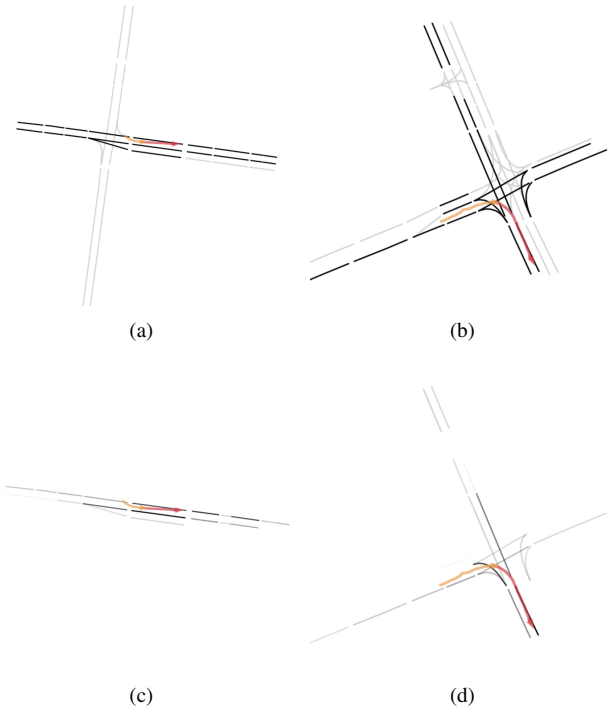

Fig. 7. Selected lanelets by Hierarchical Lane Transformer and their' scores. The agent's past trajectory is shown in orange, and the ground-truth trajectory is shown in red. In the first row, selected lanelets and unselected lanelets are presented in black and grey respectively. In the second row, the depth of the selected lanelet's color represents the level of the score. The higher the score, the more important the lanelet is to the target agent.

conclude that it's hard for TSGN to make accurate predictions when the agents intend to perform maneuver 'lane change' or 'U turn'. One possible cause of the failure is, compared with conventional maneuvers, such as 'Go straight', 'left turn' and 'right turn', the proportion of maneuver 'lane change' and 'U-turn' is very low. It is difficult for the model to learn to predict these kinds of intentions. Second, in Fig. 6(c), we observe that the prediction performance of TSGN is not robust, if the tracking information of the agent in the past is unstable. The prediction results seem to be overfitting to past perception errors. Third, when the agent moves in a complex traffic scene where the surrounding lane segments are extraordinarily tangled, e.g., the traffic scene in Fig. 6(d), TSGN fails in understanding the map structure and makes unreasonable multimodal predictions. We want TSGN to output distinct maneuvers, but it only presents predictions in one maneuver.

G. Visualization of Selected Lanelets by Hierarchical Lane Transformer and Their Scores

In Fig. 7(a) and 7(b), we present the selected lanelets by Hierarchical Lane Transformer of two different traffic scenes. Fig. 7(c) and 7(d) show the corresponding selected lanelets' scores. We observe that our model can select the most important lanelets to the target agent. The selected lanelets contain the possible future trajectories of the target agent. In addition, the prediction of lanelets' scores is reasonable.

VI. CONCLUSION

In this paper, we propose TSGN with projected vectorized representation for multi-agent trajectory prediction. It models the temporal traffic scene as a temporal scene graph and uses diverse transformer-based interaction modules and temporal dependencies modules to operate this temporal scene graph. Moreover, we present a Hierarchical Lane Transformer module inside TSGN, which effectively and efficiently captures the influence of the HD-map on the future motion of target agents. Experiments show our method achieves state-of-the-art performance on the Argoverse motion forecasting benchmark.

REFERENCES

[1] Z. Zhou, L. Ye, J. Wang, K. Wu, and K. Lu, "Hivt: Hierarchical vector transformer for multi-agent motion prediction," in *CVPR*, 2022.
[2] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou *et al.*, "Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving," in *WACV*, 2020.
[3] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen *et al.*, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *ICRA*, 2019.
[4] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Home: Heatmap output for future motion estimation," in *ITSC, 2021*.
[5] J. Hong, B. Sapp *et al.*, "Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions," in *CVPR*, 2019.
[6] Y. Chai, B. Sapp *et al.*, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," *arXiv*, 2019.
[7] J. Gao, C. Sun, H. Zhao *et al.*, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *CVPR*, 2020.
[8] H. Zhao, J. Gao, T. Lan, C. Sun *et al.*, "Tnt: Target-driven trajectory prediction," in *CoRL*, 2021.
[9] J. Gu, C. Sun, and H. Zhao, "Densetnt: End-to-end trajectory prediction from dense goal sets," in *ICCV*, 2021.
[10] M. Liang, B. Yang, R. Hu *et al.*, "Learning lane graph representations for motion forecasting," in *ECCV*, 2020.
[11] T. Gilles, S. Sabatini, D. Tsishkou *et al.*, "Thomas: Trajectory heatmap output with learned multi-agent sampling," *arXiv*, 2021.
[12] ——, "Gohome: Graph-oriented heatmap output for future motion estimation," in *ICRA*, 2022.
[13] W. Zeng, M. Liang *et al.*, "Lanercnn: Distributed representations for graph-centric motion forecasting," in *IROS*, 2021.
[14] N. Deo, E. Wolff, and O. Beijbom, "Multimodal trajectory prediction conditioned on lane-graph traversals," in *CoRL*, 2022.
[15] M. Ye, T. Cao, and Q. Chen, "Tpcn: Temporal point cloud networks for motion forecasting," in *CVPR*, 2021.
[16] M. Ye, J. Xu, X. Xu *et al.*, "Dcms: Motion forecasting with dual consistency and multi-pseudo-target supervision," *arXiv*, 2022.
[17] C. R. Qi, L. Yi *et al.*, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *NeurIPS*, 2017.
[18] X. Jia, P. Wu *et al.*, "Hdgt: Heterogeneous driving graph transformer for multi-agent trajectory prediction via scene encoding," *arXiv*, 2022.
[19] A. Vaswani, N. Shazeer, N. Parmar *et al.*, "Attention is all you need," *NeurIPS*, 2017.
[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[21] J. Devlin, M.-W. Chang *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv*, 2018.
[22] C. M. Bishop, "Mixture density networks," 1994.
[23] K. Cho, B. Van Merriënboer *et al.*, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv*, 2014.
[24] G. P. Meyer and N. Thakurdesai, "Learning an uncertainty-aware object detector for autonomous driving," in *IROS*, 2020.
[25] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *IV*, 2014.
[26] M.-F. Chang, J. Lambert, P. Sangkloy *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *CVPR*, 2019.
[27] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv*, 2017.
[28] J. Ngiam, B. Caine, *et al.*, "Scene transformer: A unified architecture for predicting multiple agent trajectories," *arXiv*, 2021.

[29] B. Varadarajan, A. Hefny *et al.*, "Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction," in *ICRA*, 2022.
[30] N. Nayakanti, R. Al-Rfou, *et al.*, "Wayformer: Motion forecasting via simple & efficient attention networks," *arXiv*, 2022.