Supplementary Material: Query-Centric Trajectory Prediction

Zikang Zhou[1,2] Jianping Wang[1,2] Yung-Hui Li[3] Yu-Kai Huang[4]
[1]City University of Hong Kong [2]City University of Hong Kong Shenzhen Research Institute
[3]Hon Hai Research Institute [4]Carnegie Mellon University

1. Implementation Details

1.1. Architecture

The hidden feature dimension is 128. All layers for information fusion have the same architecture, which is similar to the gated variant of attention mechanism used in HiVT [9]. The number of heads in the multi-head attention operator is 8. Layer normalization [1] is used in MLPs and attention layers. Our default configuration uses 3 recurrent steps in the trajectory proposal module and 2 blocks of multi-context attention in both the encoder and the decoder. Since the Waymo Open Motion Dataset (WOMD) requires models to predict 8-second future trajectories, we use 4 recurrent steps for better long-term performance. Unlike other DETR-like motion decoders which first generate a large number of trajectory candidates and then produce the final top-K predictions via heuristic post processing such as non-maximum suppression, our decoder only uses K mode queries for much higher efficiency. The model size for Argoverse 1, Argoverse 2, and WOMD are 7.2M, 7.3M, and 7.5M, respectively.

1.2. Training

We adopt the AdamW optimizer [4] to train models in an end-to-end manner for 64 epochs with a batch size of 32, a dropout rate of 0.1, and a weight decay coefficient of 1×10^{-4}. Due to the larger scale of WOMD, we halve the number of training epochs when doing experiments on this dataset. The hyperparameter λ is set to 0.1, 1.0, and 1.0 on Argoverse 1, Argoverse 2, and WOMD, respectively. The learning rate is initialized to 5×10^{-4} and is decayed using the cosine annealing scheduler [3]. Since the query-centric design has already brought many invariance properties into the model, we do not use data augmentation during training.

1.3. Ensembling

As the recent state of the art has employed heavy model ensembling to boost their performance, we additionally report the ensembling results of 5 models on Argoverse 1 and Argoverse 2 for more fair comparisons. These models are trained using different random seeds, and we use the weighted k-means algorithm to aggregate their forecasted trajectories. Specifically, we use the k-means implementation in scikit-learn [6] with all predicted trajectories' endpoints as input and the predicted classification scores as sample weights. After cluster assignment, the trajectories within each cluster are averaged.

2. Discussion on Inference Latency

Most state-of-the-art trajectory prediction models (*e.g.*, Wayformer [5] and MTR [7]) are tailored for single-agent prediction, suffering from redundant scene encoding when they are required to make predictions for multiple agents. In contrast, the query-centric design of QCNet naturally introduces roto-translation invariance in the space dimension, enabling multi-agent prediction in a single forward pass. Moreover, thanks to the additional translation invariance in the time dimension, QCNet can avoid re-normalizing and re-encoding agent states and map elements that have appeared in previous observation windows during online inference. As a result, the complexity of our encoder is an order less than existing factorized attention-based approaches.

We measure the end-to-end inference latency of our model (*i.e.*, the combination of the scene encoder, the proposal module, and the refinement module) on the Argoverse 2 validation set using an NVIDIA A40 GPU. On average, a traffic scene contains about 40 agents, and the inference latency of predicting all agents' future trajectories is 46 ms. For the densest traffic scene involving 190 agents, 169 map polygons, 50 past time steps, and 60 future time steps, our model can make predictions for all agents with the latency of 72 ms. In practice, we only need to consider the future of agents that may have impact on the safety of autonomous vehicles, so the inference latency can be lower.

3. Multi-Agent Quantitative Results

As discussed in Sec. 2, QCNet treats all agents in the scene symmetrically, enabling multi-agent prediction in a

Model	b-minFDE$_6$↓	minADE$_6$↓	minFDE$_6$↓	MR$_6$↓
GoRela	1.29	0.42	0.96	0.14
QCNet	**1.26**	**0.38**	**0.62**	**0.07**

Table 6. Marginal multi-agent prediction results on the Argoverse 2 validation set. For each scene, we evaluate the performance of predicting the "focal agent" together with one or more "scored agents" marked by the dataset.

