IAIR+: 2nd Place Solution for 2023 Waymo Open Dataset Challenge - Motion Prediction

Miao Kang[1†] Liushuai Shi[1†] Jinpeng Dong[1] Yuhao Huang[1] Ke Ye[1]
Yufeng Hu[1] Junjie Zhang[1] Yonghao Dong[1] Yizhe Li[1] Sanping Zhou[1]
[1]National Key Laboratory of Human-Machine Hybrid Augmented Intelligence,
National Engineering Research Center for Visual Information and Applications,
Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University

Abstract

This technical report presents our solution for Waymo Open Dataset Challenge 2023, Motion Prediction. The goal of motion prediction is to predict the future trajectory given a historical trajectory and corresponding environmental information, e.g.*, neighbor agents and surrounding map. Our solution builds more capacious query sets to represent multimodal motion modes based on a query-based motion transformer framework (MTR [6]). Moreover, an extensively modified polyline encoder based on an FPN-style network is used to capture rich spatio-temporal features, and a "DropKey" trick is adopted to enhance the generalization of the transformer model. In the ensemble step, our solution adopts a similar-mode ensemble, which integrates the trajectories and corresponding scores from different models for each individual query. On the Waymo Motion Prediction v1.2 dataset, our method achieves 45.61% on the validation set and 44.80% on the testing set in terms of average mAP. Our solution ranked 2nd in this challenge, and we hope this solution could provide useful guidance for future work.*

1. Our Solution

Recently, the query-based methods [1, 5–8] achieve advanced performance in motion (trajectory) prediction, which generates additional query (location point or trajectory) to represent the multimodal motion modes. Due to the effectiveness of the query, our solution builds **capacious query set** to better represent multimodal motion modes. Moreover, an extensive **modified polyline encoder** is used to capture rich spatio-temporal features and a **DropKey** trick is used to enhance the generalization of the transformer model. In the ensemble step, a **similar-mode ensemble** is used to integrate the trajectories and corresponding scores from different models for each individual query.

†Co-first authors.

We use a modified query-based motion transformer method as our basic framework referring to MTR [6] as shown in Figure 1. In the rest of the report, we mainly introduce our modified parts (Marked with color boxes in Figure 1) and how to incorporate the modified parts into the MTR. To make a clear clarification, the sign in this report is the same as the MTR.

1.1. Capacious Query Set

The multimodality of motion prediction shows the diversity of future motions, such as turning left, right, or going straight. The proposed capacious query set is obtained by clustering the training dataset to represent this multimodality. Given the training set, we normalize the trajectory of the training set with two steps. First, the beginning point of the trajectory is translated into the origin of the coordinate system. Second, we rotate the trajectory to align the X-axis. In the implementation, we empirically use the direction of the initial point of the trajectory provided in the dataset as the rotated angle.

Once obtained the normalized trajectory, we extract the goals (endpoint) of the normalized trajectory. These goals are clustered into $\mathcal{K}$ cluster centers by the K-means algorithm. The $\mathcal{K}$ cluster centers clustered from real trajectories could represent diverse motion modes. Thus, we regard the $\mathcal{K}$ cluster centers as our capacious query set $I \in \mathbb{R}^{\mathcal{K}\times 2}$.

Our capacious query set is used to place the static intention point, which is a part of the Motion Query Pair as shown in Figure 1.

1.2. Modified Polyline Encoder

Referring to MTR, we first obtain the input representation of the agent and map. Specifically, we use the vectorized representation and agent-centric normalization strategy to generate the history state of N_a agents as $A_{in} \in$

