Latent-Polar Transformer for LiDAR Point-Cloud 3D Object Detection

Manoj Bhat, Steve Han, Fatih Porikli
Qualcomm
5775 Morehouse Drive San Diego, CA USA.
{manobhat, shizhan, fporikli}@qti.qualcomm.com

Abstract

Object detection in 3-Dimentions with point clouds is widely used in the application of autonomous driving, Augmented Reality, and Virtual Reality. We address one of the challenges of reduction in latency and computation without sacrificing the detection accuracy in LiDAR based object detection while sustaining long-range precision. Latency is important because it's directly related to security and safety of human lives. We propose a novel streaming 3D object detector using polar space feature representation that can provide faster inference for the 3D object detection with rotating LiDAR devices. The proposed method can improve the object detection performance with streaming technique using the pseudo-image method for edge devices with limited memory. Our results in KITTI and Waymo dataset reveal that the learned model captures state-of-art detection accuracy with fast inference in the KITTI validation dataset with 94.7AP for cars in BEV detection.

1. Introduction

With cost reduction of the the LiDAR sensor for autonomous vehicles, the industry is re-considering utilization for level 3-4 of autonomy. Compared with camera, the LiDAR sensor can capture very precise 3D point-location information for 3D objects, such as car, pedestrian, cyclist etc. in farther-away range/distance. Modern research with Deep learning shows drastic improvement in 3D object detection in Point-clouds [13, 4, 2]. However the challenge in reduction of inference latency and computation resource without sacrificing detection accuracy remains uncharted. The faster the detection, more reduction in reaction time of autonomous driving system, avoiding the potential accidents. Simultaneously, the computation and algorithmic complexity is supposed to be small for application as there are myriad amount of models (camera, LiDAR, Radar etc.) running in parallel, for different tasks (object detection, segmentation, planning etc.).

There are many recent works to solve such challenge.

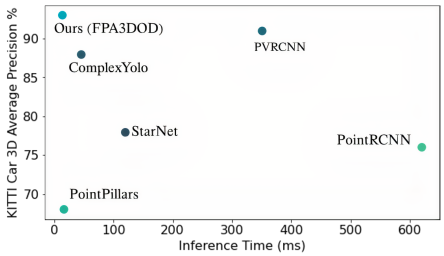

Figure 1: Inference time (ms)(Indicating average latency) vs 3D Car detection accuracy in terms of average precision evaluated on the KITTI validation dataset

To keep a good performance of detection accuracy, many early works proposed to use 3D convolution [1, 5], which can keep the 3D geometry information, but it is compute intensive and slow to process data. Hence a combination of standard 3D and 2D convolution was proposed to reduce the latency and computation [6, 14]. SECOND [13] proposed to use the 3D sparse convolution to speed up the training and inference time with less memory consumption. However, 3D sparse convolution depends on the external implementation, which is not supported by the popular deep learning libraries. Compared with the 3D sparse convolution, the 2D standard convolution can further reduce the latency and memory consumption. The PointPillars [4] proposed to use the PointNets to extract the 2D pseudo-image feature, and followed by the 2D standard convolution detection architecture, which can significantly increase the inference speed and reduce the computation resource. However, their detection accuracy fell in short compared with the methods based on 3D Sparse convolution.

- In this extended abstract, we propose an approach to extract the Polar space feature representation with PointNet followed by a transformer for 3D object detection. Reducing the memory consuming and latency, while maintaining long range detections.

- We demonstrate the performance of the method which

outperforms the state-of-the-arts in the publicly available dataset KITTI and Waymo.

2. Methodology

In this section we will describe our approach to detect 3D objects from a raw point cloud represented in Cartesian coordinate space. The architecture is very light-weight consisting of three modules: 1) Bird Eyes View Cartesian space voxelization 2) Cartesian to polar mapping 3) Transformer self-attention detection with prediction heads.

