Motion Transformer with Global Intention Localization and Local Movement Refinement

Shaoshuai Shi, Li Jiang, Dengxin Dai, Bernt Schiele
Max Planck Institute for Informatics, Saarland Informatics Campus
{sshi, lijiang, ddai, schiele}@mpi-inf.mpg.de

Abstract

Predicting multimodal future behavior of traffic participants is essential for robotic vehicles to make safe decisions. Existing works explore to directly predict future trajectories based on latent features or utilize dense goal candidates to identify agent's destinations, where the former strategy converges slowly since all motion modes are derived from the same feature while the latter strategy has efficiency issue since its performance highly relies on the density of goal candidates. In this paper, we propose the Motion TRansformer (MTR) framework that models motion prediction as the joint optimization of global intention localization and local movement refinement. Instead of using goal candidates, MTR incorporates spatial intention priors by adopting a small set of learnable motion query pairs. Each motion query pair takes charge of trajectory prediction and refinement for a specific motion mode, which stabilizes the training process and facilitates better multimodal predictions. Experiments show that MTR achieves state-of-the-art performance on both the marginal and joint motion prediction challenges, ranking 1st on the leaderboards of Waymo Open Motion Dataset. Code will be available at https://github.com/sshaoshuai/MTR.

1 Introduction

Motion forecasting is a fundamental task of modern autonomous driving systems. It has been receiving increasing attention in recent years [21, 48, 31, 61, 37] as it is crucial for robotic vehicles to understand driving scenes and make safe decisions. Motion forecasting requires to predict future behaviors of traffic participants by jointly considering the observed agent states and road maps, which is challenging due to inherently multimodal behaviors of the agent and complex scene environments.

To cover all potential future behaviors of the agent, existing approaches mainly fall into two different lines: the goal-based methods and the direct-regression methods. The goal-based methods [21, 65] adopt dense goal candidates to cover all possible destinations of the agent, predicting the probability of each candidate being a real destination and then completing the full trajectory for each selected candidate. Although these goal candidates alleviate the burden of model optimization by reducing trajectory uncertainty, their density largely affects the performance of these methods: fewer candidates will decrease the performance while more candidates will greatly increase computation and memory cost. Instead of using goal candidates, the direct-regression methods [37, 49] directly predict a set of trajectories based on the encoded agent feature, covering the agent's future behavior adaptively. Despite the flexibility in predicting a broad range of agent behaviors, they generally converge slowly as various motion modes are required to be regressed from the same agent feature without utilizing any spatial priors. They also tend to predict the most frequent modes of training data since these frequent modes dominate the optimization of the agent feature. In this paper, we present a unified framework, namely **Motion TRansformer (MTR)**, which takes the best of both types of methods.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

In our proposed MTR, we adopt a small set of novel motion query pairs to model motion prediction as the joint optimization of two tasks: The first global intention localization task aims to roughly identify agent's intention for achieving higher efficiency, while the second local movement refinement task aims to adaptively refine each intention's predicted trajectory for achieving better accuracy. Our approach not only stabilizes the training process without depending on dense goal candidates but also enables flexible and adaptive prediction by enabling local refinement for each motion mode.

Specifically, each motion query pair consists of two components, *i.e.*, a static intention query and a dynamic searching query. The static intention queries are introduced for global intention localization, where we formulate them based on a small set of spatially distributed intention points. Each static intention query is the learnable positional embedding of an intention point for generating trajectory of a specific motion mode, which not only stabilizes the training process by explicitly utilizing different queries for different modes, but also eliminates the dependency on dense goal candidates by requiring each query to take charge of a large region. The dynamic searching queries are utilized for local movement refinement, where they are also initialized as the learnable embeddings of the intention points but are responsible for retrieving fine-grained local features around each intention point. For this purpose, the dynamic searching queries are dynamically updated according to the predicted trajectories, which can adaptively gather latest trajectory features from a deformable local region for iterative motion refinement. These two queries complement each other and have been empirically demonstrated their great effectiveness in predicting multimodal future motion. Besides that, we also propose a dense future prediction module. Existing works generally focus on modeling the agent interaction over past trajectories while ignoring the future trajectories' interaction. To compensate for such information, we adopt a simple auxiliary regression head to densely predict future trajectory and velocity for each agent, which are encoded as additional future context features to benefit future motion prediction of our interested agent. The experiments show that this simple auxiliary task works well and remarkably improves the performance of multimodal motion prediction.

Our contributions are three-fold: (1) We propose a novel motion decoder network with a new concept of motion query pair, which adopts two types of queries to model motion prediction as joint optimization of global intention localization and local movement refinement. It not only stabilizes the training with mode-specific motion query pairs, but also enables adaptive motion refinement by iteratively gathering fine-grained trajectory features. (2) We present an auxiliary dense future prediction task to enable the future interactions between our interested agent and other agents. It facilitates our framework to predict more scene-compliant trajectories for the interacting agents. (3) By adopting these techniques, we propose MTR framework that explores transformer encoder-decoder structure for multimodal motion prediction. Our approach achieves state-of-the-art performance on both the marginal and joint motion prediction benchmarks of Waymo Open Motion Dataset (WOMD) [15], outperforming previous best ensemble-free approaches with +8.48% mAP gains for marginal motion prediction and $+7.98\%$ mAP gains for joint motion prediction. As of 19 May 2022, our approach ranks 1st on both the marginal and joint motion prediction leaderboards of WOMD.

2 Related Work

Motion Prediction for Autonomous Driving. Recently, motion prediction has been extensively studied due to the growing interest in autonomous driving, and it typically takes road map and agent history states as input. To encode such scene context, early works [38, 33, 6, 13, 64, 4, 9] typically rasterize them into an image so as to be processed with convolutional neural networks (CNNs). LaneGCN [29] builds a lane graph toscalability capture map topology. VectorNet [17] is widely adopted by recent works [21, 45, 37, 49] due to its efficiency and scalability, where both road maps and agent trajectories are represented as polylines. We also adopt this vector representation, but instead of building global graph of polylines, we propose to adopt transformer encoder on local connected graph, which not only better maintains input locality structure but also is more memory-efficient to enable larger map encoding for long-term motion prediction.

Given the encoded scene context features, existing works explore various strategies to model multi-modal future motion. Early works [2, 22, 41, 46, 42] propose to generate a set of trajectory samples to approximate the output distribution. Some other works [10, 23, 35, 39, 43] parameterize multi-modal predictions with Gaussian Mixture Models (GMMs) to generate compact distribution. HOME series [19, 18] generate trajectories with sampling on a predicted heatmap. IntentNet [8] considers intention prediction as a classification with 8 high level actions, while [31] proposes a region-based

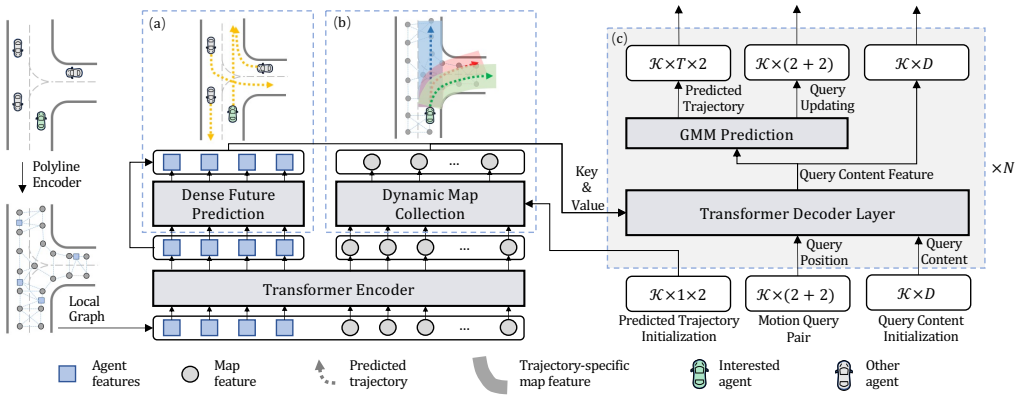


Figure 1: The architecture of MTR framework. (a) indicates the dense future prediction module, which predicts a single trajectory for each agent (*e.g.*, drawn as yellow dashed curves in the above of (a)). (b) indicates the dynamic map collection module, which collects map elements along each predicted trajectory (*e.g.*, drawn as the shadow region along each trajectory in the above part of (b)) to provide trajectory-specific feature for motion decoder network. (c) indicates the motion decoder network, where $\mathcal{K}$ is the number of motion query pairs, T is the number of future frames, D is hidden feature dimension and N is the number of transformer decoder layers. The predicted trajectories, motion query pairs, and query content features are the outputs from last decoder layer and will be taken as input to next decoder layer. For the first decoder layer, both two components of motion query pair are initialized as predefined intention points, the predicted trajectories are replaced with the intention points for initial map collection, and query content features are initialized as zeros.

training strategy. Goal-based methods [65, 42, 16, 32] are another kinds of models where they first estimate several goal points of the agents and then complete full trajectory for each goal.

