APSIPA Transactions on Signal and Information Processing, 2023, 12, e104

Original Paper

An Edge Lidar-Based Detection Method in Intelligent Transportation System

Yung-Yao Chen[1], Hsin-Chun Lin[1], Hao-Wei Hwang[1], Kai-Lung Hua[2], Yu-Ling Hsu[3] and Sin-Ye Jhong[4*]

[1]*Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan*
[2]*Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan*
[3]*Department of Industrial Management, National Taiwan University of Science and Technology, Taipei, Taiwan*
[4]*Department of Engineering Science, National Cheng Kung University, Tainan, Taiwan*

ABSTRACT

Rapid advances have occurred in Internet of Things technologies. Among Internet of Things–related applications, Internet of Vehicles (IoV) is regarded as integral infrastructure for next-generation intelligent transportation systems. IoV requires vehicles to perceive their surroundings reliably. In particular, researchers have focused on LiDAR sensing because it is robust in extreme weather. However, IoV sensing data are transmitted between vehicles and the cloud, and LiDAR requires a large quantity of data; thus, communication for cloud computing might be challenging. To address this difficulty, a LiDAR-based detection method for an IoV edge node is proposed. Small-object detection through LiDAR sensing is difficult because of the sparsity of point clouds. Although some researchers have attempted to solve this problem by fusing raw point cloud details, existing approaches still reduce model efficiency and memory cost, which is unsuitable for IoV. To overcome the problem, this paper proposes a novel model that enhances three-dimensional (3D)

*Corresponding author: Sin-Ye Jhong, n98081034@ncku.edu.tw.

Received 30 March 2023; Revised 17 June 2023
ISSN 2048-7703; DOI 10.1561/116.00000118

structural information for preserving the details of voxel features while maintaining the high efficiency and low memory usage of voxel-based methods. Experimental results indicate that the proposed method outperforms state-of-the-art LiDAR-based 3D detection methods on the widely used KITTI dataset and achieves competitive performance for all classes.

Keywords: Edge computing, intelligent transportation system, internet of vehicles, LiDAR-based detection, voxel-based detectors.

1 Introduction

In smart cities, developing an Intelligent Transportation System (ITS) is a critical task that can improve the safety and efficiency of mobility. Internet of Vehicles (IoV), which is enabled by advanced sensing, communication, and networking technology, is commonly regarded as an integral technology for ITSs. As shown in Figure 1, IoV is a network comprising a cloud service and sensor-equipped vehicles; all data are constantly exchanged over the Internet. See [13]. Robust vehicle sensing is indispensable in IoV. Light detection and ranging (LiDAR) sensors have demonstrated promising sensing ability in many applications related to autonomous cars and Advanced Driver Assistance Systems (ADAS). See [21]. However, compared with RGB cameras, LiDAR systems produce a considerably larger quantity of data, which might reduce latency in IoV applications. To address this problem, a LiDAR-based detection method for edge nodes (i.e., vehicles) is proposed in this paper.

Methods for three-dimensional (3D) LiDAR object detection tasks are gradually maturing. Because of the precise 3D structural information provided by LiDAR sensors, their performance far surpasses that of other detection methods. Although powerful, LiDAR has some unique properties that must be handled explicitly. For example, the irregular distribution of the point cloud might require unique methods to avoid the permutation variance problem; that is, the model output might change because of an input permutation despite an identical input. To solve this problem, researchers have proposed several effective methods. On the basis of the representation of the point cloud, these methods can be divided into two categories: point- and voxel-based approaches. In point-based approaches, raw point cloud are usually directly manipulated using a symmetric function (e.g., max pooling) to solve the permutation variance problem. See the discussion in [33], [25], [3] and [35].

By contrast, in voxel-based approaches, the entire point cloud is divided into voxels to regularize the irregular distribution so that it can be input into a 3D convolution process. See [6], [34], [36] and [26]. However, both approaches

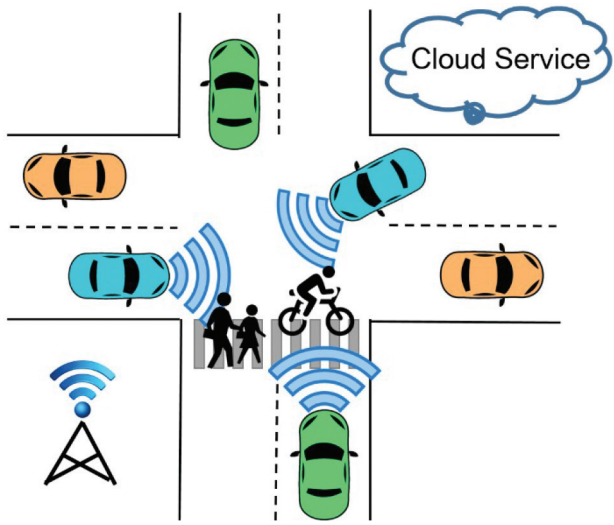

Figure 1: Application of the proposed method. In the era of IoV, the edge nodes of the network (i.e., each vehicle) should not only capture environmental sensing data but also process some preliminary data to reduce bandwidth and latency. To address this problem, a LiDAR-based detection method for edge nodes (i.e., vehicles) is presented. The developed model was modified to ensure that it has high efficiency.

have drawbacks and often exhibit poor performance in detecting cars or other small objects, respectively.

In real scenarios, cars and pedestrians must be considered equally to achieve traffic safety. A model that ignores either of them is inappropriate for an ITS. Some researchers have combined point- and voxel-based approaches into point–voxel (PV)-based methods to improve the detection of all these objects. In PV-based methods, voxel features are fused with raw point cloud features to regain the 3D structural information lost in the voxelization and convolution processes. For example, [24] proposed PV-RCNN that contains a pointwise feature extraction branch that uses the Voxel Set Abstraction (VSA) module to fuse pointwise and voxelwise features to generate a better representation to support the 3D voxel backbone. Although PV-based methods can improve performance across all classes, they also increase the computation cost of the model because of the inclusion of raw point cloud details. Thus, PV-based methods might not be the best approach to adopt. Therefore, as our research focuses on the edge computing mode, which requires relatively low computation costs and memory usage. One must determine whether the voxel representation is adequate for small-object detection or whether point cloud distribution details are essential. Consequently, in this paper, we propose a novel 3D voxel backbone that uses only voxel features to improve our model

and avoid the high loading caused by additional point cloud calculations. The proposed backbone not only uses a novel module to transform the features in the feature space but also learns the foreground voxel probability through an auxiliary supervision task.

