Supplementary Material

The supplementary document is organized as follows:

- Sec. A depicts the detailed network architectures of our transformer decoder layers.
- Sec. B provides the implementation details of TransFusion and the training settings on nuScenes and Waymo.
- Sec. C reports our sensitivity analysis of the matching cost during label assignment.
- Sec. D presents the effect of NMS on TransFusion and CenterPoint.
- Sec. E provides the results of using PointPillars as our 3D backbone.
- Sec. F discusses the effect of 2D backbone in TransFusion.
- Sec. G shows the results with different number of object queries.
- Sec. H discusses the performance gain of image information on Waymo.
- Sec. I provides visualization results on the nuScenes and Waymo datasets.

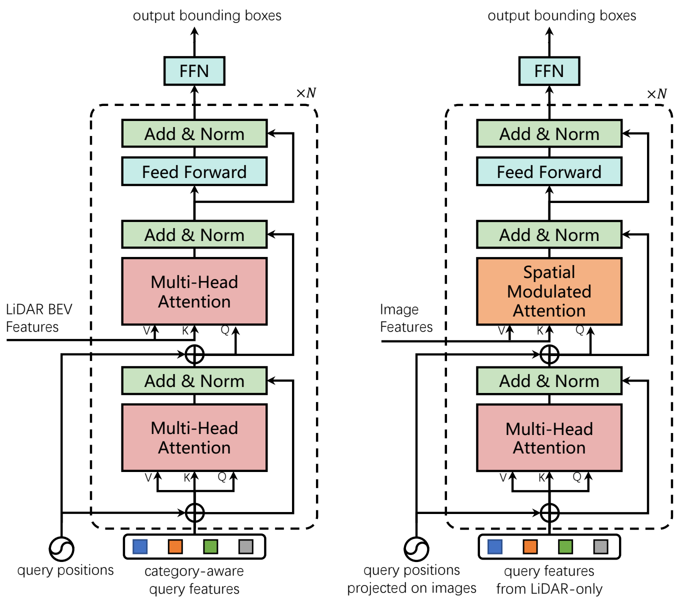

Figure 6. Left: Architecture of the transformer decoder layer for initial bounding box prediction. Right: Architecture of the transformer decoder layer for image fusion.

A. Network Architectures

The detailed architectures of the respective transformer decoder layers for initial bounding box prediction and LiDAR-camera fusion are shown in Fig. 6. Following [20], we adopt the common practice of transformer except that we use the learned positional encoding instead of the fixed sine positional encoding [46]. For the image-guided query initialization module, we use the LiDAR BEV features as query sequence and collapsed image features as key-value sequence, and only perform cross attention to save the computation cost. Our model can benefit from the efficient attention mechanisms in recent works such as [71].

B. Implementation Details

Our implementation is based on the open-sourced codebase MMDetection3D [6], which provides many popular 3D detection methods, including PointPillar, VoxelNet, and CenterPoint. For our 3D backbone, we set its hyperparameters according to CenterPoint-Voxel's official implementation. For the transformer-decoder-based detection head, the hidden dimension d is set to 256 and dropout is set to 0.1. We use $N = 200$ and 300 queries for nuScenes and Waymo since the max numbers of objects in one frame are 142 and 185, respectively. Since our object queries are non-parametric, we are able to modify the number of queries during inference. We provide the ablations on the number of object queries in Sec. G. When selecting object queries from the heatmap, we pick local maximum pixels whose values are greater than or equal to their 8-connected neighbors. To avoid mistakenly suppress nearby instances for small objects, we do not check the local maximum for pedestrian and traffic cone on nuScenes and for pedestrian and cyclist on Waymo. Following PointAugmenting [48], we adopt DLA34 of CenterNet pre-trained on monocular 3D detection task as our 2D backbone for nuScenes. Since there is no public available 2D backbone pre-trained on Waymo dataset, we train a Faster-RCNN [31] using the 2D labels provided by Waymo and use its ResNet50 and FPN as our 2D backbone. We freeze the weight of image backbone during training and set the image resolution to half of the full resolution for both nuScenes and Waymo to speed up the training process.

