Neural Computing and Applications (2022) 34:6637–6645
https://doi.org/10.1007/s00521-021-06061-z

S.I ON NC FOR INDUSTRY 4.0

GVnet: Gaussian model with voxel-based 3D detection network for autonomous driving

Peilin Qin[1] · Chuanwei Zhang[1] · Meng Dang[1]

Received: 20 January 2021 / Accepted: 19 April 2021 / Published online: 17 May 2021

Abstract
This paper proposed a two-stage Voxel-based 3D Object detector which named GVnet. Voxel-based method mainly relies on sampling and Grouping point in voxel and the feature map generated by subsequent 3D CNN to control the quality of detection. Moreover, traditional voxel feature encoder (VFE) methods cannot adjust the quality of feature map through reasonable sampling. Therefore, the method we propose is an improvement to the existing VFE. The specific operations are: First calculate the corresponding Gaussian distribution of the original point cloud data, and then sampling any number of points by controlling the confidence value to improve the performance of voxel encoder and further improve the quality of the feature map output by the 3D CNN. In addition, a voxel ROI pooling method is proposed in stage two. In ROI Pooling, the receptive field in the original space and the corresponding raw point are obtained through the mapping relationship between feature and ROI, then change the raw point to adjust the receptive field to improve the performance of classification and regression. Finally, the experimental results on the KITTI, nuScenes and Waymo dataset show that the performance of GVnet under most of the evaluation indexes is better than the current detection methods, at the cost of only a small amount of inference time.

Keywords 3D detection · Voxel-based detector · Gaussian-voxel feature encoding · Voxel-ROI pooling

1 Introduction

Three-dimensional detection is one of the most important issues in the environment perception of autonomous driving, and many outstanding researches have emerged. These tasks are usually divided into the types of sensors used: Lidar-based and multi-sensor fusion (usually RGB + Lidar).

Some representative works based on multi-sensor fusion such as: Chen et al. [1] use a compact multi-view representation to encode sparse 3D point clouds, and propose a multi-layer fusion strategy. Ku et al. [2] further extend [1], by inputting RGB image and BEV (Bird's Eye View) Map, using FPN [3] network to obtain the full resolution feature map of the two, and then extracting two feature map correspondences through crop&resize Feature crops are integrated to achieve 3D inspection. Liang et al. [4] Proposed Continuous Fusion Layer to fuse multi-scale image features into radar features. Qi et al. [5] use the combination of dimensionality reduction technology and mature 2D object detectors to complete the detection process. Liang et al. [6] can achieve a fully integrated feature representation by using the fusion between point-wise and ROI-wise features. Xu et al. [7] process the image data and original point cloud data separately with CNN and PointNet architectures, and then combine the output obtained by the new fusion network. Wang et al. [8] designed a dense pixel-level fusion method to integrate the features of RGB data and point cloud in a more appropriate way. Vora et al. [9]project each lidar point into the output of the image semantic segmentation network, and connects the channel direction activation with the intensity measurement of each

✉ Chuanwei Zhang
20105016012@stu.xust.edu.cn

Peilin Qin
19205016012@stu.xust.edu.cn

Meng Dang
dangmeng@xust.edu.cn

[1] School of Mechanical Engineering, Xi'an University of Science and Technology, Xi'an 710054, Shaanxi, China

lidar point to strengthen the point cloud through the image semantics.

For detection methods based on multi-sensor fusion, most of the work is to project point clouds onto RGB images to obtain dense expression, although in principle they can combine the advantages of multiple sensors to further improve the detection result, but the specific fusion method still needs further research and development, including how to improve the fusion effect and reduce the processing time.

Some representative works of Lidar-based detection methods such as: Zhou et al. [10] cooperate with BEV and perspective drawing, effectively using the information of the two to complement each other, and proposes the concept of dynamic voxelization, so that each point can be different Perspective learning integrates contextual information. Yang et al. [11] proposed a point-based spherical anchor point cloud target detection scheme generation model, which is universal to achieve high recall. And put forward the PointsPool layer, which integrates the advantages of point-based and voxel-based methods to achieve efficient and effective prediction. Lang et al. [12] use [13] to learn the characteristics of the point cloud in the vertical columnar organization, and convert the complex three-dimensional point cloud space into a two-dimensional plane space. Shi et al. [14] detect 3D objects from the original point cloud, and directly generates a 3D scheme from the point cloud, which has a higher recall rate than the previous scheme generation method. Yang et al. [15] use a BEV to represent the scene to estimate three-dimensional objects based on pixel-level neural networks. Ali et al. [16]use the method of deconvolution and connect the features layer by layer, so that the network can retain location information when acquiring deeper feature information. Shi et al. [17] further expand [14] and makes better use of box label information.Kuang et al. [18] improved [19], processed voxel of multiple scales, and improved detection accuracy by fusing with feature maps of different layers. Chen et al. [20] effectively integrate the coordinate and index convolution features of each point with the attention mechanism, which not only retains accurate positioning information, but also retains context information. Yang et al. [21]designed a new SA module and discarded the FP module [22], making the detection time up to 25FPS.