Recently, the large-scale Waymo Open Motion Dataset (WOMD) [15] is proposed for long-term motion prediction. To address this challenge, DenseTNT [21] adopts a goal-based strategy to classify endpoint of trajectory from dense goal points. Other works directly predict the future trajectories based on the encoded agent features [37] or latent anchor embedding [49]. However, the goal-based strategy has the efficiency concern due to a large number of goal candidates, while the direct-regression strategy converges slowly as the predictions of various motion modes are regressed from the same agent feature. In contrast, our approach adopts a small set of learnable motion query pairs, which not only eliminate the large number of goal candidates but also alleviate the optimization burden by utilizing mode-specific motion query pairs for predicting different motion modes.

Some very recent works [47, 25, 24] also achieve top performance on WOMD by exploring Mix-and-Match block [47], a variant of MultiPath++ [25] or heterogeneous graph [24]. However, they generally focus on exploring various structures for encoding scene context, while how to design a better motion decoder for multimodal motion prediction is still underexplored. In contrast, our approach focuses on addressing this challenge with a novel transformer-based motion decoder network.

Transformer. Transformer [50] has been widely applied in natural language processing [12, 3] and computer vision [14, 52, 5, 51, 62]. Our approach is inspired by DETR [5] and its follow-up works [67, 34, 60, 27, 30, 11, 63], especially DAB-DETR [30], where the object query is considered as the positional embedding of a spatial anchor box. Motivated by their great success in object detection, we introduce a novel concept of motion query pair to model multimodal motion prediction with prior intention points, where each motion query pair takes charge of predicting a specific motion mode and also enables iterative motion refinement by combining with transformer decoders.

3 Motion TRansformer (MTR)

We propose Motion TRansformer (MTR), which adopts a novel transformer encoder-decoder structure with iterative motion refinement for predicting multimodal future motion. The overall structure is illustrated in Figure 1. In Sec. 3.1, we introduce our encoder network for scene context modeling. In Sec. 3.2, we present motion decoder network with a novel concept of motion query pair for predicting multimodal trajectories. Finally, in Sec. 3.3, we introduce the optimization process of our framework.

3.1 Transformer Encoder for Scene Context Modeling

The future behaviors of the agents highly depend on the agents' interaction and road map. To encode such scene context, existing approaches have explored various strategies by building global interacting graph [17, 21] or summarizing map features to agent-wise features [37, 49]. We argue that the locality structure is important for encoding scene context, especially for the road map. Hence, we propose a transformer encoder network with local self-attention to better maintain such structure information.

Input representation. We follow the vectorized representation [17] to organize both input trajectories and road map as polylines. For the motion prediction of a interested agent, we adopt the agent-centric strategy [65, 21, 49] that normalizes all inputs to the coordinate system centered at this agent. Then, a simple polyline encoder is adopted to encode each polyline as an input token feature for the transformer encoder. Specifically, we denote the history state of N_a agents as $A_{\text{in}} \in \mathbb{N}^{N_a \times t \times C_a}$, where t is the number of history frames, C_a is the number of state information (*e.g.*, location, heading angle and velocity), and we pad zeros at the positions of missing frames for trajectories that have less than t frames. The road map is denoted as $M_{\text{in}} \in \mathbb{R}^{N_m \times n \times C_m}$, where N_m is the number of map polylines, n is the number of points in each polyline and C_m is the number of attributes of each point (*e.g.*, location and road type). Both of them are encoded by a PointNet-like [40] polyline encoder as:

$$A_{\text{p}} = \phi\left(\text{MLP}(A_{\text{in}})\right), \quad M_{\text{p}} = \phi\left(\text{MLP}(M_{\text{in}})\right), \tag{1}$$

where $\text{MLP}(\cdot)$ is a multilayer perceptron network, and ϕ is max-pooling to summarize each polyline features as agent features $A_{\text{p}} \in \mathbb{R}^{N_a \times D}$ and map features $M_{\text{p}} \in \mathbb{R}^{N_m \times D}$ with feature dimension D.

Scene context encoding with local transformer encoder. The local structure of scene context is important for motion prediction. For example, the relation of two parallel lanes is important for modelling the motion of changing lanes, but adopting attention on global connected graph equally considers relation of all lanes. In contrast, we introduce such prior knowledge to context encoder by adopting local attention, which better maintains the locality structure and are more memory-efficient. Specifically, the attention module of j-th transformer encoder layer can be formulated as:

$$G^j = \text{MultiHeadAttn}\left(\text{query}{=}G^{j-1} + \text{PE}_{G^{j-1}},\ \text{key}{=}\kappa(G^{j-1}) + \text{PE}_{\kappa(G^{j-1})},\ \text{value}{=}\kappa(G^{j-1})\right), \tag{2}$$

where $\text{MultiHeadAttn}(\cdot,\cdot,\cdot)$ is the multi-head attention layer [50], $G^0 = [A_{\text{p}}, M_{\text{p}}] \in \mathbb{N}^{(N_a+N_m)\times D}$ concatenating the features of agents and map, and $\kappa(\cdot)$ denotes k-nearest neighbor algorithm to find k closest polylines for each query polyline. PE denotes sinusoidal position encoding of input tokens, where we utilize the latest position for each agent and utilize polyline center for each map polyline. Thanks to such local self-attention, our framework can encode a much larger area of scene context.

The encoder network finally generates both agent features $A_{\text{past}} \in \mathbb{R}^{N_a \times D}$ and map features $M \in \mathbb{R}^{N_m \times D}$, which are considered as the scene context inputs of the following decoder network.

Dense future prediction for future interactions. Interactions with other agents heavily affect behaviors of our interested agent, and previous works propose to model the multi-agent interactions with hub-host based network [68], dynamic relational reasoning [28], social spatial-temporal network [59], etc. However, most existing works generally focus on learning such interactions over past trajectories while ignoring the interactions of future trajectories. Therefore, considering that the encoded features A have already learned rich context information of all agents, we propose to densely predict both future trajectories and velocities of all agents by adopting a simple regression head on A:

$$S_{1:T} = \text{MLP}(A_{\text{past}}), \tag{3}$$

where $S_i \in \mathbb{R}^{N_a \times 4}$ includes future position and velocity of each agent at time step i, and T is the number of future frames to be predicted. The predicted trajectories $S_{1:T}$ are encoded by adopting the same polyline encoder as Eq. (1) to encode the agents' future states as features $A_{\text{future}} \in \mathbb{R}^{N_a \times D}$, which are then utilized to enhance the above features A by using a feature concatenation and three MLP layers as $A = \text{MLP}([A_{\text{past}}, A_{\text{future}}])$. This auxiliary task provides additional future context information to the decoder network, facilitating the model to predict more scene-compliant future trajectories for the interested agent. The experiments in Table 3 demonstrates that this simple and light-weight auxiliary task can effectively improve the performance of multimodal motion prediction.

3.2 Transformer Decoder with Motion Query Pair

Given the scene context features, a transformer-based motion decoder network is adopted for multi-modal motion prediction, where we propose *motion query pair* to model motion prediction as the

joint optimization of global intention localization and local movement refinement. Each motion query pair contains two types of queries, *i.e.*, static intention query and dynamic searching query, for conducting global intention localization and local movement refinement respectively. As shown in Figure 2, our motion decoder network contains stacked transformer decoder layers for iteratively refining the predicted trajectories with motion query pairs. Next, we illustrate the detailed structure.

Global intention localization aims to localize agent's potential motion intentions in an efficient and effective manner. We propose *static intention query* to narrow down the uncertainty of future trajectory by utilizing different intention queries for different motion modes. Specifically, we generate $\mathcal{K}$ representative intention points $I \in \mathbb{R}^{\mathcal{K}\times 2}$ by adopting k-means clustering algorithm on the endpoints of ground-truth (GT) trajectories, where each intention point represents an implicit motion mode that considers both motion direction and velocity. We model each static intention query as the learnable positional embedding of the intention point as:

$$Q_I = \text{MLP}\left(\text{PE}(I)\right), \tag{4}$$

where $\text{PE}(\cdot)$ is the sinusoidal position encoding, and $Q_I \in \mathbb{R}^{\mathcal{K}\times D}$. Notably, each intention query takes charge of predicting trajectories for a specific motion mode, which stabilizes the training process and facilitates predicting multimodal trajectories since each motion mode has their own learnable embedding. Thanks to their learnable and adaptive properties, we only need a small number of queries (*e.g.*, 64 queries in our setting) for efficient intention localization, instead of using densely-placed goal candidates [65, 21] to cover the destinations of the agents.

