3D Pedestrian Detection Based on Pointpillar: SelfAttention-pointpillar

YunXiang Liu[1]
School of Computer Science and Information Engineering
Shanghai Institute of Technology
Shanghai, China
yxliu@sit.edu.cn, 236142149@mail sit.edu.cn

Xun Pan[2]
School of Computer Science and Information Engineering
Shanghai Institute of Technology
Shanghai, China
787127839@qq.com

Jianlin Zhu[3]
School of Computer Science and Information Engineering
Shanghai Institute of Technology
Shanghai, China
877535678@qq.com

2024 9th International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS) | 979-8-3503-6304-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICIIBMS62405.2024.10792799

***Abstract*—With the rapid development of autonomous Vehicls technology, the detection of surrounding pedestrians, vehicles, Cyclists and other targets by autonomous vehicles is an indispensable technology, especially for pedestrian detection. Therefore, there aremore and more related algorithmms and network models based on target recognition. In recent years, many scholars have stagnated in the process of discovering new algorithms and models and have rarely improved (such as SECOND, PointRCNN, PointPillars, etc.) These classic models, due to the different configuration environments of these model codes and the need to redownload each time you want to run a new model, are very time consuming and energy consuming. In order to solve these difficulties, we chose to use the OpenPCDet target detection framework to improve these models. This framework integrates all the above original object detection models to facilitate us to improve and compare the indicators between the models, and in the comparison of the results of the original model built in the OpenPCDet framework, it is found that the PointPillars modelusing 3D single-stage object detection is the most suitable for autonomous Vehicles. The recognition speed of the original PointPillars for vehicles, pedestrians and other objects can fully meet the use of autonomous Vehicles technology, but the accuracy of object recognition, especially in pedestrian detection, needs to be improved. In this regard, we propose a SelfAttention-PointPillars model. Based on the architecture of the PointPillars model and the idea of self-attention, we use our own pillar amount and modify the original backbone structure into our own self-attention network to improve the accuracy of identifying target pedestrians. We also improved the original L1 loss function into a faster weighted L2 function and we also replaced the activation function with the more efficient LeakyRelu function. Therefore, this paper mainly introduces the OpenPCDet target detection framework design SelfAttention-PointPillars and ensure the recognition speed of the benchmark mode in the KITTI dataset while improving the pedestrian target detection accuracy of bev and 3D from 50 to about 60 and improving the accuracy of the improvement in difficult scenarios is increased by 15 percentage points.**

Keywords—3D object detection; PointPillars; OpenPCDet; autonomous Vehicles

I. INTRODUCTION

In order to compare the performance of each model more co-nveniently in the study of train related to autonomous vehicles target detection, we choose to use OpenPCDet as our framework for model training. OpenPCDet (OpenPCDet) has integratedvarious original models. Similarly, we can also set our own model and dataset according to the rules of the framework. In this framework, we can easily train and test the original method and can easily modify our model and train our own model. SelfAttention-PointPillars is also made using the OpenPCDet framework. In the target recognition network model, it can be divided into 2D target detection and 3D target detection. The most representative method of 2D target detection is end-to-end fully connected neural network, and the most representative of 3D is PointPillars and other models [1][2][3]. Although the 2D tar-get detection method is simple, it can only complete the positioning function of the 2D image target on the plane and cannot solve the common 3D problems in real life(Fig.1(a)(b)). Therefor, in order to make our detection range wider and the detection more local, we mainly carry out 3D target detection instead of 2Dtarget detection.

In 3D object detection, according to the steps required for the model to complete the detection, it can be divided into single-stage detection algorithm and two-stage detection algorithm. The most commonly

979-8-3503-6304-3/24/$31.00 ©2024 IEEE

Authorized licensed use limited to: Hochschul- und Kreisbibliothek Bonn-Rhein-Sieg. Downloaded on November 13,2025 at 20:25:53 UTC from IEEE Xplore. Restrictions apply.

used methods in single-stage are PointPillars, SECOND[4], etc. The most commonly used two stage method is Point RCNN[5] and its variant Fast Point R-CNN[6]. We conducted corresponding tests on the methods of these two stages in OpenPCDet and found that the recognition speed of the two-stage model is far less than that of the single stage model[7], which leads to the great security risk of the two stage model in autonomous Vehicles, so we focus more on single stage target recognition. In the detection results of vehicles, pedestrians and cyclist through these models, we found that the accuracy of pedestrian measurement is very low, so in order to solve this problem, we propose SelfAttention-PointPillars to improve its recognition accuracy. We modified the backbone network of the original model and turned the original network into a self-attention backbone network. At the same time, we also simplified its loss function and used a more efficient activation function to ensure the speed of its original model, which was verified in the Kitti data.

The contributions of this paper are as follows:

- We use the KITTI dataset to compare the objection recognition detection of the benchmark model above OpenPCDet. In this framework, a Self-attention is added to make the identified features more stable.
- We use PointPillars as the benchmark model and improve the Backbone network into a self-attention backbone network, and use a more efficient activation function and modify a simpler loss function, so that it can maintain the speed of the original model and improve the target recognition effect of pedestrians.

