International Journal of Applied Science; Vol. 8, No. 4; 2025
ISSN 2576-7240 E-ISSN 2576-7259
https://doi.org/10.30560/ijas.v8n4p1

AttentPillars: Capture the Highlight Pillars

Shaocong Jiang[1] & Jing Chen[1]

[1] Guangdong University of Technology, China

Correspondence: Jing Chen, Guangdong University of Technology, Guangzhou, China.

Received: October 2, 2025; Accepted: October 19, 2025; Published: October 22, 2025

Abstract

3D object detection play an important role in the perception part of autonomous driving. Recent algorithms suggest two types of 3D object detection, point-based and voxel-based, have the shortcomings of slow speed and low detection accuracy respectively. PointPillars mainly focuses the information on xoy plane, it has precise detection performance and fast speed, so it is more suitable for the perception part of autonomous driving. However, PointPillars pay the same attention to each pillar, the important information is not given more attention, so that the model cannot effectively focus on important information resulting in suboptimal performance. To address this issue, we propose AttentPillars, including a new two-branch downsampling structure with max pooling, and leveraging the SiLU activation function to build the backbone. Moreover, we fuse the split attention to the upsampling module, so that the important information could be highlighted. Apart from this, we find that the common problems in autonomous driving, both the sparsity of point cloud, the occlusion, and the diversity of objects, would damage the performance of detection. In that case, we change the normal data augmentation strategy, The shape-aware data augmentation strategy is added to the data augmentation part to boost the performance of our model. Extensive experiments on the KITTI dataset indicate that our proposed AttentPillars outperforms the state-of-the-art methods with remarkable margins. In the easy, moderate, and hard difficulties, our AttentPillars performs accuracy with 89.54%, 80.25%, and 77.47% respectively.

Keywords: 3D object detection, autonomous driving, point cloud, pillars

1. Introduction

Deep learning has been applied across various domains [1]. Computer vision plays a significant role in autonomous driving [2], there are three main parts in autonomous driving, perception part, decision-making part, and control part. The perception part is to capture information about the surroundings through sensors such as cameras, LiDAR, etc., and then recognize the surrounding situation based on this acquired information. 3D object detection is widely adopted in the perception part of autonomous driving [3, 4, 5].

Different from the traditional 2D object detection, 3D point cloud often serve as the input of the 3D object detection. Firstly, a point cloud is a set of points, 3D point cloud is very special, characterized by sparsity, disorder, interconnectivity, and permutation invariance, which brings tremendous difficulties to the detection [6]. Compared with 2D data, like images, 3D point cloud have one more dimension, which requires a larger amount of memory and computation, so the detection complexity also rises significantly.

Current 3D point cloud learning approaches can be separated into point-based [7, 8, 9] and grid-based approaches [10, 11, 12, 13]. Point-based approaches directly extract features from raw 3D point cloud from LiDAR, which can preserve the local information well, PointNet++ [8] proposed Set Abstraction layer for downsampling, which is widely used in point-based approaches, SA layer enables the model to capture the local information of input point cloud, but due to the disorder of point cloud, point-based methods require a lot of memory and computation. The grid-based approaches are mainly voxel-based [10], voxel-based methods split the whole raw point cloud into regularly shaped voxels and then put them into 3D convolutional neural network, which aim to reduce the amount of calculation and memory occupation, however, due to the sparsity of point cloud, plenty of empty voxels will inevitably be generated in the process of feature encoding, so it is also a huge waste of calculation and memory if each voxel is processed. SECOND [11] replaces 3D convolution with sparse convolution, which compute only on non-empty voxels to further reduces the amount of calculation. Point-based and voxel-based hybrid methods [14, 15, 16, 17, 18] combine the advantages of the point-based and voxel-based methods, such as PVRCNN [14], and PVRCNN++ [15], which proposed Voxel Set Abstraction, combines the non-voxel features around keypoints. PVRCNN has high computational efficiency and flexible receptive field to grasp the context information, it improves the detection accuracy and reduces the computational complexity.

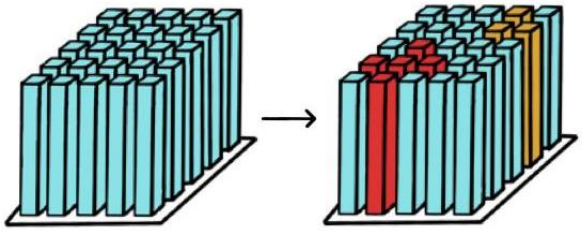

Figure 1. Pillars with more attention. Pillars, which represents important information, is given more attention and is more prominent in the later training

These methods have their advantages and disadvantages. However, in the real scene of autonomous driving, lots of challenges in autonomous driving because of various environment [19], the model is embedded in the car, and most of the detected objects are attached to the ground plane, so PointPillars [13] is more suitable for autonomous driving tasks. PointPillars divides the point cloud into pillars according to the xoy plane, and all the points falling in a pillar from the z-axis are regarded in this pillar. It directly obtains the bird's eye view feature and pays more attention to the xoy plane without paying too much attention to the information of the z-axis, which also saves a lot of calculation and memory. For autonomous driving, PointPillars [13] can reach the balance between detection accuracy and speed, but, in a scene, the objects that need to be detected, such as cars, pedestrians, and bicycles, only occupy a small part of the point cloud. If every part of point cloud scene is given the same attention, then the important information can not be highlighted, which will damage the detection performance. Although PointPillars mainly focus on the information of the xoy plane, the sparsity of point cloud seriously affects the convergence of the model because there is no mechanism for PointPillars to capture the important points that correspond to objects. We aim to highlight the important information, as Figure 1. Therefore, we proposed AttentPillars, which divides the point cloud into pillars and designs a two-branch super downsampling block with max pooling [20] and convolution. Compared with the downsampling block only with convolution, max pooling can reduce the information loss of important information in the downsampling block and capture the important information in the pseudo-image more completely. But the increase of model complexity will inevitably lead to more difficult optimization of the model, and even gradient disappearance, gradient explosion and other problems, to handle these issues, we leverage the SiLU activation function [21, 22] to build the backbone downsampling blocks, SiLU activation function is smoother than ReLU activation function [23], which is easier to optimization. In addition, after downsampling, feature maps with different sizes have different receptive fields, so they contain different information, the larger feature map has a smaller receptive field and is suitable for detecting small objects, such as pedestrians and cyclists, while the smaller feature map has a larger receptive field and is suitable for detecting larger objects, such as cars. Therefore, in order to make full use of the information obtained from feature maps of different sizes, we add split attention [24] to concatenate features after upsampling, it generates an attention weights matrix through three input features, and then multiplies the three input features with the attention weights matrix before concatenation, then concatenate them as the final feature, so the final feature can retain the important information contained in the features with different scales. More than that, we use a hybrid data augmentation strategy that includes shape-aware data augmentation, to address the common problem of the real autonomous driving scene, such as occlusion of objects, diversity of objects, and sparsity of point cloud caused by the limitation of sensor range of the LiDAR, it greatly improves the detection performance of our model. Extensive experiments validate the improvements of our work. On the KITTI dataset, our AttentPillars achieve the accuracy with 89.54%, 80.25%, and 77.47% in easy, moderate, and hard difficulties respectively, it reaches improvement of 0.47% in 3D detection. In the BEV detection, AttentPillars achieves 90.17%, 86.45%, and 81.63% in easy, moderate, and hard difficulties, it performance exceeds 80% even in the hard difficulty. It is worth mentioning that our work achieves a substantial improvement of 5.06%, 2.97%, and 1.94% in average orientation similarity(AOS), this shows that the introduction of attention mechanism makes our model more accurate in directional regression.

