Ph.D. DISSERTATION

Improving LiDAR-based 3D Object Detection with Part-Aware Data Augmentation and Mixture Density Network

구조 감응형 데이터 증강 기법과 혼합 밀도 신경망을 이용한 라이다 기반 3차원 객체 검출 개선

BY

JAESEOK CHOI

FEBRUARY 2023

Intelligent Systems
Department of Transdisciplinary Studies
Graduate School of Convergence Science and Technology
SEOUL NATIONAL UNIVERSITY

Ph.D. DISSERTATION

Improving LiDAR-based 3D Object Detection with Part-Aware Data Augmentation and Mixture Density Network

구조 감응형 데이터 증강 기법과 혼합 밀도 신경망을 이용한 라이다 기반 3차원 객체 검출 개선

BY

JAESEOK CHOI

FEBRUARY 2023

Intelligent Systems
Department of Transdisciplinary Studies
Graduate School of Convergence Science and Technology
SEOUL NATIONAL UNIVERSITY

Improving LiDAR-based 3D Object Detection with Part-Aware Data Augmentation and Mixture Density Network

구조 감응형 데이터 증강 기법과 혼합 밀도 신경망을 이용한 라이다 기반 3차원 객체 검출 개선

지도교수 곽 노 준

이 논문을 공학박사 학위논문으로 제출함

2023년 2월

서울대학교 대학원

융합과학부 지능형융합시스템전공

최 재 석

최재석의 공학박사 학위 논문을 인준함

2023년 2월

위 원 장:	이 교 구	(인)
부위원장:	곽 노 준	(인)
위 원:	박 재 흥	(인)
위 원:	이 원 종	(인)
위 원:	이 민 식	(인)

Abstract

LiDAR (Light Detection And Ranging), which is widely used as a sensing device for autonomous vehicles and robots, emits laser pulses and calculates the return time to sense the surrounding environment in the form of a point cloud. When recognizing the surrounding environment, the most important part is recognizing what objects are nearby and where they are located, and 3D object detection methods using point clouds have been actively studied to perform these tasks.

Various backbone networks for point cloud-based 3D object detection have been proposed according to the preprocessing method of point cloud data. Although advanced backbone networks have made great strides in detection performance, they are largely different in structure, so there is a lack of compatibility with each other. The problem to be solved in this dissertation is "How to improve the performance of 3D object detectors regardless of their diverse backbone network structures?". This dissertation proposes two general methods to improve point cloud-based 3D object detectors.

First, we propose a part-aware data augmentation (PA-AUG) method which maximizes the utilization of structural information of 3D bounding boxes. Since the 3D bounding box labels fit the object's boundaries and include the orientation value, they contain the structural information of the object in the box. To fully utilize the intra-object structural information, we propose a novel part-aware partitioning method which separates 3D bounding boxes with characteristic sub-parts. PA-AUG applies newly proposed data augmentation methods at the partition level. It makes various types of 3D object detectors robust and

brings the equivalent effect of increasing the train data by about 2.5×.

Second, we propose a mixture-density-based 3D object detection (MD3D). MD3D predicts the distribution of 3D bounding boxes using a Gaussian mixture model (GMM). It reformulates the conventional regression methods as a density estimation problem. Thus, unlike conventional target assignment methods, it can be applied to any 3D object detector regardless of the point cloud preprocessing method. In addition, as it requires significantly fewer hyper-parameters compared to existing methods, it is easy to optimize the detection performance. MD3D also increases the detection speed due to its simple structure.

Both PA-AUG and MD3D can be applied to any 3D object detector and shows an impressive increase in detection performance. The two proposed methods cover different stages of the object detection pipeline. Thus, they can be used simultaneously, and the experimental results show they have a synergy effect when applied together.

keywords: 3D Object Detection, LiDAR, Point cloud, Data augmentation, Mixture density networks, Autonomous driving

student number: 2017-23640

Contents

Abstract **i**

Contents **iii**

List of Tables **vii**

List of Figures **ix**

1 Introduction **1**

1.1 Problem Definition 3

1.2 Challenges 6

1.3 Contributions 8

1.3.1 Part-Aware Data Augmentation (PA-AUG) 8

1.3.2 Mixture-Density-based 3D Object Detection (MD3D) . 9

1.3.3 Combination of PA-AUG and MD3D 10

1.4 Outline 10

2 Related Works **11**

2.1 Data augmentation for Object Detection 11

2.1.1 2D Data augmentation 11

2.1.2 3D Data augmentation 12

2.2 LiDAR-based 3D Object Detection 13
2.3 Mixture Density Networks in Computer Vision 15
2.4 Datasets 16
2.4.1 KITTI Dataset 16
2.4.2 Waymo Open Dataset 18
2.5 Evaluation metric 19
2.5.1 Average Precision (AP) 19
2.5.2 Average Orientation Similarity (AOS) 22
2.5.3 Average Precision weighted by Heading (APH) 22

3 Part-Aware Data Augmentation (PA-AUG) 24
3.1 Introduction 24
3.2 Methods 27
3.2.1 Part-Aware Partitioning 27
3.2.2 Part-Aware Data Augmentation 28
3.3 Experiments 33
3.3.1 Results on the KITTI Dataset 33
3.3.2 Robustness Test 36
3.3.3 Data Efficiency Test 38
3.3.4 Ablation Study 40
3.4 Discussion 41
3.5 Conclusion 42

4 Mixture-Density-based 3D Object Detection (MD3D) 43
4.1 Introduction 43
4.2 Methods 47

4.2.1 Modeling Point-cloud-based 3D Object Detection with Mixture Density Network . . . 47
4.2.2 Network Architecture . . . 49
4.2.3 Loss function . . . 52
4.3 Experiments . . . 53
4.3.1 Datasets . . . 53
4.3.2 Experiment Settings . . . 53
4.3.3 Results on the KITTI Dataset . . . 54
4.3.4 Latency of Each Module . . . 56
4.3.5 Results on the Waymo Open Dataset . . . 58
4.3.6 Analyzing Recall by object size . . . 59
4.3.7 Ablation Study . . . 60
4.3.8 Discussion . . . 65
4.4 Conclusion . . . 66

5 Combination of PA-AUG and MD3D **71**
5.1 Methods . . . 71
5.2 Experiments . . . 72
5.2.1 Settings . . . 72
5.2.2 Results on the KITTI Dataset . . . 73
5.3 Discussion . . . 76

6 Conclusion **77**
6.1 Summary . . . 77
6.2 Limitations and Future works . . . 78
6.2.1 Hyper-parameter-free PA-AUG . . . 78
6.2.2 Redefinition of Part-aware Partitioning . . . 79

6.2.3 Application to other tasks 79

Abstract (In Korean) **94**

감사의 글 **96**

List of Tables

1.1 Comparison of the point cloud encoding methods 5

3.1 Performance comparison on the KITTI-val set 34
3.2 Parameters used in the KITTI experiments 35
3.3 Robustness Test 36
3.4 Comparison of partitioning methods 39
3.5 Comparison of the number of partitions 39

4.1 Performance comparison on the KITTI-val set 54
4.2 Latency of each module 57
4.3 Performance comparison on the Waymo open dataset with 202 validation sequences 58
4.4 Comparison of box encoding methods 62
4.5 Comparison of probability distributions 62
4.6 Effect of the number of mixture components 64
4.7 Effect of constraint on the covariance matrix 65
4.8 Detection heads 68
4.9 Backbone network architectures (1) 69
4.10 Backbone network architectures (2) 70

5.1 Performance comparison of the combination of PA-AUG and MD3D on SECOND. 74
5.2 Performance comparison of the combination of PA-AUG and MD3D on CenterPoint. 75

List of Figures

1.1 Illustrations of the 2D and 3D Object Detection 2

1.2 Illustration of the basic LiDAR-based 3D Object Detection Pipeline 4

2.1 Examples of 2D Data Augmentation 12

2.2 Examples of Instance-level 3D Data Augmentation 13

2.3 Samples of the KITTI dataset 17

2.4 Samples of the Waymo open dataset 18

2.5 Precision-recall (PR) curve 21

3.1 Comparison between 2D and 3D bounding box 25

3.2 Illustration of the part-aware partitioning 28

3.3 Illustration of the part-aware data augmentation (PA-AUG) . . . 29

3.4 Qualitative results on the corrupted KITTI datasets 37

3.5 Data Efficiency Test 38

3.6 Failure samples of the wrong partitioning methods 41

4.1 Problems caused by hand-crafting factors of anchor boxes . . . 44

4.2 The overall architecture of MD3D 49

4.3 Illustration of corner regression 51

4.4 Recalls on the KITTI-val set 59

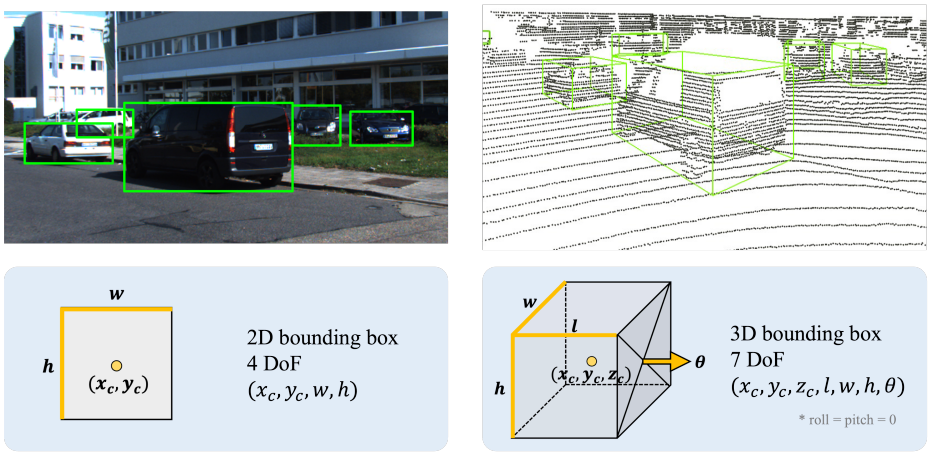
4.5 Illustration of various box encoding methods 60
4.6 Pdfs of the Gaussian, Cauchy, and Laplace distributions 63
4.7 Qualitative results for the KITTI-val set 67

Chapter 1

Introduction

3D Object Detection is the task of predicting the positions, sizes, orientations, and classes of the target objects with 3D bounding boxes in a given scene. It is an essential perception task for autonomous vehicles, robots, and surveillance systems. The most frequently used sensing devices for object detection are RGB cameras and Light Detection and Ranging (LiDAR). Each sensing device has advantages and disadvantages. As shown in Figure 1.1, the images obtained from RGB cameras contain rich texture information but lack depth information. Contrarily, the point clouds obtained from LiDAR have 3D positional information but lack texture information. Therefore, they are suitable for 2D and 3D object detection, respectively.

Many 3D object detectors utilize a monocular image as input [77, 8, 6, 50, 78] by estimating the depth values and the 3D poses of the target objects. However, the limited accuracy of the estimated values makes the detection unstable. Some stereo camera-based methods [10, 40, 88, 57, 58] have been proposed for better accurate depth estimation, but they showed poor performance with textureless regions. [2] The ideal case would be the detectors that utilize both the



(a) 2D Object Detection (b) 3D Object Detection

Figure 1.1: Illustrations of the 2D and 3D Object Detection. (a) The 2D bounding boxes consist of the 2D center coordinates (x_c, y_c) and 2D sizes (w, h). (b) The 3D bounding boxes consist of the 3D center coordinates (x_c, y_c, z_c), 3D sizes (l, w, h), and yaw angle (θ). The roll and pitch angles are ignored. Additionally, both two tasks predict the classes of the target objects.

RGB images and the point clouds [54, 11, 52, 42, 64]. They can detect objects using texture information and accurate depth values. However, it is not easy to exploit both modalities due to their different unique characteristics. Also, the detectors using multi-modal inputs are too slow to utilize in real-world environments.

In this dissertation, we mainly focus on 3D object detection using only point clouds obtained from LiDAR, which is the most balanced setting in real-world environments for autonomous vehicles and robots. The purpose of this dissertation is to propose two novel methods to enhance LiDAR-based 3D object detection. The first one is a novel data augmentation method. Data augmentation

plays an important role in boosting the performance of 3D models. However, 3D data augmentation has not been explored much compared to 2D's. So, we propose a part-aware data augmentation (PA-AUG) method which utilizes rich structural information of 3D labels. It has the equivalent effect of increasing the train data by about 2.5×. The second one is a novel anchor-free detection head which adopts the mixture density networks [4]. Unlike image-based 2D anchors, 3D anchors must be placed in a 3D space and determined differently for each class of different sizes. This imposes a significant burden on the design complexity. Therefore, we proposed a mixture-density-based 3D object detection (MD3D) method to predict the distribution of 3D bounding boxes using a Gaussian mixture model (GMM). With an anchor-free detection head, MD3D requires few hand-crafted design factors and eliminates the inefficiency of separating the regression channel for each class, thus offering both latency and memory benefits.

1.1 Problem Definition

The overall pipeline of LiDAR-based 3D object detection is similar to image-based 2D object detection. Nevertheless, their internal components are significantly different. As shown in Figure 1.2, the basic one-stage LiDAR-based 3D object detectors consist of three stages.

The first stage is data augmentation and preprocessing. Data augmentation is a method to artificially increase the amount of train data to enhance detection performance. The raw point cloud L obtained from LiDAR is represented as a set of point vectors $L \in \mathbb{R}^{N\times(3+c)}$ (3D coordinates and LiDAR features). The number of points N is different for each scene. Thus it is necessary to preprocess

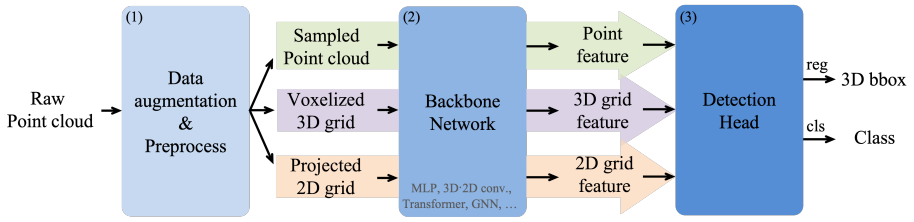


Figure 1.2: Illustration of the basic LiDAR-based 3D Object Detection Pipeline.

the point cloud to perform batch inference. The simplest preprocessing method is sampling the raw point cloud with a predefined number of points. This representation preserves most of the original data, but due to its unordered point set, it is hard to know nearby points without calculating distances for each point. Therefore 3D voxelization is a widely used method to encode point clouds into a 3D grid format. With this encoding, the time-consuming distance calculation is not needed to find nearby points. As well as 3D voxelization, projecting point clouds into 2D grids, such as bird's-eye view (BEV) or front view (FV), is also a widely used encoding method. Both of the grid encoding methods make point clouds structurally formatted. However, they lose much information by rasterizing original point clouds. There are other encoding methods, such as a graph, but those mentioned above are the most widely used.

The second stage is the backbone network. Due to the various point cloud encoding methods, there are many backbone networks according to the encoding methods. The sampled point cloud is featurized using the multi-layer perceptron (MLP) or Transformer [73], which are suitable for point sets. This representation needs to calculate the distances between each other. Hence, a local grouping method, such as k-nearest neighbor (kNN) or ball query, must be conducted before featurizing the sampled point clouds. The voxelized 3D grid

Encoding	Advantages	Disadvantages
Sampled Point cloud	• preserves almost all of the raw data • fast preprocessing time (random sampling)	• needs to calculate the distances between points from each other to find local neighboring points
Voxelized 3D grid	• easy to aggregate local features using 3D conv. • preserves more data than 2D grid	• heavy backbone network (3D CNNs)
Projected 2D grid	• fastest inference time • can use 2D object detectors' backbone networks	• loses a lot of original data due to rasterizing effect

Table 1.1: Comparison of the point cloud encoding methods.

uses 3D convolutional neural networks (CNNs) to featurize it. The point clouds obtained from LiDAR are concentrated in close regions; thus, the 3D grid is inherently sparse. Therefore, sparse convolution [26] and submanifold sparse convolution [27] are used instead of the normal 3D convolution operation to boost computation time. They calculate only the occupied voxels using a hash table which is utilized at indexing active input and output voxel positions. The 2D grids projected as FV adopts the normal 2D CNNs, but the BEV 2D grids use 2D sparse CNNs. Each encoding method has trade-offs between detection accuracy and latency. As well as the single encoding-based backbone networks, multiple backbone networks are used to exploit the advantages of different types

of features by fusing their output features. [96, 84, 59, 60, 28, 11, 45, 48]

The last stage is the detection head. The detection heads are classified into two groups; anchor[1]-based and anchor-free methods. Anchor-based detectors detect target objects by predicting offsets from nearby anchors for each class. The anchors make the training stable, but a lot of anchor-related hyper-parameters are predefined. Many anchor-free detectors use the 2D projected grid as a point cloud encoding method, and they predict target objects with a heatmap estimation.

In this dissertation, we aim to solve a question: *How to improve the detection performance regardless of the backbone network architectures?* Due to the various types of point cloud encoding methods, 3D object detectors have very different architectures. We propose two methods that can be applied to any 3D object detector. The first method is **part-aware data augmentation (PA-AUG)** [16], a novel data augmentation method that utilizes rich structural information of 3D bounding boxes. The second method is **mixture-density-based 3D object detection (MD3D)** [15], a novel detection head that reformulates predicting target objects as a density estimation problem.

1.2 Challenges

Utilizing structural information of 3D bounding boxes

Many 3D object detectors [59, 76, 7, 29] apply data augmentation, such as translation, flipping, shifting, scaling, and rotation, directly extending typical 2D data augmentation methods to 3D. These existing methods are effective in improv-

[1] anchor or anchor box is a set of predefined 3D boxes which are tiled across the scene to make the bounding box regression easy

ing performance. However, they did not fully utilize the 3D information. 3D ground-truth boxes have much richer structural information compared to 2D ground-truth boxes as they perfectly fit the object along with each direction. For example, since the 2D boxes have no structural information about the objects, they cannot tell which part of the car is the 'wheel'. However, we can be aware the wheels are located near the bottom corners using the intra-object part location information of 3D boxes. Utilizing the unique characteristics of 3D boxes enables more sophisticated and effective augmentation which 2D augmentation cannot do. In this dissertation, we propose a novel data augmentation method utilizing structural information of 3D bounding boxes.