Figure 1(a): 2d object detection

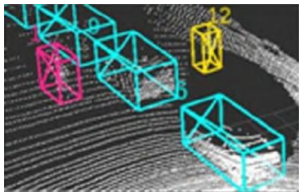

Figure 1(b): 3d object detection

II. RELATED WORK

A. *OpenPCDet*

OpenPCDet is an open source 3D object detection framework developed by the Open-MMLab team. OpenPCDet is a framework specifically designed for various target detection tasks for processing point cloud data structures. [3][4][5][8][9].It provides a rich model architecture and training, testing tool. so that in the model modification and model development learning can quickly build and train their own three dimensional target detection model. OpenPCDet is flexible and clear(Figure.2), OpenPCDet provides a variety of mainstream backbone frameworks, which can modify the corresponding network architecture according to their own needs. It defines Backbone3D, Backbone2D and DenseHead for each model, which can unify the rules for each target detection model and can be used or modified according to different needs.

Due to the differences in the definitions of coordinate systemand labels in various data sets, OpenPCDet adopts a fixed unified point cloud coordinate system [10] and a more standardized 3D detection frame definition[11], which runs through the calculation of the entire model data and the post detection process. In short, OpenPCDet can make it easier and more efficient for us to train and modify models.

The features of OpenPCDet：

- Unified modular design: OpenPCDet mainly uses the design idea of data model separation. OpenPCDet module uniformly defines standardized 3D coordinates so that point cloud data can be shared and processed by users.
- Efficient training and testing: OpenPCDet stores the configuration files of various models to facilitate the training and testing of models. OpenPCDet supports multi-GPU training, distributed training, etc. which greatly improves the efficiency of training.
- Rich original models: OpenPCDet provides many classic models including SECOND, PointRCNN, PointPillarss andother models for users to use directly.
- Loading a variety of datasets: In OpenPCDet, we can use Kitti, NuScene, Lyft, Waymo, PandaSet and other 3D object detection datasets, these datasets are also easy to setup.

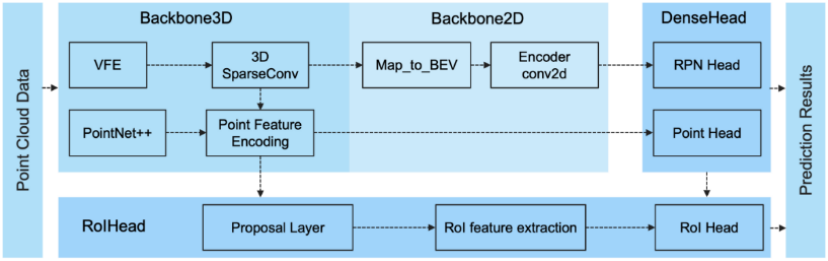

Figure 2: OpenPCDet structure

Authorized licensed use limited to: Hochschul- und Kreisbibliothek Bonn-Rhein-Sieg. Downloaded on November 13,2025 at 20:25:53 UTC from IEEE Xplore. Restrictions apply.

B. *Advantages of PointPillars*

At present, the network often used in point cloud 3D object detection has VoxelNet that directly voxelizes the feature point cloud data into 3D convolution [12]. PointPillars convert complex 3D spatial information into columnar data. SECOND network using sparse convolution to improve recognition speed. lightweight and effective point-based 3D single-stage target detector 3DSSD [13]and Two-stage PointRCNN network. Voxel-Net converts point cloud data into a voxelized representation that can result in information loss especially in sparse regions. This information loss may affect the final detection accuracy. The computational complexity of the VoxelNet algorithm is high and requires large computation-al resources. This limits its application in scenes with high real time requirements to a certain extent. So it is difficult to use in such projects as autonomous Vehicles. The SECOND network is an improvement of VoxelNet. SECOND uses the 3D convolution structure of VoxelNet. Although the detection accuracy is improved to a certain extent, the 3D convolution consumes a lot of GPU resources and has high hardware requirements. Due to the detection mechanism of SECOND, it may cause false detection in some complex scenes, especially when the target occlusion overlap is serious [3]. Although the speed of 3DSSD is relatively fast, there are still some difficulties in dealing with occlusion and multi-target, and algorithm needs to be further optimized [5]. Moreover, the source code of 3DSSD uses the SECOND model in the mm-detection 3d framework, which may limit its applicability in other frameworks, and its detection results will be less stable. The PointRCNN model requires complex operations in the process of feature extraction and proposal generation, which may lead to high computational complexity, and its calculation speed is too slow to be suitable for realtime target detection environment. Point-Pillars uses columnar data to simplify the subsequent data processing flow. After that, the encoder uses these columnar information to learn a set of features, and then disperses these features into 2D pseudo images. These images are used in the convolutional neural network[14], and finally we use the detection head [18] to perform the detection of 3D objects (Figure 3).At the same time, we compare the average recognition accuracy of these models(Table 1), It can be seen that PointPillars has the best recognition effect. So it can be concluded that the original PointPillars has the following advantages:

- efficiently process largescale point cloud data: efficiently process largescale point cloud data with efficient detection speed and achieve realtime target detection.
- Reduce computational complexity: PointPillars greatly reduces the computational complexity and improves the detection speed while maintaining the detection accuracy by using sparse convolution and other techniques.
- Using data complementarity: PointPillars can make full use of the complementarity of point cloud and image data, improve the accuracy and robustness of target detection, and has certain generalization ability for targets of different shapes and sizes.