Our main contributions are as follows:

1) We propose AttentPillars framework, which can capture the important information of the point cloud leading to better performance and detection speed.

2) We design a two-branch downsampling block with max pooling and leverage the split attention in the concatenation block to concatenation the features of upsampling, which enables features to retain several important information to improve model detection accuracy.

3) Moreover, we built the backbone with the SiLU activation function instead of the ReLU activation function, which is suitable for the optimization of the model.

2. Related Work

Computer vision can be mainly divided into image classification, object detection, and semantic segmentation. Object detection is the most popular and widely used task, compared with image classification, object detection is the task of recognizing the category of an object in a scene, but also accurately locating the position of the object in the whole scene by describing the position of the object with a bounding box, the research of object detection first started from 2D, and then a lot of work gradually appeared in the field of 3D object detection as well.

2D object detection 2D image object detection is the localization and recognition of objects in a picture, there are mainly anchor-based and anchor-free methods. SSD [25] is a one-stage object detection method, that uses anchors in various levels of feature maps for detection, RCNN [26] is the first to use deep learning in object detection, Fast RCNN [27] proposes RoI pooling, and Faster RCNN [28] design RPN to search object. Yolo[29] is an anchor-free method, which directly finishes the detection work with one forward propagation. DETR [30] introduce the attention mechanism to object detection. ViT [31] encoder image into a sequence and then process it with a transformer.

3D object detection 3D object detection was mainly image-based methods before, such as BEV(bird's eye view)-based methods, MV3D [32], is a multi-view based object detection method, which utilizes information from multiple view of the scene, and take both LiDAR point cloud and RGB images as input as to predict oriented 3D bounding boxes. Current 3D object detection methods are mainly about point-based, voxel-based, and hybrid, Pointnet [7] is the first point-based method proposed, which directly sends point cloud data into the network to train by using the symmetric function to deal with the permutation invariance to point cloud. Pointnet++ [8] proposes the set abstraction, which makes up for the shortcoming that pointnet cannot learn the local features of the point cloud, and through the flexible receptive field capture hierarchical features. [33] achieved Real-world detection by simulating the street environment like a photo. PointRCNN [9] is a two-stage method, the first stage, contains a bottom-up network, design bin-based proposal-predict method, and the second stage is a proposal refinement network. 3D-SSD [34] proposes a lightweight 3D detection model, which dropout all the upsampling layer and refinement stage to save computation and memory costs.

Although the point-based methods have the advantage of better obtaining the local information and context information of the points, their computational and memory costs are generally tremendous, so voxel-based methods emerged, Voxelnet [10] split point cloud into regular voxels, and then extract point-wise features from every voxel by VSA, and then feed them into 3D convolutional neural network for training, but due to the sparsity of 3D point cloud, it may still cause computational resources wastage, SECOND [11] use the sparse convolutional neural network instead of the normal convolutional neural network, which greatly reduces the amount of computation and also proposed an angel loss regression to predict proposals more precisely. VoxelRCNN [12] indicate that the precise location of the original points is not necessary for 3D object detection, and it shows that 3D structure is significant for 3D detection head.

Point-based and voxel-based approaches both have advantages and disadvantages, therefore, PVRCNN [14] collects the advantages of voxel-based and point-based, it is computationally efficient and captures context information, summarizes the 3D scene into a small set of keypoints through a new voxel SA block, and introduces RoI-grid pooling, through the keypoints SA layer with multiple receptive fields, the features of proposals are extracted from key points to RoI grid points. Based on PVRCNN, PVRCNN++ [15] offered some improvements, it proposed sectorized proposal-centric to better sample keypoints and VectorPool aggregation modules to aggregate local feature information from large-scale point clouds more efficiently and economically. In addition, Fast Point RCNN [18] uses both voxel representations and raw point cloud and introduces the attention mechanism into the process of point cloud semantic feature extraction. STD [16] groups points by using spherical anchors.

Pillar-based methods use another different local point aggregator, pillar, which divides the point cloud data into pillars, and then send them to the convolution layer for feature learning, PointPillars [13], which achieves the balance between detection accuracy and speed, encodes 3D point cloud information into a 2D pseudo-image, and then extracts features hierarchically by downsampling, it pays more attention to the information of xy axis and less attention to the information of z axis, and converts 3D point cloud data to 2D data, which greatly improves the detection speed and reduces occupation of memory and saves calculation. PillarNet [35] proposed the structure of "encoder-neck-head", and applied SPConv to improve the detection accuracy. FastPillars [36] proposed Max-and-Attention pillar Encoding(MAPE) module, which takes into account the local detailed geometric information of each pillar, and is more conducive to the detection of small and medium-sized objects, such as pedestrians and

bicycles. It also designs CRVNet, a lightweight and powerful backbone, which provides a simple and efficient alternative for the SPConv-based detector. Centerpoint [37] converts 3D objects to points and converts the 3D object detection to a greedy nearest point matching task. PillarNeXt [38] compares local point aggregator, voxel, pillar, and multi-view fusion from the perspective of computational budget allocation, and it claims expanding the receptive field is very important for 3D object detection. PillarNeSt [39] introduces dense ConvNet pretrain on large-scale image dataset as the backbone, and in order to reduce the information loss caused by max pooling, it added the mean pooling to extract features.

Several modules should be applied to different actual scenes according to their advantages and disadvantages, taking into account the characteristics of the input data and the need for computational cost and speed. In autonomous driving, quickly detecting and classifying the objects on the road is needed. The requirements for precise local information are not important compared to other detection tasks. So, voxel-based methods are more often applied in autonomous driving tasks, considering most of the objects in autonomous driving usually adhere to the xoy plane, it is efficient to focus on the position information of the object in the horizontal plane, Pointpillars [13] is proposed to be more suitable for use in autonomous driving.

In autonomous driving, the detection task needs to be completed faster, we need to focus on the important information of the input point cloud data. We proposed AttentPillars, leveraging max pooling, SiLU activation function [21], and split attention strategy [24] to ensure our model can capture the important information. Our method, can obtain more efficient and accurate detection results in the field of autonomous driving, has a high speed of convergence, and shows excellent generalization ability, it can be better applied to autonomous driving. Moreover, we tested the influence of various data augmentation strategies, and finally, we leveraged the shape-aware data augmentation strategy [40], which improved the detection performance by a large margin.

3. AttentPillars Architecture

We proposed AttentPillars, which consist of a feature encoding network, backbone, and detection head. The overall architecture is shown as Figure 2. The feature encoding network voxelize the input point cloud into pillars, and then converts them into 2D pseudo-images and sends them to the 2D backbone for feature representation. The backbone downsamples the feature map into three different sizes and channels sub-maps, to obtain multi-scale and multi-dimension features. Then upsampling them and sent into the detection head.

