Supplementary Material for "RangeDet: In Defense of Range View for LiDAR-based 3D Object Detection"

1. Detailed Configuration

Training scheme. All experiments use the SGD optimizer with 0.9 momentum and 1e-5 weight decay. We use 8 2080Ti GPUs for training. The mini-batch size of each GPU is 2. We train our model for 36 epochs from scratch, and adopt the cosine learning rate scheduler with an initial learning rate of 0.01.

Inference. During inference, we treat proposals with scores higher than a certain threshold as positive proposals. The threshold for vehicle/pedestrian/cyclist is 0.5/0.3/0.3, respectively. The IoU threshold in NMS (or Weighted NMS) for vehicle/pedestrian/cyclist is 0.2/0.3/0.3, respectively.

Network Details. Each stage of our backbone consists of a couple of Basicblocks [1]. The number of blocks in each stage: Res1:2, (Res2, Res3):3, (Res4, Res5):5, (Agg1, Agg2, Agg3, Agg4, Agg5):2. The number of filters in convolution layers in each stage: (Res1, Agg1, Res2, Agg2, Res3, Agg3):64, (Res4, Agg4, Res5, Agg5):128. Both classification (IoU prediction) and regression branches contain four consecutive convolutions. The kernel sizes of the first three convolutions are 3×3, and the last one is 1×1. Batch Normalization and ReLU are adopted between two consecutive convolutions.

2. Implementations of Point-based Operators

For all these point-based operators, we first re-define their neighborhood in the range view. In 3D space, kNN or ball query is adopted for neighbor query. In the range image, for each position, points in its 3×3 neighborhood are regarded as its neighbors. In 3D space, FPS is usually used to find key points for downsampling. We do not select key points since we use convolution with stride to downsample feature maps. So, these operators are applied to every position (point) in range image. For clarity, the center point of a local 3×3 neighborhood is denoted as p_i, and p_j denotes one of its neighbors. Thus, their feature vectors denoted as f_i and f_j. Their coordinates are denoted as x_i and x_j.

For PointNet-RV, the concatenation of $x_i - x_j$ and f_j is denoted as $\widehat{f}_j$. A two-layer MLP with 64 filters generates output feature f_j^o for each p_j from $\widehat{f}_j$. The output feature of p_i is the max-pooling of all f_j^o.

For EdgeConv-RV, the input of MLP is $f_i - f_j$. Other implementation details are same with PointNet-RV.

For efficient version of ContinuousConv-RV, we use the same MLP to generate weight vector w_j from $x_i - x_j$. Output feature of p_j is the element-wise product of w_j and f_j. Finally, we use channle-wise summation to aggregate output feature of every p_j.

RSConv-RV is similar to ContinuousConv-RV. The difference is RSConv-RV use max-pooling as aggregation.

For RandLA, the model first learns a feature $f_j^{(1)}$ from the concatenation of x_j and $x_i - x_j$ via a Fully-Connected (FC) layer. The FC layer has 64 filters and is followed by Batch Normalization and ReLU. Then we concatenate f_j and $f_j^{(1)}$, denoted as $f_j^{(2)}$. Another FC layer with 64 filters is used to learn the attentive pooling scores from $f_j^{(2)}$. Finally, we aggregate all $f_j^{(2)}$ to get the output feature of p_i by attentive pooling.

For a better understanding of Meta-Kernel, we summarize the differences between several closely related work and our Meta-Kernel convolution in Table **??**.

3. Detailed Results

Table 1, Table 2 and Table 3 show the detailed results of our best model.

Conditions	3D AP/APH	BEV AP/APH
Overall LEVEL_1	72.85/72.33	86.94/86.22
Overall LEVEL_2	64.03/63.57	78.07/77.40
[0m, 30m) LEVEL_1	87.96/87.44	94.35/93.77
[0m, 30m) LEVEL_2	87.17/86.66	93.66/93.09
[30m, 50m) LEVEL_1	69.03/68.53	85.66/84.93
[30m, 50m) LEVEL_2	63.11/62.64	79.80/79.09
[50m, +inf) LEVEL_1	48.88/48.35	77.01/75.92
[50m, +inf) LEVEL_2	38.42/37.98	62.73/61.78

Table 1. Detailed results of vehicle detection on WOD validation split. IoU threshold is 0.7.