Various feature encoding methods

Point cloud-based 3D object detectors adopt various data preprocessing methods: sampling point clouds with a predefined number, voxelization, projection, and their hybrid methods. Therefore, the types of feature encodings differ significantly. Hence, there are a lot of different training strategies and detection heads, which are designed for a specific feature encoding type. It is very hard to unify them with a single training strategy owing to the different characteristics of the feature encodings. In this dissertation, we propose a novel training strategy predicting 3D bounding boxes with mixture density networks. By reformulating the 3D bounding box regression with a density estimation problem, we could train various types of 3D object detectors with a single training method.

Anchor-based 3D Object Detection

The existing 3D object detectors have mostly adopted anchor-based detection methods. However, anchor-based detectors suffer from a fatal problem; they

must predefine many anchor-related hyper-parameters: 1) anchor size, 2) anchor direction, 3) stride that determines anchor placement, and 4) target assignment policy. Each of these factors dominantly influences the performance and latency of the detectors. Thus, they must be defined separately for each class. In this dissertation, we propose a novel anchor-free detection head which requires few hand-crafted design factors.

1.3 Contributions

The major contribution of this dissertation is that our proposed methods improve the 3D object detection performance regardless of the backbone network architectures. To achieve general usability, we focus on two stages of the 3D object detection pipeline: data augmentation and detection head. As well as the two proposed methods show impressive improvements on detection performance, the combination of them brings a much greater increase in accuracy. In this section, we demonstrate the contributions of them in detail.

1.3.1 Part-Aware Data Augmentation (PA-AUG)

Conventional data augmentation methods for 3D object detection are applied to a whole scene or individual objects. For example, translation, flipping, shifting, scaling, and rotation are applied to a whole scene or each object. The proposed PA-AUG divides objects into predefined partitions. Then, we stochastically apply five data augmentation methods for each partition, not for each object. The partition-level data augmentation method allows the network to recognize intra-object relations as it learns individual variations in an intra-object part. It is compatible with existing point cloud data augmentation methods and can be used

universally regardless of the detector's architecture. PA-AUG has improved the performance of state-of-the-art 3D object detector for all classes of the KITTI dataset and has the equivalent effect of increasing the train data by about 2.5×. We also show that PA-AUG not only increases performance for a given dataset but also is robust to corrupted data.

1.3.2 Mixture-Density-based 3D Object Detection (MD3D)

Many bounding box regression methods for 3D object detection come from 2D object detectors. Among them, the most widely used regression method is anchor-based regression. The design factors of anchor boxes, such as shape, placement, and target assignment policy, greatly influence the performance and latency of the 3D object detectors. Unlike image-based 2D anchors, 3D anchors must be placed in a 3D space and determined differently for each class of different sizes. This imposes a significant burden on the design complexity. To tackle this issue, various studies have been conducted on how to set the anchor form. However, for practical reasons, anchor-based methods select the anchor design by compromising between performance and latency. Consequently, only objects that are similar in shape and size to an anchor can obtain high accuracy.

We propose MD3D which predicts the distribution of 3D bounding boxes using a Gaussian Mixture Model (GMM). With an anchor-free detection head, MD3D requires few hand-crafted design factors and eliminates the inefficiency of separating the regression channel for each class, and thus offering both latency and memory benefits. MD3D is designed to utilize various types of feature encoding; therefore, it can be applied flexibly by replacing only the detection head of the existing detectors. Experimental results on the KITTI and Waymo open datasets show that the proposed method outperforms its counterparts that

are based on the conventional anchor-based detection head in its overall performance, latency, and memory.

1.3.3 Combination of PA-AUG and MD3D

Both PA-AUG and MD3D are designed to be utilized in any 3D object detectors. Thus, they can be applied to a single object detector simultaneously. The combination of them makes detectors robust, efficient, and parameter-free. The experimental results on anchor-based and anchor-free detectors show that the combination of them improves detection performance much more than when PA-AUG and MD3D are applied separately. It means they affect different aspects of detection performance and have synergy when they are applied together.

1.4 Outline

In the following chapters, this dissertation is composed as follows. In Chapter 2, we describe previous works on data augmentation methods, LiDAR-based 3D object detectors, and mixture density networks utilized in the field of computer vision. In Chapter 3, we propose a novel data augmentation method, PA-AUG. In Chapter 4, we propose a novel detection head, MD3D. In Chapter 5, we present the experimental results on the combination of PA-AUG and MD3D. Finally, Chapter 6 summarizes this dissertation and discusses limitations and future works.

Chapter 2

Related Works

2.1 Data augmentation for Object Detection

2.1.1 2D Data augmentation

It has been demonstrated that data augmentation leads to gains in 2D image tasks such as classification and object detection [94, 34, 69]. Especially, patch-based data augmentation methods that utilize patches cut and pasted among training images boosted performance. Image patches are zeroed-out in [19], which encourages the model to utilize the full context of the image, on the other hand, deleted regions become uninformative. Cutmix [90] replaces deleted regions with a patch from another image and maximizes the proportion of informative pixels. These methods, when applied to CIFAR and ImageNet datasets, greatly improve the performance. Such improvements were also shown in low-level vision tasks. Cutblur [86] cuts a low-resolution patch and replaces it with the corresponding high-resolution image region and vice versa. It has the same effect as making the image partially sparse and enables the model to learn both "how" and "where" when super-resolves the image.

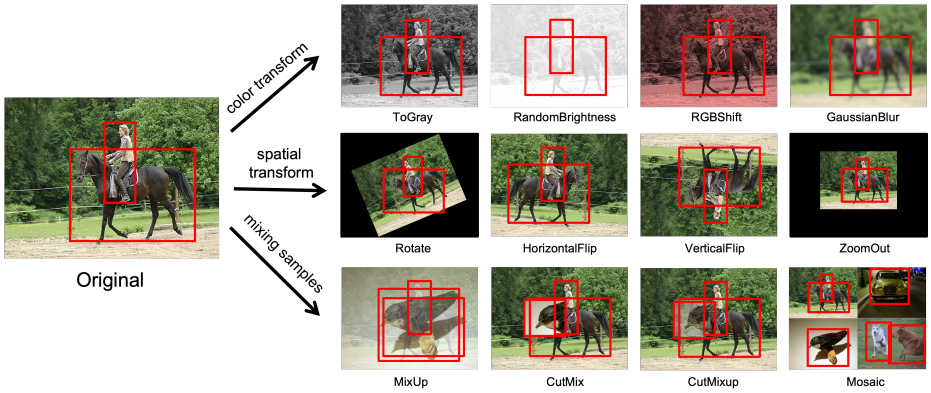


Figure 2.1: Examples of 2D Data Augmentation

In our work, the 2D image patch is extended to 3D partition. Using the 3D partition, we extend cutout [19], cutmix [90], and cutblur [86] to 3D point clouds. Five proposed augmentation methods are simultaneously applied to the partitions which gives robustness to the model and significantly improves performance.

2.1.2 3D Data augmentation

Considering the limited size of datasets for 3D object detection including KITTI datasets, data augmentation is one of the ways to alleviate overfitting and boost performance. The works [59, 76, 7] which showed the improved performance on 3D object detection adopted data augmentation methods such as translation, random flipping, shifting, scaling and rotation. Oversampling was also used to address foreground-background class imbalance problem [81, 59, 29].

Despite their effectiveness on the models, existing data augmentation methods do not fully utilize richer information of point clouds compared to the counterparts for 2D images. We propose part-aware data augmentation which takes

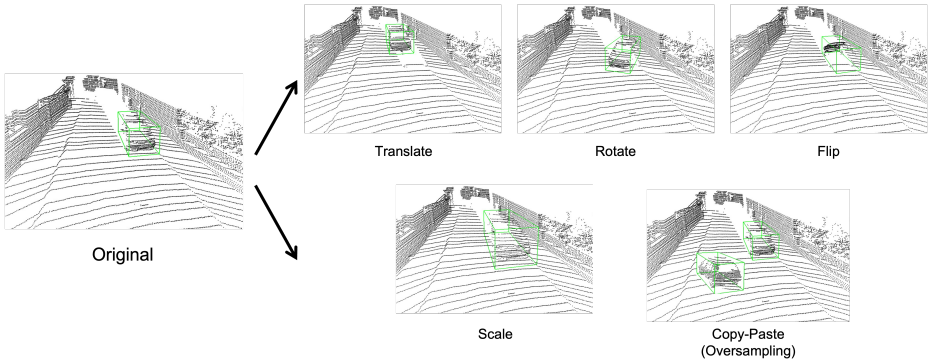


Figure 2.2: Examples of Instance-level 3D Data Augmentation.

full advantage of spatial information unique in 3D datasets.

Recently, automated data augmentation approaches have been actively studied. [13] narrowed down the search space with an evolutionary-based search algorithm and adopted the best parameters discovered. [41] jointly optimized augmentor and classifier via an adversarial learning strategy. These approaches could be incorporated with our proposed part-aware data augmentation to further enhance the performance.

2.2 LiDAR-based 3D Object Detection

As point clouds provide accurate geometric scene information, point cloud-based methods have achieved high performance in 3D object detection. [20, 38, 89, 3, 51] However, the inherent properties of irregular and sparse point clouds impose difficulties in data processing. Therefore, various forms of point cloud representations have been proposed. SECOND [81] groups the point clouds into voxels and utilizes spatially sparse convolution, thereby reducing the computational burden of the 3D convolution. PointPillars [36] encodes point clouds

into stacked pillars and operated all processes with only 2D convolutions, removing the bottleneck of 3D convolutions. PointRCNN [61] directly learns representations from raw point clouds using PointNet++ [55, 56] as a backbone network and generates bounding box proposals efficiently by taking advantage of point segmentation, which provided a powerful clue to 3D object detection. VoteNet [53] also uses PointNet++ as a backbone and detects objects using a deep Hough voting method. PV-RCNN [59] utilizes both voxel-based and point-based operations to encode multi-scale features and provide accurate location information efficiently. A graph method is adpoted in [67, 82] to detect objects.

The latest point cloud-based methods [33, 44, 75, 74, 80] compensate for the limitations of the existing detectors and significantly improve the accuracy and latency. Focal sparse convolution [12] predicts the importance of features in performing sparse convolution and selectively computes high importance features. IA-SSD [92] reduces the computational overhead of raw point-based detectors by using a learnable downsampling strategy. SST [21] improves the detection accuracy by introducing a single-stride backbone network that utilizes transformer blocks. Although the latest works have improved many aspects of the existing detectors, most have focused on improving the backbone structure.

Most anchor-free 3D object detectors [23, 32, 7, 85, 39] use a classification method based on heatmap estimation, which is primarily adopted in 2D object detection [37, 95, 71]; hence, they can only be used for 2D-projected features. In addition, the heatmap-based heads still have many hand-crafted design factors, such as, the Gaussian radius of the heatmap and the foreground assignment policy for regression. Because the proposed method does not require a GT assignment policy for regression, the design factors can be significantly reduced. There is no restriction on the input features, so the proposed method can be

flexibly applied to various types of point cloud features.

2.3 Mixture Density Networks in Computer Vision

Originally, mixture density networks (MDN) [4] was proposed to predict a continuous quantity under uncertainty. The MDN has recently attracted considerable attention, especially in object detection tasks because capturing uncertainties and coping with mislocalization have become critical issues. He *et al.* [31] measured the uncertainties of bounding boxes to deal with challenging cases, such as occlusion, while Feng *et al.* [22] extended it to LiDAR 3D vehicle detection. The MDN was also utilized to model the multi-modal nature of object detection and human pose estimation [72], and to address active learning for object detection [14].

We address the problem of complex anchor design that restricts both performance and latency in 3D object detection and apply MDN to overcome this limitation. The proposed MD3D is inspired by MDOD [87], which reformulated the 2D object detection task as a density estimation problem, and it reduced the complex processing and heuristics in the training. We extend their works to the 3D object detection task and improve the existing detectors in a plug-and-play manner with a flexible detection head that is compatible with any representation of point clouds. Unlike images, point clouds have various feature forms (BEV, FV, voxel, point, *etc.*). MD3D can be flexibly applied to these different types of features and easily replace the detection heads of existing detectors.

2.4 Datasets

In this dissertation, we focus on outdoor scene datasets which include point cloud scene data obtained from LiDAR and oriented 3D bounding box labels. There are some indoor datasets with point clouds and 3D bounding boxes. [65, 1, 17] However, their labels are axis-aligned or highly overlapped, which are not suitable for our proposed methods. The most widely used outdoor scene datasets for autonomous driving are the KITTI [24] and Waymo open datasets [68]. This section describes the details of the two datasets used for evaluation.

2.4.1 KITTI Dataset

The KITTI dataset [24] is the most widely used dataset for 3D object detection research. It contains 6 hours of traffic scenarios captured in the metropolitan area of Karlsruhe, Germany, during the daytime. The sensors used to collect the dataset are two grayscale cameras, two color cameras, and one Velodyne HDL-64E rotating 3D laser scanner (field of view: 360°horizontal, 26.8°vertical, range: 120m, 10Hz). We only use the point clouds obtained from the LiDAR device to detect target objects. The points consist of a 3D position (x, y, z) and a reflectance value (r). The number of points in a scene is not constant; it has about 120,000 points on average. The total size of the raw data is 180 GB.

The object annotations in the form of a 3D bounding box are not fully provided to the raw data. The KITTI dataset for 3D object detection consists of 7,481 training samples and 7,518 testing samples, where the training samples are generally divided into *train* split with 3,712 samples and *val* split with 3,769 samples. It contains around 80,000 annotated 3D bounding boxes encoded as 7-DOF $(x_c, y_c, z_c, l, w, h, \theta)$, representing the center 3D coordinate, length,

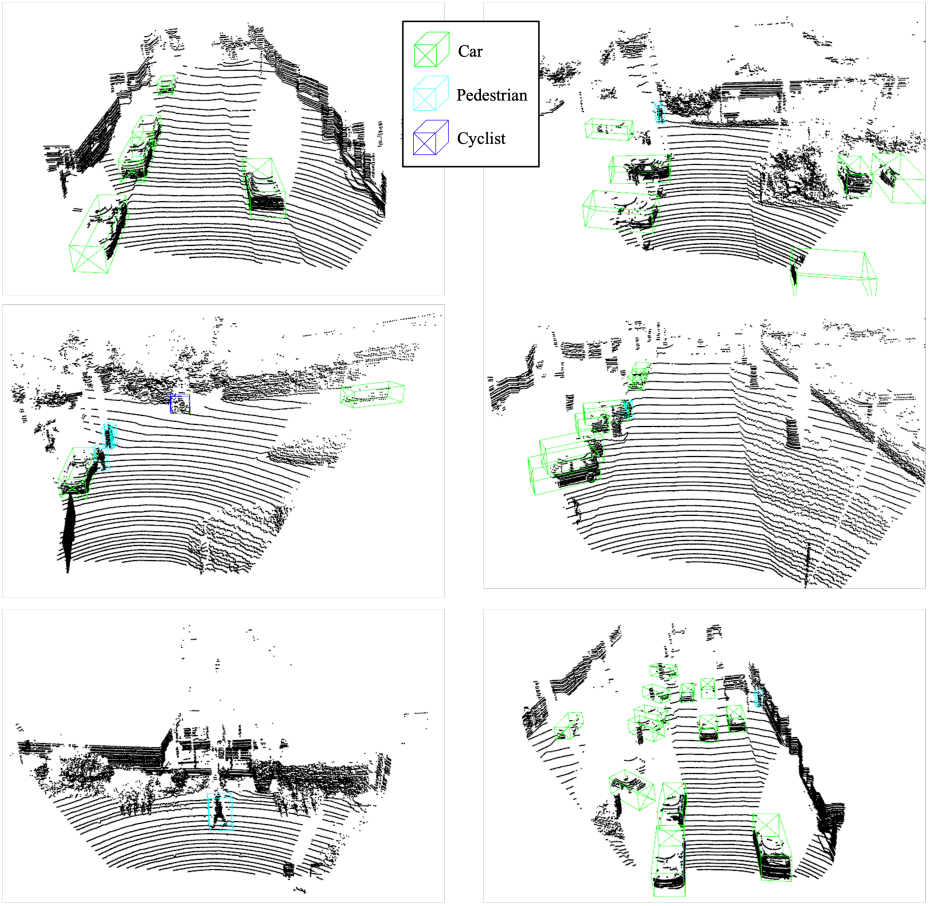


Figure 2.3: Samples of the KITTI dataset

width, height, and yaw angle, respectively. The roll and pitch angles are ignored. Because the KITTI dataset contains only 90°annotation, we clipped the scenes into (0, 70.4)m, (-40, 40)m, and (-3, 1)m for the X, Y, and Z axis ranges. The labels are classified into ‘Car’, ‘Van’, ‘Truck’, ‘Pedestrian’, ‘Person (sitting)’, ‘Cyclist’, ‘Tram’ and ‘Misc’ (e.g., trailers, segways). However, we train and evaluate only three classes (‘Car’, ‘Pedestrian’, and ‘Cyclist’) following the conventional detection methods. Moreover, it includes three levels for difficulty

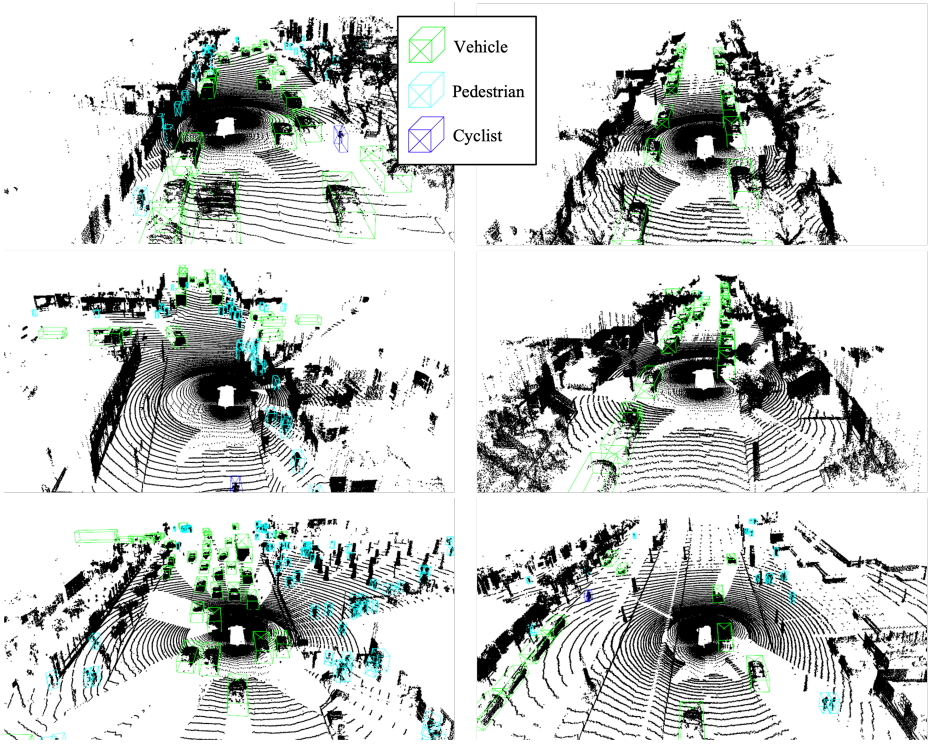


Figure 2.4: Samples of the Waymo open dataset

ratings, ‘Easy’, ‘Moderate’, and ‘Hard’, according to the height, occlusion, and truncation of GTs.