nuScenes. Following the common practice, we transform previous ten LiDAR sweeps into the current frame to produce a denser point cloud input for both training and inference. The detection range is set to $[-51.2m, 51.2m]$ for X and Y axes, and $[-5m, 3m]$ for Z axis. The maximum numbers of non-empty voxels for training and inference are set to 120,000 and 160,000, respectively. In terms of the data augmentation strategy, we adopt random flipping along both X and Y axes, global scaling with a random factor from $[0.9, 1.1]$, global rotation between $[-\pi/8, \pi/8]$, as well as the copy-and-paste augmentation [53]. We follow CBGS [69] to perform class-balanced sampling and optimize the network using the AdamW optimizer with one-cycle learning rate policy, with max learning rate 0.001, weight decay 0.01, and momentum 0.85 to 0.95. We train the 3D backbone with the first decoder layer and FFN for

20 epochs, and the LiDAR-camera fusion components for 6 epochs with batch size of 16 using 8 Tesla V100 GPUs. We use gradient clipping at an l_2 norm of 0.1 to stabilize the training process. The weighting coefficients of heatmap loss, classification loss, and regression loss are 1.0, 1.0 and 0.25, respectively. The coefficients of matching cost $\lambda_1, \lambda_2, \lambda_3$ are $0.15, 0.25, 0.25$, respectively. The sensitivity analysis of the matching cost coefficients is provided in the next section.

Waymo. For Waymo, we only use a single sweep as input and set the detection range to $[-75.2m, 75.2m]$ for X and Y axes, and $[-2m, 4m]$ for Z axis. The maximum number of non-empty voxels is set to 150,000. We adopt the same training strategies as nuScenes except: 1) The first stage training last for 36 epochs with batch size of 16 under a max learning rate of 0.001. 2) The weighting coefficient of regression loss is changed to 2.0, following CenterPoint. 3) The matching cost coefficients are set to $0.075, 0.25, 0.25$, respectively.

C. Label Assignment Strategy

Following DETR, we perform label assignment by finding the bipartite matching between predictions and ground-truth objects through the Hungarian algorithm. In Table 12, we study the effect of the coefficient of each matching cost term on the detection performance of TransFusion-L. Since the matching results are only decided by the relative values of individual matching cost terms, we keep $\lambda_2 = 0.25$ and try different values for λ_1 and λ_3. As shown in Table 12, we find the network suffers from a convergence issue when the coefficient of the classification cost is too large, and the detection performance is not sensitive to the coefficient within a reasonable range. Since the weighting coefficients of the matching cost need some tuning, we additionally propose a heuristic label assignment strategy (denoted as **Heuristic**) to avoid hyper-parameter tuning. The **Heuristic** assignment strategy follows the simple rules: each GT box will only be assigned to the predicted box with the same category and the smallest center distance. If a conflict appears, the predicted box will be matched to the closer GT box. In this way, we also find the one-to-one matching between prediction and GT but with some GT boxes unused. We find **Heuristic** works quite well for uncrowded scenes but for objects in a crowded scene, such as pedestrian or traffic cone in nuScenes, it is unable to prevent duplicate predictions, which will be further explained in Sec. D.

D. Effect of NMS

Recently, many works [2, 45, 71] in 2D detection have focused on removing the last non-differentiable component, Non-Maximum Suppression (NMS), in the detection pipeline. OneNet [42] systematically compares the end-to-end detectors with non-end-to-end detectors, and claims that the *one-to-one positive sample assignment* as well as *classification cost* in the matching cost are the two key factors in producing a large score gap between duplicate prediction and building an end-to-end detection system without NMS. We refer readers to the original paper [42] for more details. Following DETR's label assignment strategy, our method naturally satisfies these two requirements and do not need NMS. As show in Table 10, our method still maintains a high mAP without NMS while CenterPoint drops about 12% mAP. This advantage eliminates the hand-designed NMS post-processing step and makes TransFusion more practical and handy for deployment to new scenarios in the real applications. Besides, since the **Heuristic** strategy mentioned in Sec. C does not have classification cost involved in the assignment stage, this strategy is unable to produce a large score gap between duplicate prediction. This is why it does not perform as well as the baseline model on Pedestrian and Traffic cone.