Specifically, our model predicts each voxel's foreground probability to support the backbone and refinement stage. The proposed semantic-split feature transform (SSFT) module can calculate a pair of adjustment factors for each voxel feature from the corresponding predicted probability. These produced factors enhance the structural details in the feature space without directly interfering with the convolutional features. Compared with the vanilla 3D voxel backbone proposed by [6], our structure has better small-object detection performance because its additional transform module provides finer spatial information. Moreover, we constructed a new formula for the region of interest (RoI) pooling process to achieve a better representation of sampled features. To increase the effectiveness of the sampling features, we first use the distance to each grid point as a factor to avoid sampling the same point for different grid points. Furthermore, we again use the predicted probability map to improve the sampling candidate quality by providing foreground information to the refinement module. In this research, our proposed methods can be summarized by the following contributions. First, our proposed LiDAR-based 3D object detection architecture uses an edge deployment strategy to reduce network latency and ensure that the detection system can respond promptly to any given situation. Second, we propose the use of voxel features to improve small-object detection, even in the absence of raw point cloud support. Third, we designed a novel SSFT module with an additional auxiliary task to enhance the fine-grained spatial information of the 3D convolutional features. Fourth, we designed a novel RoI pooling method to improve the sampled candidate quality to improve the feature representation.

The remainder of this article is structured as follows: Section 2 presents a discussion on the related works. Section 3 describes our model's entire structure, implementation details, and formulas. Section 4 presents the experimental results, including the results of an ablation study and a comparison of the proposed model with other state-of-the-art (SOTA) models. Section 5 provides the conclusions of our study and potential directions for future research.

2 Related Works

This section describes the techniques used in 3D LiDAR object detectors. Because of the sparse and irregular distribution of LiDAR point cloud data, specific operations are required to process these data effectively. These techniques can be broadly classified into three categories according to the selected data representation: point-, voxel-, and PV-based methods.

2.1 Point-Based Methods

In point-based methods, raw point cloud features, which are distributed throughout real-world coordinates, are directly used as model input. However, these methods can result in the permutation variance problem. [19] proposed PointNet which is a novel model that addresses this problem by using the max-pooling function as a symmetry function. Nevertheless, the numerous points of LiDAR point clouds can result in high computational burden. To overcome this problem, [20] introduced PointNet++, which has a hierarchical point-set feature learning module that uses a farthest point sampling (FPS) algorithm for downsampling. The FPS algorithm considers the distance between two points and selects the farthest points from the current point to create a more uniformly distributed sampled subset. Furthermore, [25] proposed PointRCNN which adopts the PointNet++ model as its backbone to construct a two-stage detector. And due to the vanilla FPS algorithm only considers distance. [33] introduced 3DSSD that uses a novel Feature-FPS technique in which semantic feature distance is also considered for preserving helpful information (e.g., foreground features) and discarding redundant information. Moreover, [3] and [35] proposed the SASA model with semantics-guided FPS and the IA-SSD model with Ctr-aware sampling, respectively; both methods had excellent performance.

2.2 Voxel-Based Methods

In contrast to point-based methods, voxel-based methods involve dividing the space into predefined voxels for model input. This regular voxel representation enables the exploitation of common convolution processes. For example, [37] proposed VoxelNet with a voxel feature encoding module was proposed to extract features for each voxel and feed them into convolution layers to predict 3D bounding boxes. Although the convolution process improves model performance, it requires traversing the entire 3D space; thus, this process is extremely inefficient. However, because LiDAR point clouds are sparse, many voxels are meaningless or redundant. In the SECOND model, which was proposed in [32], the convolutional layers are replaced with a SparseCNN module to overcome the aforementioned problem, thereby reducing computational costs. SparseCNN only considers the necessary input, thereby reducing memory costs and improving model efficiency. Since the introduction of SparseCNN, many studies have used this module as a part of their model backbone to achieve better performance. For example, Part-A^2, which proposed by [26], based on the SparseCNN backbone while focus on distinguishing the intraobject part location of an RoI. [6] introduced Voxel-RCNN which is a simple but effective voxel-based model that indicates the importance of 3D structural information for 3D LiDAR object detection. [31] proposed a new

concept that calculates the occlude shape probability to support the 3D voxel backbone for better performance. [4] further introduced a novel convolution technique including an attention mechanism to enhance the voxel features according to their importance.

2.3 Point-Voint-Based Methods

Although point- and voxel-based methods achieve excellent performance, they have certain limitations. Point-based methods preserve the point cloud distribution in real coordinates and typically perform well for small-object detection (e.g., pedestrian and cyclist detection) but might struggle with car detection. By contrast, voxel-based methods can implement the convolution process but might also lose object details during calculations. Therefore, some researchers have attempted to fuse point- and voxel-based methods. In PV-based methods, point and voxel representations are used to obtain excellent results for all classes. For example, in [24], PV-RCNN with a VSA module was proposed for sampling voxel features in each layer with key points to enhance the RoI feature representation. [23] proposed CT3D that adopts a voxel backbone to produce RoI proposals and then samples raw point cloud features for further refinement. [1] proposed a deformable self-attention module that aims to get a representative subset that aggregates global context. [12] further introduce PDV which uses density information and voxel point centroids to enhance the fine-grained details of voxel features.

