HiVT: Hierarchical Vector Transformer for Multi-Agent Motion Prediction

Zikang Zhou[1] Luyao Ye[1] Jianping Wang[1,2] Kui Wu[3] Kejie Lu[4]
[1]City University of Hong Kong [2]City University of Hong Kong Shenzhen Research Institute
[3]University of Victoria [4]University of Puerto Rico at Mayagüez
{zikanzhou2-c,luyaoye2-c}@my.cityu.edu.hk
jianwang@cityu.edu.hk wkui@uvic.ca kejie.lu@upr.edu

Abstract

Accurately predicting the future motions of surrounding traffic agents is critical for the safety of autonomous vehicles. Recently, vectorized approaches have dominated the motion prediction community due to their capability of capturing complex interactions in traffic scenes. However, existing methods neglect the symmetries of the problem and suffer from the expensive computational cost, facing the challenge of making real-time multi-agent motion prediction without sacrificing the prediction performance. To tackle this challenge, we propose Hierarchical Vector Transformer (HiVT) for fast and accurate multi-agent motion prediction. By decomposing the problem into local context extraction and global interaction modeling, our method can effectively and efficiently model a large number of agents in the scene. Meanwhile, we propose a translation-invariant scene representation and rotation-invariant spatial learning modules, which extract features robust to the geometric transformations of the scene and enable the model to make accurate predictions for multiple agents in a single forward pass. Experiments show that HiVT achieves the state-of-the-art performance on the Argoverse motion forecasting benchmark with a small model size and can make fast multi-agent motion prediction.

1. Introduction

Navigating through dynamic environments in a safe maneuver is a critical mission of autonomous vehicles. To this end, autonomous vehicles need to understand the surroundings and anticipate the future on the road. However, it is challenging to accurately predict the future motions of nearby traffic agents, such as vehicles, bicycles, and pedestrians, whose goals or intents may be unknown. In multi-agent traffic scenarios, an agent's behavior is shaped by complex interactions with other agents. Such interactions further intertwine with the map-dependent traffic regulations, making it extremely difficult to understand the diverse behavior of multiple agents in the scene.

Recently, learning-based methods have demonstrated their effectiveness in the motion prediction task [9, 12, 31, 32, 36, 49]. Inspired by the progress in computer vision, some works rasterize the scenes into bird's eye view images and apply CNNs to make predictions [9, 12, 25]. Although these approaches are easy to implement with off-the-shelf image models, they are computationally expensive and have limited receptive fields. Given these limitations, recent works [17, 31, 49] employ a vectorized approach for more compact scene representations, which extracts a set of vectors or points from the trajectories and the map elements. The scenes are then processed by graph neural networks [6, 20, 29], Transformers [46], or point cloud models [39, 40, 47] to learn the relationships among vectorized entities such as trajectory waypoints and lane segments.

Existing vectorized approaches, however, are challenged by the need to make real-time motion predictions in rapidly changing traffic conditions. Since vectorized methods are generally not robust to translation and rotation of the reference frame, to mitigate this problem, recent research normalizes the scene to be centered at the target agent and to be aligned with the target agent's heading [17, 31, 49]. This remedy becomes problematic when a large number of agents in the scene need to be predicted, owing to the expensive cost of re-normalizing the scene and re-computing the scene features for each target agent. Further, existing works model *all-to-all* relationships in the space and the time dimensions to capture fine-grained interactions among the vectorized entities [38, 51], which inevitably leads to prohibitive computation with the increase of entities. As making accurate predictions in real time is critical for the safety of autonomous driving, we are thus motivated to push the state-of-the-art by developing a new framework to achieve faster and more accurate multi-agent motion prediction.

In a nutshell, our approach exploits the symmetries and the hierarchical structure in the problem of multi-agent motion prediction. We frame the motion prediction task in multiple stages and hierarchically model the relationships

between entities based on Transformers [46]. *In the first stage*, our framework avoids expensive all-to-all interaction modeling and extracts context features only *locally*. Specifically, we divide the scene into a set of local regions, where each local region is centered at one modeled agent. For each agent-centric local region, we extract context features from the local vectorized entities which contain rich information related to the central agent. *In the second stage*, to compensate for the restricted local receptive fields and capture long-range dependencies in the scene, we perform global message passing among agent-centric local regions by empowering the Transformer encoder with the geometric relationships between local reference frames. *Finally*, given the local and the global representations, a decoder produces future trajectories for all agents in a *single* forward pass. To further leverage the symmetries of the problem, we employ a scene representation that is agnostic about the translation of the global coordinate frame, in which we use relative positions to characterize all vectorized entities. Based on this scene representation, we introduce rotation-invariant cross-attention modules for spatial learning, which can learn local and global representations that are invariant to the rotation of the scene.

Our approach has the following clear advantages. *First*, by decomposing the problem into local context extraction and global interaction modeling, our approach can progressively aggregate information at different scales and model a large number of entities in the scene with high efficiency. *Second*, our method can learn representations robust to translation and rotation of the inputs via a translation-invariant scene representation and rotation-invariant spatial learning modules. *Third*, our model can make faster and more accurate predictions with much fewer parameters than the state-of-the-art approaches. We validate all the above advantages via extensive experiments on large-scale driving data. Our code will be publicly available.

2. Related Work

Traffic Scene Representation. Tackling the motion prediction problem requires learning rich representations from the elements of the traffic scene, including the high-definition map and the past trajectories of agents. Extensive works have used rasterized scenes as model inputs [4, 9, 12, 15, 19, 25, 43] and employed standard image models [11,24,26,44] for learning. Specifically, these methods extract map elements (*e.g.*, lane boundaries, crosswalks, traffic lights) from the high-definition map and use different colors or masks to render the scenes as bird's eye view images. The past trajectories of agents are either rasterized as additional image channels [9, 12] or processed by temporal models like RNNs [2, 41, 42].

