1st Place Solution for Waymo Open Dataset Challenge - 3D Detection and Domain Adaptation

Zhuangzhuang Ding* Yihan Hu* Runzhou Ge*
Li Huang Sijia Chen Yu Wang Jie Liao
Horizon Robotics
{zhuangzhuang.ding, yihan01.hu, runzhou.ge, yu04.wang}@horizon.ai

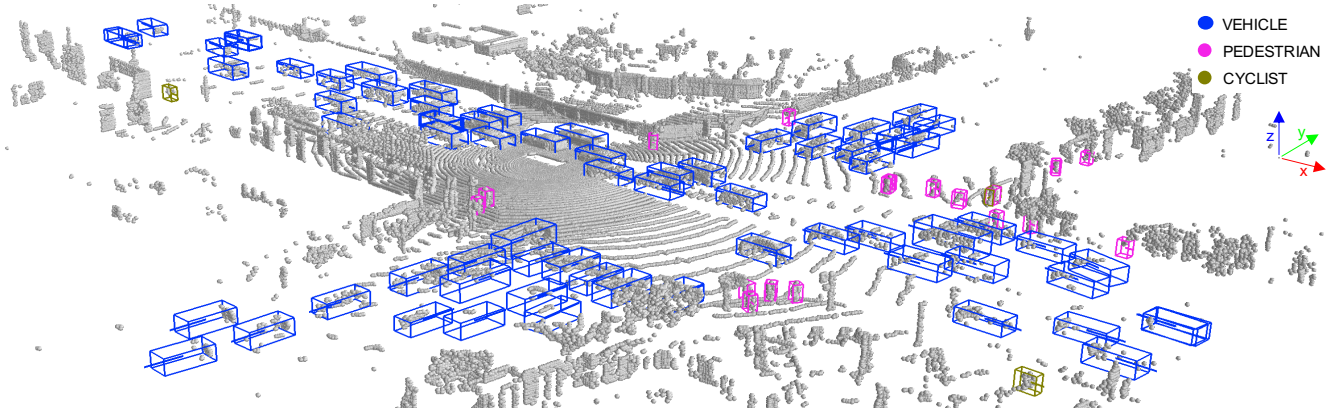

Figure 1: Detection results on testing set of Waymo Open Dataset 3D detection track. Legends indicate the color used for each class.

Abstract

*In this technical report, we introduce our solution "HorizonLiDAR3D" for the 3D detection track and the domain adaptation track in Waymo Open Dataset Challenge at CVPR 2020. Many existing 3D object detectors include prior-based anchor box design to account for different scales and aspect ratios and classes, which limits its capability of generalization to a different dataset or domain and requires post-processing (*e.g. *Non-Maximum Suppression (NMS)). We propose an anchor-free and NMS-free point cloud 3D detector AFDet, using object keypoints to encode the 3D attributes, and to learn an end-to-end point cloud object detection without hand-crafted feature engineering or anchor box design. To address the problem of detecting small objects in the far-field with sparse point clouds, we developed a painting method that fuses the camera and LiDAR data to enrich the feature representations, so as to improve the performance of the detector. The final detection performance also benefits from model ensemble and test-time augmentation in both the 3D detection track and the domain adaptation track. Our solution achieves the 1st place with* 77.11% *mAPH/L2 and* 69.49% *mAPH/L2 respectively on the 3D detection track and the domain adaptation track. Ablation studies are provided to quantify the contributions of each components (*e.g. *model optimization, data augmentation and fusion).*

1. Introduction to the Challenge

The Waymo Open Dataset challenge at CVPR 2020 is one of the largest and the most challenging competitions in the field of autonomous driving. The Waymo Open Dataset [15] is a newly released high-quality LiDAR and camera dataset. In the 3D detection track, the challenge requires the algorithm to detect the objects as a set of 3D bounding boxes. The domain adaptation track is similar to the 3D detection track, but with data collected from a different location.

2. Network

In this section, we present the details of our 3D detector in the challenge. We use our AFDet [2] as the baseline 3D point cloud detector. AFDet is an anchor-free and NMS-free 3D detector. The AFDet consists of four modules: point cloud encoder, 3D Feature Extractor, Region Proposal

*These authors contributed equally to this work.

Network (RPN), and anchor-free detector. We mainly describe the difference between the original AFDet and the one we used in this section.