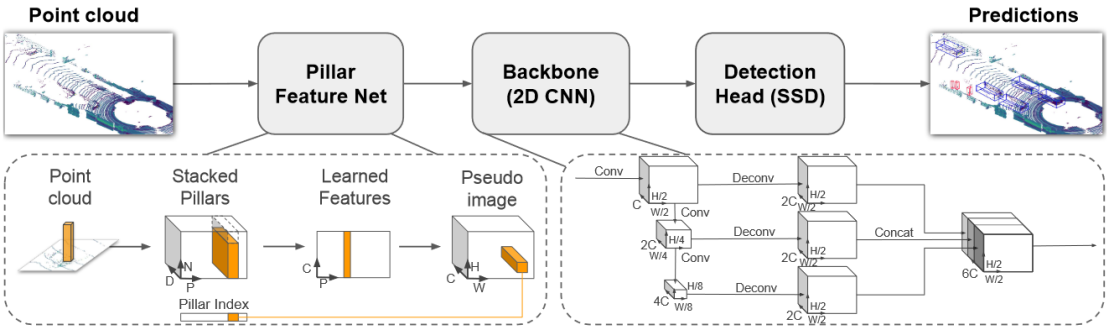

Figure 3: The raw PointPillars model

TABLE 1 : DATA COMPARISON BETWEEN POINTPILLARS MODEL AND OTHER MODELS

method	velocity _avg	car_ avg	pedestrian _ avg	cyclist_avg
PointPillars	0.0274	0.71	0.51	0.53
SECOND	0.0276	0.56	0.52	0.33
PointRCNN	0.0536	0.72	0.53	0.49
3DSSD	0.0204	0.71	0.65	0.55
VoxelNet	0.0024	0.49	0.35	0.46

C. *Advantages of Self-attention*

Although the traditional PointPillars model is superior to other models in terms of autonomous Vehicles, it is easy to lose some features in the feature ecoding part of the point cloud pillar, and the feature extraction of the pseudo image by the backbone network is not sufficient. If these problems can be solved, the detection accuracy of pedestrians can be improved. In order to improve the detection

Authorized licensed use limited to: Hochschul- und Kreisbibliothek Bonn-Rhein-Sieg. Downloaded on November 13,2025 at 20:25:53 UTC from IEEE Xplore. Restrictions apply.

accuracy of the target for pedestrians, we add point cloud coding weights to the pillar feature to enrich the feature coding. We added the self attention model and the spatial attention before the backbone features for comparison. Finally, we found that the spatial attention is not as effective as the self attention. The self-attention can transform the input image block into a feature vector, and each input element is represented by three matrices: query matrix, value matrix and key matrix matrix. Finally, the softmax function[18] is generally used to obtain a fractional vector, which represents the degree of attention of the current element and all other elements in the input sequence. Therefore, the self-attention mechanism is more convenient to process the global context information and the computational cost is basically the same as the spatial attention. The spatial attention is sensitive to noise, and the spatial resolution of the input data is strictly dependent, which makes it very unstable when processng pseudo feature images. So we choose our modified self-attention as the Entrance of the Self attention backbone network.

III. METHOD

A. Overall Architecture

The network model we proposed (Figure.4) We first divide the point cloud data into a grid according to the coordinate system where the point cloud data is located and then represent each point cloud with vector of Dimension is 9 to form a pillar data, The value of D is from the original coordinate (x, y, z) and reflection intensity r. The geometric coordinates of the point in pillar (x_e, y_e, z_e) and The difference between the original coordinates and the geometric center represents the relative position of the point $(x - x_e, y - y_e, z - z_e)$. Use N to represent the number of point clouds in pillar, E to denote the number of nonempty pillar, We can set a threshold for pillar to limit the number of point clouds inside each pillar, If the number of point clouds is less than the threshold, we set the pillar data to 0 so that we will get a (D, E, N) matrix X , After that, we pool the Xmatrix to achieve the dimension reduction operation to convert X into a feature map of (C, P) dimension. Here we also use the original model to convert the value of$P$ into $H \times W$, so that we can obtain a pseudo image of (C, H, W). After processing the original point cloud data into pseudo-images, we first use Max pooling and Avg pooling to generate two feature maps for fusion and then we use the self attention backbone network to further extract the features of the image. We use the self-attention to achieve the correlation between each feature and then use multiple 2D CNNs [15] to continuously reduce there solution of the feature map to extract the high dimensional features in the feature image. At the same time, we use the more efficient LeakyRelu [16] as the activation function to allow a small amount of gradient inflow for the value with negative output. The network processed in this way may have dimension noncorrespondence This time we can set the dimensions to the same dimension by upsampling and then link them. Finally, we use the target detection head similar to the SSD, we use the 2D intersection and comparison to match the prior box with the ground truth value to get the final result[17]. At the same time, we did not use L1 in PointPillars as a loss function but improved it into a weighted L2 [24] loss function for training.