For a training dataset of N LiDAR scans $\{(P_i, R_i) \mid i = 1, \ldots, N\}, P_i \in \mathbb{R}^{n_i \times 4}$ where i isthe point-set containing n_i LiDAR points. (x, y, z) is the Cartesian coordinate of the point relative to the scanner. We are predicting bounding boxes over this point cloud with each bounding box represented as $(c_x, c_y, c_z, w, l, h, \theta, l_i)$ including the information of centers (c_x, c_y, c_z), dimensions (w, l, h), yaw as θ and class labels l_i for each class .

2.1. Pseudo-image extraction

For a particular frame of point-cloud, the point features are projected to a canvas using a learnable kernel in voxelized space. This avoids expensive 3D point processing and Nearest-Neighbor search.

This learnable feature extractor is a single PointNet in the encoder and we construct non-empty pillars[4] to create a tensor of size (D, P, N) where D is 9 dimensional without reflectance, P is the number of voxels and N is the number of points. The discretization is different for the both KITTI and Waymo. As KITTI dataset has annotations only towards the front of the LiDAR, the input is 360° pointcloud with only points corresponding to the camera frame. The voxels are then used to generate a (C, P) tensor after a max operation on PointNet layer where C is the number of channels. This tensor is scattered over a tensor of (C, H, W). The feature extractor follows the extraction of features to a reduced pseudo-image of size $H \times W$ with channels corresponding to the voxels at the z-dimentions.

Next the extracted pseudo-image of shape $x_{\text{img}} \in \mathbb{R}^{C \times H \times W}$ is discretized into the Polar space of ring-sector combination. If R is the ring-number, in the Polar space, A patch of shape $V \times V$ is computed to be of same dimensions for all the sector regions for the ring. A CNN transforms the features from these patches of extracted features into an embedding of shape $1 \times E$. For a particular shape of feature space, the number of rings can be 5-100 and for each such ring or sector, we have a separate CNN feature extractor.

2.2. Cartesian to polar mapping

. To produce a pseudo-image which is representative of the polar pattern of the data, we map the Latent feature space into a polar pseudo-image by a tensor transformation. This mapping is produced by tensor element-wise operation according the below equations. We define the spacing in the Latent feature space S_ϕ, S_r, and Polar space as S_{ϕ_p}, S_{r_p} where,

$$S_\phi = \frac{2\pi}{H_p}, S_r = \frac{R_{max}}{W_p} \tag{1}$$

$$\phi_x = \lfloor \frac{\phi}{S_\phi} \rfloor, r_y = \lfloor \frac{r}{S_r} \rfloor \tag{2}$$

Here H and W are the dimensions of the latent feature space which is transformed to the Polar space (H_p, W_p). R_{max} is the maximum radius of the polar space. Every tensor element in the latent space is remapped to the Polar space based on the equation 2 where ϕ_x is the indice over length and ρ is over the width, with $r = \sqrt{x_l^2 + y_l^2}$ and $\phi = \tan^{-1} \frac{y_l}{x_l}$ where x_l and y_l are Cartesian coordinates or the row, column indice in the pseudo image. The tensor values are filled in between over the missing elements using bi-linear interpolation over bijective mapping. Further, these tensor projections are fed into a collection of conv operators. The conv operators parse the tensor row-wise, equivalent to parsing each sectors, or column-wise equivalent to parsing each rings over all polar sectors into the same number of vectors. After this range-wise processing, the vectors are fed to the encoder and N detections are obtained at the decoder corresponding to the number of queries.

2.3. Transformer

Since we only need to output the 3D box properties with the corresponding classes, any transformer can be utilized from the self-attention paradigm. Here a standard transformer encoder architecture is used with inputs as sequence vectors in row-wise or column-wise after addition with positional encoding. In row-wise method with horizontal feature extraction, for sector-wise processing, the encoder would have K_{row} input embedding whereas K_{col} after vertical feature extractor as in Figure 3. A transformer with complete 360° input, would have $K_{row} \times K_{col}$ inputs. Our implemented architecture for both range-wise processing and 360° processing has 6 layers of encoder, 4 layers of decoder with N query inputs. The N is decided based on expected maximum number of predictions within a point-cloud frame. The final output feature vectors for set of predictions is fed to detection heads to regress the box properties.