2.1. Point Cloud Encoder

We use a simple point cloud encoder [22] to voxelize the point cloud. Specifically, we use grid size $0.04m$, $0.04m$, $0.1m$ along x, y, z axis respectively to convert the raw point cloud into voxel presentation. In each voxel, we calculate the mean of all points in the same voxel and feed them to the next modules.

2.2. 3D Feature Extractor and RPN

We introduce two different 3D Feature Extractor *FE-v1*, *FE-v2*, and three different RPN [18, 21] *RPN-v1*, *RPN-v2*, *RPN-v3*. Their combinations B1, B2 and B3 are used in final submissions during the challenge.

SpMiddleFHD in [22] are employed as the 3D Feature Extractor in *FE-v1*. It contains four phases, and each phase contains several submanifold convolutional layers [18] and one sparse convolutional layer [18] to perform downsampling along the z-axis. The feature map is downsampled $4\times$ after *FE-v1*.

For the 3D Feature Extractor in *FE-v2*, we use a similar extractor to [5] with auxiliary branch removed. It has four sparse 3D convolution blocks, each of which is composed of sub-manifold convolutions with the kernel size of 3. The last three blocks use an additional sparse convolution with a stride of 2 to downsample feature map size. Then the feature map is squeezed along the depth dimension to obtain the bird's-eye view (BEV) representation. Eight standard 3×3 convolutions are applied to further abstract the feature. The final downsample factor for *FE-v2* is 8.

RPN has several downsample and upsample blocks. In *RPN-v1*, we use the same structure as in [8, 18]. *RPN-v1* has 3 cascaded downsample blocks, each of which downsamples the feature map by 2. So the feature map of each downsample blocks is downsampled by 2, 4, 8, respectively. Then three upsample blocks are used separately to upsample the feature map by 1, 2, 4. We concatenate them to get the output of the *RPN-v1*. So the final feature map of *RPN-v1* is downsampled $2\times$.

The *RPN-v2* shares the same structure with *RPN-v1* except for two differences. The first downsample block uses a downsample factor of 1 instead of 2, and thus the final downsample factor is 1 for *RPN-v2*. The other difference is that the channels of each block are doubled compared with *RPN-v1*.

We further improves *RPN-v3* over *RPN-v1*. Based on *RPN-v1*, the first downsample block uses a factor of 1 instead of 2. Then we reduce the number of filters for each layer to 3/4 (*e.g.* from 128 to 96, from 256 to 192). Inspired by [16], we add a simple version of BiFPN with 4 repeating blocks after the downsample blocks in the *RPN-v3*. Unlike the original BiFPN [16] with different number of filters for each repeat, we use three separate convolutional layers to produce the same number of filters (96) for the output of each downsample block. Then we repeat the block four times with the same number of filters. Our simple BiFPN does not deploy weighted fusion mechanism [16]. The final downsample factor of *RPN-v3* is 1.

	3D Feature Extractor	RPN	Downsample Factor
B1	*FE-v1*	*RPN-v1*	8
B2	*FE-v2*	*RPN-v2*	8
B3	*FE-v1*	*RPN-v3*	4

Table 1: 3D Feature Extractor and RPN combinations used in the challenge.

The 3D Feature Extractor and RPN combinations of B1, B2, and B3 can be found in Table 1. We use B1, B2 and B3 in three different models during the challenge. In 3D Feature Extractor and RPN, each convolution is followed by a batch normalization [6] and ReLU non-linearity.

2.3. Anchor Free Base Detector

We briefly introduce the anchor free base detector in the report. For more details, please refer to [2]. The AFDet [2] consists of five sub-heads, including the keypoint heatmap head, the local offset head, the z-axis location head, the 3D object size head, and the orientation head. Figure 2 shows the details of the anchor free detector.

Five Sub-Heads. The heatmap head and the offset head predict a keypoint heatmap $\hat{M} \in \mathbb{R}^{W\times H\times C}$ and a local offset regression map $\hat{O} \in \mathbb{R}^{W\times H\times 2}$ respectively, with C the number of keypoint types. The keypoint heatmap helps to locate the object center (x-y coordinates) in BEV. The offset regression map compensates the discretization error due to voxelization, and helps to recover more accurate object locations in BEV. The z-axis location head regresses the z-axis values. Additionally, we regress the object sizes $\hat{S} \in \mathbb{R}^{W\times H\times 3}$ directly. For orientation prediction, we use the *MultiBin* method following [12, 20].