2.4.2 Waymo Open Dataset

The Waymo open dataset [68] is one of the largest public 3D object detection datasets. It contains 6.4 hours of diverse scenarios captured in multiple cities: San Francisco, Phoenix, and Mountain View. The sensors used to collect the dataset are five high-resolution pinhole cameras and five LiDAR sensors. The five LiDAR sensors are installed at the top, front, rear, side left, and side right. The top LiDAR sensor has [-17.6°, 2.4°] vertical field of view (VFOV) and 75m

range. And the other LiDAR sensors have [-90°, 30°] VFOV and 20m range. The points consist of a 3D position (x, y, z), intensity (r), and elongation (e). The elongation with intensity is useful for recognizing spurious objects, such as dust, fog, and rain. The number of points in a scene is not constant; it has about 177,000 points on average. The total size of the dataset is 1.2 TB.

The Waymo open dataset includes 798 training sequences with approximately 160k samples, 202 validation sequences with 40k samples, and 150 testing sequences with 30k samples. It contains around 12M annotated 3D bounding boxes encoded as 7-DOF $(x_c, y_c, z_c, l, w, h, \theta)$, which is the same as the KITTI dataset. Because of limited resources, we trained the models with 20% samples at regular intervals for each sequence, using a total of 32k training samples. The Waymo open dataset contains a complete 360°annotation, and we clip the scenes into (-75.2, 75.2)m, (-75.2, 75.2)m, and (-2, 4)m for the X, Y, and Z axis ranges. The labels are classified into 'Vehicle', 'Pedestrian', and 'Cyclist'. And it includes two levels for difficulty ratings: 'LEVEL_2' which seems hard examples for labelers or has ≤ 5 points inside the box and 'LEVEL_1' which is the rest of the 'LEVEL_2'.

2.5 Evaluation metric

2.5.1 Average Precision (AP)

Precision and recall are calculated as follows:

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN} \tag{2.1}$$

where TP, FP, and FN refer to true positives, false positives, and false negatives.

A threshold must be predefined to classify the predicted boxes as positives or negatives. Intersection over union (IoU) is used as the threshold and is defined as follows:

$$IoU = \frac{Area(PredBox \cap GTBox)}{Area(PredBox \cup GTBox)} \tag{2.2}$$

The predicted and GT boxes are 3D bounding boxes. Thus, the area of their intersection and union can be calculated in several ways: 2D (FV), BEV, and 3D. 2D and BEV IoU are calculated with 2D projected bounding boxes in front view and bird's eye view. 3D IoU is calculated with 3D bounding boxes as they are.

Average precision (AP) is the most commonly used evaluation metric in object detection research. AP is calculated using a precision-recall (PR) curve. As changing the detector's confidence score threshold, precision and recall change. Lowering the threshold increases recall but decreases precision. Precision and recall change oppositely according to the threshold. Thus, AP is measured by calculating the area under the PR curve to represent its varying scores.

$$AP = \int_0^1 p(r)dr \tag{2.3}$$

where r and $p(r)$ represent recall and precision.

Most object detectors do not calculate AP as in Equation 2.3. They use interpolated AP to facilitate calculation. It divides the recall into equally spaced values. (e.g., 11 recall points: 0, 0.1, ..., 1.0) Then, the AP is approximated by averaging their interpolated precision values, as shown in Figure 2.5.

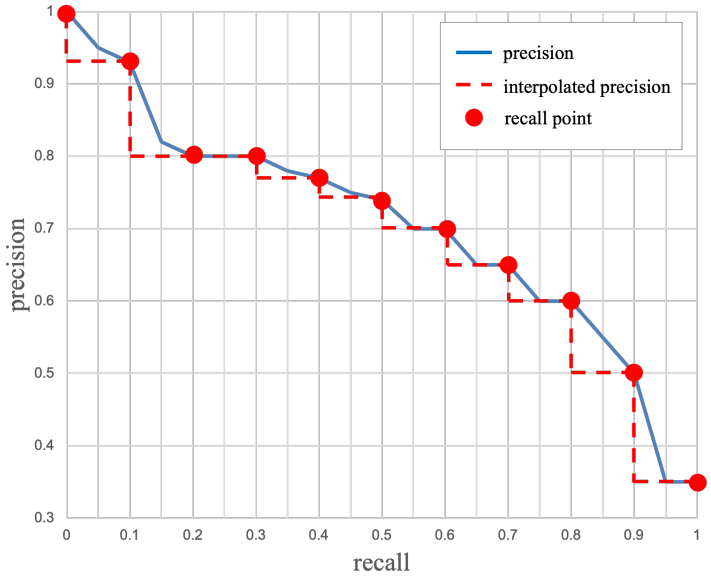


Figure 2.5: Precision-recall (PR) curve

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1.0\}} p_{interp}(r) \tag{2.4}$$

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r}) \tag{2.5}$$

where $p_{interp}(r)$ denotes the interpolated precision at a recall point r.

The AP is calculated for each class. The mean average precision (mAP) is the average of APs of all classes.

$$mAP = \frac{1}{C} \sum_{k=C}^{C} AP_k \tag{2.6}$$

where C is the number of classes.

2D, BEV, and 3D AP are calculated using 2D (FV), BEV, and 3D bounding boxes. Among them, 3D AP is the most precise metric to measure localiza-

tion accuracy. However, 3D AP cannot distinguish the bounding boxes at the same location with opposite orientations. To mitigate this problem, the KITTI and Waymo open datasets use the average orientation similarity (AOS) and the average precision weighted by heading (APH), respectively.

2.5.2 Average Orientation Similarity (AOS)

The AOS [25] is used to measure the accuracy of orientation predictions in the KITTI dataset and is defined as follows:

$$AOS = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1.0\}} \max_{\tilde{r}: \tilde{r} \geq r} s(\tilde{r}) \tag{2.7}$$

$$s(r) = \frac{1}{|\mathcal{D}(r)|} \sum_{i \in \mathcal{D}(r)} \frac{1 + cos\Delta_{\theta}^{(i)}}{2} \mathbb{1}_i \tag{2.8}$$

where $s(r) \in [0, 1]$ and $\mathcal{D}(r)$ represent the orientation similarity and the set of all detections at recall r. $\Delta_{\theta}^{(i)}$ is the difference of the orientation between prediction i and its assigned GT. $\mathbb{1}_i$ is set to 1 if ith detection is assigned to GT (exceeds predefined IoU threshold) and 0 if it is not assigned.

2.5.3 Average Precision weighted by Heading (APH)

The APH [68] is devised to incorporate heading information into AP and is defined similarly to AP:

$$APH = \int_0^1 \max\{h(r') | r' \geq r\} dr \tag{2.9}$$

$$h(r) = \min(|\Delta_{\theta}|, 2\pi - |\Delta_{\theta}|)/\pi \times p(r) \tag{2.10}$$

The only difference in APH is the weighting by heading accuracy as in Equation 2.10. The heading difference $|\Delta_\theta|$ is represented in radians within $[0, 2\pi]$.

Chapter 3

Part-Aware Data Augmentation (PA-AUG)

3.1 Introduction

Although 3D object detection research has been largely conducted, most of the works focus on architectures suitable for 3D point clouds [36, 83, 30, 76, 46, 59]. Meanwhile, data augmentation plays an important role in boosting the performance of 3D models. 3D labeling is much more time-consuming compared to 2D labeling, which leads to most of the 3D datasets having a limited amount of training samples. Yet, 3D data augmentation has not been much explored.

Many works in 3D object detection apply data augmentation, such as translation, random flipping, shifting, scaling and rotation, directly extending typical 2D augmentation methods to 3D [59, 76, 7, 29]. These existing methods are effective in improving performance. However, they did not fully utilize the 3D information. 3D ground-truth boxes have much richer structural information compared to 2D ground-truth boxes as they perfectly fit the object along with each direction. For example, the 2D label may contain other instances and background in the box, so the provided information contains much noise. On

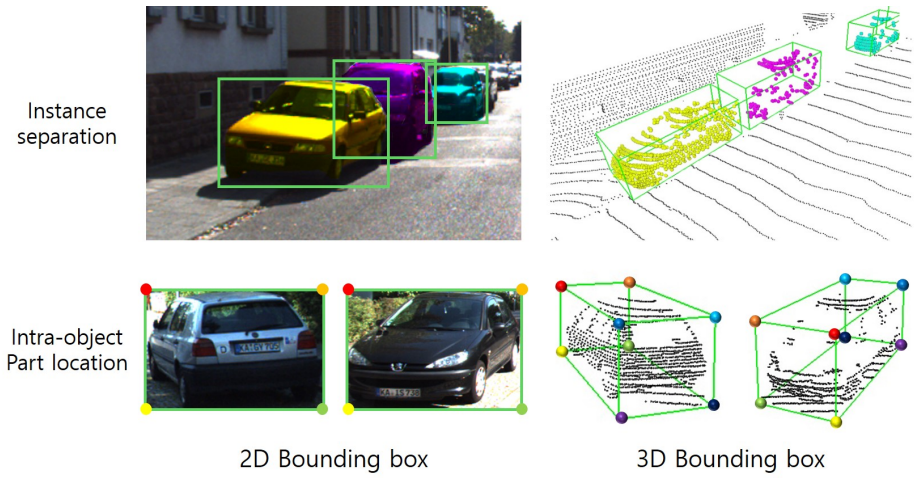


Figure 3.1: **Comparison between 2D and 3D bounding box. Top - Instance separation:** Unlike 2D, 3D has separate instances in the box and rarely contains background points. **Bottom - Intra-object Part location:** Unlike 2D, the corners in 3D boxes can be assigned to a specific order using the heading direction of the object and using this order the information of the part location of the object can be obtained (different color represents different corners).

the other hand, 3D boxes provide sufficient information of a single object that is even occluded and have little background noise (Figure 3.1, First row). Also, since the 2D boxes have no structural information about the objects, they cannot tell which part of the car is the 'wheel'. However, we can be aware the wheels are located near the bottom corners using the intra-object part location information of 3D boxes (Figure 3.1, Second row). Utilizing the unique characteristics of 3D boxes enables more sophisticated and effective augmentation which 2D augmentation cannot do.

In this chapter, we propose a part-aware data augmentation method robust to various extreme environments by using structural information of 3D ground-

truth boxes. The network can be aware of intra-object relation as it learns individual variation in an intra-object part. Our part-aware augmentation divides 3D ground-truth boxes into 8 or 4 partitions depending on the object type. It stochastically applies five augmentation methods to each partition, such as internal points dropout, cutmix [90], cutmixup [86], sparse sampling, and random noise generation. The internal points dropout removes partitions stochastically and leaves the corner of an object. It enables the network to find the entire box when only some parts of the object are given. Cutmix and cutmixup respectively replace and mix points in the partition with other points from the same class and same partition location, which give the network a regularization effect. Sparse sampling samples point clouds from a dense partition, sparsifying the partition from which the network can learn more information of distant objects. Random noise generation allows the network to learn situations of severe occlusion.

Note that [90, 86] apply cutmix and cutmixup to an image region with a patch from another class that the network could learn a relation across examples of different classes. In our work, however, points from the same class are mixed to give a regularization effect for intra-class examples. This reflects the task characteristics of 3D object detection that requires accurate localization while classifying 3 to 23 classes [24, 68, 5] centered on car, pedestrian and cyclist compared to [90, 86] which classify 1000 classes of ImageNet.

Our proposed part-aware data augmentation improves KITTI [24] Cyclist 3D AP of the PointPillars baseline [36] up to 8.91%p. Also, part-aware data augmentation enables the model to be robust in poor but inevitable environments, such as severe occlusion, low resolution, and inaccuracy due to snow or rain. In those situations, our work shows much less drop in accuracy than the baseline. In addition, part-aware augmentation performs well when data is insufficient,

which has the equivalent effect of increasing the train data by about 2.5×. As our work divides 3D box according to its structure and applies augmentation methods individually on the partitions, multiple augmentation methods are allowed to be used simultaneously without interference with each other. This can enhance the regularization effect a lot.

Our main contributions are:

- As well as a partitioning method utilizing the structural information of 3D labels, we propose five partition-based 3D LiDAR data augmentation methods which significantly enhance performance when they are used together.
- Our work is compatible with existing LiDAR data augmentation methods and boosts conventional detectors' performance with negligible cost.
- We show that proposed part-aware augmentation not only improves the recognition accuracy of given datasets but also obtains the robustness to corrupted data.

3.2 Methods

3.2.1 Part-Aware Partitioning

We propose a part-aware partitioning method that divides the object into partitions according to intra-object part location to fully utilize the structural information of 3D label. The term 'intra-object part location' used in [62] means a relative location of the 3D foreground points with respect to the corresponding bounding boxes and exploring the object part location improves performance. Part-aware partitioning is necessary to separate the characteristic sub-parts of

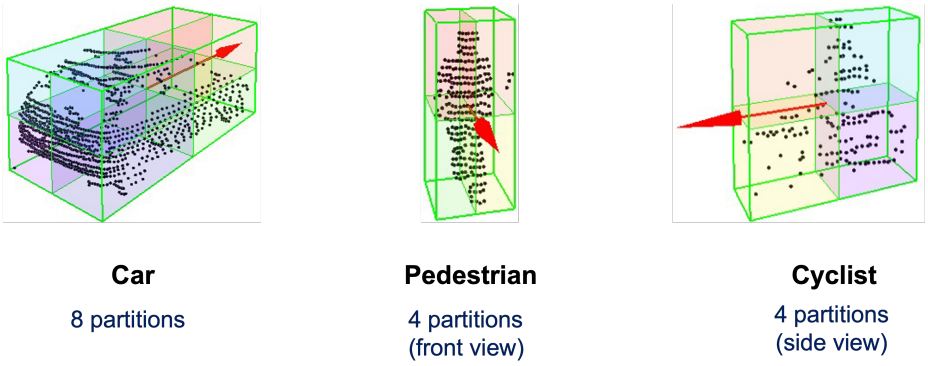


Figure 3.2: Illustration of the part-aware partitioning.

an object and it enables more diverse and efficient augmentation than existing methods. Because the location of characteristic parts for each class is different, Car, Pedestrian and Cyclist are divided into 8, 4 and 4 partitions respectively (Figure 3.2). When using partition-based augmentation, instead of applying the same augmentation to the entire object, different augmentations can be applied to each intra-object sub-parts.

3.2.2 Part-Aware Data Augmentation

Point cloud L can be expressed by the union of foreground points FP and background points BP as below:

$$L = FP \cup BP \tag{3.1}$$

$$FP = \cup_{i=1}^{N} B^{(i)}, \quad B^{(i)} = \cup_{j=1}^{T} P_j^{(i)}, \tag{3.2}$$

where B is the points in a 3D box, and N is the number of boxes in a scene. P is the internal points in a partition, and T is the number of partitions in the box.

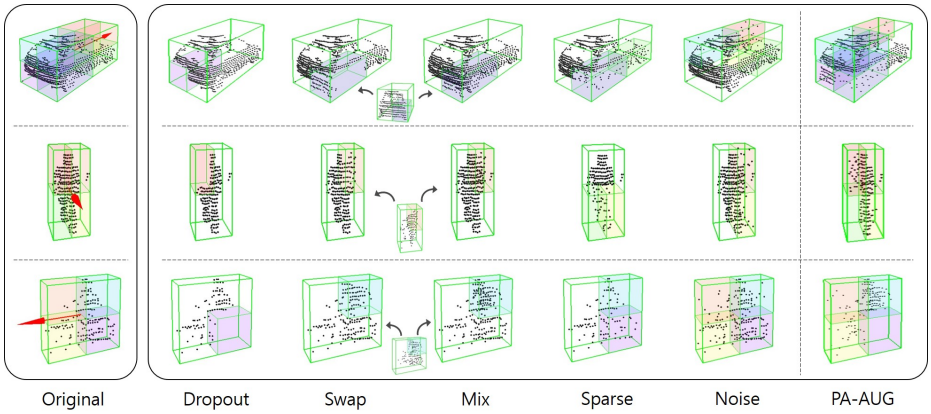


Figure 3.3: **Illustration of the part-aware data augmentation (PA-AUG).** The first column shows the original point cloud and part-aware partitioning method for Car, Pedestrian, and Cyclist classes. The other columns show examples of the proposed five partition-based augmentation methods and PA-AUG. The augmented partitions are marked with colors. Because Swap and Mix operations fetch points from different instances, the imported objects are also shown together.

The set of augmented foreground points FP_{aug} can be represented as

$$FP_{aug} = \cup_{i=1}^{N} \hat{B}^{(i)}, \quad \hat{B}^{(i)} = \cup_{j=1}^{T} \hat{P}_j^{(i)}. \tag{3.3}$$

Here, the bounding boxes and the partitions to which augmentation is applied are denoted as $\hat{B}$ and $\hat{P}$ respectively.

Dropout Partition

Dropout [66] was first used in the feature-level to increase the regularization effect of the network by randomly making the activation of some nodes zero. After that, it was shown that dropout could be effectively applied to the input

in the 2D image classification task [19]. Inspired by the previous works, we propose a partition-based dropout method that can be used effectively in 3D point clouds as below.

$$\hat{B}^{(i)} = \begin{cases} B^{(i)} & \text{if } r_i = 0 \\ \cup_{j \neq d}^{T} P_j^{(i)} & \text{if } r_i = 1 \end{cases} \quad \text{where } r_i \sim Ber(p_D). \tag{3.4}$$

In Eq. (3.4), $Ber(\cdot)$ indicates Bernoulli distribution, and dropout is applied to each bounding box with a probability of p_D. The index d indicates a randomly selected dropout partition among T partitions. Dropout using a predefined partition can remove characteristic sub-parts of an object, making learning more robust.