For Lidar-based detection, part of the work uses the Point-based method to directly process each point. The advantage of this method is that the information obtained is more sufficient, and the experience field is flexible and variable, but the disadvantage is that for the large scene of unmanned driving, the number of points is often very large, and the point-based method will encounter the challenge of huge amount of calculation. Another part of the work uses the voxel-based method, which uses 3D CNN to reduce computing time by converting the original data space into a series of voxels, but the disadvantage is that the quality of the voxel feature directly affects the detection results, and the receptive field cannot be flexibly changed.

In response to the above problems, this paper proposes a Voxel-based method that only uses Lidar information as input. Grouping raw points by establishing a Gaussian model, by sampling and grouping to generate high-quality feature map and further improve the detection accuracy.

2 Our method

This paper proposes a two-stage detector Gaussian-Voxel Detection Network (GVnet). In stage one, Gaussian-Voxel Feature Encoding is used for the raw point cloud, then voxelization is carried out, next, 3D CNN is used to generate high-quality feature maps, the feature map is passed as input to the RPN network [23] to generate a series of 3D proposals.

In stage two, voxel-ROI pooling was used to improve the RPN performance. The corresponding receptive field is obtained through the mapping relationship between raw point and feature map in voxel. Then, the receptive field is regulated by sampling any point of the Gaussian model corresponding to the raw point. This makes the features corresponding to the proposal stronger, and improves the effect of classification and regression tasks.

The overall framework of GVnet is shown in Fig. 1.

2.1 Gaussian-voxel feature encoding

Traditional VFE [17] first needs grouping and sampling raw point, and these two steps directly affect the quality of subsequent refinement. The usual operation in Sampling processing is to unify the number of points contained in each voxel. Suppose we set the number of points contained in each voxel to k. If the number of points actually contained in the Voxel is j, when $j > k$, the points in voxel are randomly dropout. If $j < k$, add a certain amount of points to this voxel. The supplement methods include: (1) Add a series of points with a value of 0. (2) Copy some points in voxel. Obviously, there is a gap between the characteristics of the point and the raw point obtained after sampling using the above method, which is not conducive to the subsequent 3D CNN for feature extraction.

In response to the above problems, this article proposes a new VFE method. First, perform Gaussian clustering on the raw data to obtain a series of Gaussian models that the class obeys, and then voxelize the overall data. When sampling each non-empty voxel, use the Gaussian model corresponding to the voxel to perform this operation. This can keep the generated point and raw point features similar

