Li3DeTr: A LiDAR based 3D Detection Transformer
Supplementary Material

Gopi Krishna Erabati and Helder Araujo
Institute of Systems and Robotics
University of Coimbra, Portugal
{gopi.erabati, helder}@isr.uc.pt

1. Implementation Details

1.1. Model Details

Our model mainly consists of two modules: CNN backbone and transformer encoder-decoder. We test with SparseConv [6] and PointPillars [7] with SECOND [14] as feature extraction network. nuScenes dataset: The point cloud range is set to $[-51.2m, 51.2m] \times [-51.2m, 51.2m] \times [-5.0m, 3.0m]$. We employ SparseConv with voxelization resolution set to $[0.1m, 0.1m, 0.2m]$ and four blocks of [3, 3, 3, 2] 3D sparse convolutions with [16, 32, 64, 128] dimensions and PointPillars with voxelization resolution set to $[0.2m, 0.2m, 8m]$ and three blocks of [3, 5, 5] convolutional layers with [64, 128, 256] dimensions. KITTI dataset: The point cloud range is set to $[0.0m, 71.4m] \times [-40.0m, 40.0m] \times [-3.0m, 1.0m]$. We employ SparseConv with voxelization resolution set to $[0.05m, 0.05m, 0.1m]$ and four blocks of [1, 3, 3, 3] 3D sparse convolutions with [16, 32, 64, 64] dimensions and PointPillars with voxelization resolution set to $[0.16m, 0.16m, 4.0m]$ and three blocks of [3, 5, 5] convolutional layers with [64, 128, 256] dimensions. We use FPN [8] to transform the features and obtain four multi-scale local LiDAR feature maps with 256 channel dimension. For the transformer encoder, we use $L_{enc} = 2$ encoder layers with multi-scale deformable self-attention [17] with [256, 256] dimensions, 8 heads, 4 levels and 4 sampling points for each query and in each head. For the decoder, we use $L_{dec} = 6$ decoder layers with hidden dimension 256 and 900 object queries.

1.2. Training Details

We implement our model in PyTorch [10] based on open-sourced MMDetection3D [3]. We train Li3DeTr network with AdamW optimizer with initial learning rate of 2×10^{-4} and weight decay of 10^{-2}. Cosine Annealing is set as learning rate scheduler with 2K warm up iterations and with minimum learning rate of 2×10^{-6}. The backbone is initialized with pretrained network [13, 14] on the same dataset. The model is trained for 30 epochs on two RTX 3090 GPUs with a batch size of 4. During test time, we take the top 300 predictions with highest category score as final predictions and we do not use any NMS.

2. More Quantitative Results

The performance of our Li3DeTr network compared with other state-of-the-art approaches on the nuScenes *val* dataset in terms of mAP and NDS is shown in Table 1. Our network outperforms all the methods in terms of mAP and stands second in terms of NDS on nuScenes *val* set. Although VISTA [4] uses NMS to remove redundant boxes, our method achieves improved performance in 3D object detection without NMS. Moreover VISTA is a plug and play module, but our approach is a *standalone* network for 3D object detection. Our network surpassed the state-of-the-art transformer based NMS-free network Object-DGCNN [13] by 2.8 % mAP and 1.6 % NDS.

Table 1: Comparison of recent works in terms of mAP and NDS on the nuScenes [1] *val* set. The scores in <u>underline</u> represent *second* position in the corresponding metrics.

Method	NDS ↑	mAP ↑	NMS
CenterPoint [15]	64.8	56.4	✓
HotSpotNet [2]	66.0	59.5	✓
VISTA-OHS [4]	**68.1**	<u>60.8</u>	✓
Object-DGCNN (voxel) [13]	66.0	58.6	✗
Ours (voxel)	<u>67.6</u>	**61.4**	✗

The performance of our network compared with state-of-the-art approaches on the KITTI [5] *val* dataset for *pedestrian* and *cyclist* categories is shown in Table 2. The results of our approach on the *car* category shows competitive performance with state-of-the-art approaches as shown in Table 2 (in main paper). However, our method could not achieve state-of-the-art performance on *pedestrian* and *cyclist* categories. The number of object samples in the

Table 2: Comparison of recent works in terms of AP_{3D} and AP_{BEV} detection on KITTI [5] *val* set. We list results for *pedestrian* and *cyclist* category for *easy*, *moderate* and *hard* samples with IoU=0.5. The scores in underline represent second rank in the corresponding metric.