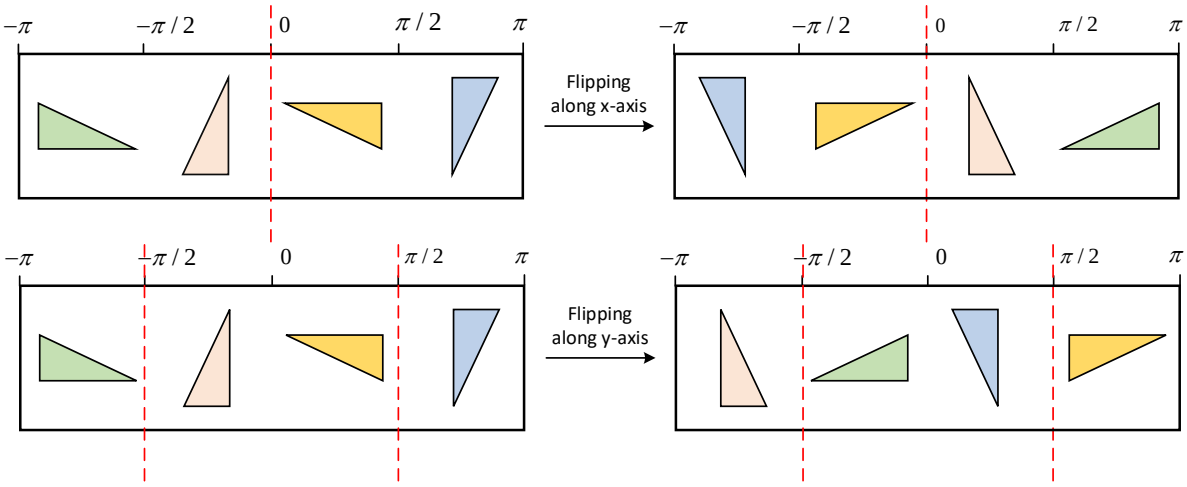


Figure 1. Illustration of random global flipping in range view. Rectangles with black solid line stand for range images. Triangles in the same color indicate an object before and after the augmentation. Red dash lines indicate the flipping axes.

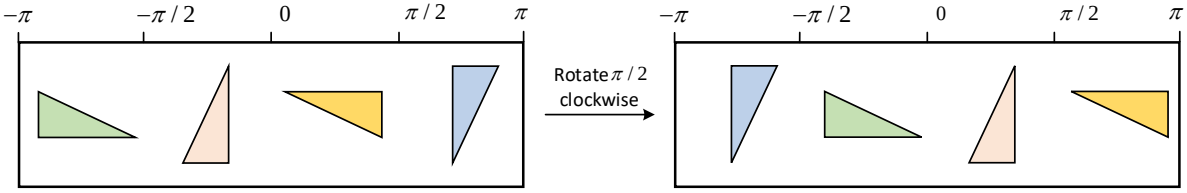


Figure 2. Illustration of random global rotation in range view.

Conditions	3D AP/APH	BEV AP/APH
Overall LEVEL_1	75.94/71.94	81.02/76.52
Overall LEVEL_2	67.60/63.89	72.94/68.66
[0m, 30m) LEVEL_1	82.20/79.01	85.96/82.48
[0m, 30m) LEVEL_2	77.92/74.82	82.13/78.71
[30m, 50m) LEVEL_1	75.39/70.93	80.96/75.94
[30m, 50m) LEVEL_2	68.03/63.87	73.68/68.93
[50m, +inf) LEVEL_1	65.74/58.31	74.44/65.47
[50m, +inf) LEVEL_2	51.33/45.23	59.65/52.01

Table 2. Detailed results of pedestrian detection on WOD validation split. IoU threshold is 0.5.

Conditions	3D AP/APH	BEV AP/APH
Overall LEVEL_1	65.67/64.39	68.45/67.07
Overall LEVEL_2	63.33/62.08	65.76/64.45
[0m, 30m) LEVEL_1	79.33/77.87	80.41/78.92
[0m, 30m) LEVEL_2	78.82/77.38	79.90/78.42
[30m, 50m) LEVEL_1	55.80/54.76	58.88/57.76
[30m, 50m) LEVEL_2	52.52/51.44	55.45/54.39
[50m, +inf) LEVEL_1	45.00/43.93	50.76/49.50
[50m, +inf) LEVEL_2	41.97/40.97	47.22/46.05

Table 3. Detailed results of cyclist detection on WOD validation split. IoU threshold is 0.5.