Loss. The heatmap head training uses the modified focal loss [9]. For the orientation prediction head, the classification part is trained with softmax while the offset part is trained with L_1 loss. L_1 loss is employed in the local offset head, the z-axis location head, and the 3D object size head training. The overall training objective is

$$\mathcal{L} = \mathcal{L}_{heat} + \lambda_{off}\mathcal{L}_{off} + \lambda_z\mathcal{L}_z + \lambda_{size}\mathcal{L}_{size} + \lambda_{ori}\mathcal{L}_{ori}, \tag{1}$$

where λ represents the weight for each sub-task. For all regression sub-tasks including local offset, z-axis location, size and orientation, we only regress N objects which are in the detection range.

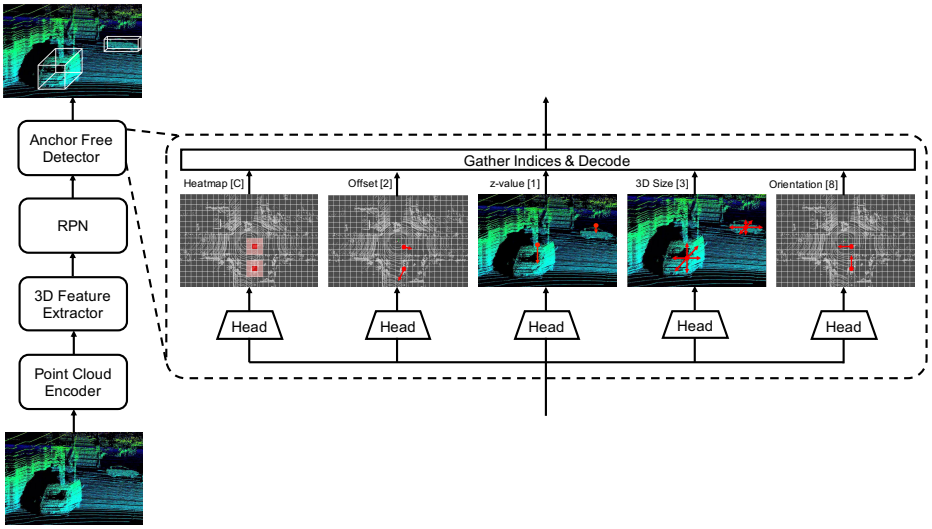

Figure 2: The framework of anchor-free one-stage 3D detection (AFDet) system. The whole pipeline consists of the Point Cloud Encoder, 3D Feature Extractor and Region Proposal Network (RPN), and the Anchor-free Detector. The number in the brackets indicates the output channels in the last convolution layer. C is the number of categories used in the detection.

Gather Indices and Decode. From the resultant keypoint heatmap, we can easily decode the center $(\hat{x}, \hat{y})$ of each object, along with the local offset correcting the discretization error. Other information such as orientation and dimension can be obtained from the regression results [2].

3. Methods

The network architecture has been explained in details in the above section. In this section, we first introduce our input format and data augmentation strategies. Then we explain a novel painting method using 2D detection, test time augmentation, and model ensemble methods.

3.1. Input and Data Augmentation

In light of nuScenes Dataset [1], we accumulate LiDAR sweeps to utilize temporal information and to densify the LiDAR point cloud. To distinguish points from different sweeps, time lag Δt is attached to the point cloud data, which is the timestamp difference between the non-keyframe sweep and the keyframe sweep. In Waymo challenge, we use past four frames combined with the current frame as our input point cloud. A Detailed ablation study can be found in Table 2.

Waymo Open Dataset [15] provides five kinds of LiDAR sweeps. To fully utilize the information, we combine all the first and second returns point cloud generated by all five LiDARs. Specifically, considering frame densification and painting, our input format should be $(x, y, z, intensity, [...painted\_scores...], \Delta t)$, with Δt the time lag as mentioned above.

We use data augmentation strategy following [18, 8]. First, we generate an annotation database containing labels and associated point cloud data. During training, we randomly select 6, 8 and 10 ground truth samples for vehicle, pedestrian and cyclist respectively, and place them into the current frame. To be compatible to painting [17], we paint point cloud data associated with the augmented samples according to their class labels. Second, we do randomly flipping along z-axis [19], global rotation following $\mathcal{U}\left(-\frac{\pi}{4}, \frac{\pi}{4}\right)$, global scaling following $\mathcal{U}\left(0.95, 1.05\right)$ and global translation along x, y, z-axis following $\mathcal{U}\left(-0.2m, 0.2m\right)$ [21, 18, 8].

