# A EXPERIMENT

## A.1 EXPERIMENT SETTINGS

In this paper, we employ a learning rate of $6 \times 10^{-4}$ and a weight decay rate of 0.01. The experiments involve 100 instance queries ($N_I = 100$) and 20 point queries ($n_p = 20$), conducted using eight NVIDIA GeForce RTX 3090 GPUs, each equipped with 24GB of memory. For the ResNet50 backbone, we set the batch size to 4, while for the ResNet18 backbone, we use a batch size of 32. Our primary focus is models that utilize camera images as input, specifically six RGB camera images. But InsightMapper can be easily adapted for multi-modal perception by modifying the input of BEVformer Li et al. (2022b).

## A.2 ROAD ELEMENTS

InsightMapper is designed to detect four types of road elements essential for vectorized HD maps: road boundaries (polyline), lane splits (polyline), pedestrian crossings (polygon), and lane centerlines (polyline with topology). At this stage, the detected HD map is an undirected graph. If directed vectorized HD maps are required for specific applications, InsightMapper can be easily adapted by employing directed ground truth HD maps as labels to train the network.

## A.3 TOPOLOGY EVALUATION METRIC

To provide a comprehensive evaluation, we report a topology-level evaluation score in this paper, namely the TOPO metric score He et al. (2018; 2020); He & Balakrishnan (2022). The TOPO metric has been used in several previous studies He et al. (2018; 2020); He & Balakrishnan (2022) to evaluate the correctness of lane and road-network graph topology. The TOPO metric first randomly samples vertices $v_i^*$ from the ground truth graph $G^*$. It then finds corresponding matched vertices $\hat{v}_i$ in the predicted graph $\hat{G}$ based on the closest distance. Using $v_i^*$ as a seed node, TOPO calculates a sub-graph $G^*_{v_i^*}$, where the distance between all vertices and $v_i^*$ is smaller than a predetermined threshold. Similarly, we obtain the sub-graph $\hat{G}_{\hat{v}_i}$ by taking $\hat{v}_i$ as the seed node. Finally, we measure the graph similarity of the two sub-graphs using precision, recall, and F1-score. The TOPO metric is the mean similarity F1-score of all sampled vertex pairs $(v_i^*, \hat{v}_i)$.

# B QUERY GENERATION

## B.1 DENOSING DETR (ABANDONED DESIGN)

In conventional object detection tasks, the denoising operation has been proven to effectively accelerate convergence and enhance overall performance Li et al. (2022a); Zhang et al. (2022); Li et al. (2023a). However, this operation requires queries to be independent and identically distributed (i.i.d.). If this condition is not met, simply adding random noise to queries may not yield superior results. In our task, due to the strong correlation among inner-instance points, the denoising operation does not improve the vectorized HD map detection results. We report the outcomes of applying Dn-DETR in Table 6.

Table 6: Quantitative results of ablation studies about denoising DETR. For all metrics, a larger value indicates better performance.

| Method | $AP_{ped}$ | $AP_{div}$ | $AP_{bound}$ | $AP_{center}$ | mAP | TOPO |
|---|---|---|---|---|---|---|
| InsightMapper-Dn-DETR | 43.74 | 51.34 | 54.48 | 42.33 | 47.97(↓0.34) | 46.90(↓0.19) |
| InsightMapper | 44.36 | 53.36 | 52.77 | 42.35 | 48.31 | 47.09 |

In our experiments, we initially add random instance-level noise equally to all points within the same instance. Subsequently, we introduce random point-level noise to each point. Despite these modifications, we observe neither a significant improvement nor faster convergence. We attribute this unsatisfactory performance to the correlation between points.

### B.2 DYNAMIC QUERY GENERATION (ABANDONED DESIGN)

Another approach to improve DETR in the query generation phase is dynamic query generation Wang et al. (2022); Zhang et al. (2022); He et al. (2022). Unlike static query generation, where all queries are randomly initialized, dynamic queries are predicted based on the features obtained by the transformer encoder. In other words, the output of the transformer encoder can be utilized to generate queries with better initialization.