3.1 Feature Encoding

Point cloud is a set of unordered and sparse points [7], therefore, directly processing point cloud will consume a lot of calculation and computer memory, resulting in a slow train speed. Because most of the objects in autonomous driving adhere to the xoy plane, such as cars, pedestrians, and cyclists, there is no need to pay too much attention to the information on the vertical, so our feature encoding part plan to convert 3D point cloud into 2D pseudo-image. Firstly, the point cloud is regularly divided into grids according to the xoy plane, the cuboid represented by each grid is a pillar, and the points falling in a grid are regarded in this pillar. In the KITTI dataset [41, 42], point is represented by a four-dimensional vector, (x, y, z, r), which respectively correspond to the coordinates on the XYZ axis and the reflection intensity of the points.

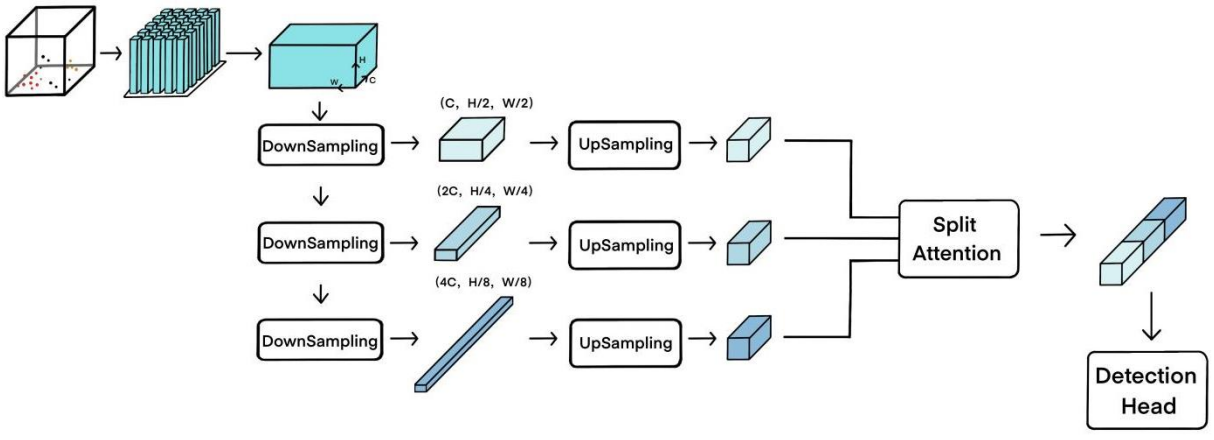

Figure 2. Network overview architecture. Our model contains three components: feature encoding network, 2D backbone, and detection head. The raw point cloud is voxelized into pillars, and convert to a pseudo-image, then the 2D backbone process the feature, detection head predict the 3D bounding boxes forobjects based the features from 2D backbone

In AttentPillars, in order to obtain more abundant features, we also add the centroid offset O_c and the grid offset feature O_g, and adopt L1 norm operation for both offsets N_g, N_c, and add horizontal and vertical offset angels θ_h, θ_v, as well as depth features F_{dep} and distance features from the point to the origin F_{dis}, each of them using Eq. 1-Eq. 6,

$$O_c = P - P_c \tag{1}$$

$$O_g = P - P_g \tag{2}$$

$$\theta_h = \arctan(\frac{y}{x}) \tag{3}$$

$$\theta_v = \arcsin(\frac{z}{x^2+y^2+z^2}) \tag{4}$$

$$F_{dep} = \sqrt{x^2 + y^2} \tag{5}$$

$$F_{dis} = \sqrt{x^2 + y^2 + z^2} \tag{6}$$

then the final feature is concatenated by this features as Eq. 7. These latent features explore the information of each pillar, and make up for loss caused by not focus on the z-axis information.

$$Y = \text{concat}(O_g, O_c, N_g, N_c, \theta_h, \theta_v, F_{dep}, F_{dis}) \tag{7}$$

After the grid voxelization and point features representation, due to the sparsity of point cloud, some pillars have no or few points, so that a lot of unnecessary memory and computation will be consumed. In order to address the problem of the number of points in different pillars, we set a hyperparameter N to limit the number of points in each pillar[13]. If the number of points in a pillar exceeds N, then random sample N points, if is less than N, padding with zero. In this way, the point cloud can be seen as a tensor with (D, P, N), where D indicates the characteristics of point, P represents the number of pillars, and N denotes the number of points. Then a simple PointNet [8] is used for this tensor to get a tensor (C, P, N), and then the tensor is transformed into a tensor of (C, P) by the max pooling at the N dimension. Finally, the point features are scattered back into their pillars to get the pseudo-image (C, H, W), where H, W denote height and width respectively, and C denotes the channels of this pseudo-image.

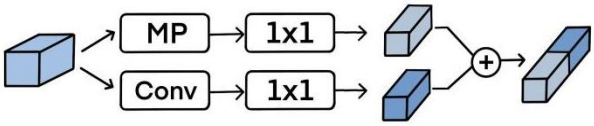

Figure 3. Downsampling block architecture. There are two branches in downsampling block, one with the max pooling operation and another one with the convolution operation, both of them can upsampling feature maps, the finaloutput of downsampling block is a concatenation of them.

3.2 2D Backbone

2D backbone consists of two subnetworks, a downsampling network and an upsampling network. In order to get better downsampling feature maps, we design a different downsampling block, which is two-branch, containing both max pooling [20] and convolutional neural network. And also introduces the SiLU activate function [22]. At the same time, in order to improve the accuracy, we added the split attention strategy [24] into the concatenation part to generate the final features, which makes the features more representative.

One is the bottom-up network, which contains three downsampling blocks illustrate as Figure 3. In AttentPillars, the pseudo-image is process by a two-branch module, and the size of the input pseudo-image is (C, H, W), firstly, the pseudo-image passes through a max pooling layer, in which the step is 2 and the kernel size is 2x2, so that the size of the obtain feature map is reduced by half $(C, H/2, W/2)$, and then put it to a 1x1 convolution to halve its channels, the size of pseudo-image become $(C/2, H/2, W/2)$, recorded as $M1$. Secondly, the input pseudo-image is sent to a 1x1 convolution network, the number of channels is halved, and then it is sent to a convolution kernel step of 2 and kernel size of 3x3, and the feature map size is $(C/2, H/2, W/2)$, which record as $M2$. Finally, we concatenate $M1$ and $M2$ among the channel dimension, and the output of the downsampling block is obtained,

with the size of $(C, H/2, W/2)$. There are three downsampling blocks in our backbone, we obtain three feature maps with different sizes, 2x, 4x, and 8x, their channel numbers are $1C, 2C,$ and $4C$ respectively.

In the downsampling block, each convolutional layer and max pooling layer is followed by a batch normalization [43] and a SiLU [21] activation function, instead of a ReLU [23] activation function. We use the SiLU activation function to build the downsampling block, which is also called Swich activation function, compared with the ReLU activation function, it is smoother and differentiable, so it is conducive to optimization. It can better capture the gradient information in the backpropagation and better preserve the input information. The function of SiLU as Eq. 8.

$$\mathrm{SiLU}(x) = x * \mathrm{sigmoid}(x) = \frac{x}{1+e^{-x}} \tag{8}$$

Through this super downsampling block with max pooling and 2D convolutional neural network at the same time. We get feature maps with multi-scale and multi-channel, which is beneficial for our model to acquire richer information. The downsampled feature maps can expand the receptive field during training, and neurons can better capture the context information. Moreover, due to the max pooling operation, our feature maps pay more attention to the useful and important points.