2.4 RoI Grid Pooling Strategies

The RoI pooling module is essential for a two-stage detector when processing in the refinement stage. However, implementing conventional RoI pooling methods in three dimensions is challenging because of the unusual data distribution of LiDAR point clouds. To overcome this problem, [25] uses the point-cloud-region pooling method, in which all the features within the RoI are pooled for further refinement. [26] discovered that this method might result in the ambiguous boxes problem and proposed Part-A^2, which includes RoI-aware pooling, to address it. Furthermore, [24] uses predefined grid points to sample RoI features, thereby increasing the sampling range for encoding more useful features. [18] proposed Pyramid-RCNN which adopts the attention module in the RoI pooling strategy, thereby enabling adaptive point features sampling. More, instead of using predefined grid points, [23] introduced CT3D that randomly samples point features, which is an effective strategy if the sample size is sufficiently large.

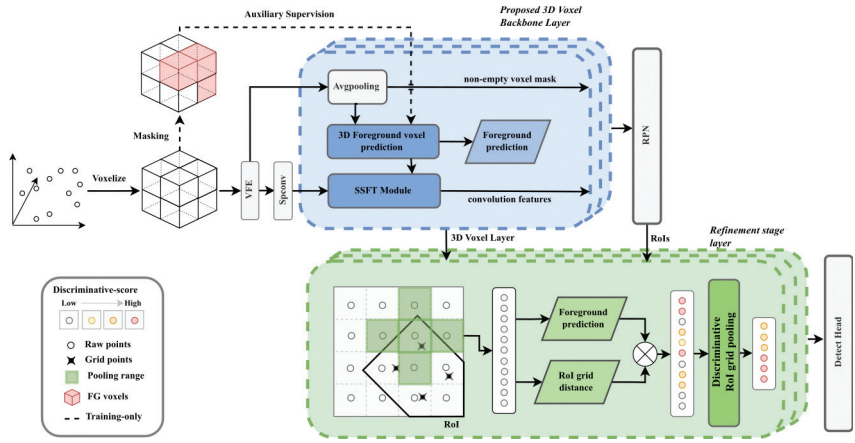

Figure 2: An overview of our model structure. The input point cloud will first voxelized and go through three of ours 3D backbone layer to produce RoIs. Then, the convolution features and the predicted foreground score will be used in further refinement stage. In refinement stage, we will utilize the score and calculate the distance to the corresponding grid points as the sample criteria.

3 Methodology

To improve small-object detection while avoiding excessive computational load, we decided to use voxel representations throughout the entire developed model. As shown in Figure 2, the point cloud is first voxelized and encoded to obtain voxel features which are fed into the proposed backbone. In each layer of the backbone, an auxiliary supervision branch is added to predict the foreground's probability, and a transformation function is used to manipulate the convolutional features in the feature space. The transformed features are used in the proposal network and refinement stage. In the refinement stage, each refinement layer only uses the features from the corresponding backbone layer to extract the RoI features separately. The foreground probability prediction and point-to-grid distance are used as evaluation factors to optimize the sample quality to produce the refined features. Finally, the sampled features from each refinement layer are concatenated and further fed into the prediction head of the model for the final prediction.

3.1 3D Foreground Voxel Supervision

Structural information is usually vital for a 3D LiDAR object detection task. As mentioned in [6], 3D structural information strongly influences the final detection performance of the VoxelRCNN model. In [6], the addition of connections between the backbone layers and the refinement stage resulted

in considerable improvement in the model performance. Furthermore, some researchers have attempted to use not only structural information but also the foreground distribution to improve model performance. Most of these researchers have used an additional auxiliary supervision task to predict the foreground probability for further model implementation. For example, [34] proposed SegVoxelNet which uses a supervision branch to predict the foreground probability of the bird's-eye-view feature map to support the model backbone.

However, to the best of our knowledge, most previous studies have only simply implemented the auxiliary branch on the two-dimensional (2D) feature map or in the final layer of the backbone. Although these approaches can improve the model performance, precise structural information might be lost during several convolution processes, thereby reducing the model effectiveness. To obtain precise 3D structural information, we attempt to exploit the auxiliary supervision concept in each backbone layer while maintaining the 3D voxel representation at each scale. Moreover, instead of repeatedly checking whether the voxel is a foreground label, training labels are generated by masking the voxels from the beginning and calculating them throughout the backbone for achieving higher efficiency. We follow the similar label definition in [34], in which voxels not only overlap with ground truth objects but also contain raw points (i.e., nonempty voxels) that can be determined as foreground labels.

Furthermore, to preserve a better feature granularity, we utilize the pooling module for generating training labels instead of the convolutional process. Specifically, the average-pooling module is implemented repeatedly to produce the foreground mask in each layer. It is worth noting that we also evaluate the performance of the max-pooling module, and the relevant results are provided in Section 4.3. We attribute the poor performance of the max-pooling module to the fact that it does not consider the differences within each foreground object. In contrast, average pooling can preserve and distinguish these differences and thus might produce results that are more similar to the real distribution. Moreover, by implementing the pooling module, the predicted foreground probability map can match the same size of the original convolutional features, and thereby can further support the model's backbone. Finally, our loss function for the auxiliary supervision task is expressed as function (1).

$$L_{fg} = \frac{1}{N_{ne}}[\beta_{C_i} \cdot L_{cls}(P_i, C_i)], \quad \forall i = 1, \ldots, N_{ne}, \tag{1}$$

where N_{ne} denotes the number of nonempty voxels; P_i and C_i represent the predicted probability and corresponding ground truth, respectively; β_{C_i} is the balance factor for each class; and L_{cls} indicates the focal loss function for classification.

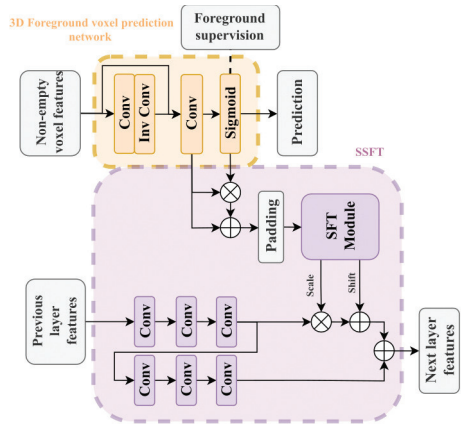

Figure 3: The structure of the proposed SSFT module and the foreground prediction network. The SSFT module (lower part) inputs the prediction from the foreground prediction network (upper part) to the SFT module to produce the scale and shift factors for guiding voxels of the main convolution feature map.

