Supplementary Material

This supplementary material contains the following sections: Sec. A shows the attention layer in the point-voxel Transformer; Sec. B reports more experimental results; Sec. C presents visualization results on the Waymo dataset.

A. Attention Layer

In this section, we show Attention$(\mathcal{F}_s, \mathcal{F}_{\text{query}}, \mathcal{P}_s, \mathcal{P}_{\text{query}})$ and use the single-head version for clarity. Given matrices of content queries $\mathcal{F}_{\text{query}} \in \mathbb{R}^{m\times d}$, features of generated tokens $\mathcal{F}_s \in \mathbb{R}^{k\times d}$, reference points $\mathcal{P}_{\text{query}} \in \mathbb{R}^{m\times 3}$, and coordinates of generated tokens $\mathcal{P}_s \in \mathbb{R}^{k\times 3}$, we first transform the input features to query Q, key K, and value V:

$$Q = \phi_q(\mathcal{F}_{\text{query}}), \quad K = \phi_k(\mathcal{F}_s), \quad V = \phi_v(\mathcal{F}_s), \tag{1}$$

where ϕ_q, ϕ_k, and ϕ_v are linear layers. To make better use of the position information, we adopt the contextual relative positional encoding [1]. Specifically, we map the relative coordinates between the query and key to the position encoding:

$$\begin{aligned} S^{u,v,w} &= \mathcal{P}^{u,w}_{\text{query}} - \mathcal{P}^{v,w}_{s}, \\ E_q = \delta_q(S), &\quad E_k = \delta_k(S), \quad E_v = \delta_v(S), \end{aligned} \tag{2}$$

where $E_q \in \mathbb{R}^{m\times k\times d}$, $E_k \in \mathbb{R}^{m\times k\times d}$, and $E_v \in \mathbb{R}^{m\times k\times d}$ are the position encoding for query, key, and value, respectively, and δ_q, δ_k, and δ_v are MLPs with two linear layers and one ReLU nonlinearity. The position encoding performs product with query and key features to obtain positional bias $B \in \mathbb{R}^{m\times k}$, which is added to the attention map:

$$\begin{aligned} B^{u,v} &= \sum_w Q^{u,w} \cdot E_q^{u,v,w} + \sum_w K^{v,w} \cdot E_k^{u,v,w}, \\ F^{u,v}_{\text{attn}} &= \sigma\left(\frac{\sum_w Q^{u,w} \cdot K^{v,w} + B^{u,v}}{\sqrt{d}}\right), \end{aligned} \tag{3}$$

where σ is the softmax normalization function. Then, we add the value feature with its corresponding position encoding and perform a weighted sum to obtain the output features $F \in \mathbb{R}^{m\times d}$:

$$F^{u,w} = \sum_v F^{u,v}_{\text{attn}} \cdot V^{v,w} + \sum_v F^{u,v}_{\text{attn}} \cdot E_v^{u,v,w}. \tag{4}$$

B. More Experiments

Table 1 illustrates that a 20× measured speedup can be achieved by sampling the reduced voxels instead of the raw points in query initialization. Table 2 shows that in the point token generation, our implemented k-nearest neighbors and interpolation outperform the naïve PyTorch implementation (in Figure 1) with 8× measured speedup and 100× memory reduction. Table 3 studies the effect when a different number of neighbors is interpolated in the point token generation. Table 4 shows the influence of sampling a different number of point and voxel tokens for each reference point in the point-voxel Transformer. Continued performance improvements can be observed as the number of tokens increases since more detailed information is preserved. Note that we sample 128 point and voxel tokens in the main text for its better performance-efficiency trade-off. Table 5 ablates the number of Transformer blocks for the point-voxel Transformer. We find that the model can achieve good performance with only a single Transformer block.

Table 1. Comparisons of the latency when sampling on the raw points and the reduced voxels in the query initialization.

Case	Points	Voxels
Latency (ms)	14.6	**0.7**

Table 2. Comparisons of our implemented k-nearest neighbors and interpolation with the naïve PyTorch implementation.

Methods	Latency (ms)	Memory (GB)
Baseline	10.9	1.14
Ours	**1.3**	**0.01**

Table 3. Ablation study of the number of neighbors to interpolate in the point token generation.

# neighbors	1	8	16
APH	64.75	**65.01**	64.96

Table 4. Ablation study of the number of sampled point and voxel tokens in the point-voxel Transformer.

# tokens	64	128	256
APH	64.43	65.01	**65.26**

Table 5. Ablation study of the number of Transformer blocks.

# blocks	1	2	3
APH	65.01	**65.03**	64.86

C. Qualitative Results

Figure 2 illustrates the visualization of our method on the Waymo dataset. Our model can predict highly accurate bounding boxes for nearby objects and also handle objects with severe occlusion, which demonstrates the high-quality prediction results of our model.

Failure Cases. In Figure 3, we observe that small objects, such as pedestrians, are sometimes not detected. These objects usually occupy fewer voxels than other objects, causing them to be less likely to be sampled. In the future, a better query initialization module could be designed to improve the quality of the reference points to recall the corresponding objects.

```python
def naive_knn_and_k_interpolate(src_xyz, tgt_xyz, tgt_feats, K):
    """
    Args:
        src_xyz: (N, 3)
        tgt_xyz: (M, 3)
        tgt_feats: (M, C)
    Return:
        src_feats: (N, C)
    """
    # (N, M, 3) -> (N, M), memory-intensive
    dist = torch.norm(src_xyz[:, None, :] - tgt_xyz[None, :, :], dim=-1)
    # (N, K), time-consumption
    k_dist, k_indices = torch.topk(dist, K, dim=-1, largest=False)
    k_dist_recip = (1.0 / (k_dist + 1e-8))
    norm = torch.sum(k_dist_recip, dim=-1, keepdim=True)
    weight = k_dist_recip / torch.clamp_min(norm, min=1e-8)
    # (N, K, C) -> (N, C), memory-intensive
    src_feats = torch.sum(tgt_feats[k_indices] * weight[:, :, None], dim=1)
    return src_feats
```

Figure 1. The naïve PyTorch implementation of k-nearest neighbors and interpolation in the point token aggregation.

Figure 2. Visualization of detection results on the Waymo validation set. We show the raw point cloud in blue, ground truth in green bounding boxes, our detected objects in gray bounding boxes, and points inside our detected boxes in orange.

Figure 3. Visualization of failure cases on the Waymo dataset.

References

[1] Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu, and Hongyang Chao. Rethinking and improving relative position encoding for vision transformer. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 1