The prediction head consist of a 5 layer MLP transforming the output last feature from the sequence of output of transformer. Each of the Cartesian coordinate and the dimensions with the $sin\theta$ and $cos\theta$ of angles are regressed with separate heads. This follows the method of training and inference using complex-euler angles for heading [7].

Our method follows a divide-and-conquer principle for dividing the polar space into K sectors. For maintaining

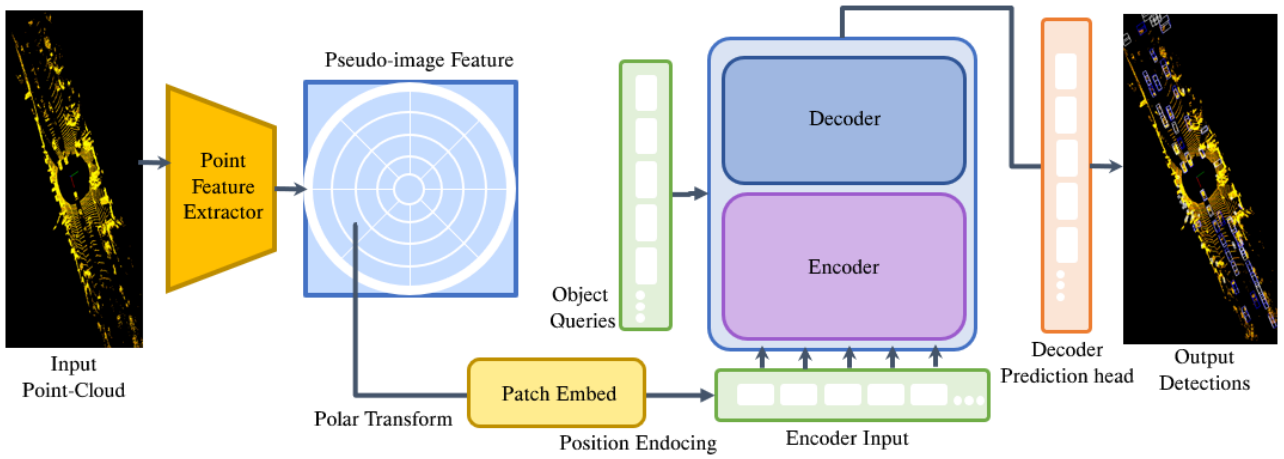

Figure 2: An overview of the proposed method. Point feature extractor extracts the features from raw-points by MLP[left]. The pseudo image is transformed to a polar latent feature tensor[middle]. After piece-wise feature extraction of features, the embeddings are fed to the transformer. The decoding heads output proposals. This allows learning detection on long-range distance, improving the object detection performance beyond 80m.

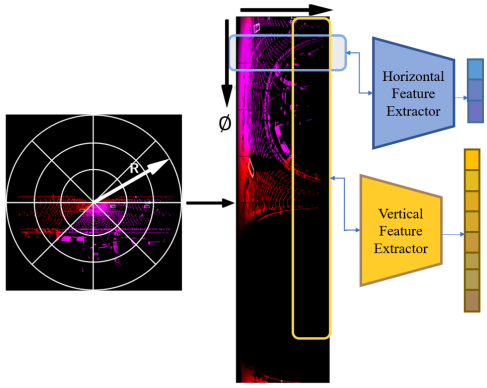

Figure 3: Transformation of the Latent feature space into the polar space and further parsing of the tensor into latent vectors with row-wise (Blue) or colum-wise(Yellow) CNNs. These vectors are fed as input to the transformer.

the efficacy of this method we also utilized stateful NMS [2] for consistency while achieving the most of detection performance at much fewer FLOPS of transformer.