After obtaining the transformer encoder output $F$, we create grids to partition $F$. Each grid $F_{x,y}$ contains the local information of the input image around coordinate $(x, y)$. Then, each grid is used to predict a dynamic query, represented as a 4-D bounding box. In conventional detection tasks, objects are typically not very large, so each grid can make satisfactory predictions of dynamic queries. However, in our vectorized HD map detection task, target objects are often very thin and very long, which cannot be effectively predicted by local grids. Consequently, dynamic queries do not yield significant improvements. We present the results on dynamic query generation in Table 7.

Table 7: Quantitative results of ablation studies about dynamic queries.

| Method | $AP_{ped}$ | $AP_{div}$ | $AP_{bound}$ | $AP_{center}$ | mAP | TOPO |
|---|---|---|---|---|---|---|
| InsightMapper with dynamic query | 43.14 | 48.86 | 52.17 | 38.65 | 45.71(↓2.60) | 44.38(↓2.71) |
| InsightMapper | 44.36 | 53.36 | 52.77 | 42.35 | 48.31 | 47.09 |

## C INNER-INSTANCE FEATURE AGGREGATION

### C.1 RATIO OF MASKED ATTENTION MASK

There is an $\epsilon$ possibility that the attention mask of the inner-instance self-attention is masked (set to one, blocked). This mask operation is inspired by He et al. (2022) to enhance the robustness of the proposed module. $\epsilon$ controls the ratio of masked attention masks. This concept is similar to dropout for better performance. However, dropout is applied after the *softmax* of attention, while the mask operation is before the *softmax*. Experimental results with different $\epsilon$ values are shown in Table 8.

Table 8: Quantitative results of ablation studies about the mask ratio.

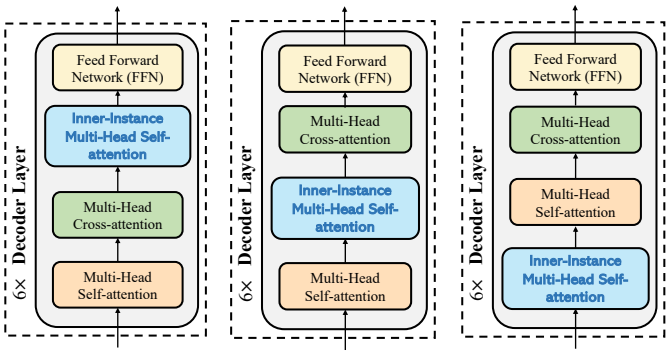
| $\epsilon$ | $AP_{ped}$ | $AP_{div}$ | $AP_{bound}$ | $AP_{center}$ | mAP | TOPO |
|---|---|---|---|---|---|---|
| 0 (No masked attn) | 42.48 | 50.66 | 53.22 | 41.54 | 46.98(↓1.33) | 45.60(↓1.49) |
| 25% (InsightMapper) | 44.36 | 53.36 | 52.77 | 42.35 | 48.31 | 47.09 |
| 50% | 43.67 | 52.57 | 53.98 | 42.95 | 48.17(↓0.14) | 47.10(↑0.01) |
| 80% | 42.27 | 52.49 | 53.64 | 41.34 | 47.41(↓0.90) | 46.21(↓0.88) |

### C.2 POSITION OF THE MODULE

For the proposed inner-instance feature aggregation self-attention module, it can be placed at multiple positions within the transformer decoder layers, as visualized in Figure 7. Position A is used in the final design of InsightMapper. Placing the inner-instance self-attention module at either position B or position C may affect the cross-attention. Cross-attention needs to treat all queries equally to avoid duplicated predictions, while the proposed module pays more attention to inner-instance points, which interferes with the cross-attention and leads to degraded results. The evaluation results are reported in Table 9.

