Supplementary Material of Real-Time Motion Prediction via Heterogeneous Polyline Transformer with Relative Pose Encoding

Anonymous Author(s)
Affiliation
Address
email

A Output representation and training strategies

For each anchor token $\hat{\mathbf{z}}_i^{\text{AG}}, i \in \{1, \dots, N_{\text{AG}} \cdot N_{\text{AC}}\}$, the confidence head predicts the logits $p_k, k \in \{1, \dots, 6\}$, whereas the trajectory head predicts 6 trajectories, each of which is represented as $(\mu_x^t, \mu_y^t, \log \sigma_x^t, \log \sigma_y^t, \rho^t, v_x^t, v_y^t, \theta^t, s^t), t \in \{1, \dots, T_f\}$, i.e. the mean of the Gaussian in x, y, the log standard deviation of the Gaussian in x, y, the correlation of the Gaussians, the velocity in x, y, the heading angle and the speed. We denote the ground truth as $(\hat{x}, \hat{y}, \hat{v}_x, \hat{v}_y, \hat{\theta}, \hat{s})$. The negative log-likelihood loss for position is formulated as

$$L_{\text{pos}} = -\log \mathcal{N}(\hat{x}, \hat{y} \mid \mu_x, \mu_y, \sigma_x, \sigma_y, \rho). \tag{1}$$

The negative cosine loss for the heading angle is formulated as

$$L_{\text{rot}} = -\cos(\hat{\theta} - \theta). \tag{2}$$

The Huber loss for velocities and speed is formulated as

$$L_{\text{vel}} = \mathcal{L}_\delta(\hat{v}_x - v_x) + \mathcal{L}_\delta(\hat{v}_y - v_y) + \mathcal{L}_\delta(\hat{s} - s), \tag{3}$$

where $\mathcal{L}_\delta$ is the Huber loss. We use $\delta = 1$ for all Huber losses. The final regression loss for a trajectory is the unweighted sum

$$L_{\text{traj}} = L_{\text{pos}} + L_{\text{rot}} + L_{\text{vel}}, \tag{4}$$

which is averaged over the future time steps where the ground truth is available. We use a hard assignment strategy, i.e. among the 6 predictions of each agent we select the one that is closest to the ground truth in terms of average displacement error and optimize only for that prediction. Denoting the index of this prediction as $\hat{k}$, we train the confidence head via the cross entropy loss by taking $\hat{k}$ as the ground truth:

$$L_{\text{conf}} = -\log \frac{\exp(p_{\hat{k}})}{\sum_{i=1}^{6} \exp(p_i)}. \tag{5}$$

The final training loss for the complete model is the unweighted sum

$$L = L_{\text{traj}} + L_{\text{conf}}. \tag{6}$$

Our output representation and training strategies are the same as prior works [2, 4, 5], except for the auxiliary losses on velocities, speeds and heading angles.

B Implementation details

B.1 Network architectures

We use ReLU activation and set the hidden dimension D to 256. Our KNARPE is implemented with multi-head attention with 4 heads. We use Transformer with pre-layer normalization [6] with a dropout rate of 0.1. The feed-forward hidden dimension of Transformers is set to 1024. The base frequency ω of the sinusoidal positional encoding is set to 1000. We train a single model for all types of agents, while each type of agent has its own anchors. The polyline-line level PointNet and MLPs have 3 layers, the intra-MP Transformer encoder has 6 layers, and the inter-class as well as the AC-to-all Transformer decoders, have 2 layers. Our HPTR has 15.2M trainable parameters in total. The same setup is used for both the WOMD dataset and the AV2 dataset.

In the following, we report the configuration of ablation models. The HPTR with diagonal attention has 6 layers of intra-MP, 3 layers of intra-TL and 3 layers of intra-AG Transformer. It has 15.4M trainable parameters. The HPTR with full attention has 6 layers of all-to-all and 6 layers of AC-to-all Transformer. It has 15.2M parameters. The HPTR with diagonal followed by full attention has 6 layers of intra-MP, 2 layers of intra-TL, 2 layers of intra-AG, 2 layers of all-to-all and 2 layers of AC-to-all Transformer. It has 15.4M trainable parameters. Both the HPTR with full attention and the HPTR with diagonal followed by full attention have to be trained on GPUs with 24GB of VRAM (RTX 3090 in our case) because they require more GPU memory at training time. The scene-centric baseline uses the scene-centric representation and the standard Transformer. Following SceneTransformer [3], the input 2D positions and 2D directions are pre-processed using sinusoidal positional encoding. The base frequency is set to 1000 for 2D positions and 10 for 2D directions. The output dimension of the positional encoding is 256. This model has 13M trainable parameters. The agent-centric baseline closely follows Wayformer [2]. It has 6 layers of all-to-all Transformer and 8 layers of AC-to-all Transformer. The number of latent queries is 192. The learning rate starts at 2e-4 and it is multiplied by 0.5 every 20 epochs. The training of the agent-centric baseline takes 100 epochs to converge. We do not use auxiliary losses on velocities, speeds and yaw angles to train this model. This model has 15.6M trainable parameters.