3.2 Semantic-Split Feature Transform

In a 3D object detection task, the performance for detecting small objects might strongly depend on the feature granularity. For example, point-based methods in which raw point features (which provide information on the point cloud distribution with real-world coordinates) are directly used usually achieve high detection performance for pedestrians and cyclists. However, voxel-based methods involve dividing the space into several voxels; this process might cause a quantitative error that harms the feature granularity. Moreover, the 3D voxel backbone comprises several convolution blocks; thus, the aforementioned error might be produced repeatedly and considerably affect small-object detection.

Therefore, a technique that can enhance spatial details is desired to eliminate the aforementioned drawback. The spatial feature transform (SFT), which has been used in image super-resolution tasks, aims to recover a low-resolution image by calculating a pair of scale and shift factors from the high-resolution condition map. See [30], [9] and [29]. The produced scale and shift factors are then applied to the original low-resolution image in the feature space. To take advantage of the SFT, the specific condition map for our task must be determined. Because the condition map requires high-resolution spatial information, a straightforward method is implementing the feature map from the previous layer. However, we realize that using the foreground probability prediction described in Section 3.1 is more suitable than the aforementioned method as the prediction map can describe more details of the distribution about the foreground voxel features.

As displayed in the upper part of Figure 3, the output of the prediction network is concatenated into our 3D backbone for each layer. Our prediction method is not directly implemented on the convolution feature map; two processes are used before its implementation. First, the foreground prediction is padded to match the feature map size to produce the SFT factors for all voxels before multiply with the convolution features. Without this process, the SFT module might only produce factors for foreground voxels, which might reduce feature consistency. Second, the padding predictions are used as the weighted factors to enhance the original foreground voxels. By ignoring the original features, the SFT module will eliminate all the effects from background features. [18] claimed that those background features can further improve model performance. Thus, we designed function (2) to produce the scale and shift factors.

$$\alpha_i, \gamma_i = M(D_i + P_i \cdot D_i), \quad i = 0, 1, 2, 3, \tag{2}$$

where D_i indicates the condition map for each layer after several convolution processes; M is the combination of mapping and padding functions; α and γ are the predicted scale and shift factors, respectively.

We determined that the goal of the SFT module is to enhance the spatial features instead of the entire original feature map. To optimize the effectiveness of this module, spatial and semantic features should be extracted separately. Several researchers have used multiple branches to extract features within convolution blocks. For example, [36] found that implementing different processes in separate feature branches for the spatial and semantic features can improve the detection performance of the model. We believe that this finding is attributable to the different usages of the aforementioned two types of features. Spatial features require finer structural information than do semantic features, whereas semantic features might require a larger receptive field that can contain more useful information. Thus, in our model, the SFT module is implemented on the extracted spatial features alone. The design of the semantic branch of our model is same as that in the [36] so that a large receptive field is ensured. We evaluated the effect of the aforementioned two-branch concept through an ablation study, which is described in Section 4.3. Finally, we designed our backbone according to the two-branch concept displayed in the lower part of Figure 3. Also, we design the whole SSFT module as the following function (3).

$$F_{i+1} = F_i^{spa} + F_i^{sem} = (F_i \cdot \alpha + \gamma) + Conv(F_i), \quad i = 0, 1, 2, 3, \tag{3}$$

where F_i indicates the input convolutional features from the previous layer; $Conv$ denotes several convolutional blocks; and F_i^{spa} and F_i^{sem} are the spatial and semantic features, respectively.

3.3 Discriminative RoI Grid Pooling

The refinement stage in two-stage detectors is critical to the final model performance. To implement the detection head with different proposal sizes, techniques such as RoI pooling and RoI Align are used. See [8], [22], [5] and [10]. For 3D cases, we may consider the voxel representation can enables the easy extension of conventional RoI pooling techniques from pixels to voxels. However, the difference between LiDAR point clouds and 2D images is not that simple. For example, because of the sparsity of the point cloud, empty voxels are generated within objects, which might considerably reduce the quality of the extracted RoI information for further refinement. Therefore, conventional 2D RoI pooling techniques might be unsuitable for exploitation.

Fortunately, researchers have attempted to solve the aforementioned challenge. For example, in [24], RoI grid pooling was developed to improve the refinement effectiveness substantially. By using a set of predefined grid points, which are spread over the entire RoI, they can sample features that are inside or near the RoI bounding boxes. However, most recent methods only focus on improving the sampled coverage while ignoring some potential problems. See the discussion in [6], [24] and [12]. We identified two problems of concern. First, because the number of sampled candidates is fixed, the sampling priority order is crucial. The final performance might highly depend on the quality of the sampled candidates. Second, in contrast to the conventional RoI pooling method, in which the RoI is divided into blocks without overlap, the features sampled from grid points within a specific range might have overlapping regions. These overlapping regions might cause repeat sampling, which reduces the diversity of the sampled information. Thus, a method that can be used to evaluate feature quality and filter out high-quality subsets is required.

Fortunately, the downsampling process used in point-based methods might meet our requirements. These methods usually aim to reduce the total number of raw point clouds while maintaining the entire point cloud's quality. They use a sampling evaluation formula to consider not only the semantic scores but also the distance between points to construct several meaningful clusters. Inspired by them, an evaluation process is adopted within the RoI pooling module of our model. Also, to overcome the aforementioned two problems, the effects of the foreground score and point-to-grid distance factor are considered in this module.