## D INTER-INSTANCE SELF-ATTENTION (ABANDONED DESIGN)

We also attempted to exploit the inter-instance information to further enhance the vectorized HD map detection results. The inter-instance self-attention module is incorporated into the decoder layers, similar to the inner-instance self-attention module, as shown in Figure 8.




(a) Position A (b) Position B (c) Position C

Figure 7: Inner-instance self-attention module at different positions in the transformer decoder. Position A is used in the final design of InsightMapper. Position B places the module right before the cross-attention module, which interferes with the cross-attention and degrades the final performance. Placing the module at position C is similar to that of position B, which affects the cross-attention and yields inferior results.

Table 9: Quantitative results of ablation studies about the position of the inner-instance self-attention module.

| Position | $AP_{ped}$ | $AP_{div}$ | $AP_{bound}$ | $AP_{center}$ | mAP | TOPO |
|---|---|---|---|---|---|---|
| *A* (InsightMapper) | 44.36 | 53.36 | 52.77 | 42.35 | 48.31 | 47.09 |
| *B* | 39.09 | 50.09 | 51.56 | 40.21 | 45.24(↓3.07) | 43.85(↓3.24) |
| *C* | 44.04 | 49.59 | 52.18 | 41.78 | 46.90(↓1.41) | 45.31(↓1.78) |

First, we predict the adjacency matrix, representing the connectivity of the predicted HD map. Some lane centerline instances may intersect with each other. Adjacent instances are illustrated by colored grids. Then, for adjacent instances, the corresponding attention mask grids are set to zero (attention is allowed). Otherwise, the attention is blocked. In this way, the information exchange between points in adjacent instances is allowed to better leverage the point correlations. However, this design significantly increases resource consumption while not noticeably improving the final results. The experiment results are shown in Table 10.

We believe the reason is the sparsity of the adjacency matrix. Under most circumstances, only a few instances intersect with each other, so the adjacency matrix is very sparse, providing limited additional inter-instance information. Furthermore, it may affect the inner-instance self-attention module, which is the main reason for the performance gains of InsightMapper.

Therefore, at this stage, inter-instance self-attention is not used in InsightMapper. But this could be an interesting topic for future exploration.

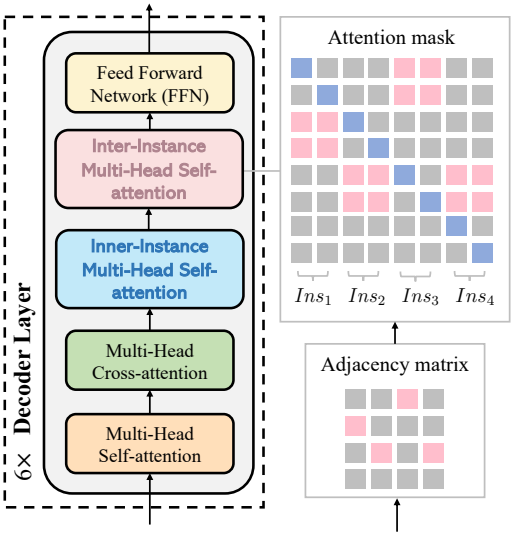



Figure 8: Decoder of InsightMapper with inter-instance self-attention module. In this module (the pink one), the attention between points of non-adjacent instances is blocked (grey grids). The attention is allowed for points of adjacent instances (pink grids), and diagonal grids (blue grids) to maintain the ego information of each point.