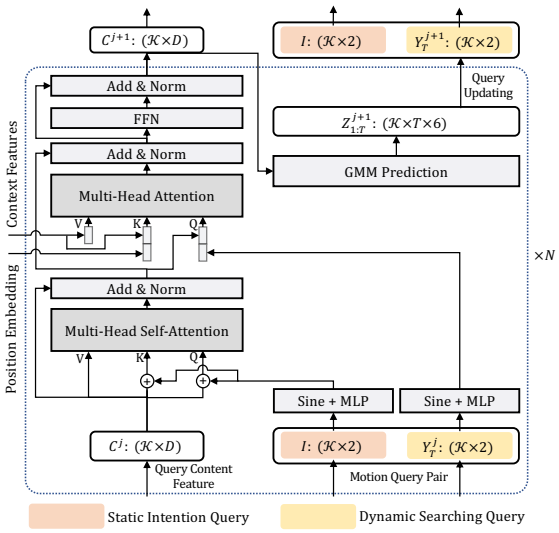


Figure 2: The network structure of our motion decoder network with motion query pair.

Local movement refinement aims to complement with global intention localization by iteratively gathering fine-grained trajectory features for refining the trajectories. We propose *dynamic searching query* to adaptively probe trajectory features for each motion mode. Each dynamic searching query is also the position embedding of a spatial point, which is initialized with its corresponding intention point but will be dynamically updated according to the predicted trajectory in each decoder layer. Specifically, given the predicted future trajectories $Y_{1:T}^j = \{Y_i^j \in \mathbb{R}^{\mathcal{K}\times 2} \mid i = 1, \cdots, T\}$ in j-th decoder layer, the dynamic searching query of $(j+1)$-th decoder layer is updated as follows:

$$Q_S^{j+1} = \text{MLP}\left(\text{PE}(Y_T^j)\right). \tag{5}$$

As shown in Figure 3, for each motion query pair, we propose a *dynamic map collection* module to extract fine-grained trajectory features by querying map features from a trajectory-aligned local region, which is implemented by collecting L polylines whose centers are closest to the predicted trajectory. As the agent's behavior largely depends on road maps, this local movement refinement strategy enables to continually focus on latest local context information for iterative motion refinement.

Attention module with motion query pair. In each decoder layer, static intention query is utilized to propagate information among different motion intentions, while dynamic searching query is utilized to aggregate trajectory-specific features from scene context features. Specifically, we utilize static intention query as the position embedding of self-attention module as follows:

$$C_{\text{sa}}^j = \text{MultiHeadAttn}(\text{query}{=}C^{j-1} + Q_I,\ \text{key}{=}C^{j-1} + Q_I,\ \text{value}{=}Q_I), \tag{6}$$

where $C^{j-1} \in \mathbb{R}^{\mathcal{K}\times D}$ is query content features from $(j-1)$-th decoder layer, C^0 is initialized to zeros, and $C_{\text{sa}}^j \in \mathbb{R}^{\mathcal{K}\times D}$ is the updated query content. Next, we utilize dynamic searching query as query position embedding of cross attention to probe trajectory-specific features from the outputs of encoder. Inspired by [34, 30], we concatenate content features and position embedding for both query and key to decouple their contributions to the attention weights. Two cross-attention modules

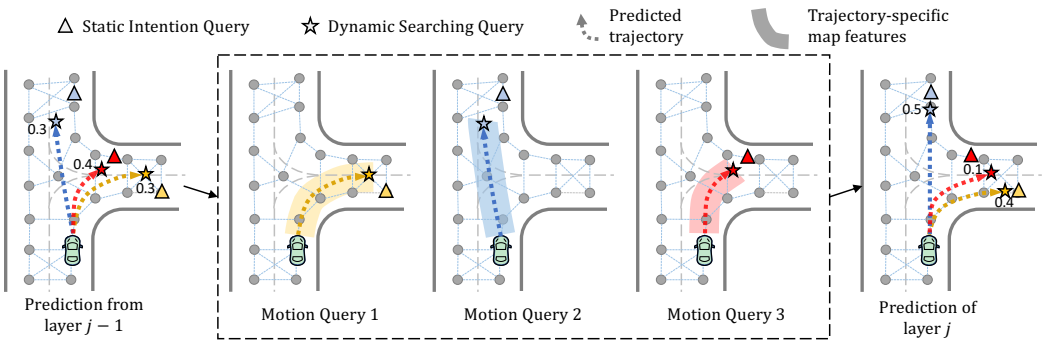


Figure 3: The illustration of dynamic map collection module for iterative motion refinement.

are adopted separately for aggregating features from both agent features A and map features M as:

$$\begin{aligned} C_A^j &= \text{MultiHeadAttn}(\text{query}=[C_{\text{sa}}^j, Q_S^j],\ \text{key}=[A, \text{PE}_A],\ \text{value}=A), \\ C_M^j &= \text{MultiHeadAttn}(\text{query}=[C_{\text{sa}}^j, Q_S^j],\ \text{key}=[\alpha(M), \text{PE}_{\alpha(M)}],\ \text{value}=\alpha(M)), \\ C^j &= \text{MLP}([C_A^j, C_M^j]) \end{aligned} \tag{7}$$

where $[\cdot,\cdot]$ indicates feature concatenation, $\alpha(M)$ is the aforementioned dynamic map collection module to collect L trajectory-aligned map features for motion refinement. Note that for simplicity, in Eq. (6) and (7), we omit the residual connection and feed-forward network in transformer layer [50].

Finally, $C^j \in \mathbb{R}^{\mathcal{K}\times D}$ is the updated query content features for each motion query pair in j-th layer.

Multimodal motion prediction with Gaussian Mixture Model. For each decoder layer, we append a prediction head to C^j for generating future trajectories. As the behaviors of the agents are highly multimodal, we follow [10, 49] to represent the distribution of predicted trajectories with Gaussian Mixture Model (GMM) at each time step. Specifically, for each future time step $i \in \{1, \cdots, T\}$, we predict the probability p and parameters $(\mu_x, \mu_y, \sigma_x, \sigma_y, \rho)$ of each Gaussian component as follows

$$Z_{1:T}^j = \text{MLP}(C^j), \tag{8}$$

where $Z_i^j \in \mathbb{R}^{\mathcal{K}\times 6}$ includes $\mathcal{K}$ Gaussian components $\mathcal{N}_{1:\mathcal{K}}(\mu_x, \sigma_x; \mu_y, \sigma_y; \rho)$ with probability distribution $p_{1:\mathcal{K}}$. The predicted distribution of agent's position at time step i can be formulated as:

$$P_i^j(o) = \sum_{k=1}^{\mathcal{K}} p_k \cdot \mathcal{N}_k(o_x - \mu_x, \sigma_x; o_y - \mu_y, \sigma_y; \rho). \tag{9}$$

where $P_i^j(o)$ is the occurrence probability of the agent at spatial position $o \in \mathbb{R}^2$. The predicted trajectories $Y_{1:T}^j$ can be generated by simply extracting the predicted centers of Gaussian components.

3.3 Training Losses

Our model is trained end-to-end with two training losses. The first auxiliary loss is $L1$ regression loss to optimize the outputs of Eq. (3). For the second Gaussian regression loss, we adopt negative log-likelihood loss according to Eq. (9) to maximum the likelihood of ground-truth trajectory. Inspired by [10, 49], we adopt a hard-assignment strategy that selects one closest motion query pair as positive Gaussian component for optimization, where the selection is implemented by calculating the distance between each intention point and the endpoint of GT trajectory. The Gaussian regression loss is adopted in each decoder layer, and the final loss is the sum of the auxiliary regression loss and all the Gaussian regression loss with equal loss weights. Please refer to appendix for more loss details.

4 Experiments

4.1 Experimental Setup

Dataset and metrics. We evaluate our approach on the large-scale Waymo Open Motion Dataset (WOMD) [15], which mines interesting interactions from real-world traffic scenes and is currently

Table 1: Performance comparison of marginal motion prediction on the validation and test set of Waymo Open Motion Dataset. †: The results are shown in *italic* for reference since their performance is achieved with model ensemble techniques. We only evaluate our default setting MTR on the test set by submitting to official test server due to the limitation of submission times of WOMD.