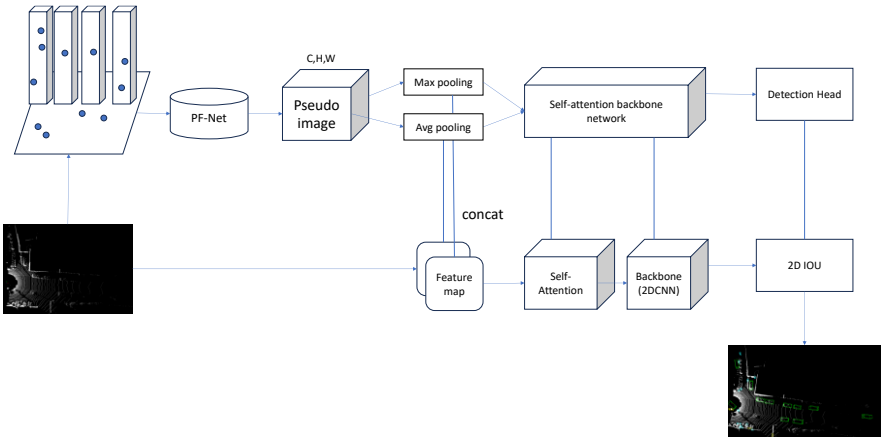

Figure 4: SelfAttention-PointPillars Overall Architecture

B. Self-attention backbone network

Self-attention backbone network (Figure.5) is mainly an improvement of the original backbone network, because PointPillars processes the 3D information of the original point cloud by means of columnar information, and then maps it into a pseudo-feature map for feature representation after feature extraction, After that, we can use the self-attention backbone network designed by us to process the two dimensional information first. First, we set the query matrix Q, the value matrix M and the key matrix V. We first use the matrix Q to multiply each vector of the feature map matrix M to obtain the corresponding weight S. Then the weight S is divided by the unit feature length $\sqrt{d_k}$ softmax function[18] to obtain the weight w_i of each feature. Theformula can be described as：

$$S_i = Q_i \times M^T{}_i \tag{1}$$

$$w_i = softmax\left(\frac{S_i}{\sqrt{d_k}}\right) \tag{2}$$

Let each pillar have its own corresponding weights so that the network can enhance the correlation between each pillar through these

Authorized licensed use limited to: Hochschul- und Kreisbibliothek Bonn-Rhein-Sieg. Downloaded on November 13,2025 at 20:25:53 UTC from IEEE Xplore. Restrictions apply.

weights when learning. We can also set the corresponding threshold. When the weight of the feature is too low, we can make the weight value of the feature evenly assigned to other features to improve the efficiency and speed of detection. We incorporate a weighted pillared pseudo feature image into a Max pooling[19] and an Avg pooling[20] respectively. Max pooling can reduce the size of the feature map, the amount of calculation and the number of parameters to help capture the most significant features, can better capture the edge and contour of the target suitable for local salient features. Avg pooling is to capture contextual information. It pays more attention to detecting the overall features and background information in the pseudo image to better capture the global features of the target. Then we connect the feature maps extracted from the two pooling layers through a 2D convolution layer with N3 × 3. There is a LeakyRelu[16] and batch norm behind each convolutional layer to keep the input of each layer of the neural network in the same distribution, prevent gradient disappearance and gradient explosion, and reduce the dependence of regularization. After the 2D convolution layer calculation, we also need to perform N deconvolution operations to upsample each feature. The activation function used in the deconvolution process is LeakyRelu and batch norm operations. Finally, we use SSD[21]to detect the target on each feature map location. We also use the 2D intersection over union[22] to compare the detection box generated by the SSD with the real value to complete all the detection targets.