3. Experiments

KITTI dataset provides 7481 training and 7518 testing data for 3D object Detection for cars cyclist and pedestrians. The proposed model is only evaluated on Car as most research is on the evaluation on the class. The detection performance is generally compared with official KITTI evaluation metrics, which are 3D detection average precision and Birds every view average precision. Whereas **Waymo open dataset** contains 798 training and 202 validation sequences for vehicle and pedestrian class. The LiDAR points are 180k per 0.1s. And the official metrics used to evaluate the performance of the model is mAP and mAP weighted by heading accuracy. The IoU threshold is 0.7 for vehicles and 0.5 for pedestrians. The latest update of the evaluation toolkit provides breakdown to 2 levels $level_1$ for boxes with more than 5 LiDAR points and $level_2$ for boxes with at least 1 LiDAR point.

3.1. Implementation Detail

We evaluated the 3D object detection task for the 360° point-cloud data on KITTI and challenging Waymo Open dataset.

Feature Extractor At the start of the pipeline, the extractor is an Multi-Perceptron layer of PointNet[9] module to parse the voxelized/discritized point-cloud into a pseudo image. The projected pseudo-image is of the shape $(C, 512, 512)$ where the C is the number of channels after encoding the features from voxels into a canvas tensor. This pseudo image tensor is transformed to a polar feature dimension of shape $(C, 1024, 258)$ by utilizing the conversion formula in equation 1 & 2. The dimentions of the Polar space tensor is $(512, 1024)$. This tensor is then cropped to construct 64 X 32 dimension crops which is input to 16 CNNs for sector-wise embedding generation with embedding shape dimention E as 512. Whereas for range-wise processing the cropped tensor is parsed by 32 CNNs for embeddings of same size. A position encoding is added to the embedding from the crops of encoding of the same size. The feature extrator is minimal and one stage for even streaming perception.

Transformer The transformer encoder-decoder network

Method names		AP_{BEV}			AP_{3D}			AOS_{AP}
Reference	Input representation	Easy	Mod.	Hard	Easy	Mod.	Hard	-
ComplexYolo [7]	Pseudo-Image	75.32	70.19	62.43	62.11	59.28	55.23	78.45
StarNet [8]	Point-Based	88.53	83.79	77.9	81.94	71.88	66.38	79.20
PointPillars [4]	Pseudo-Image	88.36	86.1	79.83	79.05	74.99	68.35	86.51
PointRCNN [11]	Point-Based	90.34	88.21	86.68	89.71	79.45	75.82	87.98
PV-RCNN [10]	Point-Voxel Fusion	92.24	89.96	78.32	90.52	**83.70**	72.32	89.94
FPA3DOD (Ours)	Pseudo-Image	**94.70**	**90.02**	**87.76**	**91.56**	82.35	**79.85**	**93.27**

Table 1: State-of-art comparisons for 3D detection on KITTI validation set.

Model	$\mathbf{mAP_{3D}}$	Parameters (millions)	FLOPs (Giga)	Average Inference Time
PointRCNN	80.2	2.20	25	620ms
PointPillars	78.9	1.43	32	16ms
FPA3DOD (ours)	93.81	0.59	6	14ms

Table 2: Comparison of parameters and flops and 3D AP of Car class in KITTI validation set

	Method Names	AP	
Metric Levels		Vehicle	Pedestrian
Level 1	PointRCNN [11]	61.2	64.8
	PointPillars [4]	59.2	62.1
	PV-RCNN [10]	73.3	60.2
	FPA3DOD(Ours)	**76.3**	**72.7**
Level 2	PointRCNN [11]	59.8	60.2
	PointPillars [4]	56.4	57.9
	PV-RCNN [10]	65.4	68.3
	FPA3DOD(Ours)	**69.8**	**70.1**

Table 3: Comparison of the state-of-the-art for 3D object detection on Waymo open dataset validation set.

is trained using an AdamW [3] optimizer with One-cycle learning curve [12] with initial learning rate of $2e^{-4}$. The batch size is set to 4 and the network is trained for 100 epochs. For the transformers, the number of encoder layers is 6 and decoder at 4. Variations at the encoder and decoder layers did not show variance in performance results either for cartesian or Polar crops. The hyper-parameter values are determined heuristically.The whole pipleine is trained from scratch for about 220GPU hours in multiple Nvidia 1080 Ti's with Waymo dataset for each model experiment either with Cartesian or Polar cropping.