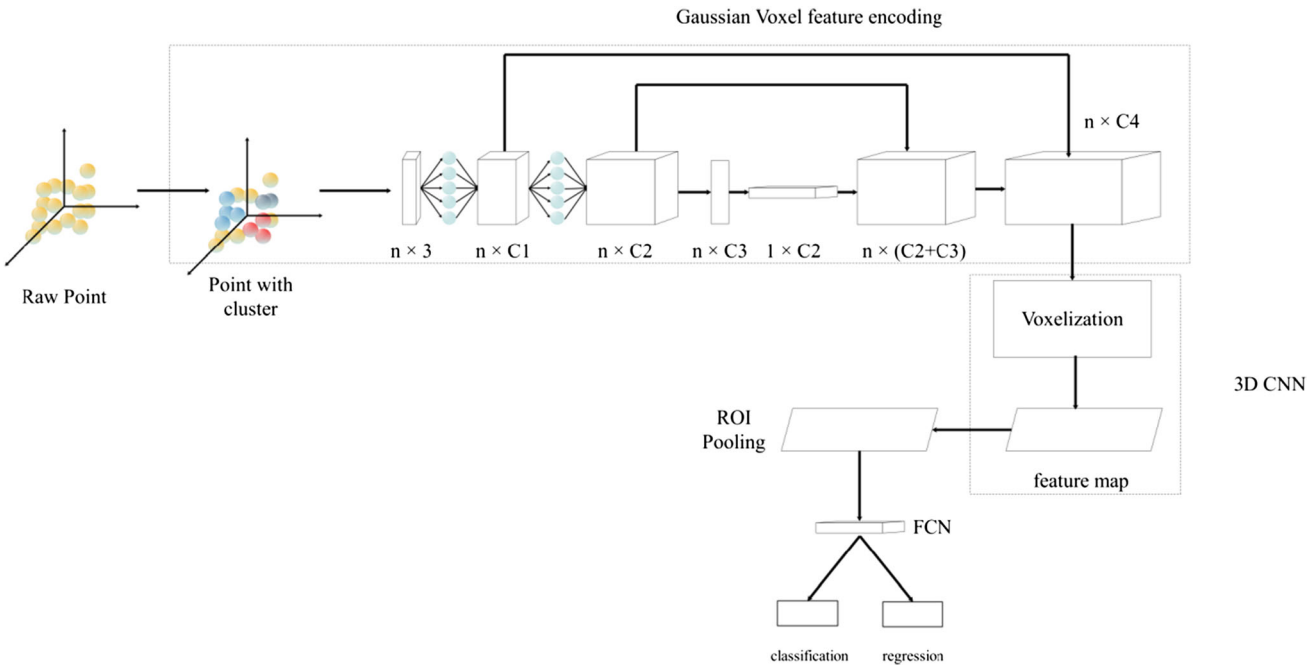

Fig. 1 GVnet framework

to the greatest extent, and improve the effect of the encoder. For the disorder of point cloud data, the Gaussian mixture model(GMM) [24] can be used for a good description.

First define x_j to represent the No.j data, where $j = 1, 2, 3, ..., N$. K is the number of Gaussian models included in the GMM. α_k is the probability that a certain data belongs to the No.k Gaussian model, where $\alpha_k \geq 0, \sum_{k=1}^{K} \alpha_k = 1$. $\varphi(x|\theta_k)$ is the distribution function of the No.k Gaussian model. γ_{jk} is the probability that the No.j data belongs to the No.k Gaussian model. Hence, the distribution of the Gaussian mixture model is:

$$P(x|\theta) = \sum_{k=1}^{K} \alpha_k \varphi(x|\theta_k) \tag{1}$$

When calculating θ. Use Maximum Log-Likelihood to calculate. Specifically:

$$\log L(\theta) = \sum_{j=1}^{N} \log P(x_j|\theta) = \sum_{j=1}^{N} \log \left(\sum_{k=1}^{K} \alpha_k \varphi(x|\theta_k) \right) \tag{2}$$

By initializing the model parameters, iteratively calculates all the parameters in the model. Each iteration includes two steps. The first step is to calculate the probability that each data j comes from the model k based on the current parameters:

$$\gamma_{jk} = \frac{\alpha_k \varphi(x_j|\theta_k)}{\sum_{k=1}^{K} \alpha_k \varphi(x_j|\theta_k)} \tag{3}$$

Then calculate the model parameters for the new iteration:

$$\begin{aligned} \mu_k &= \frac{\sum_j^N (\gamma_{jk} x_j)}{\sum_j^N \gamma_{jk}} \\ \Sigma_k &= \frac{\sum \gamma_{jk} (x_j - \mu_k)(x_j - \mu_k)^T}{\sum_j^N \gamma_{jk}} \\ \alpha_k &= \frac{\sum_{j=1}^N \gamma_{jk}}{N} \end{aligned} \tag{4}$$

Repeat the above steps until the parameters converge to get the Gaussian mixture model corresponding to the original point cloud, as shown in Fig. 2.

It can be seen from Fig. 2 that the data processed by GMM obey their respective distributions, and the blue point is the center of the class. By sampling within the corresponding Gaussian distribution, the quality of the points that are subsequently sent to the MLP network to extract features is improved, and a high-quality feature map is further obtained for use in stage two.

2.2 Voxel-ROI pooling

In order to obtain higher accuracy, the feature map generated by 3D CNN is first sent to the RPN network to generate a series of 3D proposals, and then the space size is the same through ROI Pooling [23], and the corresponding ROI vector is output to complete the classification and return. The overall process is shown in Fig. 3:

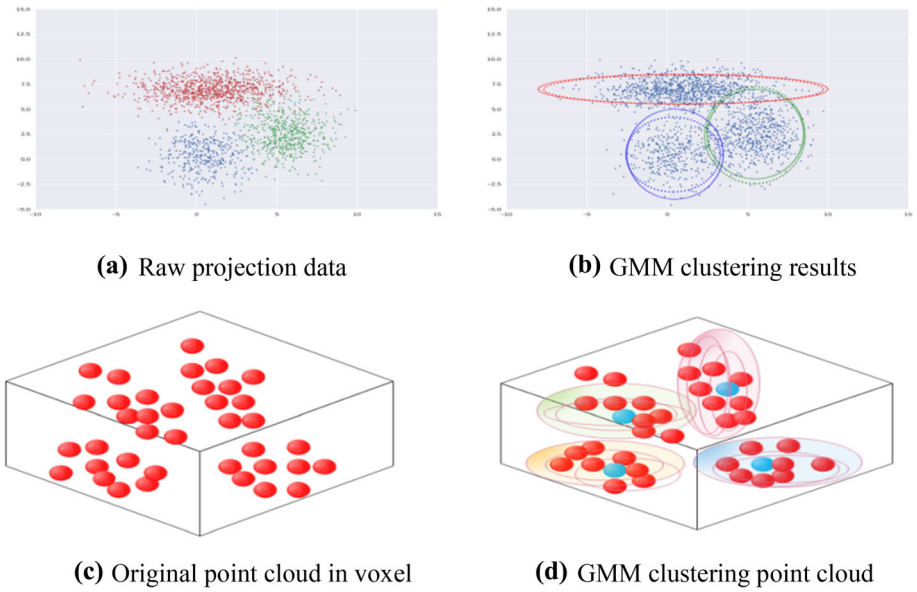

Fig. 2 GMM result comparison

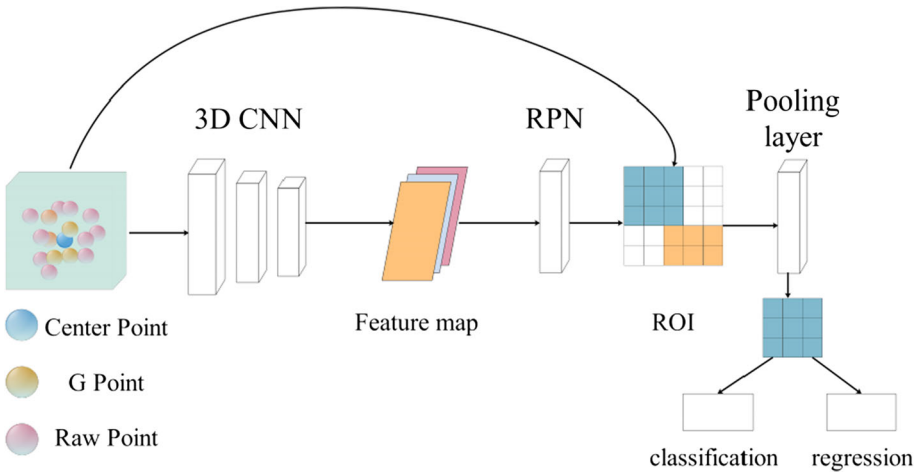

Fig. 3 Stage two work process

G Point refers to a point with high confidence that obeys the corresponding Gaussian distribution.3D sparse convolution is chosen as the backbone of 3D CNN model. The voxelization of point cloud will generate about 5 K $\sim$ 8 K voxels and about 0.005 sparse degree. Direct use of 3D convolution will consume huge computing time and memory, which can be avoided by sparse convolution. [25] limits the sparsity of output through the sparsity of input data, thus greatly reducing the computational amount of subsequent convolution operations.

A major disadvantage of using voxel-based detection is that it cannot flexibly adjust the receptive field. In response to this problem, this paper designs a pooling method with better feature acquisition performance, named voxel-ROI Pooling. The specific method is to map the feature corresponding to the 3D proposal back to Voxel before pooling processing, and obtain the Gaussian distribution information of the corresponding point. The mapping relationship is similar to the mapping relationship between feature and ROI in 2D CNN [26], as shown in Fig. 4.

As shown in Fig. 4, (x, y) represents the coordinate point on the feature map, and (x', y') represents the coordinate point in the original space corresponding to the point. The conversion relationship between the two is:

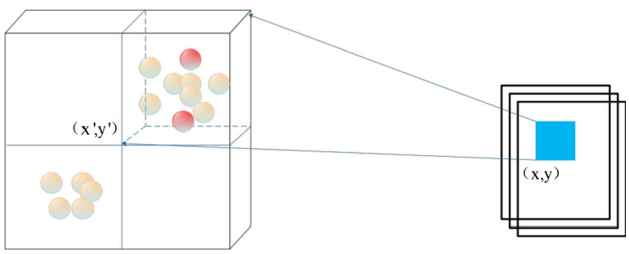

Fig. 4 ROI to raw data mapping relationship

$$(x^{'}, y^{'}) = (Sx, Sy) \tag{5}$$

$$S = \prod_{0}^{i} s_i \tag{6}$$

s represents the stride size of the middle layer. After obtaining the feature representation in raw data, this article further uses the sampling m points with the highest confidence in the raw point corresponding to each proposal as feature points to control the size and characteristics of the receptive field. The specific principle is shown in Fig. 5:

When controlling the receptive field, according to the formula:

$$Q = (W - K + 2P)/S + 1 \tag{7}$$

whereas Q is the feature size, W is the input size, K is the convolution kernel size, P is padding, and S is stride. The calculation formula of the input size can be further obtained:

$$W = (Q - 1) \cdot S - 2P + K \tag{8}$$

W here is the receptive field corresponding to the feature, and the point in the receptive field directly affects the output feature. Therefore, when doing pooling, the Voxel-ROI Pooling method can effectively adjust the receptive field and further enhance the characteristics of Voxel.

Finally, these features are sent to a layer of MLP network to output a fixed-size one-dimensional vector to further obtain the feature expression for detection.