We acquire three different scale and channel feature maps, $(C, H/2, W/2), (2C, H/4, W/4), (4C, H/8, W/8)$, next, we use deconvolution to upsampling them to the same size, and batch normalization and ReLU activate function are added, the output of upsampling block as Eq. 9.

$$Y = \mathrm{ReLU}(\mathrm{Bn}(Y_{up}(Y))) \tag{9}$$

After the upsampling, we used the weight redistribution strategy to avoid gradient explosion and disappearance, to accelerate the convergence speed and to improve the generalization ability of the model. Moreover, we introduce split attention to boost the detection accuracy, as shown in Figure 4. Which sums the three inputs and adopts a max pooling operation, then goes through a fully connected layer, a batch normalization, and ReLU activation function, and then goes through the fully connect layer respectively, finally using softmax to get the weights, we obtain the attention weight matrix, and then multiplies the inputs with the matrix, then obtains the final output features. The upsampled feature maps as the input of the split attention block, compared with the directly concatenate features in PointPillars, AttentPillars have richer and more comprehensive features, which can represent the relevant information of objects in the original point cloud and reduce the loss of raw point cloud, boost the robustness and generalization ability of the model.

Overall, the downsampling blocks include both max pooling and convolution operations, and replace the ReLU activation function with the SiLU activation function which is more suitable for training. In the features concatenation part, we leverage the split attention strategy. Our model obtains the multi-scale and multi-dimension feature maps, it can better capture objects of different sizes, feature maps with large scale, the receptive field is relatively small, so it is suitable for capturing small targets. Our model also pay more attention to the relatively more important information, for improvement of the detection accuracy.

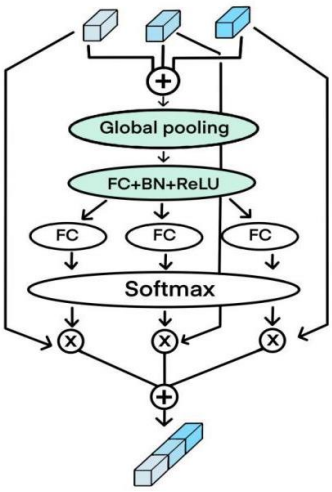

Figure 4. Split attention architecture. Concatenating three input and using max pooling operation to extract the important information, and a weight is obtained. This weight is multiplied by the original input and concatenate them to get the final output

3.3 Detection Head

Our work uses SSD [25] as the detection head of the model, different from the 3D detection head, our model converts 3D point cloud into 2D pseudo-images for processing, so we use 2D Intersection over Union(IoU) to match the ground truth and proposal. Both ground truth boxes and proposals are represented as a 7-dimension vector $(x, y, z, w, l, h, \theta)$, where x, y, z denote the centre of the bounding box, w, h, l denote the length, width, and height of the bounding box, θ denotes the orientation. We use the same loss function as SECOND [11] as Eq. 10,

$$\begin{aligned} &\Delta x = \frac{x^{gt} - x^a}{d^a}, \Delta y = \frac{y^{gt} - y^a}{d^a}, \Delta z = \frac{z^{gt} - z^a}{d^a}, \\ &\Delta w = \log \frac{w^{gt}}{w^a}, \Delta l = \log \frac{l^{gt}}{l^a}, \Delta h = \log \frac{h^{gt}}{h^a} \\ &\Delta \theta = \sin(\theta^{gt} - \theta^a) \end{aligned} \tag{10}$$

where the p^a denotes probability of the model, the x^{gt} and x^a denote the ground truth and anchor boxes respectively, and $d^a = \sqrt{(w^a)^2 + (l^a)^2}$, so the localization loss as Eq. 11,

$$\mathcal{L}_{loc} = \sum_{b \in (x,y,z,w,l,h,\theta)} SmoothL1(\Delta b) \tag{11}$$

we use the focal loss [44] as the cls loss, we set the $\alpha = 0.25$, $\gamma = 2$, as in Eq. 12,

$$\mathcal{L}_{cls} = -\alpha(1 - p^a)^\gamma \log(p^a) \tag{12}$$

In addition, we add the corner loss [32] into the loss function as Eq. 13, aiming to focus the regression of the angle of the proposals, to further improve the accuracy of the anchors,

$$\mathcal{L}_{cor} = \sum_{i \in (1,2,\ldots,8)} \sum_{T \in (x,y,z)} \Delta T_i \tag{13}$$

Overall, the final loss function as Eq. 14, where N_{pos} denotes the number of positive anchors, the $\mathcal{L}_{reg}, \mathcal{L}_{cls}, \mathcal{L}_{dir}, \mathcal{L}_{cor}$ denote the regression loss, classification loss, direction classification loss, and corner loss respectively, we set $\beta_{cls} = 2$, $\beta_{cls} = 1$, and $\beta_{cls} = 0.2$.

$$\mathcal{L}_{cor} = \frac{1}{N_{pos}} (\beta_{reg}\mathcal{L}_{reg} + \beta_{cls}\mathcal{L}_{cls} + \beta_{dir}\mathcal{L}_{dir}) + \mathcal{L}_{cor} \tag{14}$$

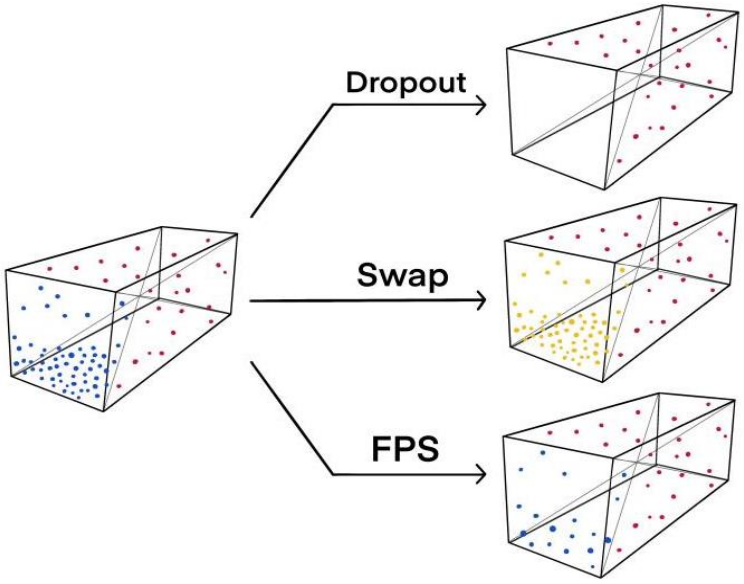

Figure 5. Shape-Aware data augmentaion. All the operations are demonstrated with the front pyramid. The first is randomly dropout one pyramid, the second is swap a pyramid with another ground truth box, the third is use the farthest point sampling strategy in one pyramid to decrease the number of point

4. Empirical Analysis

Table 1. Results on the kitti test bev detection benchmark.