3.2. Painting

Inspired by PointPainting [17], two painting methods are implemented in our solution to fuse camera information with LiDAR point cloud.

First, the 2D bounding boxes are employed to paint training, validation and testing data. Points are projected from the vehicle coordinate to the corresponding image frame. If a certain point falls inside a 2D bounding box, the classification information will be included in its attributes with additional channels. Otherwise, zeros will be filled in additional channels. The training set is painted with ground truth 2D bounding boxes, while during inference painting uses 2D detections produced by a single Cascade R-CNN based detector used in our 2D detection track.

Second, we use semantic segmentation information to paint point clouds. We start with a WideResNet38 pretrained on ImageNet [7] and finetune the network on

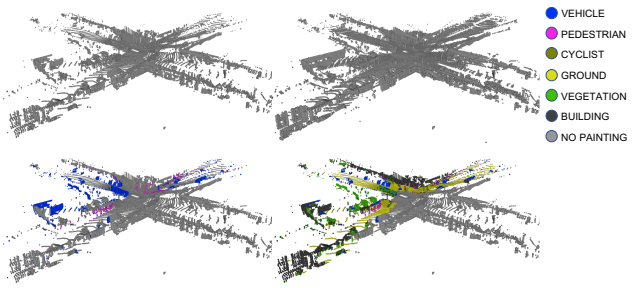

Figure 3: Point cloud with different kinds of augmentation. The top left image shows the raw point cloud. The top right image shows point cloud densified with [-4, +0] frames. The bottom left image shows point cloud painted with 2D detection boxes. The bottom right image shows point cloud painted with semantic segmentation. Note that only half of the points are painted in two bottom images because rear-view images are unavailable.

KITTI [3]. Only around half of the points can be painted in a frame because Waymo Open Dataset only provides images from front and side cameras. The improvements of introducing point painting can be seen in Table 2.

Models	Frames	Painting		L2	
		2D Box	Seg	mAP	mAPH
AFDet-B1	$[-0, +0]$	✗	✗	60.87	56.50
AFDet-B1	$[-4, +0]$	✗	✗	65.71	62.22
AFDet-B1	$[-4, +0]$	✓	✗	67.88	64.77
AFDet-B1	$[-4, +0]$	✓	✓	68.03	65.02

Table 2: 3D detection results of single model on validation set of the 3D detection track. Models are trained on 1/20 of training set and tested on 1/10 of validation set.

3.3. Further Improvements

Test Time Augmentation. We perform several different test-time augmentations, including point cloud rotation around pitch, roll and yaw axis, point cloud global scaling and point cloud translation along z-axis, which is similar to the data augmentation in the training process. Then, a 3D variant version of weighted box fusion [14] is used to ensemble all the augmented results. We use $\pm 0.5°$ for both pitch and roll rotation, [$0°$, $\pm 22.5°$, $\pm 45°$, ,$\pm 135°$,$\pm 157.5°$, $180°$] for yaw rotation, [0.95, 1.05] for global scaling, and $\pm 0.2m$ for translation along z-axis. According to our experiments, it turns out that the rotation yaw is the most effective one among other augmentation methods. Therefore, we combine the rotation yaw with all other methods.

Dataset	Thres.	Vehicle	Pedestrian	Cyclist
3D Detection	θ_{IoU}	0.80	0.70	0.65
	θ_{s1}	0.10	0.15	0.25
	θ_{s2}	0.03	0.03	0.03
Domain Adaptation	θ_{IoU}	0.80	0.70	0.65
	θ_{s1}	0.10	0.15	0.25
	θ_{s2}	0.05	0.05	0.05

Table 3: Thresholds used in the final submission. θ_{IoU}, θ_{s1} and θ_{s2} are box associating IoU threshold in ensemble, box skipping threshold used in ensemble and box skipping threshold.