Model	minADE$_6$↓	minFDE$_6$↓	MR$_6$↓
Scene Transformer	0.6117	1.2116	0.1564
HDGT	0.5703	1.1434	0.1440
DenseTNT	1.0387	1.5514	0.1573
MultiPath++*	0.5557	1.1577	0.1340
Wayformer*	0.5454	1.1280	0.1228
MTRA*	0.5640	1.1344	**0.1160**
QCNet (w/o ensemble)	**0.5345**	**1.0749**	0.1345

Table 7. Marginal multi-agent prediction results on the WOMD test set. Baselines that are known to have used ensembling are marked with symbol "*".

single forward pass without sacrificing the performance. We report the marginal multi-agent prediction results on the Argoverse 2 validation set in Tab. 6. Compared with the recent arXiv report GoRela [2], QCNet offers much stronger multi-agent prediction performance. We further evaluate our model's capability of marginal multi-agent prediction on WOMD. As shown in Tab. 7, QCNet significantly outperforms the best ensembling results on the test set of WOMD in terms of minADE$_6$ and minFDE$_6$, ranking 1st in terms of these commonly used distance-based metrics. Note that our amazing results are obtained from a single model without relying on any ensemble methods or time-consuming post processing. Using ensembling techniques similar to other entries may further boost the performance.

4. Per-Category Quantitative Results

In Tab. 8, we report the per-category quantitative results on the Argoverse 2 validation set. Compared with QML [8], the second place solution in the 2022 Argoverse 2 motion forecasting competition, our approach exhibits better performance on all categories. We also observe that the prediction result of motorcyclists is much worse than that of other categories, presumably because the motions of motorcycles are more flexible and the corresponding training samples are insufficient. A similar phenomenon can be observed on WOMD, where the trajectories of cyclists are much harder to be predicted (see Tab. 9).

Model	Category	b-minFDE$_6$↓	minADE$_6$↓	minFDE$_6$↓	MR$_6$↓
QML	Vehicle	2.13	0.83	1.45	–
	Pedestrian	**1.29**	0.35	**0.63**	–
	Motorcyclist	2.47	1.00	**1.80**	–
	Cyclist	1.90	0.66	1.23	–
	Bus	2.15	1.03	1.46	–
QCNet	Vehicle	**1.95**	**0.75**	**1.32**	0.17
	Pedestrian	**1.29**	**0.34**	**0.63**	0.05
	Motorcyclist	**2.46**	**0.90**	1.81	0.20
	Cyclist	**1.83**	**0.63**	**1.19**	0.20
	Bus	**1.88**	**0.95**	**1.25**	0.16

Table 8. Per-category prediction performance on the Argoverse 2 validation set.

Dataset	Category	minADE$_6$↓	minFDE$_6$↓	MR$_6$↓
WOMD Val Set	Vehicle	0.6276	1.2522	0.1270
	Pedestrian	0.3112	0.6441	0.0804
	Cyclist	0.6626	1.3433	0.2013
	Avg	0.5338	1.0799	0.1362
WOMD Test Set	Vehicle	0.6268	1.2478	0.1258
	Pedestrian	0.3128	0.6432	0.0783
	Cyclist	0.6638	1.3335	0.1994
	Avg	0.5345	1.0749	0.1345

Table 9. Per-category prediction performance on the validation set and the test set of WOMD.

Model	#Param	minADE$_6$↓	minFDE$_6$↓	MR$_6$↓
QCNet ($L_{\text{enc}} = L_{\text{dec}} = 1$)	5.0M	0.5472	1.1096	0.1427
QCNet ($L_{\text{enc}} = L_{\text{dec}} = 2$)	7.5M	0.5367	1.0881	0.1378
QCNet ($L_{\text{enc}} = L_{\text{dec}} = 3$)	10.0M	**0.5341**	**1.0816**	**0.1372**

Table 10. Effects of the number of the layers on the WOMD validation set.