2.3 Loss function

The loss function of the network is divided into two parts. The first part is used for classification. Because the classification of positive and negative samples of point cloud data is more uneven, so Focal Loss [27] is selected as the solution, as shown in the formula:

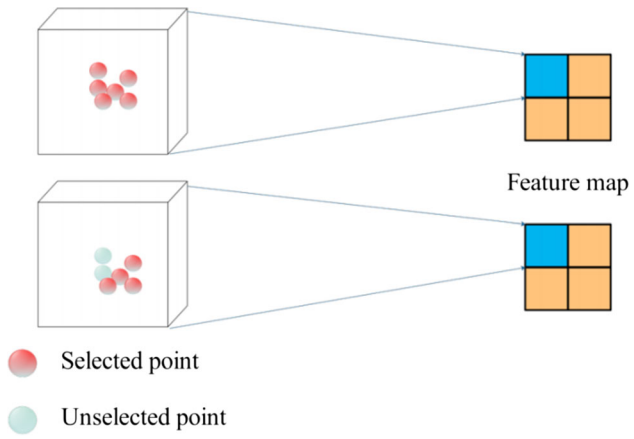

Fig. 5 Correspondence between receptive field and raw point

$$L_{cls} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{9}$$

For the negative sample $(1 - p_t)^{\gamma}$ with a larger probability to approach 0, its loss value can be reduced, thereby effectively suppressing it. For positive samples with low probability, $(1 - p_t)^{\gamma}$ has little effect on its loss, so we can increase the contribution of difficult samples to the gradient by reducing the loss of simple samples. Through the experiment of the paper, it was found that when $\gamma = 2$, $\alpha = 0.25$, GVnet has the best performance.

For the second part, smooth_{L1} [28]is used for bounding box regression:

$$L_{\text{reg}}(t^{\mu}, v) = \sum_{i \in \{x,y,z,l,w,h\}} \text{smooth}_{L1}(t_i^{\mu} - v_i) \tag{10}$$

In which:

$$\text{Smooth}_{L1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases} \tag{11}$$

Further, the total loss function can be obtained:

$$L_{\text{total}} = L_{\text{cls}} + L_{\text{reg}} \tag{12}$$

The total loss function is the sum of the above two parts without any weighting. The training details of the loss function will be explained in the experiment part.

3 Experiment

This part introduces some details of GVnet training and the specific experimental performance on KITTI [29], nuScenes [30], Waymo [31] dataset, and compares the effects with some other commonly used 3D detection models. At the end, an ablation experiment was carried out, which showed that the G-VFE and Voxel-ROI pooling proposed in this paper can effectively improve the accuracy of the model.

3.1 Training step

The KITTI dataset contains real image data collected from scenes such as urban areas, rural areas, and highways. Each image can contain up to 15 cars and 30 pedestrians, with various degrees of occlusion and truncation. In addition, KITTI dataset is also one of the important data sets in the field of autonomous driving at present, so this experiment chooses to use KITTI as the data set for network performance evaluation and Average Precision (AP) as the evaluation index [32]. And divide 40% of the training set into a validation set to monitor the performance of the model in real time to prevent over-fitting.

The nuScenes dataset consists of 1000 scenes, each of which is 20 s long and contains a variety of scenarios. In

each Scenes, there are 40 key frames, that is, 2 key frames per second, and the other frames are sweeps. Key frames are manually annotated, and there are several annotations in each frame in the form of bounding box. Not only size, range, but also category, visibility, and so on.

The Waymo dataset contains 1000 driving segments, each consisting of 20 s of continuous driving footage. Images of vehicles, pedestrians, bicycles, signage, etc., were carefully tagged, with a total of 1200 3D tags and 1.2 million 2D tags captured.

The network framework GVnet proposed in this paper is trained on a single NVIDIA RTX3090, and the training time is about 15 h for KITTI, 20 h for nuScenes, 30 h for Waymo.

In order to prevent over-fitting, a certain data augmentation method is used. First, the raw data are randomly flipped along the x-axis, and then rotated by $\left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$ along the z-axis, and finally the raw data are scaled by a certain proportion, and the zoom range is $[0.95, 1.05]$.

For each class sample in voxel, 15 points are used as features, and the threshold of NMS is set to 0.8, voxel size is $[0.1 \quad 0.1 \quad 0.2]$, for KITTI, the number of cluster sets is 10, 20 for nuScenes and Waymo. Adam is used as the optimizer, batch size is 4, total epoch is 80, and the initial learning rate is set to 0.01. In the 30th–50th epoch, the learning rate is gradually changed to 0.0001. The convergence of the loss function is shown in Fig. 6:

As can be seen from Fig. 6, because GVnet uses G-VFE and uses a more reasonable sampling operation, the initial Loss value is lower than other methods, which helps the model to converge quickly and can further improve performance.

Performance on dataset.

Table 1 shows the performance of GVnet and other network models under the KITTI test set. Through comparison, it can be seen that in terms of 3D detection, the performance of GVnet has a certain accuracy improvement compared with other methods. In the case of AP70, the three modes of easy, moderate and hard have increased by 0.3, 0.46, and 0.82, respectively. In terms of BEV detection, GVnet has also improved to varying degrees.