Ensemble Method and Naive Grid Search. As it is mentioned above, we use a 3D version of weighted boxes fusion [14] to ensemble different models with test time augmentation. Based on the similarity of our model performance, we assign equal weights to AFDet-B1, AFDet-B2 and AFDet-B3. We replace 3D IoU with BEV IoU to accelerate the ensemble process for 3D box association. A naive grid search is conducted to jointly select the best thresholds used in ensemble and submission. We search on three variables including θ_{IoU} (box associating IoU threshold in ensemble), θ_{s1} (box skipping threshold used in ensemble) and θ_{s2} (box skipping threshold for submission). Search spaces for θ_{IoU}, θ_{s1} and θ_{s2} are $[0.40, 0.80]$, $[0.00, 0.25]$, $[0.01, 0.20]$ respectively. Search intervals for θ_{IoU}, θ_{s1} and θ_{s2} are 0.05, 0.05 and 0.01 respectively. θ_{s2} is searched from 0.01 instead of 0.00 to avoid the delay caused by evaluation. All grid searches are conducted on 1/10 of the validation set. We use different thresholds for different classes in the submission. Details can be found in Table 3.

4. Experiment Settings

We build our detector based on the Det3D framework [22]. To lower the GPU memory usage, We train our detector within range [(-76.8, 76.8), (-51.2, 51.2), (-1, 3)] respect to x, y, z-axis, while enlarging the detection range to [(-80, 80), (-80, 80), (-1, 3)] at inference. The max number of objects is set to 300. For all of our models, we set max point per voxel to 5, max voxel num to $1,000,000$. For the offset regression head, we use $r = 2$ as default to regress a square area with side length 5. We use max pooling with the kernel size 3×3, stride 1, and apply AND operation between the feature map before and after the max pooling to get the peaks of the keypoint heatmaps at the inference stage. So we don't need NMS to suppress overlapped detections. The weight we use for different sub-losses are $\lambda_{off} = 1.0$, $\lambda_z = 1.5$, $\lambda_{size} = 0.3$ and $\lambda_{ori} = 1.0$. All parameters we list here are their default values unless explicitly stated otherwise.

Models	VEHICLE		PEDESTRIAN		CYCLIST		ALL	
	L2 mAP	L2 mAPH	L2 mAP	L2 mAPH	L2 mAP	L2 mAPH	L2 mAP	L2 mAPH
HorizonLiDAR3D (Ours)	**78.23**	**77.83**	**79.32**	**76.50**	**77.91**	**76.98**	**78.49**	**77.11**
PV-RCNN	73.69	72.23	73.98	70.16	72.38	71.16	73.35	71.52
TS-LidarDet	72.65	72.12	68.10	59.32	66.55	65.16	69.10	65.53
Simple Baseline v2	66.44	65.91	66.00	60.93	65.13	64.10	65.84	63.65
Det3D-Waymo-3D-FS-VS	66.03	65.11	66.38	57.83	67.60	66.18	66.67	63.04

Table 4: Top five submissions of Waymo Open Dataset Challenge on 3D detection track.

Models	VEHICLE		PEDESTRIAN		CYCLIST		ALL		DOMAIN GAP (ALL)	
	L2 mAP	L2 mAPH	L2 mAP	L2 mAPH	L2 mAP	L2 mAPH	L2 mAP	L2 mAPH	L2 mAP	L2 mAPH
HorizonLiDAR3D (Ours)	**60.39**	**60.04**	**64.00**	**61.63**	**87.55**	**86.79**	**70.65**	**69.49**	**7.84**	**7.62**
PV-RCNN-DA	59.67	59.08	48.27	45.74	28.31	27.50	45.42	44.11	27.93	27.41
Simple Baseline v2	48.47	47.92	46.13	43.21	18.84	18.71	37.81	36.62	28.03	27.02
Det3D-Waymo-DA	50.07	49.33	43.51	37.90	4.75	4.66	32.78	30.63	33.89	32.41
Simple Baseline	45.50	44.77	45.99	43.12	1.53	1.48	31.01	29.79	-	-

Table 5: Top five submissions of the domain adaptation track. Domain gap is the difference between 3d detection track and domain adaptation track.

Models	Test Time Augmentation	Naive Grid Search	L2 mAP	L2 mAPH
AFDet-B1	✗	✗	68.03	65.02
AFDet-B2	✗	✗	67.05	64.29
AFDet-B3	✗	✗	68.91	65.24
B1 + B2	✗	✗	67.97	65.71
B1 + B3	✗	✗	69.72	67.16
B1 + B2 + B3	✗	✗	71.08	68.96
B1 + B2 + B3	✓	✗	73.65	72.14
B1 + B2 + B3	✓	✓	75.21	73.63

Table 6: 3D detection ensemble results on validation set of the 3D detection track. Models are trained on 1/20 of training set and tested on 1/10 of validation set.