Table 10: Quantitative results of ablation studies about inter-instance self-attention.

| Inter-instance Self-attn | $AP_{ped}$ | $AP_{div}$ | $AP_{bound}$ | $AP_{center}$ | mAP | TOPO | FPS |
|---|---|---|---|---|---|---|---|
| Yes | 42.40 | 53.21 | 54.03 | 43.03 | 48.17(↓0.14) | 46.60(↓0.49) | 6.3 |
| No (InsightMapper) | 44.36 | 53.36 | 52.77 | 42.35 | 48.31 | 47.09 | 7.7 |

# E Other Designs

## E.1 Instance-level Class Prediction

Table 11: Quantitative results of ablation studies on the class head design.

| Aggregation Method | $AP_{ped}$ | $AP_{div}$ | $AP_{bound}$ | $AP_{center}$ | mAP | TOPO |
|---|---|---|---|---|---|---|
| Mean | 42.03 | 49.84 | 54.49 | 42.32 | 47.17(↓1.14) | 45.89(↓1.2) |
| Concatenation (InsightMapper) | 44.36 | 53.36 | 52.77 | 42.35 | 48.31 | 47.09 |

Although each instance contains $n_p$ points, it should only have one predicted class for consistency. In MapTR, after obtaining point embeddings from the transformer decoder, it calculates a new instance-level embedding by taking the mean of all points in an instance. Then, the mean embedding is sent to the class head for class prediction. Differently, in InsightMapper, we propose to use concatenation for class prediction, which preserves more information on points. Two class prediction methods are visualized in Figure 9. Experiment results are reported in Table 11. From the results, it is noted that concatenation achieves a slight performance gain. Thus, the concatenation method is used for class prediction in InsightMapper. Such a design is light and does not affect the efficiency of the model.

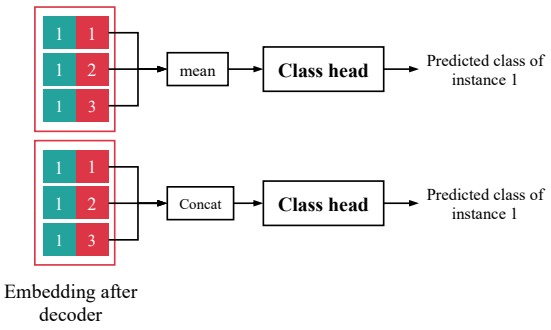



Figure 9: Class head designs. MapTR leverages the mean of points for aggregation (upper design). While InsightMapper uses concatenation for class prediction (lower design).

# F Additional Qualitative Visualizations

Qualitative visualizations are shown in Fig. 10 to Fig. 13. The predicted map contains four classes, i.e., road boundaries (green), lane splits (red), pedestrian crossing (blue), and lane centerlines (pink). We visualize the vectorized HD map of the previous SOTA method MapTR, our proposed InsightMapper, and the ground truth label.