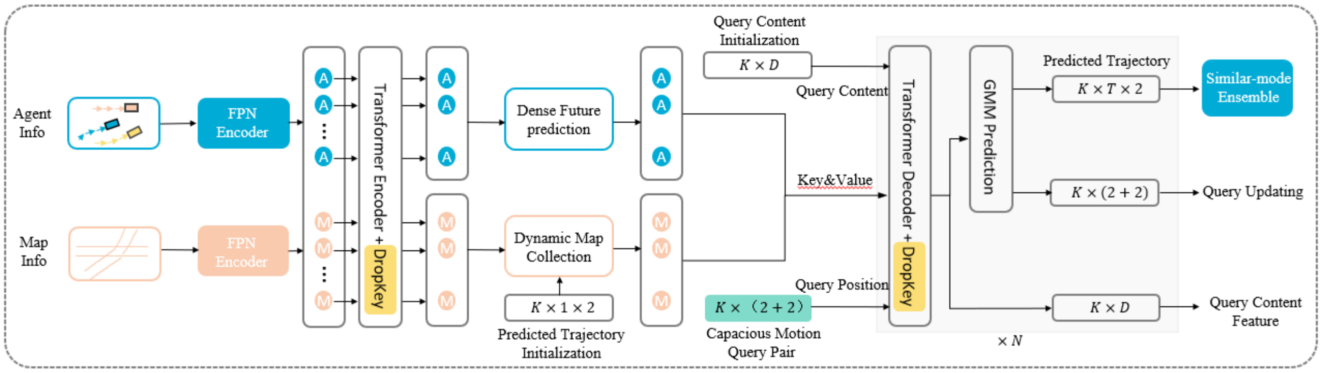

Figure 1. **Overview of our modified framework**.

$\mathbb{R}^{N_a \times t \times C_a}$ and road map $M_{in} \in \mathbb{R}^{N_m \times n \times C_m}$, where t is the number of historical frames, C_a is the dimension of each state, N_m is the number of map polylines, n is the number of points in each polyline, and C_m is the dimension of each point.

Our modified polyline encoder shown in Figure 2 uses an FPN-style network [3] to extract agent features and map features on the history state A_{in}, and road map M_{in}. Specifically, with the input features of agents or maps, we firstly leverage a multi-scale 1D CNN layers to effectively extract the spatial features in C_a dimension. With the stride of 2, the CNN extracts the features in a larger spatial scale. The sum of different level features enriches the representation ability of encoder. Then, we further utilize a linear layer to gather the temporal information and obtain spatial-temporal feature.

The output agent features $A_p \in \mathbb{R}^{N_a \times D}$ and map features $M_p \in \mathbb{R}^{N_m \times D}$ are fed into the Transformer Encoder as shown in Figure 1.

1.3. DropKey Trick

To enhance the generalization of the Transformer, we use the DropKey [2] trick in the attention mechanism at the training step. Specifically, the pseudocode of DropKey is shown in Algorithm 1.

Algorithm 1 Algorithm of DropKey

```
# N: token number, D: token dim
# Q: query (N, D), K: key (N, D), V: value (N, D)
# use_DropKey: whether use DropKey
# mask_ratio: ratio to mask

def Attention(Q, K, V, use_DropKey, mask_ratio)
    attn = (Q * (Q.shape[1] ** -0.5)) @ K.transpose(-2,
        -1)

    # use DropKey as regularizer
    if use_DropKey == True:
        m_r = torch.ones_like(attn) * mask_ratio
        attn = attn + torch.bernoulli(m_r) * -1e12

    attn = attn.softmax(dim=-1)
    x = attn @ V
    return x
```

To incorporate it into the framework of MTR, the DropKey is directly added to the multi-head attention module in the training step, while the DropKey is removed in the inference step. Note that the dropout of attention is removed when DropKey is available.

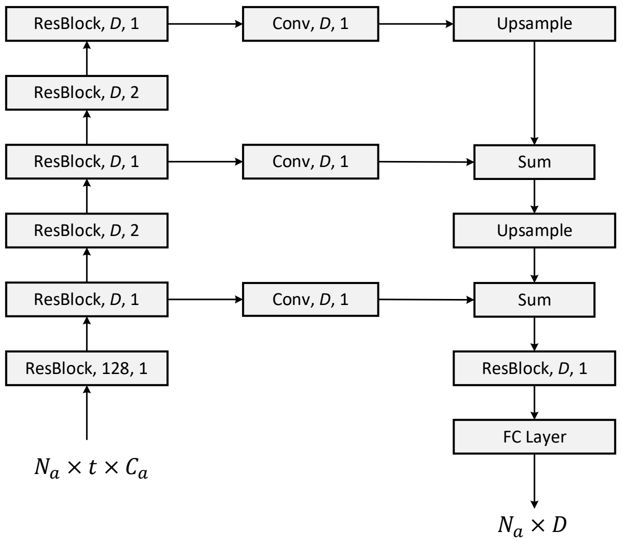

Figure 2. **The architecture of our modified Polyline Encoder (Inputting History state for illustration)**.

