Self-Attention based Feature Extractors for 3D Object Detection in Point Clouds

Arulkumar Subramaniam[1⋆], Ashish Vaswani[2], and Niki Parmar[2]

[1] Indian Institute of Technology Madras, India
aruls@cse.iitm.ac.in
[2] Google Brain
{avaswani,nikip}@google.com

Abstract. Object detection in 3D point clouds typically follows a two-stage pipeline of extracting object proposals followed by classification and regression. Existing models such as PointNet, PointNet++, StarNet use a series of *point-wise* linear transformations to learn features from point clouds. However, for smaller objects with fewer points such as pedestrians, capturing larger context around them could provide more information for accurate detection, which point-wise features fail to achieve. Self-attention has been widely used as a computational primitive across several modalities and has been shown to capture both local and global dependencies. In this work, we use a self-attention based featurizer to model local interactions within proposal neighborhoods and global interactions between proposals. This featurizer outperforms the previous point-based featurizers on the large scale Waymo 3D object detection on vehicles and pedestrians, achieving significant gains on pedestrian detection (1.8% mAP). Our ablations show that modeling both global and local interactions are important, and provide complementary gains.

Keywords: 3D object detection, Self attention, Autonomous driving

1 Introduction

Object detection in 3D point clouds is an important problem in domains such as Autonomous Driving [14, 12], Robot Navigation [11] and Human-Computer interaction. State-of-the-art (SoTA) approaches for object detection in 3D point clouds [9, 5, 10, 1, 4, 6] adopt a two-stage detection pipeline: "object proposal" followed by "classification & regression". Among them, StarNet [5], uses a simple sampling strategy to extract object proposals and achieves strong results on the recently introduced large scale Waymo 3D object detection dataset[12].

The object proposals, that comprise of object centers and a neighborhood of points around the centers, are passed through a point-based featurizer to learn point-wise representations that are eventually aggregated to compute object proposal representations. Similar to point based featurizers such as [7, 8, 3], StarNet uses a series of point-wise linear transformations to compute representations of points within each object proposal. In this work, we use the simple strategy of replacing point-based featurizers with self-attention to model interactions between the points in a neighborhood, and self-attention between object proposals

⋆ Work done during an internship at Google Brain

to model global interactions between objects. In the next section, we describe where we place our featurizers within the StarNet architecture.

2 Self-Attention featurizer for StarNet

We extend the object detection framework from Ngiam *et al.* [5] (Fig. 1) by replacing their local point-wise featurizer with a self-attention based featurizer.

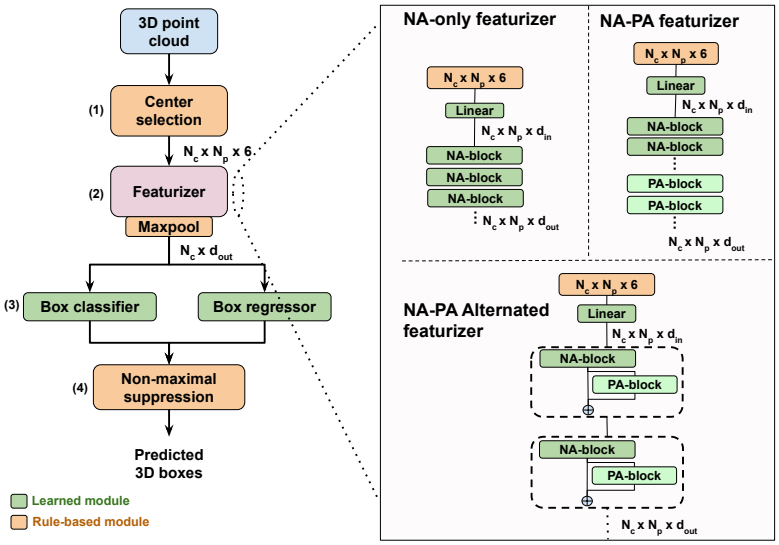

Fig. 1: Detection framework from Ngiam *et al.*[5] (Left) & Three configurations of Self-attention featurizer (Right). The detection framework consists of four core components: 1) *Center selection*, 2) *Featurizer*, 3) *Bounding Box Regression/Classification* 4) *Non-maximal suppression*

The featurizer takes the object centers and the points around each center as input. We denote the number of centers as N_c, and the number of points around each center as N_p . The input is then $(N_c \times N_p \times 6)$, where the 6 channels are the euclidean co-ordinates (x,y,z) along with range, elongation, intensity. This input is first passed through a linear layer to form a $N_c \times N_p \times d_{in}$ tensor, where d_{in} is the input feature dimension. The output of the featurizer is $N_c \times d_{out}$ where d_{out} is the output feature dimension.