Method	Cyclist						Pedestrian						NMS
	AP_{3D}			AP_{BEV}			AP_{3D}			AP_{BEV}			
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	
F-PointNet [11]	77.1	56.4	53.3	**81.8**	60.0	56.3	<u>70.0</u>	<u>61.3</u>	<u>53.5</u>	**72.3**	**66.3**	**59.5**	✓
VoxelNet [16]	67.1	47.6	45.1	74.4	52.1	50.4	57.8	53.4	48.8	<u>65.9</u>	<u>61.0</u>	<u>56.9</u>	✓
PVCNN [9]	**81.4**	59.9	56.2	-	-	-	**73.2**	**64.7**	**56.7**	-	-	-	✓
Complex-YOLO [12]	68.1	58.3	54.3	72.3	<u>63.3</u>	<u>60.2</u>	41.7	39.7	35.9	46.0	45.9	44.2	✓
PointPillars [7]	80.0	<u>62.6</u>	<u>59.5</u>	-	-	-	57.7	52.2	47.9	-	-	-	✓
SECOND [14]	<u>80.5</u>	**67.1**	**63.1**	-	-	-	56.5	52.9	47.7	-	-	-	✓
Ours	77.0	60.0	58.0	<u>80.5</u>	**64.0**	**60.3**	51.1	44.1	40.0	56.5	50.0	45.2	✗

Table 3: Performance of our network in terms of Average Precision (AP) by object category on the nuScenes *test* set. CV - Construction Vehicle, Motor - Motorcycle, Ped - Pedestrian, TC - Traffic Cone. $*$: MMDetection3D [3] implementation. The scores in green indicate the increase in performance with respect to scores in underline.

Method	Car	Truck	Trailer	Bus	CV	Bicycle	Motor	Ped	TC	Barrier	mAP
CenterPoint [15] $*$	84.6	51.0	53.2	60.2	17.5	28.7	53.7	83.4	76.7	70.9	58.0
VISTA [4]	84.4	55.1	54.2	63.7	25.1	45.4	70.0	82.8	78.5	71.4	**63.0**
Obj-DGCNN [13]	<u>84.0</u>	48.5	<u>54.0</u>	<u>57.5</u>	<u>25.2</u>	<u>32.2</u>	64.5	81.7	73.8	65.6	58.7
Ours	85.6 ↑ 1.6	50.0	56.5 ↑ 2.5	60.3 ↑ 2.8	30.3 ↑ 5.1	38.3 ↑ 6.1	65.9	83.0	75.5	68.0	61.3

training split of KITTI [5] dataset for *car* category is 83%, whereas for *pedestrian* and *cyclist* categories is 13% and 4% respectively. Due to very less number of object samples during training, our transformer network could not achieve competitive results on *pedestrian* and *cyclist* categories. In addition to this, as described in § 4.2.3 (in main paper), due to quantization of point clouds and downsampling of feature maps, our network finds it difficult to detect small size objects.

3. More Analysis and Ablation Studies

3.1. More Analysis on Object Category

The performance of our network in terms of Average Precision (AP) for each object category compared to other state-of-the-art networks on the nuScenes [1] *test* dataset is shown in Table 3. Similar to results on nuScenes *val* dataset, the global voxel features by multi-scale deformable attention [17] and our novel cross-attention block significantly improves the AP of large size objects like trailer, bus, construction vehicle compared with Object-DGCNN [13]. However, the performance on smaller objects like pedestrian and barrier is worse due to quantization of point cloud and downsampling of feature maps in the backbone to increase the receptive filed which results in information loss.

3.2. More Analysis on Inference Time

The inference speed of our approach is compared with other state-of-the-art methods as shown in Table 4. Our approach not only achieves improvement in performance but also is faster than CenterPoint [15] and Object-DGCNN [13].

Table 4: Comparison of inference speed measured on a NVIDIA RTX 3090 GPU. $*$: MMDetection3D [3] implementation

Method	CenterPoint $*$	Object-DGCNN	Ours
FPS	5.1	5.6	**7.1**

3.3. Ablation on Detection Layers

The performance of our network in terms of mAP, NDS and other TP metrics on the nuScenes [1] dataset for different number of layers in the transformer decoder is shown in Table 5. The hypothesis that we iteratively refine the object queries after each decoder layer to significantly improve the performance of the model is proved to be correct as shown in Table 5. The performance significantly improves as we increase the number of decoder layers in the network. However, the performance of the model does not improve much after 6 decoder layers, so we fix number of decoder layers

(L_{dec}) to 6.

Table 5: Performance of our network on the nuScenes *val* set by number of decoder layers. Among TP metrics only mATE, mASE, mAOE are shown.

Layer	NDS ↑	mAP ↑	mATE ↓	mASE ↓	mAOE ↓
1	63.1	54.5	37.9	27.1	30.1
2	65.7	58.4	34.2	26.6	29.5
3	67.0	60.6	32.9	26.6	28.7
4	67.4	61.2	32.8	26.5	28.6
5	67.5	61.3	32.7	26.2	28.5
6	**67.6**	**61.4**	**32.7**	**26.1**	**28.5**

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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[2] Qi Chen, Lin Sun, Zhixin Wang, Kui Jia, and Alan Yuille. Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots. In *European conference on computer vision*, pages 68–84. Springer, 2020.

[3] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. `https://github.com/open-mmlab/mmdetection3d`, 2020.

[4] Shengheng Deng, Zhihao Liang, Lin Sun, and Kui Jia. Vista: Boosting 3d object detection via dual cross-view spatial attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8448–8457, 2022.

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[6] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018.

[7] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[8] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[9] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[11] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

[12] Martin Simony, Stefan Milzy, Karl Amendey, and Horst-Michael Gross. Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[13] Yue Wang and Justin M Solomon. Object dgcnn: 3d object detection using dynamic graphs. *Advances in Neural Information Processing Systems*, 34, 2021.

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

[15] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[16] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018.

[17] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.