Matching strategy	λ_1	λ_2	λ_3	mAP	NDS	Ped.	T.C.
Hungarian	2.0	0.25	0.25	Not Converge			
Hungarian	0.5	0.25	0.25	58.5	66.0	83.8	70.5
Hungarian	0.25	0.25	0.25	59.2	66.1	85.2	71.6
Hungarian	0.15	0.25	0.25	60.0	66.8	86.1	72.1
Hungarian	0.1	0.25	0.25	59.3	66.3	85.3	71.6
Hungarian	0.15	0.25	0.5	59.2	66.1	85.2	70.2
Hungarian	0.15	0.25	0.15	59.5	66.3	85.2	71.8
Hungarian	0.15	0.25	0.1	59.0	65.9	85.1	72.4
Heuristic (w/o NMS)				56.7	65.3	67.3	56.6
Heuristic (w NMS)				60.0	67.0	85.6	71.0

Table 9. Ablation study on the matching cost coefficients in Eq. 1. 'Ped.', and 'T.C.' are short for pedestrian, and traffic cone, respectively.

Method	with NMS	without NMS
CenterPoint	57.41	45.70
TransFusion-L	59.95	59.98
TransFusion	65.58	65.60

Table 10. Effect of NMS. We report the mAP of CenterPoint and our TransFusion on nuScenes validation set. The results of CenterPoint are reproduced using MMDetection3D, which also uses a resolution of $(0.075m, 0.075m, 0.2m)$ without any test time augmentation.

E. Pillar-based 3D Backbone

To demonstrate our framework's compatibility with other 3D backbones, we use PointPillars as our 3D backbone to produce the BEV features, while keeping all the other settings the same as the main experiments. The voxel size is set to $(0.2m, 0.2m)$. As shown in Table 11, our model outperforms CenterPoint by a remarkable margin under the same pillar-based backbone, showing its great generalization ability.

	PointPillars		VoxelNet	
	mAP	NDS	mAP	NDS
CenterPoint	50.3	60.2	59.6	66.8
TransFusion-L	54.5	62.7	65.1	70.1
TransFusion	58.3	64.5	67.5	71.3

Table 11. Results on nuScenes validation set.

F. Discussions of the 2D Network

Current multi-modality detection models usually employ CNN features from 2D networks pre-trained on different tasks (i.e., segmentation or detection) and with different resolution (i.e., different levels from ResNet or DLA). There is no existing work analyzing what kind of image features are most useful for a 3D detection model, and using improper image features might prevent the release of the potential for a multi-modality detection system. We believe that the sequential design of our method enables a flexible and off-the-shelf experiment base to explore the effects of different image features. Therefore, we explore this question by fixing the 3D backbone with the first decoder layer and performing the second stage training with different image features.

Arch.	Task	mAP	NDS
DLA34	Monocular 3D Det.	65.6	69.7
R50+FPN Level 0	2D Det.	66.4	70.1
R50+FPN Level 0	2D Instance Seg.	66.6	70.1
R50+FPN Level 1	2D Instance Seg.	66.5	70.1
R50+FPN Level 2	2D Instance Seg.	66.3	70.0
R50+FPN Level 3	2D Instance Seg.	65.4	69.6

Table 12. mAP under different 2D backbones. 'Det.' and 'Seg.' are short for detection and segmentation, respectively. For DLA34 on monocular 3D detection, we acquire the CenterNet pre-trained on nuScenes[4] following PointAugmenting. For ResNet50 on instance segmentation, we acquire the model pre-trained on nuImages from MMDetection3D. For ResNet50 on 2D detection, we train the model by ourself using MMDetection3D since there is no open-sourced model weights.

From Table 12, we find image features of the 2D instance segmentation model bring the largest performance boost compared with that of detection models. In terms of different levels of the feature pyramid, the feature map of level 0 (stride 4) brings a slightly larger performance gain. We suspect the image features at that level contain more fine-grained information which is important to distinguish small or distant objects. Image features from level 1 (stride 8) and level 2 (stride 16) can bring a similar gain with a smaller resolution of feature maps, while image features from level 3 (stride 32) yields a drop of 1.2% mAP in comparison with the level-0 counterpart due to the row resolution.

[4]https://github.com/xingyizhou/CenterTrack

G. Adapt Queries at Test Time

Unlike DETR, our object queries are non-parametric and input-dependent. These two characteristics allow us to use different numbers of queries during inference. It could be useful when we have some prior knowledge about a scene, such as its crowdedness. In Table 13, we provide the performance evaluated under different object queries for the same model trained under $N = 200$ queries. Note that we use $N = 200$ to get all the numbers in the main text for its better performance-efficiency trade-off and use $N = 300$ for online submission for a slightly better performance.