With regard to the first problem, high-quality candidates are usually also foreground features because they might carry considerable object information. Thus, foreground probability prediction is performed again to increase the probability of foreground-like features being sampled. Moreover, to overcome the overlapping problem, an attempt is made to maximize the differences between the sampled subsets of all grid points. Intuitively, the shorter the distance from a point to a grid point, the longer is the distance from the

point to other points. This property is exploited to construct our sampling formula 5. Specifically, in this formula, the ratio of the point-to-grid distance to the predefined sampled range is considered as the distance factor d_f. Also, because the shorter distance to grid points the higher probability that need to be sampled. To meet this requirement, the distance factor should be modified as $1 - d_f$. Therefore, this modification will reduces the probability of sampling features near the sampled boundary. Finally, the complete sampling formula is as function (4).

$$S' = S^{\mu}(1 - d_f), \quad d_f = \frac{d_{grid}}{R_{grid}}, \tag{4}$$

where S is the predicted probability score; μ is a hyper-parameter for balancing the evaluation factor, which is the same as that used in [3]; d_{grid} and R_{grid} are the point-to-grid distance and the pre-defined sample range, respectively.

4 Experiments

4.1 Implement Details

4.1.1 Dataset

We evaluated our model on the famous KITTI dataset that introduced by [7], which contains 7481 scans of training scenes and 7518 scans of testing scenes. Our results were evaluated on a validation dataset and an online testing server. We split the training scenes into two parts: 3712 scenes comprised the training set, and 3769 scenes comprised the validation set. In the online testing, we randomly selected 80% of the training dataset for training, and the remaining data were used as the validation dataset

4.1.2 Network Architecture

The proposed 3D voxel backbone is based on those used in [6] and [24]. Within each backbone layer, a SparseCNN module with a stride of 2 is used to downsample the feature map, and this module is followed by a Submanifold CNN with a stride of 1. To match the foreground prediction map with the convolutional feature map, the pooling module is implemented with a $2 \times 2 \times 2$ kernel. The hyperparameter μ used in the function 5 is set as 1. The pooling range of our module is the same as that in [6].

4.1.3 Training and Inference Details

The proposed model was end-to-end trained with the Adam optimizer, which proposed in [14], for approximately 80 epochs. During training, the batch size

was set as 2, and the initialized learning rate was set as 0.01. For the additional auxiliary loss that present in function 1, we set the balance and focal factors to 0.25 and 2, respectively, to construct the focal loss that introduced in [16]. Finally, we implemented the same data augmentation strategies used in [6, 24, 33] to strengthen our model.

4.2 Results on KITTI Dataset

We evaluated our model on the KITTI dataset. The model performance was evaluated in terms of mean average precision (mAP) and an intersection-over-union threshold, which was 0.7, 0.5, and 0.5 for the three classes of car, pedestrian, and cyclist, respectively. For the validation set and online testing server, all performance indicators were calculated for 40 recall positions.

We compared the performance of our model with that of three powerful PV-based methods in Tables 1 and 2. Our model had higher performance at the moderate level (i.e., the main criteria for ranking in the KITTI dataset) for all three classes, especially the pedestrian class. Specifically, the pedestrian detection performance of the proposed model surpassed that of the other

Table 1: Performance comparison with state-of-the-art models on the KITTI *val* set for pedestrian 3D detection in all three difficulties and also the comparison of the GPU memory usage (MiB) and the inference time (FPS). All the 3D detection results are evaluated with average precision and calculated by 40 recall positions.

	3D Pedestrian				
Models	Easy	Mod.	Hard	Mem.	FPS
PV-RCNN, [24]	62.72	54.51	49.87	8243	17
CT3D, [23]	61.04	55.55	51.05	6485	19
PDV, [12]	66.90	60.80	55.85	7302	21
Ours	**68.53**	**61.01**	**56.41**	**5669**	**27**

Note: Our model's results are shown in bold, and the best results are underlined. Also, the above results are reproduced by the publicly release model ([28]).

Table 2: Performance comparison on the KITTI *val* set for car and cyclist 3D detection in the moderate difficulty. All the 3D detection results are evaluated with average precision and calculated by 40 recall positions.

Models	3D Car Mod.	3D Cyclist Mod.
PV-RCNN, [24]	84.33	70.40
CT3D, [23]	84.99	71.89
PDV, [12]	85.29	74.23
Ours	**85.37**	**74.27**

Note: Our model's results are shown in bold, and the best results are underlined. Also, the above results are reproduced by the publicly release model ([28]).

models for all difficulty levels. More, we visualize both the detections from our model and the voxel-based method (i.e. Voxel-RCNN) to further verify our model's performance on those difficult cases. In Figure 4, we show the highly occluded objects which are missed by voxel-based methods but successfully detected by ours. In Figure 5, we demonstrate the performance of our model for detecting crowds which still successfully detects each of them while voxel-based methods miss some of them.

Furthermore, we adopt the IPC which is equipped with an Intel XEON E-2288G @ 3.70 Hz computer processing unit and a CVB-CD1024 industrial solid-state drive; thus, it had a minimum write speed of 510 MB/s to minimize transmission latency and data loss. Also, we test the Pytorch model on one RTX3090 which shows the details in the right part of Table 1, compared to the other strong PV-based methods, our model exhibited considerably lower computational cost and memory usage. As they usually include additional point cloud calculations, point-voxel features interaction, or even the strong but heavy transformer module. Our model only utilizes existing voxel features without the additional raw point support which relieves nearly half of the memory cost from PV-RCNN. Also, as we only utilize a simple convolution structure to enhance spatial features, we can maintain the high efficiency of