3.2. Results

We present results of performance and latency in KITTI dataset and performance on both levels in Waymo datset. Table 1 provides a comparison of our performance over the state-of-art methods. The model performs well over others in mean Average precision at high recall for Birds-eye-view (BEV) and also for the mean of Average orientation KITTI metrics. Whereas PV-RCNN[10] outperforms for Moderate difficulty on range for 3D average precision. This difference is hypothesized due to imprecise prediction of smaller classes like pedestrian with large number of false positives. We also analyze the efficacy of the model in Table 3. It also validates to be faster with less parameters while being performant.

4. Conclusion

We proposed a Polar sequence-to-sequence feature extraction framework for efficient streaming 3D object detection from the LiDAR point-clouds. The method uses PointNet extractor with a few convolution layer operators on each ring/sector to produce embeddings as input to the sequence-to-sequence processing transformer model. The model is trained for NMS-free detection with Set Loss. FPA3DOD is a real-time performant and achieves state-of-the-art performance on the validation set of KITTI and Waymo open dataset benchmarks.

References

[1] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1355–1361. IEEE, 2017.

[2] Wei Han, Zhengdong Zhang, Benjamin Caine, Brandon Yang, Christoph Sprunk, Ouais Alsharif, Jiquan Ngiam, Vijay Vasudevan, Jonathon Shlens, and Zhifeng Chen. Streaming object detection for 3-d point clouds. In *European Conference on Computer Vision*, pages 423–441. Springer, 2020.

[3] Frank Hutter Ilya Loshchilov. Decoupled weight decay regularization. In *arXiv preprint arXiv:1711.05101*, 2017.

[4] 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*

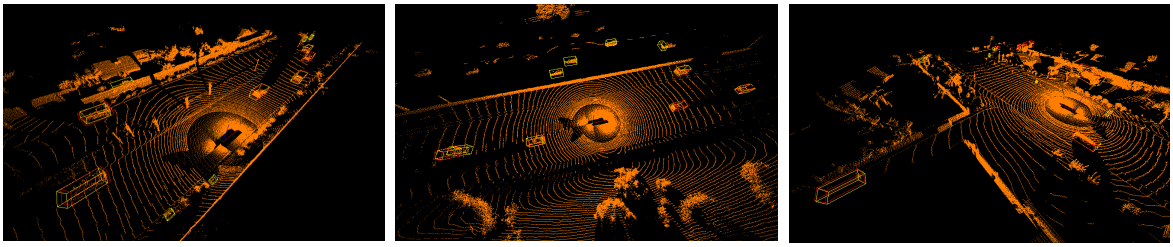

Figure 4: Qualitative description of performance of the model on Waymo data, yellow boxes are the model prediction whereas the Red boxes are the Ground Truth

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12697–12705, 2019.

[5] Bo Li. 3d fully convolutional network for vehicle detection in point cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1513–1518. IEEE, 2017.

[6] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.

[7] Karl Amende Horst-Michael Gross Martin Simon, Stefan Milz. Complex-yolo: Real-time 3d object detection on point clouds. In *arXiv:1803.06199*, 2020.

[8] Jiquan Ngiam, Benjamin Caine, Wei Han, Brandon Yang, Yuning Chai, Pei Sun, Yin Zhou, Xi Yi, Ouais Alsharif, Patrick Nguyen, et al. Starnet: Targeted computation for object detection in point clouds. *arXiv preprint arXiv:1908.11069*, 2019.

[9] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[10] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[11] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

[12] Leslie N. Smith. Cyclical learning rates for training neural networks. In *arXiv:1506.01186*, 2017.

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

[14] 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.