#queries	100	200	300	500
mAP	64.2	65.6	65.9	66.0
NDS	69.2	69.7	69.8	69.8

Table 13. Results with different numbers of queries. We keep the model unchanged and only use different numbers of queries for evaluation.

H. Dicussions on Waymo

Our TransFusion brings smaller performance gain over TransFusion-L on Waymo compared with that on nuScenes. We speculate that this is mainly due to the following two reasons:

(i) As shown in Table 1, compared with TransFusion-L, TransFusion brings the largest performance increase for bicycle (+8.7%), motorcycle (+5.4%), and construction vehicle (+4.9%) in terms of mAP on nuScenes. Due to the geometrical ambiguity, objects from the above three categories are difficult to distinguish using LiDAR information only, and thus the semantic information of images is extraordinarily important for more accurate classification. However, the categorization of Waymo is rather coarse-gained (i.e., vehicle, pedestrian, cyclist), which hides the improvement brought by the image information to some extent.

(ii) The LiDAR point clouds in Waymo are much denser than those in nuScenes (see Sec. I for visualizations). Thus the bounding box predictions of TransFusion-L are already with accurate localization, which reduces the room for further improvement by image fusion.

I. Qualitative Results

We first compare the detection results of TransFusion and TransFusion-L on the nuScenes dataset in Fig. 7. The image information improves the performance of the LiDAR-only baseline through reducing the False Positive and False Negative. More visualization results on Waymo and nuScenes datasets are shown in Fig. 8 and Fig. 9, respectively.