1.4. Similar-Mode Ensemble

In the ensemble step, we adopt our similar-mode ensemble strategy to integrate the trajectories and corresponding scores from different models as shown in Figure 3.

Given the model set $\{F_1, ..., F_M\}$, each model predicts $\mathcal{K}$ trajectories and corresponding scores. Note that each model uses the same $\mathcal{K}$ static intention points (*i.e.*, our proposed capacious query set). The trajectory $J_k^m \in \mathbb{R}^{T \times 2}$ and corresponding score $S_k^m \in \mathbb{R}^1$ is the k-th predicted trajectory of model F_m, where T is the length of future trajectory, $k \in \{1, ..., \mathcal{K}\}$, $m \in \{1, ..., M\}$. Since the $\mathcal{K}$ predicted trajectories correspond to the $\mathcal{K}$ static intention points one by one in MTR, we assume that the k-th predicted trajectories

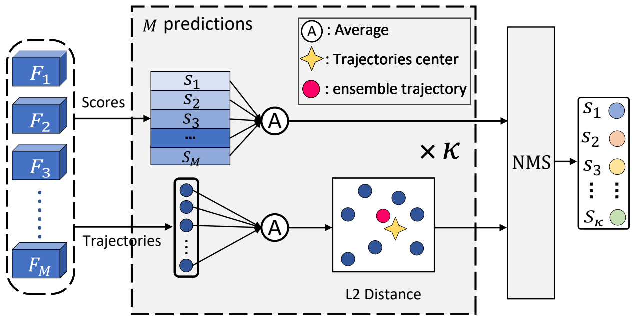

Figure 3. **Illustration of our similar-mode ensemble**.

$\{J_k^m\}^{m=\{1,...,M\}}$ have similar motion modes. Therefore, the predicted scores $\{S_k^m\}^{m=\{1,...,M\}}$ are the corresponding confidences.

Based on this assumption, we conduct the model ensemble in confidence ensemble and trajectory ensemble, respectively. For the confidence ensemble, we calculate the average score of M k-th trajectories as follows:

$$\hat{S}_k = \frac{\sum_{m=1}^{M} S_k^m}{M}, \tag{1}$$

where $\hat{S}_k$ is the confidence of the k-th predicted trajectories.

The trajectory ensemble is conducted on the goals (endpoint) of M k-th predicted trajectories. Thus, we generate M goals $\{G_k^1, ..., G_k^M\}$, which are first averaged as follows:

$$\bar{G}_k = \frac{\sum_{m=1}^{M} G_k^m}{M}, \tag{2}$$

where $\bar{G}_k$ is the averaged goal of the k-th trajectory. However, the average position can not ensure the $\bar{G}_k$ is the way-point that meets the scene content. Thus, we take the closest endpoint $\hat{G}_k$ with $\bar{G}_k$ as the final expected goal. The corresponding predicted trajectory $\hat{J}_k$ of $\hat{G}_k$ is the final k-th predicted trajectory.

For $\mathcal{K}$ predictions, we can obtain K trajectories $\{\hat{J}_1, ..., \hat{J}_K\}$ and corresponding confidences $\{\hat{S}_1, ..., \hat{S}_K\}$. Finally, a Non-Maximum Suppression (NMS) [4] is used to filter the invalid trajectories referring to MTR.

2. Experiments

2.1. Implementation Details

Settings. The number of cluster centers $\mathcal{K} = 64$ in our capacious query set. The history states have $t = 11$ frames with the dimension $C_a = 11$. We select $N_m = 768$ nearest map polylines around the interested agent. Each polyline of the road map has $n = 20$ points with the dimension $C_m = 2$. Both the dimension of agent features and map features D are set to 256. The "mask_ratio" of DropKey is set to 0.3. For other hyperparameters, we follow the setting of MTR. The detailed settings on MTR can be found in the https://github.com/sshaoshuai/MTR.

Ensemble. For the model ensemble, we use $M = 12$ models to obtain the final performance on the testing set. The model list is as follows:

1) MTR integrates the Capacious Query Set and the DropKey;

2) MTR integrates the Capacious Query Set;

3) MTR integrates the Capacious Query Set and the random mask;

4) MTR integrates the Capacious Query Set and the Modified Polyline Encoder;

5) MTR integrates the Capacious Query Set, the Modified Polyline Encoder, and 9 original Transformer Decoder layers;

6) MTR integrates the Capacious Query Set, the DropKey, and soft label supervision;

7-12) The corresponding finetune version of model 1) - 6) on the validation set.