Swap Partition

CutMix [90], which is used in 2D image recognition, proposed an augmentation method that swaps random regions extracted from training samples. It can be applied to different classes by mixing class labels and has been shown effective for regularization. Inspired by the work, we propose a swap partition operation that utilizes intra-object part location information of 3D labels. The difference from CutMix is that our method swaps partitions of the same class and the same location in an object as follows.

$$\hat{B}^{(i)} = \begin{cases} B^{(i)} & \text{if } r_i = 0 \\ \cup_{j \neq k}^{T} P_j^{(i)} \cup \hat{P}_k^{(i)} & \text{if } r_i = 1 \text{ and } \mid P_k^{(i)} \mid \neq 0 \end{cases} \tag{3.5}$$

$$\text{where } r_i \sim Ber(p_W), 1 \leq k \leq T.$$

$$\hat{P}_k^{(i)} = P_k^{(i' -> i)} \tag{3.6}$$

$$P_k^{(i'->i)} = AffineTransform_i(P_k^{(i')}) \tag{3.7}$$

for $i \neq i'$, $1 \leq i' \leq N$ and $\mid P_k^{(i')} \mid \neq 0$.

As in the Eq. (3.5) - (3.7), after selecting a box i to swap with a probability of p_W for all boxes, we swap a randomly selected non-empty k^{th} partition in box i with the k^{th} partition in box i'. When swapping partitions, the partitions of different boxes have different scales, directions, and locations. So after transforming $P_k^{(i')}$ to the canonical coordinate system, we resize it to the scale of $P_k^{(i)}$ and transform it back to the coordinate system of $P_k^{(i)}$. As a result, $P_k^{(i'->i)}$ is created and the k^{th} partition in box i is replaced with it.

Because CutMix swaps patches of random areas, object can be swapped to the background area. And it could have a bad effect on learning. However, our partition-aware swap has no such problem and maximizes the effect of intra-class regularization by swapping only between the same class.

Mix Partition

CutMixup [86], a combination of CutMix [90] and Mixup [91], blends random areas of the training images, which is a quite effective data augmentation method in the task of image super-resolution. We applied it to our partition-based augmentation and call it Mix partition. The detailed method is almost identical to the Swap partition except that Eq. (3.6) is replaced by

$$\hat{P}_k^{(i)} = P_k^{(i)} \cup P_k^{(i'->i)} \tag{3.8}$$

for $i \neq i'$, $1 \leq i' \leq N$, $\mid P_k^{(i')} \mid \neq 0$ and $r_i \sim Ber(p_M)$.

The partition to mix is selected in the same way as the Swap operation. Likewise, the same partition transformation process is applied. The only differ-

ence is that it merges $P_k^{(i)}$ and $P_k^{(i'->i)}$ when creating augmented partition $\hat{P}_k^{(i)}$ rather than $P_k^{(i)}$ is replaced by $P_k^{(i'->i)}$.

Sparsify Partition

The density of LiDAR points decreases cubically as the distance of the box increases. As the point density decreases, the shape of the object cannot be fully recognized, which is one of the most significant factors in reducing the performance of LiDAR-based detectors. We propose sparsifying partitions as an augmentation method which makes some dense partitions sparse to improve distant objects' recognition. The detail is as the following.

$$\hat{P}_j^{(i)} = \begin{cases} P_j^{(i)} & \text{if } r_j = 0 \\ S_j^{(i)} & \text{if } r_j = 1 \text{ and } \mid P_j^{(i)} \mid > C_S \end{cases} \tag{3.9}$$

$$\text{where } r_j \sim Ber(p_S).$$

As in Eq. (3.9), it selects partitions to augment with the probability of p_S among the dense partitions with the number of points over C_S. Then, C_S points of the partition are sampled using Farthest Point Sampling (FPS) and it is denoted as $S_j^{(i)} \subset P_j^{(i)}$.

Add Noise to Partition

Since the RGB-image-based detectors are greatly influenced by the illuminance of the surrounding environment, the augmentation methods that change the brightness and color help improve performance. Likewise, LiDAR-based detectors are vulnerable to weather changes such as rain or snow that can cause noise and occlusion in point cloud data. We propose a partition-based augmentation

method to be robust to noise caused by various reasons as follows:

$$\hat{P}_j^{(i)} = \begin{cases} P_j^{(i)} & \text{if } r_j = 0 \\ P_j^{(i)} \cup P_{noise} & \text{if } r_j = 1 \end{cases} \quad \text{where } r_j \sim Ber(p_N) \tag{3.10}$$

As in Eq. (3.10), it selects partitions to augment with the probability of p_N. Then, it adds randomly generated C_N noise points P_{noise} to the selected partition $P_j^{(i)}$.

PA-AUG

The five augmentation methods using part-aware partitioning introduced above can be used individually, but because the methods are independent, different augmentation methods can be applied to an object multiple times. Therefore various combination of augmentations can be created, applying each operation independently so that different operations can be applied to one partition. However, if all augmentations are used without a specific order, interference may occur between operations. In order to minimize this, we apply Dropout-Swap-Mix-Sparse-Noise in order. We call it *PA-AUG*, which stochastically applies five operations. It can take advantage of each and show a strong regularization effect.

3.3 Experiments

3.3.1 Results on the KITTI Dataset

Settings

The KITTI object detection benchmark dataset [24] consists of 7,481 training samples and 7,518 testing samples. In order to verify the effectiveness of PA-AUG, we separated the training dataset into 3,712 training samples and 3,769

Method	Car 3D (IoU=0.7)			Cyc. 3D (IoU=0.5)			Ped. 3D (IoU=0.5)			mAP
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	
PointPillars [36]	80.29	68.68	66.59	61.97	40.75	38.49	54.47	49.48	45.38	56.23
PointPillars + Dropout	80.89	72.23	68.03	66.00	44.19	41.89	55.10	50.38	45.63	58.26
PointPillars + Swap	81.45	68.60	66.85	66.66	44.94	42.62	56.92	51.97	47.32	58.59
PointPillars + Mix	81.79	70.21	67.87	62.78	40.45	38.42	**59.98**	**54.60**	**48.87**	58.33
PointPillars + Sparse	82.56	69.83	67.27	66.88	44.37	42.00	58.47	53.62	48.64	59.29
PointPillars + Noise	82.03	68.37	65.81	66.44	44.31	41.79	57.81	52.55	47.73	58.54
PointPillars + PA-AUG	**83.70**	**72.48**	**68.23**	**70.88**	**47.58**	**44.80**	57.38	51.85	46.91	**60.42**
PV-RCNN [59]	89.15	80.43	78.48	85.54	71.21	65.42	66.08	59.48	55.22	72.33
PV-RCNN + PA-AUG	**89.38**	**80.90**	**78.95**	**86.56**	**72.21**	**68.01**	**67.57**	**60.61**	**56.58**	**73.42**

Table 3.1: **Performance comparison on the KITTI-val set.** The results are the average values of three repeated experiments.

validation samples [9]. Since our augmentation methods are applied stochastically, we report the average values of 3 repeated experiments in Table 3.1.

PointPillars [36] uses two separate networks for Car and Cyclist/Pedestrian classes. We use a batch size of 2 for Car network and 1 for Cyc/Ped network. And we train 160 epochs for Car and 80 epochs for Cyc/Ped network.[1] PV-RCNN [59] uses a single network for all classes. We train the network with batch size 8 for 80 epochs. The parameters of the proposed augmentation methods are shown in Table 3.2. The left values of '/' are parameters of the Car network, and the right values are parameters of the Cyc/Ped network. Basic data augmentations such as ground-truth oversampling [81], rotation, translation, and flipping are used before applying our partition-based augmentations. For other parameters not mentioned, the settings of each original paper are used.

[1] default parameters in https://github.com/traveller59/second.pytorch

Method	Parameters						
	p_D	p_W	p_M	C_S	p_S	C_N	p_N
Dropout	1.0/0.3	-	-	-	-	-	-
Swap	-	1.0/0.7	-	-	-	-	-
Mix	-	-	0.3/1.0	-	-	-	-
Sparse	-	-	-	40/50	0.3/0.3	-	-
Noise	-	-	-	-	-	5/10	0.3/0.1
PA-AUG	0.2/0.2	0.2/0.2	0.2/0.2	40/40	0.1/0.1	10/10	0.1/0.1

Table 3.2: **Parameters used in the KITTI experiments.**

Results

Table 3.1 shows the effect of each partition-based augmentation methods and PA-AUG. Precision and recall curves are computed using 11 points. All the proposed standalone augmentation methods performed better than the baseline algorithms without our data augmentation (PointPillars [36] and PV-RCNN [59]) in most cases. We have found that the cases in which each operation significantly increases are different. For example, Dropout does not improve the Easy score of Car a lot, but it does for Mod. and Hard cases. Other operations, on the contrary, increase the Easy score a lot compared to Mod. and Hard scores. For the Cyc/Ped network, Mix operation achieves the highest scores for Pedestrian class, but scores for Cyclist class are remarkably low. Interestingly, PA-AUG achieves the highest performance improvement on average through even improvements for all scores, which means the proposed partition-based augmentations have synergy effects when used together. Also, PA-AUG improves all the scores of PV-RCNN [59], one of the current state-of-the-art LiDAR-based detectors.

Augmentation	Dataset			
	KITTI	KITTI-D	KITTI-S	KITTI-J
Baseline	67.35	58.91(-8.44)	56.89(-10.46)	56.66(-10.69)
+ Dropout	**68.05**	**64.09**(**-3.96**)	62.27(-5.78)	57.11(-10.94)
+ Swap	66.69	60.84(-5.85)	59.55(-7.14)	54.90(-11.79)
+ Mix	67.91	63.13(-4.78)	63.42(-4.49)	56.03(-11.88)
+ Sparse	67.59	62.73(-4.86)	62.90(-4.69)	40.02(-27.57)
+ Noise	65.99	58.67(-7.32)	59.64(-6.35)	**58.25**(**-7.74**)
+ PA-AUG	67.74	63.61(-4.13)	**64.20**(**-3.54**)	57.91(-9.83)

Table 3.3: **Robustness Test.** The 3D AP_{Hard} (IoU=0.7) on KITTI-val are reported. The values in parentheses are the performance decrease of each corrupted dataset compared to KITTI-val. The baseline model is PointPillars [36].

3.3.2 Robustness Test

Settings

We evaluate the robustness of our proposed augmentations using three corrupted KITTI-*val* datasets, KITTI-D, KITTI-S, and KITTI-J. KITTI-D (Dropout) is a dataset in which some of the foreground points are removed by dropping out a portion of all objects. For fairness, not making it the same as the dropout used for our proposed augmentation, a random dense area with many points is selected for the part to be dropped out. KITTI-S (Sparse) is a dataset that leaves only 30% of points using Farthest Point Sampling (FPS) across the point cloud. Finally, KITTI-J (Jittering) is a dataset that adds Gaussian noise $X \sim \mathcal{N}(0, 0.1^2)$ for all points. Each corrupted dataset is designed to closely simulate the actual scenario

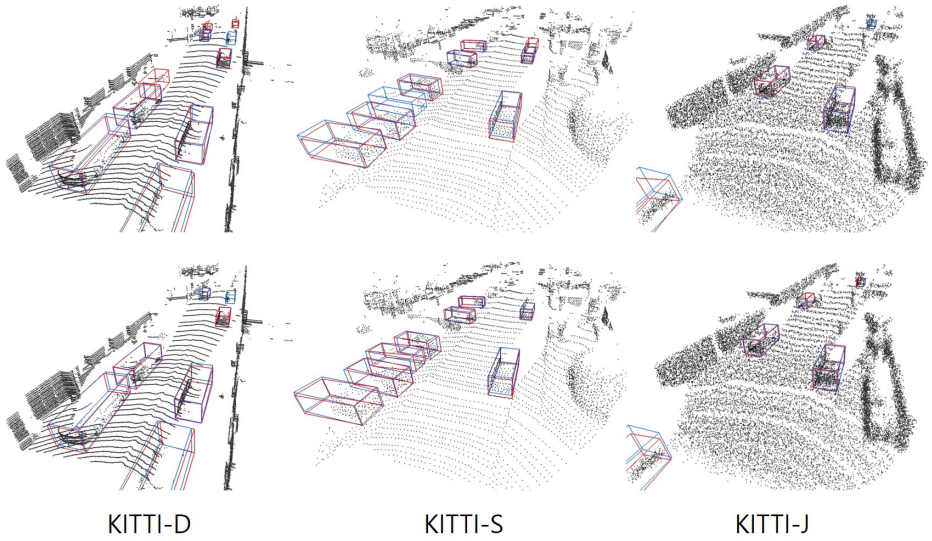


Figure 3.4: **Qualitative results on the corrupted KITTI datasets.** The upper row shows the PointPillars results, and the lower row shows the PointPillars + PA-AUG results. The ground-truth and predicted boxes are shown in blue and red, respectively.

of the cases when occlusion is severe, LiDAR has a low resolution, or LiDAR is incorrect. Some examples with detection results are shown in Figure 3.4.

Results

In Table 3.3, the 3D AP_{Hard}(IoU=0.7) scores on the KITTI-*val* and its corrupted datasets are reported. The values in parentheses are the performance decrease of each corrupted dataset compared to KITTI-*val* (leftmost). In the table, the best performance on each dataset is denoted in bold. Dropout, Swap, Mix, and Sparse operations all showed less performance decrease on the KITTI-D and KITTI-S datasets than the baseline. However, the performance decreased significantly

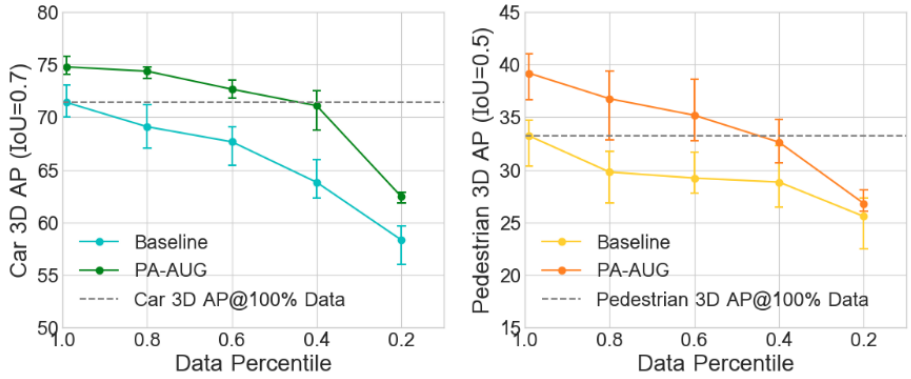


Figure 3.5: **Data Efficiency Test.** The graphs show the 3D AP_{Easy} scores on KITTI-*val* according to the size of the training data subsets. All other data augmentations except PA-AUG are not used.

on the KITTI-J dataset. On the other hand, Noise operation showed a smaller decrease than the baseline on every corrupted dataset. PA-AUG takes advantage of each operation evenly and shows the most robust performance for corrupted datasets.

3.3.3 Data Efficiency Test

We downsample the KITTI datasets, taking subsets with number of 20%, 40%, 60%, 80% training examples to verify how PA-AUG performs under very little data. Green and orange dots in Figure 3.5 show the performance of PA-AUG with the full datasets and four subsets, respectively indicating Car and Pedestrian categories. Cyan and yellow dots in Figure 3.5 show the results of the baselines. In these experiments, all other data augmentations except PA-AUG are not used to verify the effectiveness of PA-AUG alone. The results show that PA-AUG is effective not only in the full dataset, but also in data subsets. PA-

Method	KITTI-*val* 3D AP_{Easy}		
	Car@0.7	Cyclist@0.5	Pedestrian@0.5
Random	80.64	62.42	55.60
Part-aware	**83.70**	**70.88**	**57.38**

Table 3.4: **Comparison of partitioning methods.**

# Partitions	KITTI-*val* 3D AP_{Easy}		
	Car@0.7	Cyclist@0.5	Pedestrian@0.5
2	80.85	68.81	57.14
4	80.67	**70.88**	**57.38**
8	**83.70**	67.87	52.94

Table 3.5: **Comparison of the number of partitions.**

AUG using only 40% of examples achieves 3D AP comparable with the baselines using full datasets in both Car and Pedestrian. That is, PA-AUG is about 2.5$\times$ more data-efficient.

We notice that the performance drop in PA-AUG is steeper than the baseline as the size of the datasets decreases. This phenomenon is due to the relative lack of information of original objects in PA-AUG since modified and augmented datasets are provided where the original data itself is highly insufficient. The performance drop may slow down when smaller augmentation parameters are applied. Even so, PA-AUG shows the higher performances in full datasets and all subsets than the baseline since the improvement is much more significant.

3.3.4 Ablation Study

Partitioning Method

To verify the need for the part-aware partitioning method, we randomly create partitions without part information. The random partitions are created with the same number and the same direction as the part-aware partitions, but the scales and positions are randomly generated for each object. We apply the proposed partition-based augmentations equally to the random partitions and the part-aware partitions. As shown in Table 3.4, random partition-based augmentation has significantly less performance improvement than part-aware partition for all classes. From this result, it can be seen that the part information plays a critical role in applying the proposed partition-based augmentations.

The Number of Partitions

Since we roughly know the location of the characteristic sub-parts for each class, we defined a different number of partitions for each class using this prior knowledge. In order to check whether the number of partitions defined is actually the most effective, we experiment varying the number of partitions. As shown in Table 3.5, the best performance can be achieved by using 8, 4 and 4 partitions for Car, Cyclist and Pedestrian classes. From these results, it can be seen that if the number of partitions is too large or too small, the effect on the original data becomes too small or too large, thus weakening the regularization effect and reducing the performance.