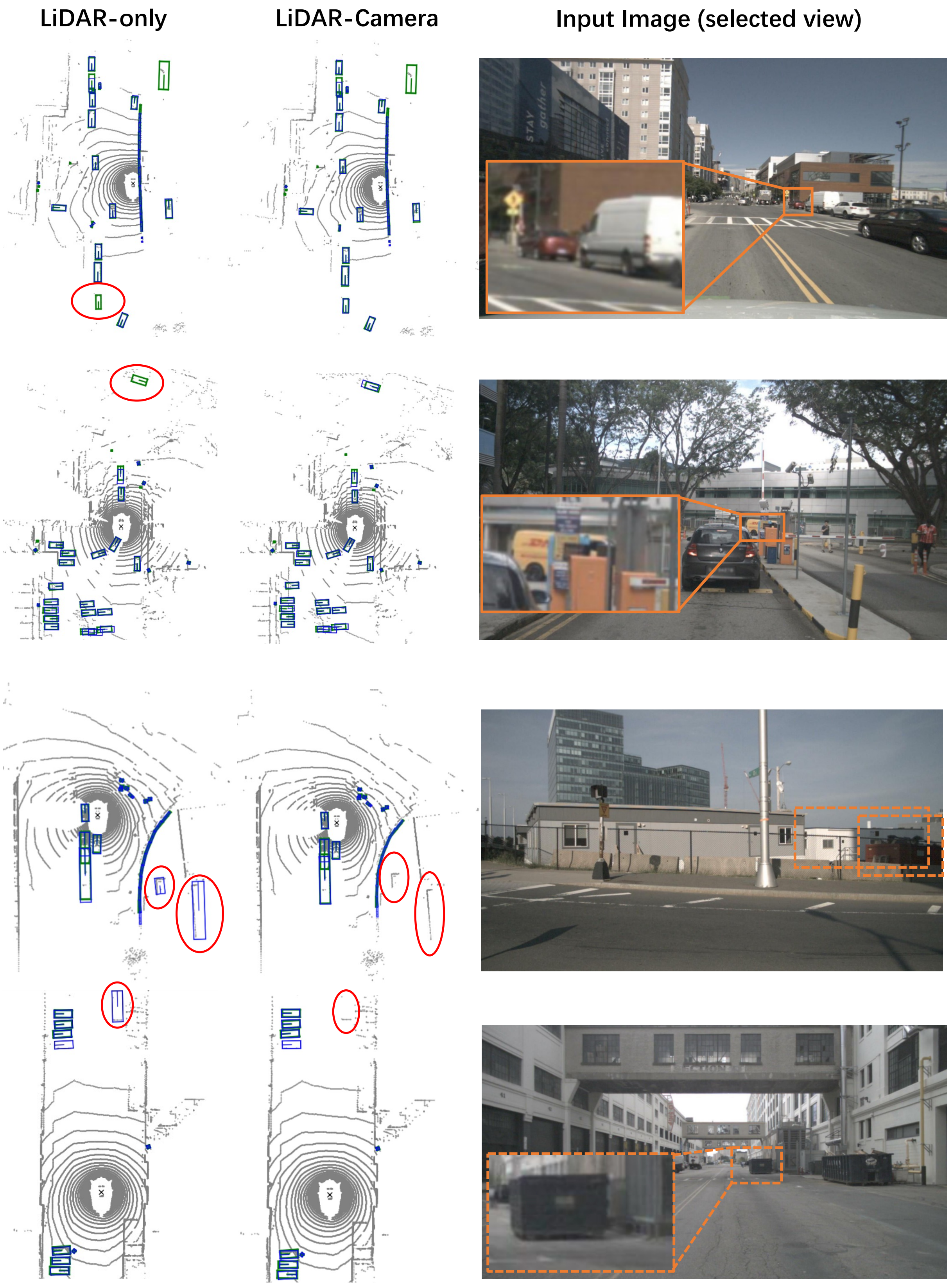

Figure 7. Qualitative comparison between TransFusion-L and TransFusion on the nuScenes dataset. Blue boxes and green boxes are the predictions and ground-truth, respectively. Best viewed with color and zoom-in.

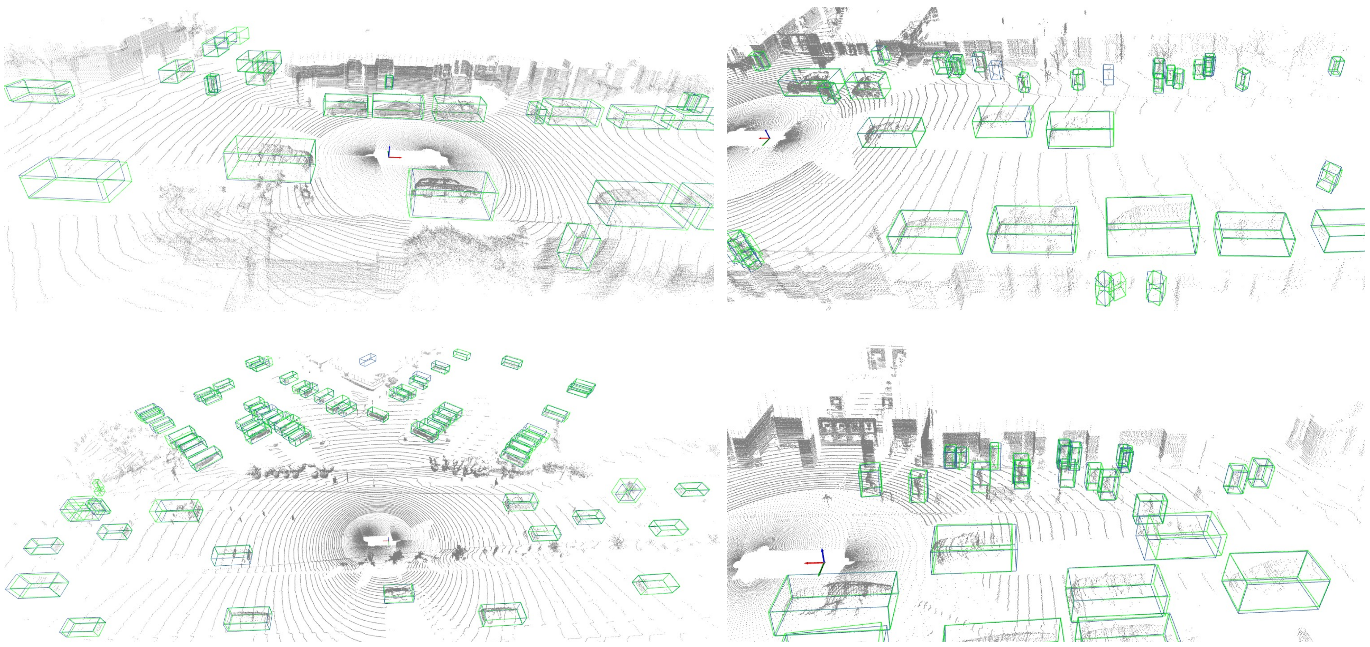

Figure 8. Visualization of detection results on the Waymo dataset. Our model predicts highly accurate bounding boxes for nearby vehicles and pedestrians (note that cyclists are very rare in the dataset) and also handles objects with severe occlusion. Blue boxes and green boxes are the predictions and ground-truth, respectively. Best viewed with color and zoom-in.