	Method	Reference	minADE ↓	minFDE ↓	Miss Rate ↓	**mAP** ↑
Test	MotionCNN [26]	CVPRw 2021	0.7400	1.4936	0.2091	0.2136
	ReCoAt [66]	CVPRw 2021	0.7703	1.6668	0.2437	0.2711
	DenseTNT [21]	ICCV 2021	1.0387	1.5514	0.1573	0.3281
	SceneTransformer [37]	ICLR 2022	0.6117	**1.2116**	0.1564	0.2788
	MTR (Ours)	-	**0.6050**	1.2207	**0.1351**	**0.4129**
	†MultiPath++ [49]	ICRA 2022	*0.5557*	*1.1577*	*0.1340*	*0.4092*
	†MTR-Advanced-ens (Ours)	-	*0.5640*	*1.1344*	*0.1160*	*0.4492*
Val	MTR (Ours)	-	0.6046	1.2251	0.1366	**0.4164**
	MTR-e2e (Ours)	-	**0.5160**	**1.0404**	**0.1234**	0.3245
	†MTR-ens (Ours)	-	*0.5686*	*1.1534*	*0.1240*	*0.4323*
	†MTR-Advanced-ens (Ours)	-	*0.5597*	*1.1299*	*0.1167*	*0.4551*

Table 2: Performance comparison of joint motion prediction on the interactive validation and test set of Waymo Open Motion Dataset.

	Method	Reference	minADE ↓	minFDE ↓	Miss Rate ↓	**mAP** ↑
Test	Waymo LSTM baseline [15]	ICCV 2021	1.9056	5.0278	0.7750	0.0524
	HeatIRm4 [36]	CVPRw 2021	1.4197	3.2595	0.7224	0.0844
	AIR^2 [58]	CVPRw 2021	1.3165	2.7138	0.6230	0.0963
	SceneTransformer [37]	ICLR 2022	0.9774	2.1892	0.4942	0.1192
	M2I [45]	CVPR 2022	1.3506	2.8325	0.5538	0.1239
	MTR (Ours)	-	**0.9181**	**2.0633**	**0.4411**	**0.2037**
Val	MTR (Ours)	-	0.9132	2.0536	0.4372	0.1992

the most diverse interactive motion dataset. There are two tasks in WOMD with separate evaluation metrics: (1) The *marginal motion prediction challenge* that independently evaluates the predicted motion of each agent (up to 8 agents per scene). (2) The *joint motion prediction challenge* that needs to predict the joint future positions of 2 interacting agents for evaluation. Both of them provide 1 second of history data and aim to predict 6 marginal or joint trajectories of the agents for 8 seconds into the future. There are totally $487k$ training scenes, and about $44k$ validation scenes and $44k$ testing scenes for each challenge. We utilize the official evaluation tool to calculate the evaluation metrics, where the mAP and miss rate are the most important ones as in the official leaderboard[55, 54].

Implementation details. For the context encoding, we stack 6 transformer encoder layers. The road map is represented as multiple polylines, where each polyline contains up to 20 points (about $10m$ in WOMD). We select $N_m = 768$ nearest map polylines around the interested agent. The number of neighbors in encoder's local self-attention is set to 16. The encoder hidden feature dimension is set as $D = 256$. For the decoder modules, we stack 6 decoder layers. L is set to 128 to collect the closest map polylines from context encoder for motion refinement. By default, we utilize 64 motion query pairs where their intention points are generated by conducting k-means clustering algorithm on the training set. To generate 6 future trajectories for evaluation, we use non-maximum suppression (NMS) to select top 6 predictions from 64 predicted trajectories by calculating the distances between their endpoints, and the distance threshold is set as $2.5m$. Please refer to Appendix for more details.

Training details. Our model is trained in an end-to-end manner by AdamW optimizer with a learning rate of 0.0001 and batch size of 80 scenes. We train the model for 30 epochs with 8 GPUs (NVDIA RTX 8000), and the learning rate is decayed by a factor of 0.5 every 2 epochs from epoch 20. The weight decay is set as 0.01 and we do not use any data augmentation.

MTR-e2e for end-to-end motion prediction. We also propose an end-to-end variant of MTR, called MTR-e2e, where only 6 motion query pairs are adopted so as to remove NMS post processing. In the training process, instead of using static intention points for target assignment as in MTR, MTR-e2e selects positive mixture component by calculating the distances between its 6 predicted trajectories and the GT trajectory, since 6 intention points are too sparse to well cover all potential future motions.

4.2 Main Results

Performance comparison for marginal motion prediction. Table 1 shows our main results for marginal motion prediction, our MTR outperforms previous ensemble-free approaches [21, 37] with

Table 3: Effects of different components in MTR framework. All models share the same encoder network. "latent learnable embedding" indicates using 6 latent learnable embeddings as queries of decoder network, and "iterative refinement" indicates using 6 stacked decoders for motion refinement.

Global Intention Localization	Iterative Refinement	Local Movement Refinement	Dense Future Prediction	minADE ↓	minFDE ↓	Miss Rate ↓	**mAP** ↑
Latent learnable embedding	×	×	×	0.6829	1.4841	0.2128	0.2633
Static intention query	×	×	×	0.7036	1.4651	0.1845	0.3059
Static intention query	✓	×	×	0.6919	1.4217	0.1776	0.3171
Static intention query	✓	✓	×	0.6833	1.4059	0.1756	0.3234
Static intention query	✓	×	✓	0.6735	1.3847	0.1706	0.3284
Static intention query	✓	✓	✓	**0.6697**	**1.3712**	**0.1668**	**0.3437**

remarkable margins, increasing the mAP by $+8.48\%$ and decreasing the miss rate from 15.64% to 13.51%. In particular, our single-model results of MTR also achieve better mAP than the latest work MultiPath++ [49], where it uses a novel model ensemble strategy that boosts its performance.

Table 1 also shows the comparison of MTR variants. MTR-e2e achieves better minADE and minFDE by removing NMS post-processing, while MTR achieves better mAP since it learns explicit meaning of each motion query pair that produces more confident intention predictions. We also propose a simple model ensemble strategy to merge the predictions of MTR and MTR-e2e and utilize NMS to remove redundant predictions (denoted as MTR-ens), and it takes the best of both models and achieves much better mAP. By adopting such ensemble strategy to 7 variants of our framework (*e.g.*, more decoder layers, different number of queries, larger hidden dimension), our advanced ensemble results (denoted as MTR-Advanced-ens) achieve best performance on the test set leaderboard.

Performance comparison for joint motion prediction. To evaluate our approach for joint motion prediction, we combine the marginal predictions of two interacting agents into joint prediction as in [7, 15, 45], where we take the top 6 joint predictions from 36 combinations of these two agents. The confidence of each combination is the product of marginal probabilities. Table 2 shows that our approach outperforms state-of-the-arts [37, 45] with large margins on all metrics. Particularly, our MTR boosts the mAP from 12.39% to 20.37% and decreases the miss rate from 49.42% to 44.11%. The remarkable performance gains demonstrate the effectiveness of MTR for predicting scene-consistent future trajectories. Besides that, we also provide some qualitative results in Figure 5 to show our predictions in complicated interacting scenarios.

As of May 19, 2022, our MTR ranks 1st on the motion prediction leaderboard of WOMD for both two challenges [55, 54]. Our approach with more ensembled variants of MTR (*i.e.*, MTR-Advacned-ens) also won the champion of Motion Prediction Challenge in Waymo Open Dataset Challenge 2022 [56, 44]. The significant improvements manifest the effectiveness of MTR framework.

4.3 Ablation Study

We study the effectiveness of each component in MTR. For efficiently conducting ablation experiments, we uniformly sampled 20% frames (about $97k$ scenes) from the WOMD training set according to their default order, and we empirically find that it has similar distribution with the full training set. All models are evaluated with marginal motion prediction metric on the validation set of WOMD.

Effects of the motion decoder network. We study the effectiveness of each component in our decoder network, including global intention localization, iterative refinement and local movement refinement. Table 3 shows that all components contributes remarkably to the final performance in terms of the official ranking metric mAP. Especially, our proposed static intention queries with intention points achieves much better mAP (*i.e.*, $+4.26\%$) than the latent learnable embeddings thanks to its mode-specific querying strategy, and both the iterative refinement and local movement refinement strategy continually improve the mAP from 30.59% to 32.34% by aggregating more fine-grained trajectory features for motion refinement.

Effects of dense future prediction. Table 3 shows that our proposed dense future prediction module significantly improves the quality of predicted trajectories (*e.g.*, $+1.78\%$ mAP), which verifies that future interactions of the agents' trajectories are important for motion prediction and our proposed strategy can learn such interactions to predict more reliable trajectories.