B.2 Pre-processing and post-processing

Our pre-processing and post-processing closely follow MPA [1]. The post-processing manipulates only the confidences via greedy non-maximum suppression. The distance threshold is 2.5m for vehicles, 1m for pedestrians and 1.5m for cyclists. We use the average displacement error to compute the distance between predicted trajectories. For AV2 we simply use softmax with a temperature of 0.5 instead of doing non-maximum suppression.

B.3 Training details

Due to the large size of motion prediction datasets, each epoch would take a very long time if trained on the complete training split. In order to track losses more frequently, we randomly sample a fraction of all training data in each epoch. This is equivalent to using the complete training dataset if the training runs for many epochs. We observe a statistically significant correlation between the model performance and the initialization of anchors. We recommend to use a large variance for the initialization distributions. Specifically, we use Xavier initialization and multiply the initialized values by 5.

C Additional ablation studies

In Table 1 we ablate different ways to incorporate RPE into the dot-product attention. The differences are insignificant in terms of performance. However, our approach, i.e. adding projected RPE to projected key and value, consumes less memory at training time. We use this setup in our main paper because it can be trained on the RTX 2080 Ti GPUs (12GB VRAM), which are more accessible than the RTX 3090 GPUs (24GB VRAM) in practice.

D Additional results

In Tables 2, 3, and 4, we provide the complete results of our HPTR on the WOMD *test* split, the WOMD *valid* split, and the AV2 *valid* split, respectively.

In Figures 1, 2, and 3, we provide more qualitative results on WOMD *valid* of our HPTR predicting vehicles, pedestrians, and cyclists, respectively.

Table 1: Ablation on WOMD valid split. We study different ways to incorporate the RPE into the dot-product attention. Performance is reported as the mean plus-minus 3 standard deviations over 3 training seeds. Models are trained for 60 epochs. OOM: out of memory. q: query. k: key. v: value.

model description	concat RPE	query RPE	train on 2080ti	mem% on 3090	Min FDE ↓	Soft mAP ↑
ours (add proj. RPE to proj. k, v)	×	×	✓	58.2	1.143 ± 0.039	0.401 ± 0.007
ours without q, k, v bias	×	×	✓	58.2	1.140 ± 0.021	0.396 ± 0.009
add proj. RPE to proj. q, k, v	×	✓	OOM	66.2	1.144 ± 0.036	0.397 ± 0.006
concat. RPE to k, v	✓	×	OOM	71.6	1.138 ± 0.026	0.395 ± 0.006
concat. RPE to q, k, v	✓	✓	OOM	90.5	1.133 ± 0.024	0.396 ± 0.006

Table 2: Complete results of our HPTR on the WOMD *test* split.