Rasterized methods are compatible with the mature techniques in computer vision but are also costly and inefficient in learning. Recently, vectorized methods [17, 31, 49] have gained popularity for their efficient sparse encoding and the capability of capturing complex structural information. Unlike the rasterized approaches, these methods treat the scene as a set of entities associated with semantic and geometric attributes and learn the relationships between entities. VectorNet [17] models the interactions among lane and trajectory polylines with graph neural networks. It is also used as the backbone network by some following works [22,32,52]. LaneGCN [31] builds a lane graph from the lane segments and exploits multi-scale graph convolutional networks to learn representations for graph nodes. TPCN [49] extends point cloud models to learn from the spatial-temporal point set composed of trajectory waypoints and lane points. Our scene representation also falls into this category but differs in that all vectorized entities are characterized by relative positions, making our representation *translation-invariant*.

Motion Prediction. Since social interactions are ubiquitous in traffic scenarios and significantly impact the future motions of traffic agents, many motion prediction approaches have considered the dependencies between agents' behaviors and reasoned agent-agent interactions using social pooling [2,13,23], graph neural networks [8,27,31,37], or attention mechanisms [17, 30, 32, 36, 38, 43, 48, 50]. Inspired by the success of Transformer models [46] in a wide range of domains [5, 7, 14, 16], some recent works employ Transformers in the motion prediction task to model spatial relationships, temporal dependencies, and relationships between agents and map elements [21, 30, 32, 36, 38, 50, 51]. In comparison, our Transformer architecture differs from existing ones by learning local and global representations hierarchically. This hierarchical strategy helps the model learn multi-scale features and is more efficient than those performing all-to-all message passing along the space and the time axes. Moreover, we model multiple agents via an agent-centric representation that is invariant to translation and rotation of the scene. The hierarchical architecture and the symmetric designs enable our method to achieve the state-of-the-art prediction performance with fewer parameters and less computational cost than other approaches.

3. Approach

3.1. Overall Framework

An overview of our proposed framework is illustrated in Fig. 1. We first organize a traffic scene as a collection of vectorized entities. Based on this scene representation, our framework hierarchically aggregates spatial-temporal information in the scene. In the first stage, we encode rotation-invariant local context features for each agent. The aggregation of ego-motion, neighboring agents' motions, and local map structure can provide rich information related to the

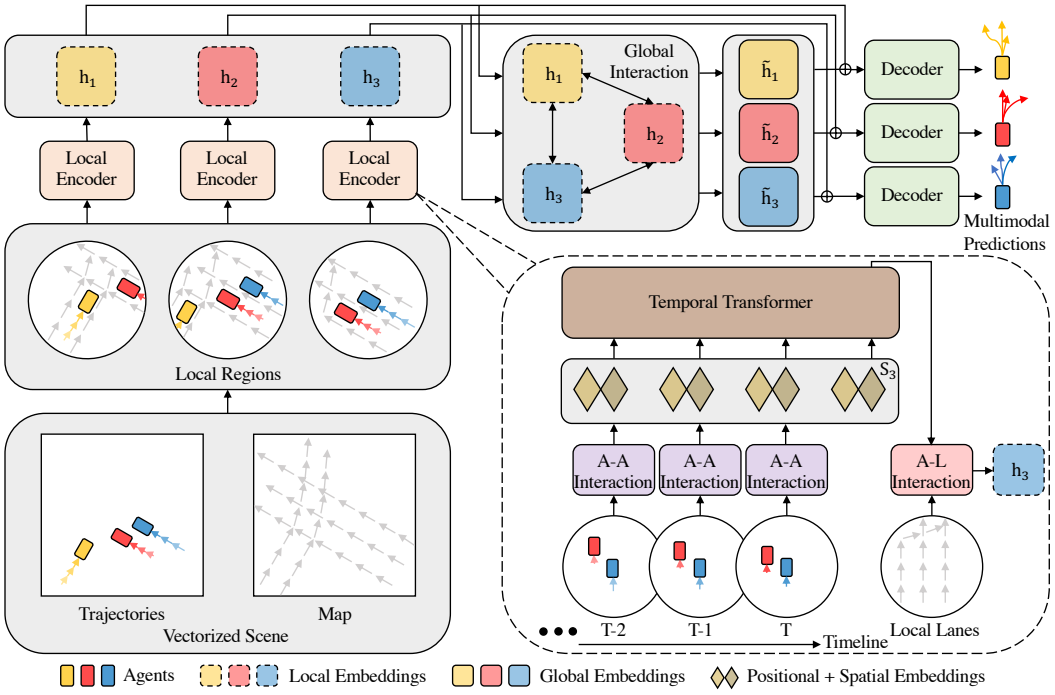

Figure 1. Overview of HiVT. A-A Interaction and A-L Interaction denote Agent-Agent and Agent-Lane Interaction respectively.

modeled agent. In the second stage, a global interaction module aggregates the local context of different agents and updates each agent's representation to capture long-range dependencies and scene-level dynamics. Finally, the hierarchically learned representations are used to make multimodal trajectory predictions for all agents simultaneously.