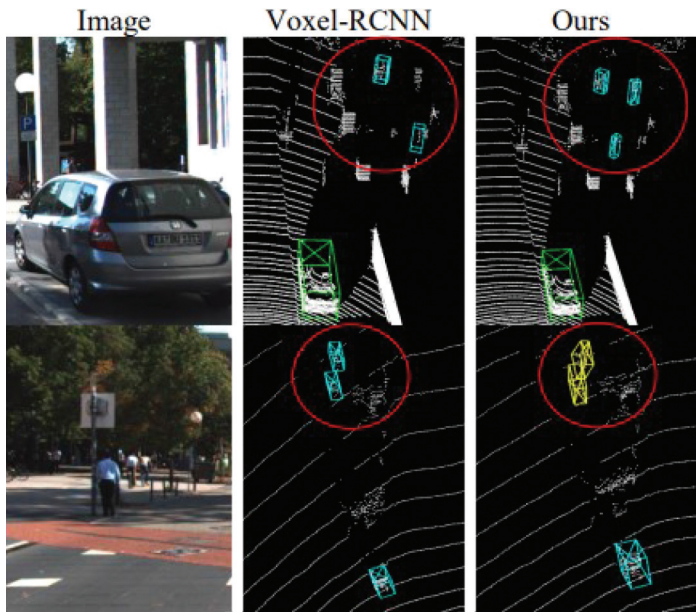

Figure 4: The visualization of highly occluded objects detections on KITTI *val* set. Note that the green, yellow and cyan boxes indicates the class of car, cyclist and pesdestrian, respectively. Also, the red circle denotes the main differences amoing the models' detection.

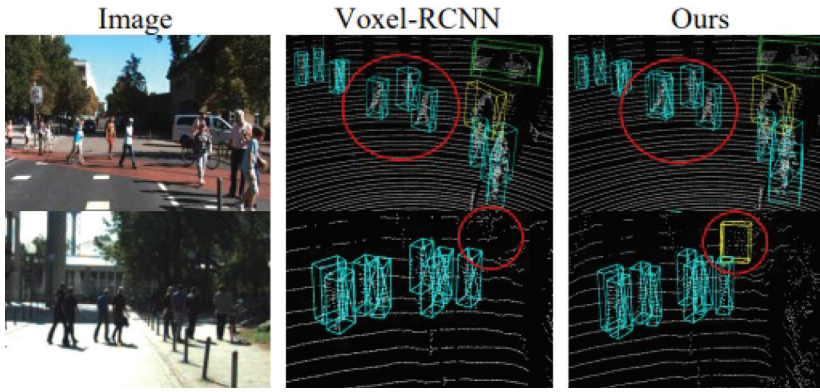

Figure 5: The visualization of dense crowd detections on KITTI *val* set. Note that the green, yellow and cyan boxes indicates the class of car, cyclist and pesdestrian, respectively. Also, the red circle denotes the main differences amoing the models' detection.

voxel-based methods which is usually the PV-based methods' weakness. These two advantages make our model more suitable for edge deployment.

The results obtained with the different models on the KITTI official testing server are listed in Table 3. In Table 3, the proposed model performs well for all the classes and achieves the best 3D mAP performance among the tested models no matter which type of method. We can see that although voxel-based methods are usually worse on small object detection, we prove that voxel representation is still adequate for small-object detection as our performance on pedestrian detection surpasses several point-based and PV-based methods. On the other hand, although raw point cloud distribution details are not essential in our case, we still realize that utilizing finer voxel features to mimic the point cloud distribution details is still helpful for small-object detection.

4.3 Ablation Studies

Table 4 presents the results of an ablation study on the proposed model. In experiment 1, we determined the performance of the baseline model, which uses the same model structure in [6], to verify our improvements over voxel-based detectors. However, because this model was originally designed for car detection only, we had to modify several hyperparameters to enable multiclass detection in the subsequent experiments. In experiment 2, the auxiliary supervision network was conducted to provide foreground information to the backbone. The foreground prediction was then directly added to the convolutional features voxel by voxel in each backbone layer. However, because this connection is simple, it only resulted in a small increase in performance. This result might

Table 3: Performance comparison on the KITTI *test* set for all classes 3D detection in the moderate difficulty and also the comparison of the 3D mean average precision (mAP). All the 3D detection results are evaluated with average precision and calculated by 40 recall positions.

	3D Mod.			3D
Models	Car	Ped.	Cyc.	mAP
PointRCNN, [25]	75.64	39.37	58.82	60.33
Point-GNN, [27]	79.47	43.77	63.48	63.90
3DSSD, [33]	79.57	44.27	64.10	65.01
IA-SSD, [35]	80.32	41.03	66.25	64.39
SECOND, [32]	75.96	35.52	60.82	59.29
PointPillars, [15]	74.31	41.92	58.65	60.65
TANet, [17]	75.94	44.34	59.44	61.71
Part-A^2, [26]	78.49	43.35	63.52	63.99
SVGA-Net, [11]	80.47	40.39	62.28	62.92
PV-RCNN, [24]	81.45	43.29	63.71	64.74
SA-Det3D, [1]	81.46	40.89	68.54	65.04
PDV, [12]	81.86	40.56	67.81	65.31
Ours	**82.00**	**42.25**	**66.46**	**65.64**

Note: Our model's results are shown in bold, and the best results are underlined.

be attributable to the corruption of the different receptive fields between the predicted foreground probability and the convolutional features.

In experiment 3, we replaced the simple addition process in the second experiment with the proposed SSFT module. The feature transformation and additional convolution processes were clearly beneficial; the model performed well in not only car detection but also small-object detection. In particular, the model performance for cyclist detection in experiment 3 was approximately 1.31% higher than that in experiment 2. Finally, experiment 4 demonstrated the effectiveness of the proposed RoI pooling module. In contrast to the original RoI grid pooling method proposed by [24], the point-to-grid distance and predicted foreground probability are considered in the proposed RoI pooling method; thus, the final model performance was 1.15% higher than the performance of the original baseline model in pedestrian detection.

Although Table 4 clearly demonstrates the effectiveness of the SSFT module, our assumptions regarding the usage of the SFT module must still be validated. In Section 3.2, an improved performance is assumed to be achieved when SFT factors are only implemented on the spatial branch. Therefore, we designed two experiments for further validation. In the first experiment, the unmodified SFT that used in [30] module was used; this module implements its produced factors on the entire feature map. In the second experiment, the proposed SSFT module was adopted. As presented in Table 5, the SSFT module

Table 4: Ablation studies on the *mod* level of the KITTI *val* set for all classes 3D detection and all the results are evaluated with average precision and calculated by 40 recall positions.