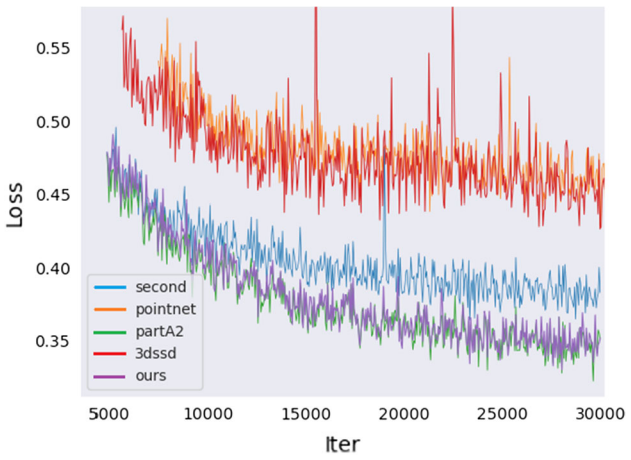

Fig. 6 Comparison of Loss convergence with different models

Table 2 shows the performance of GVnet on the val set. It can be seen that GVnet is a two-stage network framework. Although a part of the inference time is added, the gain is an increase in accuracy, which is consistent with the design concept of the two-stage detector.

Table 3 shows the performance of GVnet and other network models under the nuScenes dataset. It can be seen that our method has a good performance in most class, and mAP improved by 2.07 compared to Point Pillars.For datasets of such large scenarios, GVnet can still achieve good performance by increasing the number of clusters sets.

Table 4 shows the performance of GVnet and other detection methods under the Waymo dataset. For 3D detection task, GVnet improved mAP by 3.78 compared to other methods, for BEV detection, GVnet improved mAP by 0.45.

In terms of point cloud visualization, each frame of KITTI dataset contains a large number of points, so the use of Meshlab and other software can not display the scene details well. Therefore, in order to better demonstrate the detection effect of GVnet, this experiment visualized the detection results on KITTI Viewer Web [23], and some of the results are shown in Fig. 7.

It can be seen from Fig. 7 that GVnet has a good performance in terms of recall rate and accuracy, but the price is that more complicated sampling operations lead to false detection of some plausible targets.

3.2 Ablation experiment

In order to better explain the rationality of the G-VFE and V-ROI Pooling operations, this paper carried out an ablation experiment using KITTI dataset. G-VFE and original VFE are used as the encoder method of the network backbone, and then the original ROI Pooling and V-ROI Pooling are used for feature extraction, respectively. The combined experimental results are shown in Table 5.

It can be seen from Table 5 that the network using G-VFE + V-ROI Pooling performs best under the Mod.-mAP indicator. The network that only uses G-VFE + ROI Pooling generates a higher-quality feature map, so its performance is 1.47 higher than the original VFE + ROI Pooling mAP. In summary, it can be seen that the G-VFE and V-ROI Pooling modules proposed in this article can effectively improve the overall accuracy of the network model by adjusting the performance of the corresponding parts, and fully demonstrate its rationality. However, performing GMM clustering before feature extraction will

Table 1 Comparison of different network models on KITTI test set

	Easy	Moderate	Hard	Easy	Moderate	Hard
	3D detection-AP70			Bev detection-AP70		
PointRCNN	88.68	78.50	77.66	90.08	87.70	86.29
PointPillars	86.28	77.11	74.01	89.85	87.38	85.11
DV-pointpillars	86.74	77.24	74.61	89.81	87.24	85.51
Second	87.03	77.42	74.76	89.62	87.11	85.57
Part-A2	88.59	78.68	76.93	90.09	88.15	86.09
STD	88.96	78.59	76.96	**90.27**	**88.16**	85.35
3DSSD	88.57	78.59	77.56	90.09	87.91	86.66
GVnet(Ours)	**89.26**	**79.14**	**78.48**	90.07	87.87	**87.20**
Improvement	+ 0.30	+ 0.46	+ 0.82	− 0.20	− 0.29	+ 0.54
	3D detection-AP50			Bev detection-AP50		
PointRCNN	89.58	80.34	77.87	93.23	89.07	86.83
PointPillars	90.70	89.97	89.21	90.72	90.06	89.41
DV-pointpillars	90.66	89.69	89.01	90.67	89.77	89.23
Second	91.99	82.86	80.37	92.99	90.14	88.34
Part-A2	90.57	89.71	89.05	90.58	89.83	89.27
STD	**96.27**	90.06	89.16	95.84	90.11	89.29
3DSSD	95.75	89.95	**89.46**	95.79	90.00	**89.58**
GVnet(Ours)	95.79	**90.07**	89.07	**96.33**	**90.43**	89.23
Improvement	− 0.48	+ 0.01	− 0.39	+ 0.49	+ 0.29	− 0.35

Bold represents the best performance

Table 2 Comparison on KITTI val

Method	Inference time	mAP
3DSSD	**0.2119**	78.81
Second	0.2121	77.63
PointRCNN	0.4755	78.98
STD	0.4904	79.10
GVnet (Ours)	0.3961	**79.69**

Bold represents the best performance

cause additional computational burden. This shortcoming has been reflected in the inference time, and a high-performance GPU is required to run this method.