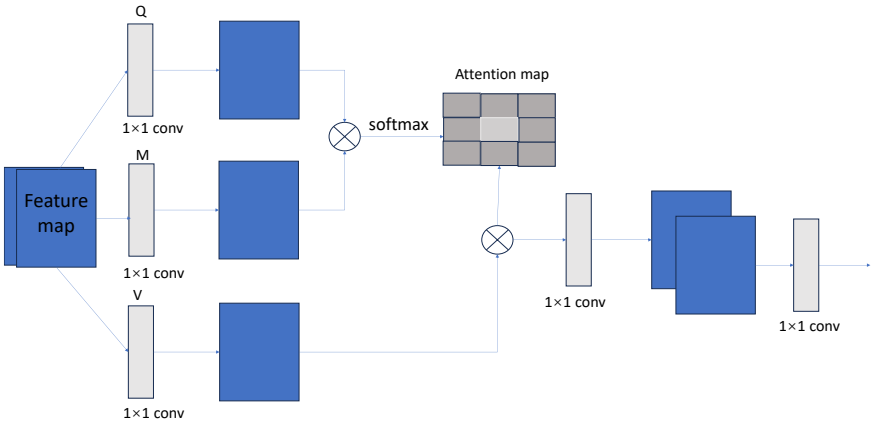

Figure 5: Self-attention backbone network

C. Loss Functions

In the loss function, we represent each 3D recognition box with a 7 dimensional vector. $(x, y, z, w, h, l, \theta)$ where (x, y, z) is the coordinate position of the recognition box, (w, h, l) is the size data of the recognition box, θ represents the direction angle of the recognition object. Because the smoothL1[23]used in the original model is sensitive to extreme values, it may lead to the problem of extreme prediction values. Therefore, there are many cases of false detection and missed detection, especially in the detection of pedestrians and cyclists. The problem of inaccurate identification is likely to occur. Therefore, we use the weighted L2 [24] loss function, which is more stable and smoother than smoothL1, to calculate the detection box regression: We first set seven weights for $(x, y, z, w, h, l, \theta)$ which are $(p_x, p_y, p_z, p_w, p_h, p_l, p_\theta)$

The corresponding Loss function of (x, y) is:

$$d^a = \sqrt{(w^a)^2 + (l^a)^2} \tag{3}$$

$$loss_i = \frac{(i^{gt} - i^a)^2}{d^a} \tag{4}$$

Where $i \in (x, y), i^{gt}$ and i^a represent the coordinates of the true value and the anchor box, respectively.

The loss function of z value can be expressed as:

$$loss_z = \frac{(z^{gt} - z^a)^2}{h^a} \tag{5}$$

The loss function of $j \in (w, l, h)$ can be expressed as:

$$loss_j = \left(log\frac{j^{gt}}{j^a}\right)^2 \tag{6}$$

Finally, the loss function of θ is expressed as:

$$loss_\theta = sin(\theta^{gt} - \theta^a) \tag{7}$$

The total regression loss function can be expressed as:

$$loss_{reg} = \sum_{\alpha \in (p_x, p_y, p_z, p_w, p_h, p_l, p_\theta), \beta \in (i,j,\theta)} \alpha loss_\beta \tag{8}$$

IV. EXPERIMENTS

A. Experiments Setting

Dataset setting At present, the commonly used computer vision data sets in the field of autonomous Vehicles are KITTI[25], Waymo and nuScenes[26][27]. Since the original PointPillars model is evaluated using the KITTI data set, the data set we use is also the KITTI data set to compare the original PointPillars model. The KITTI dataset contains the collection of real image data from urban, highway and rural scenes. Each image contains up to15 vehicles and 30 pedestrians, and there are various degrees of occlusion and truncation. The dataset is divided into three levels: easy, medium and hard. In the data set, we mainly set the parameter value of the processed data set POINT_CLOUD_RANGE to [0,-40,-3,70.4,40,1] to make the training data set more effective. In order to better adapt to the performance of our hardware, we set the MAX_POINTS_PER_VOXEL to 5, the training value in MAX_POINTS_PER_VOXEL to 16000, and the test value to 40000. In this way, we can unify the evaluation indexes of object recognition accuracy and recognition speed.

Authorized licensed use limited to: Hochschul- und Kreisbibliothek Bonn-Rhein-Sieg. Downloaded on November 13,2025 at 20:25:53 UTC from IEEE Xplore. Restrictions apply.

Measurements and Parameters According to the performance of our own GPU, CPU and other devices, we mainly set the value of BATCH_SIZE_PER_GPU to 4 and NUM_EPOCHS to 80 in OpenPCDet to give full play to the best performance of the device. At the same time, the trained MAX_NUMBER_OF_VOXELS is set to 16000, and the MAX_NUMBER_OF_VOXELS used in the test is set to 40000 to increase the recognition effect. The model's execution speed sec_per_example (s), car, pedestrian and cyclist 's AP value (%) of 3D detection and bev are used as evaluation indicator.

B. Results

We identify and detect SelfAttention-PointPillars and PointPillars according to the three targets of vehicle, pedestrian and cyclists in the KITTI dataset. By visualizing its data (Figure.6,7), we can see that the SelfAttention-PointPillars we use can detect targets missed by PointPillars, especially in pedestrian detection. Our model shows better results than PointPillars, At the same time, we compared with the test results of 3d in pointpilllar (Table.2,3,4), the test results of bev [28] (Table.5,6,7),and the running speed (Table.8). In the comparison of these test results, we can see that PointPillars and SelfAttention-PointPillars have very high recognition accuracy for vehicles in both 3d detection and bev detection. The effect of SelfAttention-PointPillars recognition is even better. The most noteworthy is that in the detection accuracy of pedestrians, we have increased the original detection accuracy by about 10 %, which also verifies the comparison results of Figure.6 and 7.