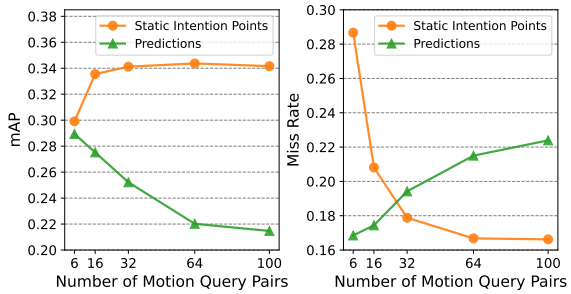
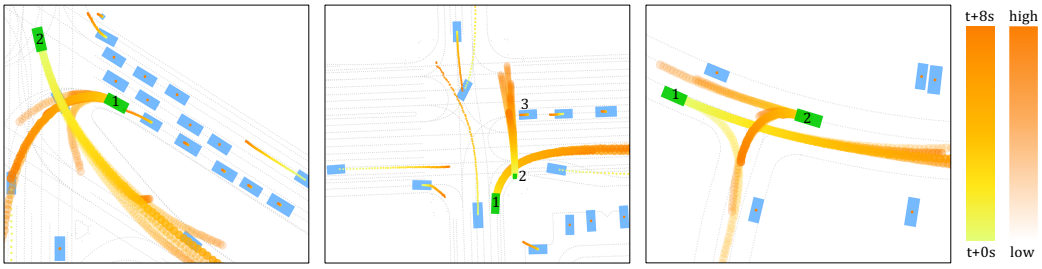


Figure 4: MTR framework with different number of motion query pairs, and two different colored lines demonstrate different strategies for selecting the positive mixture component during training process.

Table 4: Effects of local self-attention in transformer encoder. "#polyline" is the number of input map polylines used for context encoding, and a large number of polylines indicate that there is a larger map context around the interested agent. "OOM" indicates running out of memory.

Attention	#Polyline	minADE ↓	minFDE ↓	MR ↓	**mAP** ↑
Global	256	0.683	1.4031	0.1717	0.3295
Global	512	0.6783	1.4018	0.1716	0.3280
Global	768	OOM	OOM	OOM	OOM
Local	256	0.6724	1.3835	0.1683	0.3372
Local	512	0.6707	1.3749	0.1670	0.3392
Local	768	**0.6697**	**1.3712**	0.1668	0.3437
Local	1024	0.6757	1.3782	**0.1663**	**0.3452**



(a) **V2** is passing the intersection to turn left with high speed. Our model predicts multimodal behaviors for **V1:** turn left or make a U-turn. In any case, **V1** is predicted to yield for **V2**.

(b) **P2** is passing the road through the crosswalk while **V1** is on the right-turn lane to turn right. Both **V1** and **V3** are predicted to yield for **P2**.

(c) Our model predicts multimodal behaviors for **V1**: go straight and turn right, since it still has a distance to the intersection. **V2** is predicted to yield for **V1** when turning left, since **V1** is moving fast towards the intersection.

Figure 5: Qualitative results of MTR framework on WOMD. There are two interested agents in each scene (green rectangle), where our model predicts 6 multimodal future trajectories for each of them. For other agents (blue rectangle), a single trajectory is predicted by dense future prediction module. We use gradient color to visualize the trajectory waypoints at different future time step, and trajectory confidence is visualized by setting different transparent. Abbreviation: Vehicle (**V**), Pedestrian (**P**).

Effects of local attention for context encoding. Table 4 shows that by taking the same number of map polylines as input, local self-attention in transformer encoder achieves better performance than global attention (*i.e.*, $+0.77\%$ mAP for 256 polylines and $+1.12\%$ mAP for 512 polylines), which verifies that the input local structure is important for motion prediction and introducing such prior knowledge with local attention can benefit the performance. More importantly, local attention is more memory-efficient and the performance keeps growing when improving the number of map polylines from 256 to 1,024, while global attention will run out of memory due to its quadratic complexity.

Effects of the number of motion query pairs with different training strategies. As mentioned before, during training process, MTR and MTR-e2e adopt two different strategies for assigning positive mixture component, where MTR depends on static intention points (denoted as α) while MTR-e2e utilizes predicted trajectories (denoted as β). Figure 4 investigates the effects of the number of motion query pairs under these two strategies, where we have the following observations: (1) When increasing the number of motion query pairs, strategy α achieves much better mAP and miss rate than strategy β. Because intention query points can ensure more stable training process since each intention query points is responsible to a specific motion mode. In contrast, strategy β depends on unstable predictions and the positive component may randomly switch among all components, so a large number of motion query pairs are hard to be optimized with strategy β. (2) The explicit meaning of each intention query point also illustrates the reason that strategy α consistently achieves much better mAP than strategy β, since it can predict trajectories with more confident scores to benefit mAP metric. (3) From another side, when decreasing the number of motion query pairs, the miss rate of strategy α greatly increases, since a limit number of intention query points can not well cover all potential motions of agents. Conversely, strategy β works well for a small number of motion query pairs since its queries are not in charge of specific region and can globally adapt to any region.

5 Conclusion

In this paper, we present MTR, a novel framework for multimodal motion prediction. The motion query pair is defined to model motion prediction as the joint optimization of global intention localization and local movement refinement. The global intention localization adopts a small set of learnable static intention queries to efficiently capture agent's motion intentions, while the local movement refinement conducts iterative motion refinement by continually probing fine-grained trajectory features. The experiments on both marginal and joint motion prediction challenges of large-scale WOMD dataset show that our approach achieves state-of-the-art performance.

Limitations. The proposed framework adopts an agent-centric strategy to predict multimodal future trajectories for one interested agent, leading redundant context encoding for multiple interested agents in the same scene. Hence, how to develop a framework that can simultaneously predict multimodal motion for multiple agents is one important future work. Besides, the rule-based post-processing can result in suboptimal predictions for minADE and minFDE metrics, and how to design a better strategy to produce a required number of future trajectories (e.g., 6 trajectories) from full multimodal predictions (e.g., 64 predictions) is also worth exploring for a more robust framework.

References

[1] Argoverse 2. Argoverse 2: Motion forecasting competition. `https://eval.ai/web/challenges/challenge-page/1719/leaderboard/4098`, 2022. Accessed: 2022-08-02.

[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *CVPR*, 2016.

[3] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. In *ICLR*, 2022.

[4] Yuriy Biktairov, Maxim Stebelev, Irina Rudenko, Oleh Shliazhko, and Boris Yangel. Prank: motion prediction based on ranking. In *NeurIPS*, 2020.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[6] Sergio Casas, Cole Gulino, Renjie Liao, and Raquel Urtasun. Spagnn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. In *ICRA*, 2020.

[7] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *ECCV*, 2020.

[8] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *CoRL*, 2018.

[9] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *CVPR*, 2021.

[10] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *CoRL*, 2019.

[11] Xuesong Chen, Shaoshuai Shi, Benjin Zhu, Ka Chun Cheung, Hang Xu, and Hongsheng Li. Mppnet: Multi-frame feature intertwining with proxy points for 3d temporal object detection. In *ECCV*, 2022.

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

[13] 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 *WACV*, 2020.

[14] 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 *ICLR*, 2021.

[15] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *ICCV*, 2021.

[16] Liangji Fang, Qinhong Jiang, Jianping Shi, and Bolei Zhou. Tpnet: Trajectory proposal network for motion prediction. In *CVPR*, 2020.

[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 *CVPR*, 2020.

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

[19] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Home: Heatmap output for future motion estimation. In *ITSC*, 2021.

[20] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Thomas: Trajectory heatmap output with learned multi-agent sampling. In *arXiv preprint arXiv:2110.06607*, 2021.

[21] Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *ICCV*, 2021.

[22] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *CVPR*, 2018.

[23] Joey Hong, Benjamin Sapp, and James Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *CVPR*, 2019.

[24] Xiaosong Jia, Penghao Wu, Li Chen, Hongyang Li, Yu Liu, and Junchi Yan. Hdgt: Heterogeneous driving graph transformer for multi-agent trajectory prediction via scene encoding. In *arXiv preprint arXiv:2205.09753*, 2022.

[25] Stepan Konev. Mpa: Multipath++ based architecture for motion prediction. In *arXiv preprint arXiv:2206.10041*, 2022.

[26] Stepan Konev, Kirill Brodt, and Artsiom Sanakoyeu. Motioncnn: A strong baseline for motion prediction in autonomous driving. In *Workshop on Autonomous Driving, CVPR*, 2021.

[27] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *CVPR*, 2022.

[28] Jiachen Li, Fan Yang, Masayoshi Tomizuka, and Chiho Choi. Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning. In *NeurIPS*, 2020.