5. Ablation Study on WOMD

5.1. Effects of the Number of Layers

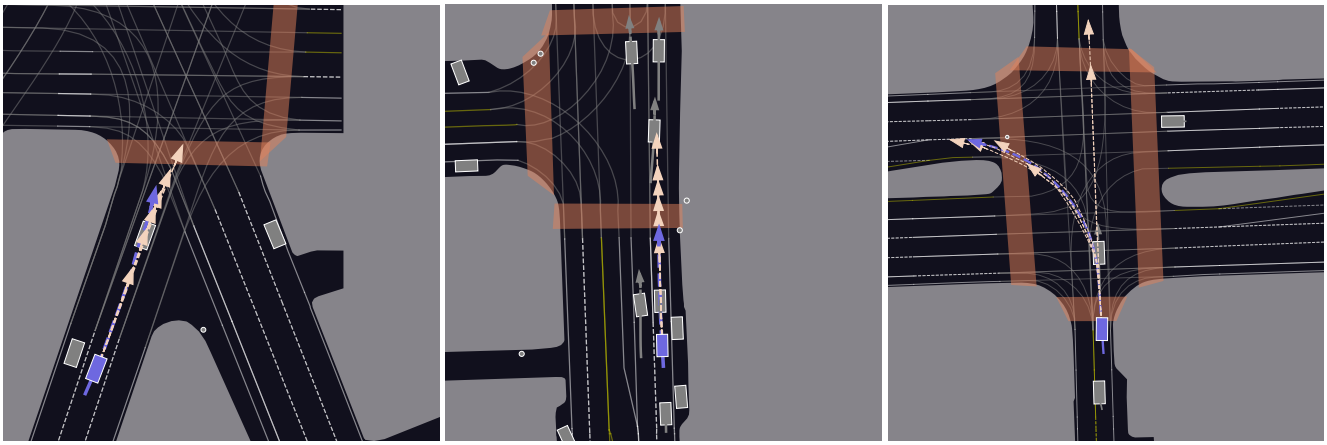
As shown in Tab. 10, increasing L_{enc} and L_{dec} from 1 to 3 can improve the prediction performance on WOMD, but our main results reported in Tab. 7 and Tab. 9 are based on the two-layer configuration with 7.5M model parameters in total.

5.2. Effects of the Number of Recurrent Steps

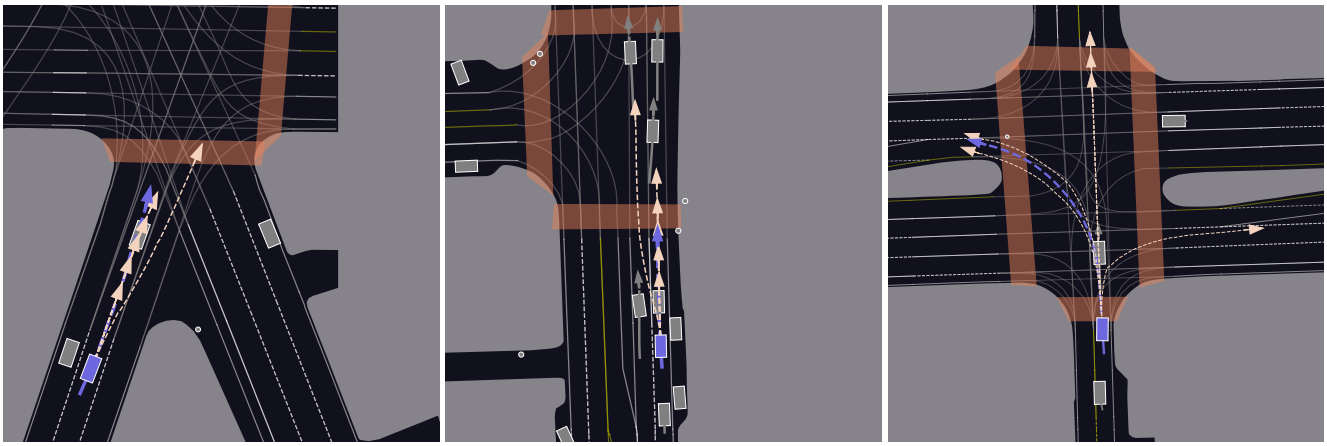
We try to vary the number of recurrent steps in the trajectory proposal module. Table 11 shows that using 4 recurrent steps (*i.e.*, decoding two-second trajectories at each recurrence) can achieve better results on WOMD.