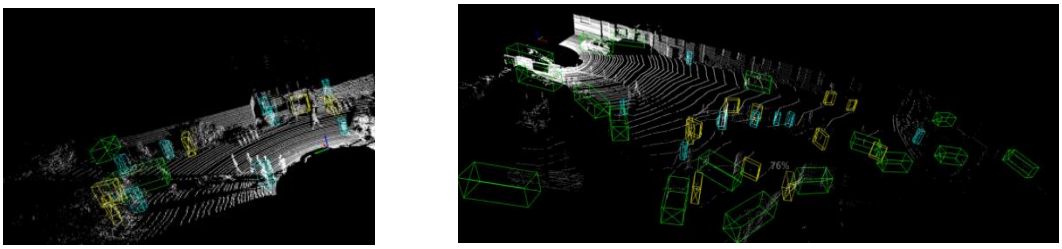

Figure.6: Results of PointPillar detection

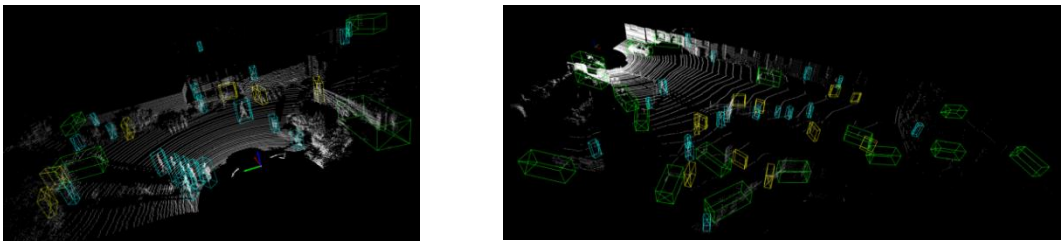

Figure.7: Results of detection SelfAttention-PointPillars

TABLE 2: CAR 3D AP VALUE DETECTION COMPARISON RESULTS

model /AP	**car_3d(easy)**	**car_3d(medium)**	**car_3d(hard)**
PointPillars	95.1549	93.5383	91.3549
ours	**95.4031**	**93.7817**	**91.4888**

TABLE 3: PEDESTRIAN 3D AP VALUE DETECTION COMPARISON RESULTS

	Pedestrian_3d(easy)	**Pedestrian_3d(medium)**	**Pedestrian_3d(hard)**
PointPillars	48.4317	42.8014	39.1502
ours	**66.6946**	**63.6075**	**61.3495**

Authorized licensed use limited to: Hochschul- und Kreisbibliothek Bonn-Rhein-Sieg. Downloaded on November 13,2025 at 20:25:53 UTC from IEEE Xplore. Restrictions apply.

TABLE 4: CYCLIST 3D AP VALUE DETECTION COMPARISON RESULTS

	Cyclist_3d(easy)	Cyclist_3d(medium)	Cyclist_3d(hard)
PointPillars	71.5131	53.7886	50.0979
ours	**70.8864**	**54.2967**	**50.6651**

TABLE 5: CAR BEV AP VALUE DETECTION COMPARISON RESULTS

	car_bev(easy)	car_bev(medium)	car_bev(hard)
PointPillars	95.2117	94.2855	93.0410
ours	**95.4439**	**94.1272**	**93.3336**

TABLE 6: PEDESTRIAN BEV AP VALUE DETECTION COMPARISON RESULTS

	Pedestrian_bev(easy)	Pedestrian_bev(medium)	Pedestrian_bev(hard)
PointPillars	54.8937	49.6262	46.3343
ours	**66.8145**	**63.8844**	**61.6408**

TABLE 7: CYCLIST BEV AP VALUE DETECTION COMPARISON RESULTS

	Cyclist_bev(easy)	Cyclist_bev(medium)	Cyclist_bev(hard)
PointPillars	75.4467	58.0293	54.618
ours	**74.2456**	**59.6551**	**54.8077**

TABLE 8: THE SPEED COMPARISON RESULTS OF EACH MODEL

Methods / Indicators	sec_per_example(s)
PointPillars	0.0274
SECOND	0.0266
pointRCNN	0.0536
3DSSD	0.0204
ours	**0.0276**

C. Ablation Study

The validity of the parameters in the overall experiment, we mainly changed the initial value of batch_size. We used the five values of [1,2,3,4,5,7] to observe the recognition accuracy of PointPillars and found that the convergence effect was best when the value of BATCH_SIZE was 4. We also use [0.001,0.003,0.1] and [0.01,0.04,0.06] to arrange and combine the two parameters of LR and WEIGHT_DECAY respectively. Finally, when the value of BATCH _ SIZE is constant, LR is 0.003 and WEIGHT _ DECAY is 0.04, the effect is the best.

Comparison of activation function and loss function (Table.9,10) We added the corresponding network, although the accuracy has been greatly improved, but its speed has not reached our expected standard. At this time, we first replaced the ReLu activation function used in the original model with the Sigmoid function. The experiment finally found that the accuracy of the output was reduced.