3.2. Scene Representation

A traffic scene consists of agents and map information. To represent the scene in a structured way, we extract vectorized entities from the scene, including the trajectory segments of traffic agents and the lane segments in map data. A vectorized entity is associated with semantic and geometric attributes. Compared with previous vectorized methods [17, 31, 49] in which the geometric attributes of agents or lanes involve the absolute positions of points, our representation avoids using any absolute positions and characterizes the geometric attributes with relative positions, which makes the scene a vector set entirely. Specifically, an agent i's trajectory is represented as $\{\mathbf{p}_i^t - \mathbf{p}_i^{t-1}\}_{t=1}^{T}$, where $\mathbf{p}_i^t \in \mathbb{R}^2$ is agent i's location at time step t and T is the total historical time steps. For a lane segment ξ, the geometric attribute is given by $\mathbf{p}_\xi^1 - \mathbf{p}_\xi^0$, where $\mathbf{p}_\xi^0 \in \mathbb{R}^2$ and $\mathbf{p}_\xi^1 \in \mathbb{R}^2$ are the starting and the ending coordinates of ξ. By transforming the point set into a vector set, such a representation guarantees translation invariance naturally. However, the information about the relative positions between entities is also discarded. To preserve spatial relationships, we introduce relative position vectors for agent-agent and agent-lane pairs. For instance, the position vector of agent j relative to agent i at time step t is $\mathbf{p}_j^t - \mathbf{p}_i^t$, which fully depicts the spatial relationship of the two agents and is also translation-invariant. Without any loss of information, our scene representation ensures that any learnable functions applied to it will necessarily respect translation invariance.

3.3. Hierarchical Vector Transformer

To accurately predict the future trajectories of traffic agents in a highly dynamic environment, the model needs to effectively learn the spatial-temporal relationships among a large number of vectorized entities. Transformers [46] have shown promise in capturing long-range dependencies between entities in a variety of tasks [5, 7, 14, 16, 33]. However, directly applying Transformers to the spatial-temporal entities suffers from the complexity of $O((NT + L)^2)$, where N, T, and L are the numbers of agents, historical time steps, and lane segments, respectively. To efficiently learn from a large number of entities, our model factorizes the space and the time dimensions and learns spatial relationships only *locally* at each time step. Specifically, we divide the space into N local regions, and each local region is centered at one agent in the scene. Within each local region are the trajectory segments and the local environment of the central agent, where the environmental information involves the trajectory segments of the neighboring agents and the local lane segments surrounding the

central agent. For each local region, we aggregate the local information into a single feature vector by sequentially modeling agent-agent interactions per time step, temporal dependencies per agent, and agent-lane interactions at the current time step. After aggregation, the feature vector contains rich information related to the central agent. On the other hand, the computational complexity is reduced from $O((NT+L)^2)$ to $O(NT^2+TN^2+NL)$ by the factorization of the space and the time dimensions and is further reduced to $O(NT^2+TNk+N\ell)$ by limiting the radius of the local regions, where $k<N$ and $\ell<L$.

While the local encoder can learn rich representations locally, the amount of information is limited by the range of the local regions. To avoid sacrificing prediction performance, we further employ a global interaction module to compensate for the restricted local receptive fields and capture scene-level dynamics, in which we perform message passing among local regions. The global interaction module can significantly enhance the expressiveness of the model at the cost of $O(N^2)$ complexity, which is relatively lightweight compared with the local encoder.

The multi-agent motion prediction problem exhibits translation and rotation symmetries. Existing methods [17, 31] re-normalize all vectorized entities w.r.t. each agent and make single-agent predictions multiple times to achieve invariance. This paradigm scales linearly w.r.t. the number of agents. By comparison, our model can make predictions for all agents in a single forward pass without sacrificing invariance via using the translation-invariant scene representation and rotation-invariant spatial learning modules. We illustrate the model components in more detail as follows.

3.3.1 Local Encoder

Agent-Agent Interaction. The agent-agent interaction module aims to learn the relationships between the central agent and the neighboring agents for each local region at each time step. To exploit the symmetries of the problem, we introduce a rotation-invariant cross-attention block to aggregate the spatial information. Specifically, we uniformly take the central agent i's latest trajectory segment $\mathbf{p}_i^T-\mathbf{p}_i^{T-1}$ as the reference vector of the local region and rotate all local vectors according to the reference vector's orientation $\boldsymbol{\theta}_i$. Based on the rotated vectors and their associated semantic attributes, we use Multi-Layer Perceptrons (MLPs) to obtain the central agent i's embedding $\mathbf{z}_i^t\in\mathbb{R}^{d_h}$ and any neighboring agent j's embedding $\mathbf{z}_{ij}^t\in\mathbb{R}^{d_h}$ at any time step t:

$$\mathbf{z}_i^t=\phi_{\text{center}}\left(\left[\mathbf{R}_i^\top\left(\mathbf{p}_i^t-\mathbf{p}_i^{t-1}\right),\mathbf{a}_i\right]\right), \tag{1}$$

$$\mathbf{z}_{ij}^t=\phi_{\text{nbr}}\left(\left[\mathbf{R}_i^\top\left(\mathbf{p}_j^t-\mathbf{p}_j^{t-1}\right),\mathbf{R}_i^\top\left(\mathbf{p}_j^t-\mathbf{p}_i^t\right),\mathbf{a}_j\right]\right), \tag{2}$$

where $\phi_{\text{center}}(\cdot)$ and $\phi_{\text{nbr}}(\cdot)$ are two different MLP blocks, $\mathbf{R}_i\in\mathbb{R}^{2\times2}$ is the rotation matrix parameterized by $\boldsymbol{\theta}_i$, $\mathbf{a}_i$ and $\mathbf{a}_j$ are the semantic attributes of agent i and agent j, respectively. Since all geometric attributes are normalized w.r.t. the central agent before they are fed into MLPs, these embeddings are unaffected by the rotation of the global coordinate frame. Apart from the trajectory segments, the inputs of $\phi_{\text{nbr}}(\cdot)$ also contain neighboring agents' position vectors relative to the central agent, making the neighboring embeddings spatially aware. The embedding of the central agent is then converted to the query vector, and the embeddings of the neighboring agents are used to calculate the key and the value vectors:

$$\mathbf{q}_i^t=\mathbf{W}^{Q^{\text{space}}}\mathbf{z}_i^t,\quad \mathbf{k}_{ij}^t=\mathbf{W}^{K^{\text{space}}}\mathbf{z}_{ij}^t,\quad \mathbf{v}_{ij}^t=\mathbf{W}^{V^{\text{space}}}\mathbf{z}_{ij}^t, \tag{3}$$

where $\mathbf{W}^{Q^{\text{space}}},\mathbf{W}^{K^{\text{space}}},\mathbf{W}^{V^{\text{space}}}\in\mathbb{R}^{d_k\times d_h}$ are learnable matrices for linear projection and d_k is the dimension of the transformed vectors. The resulting query, key, and value vectors are taken as inputs to the scaled dot-product attention block:

$$\boldsymbol{\alpha}_i^t=\text{softmax}\left(\frac{\mathbf{q}_i^{t^\top}}{\sqrt{d_k}}\cdot\left[\left\{\mathbf{k}_{ij}^t\right\}_{j\in\mathcal{N}_i}\right]\right), \tag{4}$$

$$\mathbf{m}_i^t=\sum_{j\in\mathcal{N}_i}\boldsymbol{\alpha}_{ij}^t\mathbf{v}_{ij}^t, \tag{5}$$

$$\mathbf{g}_i^t=\text{sigmoid}\left(\mathbf{W}^{\text{gate}}\left[\mathbf{z}_i^t,\mathbf{m}_i^t\right]\right), \tag{6}$$

$$\hat{\mathbf{z}}_i^t=\mathbf{g}_i^t\odot\mathbf{W}^{\text{self}}\mathbf{z}_i^t+\left(1-\mathbf{g}_i^t\right)\odot\mathbf{m}_i^t, \tag{7}$$

where $\mathcal{N}_i$ is the set of agent i's neighbors, $\mathbf{W}^{\text{gate}}$ and $\mathbf{W}^{\text{self}}$ are learnable matrices, and $\odot$ denotes element-wise product. Compared with the standard scaled dot-product attention [46], our variant uses a gating function to fuse the environmental features $\mathbf{m}_i^t$ with the central agent's features $\mathbf{z}_i^t$, enabling the block to have more control over the feature update. Like the original Transformer architecture, our attention block can also be extended to multiple heads. The outputs of the multi-head attention block are passed through an MLP block to obtain the spatial embedding $\mathbf{s}_i^t\in\mathbb{R}^{d_h}$ of agent i at time step t. In addition, we apply Layer Normalization [3] before each block and residual connections [24] after each block. In practice, this module can be implemented using efficient scatter and gather operations to parallelize the learning across all local regions and all time steps.

Temporal Dependency. To further capture the temporal information of each local region, we employ a temporal Transformer encoder on top of the agent-agent interaction module. For any central agent i, the input sequence of this module is composed of the embeddings $\{\mathbf{s}_i^t\}_{t=1}^T$ returned by the agent-agent interaction module at different time steps. Similar to BERT [14], we append an extra learnable token $\mathbf{s}^{T+1}\in\mathbb{R}^{d_h}$ to the end of the input sequence. Then, we add learnable positional embeddings to all tokens and stack the

tokens into a matrix $\mathbf{S}_i \in \mathbb{R}^{(T+1)\times d_h}$, which is fed into the temporal attention block:

$$\mathbf{Q}_i = \mathbf{S}_i \mathbf{W}^{Q^{\text{time}}}, \quad \mathbf{K}_i = \mathbf{S}_i \mathbf{W}^{K^{\text{time}}}, \quad \mathbf{V}_i = \mathbf{S}_i \mathbf{W}^{V^{\text{time}}}, \tag{8}$$

$$\hat{\mathbf{S}}_i = \text{softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V}_i, \tag{9}$$

$$\mathbf{M}_{uv} = \begin{cases} -\infty & \text{if } u < v\,; \\ 0 & \text{otherwise}\,, \end{cases} \tag{10}$$

where $\mathbf{W}^{Q^{\text{time}}}, \mathbf{W}^{K^{\text{time}}}, \mathbf{W}^{V^{\text{time}}} \in \mathbb{R}^{d_h \times d_k}$ are learnable matrices, and $\mathbf{M} \in \mathbb{R}^{(T+1)\times(T+1)}$ is the temporal mask that enforces the tokens only attend to the preceding time steps. The temporal learning module also consists of alternating multi-head attention blocks and MLP blocks. We input the updated extra tokens, which summarize the spatial-temporal features of the local regions, to the subsequent agent-lane interaction module.

Agent-Lane Interaction. The local map structure can indicate the future intent of the central agent. Thus, we incorporate the local map information into the embeddings. We first rotate the local lane segments and the agent-lane relative position vectors at the current time step T. The rotated vectors are then encoded by an MLP:

$$\mathbf{z}_{i\xi} = \phi_{\text{lane}}\left(\left[\mathbf{R}_i^\top\left(\mathbf{p}_\xi^1 - \mathbf{p}_\xi^0\right), \mathbf{R}_i^\top\left(\mathbf{p}_\xi^0 - \mathbf{p}_i^T\right), \mathbf{a}_\xi\right]\right), \tag{11}$$

where $\phi_{\text{lane}}(\cdot)$ is the MLP encoder for the lane segments, $\mathbf{R}_i \in \mathbb{R}^{2\times 2}$ is the rotation matrix of agent i's local region, $\mathbf{p}_\xi^0 \in \mathbb{R}^2$, $\mathbf{p}_\xi^1 \in \mathbb{R}^2$, and $\mathbf{a}_\xi$ are the starting location, the ending location, and the semantic attributes of lane segment ξ, respectively. With the central agent's spatial-temporal features as the query input and the MLP-encoded lane segment features as the key/value inputs, the agent-lane attention is calculated in the same way as described from Eq. (3) to Eq. (7). We further apply another MLP block to obtain the final local embedding $\mathbf{h}_i \in \mathbb{R}^{d_h}$ of the central agent i.