6. Analysis of Qualitative Results

In this section, we give more detailed analysis of the qualitative results on the Argoverse 2 validation set. In



(a) QCNet w/o refinement



(b) QCNet w/ refinement

Figure 5. Qualitative results on the Argoverse 2 validation set. Although two model variants almost perform equally well on the commonly used metrics when tested in these scenarios, we find that adding the refinement module can improve the multimodality of the prediction.

#Recurrent Step	$\text{minADE}_6\downarrow$	$\text{minFDE}_6\downarrow$	$\text{MR}_6\downarrow$
2 (4 sec/step)	0.5392	1.0953	0.1398
4 (2 sec/step)	**0.5367**	**1.0881**	**0.1378**

Table 11. Effects of the number of recurrent steps on the WOMD validation set.

Sec. 6.1, we demonstrate that the commonly used metrics for trajectory prediction, such as minADE_6 and minFDE_6, cannot always reflect models' actual performance objectively. In Sec. 6.2, we show some common failure cases where QCNet cannot offer reliable performance.

6.1. Beyond Metrics

In the examples shown in Fig. 5, two model variants almost perform equally well on the commonly used metrics for trajectory prediction. Without the refinement module, our model has already been able to cover the ground truth (see Fig. 5a). However, it does not mean that the refinement module has no advantages. In the fist two examples of Fig. 5b, a vehicle stops in front of the target agent. With the help of the refinement module, our model not only successfully covers the ground truth but also reasonably predicts the potential lane change behavior. In the third example where the target agent is at an intersection, the refinement module enables our model to maximally cover all possibilities, including going straight, left turn, and right turn. All these cases demonstrate that the refinement module can significantly enhance the multimodality, while the commonly used metrics cannot reflect its merits. Thus, we believe that the trajectory prediction community should consider more reasonable metrics in the future.

6.2. Failure Cases

Knowing when a data-driven model may fail is important for safety-critical applications such as autonomous driving. Although our approach has achieved outstanding per-