Authorized licensed use limited to: Hochschul- und Kreisbibliothek Bonn-Rhein-Sieg. Downloaded on November 13,2025 at 20:25:53 UTC from IEEE Xplore. Restrictions apply.

We replace the original activation function with the Tanh function [29]and the PReLU for testing. It is found that although the accuracy can be maintained, the speed is still very slow. Finally, we use the LeakyRelu to find that the improved accuracy is more stable and maintains the high accuracy of the vehicle while the accuracy of pedestrian recognition becomes higher, but the speed needs to be optimized. Finally, we think that the number of parameters and the amount of calculation used in the original loss function are too large. We use 0-1 Loss, Mean Square Error (MSE) [30], CrossEntropy Loss [31] and weighted L2 loss[24] respectively. Finally, we find that the results trained by the weighted L2, except for other loss functions, although the speed will be greatly improved, the accuracy is very unstable. Finally, we find that weighted L2 can not only maintain the speed corresponding to the original model but also maintain the recognition accuracy of the new model.

TABLE 9: COMPARISON RESULTS OF ACTIVATION FUNCTIONS

activation function	sec_per_example(s)	Car AP_avg	Pedestrian AP_avg	Cyclist AP_avg
Sigmoid	0.0089	0.50	0.25	0.34
Tanh	0.0286	0.50	0.50	0.50
PReLU	0.0289	0.61	0.50	0.50
ReLu	0.0274	0.71	0.51	0.53
Leaky Relu(ours)	**0.0276**	**0.75**	**0.65**	**0.54**

TABLE 10: THE SPEED COMPARISON RESULTS OF THE LOSS FUNCTION(AT THIS TIME, THE ACTIVATION FUNCTION HAS BEEN SELECTED TO LEAKY RELU)

loss function	sec_per_example(s)	Car AP_avg	Pedestrian AP_avg	Cyclist AP_avg
CrossEntropy Loss	0.0126	0.67	0.25	0.25
Mean Square Error	0.0125	0.68	0.25	0.25
0-1 Loss	0.0094	0.45	0.15	0.16
L1 loss	0.0245	0.71	0.51	0.53
weighted L2(ours)	**0.0276**	**0.75**	**0.65**	**0.54**

Selection of model First, we add some 2D convolution operations to the backbone network so that we feel that we can more easily extract the feature of the identified objects, but the results show that all the recognition results have been improved but the recognition speed has slowed down by nearly twice, and the prediction effect of pedestrians and Cyclists is still poor. Then we use the attention model which can assign different weights to the input features of the feature images. Because the accuracy of vehicle recognition is very high, we use the idea of attention [32]to improve the weight value of pedestrian recognition, which can make the model focus on the key information adaptively and better. The final results show that the recognition accuracy of vehicle and cyclist is also improved without retaining the recognition accuracy of vehicle and cyclist.

V. CONCLUSION

In this paper, we propose a simple and effective SelfAttention-PointPillars model. It uses the pseudo-image generated by PF-Net to reduce the dimension of the original 3D image to improve the efficiency of recognition. Then the proposed features are pooled to capture the most significant features and then trained in our self-attention backbone network to improve the accuracy of pedestrian target recognition. This method has a significant improvement in the recognition effect of pedestrians, but some of the original feature information cannot be well retained when converting 3D images into pseudo-images. If these information can be retained more, the recognition effect of difficult scenes will certainly have a good improvement. Therefore, what we want to do next is to improve the process of converting 3D images into pseudo-images, so that the recognition accuracy of complex scenes can be improved.

REFERENCES

[1] Jules Karangwa, Jun Liu, and Zixuan Zeng. 2023. Vehicle Detection for Autonomous Driving: A Review of Algorithms and Datasets. Trans. Intell. Transport. Sys. 24, 11 (Nov. 2023), 11568–11594. https://doi.org/10.1109/TITS.2023.3292278

[2] Wang, Jianfeng et al. “End-to-End Object Detection with Fully Convolutional Network.” 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020): 15844-15853.

Authorized licensed use limited to: Hochschul- und Kreisbibliothek Bonn-Rhein-Sieg. Downloaded on November 13,2025 at 20:25:53 UTC from IEEE Xplore. Restrictions apply.

[3] Lang, Alex H. et al. “PointPillars: Fast Encoders for Object Detection From Point Clouds.” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2018): 12689-12697.

[4] Yan Y, Mao Y, Li B. SECOND: Sparsely Embedded Convolutional Detection. Sensors. 2018; 18(10):3337. https://doi.org/10.3390/s18103337

[5] Shi, Shaoshuai et al. “PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud.” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2018): 770-779.

[6] Y. Chen, S. Liu, X. Shen and J. Jia, "Fast Point R-CNN," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 9774-9783, doi: 10.1109/ICCV.2019.00987.