After sequentially modeling agent-agent interactions, temporal dependencies, and agent-lane interactions, the embeddings have fused rich information related to the central agents of the local regions.

3.3.2 Global Interaction Module

We introduce a global interaction module to capture long-range dependencies in the scene. Since local features are extracted in agent-centric coordinate frames, the global interaction module needs to bridge inter-frame geometric relationships when performing message passing among local regions. To this end, we extend the Transformer encoder to be aware of the differences between local coordinate frames. For example, the differences between agent i's and agent j's coordinate frames can be parameterized by $\mathbf{p}_j^T - \mathbf{p}_i^T$ and $\Delta\boldsymbol{\theta}_{ij}$, where $\Delta\boldsymbol{\theta}_{ij}$ denotes $\boldsymbol{\theta}_j - \boldsymbol{\theta}_i$. When performing message passing from agent j to agent i, we use an MLP $\phi_{\text{rel}}(\cdot)$ to obtain the pairwise embedding $\mathbf{e}_{ij}$:

$$\mathbf{e}_{ij} = \phi_{\text{rel}}\left(\left[\mathbf{R}_i^\top\left(\mathbf{p}_j^T - \mathbf{p}_i^T\right), \cos\left(\Delta\boldsymbol{\theta}_{ij}\right), \sin\left(\Delta\boldsymbol{\theta}_{ij}\right)\right]\right). \tag{12}$$

The pairwise embedding is then incorporated into the transformation of the vectors,

$$\begin{aligned} \tilde{\mathbf{q}}_i &= \mathbf{W}^{Q^{\text{global}}}\mathbf{h}_i, \\ \tilde{\mathbf{k}}_{ij} &= \mathbf{W}^{K^{\text{global}}}\left[\mathbf{h}_j, \mathbf{e}_{ij}\right], \\ \tilde{\mathbf{v}}_{ij} &= \mathbf{W}^{V^{\text{global}}}\left[\mathbf{h}_j, \mathbf{e}_{ij}\right], \end{aligned} \tag{13}$$

where $\mathbf{h}_i$ and $\mathbf{h}_j$ are the local embeddings of agent i and agent j, and $\mathbf{W}^{Q^{\text{global}}}, \mathbf{W}^{K^{\text{global}}}, \mathbf{W}^{V^{\text{global}}}$ are learnable matrices. To capture pairwise interactions globally, we apply the same spatial attention block as in the local encoder, which is followed by an MLP block that outputs the global representation $\tilde{\mathbf{h}}_i$ for any agent i.

3.3.3 Multimodal Future Decoder

The future motions of traffic agents are inherently multimodal. Thus, we parameterize the distribution of future trajectories as a mixture model where each mixture component is a Laplace distribution. The predictions are made for all agents in a single shot. For each agent i and each component f, an MLP receives the local and the global representations as inputs and outputs the location $\boldsymbol{\mu}_{i,f}^t \in \mathbb{R}^2$ and its associated uncertainty $\mathbf{b}_{i,f}^t \in \mathbb{R}^2$ of the agent per future time step in the local coordinate frame. The output tensor of the regression head has the shape of $[F, N, H, 4]$, where F is the number of mixture components, N is the number of agents in the scene, and H is the number of predicted future time steps. We also use another MLP followed by a softmax function to produce the mixing coefficients of the mixture model for each agent, which have the shape of $[N, F]$.

3.4. Training

We employ the variety loss [23, 45] to encourage the diversity of multiple trajectory hypotheses, which optimizes only the best of the F predictions during training. Before optimization, we first calculate the errors between the ground-truth locations and the locations of the F mixture components predicted by the model for each agent and each time step. Then, we sum up the errors across all future time steps to obtain a matrix of shape $[F, N]$, according to which we select the trajectory with minimum error for each agent, *i.e.*, finding the minimum value of each column in the error matrix. The final loss function comprises the regression loss $\mathcal{L}_{\text{reg}}$ and the classification loss $\mathcal{L}_{\text{cls}}$ with equal weights:

$$\mathcal{L} = \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{cls}}. \tag{14}$$

We adopt the negative log-likelihood as the regression loss:

$$\mathcal{L}_{\text{reg}} = -\frac{1}{NH}\sum_{i=1}^{N}\sum_{t=T+1}^{T+H}\log \mathrm{P}\left(\mathbf{R}_i^{\top}\left(\mathbf{p}_i^t - \mathbf{p}_i^T\right) \mid \hat{\boldsymbol{\mu}}_i^t, \hat{\mathbf{b}}_i^t\right), \tag{15}$$

where $\mathrm{P}(\cdot \mid \cdot)$ is the probability density function of Laplace distribution, and $\{\hat{\boldsymbol{\mu}}_i^t\}_{t=T+1}^{T+H}, \{\hat{\mathbf{b}}_i^t\}_{t=T+1}^{T+H}$ are the locations and the uncertainties of the best-predicted trajectory for agent i. We use the cross-entropy loss as the classification loss to optimize the mixing coefficients.

4. Experiments

4.1. Experimental Setup

Dataset. We evaluate our prediction framework on the large-scale Argoverse motion forecasting dataset [10], which provides the trajectories of agents and the high-definition map data. The dataset contains 323557 real-world driving scenarios and is split into training, validation, and test sets, with 205942, 39472, and 78143 samples. All training and validation scenarios are 5-second sequences sampled at 10 Hz, while only the first 2-second trajectories are publicly available in the test set. Given the initial 2-second observations, the Argoverse Motion Forecasting Challenge requires predicting the 3-second future movements of the agents.