METHOD	CAR IOU=0.7			PESDESTRIAN IOU=0.5			CYCLIST IOU=0.5		
	EASY	MOD	HARD	EASY	MOD	HARD	EASY	MOD	HARD
MV3D	86.02	76.90	68.49	-	-	-	-	-	-
VOXELNET	89.35	79.26	77.39	46.13	40.73	38.11	66.70	54.76	50.55
SECOND	88.07	79.37	77.95	55.10	46.27	44.76	73.67	56.04	48.78
F-POINTNET	88.70	84.00	75.33	58.09	50.22	47.20	75.35	61.96	54.68
3D IOU LOSS	91.36	86.22	81.20	-	-	-	-	-	-
POINTRCNN	89.47	85.68	79.10	-	-	-	-	-	-
PIXOR++	89.38	83.70	77.97	-	-	-	-	-	-
POINTPILLARS	88.35	86.10	79.83	**58.66**	**50.23**	**47.19**	**79.14**	62.25	**56.00**
OURS	**90.17**	**86.45**	**81.63**	45.57	37.43	35.00	77.18	**62.8**	54.68

Table 2. Results on the kitti test 3d detection benchmark

METHOD	CAR IOU=0.7			PESDESTRIAN IOU=0.5			CYCLIST IOU=0.5		
	EASY	MOD	HARD	EASY	MOD	HARD	EASY	MOD	HARD
MV3D	71.09	62.35	55.12	-	-	-	-	-	-
VOXELNET	77.47	65.11	57.73	39.48	33.68	31.50	61.22	48.36	44.37
F-POINTNET	81.20	70.39	62.19	51.21	44.89	40.23	71.96	56.77	50.39
PV-RCNN	90.25	81.43	76.82	52.17	43.29	40.29	78.60	63.71	57.65
POINTPILLARS	82.32	73.00	66.74	40.65	32.74	30.19	75.78	54.53	47.85
OURS	81.93	**73.47**	**68.34**	40.64	32.65	30.15	69.72	54.53	48.02

Table 3. Results on the kitti test average orientation similarity (aos) detection benchmark.

METHOD	CAR IOU=0.7			PESDESTRIAN IOU=0.5			CYCLIST IOU=0.5		
	EASY	MOD	HARD	EASY	MOD	HARD	EASY	MOD	HARD
SUBCNN	90.61	88.43	78.63	78.33	66.28	61.37	71.39	63.41	56.34
AVOD-FPN	89.95	87.13	79.74	53.36	44.92	43.77	67.61	57.53	54.16
SECOND	87.84	81.31	71.95	51.56	43.51	38.78	80.97	57.20	55.14
POINTPILLARS	90.19	88.76	86.38	58.05	49.66	47.88	82.43	68.16	61.96
OURS	**95.25**	**91.73**	**88.32**	44.17	35.41	33.11	80.21	65.24	57.99

In this section, we introduce the experimental of our work, the KITTI dataset [41, 42], and the details of implementation, and compare our method with the current mainstream model. We also introduce the related ablation experiments to find the influence of each component on our experimental results. Furthermore, we introduce the shape-aware data augmentation strategy [40] into data preprocessing, and combine other data augmentation strategies to conduct ablation experiments to study the influence on our experimental results.

4.1 KITTI Dataset

KITTI dataset [42] is the most commonly used public data set in the model evaluation of autonomous driving, which consists of data collected by various sensors on urban roads and highways. It includes camera and LiDAR and millimeter wave radar data. The KITTI benchmark requires the detection of cars, pedestrians, and cyclists. According to the degree of occlusion, occlusion extent and box height of the object, each category is separated into three levels: easy, moderate, and hard, which is equivalent to the difficulty of the objects. Easy means that the object can be observed easily, moderate means that part of it is not easy to observe, and hard means that it is hard to detect. The setup of the dataset is the same as PointPillars [13]. The training set contains 7481 samples and the test set contains 7518 samples. We divide the training data set into 3712 training samples and 3769 validation samples with the same segmentation as the PointPillars to ensure the fairness of the comparative experiments and reduce the comparison error caused by the segmentation of the dataset.

4.2 Implementation Details

In our experiments, we set x, y resolution: 0.16m, the maximum of pillars(P) is 12000, and the maximum number of points in each pillar(N) is 100, if it exceeds, 100 points will be randomly sampled, and if it is insufficient, it will be padding with 0. The size of each pillar is [0.16, 0.16, 4], where [0.16, 0.16] is the size after voxelization in the xoy plane.

In our training, we set 120 training epochs to training, the x, y, z range is [(0, 70.4), (-40, -40), (-3, 1)]. We only pay attention to the front-view region of the point cloud scene, the rest of points and ground truth boxes are not used for training. We use the same anchors and matching strategy as VoxelNet [10], each anchor is represented by width, length, and height and z centre, and IoU is set to judge whether the sample is a positive sample or a negative sample. For the three categories in KITTI dataset, due to their differences in shape and volume, they need different anchor sizes, so we set three sizes for each category and set two orientations $0°$ and $90°$ for each anchor. For the car category, we set each anchor has width, length, and height of (3.9, 1.6, 1.56) meters with a z centre of -1.78 meters, and the matching threshold of the positive and negative is 0.6 and 0.45 respectively. For pedestrian and cyclist categories, the anchor has width, length, and height of (0.8, 0.6, 1.73) meters and (1.75, 0.6, 1.73) meters with a z centre -0.6 meters, the matching threshold of the positive and negative is 0.5 and 0.35 respectively. In the process of sample distribution, we use the axis-aligned target assigner [45], samples with the highest IoU or IoU exceeding the set positive sample IoU threshold are recorded as positive samples, and those with IoU lower than the set negative sample threshold are recorded as negative samples, and those with IoU at the between of positive threshold and negative threshold are discarded.