Experiments	FG	SSFT	DP	Car	Pedestrian	Cyclist
(1)				84.87	59.87	73.23
(2)	V			85.08	59.91	73.02
(3)	V	V		85.34	60.30	74.33
(4)	V	V	V	85.37	61.01	74.27

Note: FG: foreground prediction network; SSFT: semantic-split feature transform; DP:discriminative RoI grid pooling.

Table 5: Ablation studies of the effectiveness of the semantic split concept for all classes 3D detection in the moderate difficulty and all the results are evaluated with average precision and calculated by 40 recall positions.

	3D Moderate		
Methods	Car	Pedestrian	Cyclist
w/o sematic split	85.28	59.06	73.01
with sematic split	85.34	60.30	74.33

Table 6: Ablation studies of the comparison of the pooling strategies for all classes 3D detection in the moderate difficulty and their average. All the results are evaluated with average precision and calculated by 40 recall positions.

	3D Moderate			
Methods	Car	Pedestrian	Cyclist	Average
Max-pooling	85.30	57.31	73.52	72.05
Avg-pooling	85.34	60.30	74.33	73.32

outperformed the SFT module, especially for cyclist detection (improvement of greater than 1%). Therefore, the ablation study successfully validated our previous assumption.

To improve the precision of the foreground probability prediction, feature granularity must be preserved during the pooling process. Thus, the selection of the pooling method might influence the final model performance. We designed a pair of experiments to compare the performance between the max-pooling and average-pooling processes, and the results are listed in Table 6. In the experiment, the performance achieved with max pooling for pedestrian and cyclist detection was substantially lower than that achieved with average pooling. This result might be attributable to the fact that the max-pooling process might discard some of the distribution information, which results in the loss of fine details. By contrast, the average-pooling process can preserve the feature distribution of high-granularity features, which contain fine-grained information.

We compared the proposed RoI pooling module with sampling formulas proposed in Table 7. Because this module achieves the greatest improvement for pedestrian detection, we focused on this class for detail comparison. Many studies have demonstrated the effectiveness of including semantic information in the sampling evaluation formula. See the discussion in [33], [3] and [35]. Experiment 1 revealed that performance reduced if foreground information alone was used. This result indicates that the distance factor plays a key role in the sampling evaluation formula and should not be removed. Therefore, on the basis of [35], we designed a formula including the foreground probability and the distance from a specific point to the RoI center. In experiment 2, although this adjustment increased the final model performance, the performance was still unsatisfactory, especially for the hard difficulty level. This result can be attributed to the fact that point cloud downsampling techniques are different from the method used by the proposed RoI pooling module. In downsampling techniques, the instance center, of which only one exists per cluster that ensure the sampling overlaps do not occur. By contrast, in RoI sampling, several grid points might exist inside an RoI, which results in multiple overlapping regions. Therefore, replacing the distance to the RoI center with the point-to-grid distance improved the detection performance for all difficulty levels.

Also, to further validate that the SSFT module works well in our model, we design an additional experiment that utilizes other feature enhancement methods to compare. We refer to the attention mechanisms backbone that has been proposed by [4], which has achieved a high ranking in KITTI, to be our comparison model. Table 8 shows the comparison between the adoption of different backbones. Our model gets better performance in all classes, especially the Cyclist class which successfully shows our advantage. Moreover, our model has a higher FPS than the heavy attention mechanisms backbone.

To further demonstrate the generalization ability of our model, we choose to additionally validate several experiments on the nuScenes dataset proposed by [2]. The experiments are shown in Table 9 where we compare all 10 classes of nuScenes dataset against our baseline model (i.e. Voxel-RCNN). As the results show, our model successfully improves most of the classes' performance from

Table 7: Ablation studies of the comparison of the RoI pooling methods for the pedestrian 3D detection in the all difficulties and all the results are evaluated with average precision and calculated by 40 recall positions.

	3D Pedestrian		
Methods	Easy	Moderate	Hard
Foreground-only	66.13	59.12	54.76
Foreground + RoI distance	66.86	60.08	54.65
Ours	68.53	61.01	56.41

Table 8: Ablation studies of the backbone structure adoption on the *mod* level of the KITTI *val* set and all the results are evaluated with average precision and calculated by 40 recall positions. Also, we provide FPS for the comparison of the model efficiency.

Methods	3D Moderate			FPS
	Car	Pedestrian	Cyclist	
Voxel-RCNN, [6]	84.87	59.87	73.23	30
FocalConv*, [4]	85.26	60.04	73.08	19
Ours	85.34	60.30	74.33	29

Note: The above results are reproduced by the publicly release model ([28]).
*: Note that we only adopt the attention backbone structure while the remaining part is the same as the baseline model.

Table 9: Performance comparison with state-of-the-art models on the Nuscenes *val* set for all 10 classes and all the 3D detection results are evaluated with average precision.

Models	Car	Truck	CV	Bus	Trailer	Barrier	Motor.	Cyc.	Ped.	Cone
Voxel-RCNN, [6]	68.88	28.03	6.04	54.34	17.99	25.37	22.39	7.21	59.55	25.39
Ours	69.25	28.95	6.83	54.31	18.04	26.57	23.75	8.46	59.56	25.08

Note: The above results are reproduced by the publicly release model ([28]). Note that CV stands for construction vehicle and Cone for traffic cone.

the voxel-based methods which indicates the high generalization ability of our model and proves the ability to handle more diverse classes in the IoV scenarios. More, performance for small object detections (e.g. motorcycle, bicycle) also achieve considerable improvements which show our model's advantage. Note that our results did not adopt the data augmentation proposed by [38] and trained for 20 epochs.