Metrics. We evaluate our model on the standard metrics for motion prediction, including minimum Average Displacement Error (minADE), minimum Final Displacement Error (minFDE), and Miss Rate (MR). These metrics allow models to forecast up to 6 trajectories for each agent. The metric minADE measures the ℓ_2 distance in meters between the best-predicted trajectory and the ground-truth trajectory averaged over all future time steps, while minFDE measures the error at the final future time step. The best-predicted trajectory is defined as the one that has the minimum endpoint error. MR refers to the ratio of scenarios where the distance between the ground-truth endpoint and the best-predicted endpoint is above 2.0 meters.

Implementation Details. We train our model for 64 epochs on an RTX 2080 Ti GPU using AdamW optimizer [35], with the batch size, initial learning rate, weight decay, and dropout rate set to 32, 3×10^{-4}, 1×10^{-4}, and 0.1, respectively. The learning rate is decayed using the cosine annealing scheduler [34]. Our model consists of 1 layer of agent-agent and agent-lane interaction module, 4 layers of temporal learning module, and 3 layers of global interaction module. The number of heads in all multi-head attention blocks is 8. The radius of all local regions is 50 meters. We follow the convention in the baselines and set the number of the predicted modes F to 6. We do not use tricks such as ensemble methods and data augmentation. We conduct experiments based on a small model with 64 hidden units and a large model with 128 hidden units, termed as HiVT-64 and HiVT-128, respectively.

A-A	Temporal	A-L	Global	minADE	minFDE	MR
	✓	✓	✓	0.71	1.07	0.11
✓		✓	✓	1.00	1.56	0.21
✓	✓		✓	0.77	1.25	0.14
✓	✓	✓		0.73	1.13	0.12
✓	✓	✓	✓	**0.69**	**1.04**	**0.10**

Table 1. Ablation studies on the components of our framework.

Gated Update	Temporal Mask	minADE	minFDE	MR
		0.70	1.07	0.11
	✓	0.70	1.05	0.11
✓		0.70	1.05	0.11
✓	✓	**0.69**	**1.04**	**0.10**

Table 2. Ablation studies on the attention blocks.

4.2. Ablation Studies

We conduct ablation studies on the Argoverse validation set. Unless specified, experimental results are based on our 64-dimension model HiVT-64.

Importance of Each Module. We demonstrate each module's contribution to the prediction performance by alternately removing one of the components. As shown in Tab. 1, each component can improve the performance to a certain degree. First, without the agent-agent interaction module, the model cannot capture fine-grained local interactions at previous time steps and suffers from performance drop. We also note that adding more layers of this module can further improve the performance, but we keep using one layer for higher efficiency. Second, the temporal learning module has the most significant impact on the performance since inferring the future motions of traffic agents in highly dynamic traffic scenarios relies heavily on the historical information. Third, the lane information plays a crucial role in motion predictions since traffic agents usually move along the lanes due to the constraint of traffic rules. Moreover, the global interaction module can noticeably improve the prediction performance. This result validates its capacity for capturing long-range dependencies.

Ablation Studies on the Attention Blocks. We evaluate the effect of the gated update function in the spatial attention block and the temporal mask in the temporal attention block. As shown in Tab. 2, using the gating function can improve the prediction performance, presumably because some agents do not interact much with the environment. The results in Tab. 2 also show that removing the temporal mask in the temporal attention block can lead to worse performance, which suggests that preventing tokens from attending to subsequent time steps is beneficial to the model.

Model	Representation	Spatial Module	minADE	minFDE	MR
HiVT-64	Point Set	w/o Rotation	1.09	1.89	0.26
HiVT-64	Vector Set	w/o Rotation	0.73	1.13	0.12
HiVT-64	Vector Set	w/ Rotation	0.69	1.04	0.10
HiVT-128	Vector Set	w/o Rotation	0.69	1.04	0.10
HiVT-128	Vector Set	w/ Rotation	**0.66**	**0.96**	**0.09**

Table 3. Impact of translation and rotation invariance on the prediction performance.

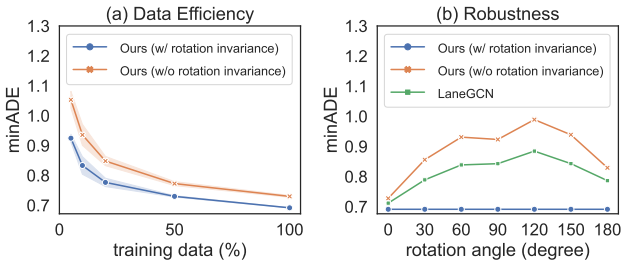

Figure 2. Data efficiency and robustness of models.

Importance of Translation and Rotation Invariance. We measure the importance of translation and rotation invariance quantitatively. In the ablation studies, we first represent the scene as a point set and normalize the coordinates according to the autonomous vehicle's position and heading at the current time step. This representation is scene-centric and is *not* invariant to the translation of the scene. We further replace the rotation-invariant cross-attention blocks in the spatial learning modules with their non-invariant counterparts. As shown in the first row of Tab. 3, learning under a scene-centric representation cannot produce reliable prediction results. The results in Tab. 3 also show that employing our proposed translation-invariant representation and rotation-invariant spatial learning modules can notably improve the prediction performance, which suggests that the geometric priors can ease the learning difficulty of models. Interestingly, although the model size of HiVT-128 is nearly three times larger than HiVT-64 (see Tab. 4), without the inductive bias of rotation invariance, HiVT-128 cannot outperform HiVT-64. This phenomenon shows the importance of symmetries to parameter efficiency.

We further demonstrate the data efficiency and the robustness that benefited from rotation invariance. We train HiVT-64 with and without the inductive bias of rotation invariance using 5%, 10%, 20%, 50%, and 100% of the training data and evaluate the performance on the validation set. As shown in Fig. 2 (a), the rotation-invariant model requires fewer training data to achieve similar performance to its non-invariant counterpart. In Fig. 2 (b), we show that our proposed spatial learning module is robust to the rotation of the scene, while the prediction performance of its counterpart is severely affected by the rotation angle.