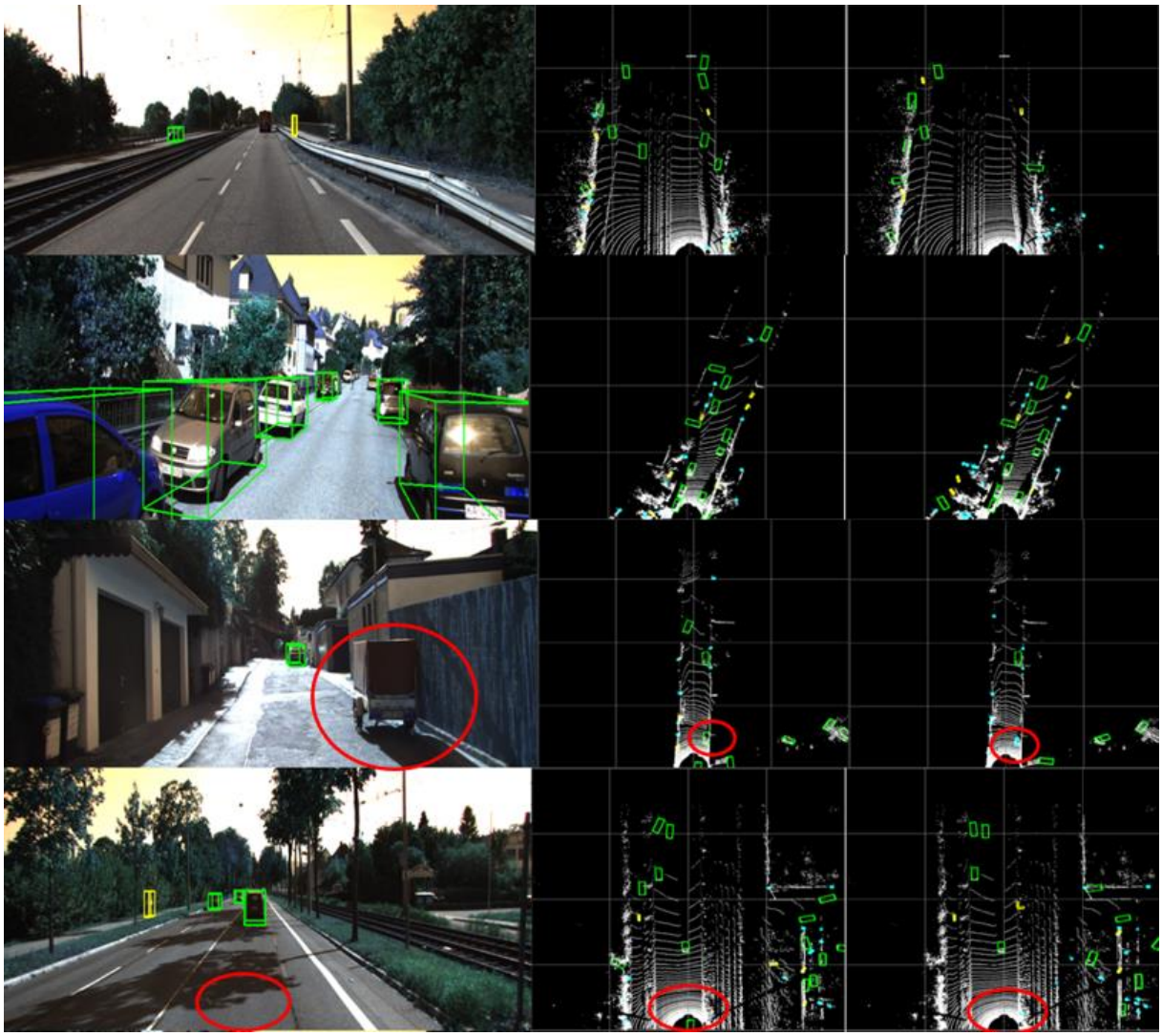

Figure 6. Visualization results. The green bounding boxes represent car, the first column shows the position of ground truth in the front view, the second columnshow the detection results of bounding box output of

PointPillars in the point cloud bird's eye view, the third column show the detection results of boundingbox output of AttentPillars in the point cloud bird's eye view

4.2 Data Augmentation

Data augmentation is very important for model training. We not only use common 3D data augmentation strategies to improve the generalization ability of the model and reduce overfitting, for the common issues in autonomous driving, such as occlusion of objects, diversity of objects (cars of different brands and cyclists in different directions, etc.), and similar objects(cars of the same brand, pedestrians in the same colour) and the sparsity of point cloud because of the limitation of sensing range of LiDAR, we also introduce shape-aware data augmentation [40] to address these issues.

Shape-Aware Data Augmentation We introduce shape-aware data augmentation to enrich the dataset and boost the generalization ability of our model. As shown in Figure 5. Shape-aware data augmentation is a plug-and-play block, which has a significant effect on the part occlusion, diversity of object shapes, and the change of distance. Firstly, for each object in the point cloud, we find the centroid of their ground truth box, secondly, connect the centroid with each vertex of the ground truth box, each ground truth box is divided into six pyramids, finally each object is divided into six subsets, and then performs three following operations to randomize the dataset. The first is to randomly dropout one of the subsets, which can simulate the partial occlusion of objects. The second is to exchange a subset with the same pyramid subset of another object in the current scene, thus increasing the diversity of samples by using the surface similarity between objects. The third is to randomly use the farthest point sampling [49] to sparsely sample the points in the subset, which simulates the sparsity of point cloud caused by the limited sensing distance of LiDAR. We also use common data augmentation such as local translation, local rotation, and local scaling strategies to enrich the dataset and enhance the diversity of the dataset. These data augmentation strategies are very helpful to improve the performance of our method.

4.4 Result

Our experiment runs on four NVIDIA RTX 3090 GPUs, and the CPU is AMD 32-core Processor. We uploaded our results to the KITTI official website to evaluate our method, it offers the results on bird's eye viewing(BEV), 3D detection, and average orientation similarity(AOS). KITTI [41] ranks the models among the moderate level of the car category.

BEV Results Our BEV result is shown in Table 1. In the car category, our method achieves the best performance, compared with the previous PointPillars [13], our AttentPillars have a significant improvement, with an increase of 1.82%, 0.35%, and 1.85% in the three difficulties of easy, moderate, and hard respectively. And it is almost 7% promotion to the previous SECOND in the moderate difficulty. Moreover, on the hard difficulty, the accuracy of our method exceeds 80%, which is not achieved by most of other methods.

3D Detection Results We partitioned the trainval dataset to serve as a validation set for testing purposes. The 3D detection results on KITTI val set are shown in Table 2. Our method has a significant improvement in the car category compared with PointPillars. In the easy and moderate difficulties, the accuracy of AttentPillars is nearly 7% higher than PointPillars, in the hard difficulty, our method is nearly 10% higher than PointPillars. We analyze the reason that our model has a strong ability to capture important information, in the easy and moderate difficulties, the objects are obvious, thus on the feature map, its corresponding information is large and distinct, therefore, although the PointPillars do not have a strong ability to capture the important information, it still can extract these object features and detect them, so the improvement is not as obvious as the hard difficulty. However, in the hard difficulty, due to the objects are not obvious, the relevant area in the feature map will be smaller, and after a series of downsampling, the perception ability of the model to these features will be weakened, but in AttentPillars, the max pooling downsampling can capture important information at the beginning, even if the relevant area of objects on the feature maps is small, and this important information will be more prominent in the concatenation module by the split attention strategy. So AttentPillars has obtained better detection accuracy.

AOS Results Although our method predicts a 3D bounding box with direction, there is no performance index to predict the direction in BEV and 3D detection. KITTI dataset used average orientation similarity (AOS) to represent the performance index of direction prediction, as shown in Table 3, AttentPillars performs surprisingly outstanding, our method not only achieves the best prediction, but also improves the difficulty level of easy, moderate, and hard by 5.06%, 2.97%, and 1.94% compared with PointPillars. It is worth mentioning that it is 10% and 17% higher than SECOND [11] in moderate and hard difficulties, which is great progress. In the cyclist

category, our method achieved good detection accuracy, so it can be seen that the SA data augmentation makes our model a strong perception of the direction of the objects.

Table 4. Backbone ablation experiment results on kitti val dataset. the ablation study has three variables, approach of downsampling, activation function, and concatenation strategy of upsampling, the first and second columns are the approach of downsampling, the third and fourth columns are the activation function in downsampling block, and the fifth to seventh columns are the concatenation strategy after upsampling("add" means multi-scale feature addition, "max" means aggregation maximum, and "cat" means multi-scale feature concatenation). "✓" means adopted this strategy, the last three columns are the accuracy on easy, moderate, and hard difficulties

Conv	Two-branch	ReLU	SiLU	add	max	cat	Easy	Mod	Hard
✓				✓			87.32	77.75	74.78
	✓	✓		✓			87.63	78.29	75.31
	✓		✓	✓			87.95	78.55	75.48
✓					✓		87.83	78.28	75.29
	✓	✓			✓		88.09	78.83	75.65
	✓		✓		✓		86.83	78.13	75.11
✓						✓	88.27	78.80	75.78
	✓	✓				✓	87.70	78.62	75.61
	✓		✓			✓	88.23	78.94	75.87

4.5 Ablation Experiment

In the 2D backbone, we studied the strategy of different downsampling blocks replacing different activation functions, and adopted different strategies in the feature fusion part after upsampling, and used a variety of data augmentation strategies. We take some ablation studies in the category of car with three difficulties: easy, moderate, and hard, to analyze the influence of each part on the overall performance of the model.