We use AdamW [10] optimizer with one-cycle policy [4]. We set learning rate max to 3×10^{-3}, division factor to 2, momentum ranges from 0.95 to 0.85, fixed weight decay to 0.01 to achieve convergence. To quickly verify our idea, we sample the training set every 20 frames, and validation set every 10 frames according to their timestamps. Unless we explicitly indicate, all models used for ablation study, including Table 2 and Table 6, are trained on 1/20 training data, validation and tested on 1/10 validation data. In the final submission, we first train 60 epochs on 1/20 training data and finetune 10 epochs on the whole trainval data for AFDet-B1 and AFDet-B2. We only train 60 epochs on 1/20 training data and finetune 3 epochs on the whole trainval data for AFDet-B3. Besides, we adopt half-precision (FP16) [11, 13] for B3 model to lower GPU memory usage and speed up the training process. We ensemble AFDet-B1, AFDet-B2 and AFDet-B3 for 3D detection and AFDet-B1 and AFDet-B2 for domain adaptation in our final submission. From our submission result, there is $1\% \sim 2\%$ gap between the test set and validation set. We can speculate that the test set is easier than the validation set.

	3D training		3D validation		Domain training		Domain validation	
	Num	Per	Num	Per	Num	Per	Num	Per
Object	6.3M	100%	1.6M	100%	260K	100%	45.2K	100%
Vehicle	4.2M	67.15%	1.1M	68.42%	242K	93.15%	43K	95.11%
Pedestrian	2.0M	32.07%	0.5M	30.80%	17.6K	6.78%	2.2K	4.89%
Cyclist	49K	0.78%	12.3K	0.78%	184	0.07%	0	0

Table 7: Object number and percentage for each class of the 3D detection and the domain adaptation track.

5. Results

3D Detection. To study the effectiveness of each module used in our solution, we conduct ablation experiments on the 1/10 validation set as shown in Table 2 and Table 6. We also show the leaderboard of the Waymo OpenDataset Challenge - 3D Detection track in the Table 4.

Domain Adaptation. Waymo Open Dataset [15] also provides additional data collected from different locations with only a small subset annotated to study the effectiveness of domain diversity. We apply directly our models trained for the 3D track to the domain adaptation data. For the final submission, AFDet-B1 and AFDet-B2 were applied with test time augmentation and the results were merged by