[29] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *ECCV*, 2020.

[30] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. DAB-DETR: Dynamic anchor boxes are better queries for DETR. In *ICLR*, 2022.

[31] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal motion prediction with stacked transformers. In *CVPR*, 2021.

[32] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *ECCV*, 2020.

[33] Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. Mantra: Memory augmented networks for multiple trajectory prediction. In *CVPR*, 2020.

[34] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional detr for fast training convergence. In *ICCV*, 2021.

[35] 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 *ICRA*, 2020.

[36] Xiaoyu Mo, Zhiyu Huang, and Chen Lv. Multi-modal interactive agent trajectory prediction using heterogeneous edge-enhanced graph attention network. In *Workshop on Autonomous Driving, CVPR*, 2021.

[37] Jiquan Ngiam, Vijay Vasudevan, Benjamin Caine, 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 future trajectories of multiple agents. In *ICLR*, 2022.

[38] Seong Hyeon Park, Gyubok Lee, Jimin Seo, Manoj Bhat, Minseok Kang, Jonathan Francis, Ashwin Jadhav, Paul Pu Liang, and Louis-Philippe Morency. Diverse and admissible trajectory forecasting through multimodal context understanding. In *ECCV*, 2020.

[39] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *CVPR*, 2020.

[40] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[41] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *ECCV*, 2018.

[42] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *ICCV*, 2019.

[43] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *ECCV*, 2020.

[44] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Mtr-a: 1st place solution for 2022 waymo open dataset challenge–motion prediction. *arXiv preprint arXiv:2209.10033*, 2022.

[45] Qiao Sun, Xin Huang, Junru Gu, Brian C Williams, and Hang Zhao. M2i: From factored marginal trajectory prediction to interactive prediction. In *CVPR*, 2022.

[46] Charlie Tang and Russ R Salakhutdinov. Multiple futures prediction. In *NeurIPS*, 2019.

[47] Xiaocheng Tang, Soheil Sadeghi Eshkevari, Haoyu Chen, Weidan Wu, Wei Qian, and Xiaoming Wang. Golfer: Trajectory prediction with masked goal conditioning mnm network. In *arXiv preprint arXiv:2207.00738*, 2022.

[48] Ekaterina Tolstaya, Reza Mahjourian, Carlton Downey, Balakrishnan Vadarajan, Benjamin Sapp, and Dragomir Anguelov. Identifying driver interactions via conditional behavior prediction. In *ICRA*, 2021.

[49] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi Pang Lam, Dragomir Anguelov, et al. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In *ICRA*, 2022.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[51] Jiashun Wang, Huazhe Xu, Medhini Narasimhan, and Xiaolong Wang. Multi-person 3d motion prediction with multi-range transformers. In *NeurIPS*, 2021.

[52] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[53] Yuting Wang, Hangning Zhou, Zhigang Zhang, Chen Feng, Huadong Lin, Chaofei Gao, Yizhi Tang, Zhenting Zhao, Shiyu Zhang, Jie Guo, et al. Tenet: Transformer encoding network for effective temporal flow on motion prediction. In *arXiv preprint arXiv:2207.00170*, 2022.

[54] Waymo. Waymo open dataset interaction prediction challenge 2021. `https://waymo.com/open/challenges/2021/interaction-prediction/`, 2021. Accessed: 2022-05-19.

[55] Waymo. Waymo open dataset motion prediction challenge 2021. `https://waymo.com/open/challenges/2021/motion-prediction`, 2021. Accessed: 2022-05-19.

[56] Waymo. Waymo open dataset motion prediction challenge 2022. `https://waymo.com/open/challenges/2022/motion-prediction/`, 2022. Accessed: 2022-09-21.

[57] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *NeurIPS*, 2021.

[58] David Wu and Yunnan Wu. Air2 for interaction prediction. In *Workshop on Autonomous Driving, CVPR*, 2021.

[59] Yi Xu, Dongchun Ren, Mingxia Li, Yuehai Chen, Mingyu Fan, and Huaxia Xia. Tra2tra: Trajectory-to-trajectory prediction with a global social spatial-temporal attentive neural network. In *RA-L*, 2021.

[60] Zetong Yang, Li Jiang, Yanan Sun, Bernt Schiele, and Jiaya Jia. A unified query-based paradigm for point cloud understanding. In *CVPR*, 2022.

[61] Maosheng Ye, Tongyi Cao, and Qifeng Chen. Tpcn: Temporal point cloud networks for motion forecasting. In *CVPR*, 2021.

[62] Fangao Zeng, Bin Dong, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Motr: End-to-end multiple-object tracking with transformer. In *arXiv preprint arXiv:2105.03247*, 2021.

[63] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *arXiv preprint arXiv:2203.03605*, 2022.

[64] Yifan Zhang, Jinghuai Zhang, Jindi Zhang, Jianping Wang, Kejie Lu, and Jeff Hong. A novel learning framework for sampling-based motion planning in autonomous driving. In *AAAI*, 2020.

[65] 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 *CoRL*, 2020.

[66] Chen Lv Zhiyu Huang, Xiaoyu Mo. Recoat: A deep learning framework with attention mechanism for multi-modal motion prediction. In *Workshop on Autonomous Driving, CVPR*, 2021.

[67] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021.

[68] Yanliang Zhu, Deheng Qian, Dongchun Ren, and Huaxia Xia. Starnet: Pedestrian trajectory prediction using deep neural network in star topology. In *IROS*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work is only for academic research purpose.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Our implementation details are posted on Section 4.1 and the supplemental materials. The data is public dataset and our code will be available.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.1 for training details, and the data splits of ablation experiments are mentioned in Section 4.3.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We use fixed random seed.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.1.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See our reference.
   (b) Did you mention the license of the assets? [No] They are publicly available for academic use.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] They are publicly available for academic use.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We only use the public dataset.

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Implementation Details

More architecture details. We train a single model for predicting the future motion of all three categories (*i.e.*, Vehicle, Pedestrian, Cyclist), and each category has their own motion query pairs.

The input history state A_{in} contains three types of information, including agent history motion state (*i.e.*, position, object size, heading angle and velocity), one-hot category mask of each agent and one-hot time embedding of each history time step. The polyline encoder for A_{in} contains a three-layer MLP with feature dimension 256. The input map feature M_{in} contains three types of information, including the position of each polyline point, the polyline direction at each point, and the type of each polyline. The polyline encoder for M_{in} contains a five-layer MLP with feature dimension 64, where we adopt a smaller feature dimension since the number of map polylines is much larger than the number of agents. Both two polyline encoders are finally projected to 256 feature dimension with another linear layer separately.

For the dense future prediction module, we adopt a three-layer MLP with intermediate feature dimension 512 for predicting future position and velocity of all agents. For the prediction head in each decoder layer, we adopt a three-layer MLP with intermediate feature dimension 512, and the model weights are not shared across different decoder layers.

The details of Gaussian regression loss. Given the predicted Gaussian Mixture Models for a specific future time step, we adopt negative log-likelihood loss to maximum the likelihood of the agent's ground-truth position $(\hat{Y}_x, \hat{Y}_y)$ at this time step, and the detailed loss can be formulated as:

$$
\begin{aligned}
L_G &= -\log \mathcal{N}_h(\hat{Y}_x - \mu_x, \sigma_x; \hat{Y}_y - \mu_y, \sigma_y; \rho) - \log(p_h) \qquad (10) \\
&= \log \sigma_x + \log \sigma_y + 0.5 \log(1-\rho^2) + \frac{1}{2(1-\rho^2)} \left(\left(\frac{d_x}{\sigma_x}\right)^2 + \left(\frac{d_y}{\sigma_y}\right)^2 - 2\rho \frac{d_x d_y}{\sigma_x \sigma_y} \right) - \log(p_h),
\end{aligned}
$$

where $d_x = \hat{Y}_x - \mu_x$, $d_y = \hat{Y}_y - \mu_y$, and $\mathcal{N}_h(\mu_x, \sigma_x; \mu_y, \sigma_y; \rho)$ is the selected positive Gaussian component for optimization. p_h is the predicted probability of this selected positive Gaussian component, and we adopt cross entropy loss in the above equation to maximum the probability of selected positive Gaussian component.

Final training losses. Given the above definition of Gaussian regression loss, the final training loss of MTR framework is the sum of all the Gaussian regression loss in each decoder layer and the auxiliary regression loss with equal loss weights.