The self-attention featurizer comprises of a series of self-attention blocks followed by a max pooling layer. Each block consists of a multi-head self-attention layer[13] followed by a feed-forward layer. The details about the block can be found in the supplementary material.

We apply self-attention at two granularities:

Neighborhood self-attention (NA): We apply the self-attention block for points within each center to learn local shape and context cues. In NA-block, a shared SA-block is applied on neighborhood point features $(N_p \times d_{in})$ of each center.

Proposal self-attention (PA): To model relationships between object proposals with self-attention, we first max-pool the $N_p \times d_{in}$ representations of points

within each proposal neighborhood resulting in a single d_{in} dimensional representation for each of the N_c centers and then apply self-attention between these centers. To inject this information back into the point representations, for each proposal, we concatenate the vector from the result of the proposal attention with the point representations of that proposal and project the concatenated vectors back to d_{in} dimensions with a $2d_{in} \times d_{in}$ linear transformation.

By arranging NA-blocks and PA-blocks, we formulate three different configurations of featurizer namely, 1) NA-only featurizer, 2) NA-PA featurizer, and 3) NA-PA Alternated featurizer as shown in (Fig. 1 Right).

3 Experiments

We use the **Waymo Open Dataset(WOD)[12]**, a large-scale dataset consisting frames of 1000 segments of 20 seconds LiDAR measurements (rate of 10 Hz). In our experiments, we use two classes: Pedestrian and Vehicle. The performance is evaluated using the measure of mean average precision (mAP) on Waymo validation set. Details about our training setup and hyper-parameters can be found in the supplementary section.

3.1 Results

Models	#params	#GFlops	Pedestrian mAP	Vehicle mAP
PointPillars[2]	-	3700	62.1	57.2
MVF[15]	-	-	65.3	**62.9**
StarNet[5]	1.483 M	136.56	66.8	53.7
$N_{NA} = 4$	0.316 M	128.7	67.83	53.91
$N_{NA} = 10$	0.467 M	317.15	**69.05**	59.2
$N_{NA} = 4, N_{PA} = 4$	0.421 M	130.08	67.64	58.09
$N_{alternate} = 4$	0.421 M	131.8	68.3	58.66

Table 1: Comparisons on Waymo validation set. $N_{NA}, N_{PA}, N_{alternate}$ are number of NA-, PA-, Alternated NA and PA blocks respectively.

In Table 1, we see that by replacing the StarNet featurizer with only 4 NA-blocks, we outperform the baseline significantly on pedestrian detection with fewer FLOPS and an order of magnitude fewer parameters. Scaling to 10 NA-blocks, we achieve +3.25% and +5.5% than StarNet on pedestrian and vehicle detection respectively. However, combining NA and PA gives us better FLOP-accuracy trade-off, outperforming the baseline on both pedestrian and vehicle detection. We find that an effective strategy for mixing NA and PA blocks is to alternate between them as depicted in Fig. 1 (Right).

3.2 Ablations

Increasing number of attention blocks improves performance Scaling the NA-only featurizer by increasing the number of attention blocks consistently improves mAP on both vehicles and pedestrians (Fig. 2 Left). Improvements on pedestrian class flattens out after 8 attention blocks.

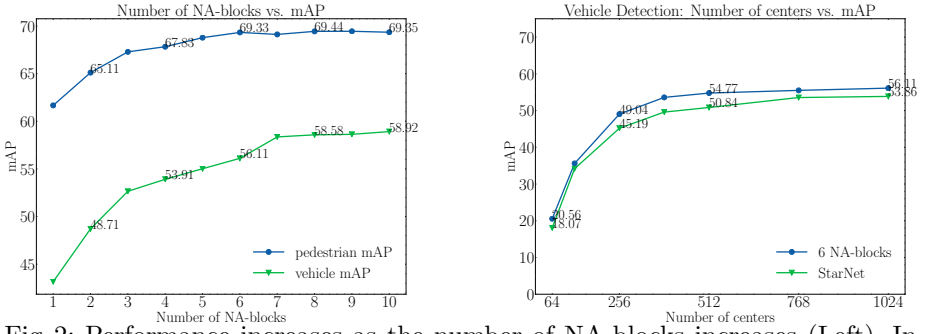

Fig. 2: Performance increases as the number of NA-blocks increases (Left). Increasing center proposals improves the coverage of 3D scene, thus improves the detection performance (Right)

Increasing number of centers To analyze the influence of center proposal stage, we experiment with number of centers used for prediction. Fig. 2 (Right) reveals that more number of centers leads to higher coverage of the scene and thus increased performance.