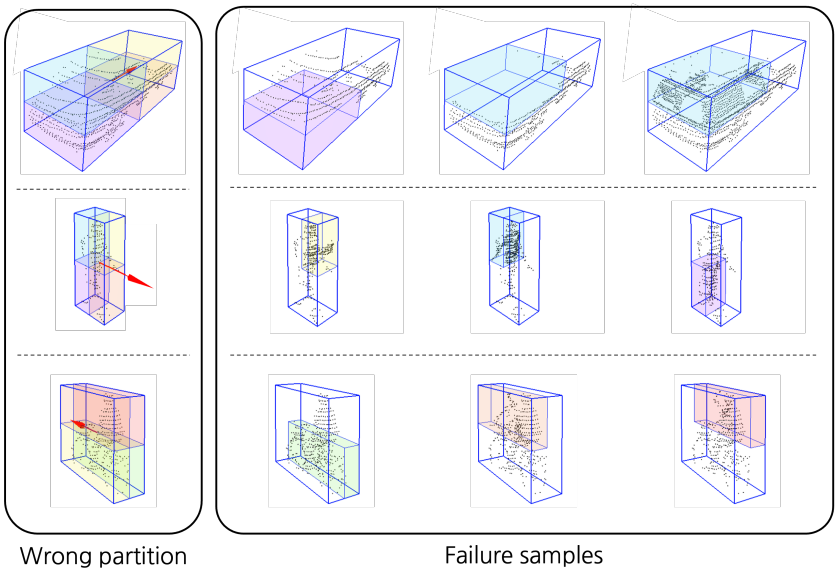


Figure 3.6: **Failure samples of the wrong partitioning methods.**

3.4 Discussion

We have defined part-aware partitioning in Section 3.2.1. It divides Car, Pedestrian, and Cyclist into 8, 4 (front view), and 4 (side view) partitions with different views. It may seem quite confusing and heuristic but there are two fundamental rules for it: 1) the partitions should divide characteristic sub-parts, *i.e.,* wheels and body for Car, arms and legs for Pedestrian, wheels and human for Cyclist, 2) the partitions should contain only a single characteristic sub-part. Applying the proposed PA-AUG using wrong partitioning methods which violate the rules, would generate inappropriate data samples.

Figure 3.6 shows some wrong partitioning examples and their failure samples. If we divide Car with 4 (side view) partitions, the lower partitions contain two wheels simultaneously, which violates rule 2. Thus, it can overly dropout

important sub-parts or swap almost the whole upper body. If Pedestrian and Cyclist are divided into 4 (side view) and 4 (front view) partitions, the partitions do not contain the characteristic sub-parts, which violates rule 1. Therefore, they can generate Pedestrian samples with 3 or 4 arms and legs or Cyclist samples with two humans and 3 or 4 wheels. These failure samples would hamper detection performance.

3.5 Conclusion

We have presented PA-AUG which makes better use of 3D information of point clouds than the conventional methods. We divide the objects into 8 or 4 partitions according to intra-object part location and apply five separate augmentation methods which can be used simultaneously in a partition-based way. The proposed data augmentation methods can be universally applied to any architecture, and PA-AUG further improves one of the SOTA detectors on KITTI dataset. Experimental results show that PA-AUG can improve robustness to corrupted data and enhance data efficiency. Because of the generality of the proposed methods, we believe that it can be used in any tasks utilizing 3D point clouds such as semantic segmentation and object tracking. However, there are some limitations in applying PA-AUG to other 3D datasets. The bounding box must be oriented and objects must not overlap too much. These limitations make it difficult to apply our method to indoor datasets such as ScanNet [65, 1, 17].

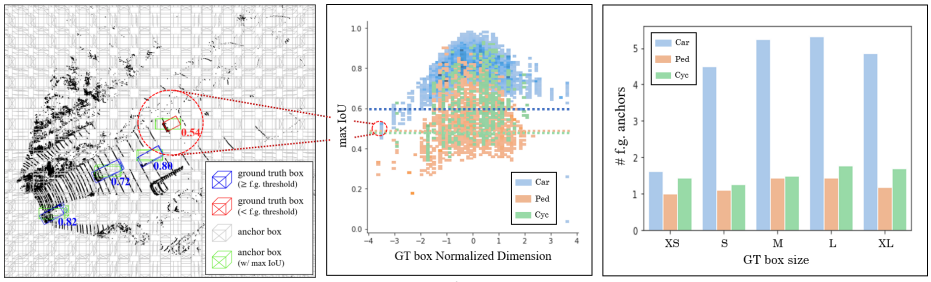
Chapter 4

Mixture-Density-based 3D Object Detection (MD3D)

4.1 Introduction

Raw point clouds obtained by LiDAR sensors have noisy sparse representation with an imbalance sampling problem, which causes many occluded surfaces to be without any points. Therefore, methods have been devised to compensate for the weaknesses of point clouds at various levels. Several methods have been proposed to enhance the performance of object detectors at the input level [16, 79]. At the feature level, there are various representation forms, such as raw points [55, 56, 61, 49], voxels [81, 93, 35, 18], graphs [63], and their hybrids [59, 30, 47]. Two streams of works based on the presence or absence of anchor boxes have been proposed at a higher level. The former is known as an *anchor-based* method [36, 81, 59, 18, 83, 30], and the latter is an *anchor-free* method [61, 85, 23, 32, 7, 39].

The existing 3D object detectors have mostly adopted anchor-based detection methods. In this study, we present the anchor or anchor box as a set of predefined 3D boxes for each object class by using the scale and aspect ra-



(a) (b) (c)

Figure 4.1: **Problems caused by hand-crafting factors of anchor boxes.** (a) shows the BEV of a point cloud with the equally spaced anchors and ground truth (GT) boxes of the car class. The values under each GT represent the maximum IoU with anchors. (b) shows the IoU value with the best-matched anchor box for all GT boxes in KITTI-*train* data. The farther the size of the GT box is from the mean size (0), the lower the maximum IoU of the box. Dotted lines represent foreground IoU thresholds for each class. If the maximum IoU of a GT box does not exceed the threshold, at least one anchor box with the highest IoU value is forced to be assigned for each GT. (c) the average number of assigned anchors for each GT box size group. It shows how the number of anchors assigned to foregrounds varies by class and size.

tio of the class. Anchors are tiled across the scene (gray boxes in Figure 4.1 (a)), and anchor-based detectors detect target objects by predicting offsets from nearby anchors for each class. However, anchor-based detectors suffer from a fatal problem; they must predefine many anchor-related hyper-parameters: 1) anchor size, 2) anchor direction, 3) stride that determines anchor placement, and 4) target assignment policy. Each of these factors dominantly influences the performance and latency of the detectors. Thus, they must be defined separately for each class.

Existing anchor-based methods usually set their default anchor size as the average of all ground truth (GT) bounding boxes with 0 and 90 degrees rotations. Unifying the anchor size as a representative value, *i.e.,* the average size of objects in the training data, would be a good compromise in the trade-off between speed and performance. However, objects with significant deviations from the default anchor size inevitably tend to be overlooked. Another hyper-parameter that biases a detector is the stride between neighboring anchors. The spatial dimensions of the last feature map determine the anchor stride. For example, in the KITTI dataset [24], a typical compact detector has a 200×176 feature map; the anchor-to-anchor interval is about 0.4m, which is relatively narrow for large objects such as *Car*. Contrastingly, for small objects, such as *Pedestrian* and *Cyclist*, it is too wide to cover all GTs. Accordingly, the IoU value between an anchor and a GT varies significantly for each object class, and hence the foreground threshold is inevitably set differently for each class. In addition, calculating the IoU in a 3D space can result in extremely low IoU values, leading to multiple placements of anchors on the z-axis (vertical direction). To mitigate this problem, conventional methods ignore the z-axis using Bird's Eye View (BEV) 2D IoU.

Figure 4.1 shows the problems caused by hand-crafting factors in the anchor-based detection methods in detail. Notably, the average number of assigned anchors for Car with an extreme size (XS in (c)) is much smaller than the average-sized Car. Consequently, outlier objects with extreme sizes show low recognition rates compared to objects within the normal range because of an insufficient number of assigned anchors. Moreover, regardless of the GT box size, the number of foreground anchors for Car is overwhelmingly larger than those for the other two classes. This is because the strides of the anchors cannot be assigned

differently for each class. A relatively larger stride for Pedestrian and Cyclist is likely to cause the objects belonging to these classes to be unable to match with any of the anchor boxes. Therefore, an unsuitable anchor box with a low IoU value, which is also the highest among the other anchor boxes, is forcibly assigned to avoid the zero assignment. Through this, it can be pointed out that it is not the number of GT boxes or other biases in the training data but the inherent structural limitations of the anchor-based detection methods that severely affect the inferior detection performance of Pedestrian and Cyclist.

Our proposed Mixture Density network for 3D Object Detector (MD3D) is a method of estimating the distribution of 3D bounding boxes in point clouds with a Gaussian Mixture Model (GMM), which is free from the problems experienced by anchor-based detectors mentioned above. Another significant merit of the MD3D is that it is free from the discrepancy between classification and regression loss. Most 3D object detectors use the focal loss [43] for a classification loss to adjust their weights according to the estimation accuracy, allowing them to learn well about data with fewer samples. By contrast, regression loss treats the anchors assigned as the foreground equally. Therefore, with typical regression loss, it is inevitable that classes whose GT box shapes are concentrated close to the mean, *i.e.,* Cars in the KITTI dataset, have superior performance. Our MD3D estimates the 3D bounding box in a distribution form throughout the scenes without any process of assigning the GT during regression learning and without distinguishing classes or box sizes. Thus, it can reduce heuristic design factors and cover a wider variety of data samples. The contributions of this study are as follows.

- Among point-cloud-based 3D object detectors, we first propose an anchor-

free detection method that estimates the density of bounding boxes and no longer requires a heuristic ground truth assignment.

- Our proposed MD3D is applicable to any type of point cloud feature encoding methods that enables it to be plugged and played easily to the existing detectors.

- MD3D shows superior performance and latency compared to the existing detection heads and facilitates learning by minimizing hand-crafted design factors.

4.2 Methods

4.2.1 Modeling Point-cloud-based 3D Object Detection with Mixture Density Network

In point cloud-based 3D object detection, the input point cloud can be expressed as $L \in \mathbb{R}^{N\times(3+c)}$ (3D coordinates and point cloud features), and the position, size, and direction of an object can be expressed as a 3D bounding box $B \in \mathbb{R}^{N_{gt}\times 7}$. Here, N represents the number of points in the scene, and N_{gt} is the number of GT boxes. To regress object B from input L, we estimate the conditional probability distribution $p(B|L)$.

A mixture density network (MDN) [4] is a neural network, whose target is to learn the probability density function (pdf). We applied MDN to point cloud-based 3D object detection to predict the distribution of multiple bounding boxes for a given scene (point cloud), and estimate the target 3D bounding box B for the input point cloud L as a mixture model. We use the conventional GMM as the target pdf, which can be expressed as:

$$p(B|L) = \sum_{k=1}^{K} \phi_k \times \mathcal{N}(B|\mu_k, \Sigma_k) \tag{4.1}$$

$$\mathcal{N}(B|\mu_k, \Sigma_k) = \frac{\exp\left(-\frac{1}{2}(B-\mu_k)^\top \Sigma_k^{-1}(B-\mu_k)\right)}{\sqrt{(2\pi)^7|\Sigma_k|}} \tag{4.2}$$

where K is the number of mixture components, which is determined by the spatial resolution of the BEV feature or the number of point features N, and ϕ_k is the mixing coefficient. For the efficiency of the model, we assume that each element of $\mu_k \in \mathbb{R}^7$ is independent and the covariance matrix is diagonal, that is, $\Sigma_k = \text{diag}(\sigma_k^2)$ where $\sigma_k^2 \in \mathbb{R}^7$, rather than dealing with a full covariance matrix $\Sigma_k \in \mathbb{R}^{7\times7}$.

B is composed of the center position, box dimension, and yaw angle, so $B_{origin} = \{x_c, y_c, z_c, l, w, h, \theta\}$. We encode the B_{origin} as $B_{corner} = \{\boldsymbol{C}_{flt}, \boldsymbol{C}_{brb}, w\} \in \mathbb{R}^7$, which consists of the front-left-top corner $\boldsymbol{C}_{flt} = \{x, y, z\}_{flt}$, the back-right-bottom corner $\boldsymbol{C}_{brb} = \{x, y, z\}_{brb}$, and width w. Among the various ways to encode bounding box B, encoding it with two opposite corners and width can result in a more accurate regression for the bounding box. This can be attributed to the nature of the point clouds obtained by LiDAR, in which the points are not in the center of an object but are concentrated in one corner. The corner on the hindside without points can be easily regressed using peripheral point features owing to the symmetry of the target object. As part of the post-processing, we decode the B_{corner} back to the B_{origin}.

Existing anchor-based regression methods learn several B's separately in L, where each anchor's design and matching algorithm become critical elements in training. However, because our method learns by representing the distribution of multiple B's as one mixture model with the conditional distribution $p(B|L)$,

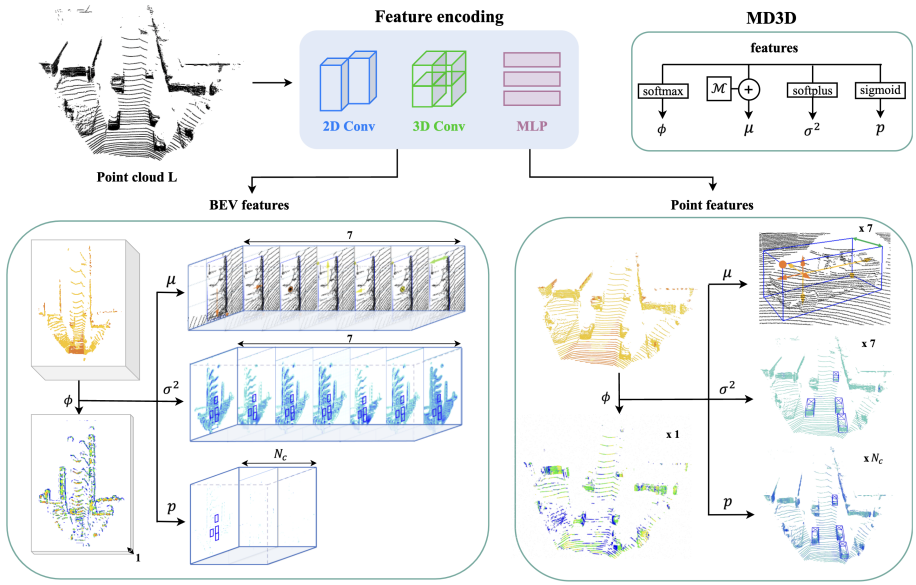


Figure 4.2: **The overall architecture of MD3D.** Regardless of the encoding types and feature forms, MD3D is applicable to existing detectors in a plug-and-play manner. MD3D predicts the mixture parameters ϕ, μ, and σ^2 from a regression branch which compose the distribution of multiple bounding boxes for a given point cloud. A classification branch predicts class probability p for each class $c \in [N_c]$.

unnecessary heuristic design can be eliminated.

4.2.2 Network Architecture

The MD3D consists of a regression branch that predicts three mixture parameters ϕ_k, μ_k, and σ_k^2 for $k \in [K]$, and a classification branch that predicts class probability p. The backbone of the existing 3D object detectors, which encodes the feature of a point cloud, remains unchanged. We apply the MD3D to most commonly used forms of head features, BEV-type features, and point-type fea-

tures. Their structures are shown in Figure 4.2; MD3D can be applied to any form and is compatible with many different methods of encoding point cloud features.

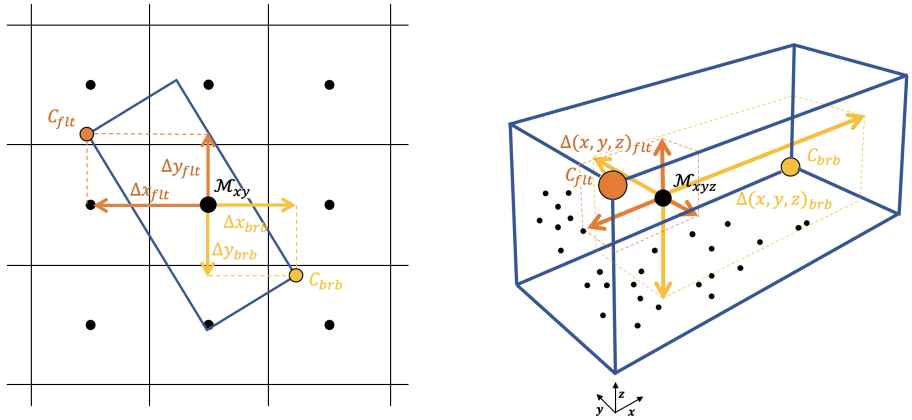
The MD3D for BEV features has a structure similar to that of MDOD [87], an MDN-based 2D object detector. The BEV feature has the shape of $H \times W \times C$, where H, W, and C represents the height, width, and number of channels, respectively. Accordingly, the number of mixture components K becomes $H \times W$, and the mixing coefficient $\phi \in \mathbb{R}^{H \times W \times 1}$ is forced to satisfy $\sum_k \phi_k = 1$, using softmax for the feature output. As shown in Figure 4.3 (a), $\mu \in \mathbb{R}^{H \times W \times 7}$ does not predict B_{corner} directly but predicts the offsets from the center coordinates of each feature $\mathcal{M}_{xy}$. For z and w, we use the raw output rather than the offset. The process is formulated as follows:

$$\begin{aligned} \mu_{BEV} = (&\mathcal{M}_x + \Delta x_{flt}, \mathcal{M}_y + \Delta y_{flt}, z_{flt}, \\ &\mathcal{M}_x + \Delta x_{brb}, \mathcal{M}_y + \Delta y_{brb}, z_{brb}, w). \end{aligned} \tag{4.3}$$

$\sigma^2 \in \mathbb{R}^{H \times W \times 7}$ predicts values greater than zero using softplus activation. $p \in \mathbb{R}^{H \times W \times N_c}$ predicts the classification probability for each class using sigmoid activation, where N_c denotes the number of classes which is set to 3 (Car, Pedestrian, Cyclist) in our experiments.

Existing anchor-based regression methods use anchors defined differently per class, and generally they use anchors in the two directions of 0 and 90 degrees. Therefore, the number of output boxes is $H \times W \times N_c \times 2$, which is $N_c \times 2$ times higher than that of our MD3D. Consequently, MD3D has advantages in terms of the number of parameters, inference time, and post-processing time.