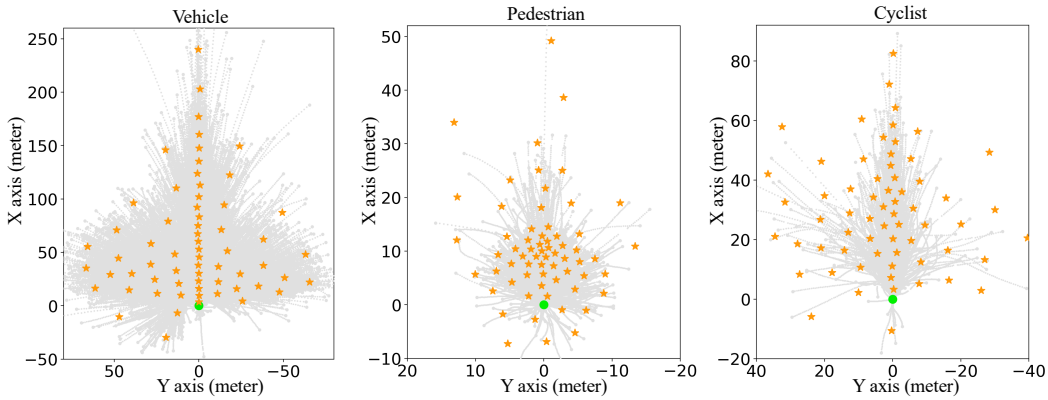


Figure 6: The distribution of static intention points for each category. The green point is the current position of the agent, and the intention points are shown as orange stars. The gray dotted lines indicate the distribution of ground-truth trajectory for each category, and note that only 10% ground-truth trajectories are drawn in the figure for better visualization.

The distribution of static intention points. As mentioned in the paper, we adopt k-means clustering algorithm to generate 64 static intention points for each category. As shown in Figure 6, the generated

intention points can well cover most motion intentions by jointly considering both the velocity and direction of future motion. For example, vehicle category has a series of intention points in the heading direction that actually model the same intent direction with different velocities.

B Per-class Results of MTR Framework

As shown in Table 5 and 6, we report the per-category performance of our approach for both the marginal and joint motion prediction challenges of Waymo Open Motion Dataset [15] for reference.

Table 5: Per-class performance of marginal motion prediction on the validation and test set of Waymo Open Motion Dataset. †: The results are shown in *italic* for reference since the performance is achieved with model ensemble technique.

	Setting	Category	minADE ↓	minFDE ↓	Miss Rate ↓	mAP ↑
Test	MTR	Vehicle	0.7642	1.5257	0.1514	0.4494
		Pedestrian	0.3486	0.7270	0.0753	0.4331
		Cyclist	0.7022	1.4093	0.1786	0.3561
		Avg	0.6050	1.2207	0.1351	0.4129
Val	MTR	Vehicle	0.7665	1.5359	0.1527	0.4534
		Pedestrian	0.3456	0.7260	0.0741	0.4364
		Cyclist	0.7016	1.4134	0.1830	0.3594
		Avg	0.6046	1.2251	0.1366	0.4164
	MTR-e2e	Vehicle	0.6087	1.2161	0.1160	0.3708
		Pedestrian	0.3046	0.6254	0.0696	0.3028
		Cyclist	0.6346	1.2796	0.1845	0.2998
		Avg	0.5160	1.0404	0.1234	0.3245

Table 6: Per-class performance of joint motion prediction on the validation and test set of Waymo Open Motion Dataset.

	Setting	Category	minADE ↓	minFDE ↓	Miss Rate ↓	mAP ↑
Test	MTR	Vehicle	0.9793	2.2157	0.3833	0.2977
		Pedestrian	0.7098	1.5835	0.3973	0.2033
		Cyclist	1.0652	2.3908	0.5428	0.1102
		Avg	0.9181	2.0633	0.4411	0.2037
Val	MTR	Vehicle	0.9675	2.1728	0.3722	0.3018
		Pedestrian	0.7097	1.5522	0.3984	0.1892
		Cyclist	1.0623	2.4357	0.5410	0.1065
		Avg	0.9132	2.0536	0.4372	0.1992

C Evaluation of Dense Future Prediction

As introduced in the paper, we propose the dense future prediction module to enable the future interactions among the agents, where the accuracy of the predicted trajectories for all neighboring agents is important for predicting better multimodal trajectories of our interested agent. As shown in Table 7, our simple dense future prediction module can predict accurate future trajectory for all agents, where the average miss rate with $6.0m$ distance threshold is 21.11% for future motion prediction of 8 seconds. It's worth noting that this miss rate is achieved with a single predicted future trajectory for each agent. Although the trajectory generated by the dense future prediction module is still not as accurate as that of the decoder network, they can already benefit the multimodal motion prediction of our interested agent by enabling the future interactions of the agents.

D Performance Comparison on Argoverse 2 Dataset

The Argoverse 2 Motion Forecasting Dataset [57] is another large-scale dataset for motion prediction, which contains 250,000 scenarios for training and validation. The model needs to take the history five seconds of each scenarios as niput, and predict the six-second future trajectories of one interested agent, where HDMap is always available to provide map context information. To train our model on

Table 7: The performance of dense future prediction on the validation set of Waymo Open Motion Dataset. The dense future prediction module predicts a single future trajectory (8 seconds into the future) for each agent in the validation set. ADE indicates average displacement error, and FDE indicates final displacement error. The FDE is evaluated based on the last time step of future 8 seconds, and the miss rate is evaluated based on FDE with different distance thresholds.

Category	ADE ↓	FDE ↓	Miss Rate (thresh=$2m$) ↓	Miss Rate (thresh=$6m$) ↓
Vehicle	0.8755	3.4929	0.3343	0.1984
Pedestrian	0.4998	1.9685	0.3317	0.0601
Cyclist	1.6565	6.3868	0.7697	0.3747
Avg	1.0106	3.9494	0.4786	0.2111

Table 8: The performance comparison with the top-10 submissions on the test set leaderboard of Argoverse2 dataset. K is the number of predicted trajectories for calculating the evaluation metrics.

Method	Miss Rate (K=6) ↓	Miss Rate (K=1) ↓	brier-minFDE (K=6) ↓
MTR (Ours)	**0.15**	**0.58**	1.98
TENET [53]	0.19	0.61	**1.90**
OPPred	0.19	0.60	1.92
Qml	0.19	0.62	1.95
GANet	0.17	0.60	1.97
VI LaneIter	0.19	0.61	2.00
QCNet	0.21	0.60	2.14
THOMAS [20]	0.20	0.64	2.16
HDGT [24]	0.21	0.66	2.24
GNA	0.29	0.71	2.45
vilab	0.29	0.71	2.47

this dataset, we adopt the same hyper-parameters as in Waymo dataset, except that the model takes five-second history trajectories as input and needs to predict six-second future trajectories.

We compare out approach with the top-10 submissions on the leaderboard of Argoverse2 dataset [1], where most submissions are tailored for Argoverse2 Motion Forcasting Competition 2022 and are highly competitive. As shown in Table 8, our MTR framework achieves new state-of-the-art performance with remarkable gains on the miss-rate related metrics, demonstrating the great generalizability and robustness of our approach.

Table 9: Effects of the number of neighbors for local self-attention in transformer encoder.

#neighbors	minADE ↓	minFDE ↓	MR ↓	**mAP** ↑
4	**0.6677**	1.3724	0.1672	0.3405
8	0.6681	**1.3673**	0.1670	0.3428
16	0.6697	1.3712	**0.1668**	**0.3437**
32	0.6701	1.3763	0.1678	0.3416
64	0.6727	1.3756	0.1687	0.3367

Table 10: Effects of the number of map polylines for dynamic map collection.

#Polyline	minADE ↓	minFDE ↓	MR ↓	**mAP** ↑
32	0.6735	1.3847	0.1701	0.3317
64	0.6699	**1.3650**	0.1672	0.3386
128	**0.6697**	1.3712	0.1668	**0.3437**
256	0.6704	1.3729	**0.1665**	0.3396

E More Ablation Studies

Effects of the number of neighbors for local self-attention. As shown in Table 9, using 4 neighbors in local self-attention already achieves good performance in terms of mAP, and the mAP metric keeps growing when increasing the number of neighbors from 4 to 16. However, when we increase the neighbors to 64 for conducting local self-attention, the performance drops a bit (*i.e.*, from 34.37% to 33.67%). It demonstrates that a small number of neighbors can better maintain the local structure of input elements and is easier to be optimized for achieving better performance, and a small number of neighbors is also much more computational- and memory-efficient than a larger number of neighbors for conducting self-attention.

Effects of the number of polylines for dynamic map collection. Table 10 shows that increasing the number of collected map polylines from 32 to 128 can constantly improve the mAP metric by retrieving trajectory-specific map features with larger receptive field. However, the performance drops a bit (-0.41% mAP) when collecting 256 map polylines for refining the trajectory, which shows that a larger number of collected map polylines may involve more noise and can not provide accurate trajectory-specific map features for refinement.