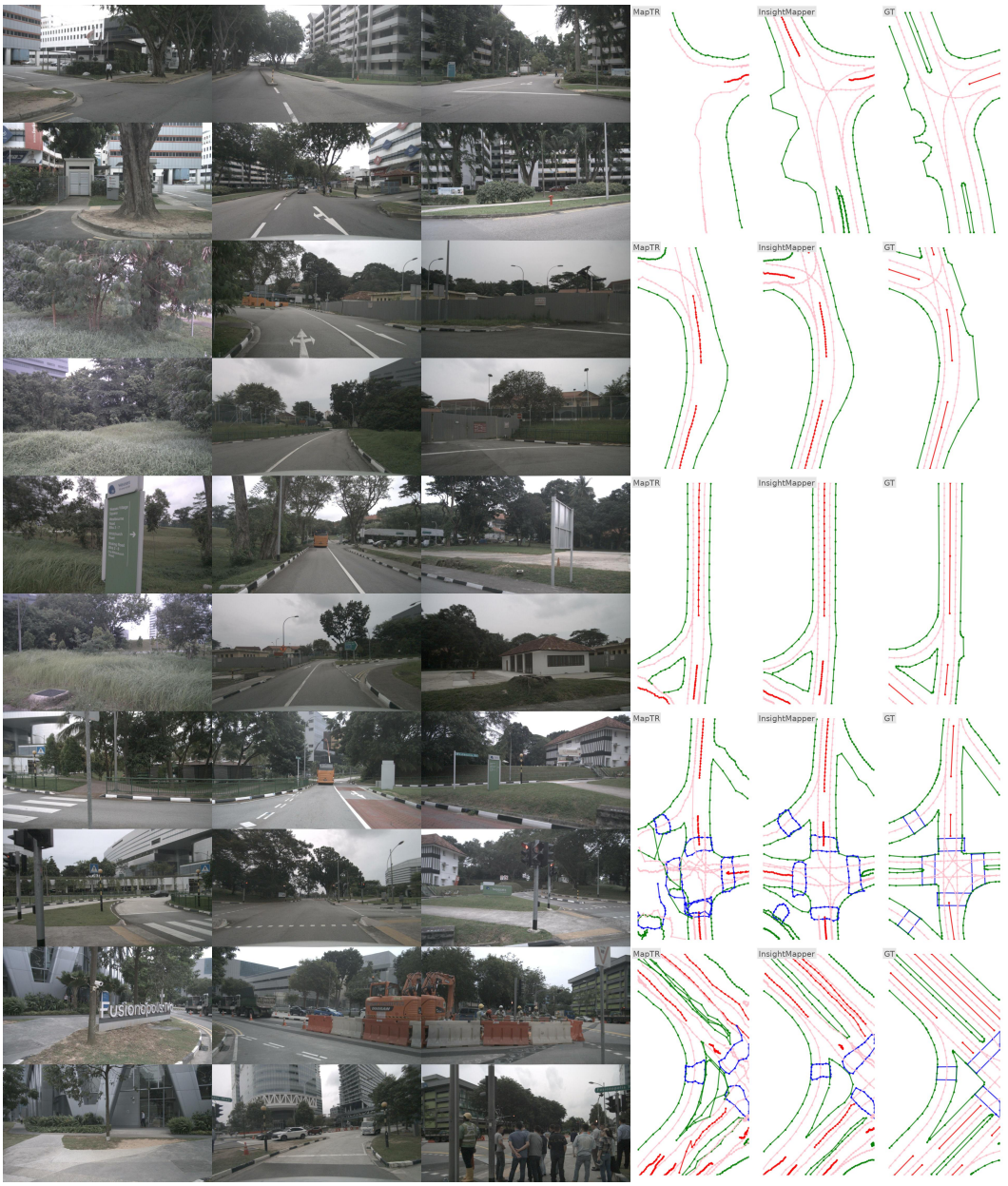



Figure 10: Qualitative visualization. Left three columns are input 6 RGB camera images. For map columns, the first column presents MapTR's results, the second column features InsightMapper's outcomes, and the final column depicts the ground truth map. The predicted map contains four classes, i.e., road boundaries (green), lane splits (red), pedestrian crossing (blue), and lane centerlines (pink).