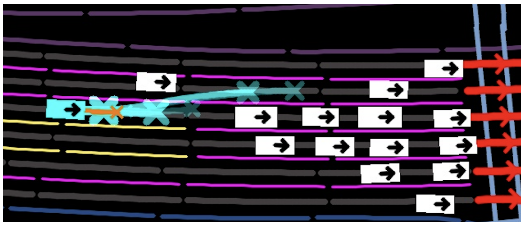
Object Type	Measurement Time (s)	Soft mAP ↑	mAP ↑	Min ADE ↓	Min FDE ↓	Miss Rate ↓	Overlap Rate ↓
Vehicle	3	0.5631	0.5475	0.2795	0.4997	0.0927	0.0190
Vehicle	5	0.4687	0.4623	0.5714	1.1020	0.1297	0.0415
Vehicle	8	0.3697	0.3664	1.0739	2.2753	0.1787	0.0915
Vehicle	Avg	0.4671	0.4587	0.6416	1.2923	0.1337	0.0507
Pedestrian	3	0.4534	0.4427	0.1637	0.3111	0.0676	0.2408
Pedestrian	5	0.3422	0.3370	0.3220	0.6616	0.0938	0.2648
Pedestrian	8	0.2792	0.2751	0.5722	1.2778	0.1248	0.2952
Pedestrian	Avg	0.3582	0.3516	0.3526	0.7502	0.0954	0.2669
Cyclist	3	0.4334	0.4267	0.3266	0.6078	0.1859	0.0494
Cyclist	5	0.3587	0.3552	0.6166	1.2085	0.1922	0.0900
Cyclist	8	0.3025	0.3006	1.0825	2.3096	0.2250	0.1369
Cyclist	Avg	0.3649	0.3608	0.6752	1.3753	0.2011	0.0921
Avg	3	0.4833	0.4723	0.2566	0.4729	0.1154	0.1030
Avg	5	0.3899	0.3848	0.5033	0.9907	0.1386	0.1321
Avg	8	0.3171	0.3140	0.9095	1.9543	0.1762	0.1745
Avg	Avg	0.3968	0.3904	0.5565	1.1393	0.1434	0.1366

Table 3: Complete results of our HPTR on the WOMD *valid* split.

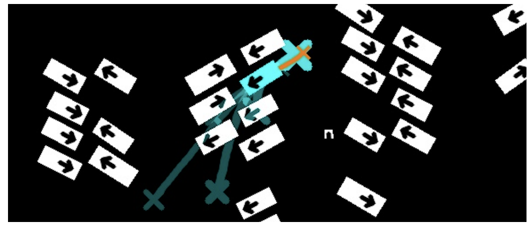
Object Type	Measurement Time (s)	Soft mAP ↑	mAP ↑	Min ADE ↓	Min FDE ↓	Miss Rate ↓	Overlap Rate ↓
Vehicle	3	0.5611	0.5451	0.2796	0.4988	0.0934	0.0186
Vehicle	5	0.4704	0.4637	0.5698	1.0986	0.1297	0.0405
Vehicle	8	0.3678	0.3644	1.0731	2.2909	0.1824	0.0909
Vehicle	Avg	0.4664	0.4577	0.6408	1.2961	0.1352	0.0500
Pedestrian	3	0.4923	0.4802	0.1454	0.2674	0.0478	0.2358
Pedestrian	5	0.4055	0.3993	0.2782	0.5488	0.0661	0.2605
Pedestrian	8	0.3639	0.3577	0.4834	1.0157	0.0813	0.2901
Pedestrian	Avg	0.4206	0.4124	0.3023	0.6106	0.0651	0.2621
Cyclist	3	0.4606	0.4519	0.3309	0.6021	0.1779	0.0523
Cyclist	5	0.3822	0.3788	0.6136	1.1843	0.1905	0.0897
Cyclist	8	0.2962	0.2943	1.0661	2.3242	0.2239	0.1433
Cyclist	Avg	0.3797	0.3750	0.6702	1.3702	0.1974	0.0951
Avg	3	0.5047	0.4924	0.2519	0.4561	0.1064	0.1022
Avg	5	0.4194	0.4139	0.4872	0.9439	0.1288	0.1302
Avg	8	0.3427	0.3388	0.8742	1.8769	0.1626	0.1748
Avg	Avg	0.4222	0.4150	0.5378	1.0923	0.1326	0.1357

Table 4: Complete results of our HPTR on the AV2 *test* split.

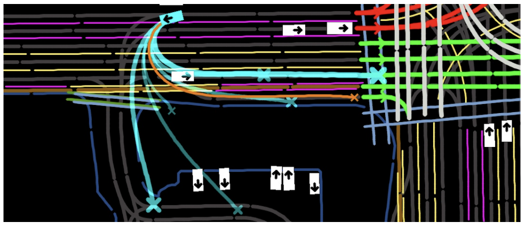
$\text{minFDE}_6\downarrow$	$\text{minFDE}_1\downarrow$	$\text{minADE}_6\downarrow$	$\text{minADE}_1\downarrow$	Miss $\text{Rate}_6\downarrow$	Miss $\text{Rate}_1\downarrow$	brier-$\text{minFDE}_6\downarrow$
1.43	4.61	0.73	1.84	0.19	0.61	2.03



(a) Change lane.