Table 11: Effects of the number of decoder layers.

#decoders	minADE ↓	minFDE ↓	MR ↓	**mAP** ↑
3	0.6717	1.3796	0.1686	0.3360
6	0.6697	1.3712	0.1668	**0.3437**
9	**0.6658**	**1.3621**	**0.1661**	**0.3437**

Table 12: Effects of the distribution of static intention points.

#Distribution	minADE ↓	minFDE ↓	MR ↓	**mAP** ↑
uniform grids	0.7214	1.5563	0.1970	0.3178
k-means	**0.6697**	**1.3712**	**0.1668**	**0.3437**

Effects of the number of decoder layers. Table 11 shows that increasing the number of decoder layers can constantly improve the performance, which demonstrates that our stacked transformer decoder layers can iteratively refine the predicted trajectories by continually aggregating more accurate trajectory-specific features. By default, we adopt 6 decoder layers by considering the trade-off between the performance and the efficiency of MTR framework.

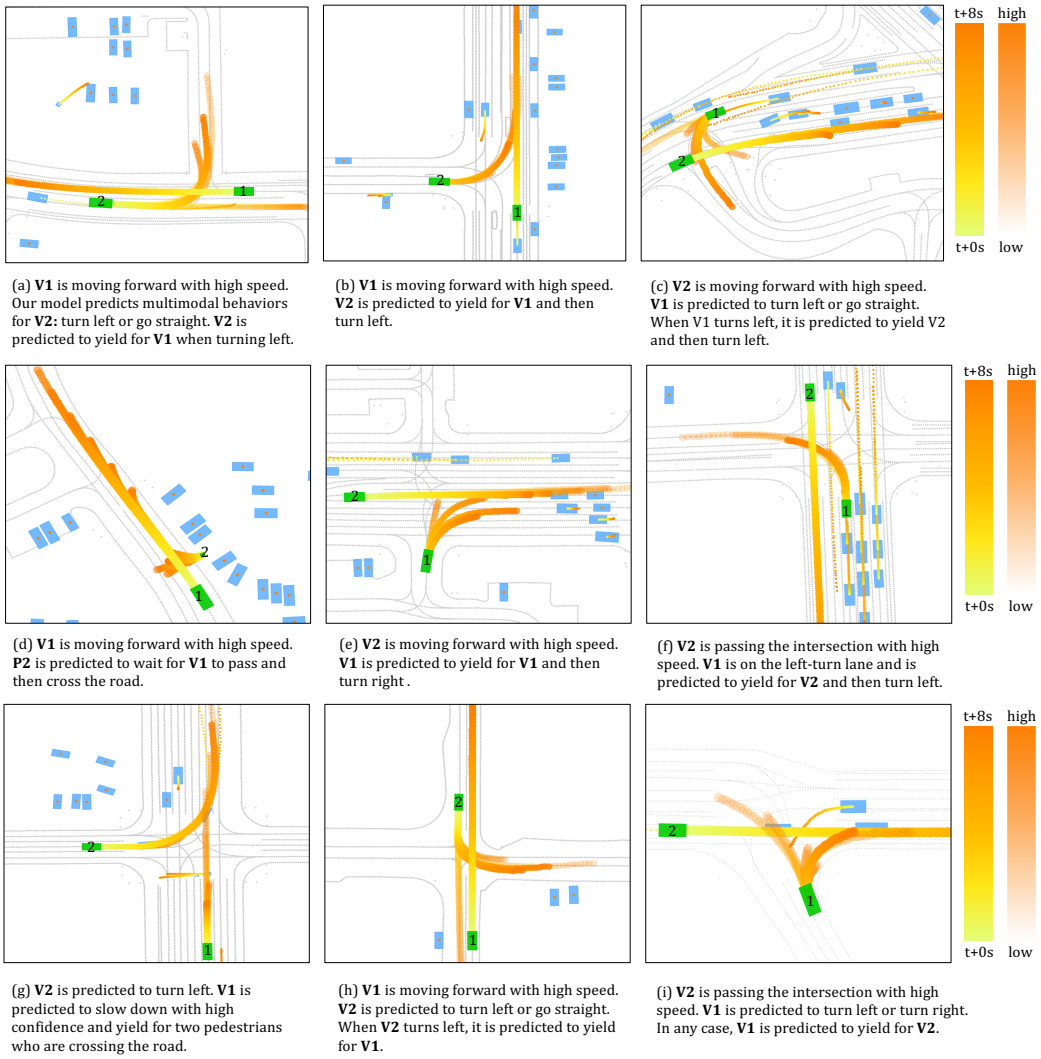
Effects of the distribution of static intention points. As introduced in the paper, we adopt k-means algorithm to generate 64 static intention points as the basis of our motion query pairs. We ablate another simple uniform sampling strategy to generate the same number of static intention points for each category, where we uniformly sample $8 \times 8 = 64$ static intention points by considering the range of trajectory distribution of each category (see Figure 6). As shown in Table 12, the performance drops significantly when replacing k-means algorithm with the uniform sampling for generating static intention points. It verifies that our k-means based algorithm can produce better distribution of static intention points, which can capture more accurate and more complete future motion intentions with a small number of static intention points. Figure 6 also demonstrates the effectiveness of our default static intention points for capturing multimodal motion intentions.

F Qualitative Results

We provide more qualitative results of our MTR framework in Figure 7.

G Notations

As shown in Table 13, we provide a lookup table for notations in the paper.



(a) **V1** is moving forward with high speed. Our model predicts multimodal behaviors for **V2:** turn left or go straight. **V2** is predicted to yield for **V1** when turning left.

(b) **V1** is moving forward with high speed. **V2** is predicted to yield for **V1** and then turn left.

(c) **V2** is moving forward with high speed. **V1** is predicted to turn left or go straight. When V1 turns left, it is predicted to yield V2 and then turn left.

(d) **V1** is moving forward with high speed. **P2** is predicted to wait for **V1** to pass and then cross the road.

(e) **V2** is moving forward with high speed. **V1** is predicted to yield for **V1** and then turn right .

(f) **V2** is passing the intersection with high speed. **V1** is on the left-turn lane and is predicted to yield for **V2** and then turn left.

(g) **V2** is predicted to turn left. **V1** is predicted to slow down with high confidence and yield for two pedestrians who are crossing the road.

(h) **V1** is moving forward with high speed. **V2** is predicted to turn left or go straight. When **V2** turns left, it is predicted to yield for **V1**.

(i) **V2** is passing the intersection with high speed. **V1** is predicted to turn left or turn right. In any case, **V1** is predicted to yield for **V2**.

Figure 7: Qualitative results of MTR framework on WOMD. There are two interested agents in each scene (green rectangle), where our model predicts 6 multimodal future trajectories for each of them. For other agents (blue rectangle), a single trajectory is predicted by dense future prediction module. We use gradient color to visualize the trajectory waypoints at different future time step, and trajectory confidence is visualized by setting different transparent. Abbreviation: Vehicle (V), Pedestrian (P).

Table 13: Lookup table for notations in the paper.

A_{in}	input history motion state of agent
M_{in}	input map features with polyline representation
C_a	input feature dimension of agent's state
N_a	number of agents
t	number of given history frames
N_m	number of map polylines
n	number of points in each map polyline
A_{past}	agent features after polyline encoder
A_{future}	encoded future features of the densely predicted trajectories for all agents
A	agent features in transformer encoder
M	map features in transformer encoder
D	hidden feature dimension of transformer
PE	function of sinusoidal position encoding
κ	function of k-nearest neighbor algorithm
T	number of future frames to be predicted
S_i	predicted future position and velocity of all agents at time step i
I	static intention points
Q_I	static intention query
Q_S^j	dynamic searching query in j-th decoder layer
$Y_{1:T}^j$	predicted future trajectories in j-th decoder layer
L	number of map polylines collected along the predicted trajectory
C^{j-1}	input query content features of j-th decoder layer
C_{sa}^j	query content features after self-attention of j-th decoder layer
C_A^j	query agent features after cross-attention of j-th decoder layer
C_M^j	query map features after cross-attention of j-th decoder layer
α	function of dynamic map collection
$Z_{1:T}^j$	predicted GMM parameters in j-th decoder layer
p_k	predicted probability of k-th component in GMM
$\mathcal{N}_k$	function of k-th component in GMM
$\mathcal{K}$	number of components for each GMM
$P_i^j(o)$	predicted occurrence probability of the agent at spatial position o and time step i