The MD3D for the point feature has some minor modifications from that of the BEV feature because the input shape is slightly different. Because the point feature has the form of $N \times C$, where N is the number of points in a



(a) BEV feature (b) Point feature

Figure 4.3: **Illustration of corner regression.** MD3D does not predict the corners ($\boldsymbol{C}_{flt}$ and $\boldsymbol{C}_{brb}$) of bounding boxes directly but rather predicts the offsets ($\Delta(x,y,z)_{flt}$ and $\Delta(x,y,z)_{brb}$) from $\mathcal{M}_{xy}$ or $\mathcal{M}_{xyz}$, the center coordinates of the BEV feature or point feature, respectively.

scene, the number of mixture components becomes $K = N$. Accordingly, it becomes $\phi \in \mathbb{R}^{N\times 1}$, $\mu \in \mathbb{R}^{N\times 7}$, $\sigma^2 \in \mathbb{R}^{N\times 7}$, and $p \in \mathbb{R}^{N\times N_c}$. Furthermore, as shown in Figure 4.3 (b), the reference point $\mathcal{M}_{xyz}$ becomes the original (x, y, z) coordinates of the point, and μ predicts an offset from $\mathcal{M}_{xyz}$, except for w:

$$\begin{aligned}\mu_{point} = (&\mathcal{M}_x + \Delta x_{flt}, \mathcal{M}_y + \Delta y_{flt}, \mathcal{M}_z + \Delta z_{flt}, \\ &\mathcal{M}_x + \Delta x_{brb}, \mathcal{M}_y + \Delta y_{brb}, \mathcal{M}_z + \Delta z_{brb}, w).\end{aligned} \tag{4.4}$$

At inference time, because the values of μ are highly likely to be close to the local maximum of the predicted GMM, we use μ of each mixture component as an independent output box. To improve the inference speed, σ^2 is not used and ϕ is used to filter out unnecessary boxes. In addition, the mixing coefficient ϕ is very low for the location where no object exists, as in the example in the Figure

4.2, hence many output boxes can be filtered out. Then, using non-maximum suppression (NMS), boxes in which p is the local maximum are finally extracted.

4.2.3 Loss function

L_{MDN}, the regression loss, is used to learn the GMM parameters ϕ, μ, and σ^2 with a negative log-likelihood as follows:

$$L_{MDN} = -\frac{1}{N_{gt}} \sum_{n=1}^{N_{gt}} \log \left(\sum_{k=1}^{K} \phi_k \times \mathcal{N}(B_n | \mu_k, \Sigma_k) \right). \tag{4.5}$$

Here, N_{gt} is the number of GT bounding boxes in the scene.

For classification loss, we use the most commonly used focal loss [43], as shown below:

$$\begin{aligned} L_{focal} &= -\alpha_t (1 - p_t)^{\gamma} log(p_t) \\ \text{where } p_t &= \begin{cases} p & \text{for foreground box} \\ 1 - p & \text{otherwise.} \end{cases} \end{aligned} \tag{4.6}$$

Among the boxes predicted in the regression branch, when the 3D IoU of a box exceeds 0.5, we assign it to the foreground; otherwise, we assign it to the background. We use $\alpha_t = 0.25$ and $\gamma = 2$.

The loss of the MD3D head is the sum of the MDN loss and focal loss, as follows:

$$L_{MD3D} = L_{MDN} + \beta \cdot L_{focal}. \tag{4.7}$$

For one-stage detectors, we use MD3D loss as a final loss, and for two-stage detectors, we replace the region proposal network (RPN) loss with MD3D loss because the MD3D head is utilized in the RPN. We use $\beta = 500$.

4.3 Experiments

4.3.1 Datasets

We evaluated the proposed method on the KITTI dataset [24], one of the most popular datasets for 3D object detection for autonomous driving. It consists of 7,481 training samples and 7,518 testing samples, where the training samples are generally divided into *train* split with 3,712 samples and *val* split with 3,769 samples. Because the KITTI dataset contains only 90-degree annotation, we clipped the scenes into (0, 70.4)m, (-40, 40)m, and (-3, 1)m for the X, Y, and Z axis ranges. We also experimented on a large-scale Waymo open dataset [68] to verify whether the performance of the MD3D improved regardless of the data size. The Waymo dataset includes 798 training sequences with approximately 160k samples and 202 validation sequences with 40k samples. Because of limited resources, we trained the models with 20% samples at regular intervals for each sequence, using a total of 32k training samples. The Waymo dataset contains a complete 360-degree annotation, and we clip the scenes into (-75.2, 75.2)m, (-75.2, 75.2)m, and (-2, 4)m for the X, Y, and Z axis ranges. We primarily focused on outdoor scene datasets whose target objects are occlusion-free in the BEV.

4.3.2 Experiment Settings

We conducted the experiments with the same factors as the existing 3D object detectors, except that the detection head was replaced with MD3D. Most of the configurations are from OpenPCDet [70], one of the most commonly used codebases for 3D object detection. The detailed network structures of each detector are shown in Tables 4.8, 4.9, and 4.10. The baseline detectors may differ slightly

Method	Car 3D AP (IoU=0.7)			Ped 3D AP (IoU=0.5)			Cyc 3D AP (IoU=0.5)			mAP
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	
Anchor-based										
PointPillars [36]	86.63	76.83	75.04	**55.86**	**50.18**	**46.45**	80.62	62.71	59.24	65.96
PointPillars+MD3D	**87.78**	**77.37**	**75.33**	53.47	44.90	42.40	**84.30**	**67.65**	**63.41**	**66.29**
SECOND [81]	88.13	78.34	77.10	55.45	51.28	47.09	80.49	65.98	61.62	67.28
SECOND+MD3D	**89.12**	**78.85**	**77.18**	**64.41**	**57.07**	**50.79**	**87.00**	**72.10**	**65.66**	**71.35**
PV-RCNN [59]	**89.31**	**83.04**	**78.76**	64.65	57.96	53.68	85.50	**71.61**	**68.14**	72.52
PV-RCNN+MD3D	89.09	81.33	78.41	**65.33**	**58.87**	**54.70**	**86.29**	71.55	67.47	**72.56**
Anchor-free										
CenterPoint [85]	85.25	77.45	76.09	56.72	52.79	49.66	82.71	67.43	62.97	67.90
CenterPoint+MD3D	**89.10**	**78.88**	**77.60**	**63.42**	**55.96**	**50.02**	**86.93**	**70.35**	**65.53**	**70.86**
PointRCNN [61]	**88.14**	**78.12**	**77.43**	66.79	61.90	56.29	86.33	71.83	**68.93**	72.86
PointRCNN+MD3D	86.75	77.17	75.65	**70.59**	**62.61**	**57.14**	**86.54**	**72.27**	68.50	**73.02**

Table 4.1: **Performance comparison on the KITTI-*val* set.** The results were evaluated by the AP with 11 recall positions, and the average values of three repeated experiments were reported for each AP.

in performance owing to the gap between the settings of the original paper. As MD3D is plugged and played with the existing detectors, the only modification in the MD3D experimental setting is the detection head for one-stage detectors, and RPN head for two-stage detectors.

4.3.3 Results on the KITTI Dataset

As shown in Table 4.1, we conducted the KITTI-*val* dataset experiment to compare the performance of the baselines, three anchor-based detectors, and two anchor-free detectors. We mark '+MD3D' when the proposed method, MD3D, is applied. We calculated the average precision (AP) by creating a precision-

recall curve along with changes to the confidence threshold and averaging the precision values at 11 recall points. The IoU thresholds for 3D AP were set to 0.7, 0.5, and 0.5 for Car, Pedestrian, and Cyclist, respectively, and mAP is the mean 3D AP score for all classes.

MD3D improves the performance of all anchor-based detectors for most classes and difficulty levels. In the case of SECOND, a significant improvement was achieved in all settings, especially in the Pedestrian and Cyclist classes. This difference in performance gain arises from the difference in the feature dimension size. Unlike PointPillars, which use 248×216 features, SECOND uses 200×176-size features. In other words, PointPillars has a lower anchor stride than SECOND, so its anchor boxes have already been excessively assigned as a foreground for even small-size GTs of Pedestrian and Cyclist, which enables sufficient learning for both classes at the cost of latency. As a result, even if MD3D was applied, a significant performance improvement would not be achieved. However, with a larger anchor stride, SECOND assigns an average of 1.4 and 1.6 anchors to Pedestrian and Cyclist, respectively, so they are not trained sufficiently. Therefore, with MD3D learning a single GMM regardless of the size of the object, SECOND+MD3D significantly improves the performance for Pedestrian and Cyclist compared to SECOND. The performance improvement of PV-RCNN is marginal because MD3D is applied only to the RPN of the first stage. Regardless of the precision, in the RPN, the recall value increases with the increase in number of proposal boxes. However, as can be seen in Figure 4.7, MD3D is more effective at removing false positives than the existing head; therefore, the performance improvement of an RPN head is insufficient.

The MD3D also shows superior performance compared to anchor-free detection heads. Compared to CenterPoint [85], which uses a heatmap-based de-

tection head and regresses the bounding box with center coordinates, MD3D significantly improves the performance for all classes. This performance improvement is inherently attributed to MD3D's effective learning for small classes, such as Pedestrian and Cyclist, in addition to the change of box encoding scheme from center to corner. The Gaussian radius of the heatmap depends on the size of the GT box; therefore, small objects are not sufficiently trained as large ones. For another anchor-free detector, PointRCNN [61], which is a two-stage detector that utilizes point features, we replace only the regression branch with MD3D while leaving the classification branch that performs foreground segmentation in the RPN as it is. With this modification, the mAP is increased slightly, and the latency was significantly reduced. This shows that our MDN-based corner regression offers an advantage over PointRCNN's bin-based residual regression. For both the training and inference phases, we keep the top 512 proposals for refinement of the stage-2 sub-network.

4.3.4 Latency of Each Module

We replace the detection heads of each baseline detector with MD3D maintaining other modules unchanged. The detailed detection head structures are shown in Table 4.8. Replacing the head with ours affects not only the head module's latency but also the following modules' latency. Thus we measure the latency of each module, and the results are shown in Table 4.2. Latency was measured using Titan RTX with a batch size of 1.

'Backbone' includes the pre-processing time and the inference time of the backbone network. 'Head' shows the inference time of the detection head for one-stage detectors and the RPN head for two-stage detectors. 'Refinement' shows the refinement time of proposals. 'Post' includes the post-processing

Method	Latency (ms)				
	Backbone	Head	Refinement	Post	Total
Anchor-based					
PointPillars [36]	33.6	6.6	-	10.1	50.3
PointPillars+MD3D	33.6	**5.3**	-	**8.1**	**47.0**
SECOND [81]	19.2	1.2	-	1.5	21.9
SECOND+MD3D	19.2	**1.1**	-	**1.2**	**21.5**
PV-RCNN [59]	68.8	1.2	17.5	2.6	90.1
PV-RCNN+MD3D	68.8	**1.1**	**17.4**	2.6	**89.9**
Anchor-free					
CenterPoint [85]	23.6	1.5	-	3.0	28.1
CenterPoint+MD3D	23.6	**1.1**	-	**2.0**	**26.7**
PointRCNN [61]	40.8	**1.5**	91.8	1.3	135.4
PointRCNN+MD3D	40.8	1.6	**61.9**	**1.2**	**105.5**

Table 4.2: **Latency of each module.** We replace the detection head with the proposed MD3D. Latency is measured using Titan RTX with a batch size of 1.

time, such as NMS. And 'Total' includes the whole inference time from pre-processing to post-processing.

The detection heads of PointPillars, SECOND, PV-RCNN, and CenterPoint use BEV features. And their differences in latency mostly come from the spatial dimension of their BEV feature. PointPillars uses a 248×216-size feature, and others use a 200×176. Thus, the difference in 'Head' latency is higher in PointPillars than others. MD3D filters out false positives using mixing coefficient ϕ. Therefore, the 'Post' latency decreases by reducing the number of proposals for NMS. PointRCNN uses the point feature and it predicts proposals without separating channels for each class. Thus, the 'Head' latency slightly increases but 'Refinement' latency significantly decreases because unnecessary boxes are removed using ϕ before the NMS.

Difficulty	Method	Veh AP	Veh APH	Ped AP	Ped APH	Cyc AP	Cyc APH
LEVEL_1	SECOND	67.46	66.87	57.16	47.56	55.76	54.40
	SECOND+MD3D	**69.27**	**68.51**	**63.70**	**55.80**	**67.61**	**66.22**
	Improvements	*+1.81*	*+1.64*	*+6.54*	*+8.24*	*+11.85*	*+11.82*
LEVEL_2	SECOND	59.12	58.59	49.38	41.03	53.87	52.55
	SECOND+MD3D	**60.74**	**60.05**	**54.43**	**47.58**	**65.31**	**63.97**
	Improvements	*+1.62*	*+1.46*	*+5.05*	*+6.55*	*+11.44*	*+11.42*

Table 4.3: Performance comparison on the Waymo open dataset with 202 validation sequences.

4.3.5 Results on the Waymo Open Dataset

For the Waymo dataset, we report the AP and the average precision weighted by heading (APH) of SECOND [81] with and without the MD3D head, respectively. The AP was calculated by averaging the precision values at 11 recall points identically to that of the KITTI dataset. APH is calculated similarly to AP but uses precision values weighting each true positive by heading accuracy. We evaluated the models into two difficulty levels: LEVEL 1 includes GT boxes with at least five inside points, and LEVEL 2 includes GTs with at least one inside point. As shown in Table 4.3, the proposed MD3D improved the baseline at all levels and all classes. Note that the MD3D leads to a significant gain in Pedestrian and Cyclist classes, whose GTs are relatively small. This verifies the advantages of the MD3D predicting bounding boxes in a probabilistic and anchor-free manner.

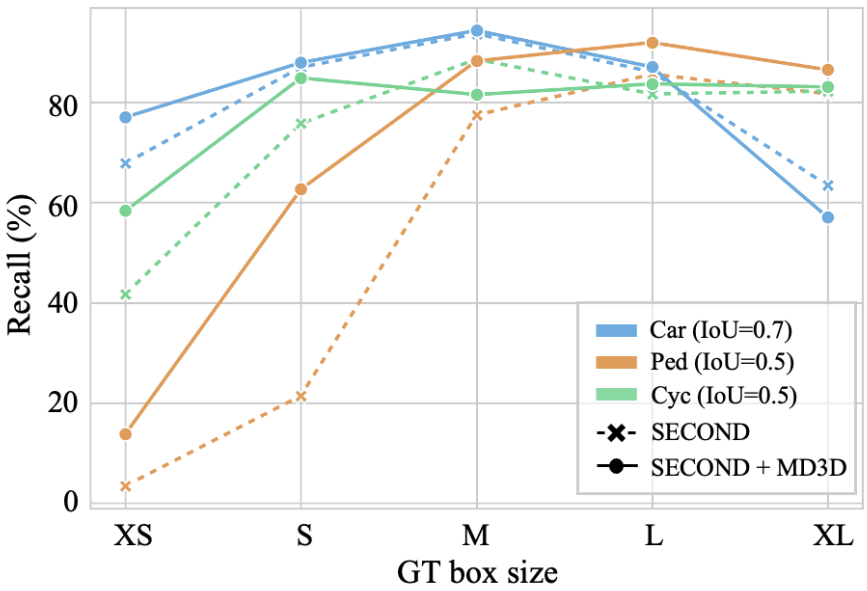


Figure 4.4: **Recalls on the KITTI-*val* set.** MD3D offers a clear advantage in predicting small objects, where only a limited number of anchor boxes are assigned to the existing detectors.

4.3.6 Analyzing Recall by object size

As shown in Figure 4.1, anchor-based detectors have an insufficient number of anchors assigned to the foreground for extremely small objects. We measured recall by object size to verify that the proposed method, not in the use of anchors, could improve this inherent limitation. We used SECOND as the base model, and considered bounding boxes before NMS to focus on the regression results. As shown in Figure 4.4, there is an improvement for XS-sized GT boxes in the Car class, where the lack of assigned anchors has caused harm to the performance of the existing detector. The improvement is insignificant for other sized Car boxes because the base model has already assigned an excessive num-

ber of anchors to the foreground. In addition, in Pedestrian and Cyclist classes, the recall of GT with a size close to the average is already high enough for the base model, so the increase is small; however, for extremely small size cases, the increase is noticeably significant. Therefore, the proposed MD3D can delicately detect small objects compared to anchor-based detection heads.

4.3.7 Ablation Study

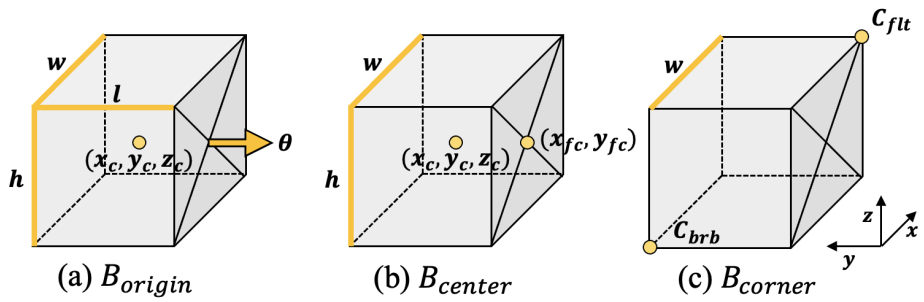


Figure 4.5: **Illustration of various box encoding methods.** The B_{origin} consists of the center coordinate (x_c, y_c, z_c), the dimension (l, w, h), and the yaw angle θ. The B_{center} consists of the center coordinate (x_c, y_c, z_c), the front-center coordinate (x_{fc}, y_{fc}), and the dimension (w, h). We use B_{corner} consisting of the front-left-top corner $\boldsymbol{C}_{flt} = (x_{flt}, y_{flt}, z_{flt})$, the back-right-bottom corner $\boldsymbol{C}_{brb} = (x_{brb}, y_{brb}, z_{brb})$, and width w.

Box encoding methods

As shown in Figure 4.5, we conducted an ablation study of the box encoding method on the KITTI validation dataset with models applying MD3D to PointPillars. To better demonstrate the performance of each box encoding method

regarding the headings of predicted boxes, we used the average orientation similarity (AOS) metric, along with 3D AP, which assesses cosine similarities between the angles of the estimated and GT heading orientations. The results are presented in Table 4.4. First, using the GT box in its original form, $B_{origin} = \{x_c, y_c, z_c, l, w, h, \theta\}$ results in a very low AOS AP and thus a low 3D AP. This is because of discontinuous θ, which is the same box when the yaw angle θ is 0 and 2π, but the loss is calculated differently. Therefore, we experimented with B_{sincos}, which changes θ to a continuous value, $(sin(\theta), cos(\theta)) \in \mathbb{R}^2$, to avoid ambiguity. Both AOS and 3D AP have some increases. Still, because $sin(\theta)$ and $cos(\theta)$ are mutually dependent and periodic, B_{sincos} is not entirely appropriate for our GMM modeling, leaving room for improvement. Therefore we devised a novel method of finding the front-center coordinate value of the box, as shown in (b), to predict the box without θ. This $B_{center} = \{x_c, y_c, z_c, x_{fc}, y_{fc}, w, h\}$ has a notable improvement over the previous two approaches. In addition, another variant method, B_{corner}, predicting two corners with w, yields the highest performance. This is because it is easy to localize the corner of an object owing to the characteristics of the point clouds obtained by LiDAR.