4 Conclusion

This paper proposes a two-stage 3d detection network. The original Voxel Feature Encoder is optimized in stage one. The specific method is to solve the Gaussian distribution

Table 3 Test on nuScenes dataset

Method	Car	Ped	Bus	Truck	Trailer	Moto	Bicycle	Barrier	mAP
Point Pillars	70.58	58.34	37.40	26.67	21.31	18.45	**2.06**	31.59	33.30
SECOND	74.65	58.98	33.28	21.83	13.16	17.80	0.37	**32.04**	31.51
GVnet(Ours)	**76.20**	**59.16**	**42.33**	**29.74**	**22.08**	**20.66**	0.83	31.98	**35.37**
Improvement	+ 1.55	+ 0.18	+ 4.93	+ 3.07	+ 0.77	+ 2.21	− 1.23	− 0.06	+ 2.07

Bold represents the best performance

Table 4 Test on Waymo dataset

Method	3D mAP(IoU = 0.7)				BEV mAP(IoU = 0.7)			
	Overall	0–30	30–50	30–50	Overall	0–30	30–50	50-Inf
MVF	60.84	84.17	62.46	37.66	78.40	**91.59**	78.13	58.57
Point Pillars	55.58	79.93	51.70	30.47	73.51	91.96	73.38	54.75
GVnet(Ours)	**64.62**	**88.01**	**63.91**	**39.18**	**78.85**	91.17	**78.36**	**59.81**
Improvement	+ 3.78	+ 3.84	+ 1.45	+ 1.52	+ 0.45	− 0.37	+ 0.23	+ 1.24

Bold represents the best performance

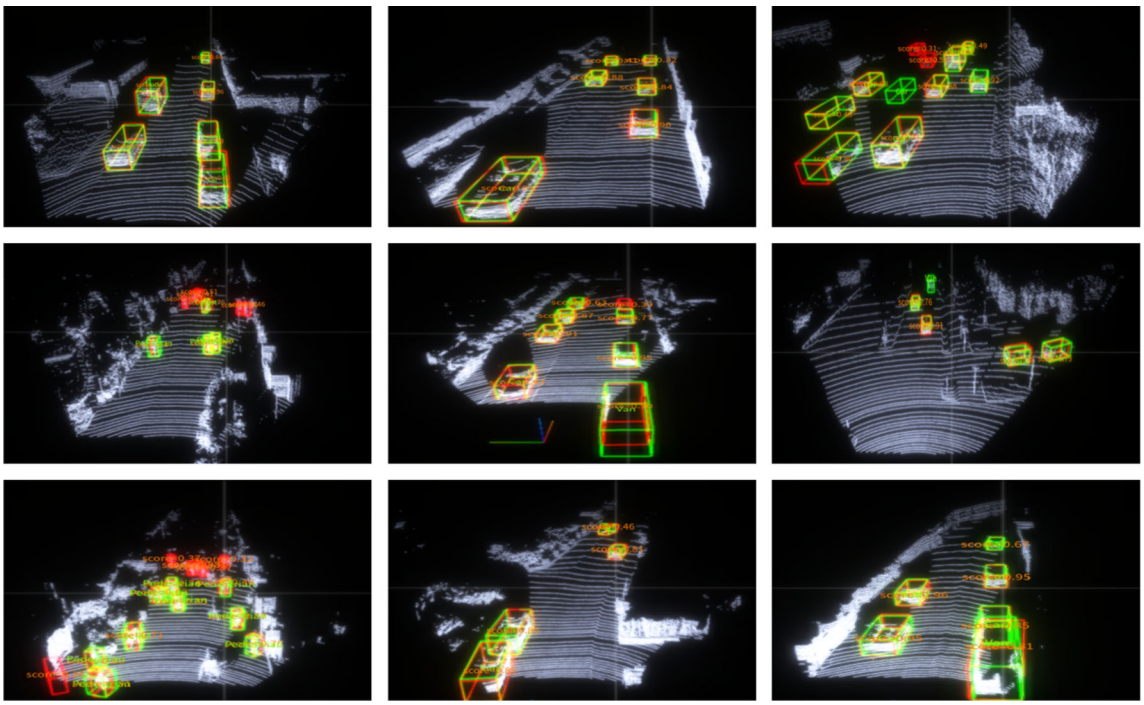

Fig. 7 Visual performance of GVnet on KITTI dataset

Table 5 Comparison of experimental effects of different components of the network model

G-VFE	VFE	ROI Pooling	V-ROI Pooling	Mod. mAP
√		√		77.60
	√	√		76.13
	√		√	–
√			√	**79.14**