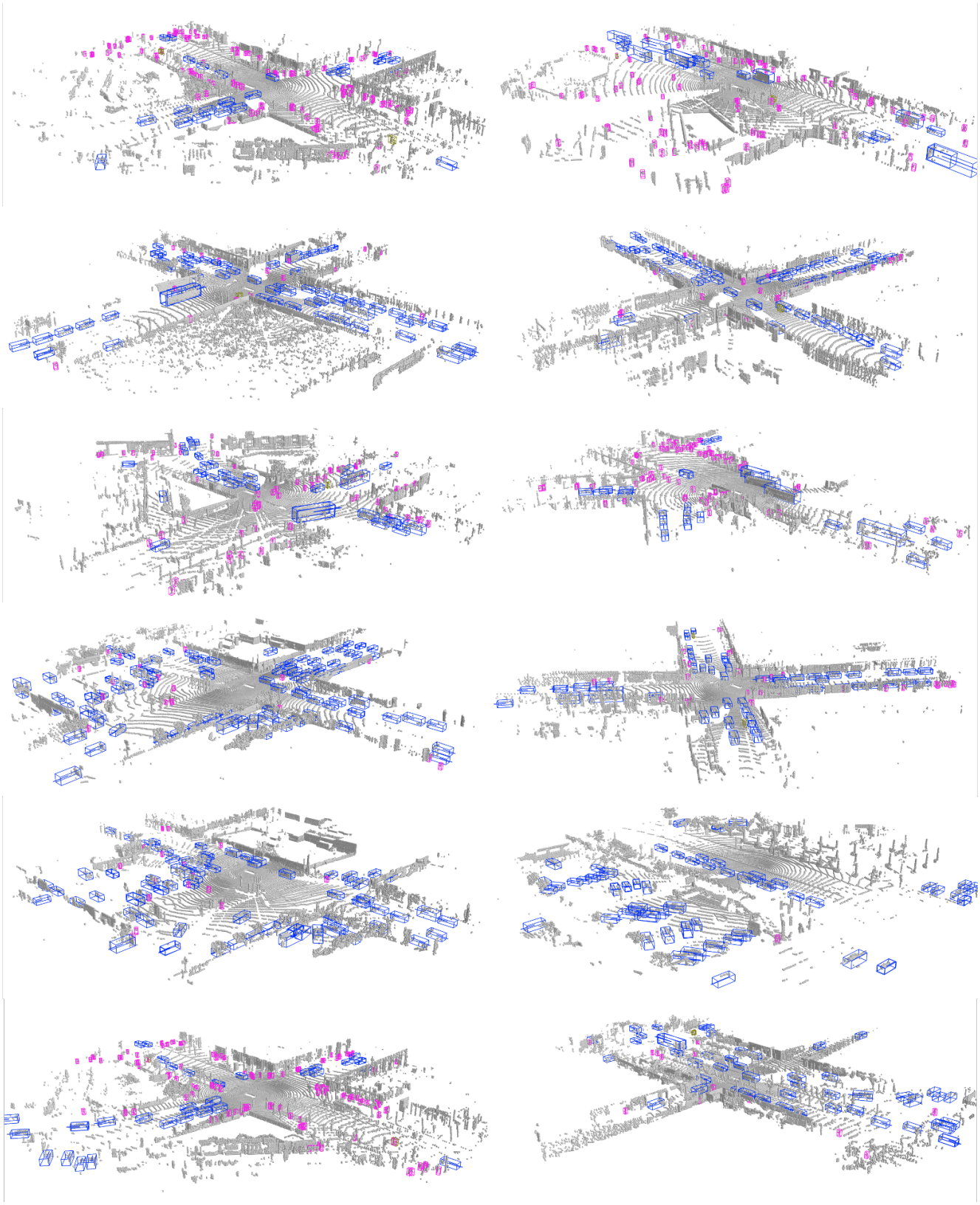

Figure 4: Examples of results on testing set of the 3D detection track, with bounding boxes filtered at 0.15 during visualization.

naive grid search. θ_{IoU}, θ_{s1} and θ_{s2} for our submission is listed in Table 3. The results show that our solution outperforms other methods in domain adaptation by a big margin as shown in Table 5.

This result demonstrated the benefits of using anchor-free model and its capability of generalizing to a different dataset or domain. Another reason might be due to the fact that most other methods finetuned their models using annotated domain adaptation data, which is highly imbalanced. As shown in Table 7, we compared the statistics computed from 3D detection and domain adaptation datasets respectively. For example, only 0.07% of the objects are cyclists in the domain adaptation training set, which is 10× less than the 0.78% in the 3D detection training set. Moreover, there are only 184 cyclists with more than one LiDAR point in the training and validation set which would cause problems for both training or testing. The detection or mis-detection of one cyclist could result in a fluctuation of accuracy during validation as there are only a few of them in the entire dataset. Fine-tuning on the domain adaptation dataset might comprise the performance of the resulting detector, especially for rare classes such as cyclist.

6. Conclusion

A high-performance 3D detection framework is proposed and achieved the 1st place on 3D detection track and the domain adaptation track in Waymo Open Dataset Challenge at CVPR 2020.

References

[1] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 3

[2] Runzhou Ge, Zhuangzhuang Ding, Yihan Hu, Yu Wang, Sijia Chen, Li Huang, and Yuan Li. Afdet: Anchor free one stage 3d object detection. In *CVPR Workshops*, 2020. 1, 2, 3

[3] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 4

[4] Sylvain Gugger. The 1cycle policy. https://sgugger.github.io/the-1cycle-policy.html, 2018. 5

[5] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *CVPR*, 2020. 2

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 3

[8] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 2, 3

[9] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *CVPR*, 2017. 2

[10] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 5

[11] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017. 5

[12] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *CVPR*, 2017. 2

[13] NVIDIA. Apex. https://github.com/NVIDIA/apex. 5

[14] Roman Solovyev and Weimin Wang. Weighted boxes fusion: ensembling boxes for object detection models. *arXiv preprint arXiv:1910.13302*, 2019. 4

[15] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. *arXiv preprint arXiv:1912.04838*, 2019. 1, 3, 5

[16] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, 2020. 2

[17] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *CVPR*, 2020. 3

[18] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2, 3

[19] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 2018. 3

[20] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 2

[21] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 2, 3

[22] Benjin Zhu and Bingqi Ma. Det3d. https://github.com/poodarchu/Det3D. 2, 4