The random mask is a data argumentation that randomly masks 15% input polylines. Add, the soft label supervision is replacing the one-hot label in Decoder Loss of MTR by the SOFTMAX of the negative distance between the goal and static intention point. The finetune versions are generated by finetuning the models on the validation set with 10 epochs and the learning rate is set to $6.25e-06$.

2.2. Experimental Results

Ablation Study. The experimental results of the ablation study on the validation set are shown in Table 1. Specifically, compared with the baseline method (MTR), the DropKey improves the mAP from 0.4202 to 0.4389 and the capacious query set improves the mAP from 0.4202 to .4494. What's more, the fully modified method achieves 0.4561 performance in terms of mAP. These experimental results demonstrate the effectiveness of our modified parts.

Results on Testing Set. As shown in Table 2, the IAIR+(ours) is our best submission, ranking $2nd$ in the Waymo Open Dataset Challenge 2023, Motion Prediction Track. The IAIR+(w/o finetune) is the first submitted result that does not finetune on the validation set. It shows a lower performance demonstrating the effectiveness of model finetune.

References

[1] Junru Gu, Qiao Sun, and Hang Zhao. Densetnt: Waymo open dataset motion prediction challenge 1st place solution. *arXiv preprint arXiv:2106.14160*, 2021. 1

[2] Bonan Li, Yinhan Hu, Xuecheng Nie, Congying Han, Xiangjian Jiang, Tiande Guo, and Luoqi Liu. Dropkey for vi-

model	Modified Encoder	Origin Encoder	DropKey	Capacious Query Set	random mask	mAP ↑	minADE ↓	minFDE ↓	MissRate ↓
MTR(baseline)	-	✓	-	-	-	0.4202	0.5938	1.2046	0.1314
-	-	✓	✓	-	-	0.4389	0.5934	1.2098	0.1340
-	-	✓	-	✓	-	0.4494	**0.5806**	**1.1869**	0.1306
-	-	✓	✓	✓	-	0.4439	0.5887	1.2075	0.1350
-	-	✓	-	✓	✓	0.4488	0.5964	1.2155	**0.1293**
Ours	✓	-	✓	✓	✓	**0.4561**	0.5871	1.1987	0.1333

Table 1. Ablation studies of our solution on the validation set

method	Soft mAP ↑	mAP ↑	minADE ↓	minFDE ↓	Miss Rate ↓	Overlap Rate ↓
MTR++_Ens	0.4738	0.4634	0.5581	1.1166	0.1122	0.1276
IAIR+(ours)	0.4480	0.4347	0.5783	1.1679	0.1238	0.1263
IAIR+(w/o finetune)	0.4440	0.4311	0.5781	1.1669	0.1248	0.1268
GTR-R36	0.4384	0.4255	0.6005	1.2225	0.1330	0.1279
GTR	0.4365	0.4230	0.5871	1.2096	0.1309	0.1271
DM	0.4362	0.4301	0.6293	1.2723	0.1473	0.1364

Table 2. Performance on the test leaderboard of motion prediction track of Waymo Open Dataset Challenge. The Soft mAP is the official ranking metric while the miss rate is the secondary ranking metric.

sion transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22700–22709, 2023. 2

[3] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 541–556. Springer, 2020. 2

[4] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *18th international conference on pattern recognition (ICPR'06)*, volume 3, pages 850–855. IEEE, 2006. 3

[5] Liushuai Shi, Le Wang, Chengjiang Long, Sanping Zhou, Fang Zheng, Nanning Zheng, and Gang Hua. Social interpretable tree for pedestrian trajectory prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2235–2243, 2022. 1

[6] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. *arXiv preprint arXiv:2209.13508*, 2022. 1

[7] Chenxin Xu, Weibo Mao, Wenjun Zhang, and Siheng Chen. Remember intentions: retrospective-memory-based trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6488–6497, 2022. 1

[8] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. In *Conference on Robot Learning*, pages 895–904. PMLR, 2021. 1