Comparison of NA and PA blocks NA with 10 blocks achieves good performance, however, adds a significant computational overhead. With respect to computation complexity, NA-only featurizer with 4 NA-blocks outperforms StarNet with better FLOP-accuracy trade-offs (Fig. 3). PA possess negligible parameter and computation overhead. We find the best strategy is to combine 4 PA-blocks with 4 NA blocks, improving the performance of vehicle detection by 4.18% (Table 2).

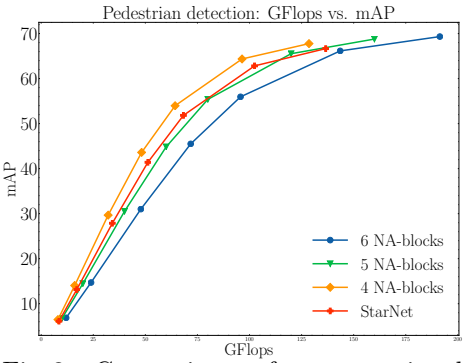

Fig. 3: Comparison of computational complexity over baseline[5]. We observe that NA-only featurizer with 4 blocks achieves better FLOP-accuracy trade-offs than StarNet.

Increasing number of attention heads We analyze the behavior of attention heads following the practice from [13]. Increasing heads to 4 improves performance while adding minimal computation as shown in (Table 3).

N_{NA}	N_{PA}	#params	#G Flops	Pedestrian mAP	Vehicle mAP
4	0	0.32M	128.71	**67.83**	53.91
4	2	0.37M	129.04	67.3	55.66
4	4	0.42M	130.08	67.64	**58.09**

Table 2: Ablation on number of PA-blocks. N_{NA}, N_{PA} are number of NA-, PA-blocks respectively.

N_{NA}	N_h	Vehicle mAP
6	1	54.06
6	2	55.62
6	4	**56.11**

Table 3: Ablation on number of attention heads

References

1. Chen, X., Kundu, K., Zhu, Y., Berneshawi, A.G., Ma, H., Fidler, S., Urtasun, R.: 3d object proposals for accurate object class detection. In: Advances in Neural Information Processing Systems. pp. 424–432 (2015)
2. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 12697–12705 (2019)
3. Liu, Y., Fan, B., Meng, G., Lu, J., Xiang, S., Pan, C.: Densepoint: Learning densely contextual representation for efficient point cloud processing. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5239–5248 (2019)
4. Naiden, A., Paunescu, V., Kim, G., Jeon, B., Leordeanu, M.: Shift r-cnn: Deep monocular 3d object detection with closed-form geometric constraints. In: 2019 IEEE International Conference on Image Processing (ICIP). pp. 61–65. IEEE (2019)
5. Ngiam, J., Caine, B., Han, W., Yang, B., Chai, Y., Sun, P., Zhou, Y., Yi, X., Alsharif, O., Nguyen, P., et al.: Starnet: Targeted computation for object detection in point clouds. arXiv preprint arXiv:1908.11069 (2019)
6. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from rgb-d data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
7. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
8. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems. pp. 5099–5108 (2017)
9. Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H.: Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10529–10538 (2020)
10. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–779 (2019)
11. Song, S., Lichtenberg, S.P., Xiao, J.: Sun rgb-d: A rgb-d scene understanding benchmark suite. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 567–576 (2015)
12. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., Anguelov, D.: Scalability in perception for autonomous driving: Waymo open dataset (2019)
13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
14. Yang, B., Luo, W., Urtasun, R.: Pixor: Real-time 3d object detection from point clouds. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 7652–7660 (2018)
15. Zhou, Y., Sun, P., Zhang, Y., Anguelov, D., Gao, J., Ouyang, T., Guo, J., Ngiam, J., Vasudevan, V.: End-to-end multi-view fusion for 3d object detection in lidar point clouds. In: Conference on Robot Learning. pp. 923–932 (2020)