Probability distributions

In MD3D, it is essential to choose a proper probability density function that fits the data characteristics of the input point clouds and the output 3D bounding boxes, because it substantially impacts the model's performance. We consider the Laplace, Cauchy, and Gaussian distributions, which are symmetric and have the same number of parameters. We applied them to PointPillars and SECOND baselines and compared them to the KITTI-*val* dataset. As shown in Figure 4.6, the shapes of the three distributions differ in peak height and tail length. The

Box encoding	AOS AP_{Hard}		3D AP_{Hard}	
	Ped	Cyc	Ped	Cyc
B_{origin}	31.89	64.29	32.70	59.91
B_{sincos}	31.69	67.03	36.90	61.93
B_{center}	**39.51**	68.02	41.88	62.61
B_{corner}	39.31	**71.13**	**42.40**	**63.41**

Table 4.4: Comparison of box encoding methods.

PDF form	KITTI-*val* 3D mAP	
	PointPillars+MD3D	SECOND+MD3D
Laplace	63.98	70.47
Cauchy	64.91	70.75
Gaussian	**66.29**	**71.35**

Table 4.5: Comparison of probability distributions.

point clouds have sparse representations, implying that the area occupied by actual points is very small compared with the space of the entire area when voxelized. This makes our MD3D have a considerable number of mixture components because MD3D requires output for the entire feature space. Therefore, the Gaussian distribution with the shortest tail is the most suitable for MD3D because it can effectively suppress the probability of boxes with high uncertainty from unnecessary empty spaces. Table 4.5 also shows that the Gaussian distribution was the most effective for both models.

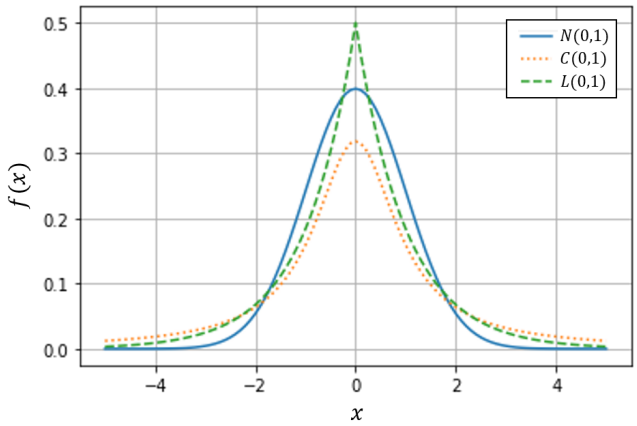


Figure 4.6: Pdfs of the Gaussian, Cauchy, and Laplace distributions.

The number of Mixture components

Table 4.1 shows that MD3D has a significant performance improvement for detectors with smaller input feature dimension and therefore fewer anchors. To verify the effect of the number of anchors and the number of mixture components K, we set SECOND as the base model and compared the performance by adjusting the horizontal and vertical dimensions of the input feature by 1/4, 1/2, and 2 times, respectively. As shown in Table 4.6, anchors are placed separately in two directions, 0 and 90 degrees for each class. Thereby, anchor-based detectors output anchor boxes six times more than K, even for features of the same size. The anchor-based detector with more anchors achieves higher performance in obtaining better foreground GT assignments, whereas MD3D with an unnecessarily large K tends to learn poorly and achieve lower performance. However, comparing them in a small feature dimension of 50×44, SECOND achieves a very low mAP (50.10) despite predicting six times more boxes than MD3D (64.86). SECOND needs to predict 200×176×6 boxes, 96 times more

Method	# anchors	KITTI-*val* 3D AP_{mean}			mAP
	K	Car	Ped	Cyc	
SECOND	50×44×6	66.12	29.91	54.27	50.10
	100×88×6	78.78	33.37	62.51	58.22
	200×176×6	**81.19**	51.27	69.36	67.28
	400×352×6	80.28	**63.58**	**72.98**	**72.28**
SECOND +MD3D	50×44	72.94	49.85	71.78	64.86*(+14.76)*
	100×88	81.10	55.05	74.45	70.20*(+11.98)*
	200×176	**81.72**	57.42	**74.92**	**71.35***(+ 4.08)*
	400×352	81.07	**57.46**	74.46	71.00*(- 1.28)*

Table 4.6: Effect of the number of mixture components.

boxes than MD3D, to achieve similar performance (67.28). Therefore, the performance of MD3D is maximized when used in compact detectors.

Covariance matrix

We modeled the point cloud-based 3D object detection with the multivariate GMM using only the diagonal elements of a covariance matrix rather than a full matrix. Training with a full covariance matrix means that a detector learns the correlation of the elements of $B_{corner} \in \mathbb{R}^7$, whereas the diagonal matrix does not. Table 4.7 presents the results of applying these two methods to PointPillars. In the case of Pedestrian, whose intraclass correlation is high because the objects share similar shapes, the full covariance model achieves higher performance. Except for Pedestrian, the models using only the diagonal matrix outperformed

Covariance matrix	KITTI-*val* 3D AP_{mean}			mAP	MD3D params
	Car	Ped	Cyc		
Full	75.06	**53.12**	68.35	65.51	15,018
Diagonal	**80.16**	46.92	**71.79**	**66.29**	**6,930**

Table 4.7: **Effect of constraint on the covariance matrix.** Assuming independence between elements achieves relatively efficient and effective results.

those using the entire matrix. Therefore, we decided to use only the diagonal elements of the covariance matrix because it achieves a slightly better mAP and uses half the number of parameters.

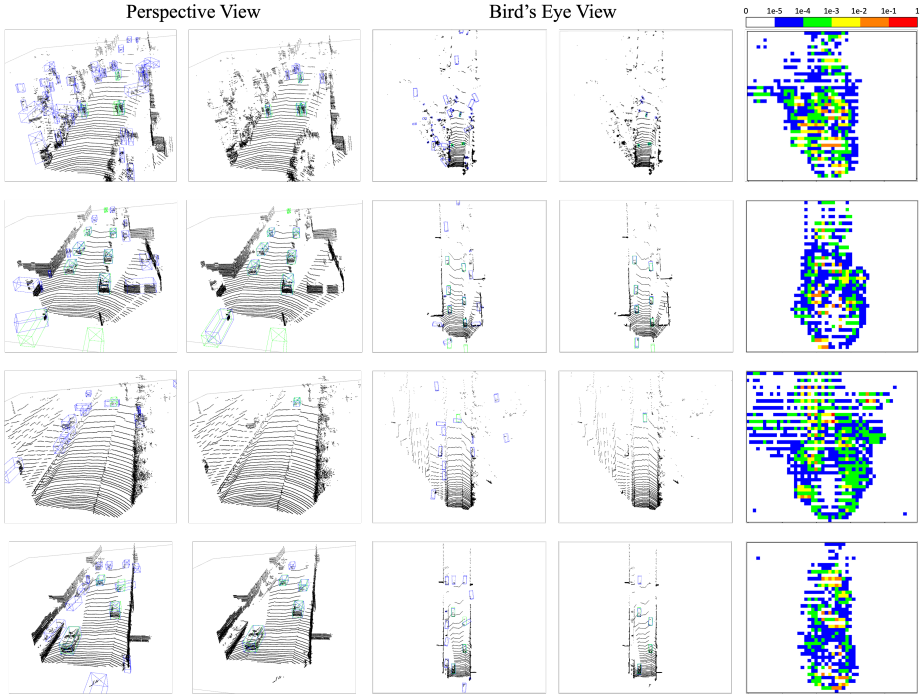
4.3.8 Discussion

MD3D showed superior performance and latency regardless of the backbone network types and the use of anchors (Tables 4.1 and 4.3). It is especially effective for one-stage detectors, such as SECOND and CenterPoint, which output relatively small feature maps (Table 4.6). The reason for their dramatic increase in performance is the higher recall than that of existing heads for small objects, such as Pedestrian and Cyclist (Figure 4.4). However, MD3D has an advantage in terms of recall rather than precision (Figure 4.7), the performance improvement for two-stage detectors, such as PV-RCNN and PointRCNN is marginal. In addition, MD3D has advantages regarding the number of parameters and latency because the box prediction channels are not separated by class. However, because of the unified channel across classes, the performance on datasets with many classes may be limited, which we will attempt to overcome in the future

work.

4.4 Conclusion

Most of the existing point cloud-based 3D object detectors apply a specific target assignment policy to the GT boxes to regress 3D bounding boxes. Because this training method needs to optimize many hand-crafted design factors, it takes significant amount of effort to utilize and places many restrictions on the network structure. In this chapter, we proposed MD3D, which reformulates the regression of 3D bounding boxes in point clouds as a density estimation problem. The MD3D is easy to use and can be applied to various types of feature encoding methods without considering the target assignment policy and network structure. Experiments on the KITTI and Waymo datasets show that the proposed method outperforms conventional methods in terms of performance, speed, ease of use, and flexibility. Although we only considered point clouds as inputs, the MD3D can be easily applied to other various types of inputs. Furthermore, we expect MD3D is utilized for multi-modal inputs by fusing the mixture density outputs.



(a) SECOND (b) MD3D (c) SECOND (d) MD3D (e) mixing coeff. ϕ

Figure 4.7: **Qualitative results for the KITTI-*val* set.** Green boxes indicate GT boxes, and blue boxes indicate predicted boxes with the confidence of over 0.1. MD3D improves the existing detector regarding precision and recall, whereas the mixing coefficient ϕ efficiently filters out unlikely boxes. We filtered out the predictions with $\frac{\phi}{max(\phi)} < 0.001$ before the NMS. We used SECOND with a 50×44 feature as the base model for this comparison.

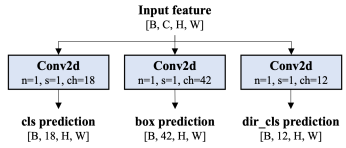
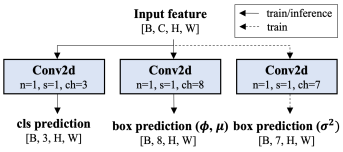
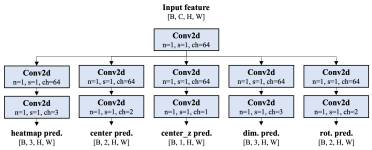
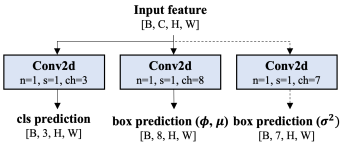
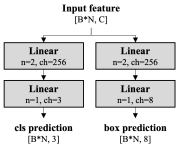
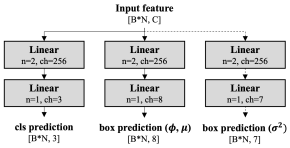
Method	Detection head	
	Original	MD3D
PointPillars SECOND PV-RCNN		
CenterPoint		
PointRCNN		

Table 4.8: **Detection heads.** The structures and parameters of the original heads are the default settings of OpenPCDet [70]. The structures and parameters of the MD3D BEV-heads (PointPillars, SECOND, PV-RCNN, and CenterPoint) follow the simplest original BEV head and the MD3D point-head (PointRCNN) follows the original PointRCNN head. σ^2 (dotted line) is not used in the inference phase.

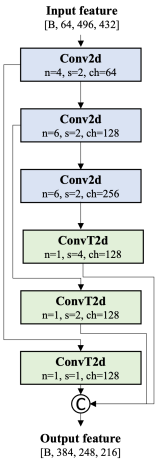
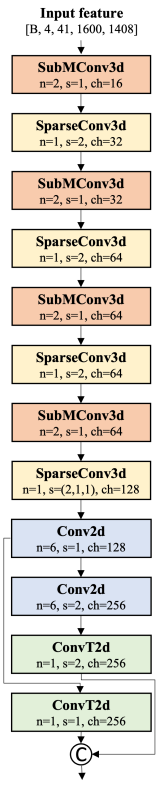
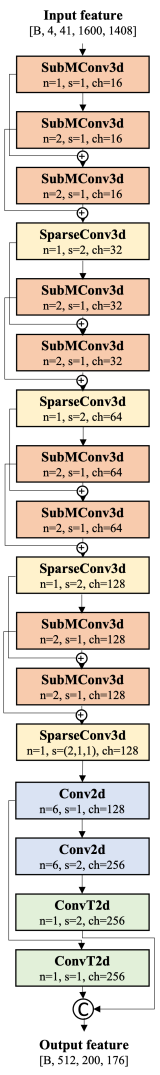
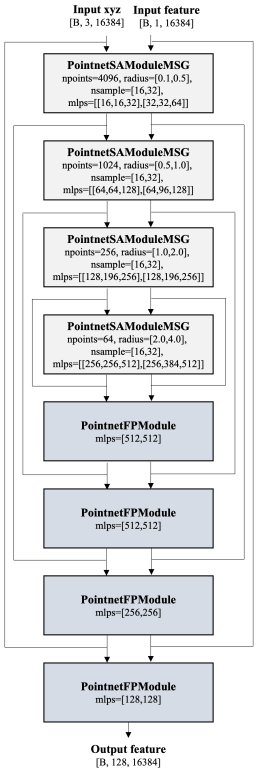
Method	PointPillars	PV-RCNN	CenterPoint	PointRCNN
Backbone Network				

Table 4.9: **Backbone network architectures (1).** Every model used the default settings of OpenPCDet [70].

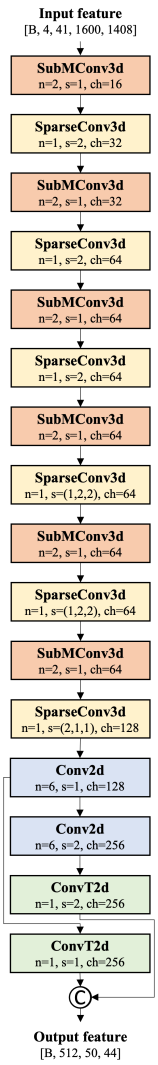
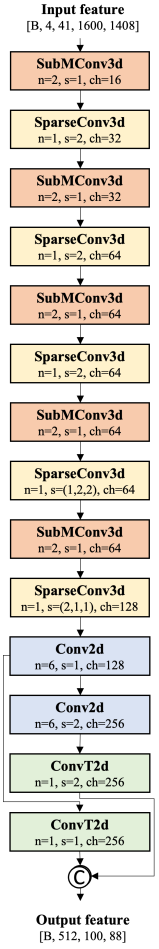
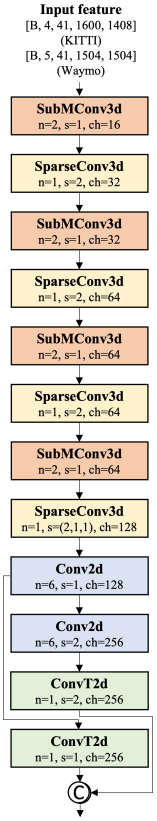
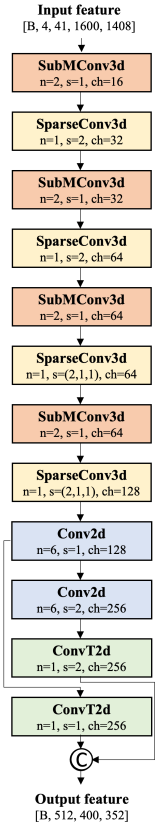
Method	SECOND (50×44)	SECOND (100×88)	SECOND (200×176)	SECOND (400×352)
Backbone Network				

Table 4.10: **Backbone network architectures (2).** The backbone structures of SECOND(50×44, 100×88, and 400×352) were designed by adding some sparse convolution blocks or changing the stride parameters of the default structure of SECOND(200×176).

Chapter 5

Combination of PA-AUG and MD3D

We have proposed two novel methods for improving LiDAR-based 3D object detection: PA-AUG (Chapter 3) and MD3D (Chapter 4). PA-AUG makes 3D object detectors robust to various extreme environments by using structural information of 3D ground-truth boxes. Also, the detectors can be aware of intra-object relation as it learns individual variation in an intra-object part. MD3D is an anchor-free detection method estimating the distribution of 3D bounding boxes in point clouds with a Gaussian Mixture Model (GMM). It is free from the problems experienced by anchor-based detectors. Thus, it improves the detection accuracy and latency of conventional detectors. In this chapter, we show the experimental results of the combination of PA-AUG and MD3D to verify their effectiveness when they are applied together.

5.1 Methods

PA-AUG is a data augmentation method, and MD3D is a detection head, which means they are independent of each other. Therefore, they can be applied to-

gether to an object detector without modifying any structure or hyper-parameters.

The order of the data augmentations is important. So we keep the conventional data augmentations as they are and apply PA-AUG right after them. The hyper-parameters for PA-AUG remain unchanged, which is shown in Table 3.2.

MD3D is applicable to any type of point cloud feature encoding method that allows it to be plugged and played easily to the existing detectors. Thus, the structure and hyper-parameters for MD3D remain unchanged, which is shown in Table 4.8.

5.2 Experiments

5.2.1 Settings

We choose one anchor-based 3D object detector, SECOND [81], and one anchor-free 3D object detector, CenterPoint [85], as baseline models. We use the baseline code from OpenPCDet [70], one of the most widely used codebases. The two baseline models use the same data augmentation methods: gt-sampling, flipping, rotating, and scaling. Thus, PA-AUG is applied right after scaling. We mark '+ PA-AUG' for this test in Tables 5.1 and 5.2. MD3D replaces the original detection head and loss of SECOND and CenterPoint. And we mark '+ MD3D' for this variant model. We evaluated each method three times and calculated the average APs. The repeated tests are represented in #1, #2, and #3.

We evaluated the proposed methods on the KITTI dataset [24], one of the most popular datasets for 3D object detection for autonomous driving. It consists of 7,481 training samples and 7,518 testing samples, where the training samples are generally divided into *train* split with 3,712 samples and *val* split with 3,769 samples. Because the KITTI dataset contains only 90-degree anno-

tation, we clipped the scenes into (0, 70.4)m, (-40, 40)m, and (-3, 1)m for the X, Y, and Z axis ranges.