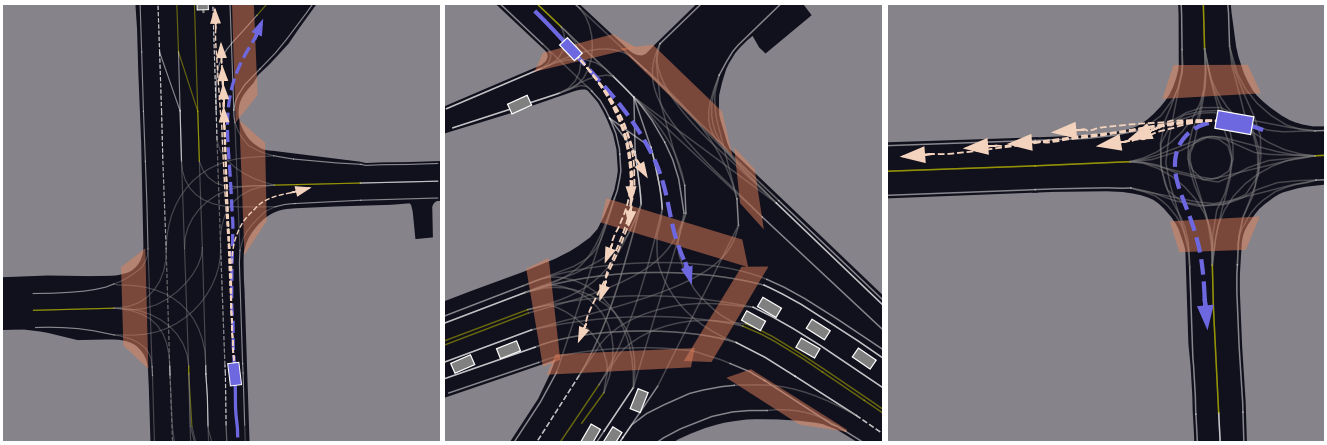


Figure 6. Failure cases on the Argoverse 2 validation set, where the geometry and topology of the map are extremely complex.

formance on motion forecasting benchmarks, it is unclear whether the prediction model can enable safe motion planning, given that failure cases still exist. In this section, we provide some qualitative results to help readers understand when our model may fail. Hopefully, these failure cases provide some insights on what future work can be done to make a trajectory predictor more powerful and more robust.

6.2.1 Failing to Understand Complex Map Geometry and Topology

While the map encoder and the agent-map fusion layers have incorporated map information for modeling, we find that failure cases still exist in scenarios where the map geometry and topology are complex. Some of these cases are shown in Fig. 6. In the left example, our model successfully predicts the potential right turn at the first junction but misses the turn at the second one. In the middle example, multiple lanes intertwine at the intersection, making it difficult for the model to understand the connectivity of the lanes. For this reason, our model misses the target agent's intention. In the right example, the model ignores the left turn behavior due to the unusual lane curvature at the roundabout. We believe that more powerful map encoder can better handle these challenging scenarios.

6.2.2 Failing to Predict Lane Change Behavior

Lane change is a common driving behavior in daily traffic scenarios. Although missing a lane change only leads to relatively small prediction error, the ability of predicting such behavior has significant impact on the safety of autonomous vehicles. Figure 7 shows some cases where our model fails to capture the lane change intention of agents. In the left example, the target agent crosses multiple lanes during the turn, while the predictions of our model do not cover all candidate lanes. In the middle example, a vehicle stops in front of the target agent. Although our model has reasonably predicted the intentions of deceleration and right lane change, it ignores the possibility of left lane change. In the right example, the target agent changes a lane when crossing the intersection. However, from the visualization we cannot observe any cue that can help the prediction. We attribute such failure cases to the partial observability of the environment. Future work on end-to-end perception and prediction may be able to capture more information (*e.g.*, the flashing lights of vehicles) and mitigate this problem.

6.2.3 Failing to Predict U-Turn Behavior

We note that driving data are highly imbalanced: in most scenarios, agents only exhibit trivial behavior such as going straight with nearly constant velocity. For this reason, data-driven models struggle to handle those rare cases. A typical example is the U-turn behavior. As shown in Fig. 8, our model fails to capture the U-turn behavior when the target agent is almost static over the observation window. To deal with such cases, we may need to balance the data distribution.