4. Illustration of Data Augmentation

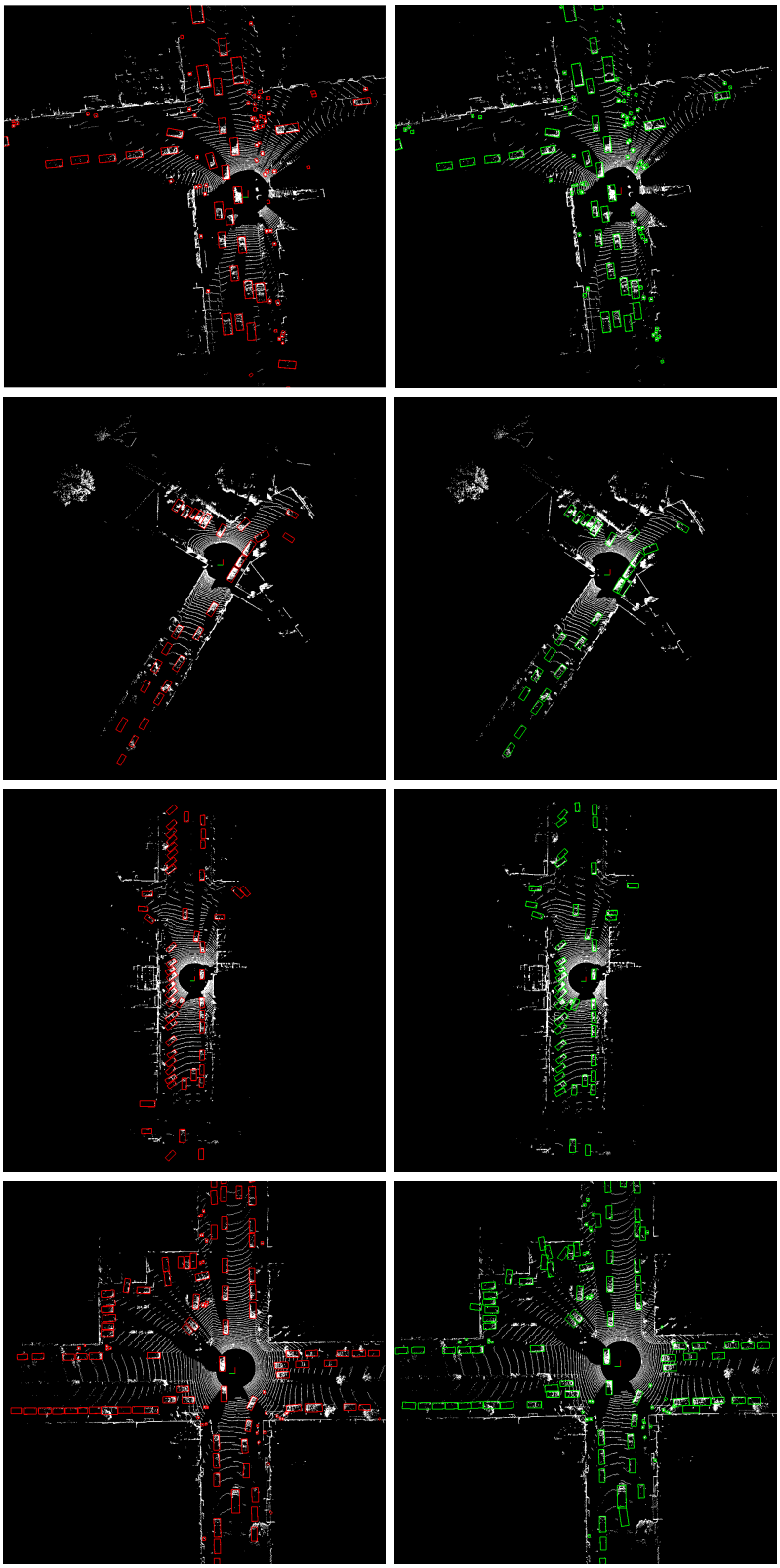
We have described data augmentation in range view in the main paper. For a better understanding, we provide illustrations of global flipping and global rotation, shown in Fig. 1 and 2.

5. Qualitative Results

The qualitative results are showing in Fig. 3. Note that most false negatives containing no points, so they can not be detected using a single-frame algorithm.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016. 1



Ground Truth Prediction

Figure 3. Qualitative results.