Method	minADE	minFDE	MR	#Param
HiVT-64	0.8067	1.2433	0.1396	**662K**
HiVT-128	**0.7735**	**1.1693**	0.1267	2,529K
LaneGCN [31]	0.8679	1.3640	0.1634	3,701K
Scene Transformer [38]	0.8026	1.2321	0.1255	15,296K
DenseTNT [22]	0.8817	1.2815	0.1258	1,103K
MultiModalTransformer [28]	0.8372	1.2905	0.1429	6,328K
mmTransformer [32]	0.8436	1.3383	0.1540	2,607K
HOME+GOME [18, 19]	0.8904	1.2919	**0.0846**	5,100K
TPCN [49]	0.8153	1.2442	0.1333	-

Table 4. Quantitative results on the Argoverse Motion Forecasting Leaderboard [1]. The model sizes are reported by the authors or are calculated using the official implementations. Symbol "-" means model size unknown because the authors have not disclosed it and no open-source implementation is available.

When only one agent in the scene needs to be predicted, existing vectorized methods [17, 31, 49] achieve translation and rotation invariance by normalizing all entities in the scene w.r.t. the agent to be predicted. However, in the setting of multi-agent prediction, these methods cannot accurately predict the motions of multiple agents in a single forward pass since only one agent in the scene is normalized each time. To verify this, we evaluate the robustness of LaneGCN [31] to the rotation of the scene. Like many other competitive methods on the Argoverse dataset, LaneGCN normalizes the scene to be centered at the target agent and to be aligned with the target agent's heading. The results in Fig. 2 (b) show that the prediction performance of LaneGCN is sensitive to the rotation angle of the scene, indicating that it is unable to make accurate predictions for agents with arbitrary headings. As a result, making predictions for multiple agents requires re-normalizing the scene and performing forward passes multiple times to achieve invariance. By contrast, HiVT models all agents symmetrically and makes predictions for all of them in a single shot without sacrificing invariance, translating to fast and accurate multi-agent prediction.

4.3. Results

Comparison with State-of-the-art. We compare our method with the state-of-the-art models on the Argoverse test set. The results in Tab. 4 were collected from the Argoverse leaderboard [1] on 16/11/2021. HOME+GOHOME [18, 19] is the only rasterized method in the table, and it uses much more parameters than most vectorized methods but does not perform well on the metrics except MR. Using 82.1%, 74.6%, and 40.0% fewer parameters, HiVT-64 significantly outperforms LaneGCN [31], mmTransformer [32], and DenseTNT [22] on the metrics of minADE and minFDE. Compared with Scene Transformer [38] and MultiModalTransformer [28], which are two Transformer-based motion prediction models proposed

Model	Speed (ms)	minADE	minFDE	MR
LaneGCN [31]	173	0.71	1.08	0.10
DenseTNT (w/ 100ms opt.) [22]	2644	0.73	1.05	0.10
DenseTNT (w/ goal set pred.) [22]	531	0.75	1.05	0.10
HiVT-64 (local only, r=20)	38	0.74	1.17	0.13
HiVT-64 (local only, r=50)	52	0.73	1.13	0.12
HiVT-64 (local+global, r=20)	39	0.70	1.08	0.11
HiVT-64 (local+global, r=50)	53	0.69	1.04	0.10
HiVT-64 (local+global, r=80)	68	0.69	1.04	0.10
HiVT-128 (local+global, r=50)	69	**0.66**	**0.96**	**0.09**

Table 5. The inference speed and the prediction performance of models on the Argoverse validation set. Symbol "r" denotes the radius of the local regions in meters of our models.

recently, HiVT-64 uses 95.7% and 89.5% fewer parameters but still achieves on par or better performance. With 83.5% and 60.0% fewer parameters than Scene Transformer and MultiModalTransformer, HiVT-128 outperforms all methods shown in Tab. 4 in terms of minADE and minFDE. The above results show the superior prediction performance and parameter efficiency of our approach. Our method ranked 1^{st} in terms of minADE on 16/11/2021 and remains competitive ranking on the Argoverse leaderboard.

Inference Speed. We compare the inference speed of models on the Argoverse validation set using an RTX 2080 Ti GPU and a batch size of 32. Such a batch size is close to the average number of agents per scene. As shown in Tab. 5, all variants of our model have faster inference speed than the baselines, and the prediction accuracy of the full model surpasses the baselines when the radius of the local regions is no less than 20 meters. Although in Tab. 5 we assume that multiple forward passes are needed for multi-agent predictions and show the inference speed when the batch size is 32, our approach can actually make accurate predictions for all agents using a single forward pass due to the symmetric designs. When the batch size is one and the radius is 50 meters, the average inference speed of HiVT-128 is around 20 ms, which satisfies the real-time requirement.

From Tab. 5 we can see that adding the global interaction module introduces negligible computational cost but substantially improves the prediction performance. This result validates the effectiveness of the global interaction module. We also vary the radius of the local regions to obtain models with different computational complexity. Table 5 shows that reducing the radius can speed up the inference of the overall model, and using a much larger radius of 80 meters suffers from slower inference but does not help the performance. Our local-global architecture allows the practitioners to choose the appropriate size of local regions based on the requirement for prediction accuracy and the constraint of computing resources.

Qualitative Results. We present the qualitative results of HiVT-128 on the Argoverse validation set. For clarity, we visualize only two agents per scene. As shown in Fig. 3, our model can make accurate, multimodal, and reasonable predictions for multiple agents simultaneously in complex traffic scenarios. Interestingly, although the dataset does not contain information about the traffic light state, the upper-left example shows that our model successfully predicts the sudden acceleration of vehicles at an intersection.