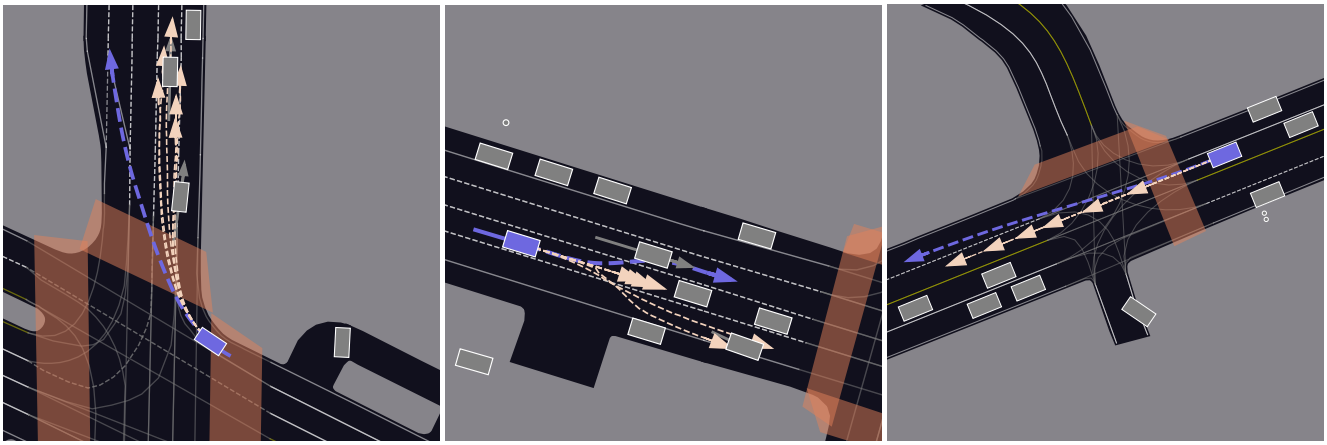


Figure 7. Lane change cases on the Argoverse 2 validation set.

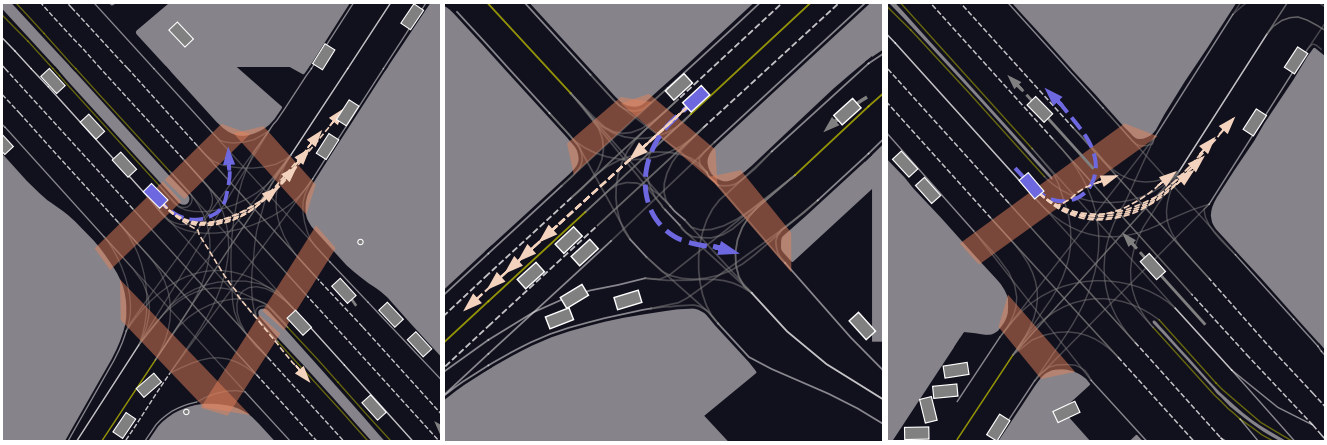


Figure 8. U-turn cases on the Argoverse 2 validation set.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 1

[2] Alexander Cui, Sergio Casas, Kelvin Wong, Simon Suo, and Raquel Urtasun. Gorela: Go relative for viewpoint-invariant motion forecasting. *arXiv preprint arXiv:2211.02545*, 2022. 2

[3] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 1

[4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 1

[5] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. *arXiv preprint arXiv:2207.05844*, 2022. 1

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research (JMLR)*, 2011. 1

[7] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 1

[8] Tong Su, Xishun Wang, and Xiaodong Yang. Qml for argoverse 2 motion forecasting challenge. *arXiv preprint arXiv:2207.06553*, 2022. 2

[9] Zikang Zhou, Luyao Ye, Jianping Wang, Kui Wu, and Kejie Lu. Hivt: Hierarchical vector transformer for multi-agent motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1