For the downsampling module in the backbone, we compare the convolution downsampling in PointPillars with the two-branch downsampling with other modules, to ensure the fairness and accuracy of the ablation experiment. We also compare the experimental results replaced by SiLU activation function [23] with those with the ReLU activation function [22], and we will also study the influence of different upsampling fusion strategies on the experimental results, shown in the Table 4.

Table 5. Inference result

Method	Hz	Speed(ms)
PointPillars	63	12.29
AttentPillars	62	12.87

The experimental result illustrates that two-branch downsampling is better than the original convolution downsampling in three levels of difficulty. In the moderate difficulty, the accuracy of two-branch downsampling can be improved by about 0.5%, the improvement is not obvious only when the upsampling module uses concatenation strategies. We analyze that because the max pooling operation in two-branch downsampling can effectively extract the highlighted information in the feature maps, and then the important information can be better highlighted by adding or taking the maximum fusion method, so the performance is improved, while the concatenation is to concatenate the features to get multi-dimensional features, so the highlighted information may be weakened after concatenation, so the improvement is not obvious. We also compare the activation function, the results show that the detection accuracy is slightly improved. We surmise that the SiLU function is smoother than the ReLU function, which is more conducive to the convergence and optimization of the model, and after the SiLU activation function is used, the original information of the input is better preserved, the detect performance is improved thanks to the SiLU activation function and the max pooling layer. For the feature fusion module, we not only experimented with the concatenation strategies, but also experimented with adding and taking the maximum, the result is that the concatenation is the best, the taking maximum is the second, and the adding is the worst, we speculate that it is because the approach of taking maximum and adding combines three feature maps into a feature map with the same size, while the concatenation acquire a multi-dimensional feature map, so the accuracy is better.

Table 6. data augmentation ablation experiment results on kitti val dataset. the first column is the strategy of data augmentation, the last three columns are the accuracy on easy, moderate, and hard difficulties

Data augmentation strategy	Easy	Mod	Hard
Local translation	83.8138	74.9117	72.2994
Local rotation	88.0128	78.6722	75.6691
Local scaling	88.4555	79.1145	76.1160
Local rotation + Shape-Aware	88.2669	79.0722	76.1264
Local rotation + Local scaling + Shape-Aware(ReLU)	88.2079	78.8001	75.7764
Local rotation + Local scaling + Shape-Aware(SiLU)	89.5394	80.2455	77.4706

In addition to the shape-aware data augmentation strategy, we also adopt three classical point cloud data augmentation strategies: local translation, local rotation, and local scaling to enrich our dataset, to improve the generalization performance of the model. Ablation experiments in the category of car, as shown in Table 6, the strategies of local rotation and local scaling have boosted the performance of the model. We compare the second row with the fourth row of Table 6, it can be seen that when the shape-aware data augmentation strategy is added, the detection accuracy is improved by 0.2%, 0.4%, 0.5 in easy, moderate, and hard difficulties, we analyze in more difficult level, the effect of shape-aware data augmentation strategy is more significant. The shape-aware data augmentation strategy truly can improve the generalization ability of the model. We fuse three data augmentation strategies: local rotation, local scaling, and SA strategies in the final, and the accuracy of the model has been tremendously improved. At the same time, we also compare the differences between the SiLU activation function and the ReLU activation function to build the backbone, we find that the backbone built by SiLU function has achieved the best performance in three difficulties, it can be seen that replace the activation function, which better captures the input original information and has an obvious effect on enhancing the generalization performance of the model.

We set the batch size to 1 to measure the speed of our model. As shown in Table 5, the speed of our model is 12.87ms, which has a subtle difference from PointPillars, although the split attention strategy leads to the increase of parameters, due to the smoothness of SiLU function and the powerful ability of extracting important information of the model, our speed is not much slower than that of PointPillars. Hz represents the number of samples that can be detected in one second, and speed represents the time required for one sample.

4.6 Visualization

We visualized the detection results of PointPillars and AttentPillars on KITTI test set, at the same time, we visualized the ground truth results, to compare the actual detection accuracy of AttentPillars and PointPillars, all the visualization results are shown in Figure 6. The visualization results of PointPillars are shown in the second column of Figure 6, and the visualization results of AttentPillars are shown in the third column of Figure 6. The green bounding boxes represent cars. As we can see, PointPillars is difficult to distinguish some small trailers, which leads to some false positive samples, while AttentPillars can accurately generate bounding boxes of these objects and predict their directions.

In addition, as shown the last row of the Figure 6, there is no object in the position of red circle in ground truth, but the detection result of PointPillars generates a green bounding box here, which is a serious mistake in autonomous driving, however, our AttentPillars does not have this bad problem, and it generates accurate bounding boxes. We suppose that PointPillars cannot accurately capture the important information in the point cloud at first, which leads to mistaking other information for car in the detection, fortunately our AttentPillars highlighted the important information, and generates the accurate bounding boxes. Overall, our AttentPillars outperforms PointPillars both in bounding box generation and direction prediction.

5. Conclusion

In this paper, we propose an end-to-end 3D point cloud object detection algorithm, AttentPillars, which can pay more attention to the valuable information in point cloud. At the same time, we introduce shape-aware data augmentation to boost the generalization ability of our model. Our work is verified on KITTI dataset, and the detection is better than PointPillars, especially in the regression of AOS, our method has achieved nearly 3% improvement. However, in pedestrian and cyclist categories, our method is equivalent to PointPillars, the detection head uses SSD, in future work, to enhance detection accuracy in the pedestrian and cyclist classes, we intend to design a novel detection head, that has better perception of pedestrians and cyclists.

Acknowledgement

We would like to express our sincere gratitude to the editors and reviewers for their valuable comments and suggestions. We also extend our thanks to the Guangdong University of Technology for providing the necessary resources and support for this research. Special thanks to our colleagues and collaborators for their insightful discussions and contributions throughout this project.

References

[1] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

[2] Janai, J., Güney, F., Behl, A., & Geiger, A. (2020). Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, *12*(1-3), 1–308. https://doi.org/10.1561/0600000079

[3] Qian, R., Lai, X., & Li, X. (2022). 3D object detection for autonomous driving: A survey. *Pattern Recognition*, *130*, 108796. https://doi.org/10.1016/j.patcog.2022.108796

[4] Chen, L., Li, Y., Huang, C., Li, B., Xing, Y., Tian, D., Li, L., Hu, Z., Na, X., Li, Z., Teng, S., Lv, C., Wang, J., Cao, D., Zheng, N., & Wang, F.-Y. (2023). Milestones in autonomous driving and intelligent vehicles: Survey of surveys. *IEEE Transactions on Intelligent Vehicles*, *8*(2), 1046–1056. https://doi.org/10.1109/TIV.2022.3223131

[5] Xiao, Y.-P., Lai, Y.-K., Zhang, F.-L., Li, C., & Gao, L. (2020). A survey on deep geometry learning: From a representation perspective. *Computational Visual Media*, *6*, 113–133. https://doi.org/10.1007/s41095-020-0174-8

[6] Tang, X., Huang, F., Li, C., & Ban, D. (2023). A survey on end-to-end point cloud learning. *IET Image Processing*, *17*(5), 1307–1321. https://doi.org/10.1049/ipr2.12729

[7] Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). 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).