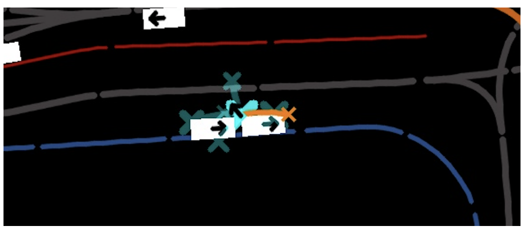
(b) Reverse.



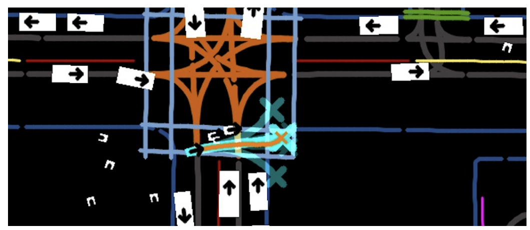
(c) U-Turn.

(d) Multi-modal futures.

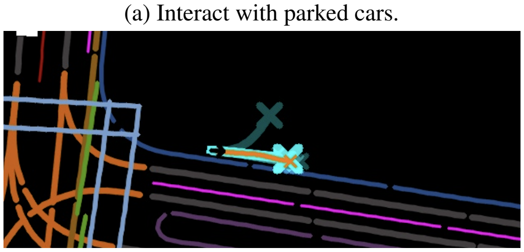
Figure 1: Qualitative results of HPTR predicting **vehicles**. Scenarios are selected from the WOMD validation dataset. The ground truth is in orange. The target agent and the predictions are in cyan. The most confident prediction has the least transparent color, the thickest line and the biggest cross.



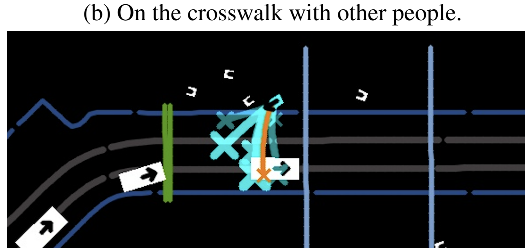
(a) Interact with parked cars.



(b) On the crosswalk with other people.

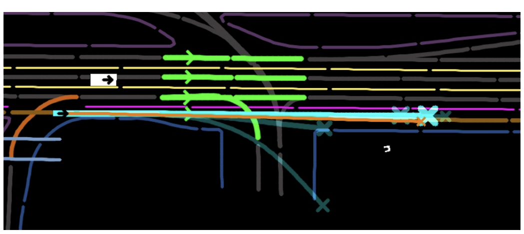


(c) Walk alone on the road edge.

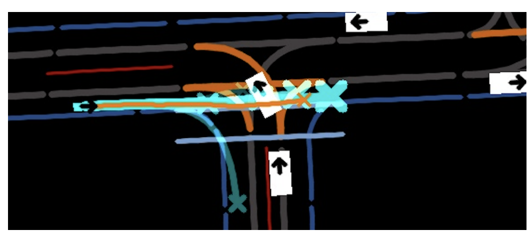


(d) Cross the street without using crosswalk.

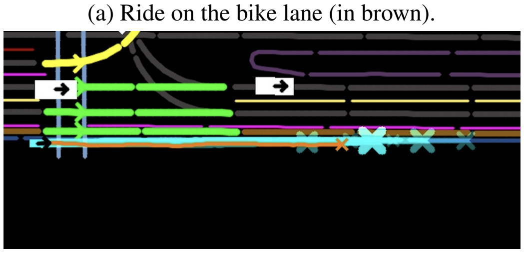
Figure 2: Qualitative results of HPTR predicting **pedestrians**. Read as Figure 1.



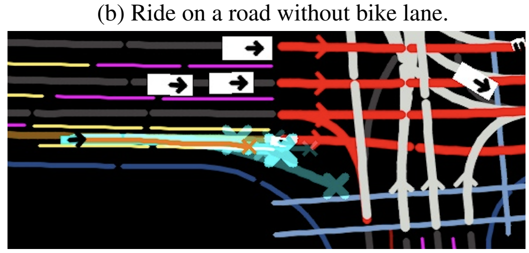
(a) Ride on the bike lane (in brown).



(b) Ride on a road without bike lane.



(c) Ride on the road edge.



(d) Stop at red light.

Figure 3: Qualitative results of HPTR predicting **cyclists**. Read as Figure 1.

References

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

[2] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. In *ICRA*, 2023.

[3] 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*, 2021.

[4] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. In *NeurIPS*, 2022.

[5] 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.

[6] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *ICML*, 2020.