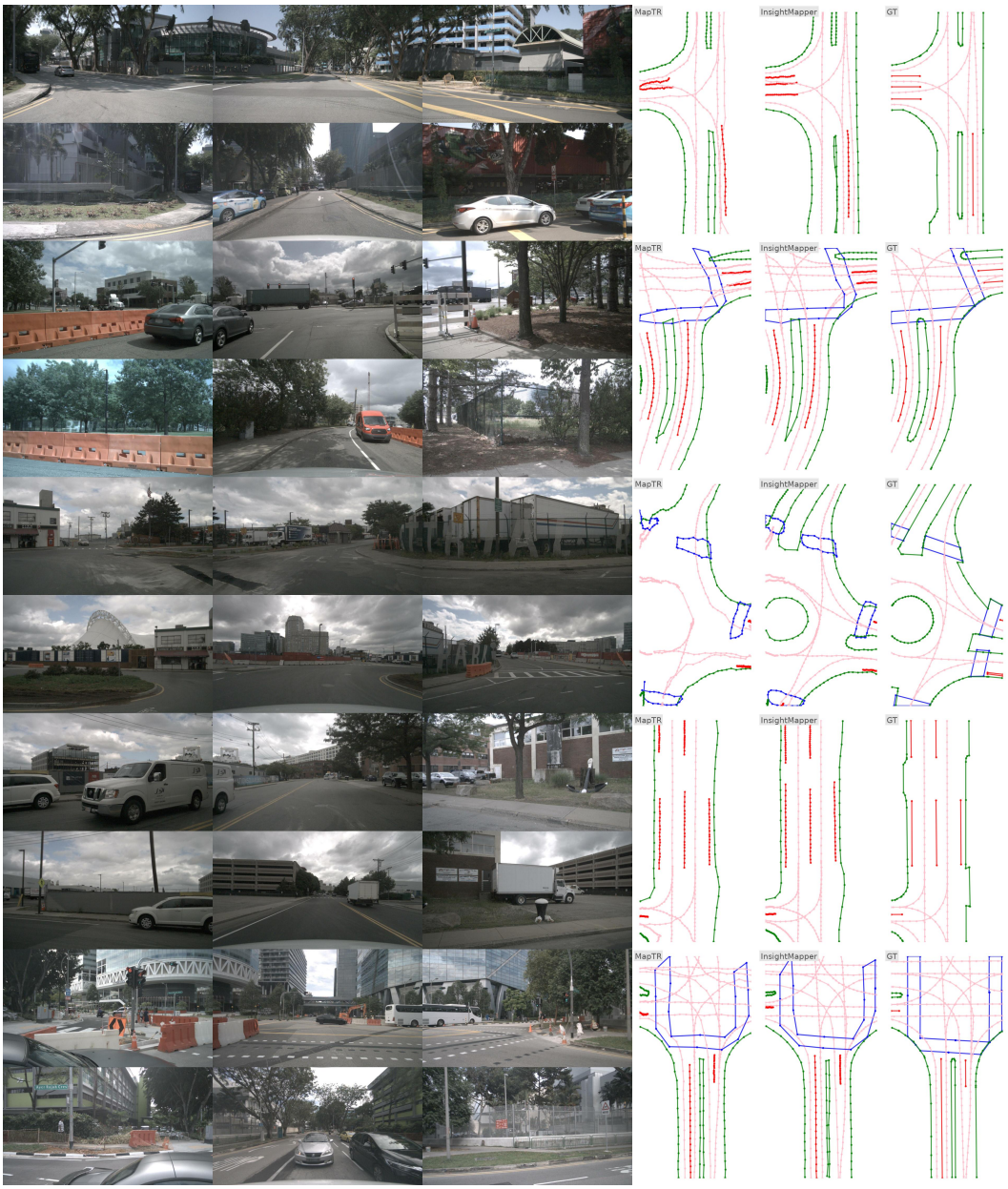



Figure 11: Qualitative visualization. Left three columns are input 6 RGB camera images. For map columns, the first column presents MapTR's results, the second column features InsightMapper's outcomes, and the final column depicts the ground truth map. The predicted map contains four classes, i.e., road boundaries (green), lane splits (red), pedestrian crossing (blue), and lane centerlines (pink).