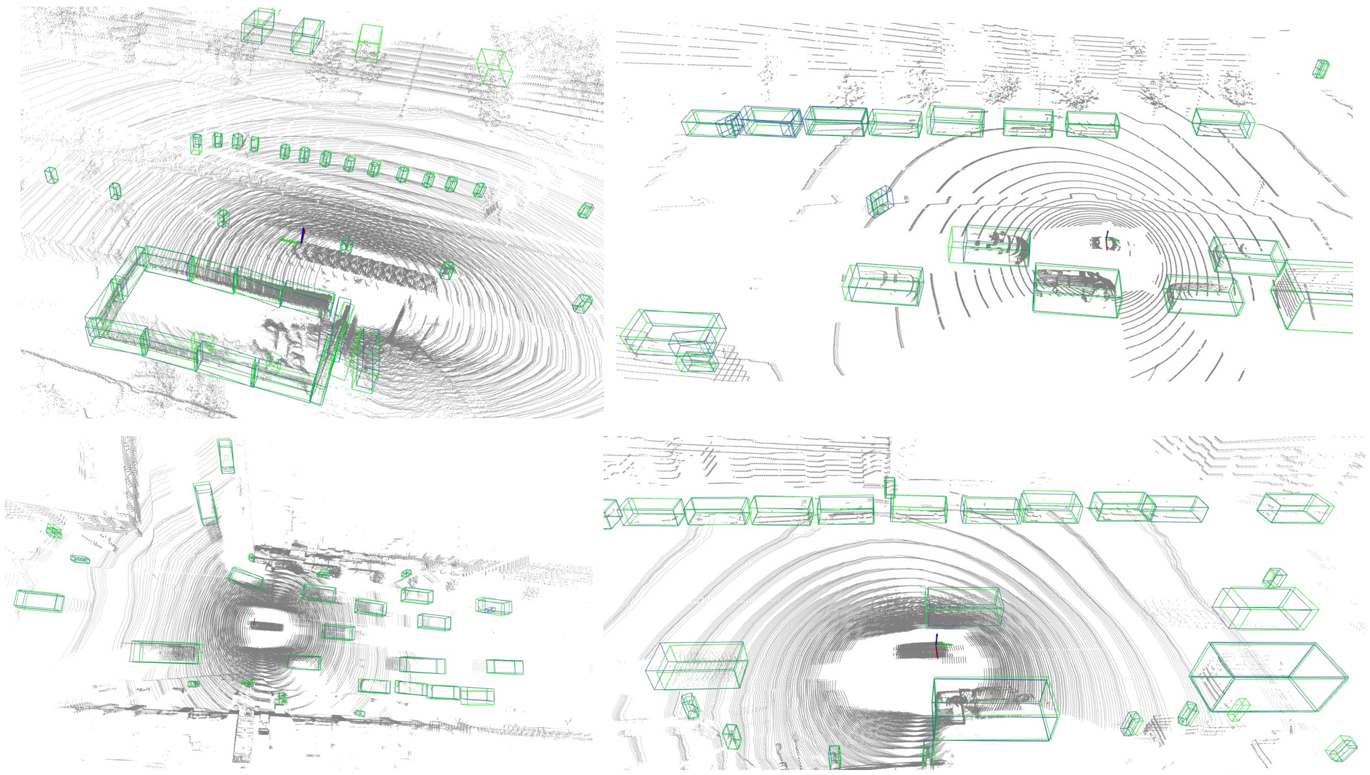

Figure 9. Visualization of detection results on the nuScenes dataset. Compared with Waymo, nuScenes has much sparser point clouds and smaller objects such as traffic cones. Nevertheless, our model successfully detects such objects even with only few points observed. Blue boxes and green boxes are the predictions and ground-truth, respectively. Best viewed with color and zoom-in.