[8] Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, *30*.

[9] Shi, S., Wang, X., & Li, H. (2019). PointRCNN: 3D object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 770–779). https://doi.org/10.1109/CVPR.2019.00086

[10] Zhou, Y., & Tuzel, O. (2018). VoxelNet: End-to-end learning for point cloud based 3D object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4490–4499). https://doi.org/10.1109/CVPR.2018.00472

[11] Yan, Y., Mao, Y., & Li, B. (2018). SECOND: Sparsely embedded convolutional detection. *Sensors*, *18*(10), 3337. https://doi.org/10.3390/s18103337

[12] Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., & Li, H. (2021). Voxel R-CNN: Towards high performance voxel-based 3D object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence* (*Vol. 35*, *No. 2*, pp. 1201–1209). https://doi.org/10.1609/aaai.v35i2.16207

[13] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., & Beijbom, O. (2019). PointPillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 12697–12705). https://doi.org/10.1109/CVPR.2019.01298

[14] Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., & Li, H. (2020). 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). https://doi.org/10.1109/CVPR42600.2020.01054

[15] Shi, S., Jiang, L., Deng, J., Wang, Z., Guo, C., Shi, J., Wang, X., & Li, H. (2023). PV-RCNN++: Point-voxel feature set abstraction with local vector representation for 3D object detection. *International Journal of Computer Vision*, *131*(2), 531–551. https://doi.org/10.1007/s11263-022-01710-9

[16] Yang, Z., Sun, Y., Liu, S., Shen, X., & Jia, J. (2019). STD: Sparse-to-dense 3D object detector for point cloud. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 1951–1960). https://doi.org/10.1109/ICCV.2019.00204

[17] He, C., Zeng, H., Huang, J., Hua, X.-S., & Zhang, L. (2020). Structure aware single-stage 3D object detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11873–11882). https://doi.org/10.1109/CVPR42600.2020.01189

[18] Chen, Y., Liu, S., Shen, X., & Jia, J. (2019). Fast Point R-CNN. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9775–9784). https://doi.org/10.1109/ICCV.2019.00987

[19] Wang, K., Zhou, T., Li, X., & Ren, F. (2022). Performance and challenges of 3D object detection methods in complex scenes for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, *8*(2), 1699–1716. https://doi.org/10.1109/TIV.2022.3213796

[20] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

[21] Elfwing, S., Uchibe, E., & Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, *107*, 3–11. https://doi.org/10.1016/j.neunet.2017.12.012

[22] Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7464–7475). https://doi.org/10.1109/CVPR52729.2023.00721

[23] Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (JMLR Workshop and Conference Proceedings, pp. 315–323).

[24] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1492–1500). https://doi.org/10.1109/CVPR.2017.634

[25] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14* (pp. 21–37). Springer. https://doi.org/10.1007/978-3-319-46448-0_2

[26] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580–587). https://doi.org/10.1109/CVPR.2014.81

[27] Girshick, R. (2015). Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440–1448). https://doi.org/10.1109/ICCV.2015.169

[28] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, *28*.

[29] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779–788). https://doi.org/10.1109/CVPR.2016.91

[30] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision* (pp. 213–229). Springer. https://doi.org/10.1007/978-3-030-58452-8_13

[31] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., & Houlsby, N. (2020). *An image is worth 16x16 words: Transformers for image recognition at scale*. arXiv preprint arXiv:2010.11929.

[32] Chen, X., Ma, H., Wan, J., Li, B., & Xia, T. (2017). Multi-view 3D object detection network for autonomous driving. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/CVPR.2017.691

[33] Li, Z., & Zhu, J. (2023). Point-based neural scene rendering for street views. *IEEE Transactions on Intelligent Vehicles*. https://doi.org/10.1109/TIV.2023.3304347

[34] Yang, Z., Sun, Y., Liu, S., & Jia, J. (2020). 3DSSD: Point-based 3D single stage object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11040–11048). https://doi.org/10.1109/CVPR42600.2020.01105

[35] Shi, G., Li, R., & Ma, C. (2022). PillarNet: Real-time and high-performance pillar-based 3D object detection. In *European Conference on Computer Vision* (pp. 35–52). Springer. https://doi.org/10.1007/978-3-031-20080-9_3

[36] Zhou, S., Tian, Z., Chu, X., Zhang, X., Zhang, B., Lu, X., Feng, C., Jie, Z., Chiang, P. Y., & Ma, L. (2023). *FastPillars: A deployment-friendly pillar-based 3D detector*. arXiv preprint arXiv:2302.02367.

[37] Yin, T., Zhou, X., & Krahenbuhl, P. (2021). Center-based 3D object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11784–11793). https://doi.org/10.1109/CVPR46437.2021.01161

[38] Li, J., Luo, C., & Yang, X. (2023). PillarNext: Rethinking network designs for 3D object detection in LiDAR point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 17567–17576). https://doi.org/10.1109/CVPR52729.2023.01685

[39] Mao, W., Wang, T., Zhang, D., Yan, J., & Yoshie, O. (2023). *PillarNeXt: Embracing backbone scaling and pretraining for pillar-based 3D object detection*. arXiv preprint arXiv:2311.17770.

[40] Zheng, W., Tang, W., Jiang, L., & Fu, C.-W. (2021). SE-SSD: Self-ensembling single-stage object detector from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 14494–14503). https://doi.org/10.1109/CVPR46437.2021.01426

[41] Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition* (pp. 3354–3361). IEEE. https://doi.org/10.1109/CVPR.2012.6248074

[42] Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, *32*(11), 1231–1237. https://doi.org/10.1177/0278364913491297

[43] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (p. 448–456). pmlr.

[44] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980–2988). https://doi.org/10.1109/ICCV.2017.324

[45] Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263–7271). https://doi.org/10.1109/CVPR.2017.690

[46] Qi, C. R., Liu, W., Wu, C., Su, H., & Guibas, L. J. (2018). Frustum PointNets for 3D object detection from RGB-D data. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 918–927). https://doi.org/10.1109/CVPR.2018.00102

[47] Zhou, D., Fang, J., Song, X., Guan, C., Yin, J., Dai, Y., & Yang, R. (2019). IoU loss for 2D/3D object detection. In *2019 international conference on 3D vision (3DV)* (pp. 85–94). IEEE. https://doi.org/10.1109/3DV.2019.00019

[48] Ku, J., Mozifian, M., Lee, J., Harakeh, A., & Waslander, S. L. (2018). Joint 3D proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1–8). IEEE. https://doi.org/10.1109/IROS.2018.8594049

[49] Moenning, C., & Dodgson, N. A. (2003). *Fast marching farthest point sampling* (Tech. Rep.). University of Cambridge, Computer Laboratory.