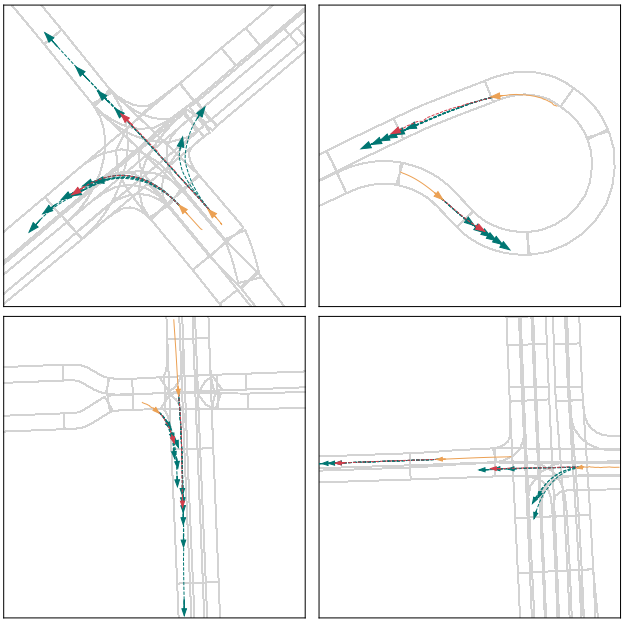

Figure 3. Qualitative results of HiVT-128. The past trajectories are shown in yellow, the ground-truth trajectories are shown in red, and the predicted trajectories are shown in green.

5. Conclusion

In this paper, we propose a novel framework for multi-agent motion prediction, which hierarchically models the interactions between vectorized entities through local context extraction and global interaction modeling. Built upon this framework, we present a translation-invariant scene representation and a rotation-invariant transformer architecture for learning. The symmetric designs can noticeably improve the model's prediction accuracy, data efficiency, and parameter efficiency. Experiments show that our approach achieves the state-of-the-art performance on the Argoverse motion forecasting benchmark with much fewer model parameters and faster inference speed than existing solutions.

Acknowledgement

This work was partially supported by Hong Kong Research Grant Councilunder NSFC/RGC NCityU 140/20, Science and Technology Innovation Committee Foundation of Shenzhen under Grant No. JCYJ20200109143223052.

References

[1] Argoverse motion forecasting competition. https://eval.ai/web/challenges/challenge-page/454/overview. Accessed: 2021-11-16. 7

[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4

[4] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Robotics: Science and Systems (RSS)*, 2019. 2

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2, 3

[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. 1

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2, 3

[8] Sergio Casas, Cole Gulino, Renjie Liao, and Raquel Urtasun. Spagnn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020. 2

[9] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Conference on Robot Learning (CoRL)*, 2019. 1, 2

[10] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 6

[11] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[12] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019. 1, 2

[13] Nachiket Deo and Mohan M Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018. 2

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2, 3, 4

[15] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, Nitin Singh, and Jeff Schneider. 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)*, 2020. 2

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 2, 3

[17] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 3, 4, 7

[18] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Gohome: Graph-oriented heatmap output forfuture motion estimation. *arXiv preprint arXiv:2109.01827*, 2021. 7

[19] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Home: Heatmap output for future motion estimation. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2021. 2, 7

[20] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017. 1

[21] Francesco Giuliari, Irtiza Hasan, Marco Cristani, and Fabio Galasso. Transformer networks for trajectory forecasting. In *25th International Conference on Pattern Recognition (ICPR)*, 2020. 2

[22] Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2, 7, 8

[23] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 5

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 4

[25] Joey Hong, Benjamin Sapp, and James Philbin. 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)*, 2019. 1, 2

[26] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[27] Yingfan Huang, HuiKun Bi, Zhaoxin Li, Tianlu Mao, and Zhaoqi Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 2

[28] Zhiyu Huang, Xiaoyu Mo, and Chen Lv. Multi-modal motion prediction with transformer-based neural network for autonomous driving. *arXiv preprint arXiv:2109.06446*, 2021. 7

[29] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 1

[30] Lingyun Luke Li, Bin Yang, Ming Liang, Wenyuan Zeng, Mengye Ren, Sean Segal, and Raquel Urtasun. End-to-end contextual perception and prediction with interaction transformer. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. 2

[31] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 3, 4, 7, 8

[32] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal motion prediction with stacked transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 7

[33] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

[34] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 6

[35] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 6

[36] Jean Mercat, Thomas Gilles, Nicole El Zoghby, Guillaume Sandou, Dominique Beauvois, and Guillermo Pita Gil. Multi-head attention for multi-modal joint vehicle motion forecasting. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020. 1, 2

[37] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[38] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene transformer: A unified architecture for predicting multiple agent trajectories. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 1, 2, 7

[39] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[40] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 1

[41] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2

[42] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 2

[43] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[44] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019. 2

[45] Luca Anthony Thiede and Pratik Prabhanjan Brahma. Analyzing the variety loss in the context of probabilistic trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 5

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 1, 2, 3, 4

[47] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019. 1

[48] Luyao Ye, Zezhong Wang, Xinhong Chen, Jianping Wang, Kui Wu, and Kejie Lu. Gsan: Graph self-attention network for learning spatial-temporal interaction representation in autonomous driving. *IEEE Internet of Things Journal*, 2021. 2

[49] Maosheng Ye, Tongyi Cao, and Qifeng Chen. Tpcn: Temporal point cloud networks for motion forecasting. In *Proceed-*

ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021. 1, 2, 3, 7

[50] Cunjun Yu, Xiao Ma, Jiawei Ren, Haiyu Zhao, and Shuai Yi. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[51] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1, 2

[52] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. In *Conference on Robot Learning (CoRL)*, 2020. 2