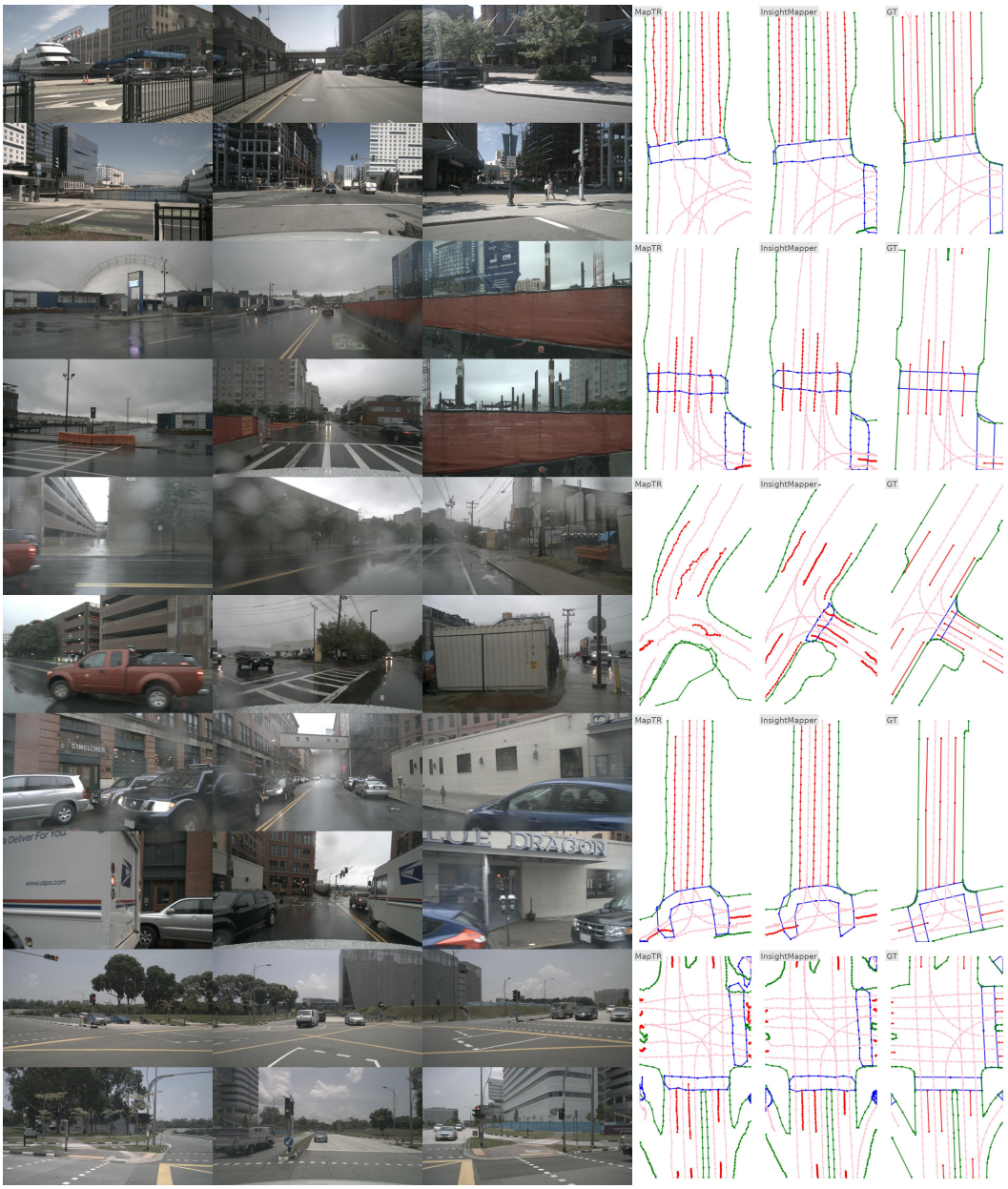



Figure 12: Qualitative visualization. Left three columns are input 6 RGB camera images. For map columns, the first column presents MapTR's results, the second column features InsightMapper's outcomes, and the final column depicts the ground truth map. The predicted map contains four classes, i.e., road boundaries (green), lane splits (red), pedestrian crossing (blue), and lane centerlines (pink).


Figure 13: Qualitative visualization. Left three columns are input 6 RGB camera images. For map columns, the first column presents MapTR's results, the second column features InsightMapper's outcomes, and the final column depicts the ground truth map. The predicted map contains four classes, i.e., road boundaries (green), lane splits (red), pedestrian crossing (blue), and lane centerlines (pink).