Bold represents the best performance

that the raw point conforms to, and then use the Gaussian Mixture Model for voxel feature encoding, and perform 3D CNN to obtain high-quality feature maps. In the ROI Pooling of stage two, the feature is sampled based on the raw point in voxel to control the change of the receptive field and enhance the feature. This approach is designed to focus on key points and ignore unimportant point clouds. And then send the output feature to the fully connection layer to complete the classification and regression of the object. Finally, the proposed method was compared with other methods on the dataset, and the ablation experiment was carried out to demonstrate the rationality of GVnet proposed in this article. However, the number of categories of GMM clustering needs to be manually set in advance, which leads to the failure of end-to-end training of the entire network model, which reduces the generalization ability of the model. For different data sets, it is necessary to set the number of different categories in advance. Therefore, using GMM as a module of the network and adding it to the training process requires further research and development.

Acknowledgements The authors would like to thanks the National Natural Science Foundation of China (51974229)for their support in this research.

Declarations

Conflict of interests The author(s) declared no potential conflicts of interest with respect to the research, author- ship, and/or publication of this article.

References

1. Chen X, Ma H, Wan J, Li B, Xia T (2017) Multi-view 3d object detection network for autonomous driving. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1907–1915
2. Ku J, Mozifian M, Lee J, Harakeh A, Waslander SL (2018) Joint 3d proposal generation and object detection from view aggregation. In: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 1–8
3. Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2117–2125
4. Liang M, Yang B, Wang S, Urtasun R (2018) Deep continuous fusion for multi-sensor 3d object detection. In: Proceedings of the European conference on computer vision (ECCV), pp 641–656
5. Qi CR, Liu W, Wu C, Su H, Guibas LJ (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
6. Liang M, Yang B, Chen Y, Hu R, Urtasun R (2019) Multi-task multi-sensor fusion for 3d object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7345–7353
7. Xu D, Anguelov D, Jain A (2018) Pointfusion: deep sensor fusion for 3d bounding box estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 244–253
8. Wang C, Xu D, Zhu Y, Martín-Martín R, Lu C, Fei-Fei L, Savarese S (2019). Densefusion: 6d object pose estimation by iterative dense fusion. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 3343–3352
9. Vora S, Lang AH, Helou B, Beijbom O (2020) Pointpainting: sequential fusion for 3d object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4604–4612
10. Zhou Y, Sun P, Zhang Y, Anguelov D, Gao J, Ouyang T, Vasudevan V et al (2020) End-to-end multi-view fusion for 3d object detection in lidar point clouds. In: Conference on robot learning. PMLR, pp 923–932
11. 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 international conference on computer vision, pp 1951–1960
12. Lang AH, 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 conference on computer vision and pattern recognition, pp 12697–12705
13. Qi CR, Su H, Mo K, Guibas LJ (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
14. 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
15. Yang, B, Luo W, Urtasun R (2018) Pixor: real-time 3d object detection from point clouds. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7652–7660
16. Ali W, Abdelkarim S, Zidan M, Zahran M, El Sallab A (2018) Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud. In: Proceedings of the European conference on computer vision (ECCV)
17. Shi S, Wang Z, Shi J, Wang X, Li H (2020) From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. IEEE Trans Pattern Anal Mach Intell
18. Kuang H, Wang B, An J, Zhang M, Zhang Z (2020) Voxel-FPN: multi-scale voxel feature aggregation for 3D object detection from LIDAR point clouds. Sensors 20(3):704
19. 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
20. Chen Y, Liu S, Shen X, Jia J (2019) Fast point r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 9775–9784
21. 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
22. Qi CR, Yi L, Su H, Guibas LJ (2017) Pointnet++: deep hierarchical feature learning on point sets in a metric space. Adv Neural Inf Process Syst 30:5099–5108
23. Ren S, He K, Girshick R, Sun J (2016) Faster r-cnn: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39(6):1137–1149
24. Rasmussen C (1999) The infinite Gaussian mixture model. Adv Neural Inf Process Syst 12:554–560
25. Yan Y, Mao Y, Li B (2018) Second: sparsely embedded convolutional detection. Sensors 18(10):3337
26. He K, Zhang X, Ren S, Sun J (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans Pattern Anal Mach Intell 37(9):1904–1916
27. Lin TY, 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
28. Girshick R (2015) Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448
29. 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. IEEE, pp 3354–3361
30. Caesar H, Bankiti V, Lang AH, Vora S, Liong VE, Xu Q, Beijbom O et al (2020) nuscenes: a multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 11621–11631
31. Sun P, Kretzschmar H, Dotiwalla X, Chouard A, Patnaik V, Tsui P, Anguelov D et al (2020) Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2446–2454
32. Everingham M, Van Gool L, Williams CK, Winn J, Zisserman A (2010) The pascal visual object classes (voc) challenge. Int J Comput Vision 88(2):303–338