5 Conclusion

In this paper, we propose a novel voxel-based model that aims to improve small-object detection while reducing computational cost and memory usage. By using local voxel features and corresponding foreground predictions, the structural information within the feature space is enhanced. In addition, we designed an RoI pooling module to consider the foreground probability and point-to-grid distance to increase the sampling quality. The proposed model outperformed SOTA methods on the KITTI dataset for three classes (cyclist, pedestrian, and car). The results of this study indicate that our approach effectively addresses the object detection challenge for IoV systems in which the LiDAR data size can pose substantial communication challenges for cloud computing methods. The proposed method has better performance and lower computational and memory requirements than do existing approaches. In the future, researchers can explore additional lightweight modifications to

the proposed model and investigate its applicability to other datasets and scenarios.

References

[1] P. Bhattacharyya, C. Huang, and K. Czarnecki, "SA-Det3D: Self-attention based context-aware 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, 2021, 3022–31.

[2] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition,* 2020, 2020, 11621–31.

[3] C. Chen, Z. Chen, J. Zhang, and D. Tao, "SASA: Semantics-augmented set abstraction for point-based 3d object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence,* 2022, 36(1), 2022, 221–9.

[4] Y. Chen, Y. Li, X. Zhang, J. Sun, and J. Jia, "Focal sparse convolutional networks for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2022, 2022, 5428–37.

[5] Y.-Y. Chen, S.-Y. Jhong, G.-Y. Li, and P.-H. Chen, "Thermal-based pedestrian detection using faster r-cnn and region decomposition branch," in *Proceedings of the International Symposium on Intelligent Signal Processing and Communication Systems,* IEEE, 2019, 2019, 1–2.

[6] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel R-CNN : Towards high performance voxel-based 3d object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence,* 2021, 35(2), 2021, 1201–9.

[7] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* IEEE, 2012, 2012, 3354–61.

[8] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE/CVF International Conference on Computer Vision,* 2015, 2015, 1440–8.

[9] J. Gu, H. Lu, W. Zuo, and C. Dong, "Blind super-resolution with iterative kernel correction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2019, 2019, 1604–13.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE/CVF International Conference on Computer Vision,* 2017, 2017, 2961–9.

[11] Q. He, Z. Wang, H. Zeng, Y. Zeng, and Y. Liu, "SVGA-Net: Sparse voxel-graph attention network for 3d object detection from point clouds," in *Proceedings of the AAAI Conference on Artificial Intelligence,* 2022, 36(1), 2022, 870–8.

[12] J. S. Hu, T. Kuai, and S. L. Waslander, "Point density-aware voxels for lidar 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2022, 2022, 8469–78.

[13] S.-Y. Jhong, Y.-Y. Chen, C.-H. Hsia, S.-C. Lin, K.-H. Hsu, and C.-F. Lai, "Nighttime object detection system with lightweight deep network for internet of vehicles," *Journal of Real-Time Image Processing,* 18(4), 2021, 1141–55.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980, 2014*, 2014.

[15] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2019, 2019, 12697–705.

[16] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision,* 2017, 2017, 2980–8.

[17] Z. Liu, X. Zhao, T. Huang, R. Hu, Y. Zhou, and X. Bai, "TANet: Robust 3d object detection from point clouds with triple attention," in *Proceedings of the AAAI Conference on Artificial Intelligence,* 2020, 34(7), 2020, 11677–84.

[18] J. Mao, M. Niu, H. Bai, X. Liang, H. Xu, and C. Xu, "Pyramid R-CNN: Towards better performance and adaptability for 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision,* 2021, 2021, 2723–32.

[19] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2017, 2017, 652–60.

[20] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in Neural Information Processing Systems,* 2017, 2017, 5099–108.

[21] K. Qian, S. Zhu, X. Zhang, and L. E. Li, "Robust multimodal vehicle detection in foggy weather using complementary lidar and radar signals," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2021, 2021, 444–53.

[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2017, 39(6), 2017, 1137–49.

[23] H. Sheng, S. Cai, Y. Liu, B. Deng, J. Huang, X.-S. Hua, and M.-J. Zhao, "Improving 3d object detection with channel-wise transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision,* 2021, 2021, 2743–52.

[24] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "PV-RCNN: Point-voxel feature set abstraction for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2020, 2020, 10529–38.

[25] S. Shi, X. Wang, and H. Li, "PointRCNN: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2019, 2019, 770–9.

[26] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2020, 43(8), 2020, 2647–64.

[27] W. Shi and R. Rajkumar, "Point-GNN: Graph neural network for 3d object detection in a point cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2020, 2020, 1711–9.

[28] O. D. Team, "OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds," https://github.com/open-mmlab/OpenPCDet, 2020.

[29] X. Wang, Y. Li, H. Zhang, and Y. Shan, "Towards real-world blind face restoration with generative facial prior," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2021, 2021, 9168–78.

[30] X. Wang, K. Yu, C. Dong, and C. C. Loy, "Recovering realistic texture in image super-resolution by deep spatial feature transform," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2018, 2018, 606–15.

[31] Q. Xu, Y. Zhong, and U. Neumann, "Behind The curtain: Learning occluded shapes for 3d object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(3), 2022, 2893–901.

[32] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors,* 2018, 18(10), 2018, 3337.

[33] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-based 3d single stage object detector," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2020, 2020, 11040–8.

[34] H. Yi, S. Shi, M. Ding, J. Sun, K. Xu, H. Zhou, Z. Wang, S. Li, and G. Wang, "SegVoxelNet: Exploring semantic context and depth-aware features for 3d vehicle detection from point cloud," in *Proceedings of*

the IEEE International Conference on Robotics and Automation, IEEE, 2020, 2020, 2274–80.

[35] Y. Zhang, Q. Hu, G. Xu, Y. Ma, J. Wan, and Y. Guo, "Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2022, 2022, 18953–62.

[36] W. Zheng, W. Tang, S. Chen, L. Jiang, and C.-W. Fu, "CIA-SSD: Confident iou-aware single-stage object detector from point cloud," in *Proceedings of the AAAI Conference on Artificial Intelligence,* 2021, 35(4), 2021, 3555–62.

[37] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2018, 2018, 4490–9.

[38] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, "Class-balanced grouping and sampling for point cloud 3d object detection," *arXiv:1908.09492, 2019*, 2019.