[7] Zhu, C., Chen, F., Shen, Z., Savvides, M. (2020). Soft Anchor-Point Object Detection. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, JM. (eds) Computer Vision – ECCV 2020. ECCV 2020. Lecture Notes in Computer Science(), vol 12354. Springer, Cham. https://doi.org/10.1007/978-3-030-58545-7_6

[8] Chen, Xuesong & Shi, Shaoshuai & Zhu, Benjin & Cheung, Ka & Xu, Hang & Li, Hongsheng. (2022). MPPNet: Multi-frame Feature Intertwining with Proxy Points for 3D Temporal Object Detection. 10.1007/978-3-031-20074-8_39.

[9] Chen, Yukang et al. “Focal Sparse Convolutional Networks for 3D Object Detection.” 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2022): 5418-5427.

[10] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu and M. Bennamoun, "Deep Learning for 3D Point Clouds: A Survey," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 12, pp. 4338-4364, 1 Dec. 2021,

[11] Zamanakos, Georgios et al. “A comprehensive survey of LIDAR-based 3D object detection methods with deep learning for autonomous driving.” Comput. Graph. 99 (2021): 153-181.

[12] Zhou, Yin and Oncel Tuzel. “VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection.” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2017): 4490-4499.

[13] Yang, Zetong et al. “3DSSD: Point-Based 3D Single Stage Object Detector.” 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020): 11037-11045.

[14] D. Tran, L. Bourdev, R. Fergus, L. Torresani and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 4489-4497, doi: 10.1109/ICCV.2015.510.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.

[16] T. Jiang and J. Cheng, "Target Recognition Based on CNN with L-eakyReLU and PReLU Activation Functions," 2019 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC), Beijing, China, 2019, pp. 718-722, doi: 10.1109/SDPC.2019.00136.

[17] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. International Journal of Computer Vision, 2010.

[18] Liang, X., Wang, X., Lei, Z., Liao, S., Li, S.Z. (2017). Soft-Margin Softmax for Deep Classification. In: Liu, D., Xie, S., Li, Y., Z-hao, D., El-Alfy, ES. (eds) Neural Information Processing. ICONIP 2017. Lecture Notes in Computer Science(), vol 10635. Springer, Cham. https://doi.org/10.1007/978-3-319-70096-0_43

[19] Yu, D., Wang, H., Chen, P., Wei, Z. (2014). Mixed Pooling for C-onvolutional Neural Networks. In: Miao, D., Pedrycz, W., Ślęzak, D., Peters, G., Hu, Q., Wang, R. (eds) Rough Sets and Knowledge Technology. RSKT 2014. Lecture Notes in Computer Science(), vol 8818. Springer, Cham. https://doi.org/10.1007/978-3-319-11740-9_34

[20] Zhao, L., Zhang, Z. A improved pooling method for convolutional neural networks. Sci Rep 14, 1589 (2024). https://doi.org/10.1038/s41598-024-51258-6

[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In ECCV, 2016.

[23] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid and S. Savarese, "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 658-666, doi: 10.1109/CVPR.2019.00075

[24] C. Liu et al., "Adaptive Smooth L1 Loss: A Better Way to Regress Scene Texts with Extreme Aspect Ratios," 2021 IEEE Symposium on Computers and Communications (ISCC), Athens, Greece, 2021, pp. 1-7, doi: 10.1109/ISCC53001.2021.9631466.

[25] Li, Z., Zhou, X. Weighted L2 Approximation of Analytic Sections. J Geom Anal 32, 44 (2022). https://doi.org/10.1007/s12220-021-00772-4

[26] Geiger A, Lenz P, Stiller C, Urtasun R. Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research. 2013;32(11):1231-1237. doi:10.1177/0278364913491297

[27] P. Sun et al., "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 2443-2451, doi: 10.1109/CVPR42600.2020.00252.

[28] Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., & Beijbom, O. (2019). nuScenes: A Multimodal Dataset for Autonomous Driving. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 11618-11628.

[29] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In CVPR, 2012 .

[30] Xin Wang, Yi Qin, Yi Wang, Sheng Xiang, Haizhou Chen,ReLTanh: An activation function with vanishing gradient resistance for SAE-based DNNs and its application to rotating machinery fault

Authorized licensed use limited to: Hochschul- und Kreisbibliothek Bonn-Rhein-Sieg. Downloaded on November 13,2025 at 20:25:53 UTC from IEEE Xplore. Restrictions apply.

diagnosis,Neurocomputing,Volume 363,2019,Pages 88-98,ISSN 0925-2312,https://doi.org/10.1016/j.neucom.2019.07.017.

[31] Kim, Taehyeon et al. “Comparing Kullback-Leibler Divergence and Mean Squared Error Loss in Knowledge Distillation.” International Joint Conference on Artificial Intelligence (2021).

[32] Mao, A., Mohri, M., & Zhong, Y. (2023). Cross-Entropy Loss Functions: Theoretical Analysis and Applications. ArXiv, abs/2304.0728

Authorized licensed use limited to: Hochschul- und Kreisbibliothek Bonn-Rhein-Sieg. Downloaded on November 13,2025 at 20:25:53 UTC from IEEE Xplore. Restrictions apply.