5.2.2 Results on the KITTI Dataset

The experimental results of PA-AUG, MD3D, and their combination on SECOND [81] are shown in Table 5.1. PA-AUG improves the mAP of SECOND by 0.93. The most noticeable improvements occur in the Pedestrian class. MD3D significantly increases the mAP by 4.08. It considerably improves the APs of Pedestrian and Cyclist classes. And the combination of the two methods improves the mAP by 4.92, which is the best performance among the variants of SECOND. The big leap in performance comes from the improvements in Pedestrian and Cyclist classes. The increase in Car class is not significant.

Table 5.2 shows the experimental results of PA-AUG, MD3D, and their combination on CenterPoint [85]. PA-AUG improves the mAP of CenterPoint by 1.29. Similar to the results on SECOND, most of the improvements come from Pedestrian. MD3D increases the mAP by 2.97. It shows high performance improvements for all classes. The combination of the two methods improves the mAP by 3.40. Compared to the results on SECOND, it showed relatively even increases for all classes.

Experiments	Car 3D AP (IoU=0.7)			Ped 3D AP (IoU=0.5)			Cyc 3D AP (IoU=0.5)			mAP
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	
SECOND [81]										
#1	88.68	78.50	77.23	54.79	50.12	46.68	80.35	65.05	61.12	66.95
#2	87.73	78.18	76.94	56.26	52.58	48.13	78.73	65.38	60.22	67.13
#3	87.99	78.33	77.13	55.31	51.15	46.45	82.37	67.51	63.51	67.75
Average	**88.13**	**78.34**	**77.10**	**55.45**	**51.28**	**47.09**	**80.49**	**65.98**	**61.62**	**67.28**
SECOND + PA-AUG										
#1	87.80	78.33	77.19	57.67	53.98	49.39	80.38	66.05	61.76	68.06
#2	88.58	78.60	77.46	57.86	54.16	49.69	79.41	63.65	61.31	67.86
#3	88.20	78.20	76.86	59.96	55.45	51.31	79.78	65.92	62.51	68.69
Average	**88.19**	**78.38**	**77.17**	**58.49**	**54.53**	**50.13**	**79.85**	**65.21**	**61.86**	**68.20**
Improvements	*+0.06*	*+0.04*	*+0.07*	*+3.04*	*+3.25*	*+3.04*	*-0.63*	*-0.77*	*+0.25*	*+0.93*
SECOND + MD3D										
#1	89.13	78.78	76.99	64.07	56.79	50.37	87.45	72.19	65.72	71.28
#2	88.92	78.97	77.46	64.70	57.28	50.97	86.46	72.71	66.18	71.52
#3	89.30	78.81	77.10	64.46	57.16	51.03	87.09	71.38	65.07	71.26
Average	**89.12**	**78.85**	**77.18**	**64.41**	**57.07**	**50.79**	**87.00**	**72.10**	**65.66**	**71.35**
Improvements	*+0.98*	*+0.52*	*+0.08*	*+8.96*	*+5.79*	*+3.70*	*+6.51*	*+6.12*	*+4.04*	*+4.08*
SECOND + PA-AUG + MD3D										
#1	88.96	78.66	76.99	66.18	59.03	52.60	87.27	71.98	65.98	71.96
#2	89.41	79.01	77.20	66.93	59.77	52.78	88.36	71.89	66.08	72.38
#3	89.16	78.87	76.78	67.36	59.69	52.83	87.20	72.09	66.27	72.25
Average	**89.17**	**78.85**	**76.99**	**66.82**	**59.50**	**52.74**	**87.61**	**71.99**	**66.11**	**72.20**
Improvements	*+1.04*	*+0.51*	*-0.11*	*+11.37*	*+8.22*	*+5.65*	*+7.13*	*+6.01*	*+4.49*	*+4.92*

Table 5.1: **Performance comparison of the combination of PA-AUG and MD3D on SECOND [81].** The results are evaluated by the AP with 11 recall positions on the KITTI-*val* set, and the average values of three repeated experiments are reported for each AP. The highest average APs are underlined.

Experiments	Car 3D AP (IoU=0.7)			Ped 3D AP (IoU=0.5)			Cyc 3D AP (IoU=0.5)			mAP
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	
CenterPoint [85]										
#1	84.93	77.17	75.98	57.86	53.42	50.53	83.17	66.82	62.50	68.04
#2	84.99	77.42	76.06	57.19	52.95	49.23	82.32	67.02	62.92	67.79
#3	85.83	77.75	76.22	55.12	52.00	49.23	82.65	68.44	63.47	67.86
Average	**85.25**	**77.45**	**76.09**	**56.72**	**52.79**	**49.66**	**82.71**	**67.43**	**62.97**	**67.90**
CenterPoint + PA-AUG										
#1	86.12	78.32	76.48	58.48	55.53	51.97	83.74	68.26	64.96	69.32
#2	85.47	77.91	76.21	61.59	57.72	53.99	82.54	67.77	63.47	69.63
#3	86.29	78.39	76.33	58.90	55.66	51.98	80.14	67.50	62.38	68.62
Average	**85.96**	**78.21**	**76.34**	**59.66**	**56.30**	**52.65**	**82.14**	**67.84**	**63.60**	**69.19**
Improvements	*+0.71*	*+0.76*	*+0.26*	*+2.93*	*+3.51*	*+2.98*	*-0.57*	*+0.42*	*+0.63*	*+1.29*
CenterPoint + MD3D										
#1	89.07	78.91	77.66	63.18	55.93	50.11	86.68	71.80	65.59	70.99
#2	89.17	78.80	77.50	63.69	55.89	49.77	86.83	66.91	64.99	70.40
#3	89.05	78.92	77.63	63.39	56.06	50.19	87.28	72.35	66.00	71.21
Average	**89.10**	**78.88**	**77.60**	**63.42**	**55.96**	**50.02**	**86.93**	**70.35**	**65.53**	**70.86**
Improvements	*+3.85*	*+1.43*	*+1.51*	*+6.69*	*+3.17*	*+0.36*	*+4.22*	*+2.93*	*+2.57*	*+2.97*
CenterPoint + PA-AUG + MD3D										
#1	89.44	79.04	77.65	65.20	58.21	51.99	86.79	66.74	65.28	71.15
#2	88.96	78.86	77.47	65.18	58.29	52.13	86.60	66.85	65.58	71.10
#3	89.59	79.22	77.44	67.03	59.44	52.73	87.15	66.71	65.44	71.64
Average	**89.33**	**79.04**	**77.52**	**65.80**	**58.64**	**52.29**	**86.84**	**66.77**	**65.43**	**71.30**
Improvements	*+4.08*	*+1.59*	*+1.44*	*+9.08*	*+5.85*	*+2.62*	*+4.13*	*-0.66*	*+2.47*	*+3.40*

Table 5.2: **Performance comparison of the combination of PA-AUG and MD3D on CenterPoint [85]** The results are evaluated by the AP with 11 recall positions on the KITTI-*val* set, and the average values of three repeated experiments are reported for each AP. The highest average APs are underlined.

5.3 Discussion

In this chapter, we have presented the experimental results of the combination of PA-AUG and MD3D on SECOND and CenterPoint, which are anchor-based and anchor-free 3D object detectors. The results show the combination of the two proposed methods has a synergy effect when they are used together. The improvements in PA-AUG and MD3D are cumulated to the improvements in the combination, which means both of them enhance different aspects of object detectors independently. However, it also shows some negative effects on APs, such as Car (Hard) in SECOND and Cyclist (Moderate) in CenterPoint. Therefore, the backbone network structures affect the improvements of PA-AUG, MD3D, and their combination.

Chapter 6

Conclusion

6.1 Summary

The general pipeline of LiDAR-based 3D object detection methods consists of three stages: 1) data augmentation & preprocessing, 2) backbone network, and 3) detection head. (Section 1.1) Previous studies have focused on the backbone network architectures which is the most core part for overall performance and speed. Therefore, a lot of highly optimized 3D object detectors have been proposed [81, 36, 59, 61, 85]. However, their backbone network architectures greatly vary each other owing to their different preprocessing methods. In this dissertation, we aim to improve 3D object detectors regardless of their backbone network architectures.

In chapter 3, we propose a novel part-aware data augmentation (PA-AUG) which makes better use of 3D information of point clouds than the conventional methods. We divide the objects into 8 or 4 partitions according to intra-object part location and apply five separate augmentation methods which can be used simultaneously in a partition-based way. The proposed data augmentation

methods can be universally applied to any architecture, and PA-AUG further improves one of the SOTA detectors on the KITTI dataset. Experimental results show that PA-AUG can improve robustness to corrupted data and enhance data efficiency.

In chapter 4, we propose a novel mixture-density-based 3D object detection (MD3D) which reformulates the regression of 3D bounding boxes in point clouds as a density estimation problem. The MD3D is easy to use and can be applied to various types of feature encoding methods without considering the target assignment policy and network structure. Experiments on the KITTI and Waymo datasets show that the MD3D outperforms conventional methods in terms of performance, speed, ease of use, and flexibility.

In chapter 5, we show the experimental results on the combination of PA-AUG and MD3D. We apply the combination of the proposed two methods to SECOND [81] and CenterPoint [85], which are anchor-based and anchor-free 3D object detectors. Their results on the KITTI dataset show the combination significantly improves the detection performance of both detectors. The increases in performance seem to be accumulated as applying PA-AUG and MD3D, which implies they improve different aspects of the detection performance and have synergy when they are used together.

6.2 Limitations and Future works

6.2.1 Hyper-parameter-free PA-AUG

PA-AUG includes five partition-based data augmentation methods: dropout, swap, mix, sparse, and noise. They are applied stochastically to each partition. Thus, PA-AUG requires many hyper-parameters, *i.e.,* the probabilities of each method,

the number of partitions to drop, swap, and mix, the number of points to sparsify and generate noise. It is a very time-consuming task to optimize these hyper-parameters for each dataset. Therefore, an automated PA-AUG, which utilizes the loss or accuracy value to manipulate the hyper-parameters automatically, will be essential to apply to multiple datasets.

6.2.2 Redefinition of Part-aware Partitioning

We divide the Car, Pedestrian, and Cyclist with 8, 4, and 4 partitions which separate the characteristic sub-parts of each class. It is defined using human's prior knowledge and the experimental results (Section 3.3.4) show it is more effective than other partitioning methods. However, there are many classes which are hard to define the characteristic sub-parts in the real world. So it may be ineffective to define the part-aware partitioning method with human's prior knowledge. Therefore, a unified partitioning method for all classes or a self-manipulating dynamic partitioning method will be needed when applying PA-AUG to datasets with a lot of classes.

6.2.3 Application to other tasks

Both PA-AUG and MD3D can be applied to other tasks, such as semantic segmentation and object tracking. PA-AUG is beneficial to learn the semantic relation in intra-object sub-parts. Thus, it would be effective to point cloud-based 3D semantic segmentation task and object tracking task. MD3D could be easily utilized in semantic segmentation and object tracking by modifying the loss function and output channels. And its hyper-parameter-free learning method would be working well with other vision tasks.

Bibliography

[1] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

[2] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis. A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3782–3795, 2019.

[3] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. De La Escalera. Birdnet: a 3d object detection framework from lidar information. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3517–3523. IEEE, 2018.

[4] C. M. Bishop. Mixture density networks. 1994.

[5] 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*, pages 11621–11631, 2020.

[6] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2040–2049, 2017.

[7] Q. Chen, L. Sun, Z. Wang, K. Jia, and A. Yuille. Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots. In *European conference on computer vision*, pages 68–84. Springer, 2020.

[8] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2147–2156, 2016.

[9] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pages 424–432, 2015.

[10] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals using stereo imagery for accurate object class detection. *IEEE transactions on pattern analysis and machine intelligence*, 40(5):1259–1272, 2017.

[11] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[12] 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*, pages 5428–5437, 2022.

[13] S. Cheng, Z. Leng, E. D. Cubuk, B. Zoph, C. Bai, J. Ngiam, Y. Song, B. Caine, V. Vasudevan, C. Li, et al. Improving 3d object detection through progressive population based augmentation. *arXiv preprint arXiv:2004.00831*, 2020.

[14] J. Choi, I. Elezi, H.-J. Lee, C. Farabet, and J. M. Alvarez. Active learning for deep object detection via probabilistic modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10264–10273, 2021.

[15] J. Choi, Y. Song, Y. Kim, J. Yoo, and N. Kwak. Md3d: Mixture-density-based 3d object detection in point clouds. *IEEE Access*, 10:104011–104022, 2022.

[16] J. Choi, Y. Song, and N. Kwak. Part-aware data augmentation for 3d object detection in point cloud. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3391–3397. IEEE, 2020.

[17] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[18] 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*, volume 35, pages 1201–1209, 2021.

[19] T. DeVries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[20] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1355–1361, 2017.

[21] L. Fan, Z. Pang, T. Zhang, Y.-X. Wang, H. Zhao, F. Wang, N. Wang, and Z. Zhang. Embracing single stride 3d object detector with sparse transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8458–8468, 2022.

[22] D. Feng, L. Rosenbaum, and K. Dietmayer. Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3266–3273. IEEE, 2018.

[23] R. Ge, Z. Ding, Y. Hu, Y. Wang, S. Chen, L. Huang, and Y. Li. Afdet: Anchor free one stage 3d object detection. *arXiv preprint arXiv:2006.12671*, 2020.

[24] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[25] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[26] B. Graham. Sparse 3d convolutional neural networks. *arXiv preprint arXiv:1505.02890*, 2015.

[27] B. Graham and L. van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

[28] T. Guan, J. Wang, S. Lan, R. Chandra, Z. Wu, L. Davis, and D. Manocha. M3detr: Multi-representation, multi-scale, mutual-relation 3d object detection with transformers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 772–782, 2022.

[29] M. Hahner, D. Dai, A. Liniger, and L. Van Gool. Quantifying data augmentation for lidar based 3d object detection. *arXiv preprint arXiv:2004.01643*, 2020.

[30] C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang. Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11873–11882, 2020.

[31] Y. He and J. Wang. Deep mixture density network for probabilistic object detection. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10550–10555. IEEE, 2020.

[32] Y. Hu, Z. Ding, R. Ge, W. Shao, L. Huang, K. Li, and Q. Liu. Afdetv2: Rethinking the necessity of the second stage for object detection from point clouds. *arXiv preprint arXiv:2112.09205*, 2021.

[33] Y. Huang, X. Liu, Y. Zhu, Z. Xu, C. Shen, Z. Che, G. Zhang, Y. Peng, F. Feng, and J. Tang. Label-guided auxiliary training improves 3d object detector. *arXiv preprint arXiv:2207.11753*, 2022.

[34] H. Inoue. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, 2018.

[35] H. Kuang, B. Wang, J. An, M. Zhang, and Z. Zhang. Voxel-fpn: Multi-scale voxel feature aggregation for 3d object detection from lidar point clouds. *Sensors*, 20(3):704, 2020.

[36] 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 Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[37] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*, pages 734–750, 2018.

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

[39] J. Li, H. Dai, L. Shao, and Y. Ding. Anchor-free 3d single stage detector with mask-guided attention for point cloud. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 553–562, 2021.

[40] P. Li, T. Qin, et al. Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 646–661, 2018.

[41] R. Li, X. Li, P.-A. Heng, and C.-W. Fu. Pointaugment: an auto-augmentation framework for point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6378–6387, 2020.

[42] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019.

[43] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[44] C. Liu, C. Gao, F. Liu, J. Liu, D. Meng, and X. Gao. Ss3d: Sparsely-supervised 3d object detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8428–8437, 2022.

[45] Z. Liu, H. Tang, Y. Lin, and S. Han. Point-voxel cnn for efficient 3d deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[46] 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 *AAAI*, 2020.

[47] 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*, pages 2723–2732, 2021.

[48] Z. Miao, J. Chen, H. Pan, R. Zhang, K. Liu, P. Hao, J. Zhu, Y. Wang, and X. Zhan. Pvgnet: A bottom-up one-stage 3d object detector with integrated multi-level features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3279–3288, 2021.

[49] I. Misra, R. Girdhar, and A. Joulin. An end-to-end transformer model for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2906–2917, 2021.

[50] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE*

conference on Computer Vision and Pattern Recognition, pages 7074–7082, 2017.

[51] A. Paigwar, O. Erkent, C. Wolf, and C. Laugier. Attentional pointnet for 3d-object detection in point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[52] A. Prakash, K. Chitta, and A. Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7077–7087, 2021.

[53] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019.

[54] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

[55] 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 conference on computer vision and pattern recognition*, pages 652–660, 2017.

[56] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[57] R. Qian, D. Garg, Y. Wang, Y. You, S. Belongie, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao. End-to-end pseudo-lidar for

image-based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5881–5890, 2020.

[58] Z. Qin, J. Wang, and Y. Lu. Triangulation learning network: from monocular to stereo 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7615–7623, 2019.

[59] 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*, pages 10529–10538, 2020.

[60] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang, and H. Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *arXiv preprint arXiv:2102.00463*, 2021.

[61] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

[62] 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.

[63] 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*, pages 1711–1719, 2020.

[64] V. A. Sindagi, Y. Zhou, and O. Tuzel. Mvx-net: Multimodal voxelnet for 3d object detection. In *2019 International Conference on Robotics and*

Automation (ICRA), pages 7276–7282. IEEE, 2019.

[65] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[66] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[67] F. Su, H. Zhu, T. Chen, L. Li, F. Yang, H. Peng, L. Tang, X. Zuo, Y. Liang, and S. Ying. An anchor-based graph method for detecting and classifying indoor objects from cluttered 3d point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 172:114–131, 2021.

[68] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov. Scalability in perception for autonomous driving: Waymo open dataset, 2019.

[69] L. Taylor and G. Nitschke. Improving deep learning with generic data augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1542–1547. IEEE, 2018.

[70] O. D. Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020.

[71] Z. Tian, C. Shen, H. Chen, and T. He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019.

[72] A. Varamesh and T. Tuytelaars. Mixture dense regression for object detection and human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13086–13095, 2020.

[73] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[74] H. Wang, S. Shi, Z. Yang, R. Fang, Q. Qian, H. Li, B. Schiele, and L. Wang. Rbgnet: Ray-based grouping for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1110–1119, 2022.

[75] Y. Wang, T. Ye, L. Cao, W. Huang, F. Sun, F. He, and D. Tao. Bridged transformer for vision and point cloud 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12114–12123, 2022.

[76] Z. Wang and K. Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1742–1749, 2019.

[77] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Data-driven 3d voxel patterns for object category recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1903–1911, 2015.

[78] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 924–933. IEEE, 2017.

[79] Q. Xu, Y. Zhong, and U. Neumann. Behind the curtain: Learning occluded shapes for 3d object detection. *arXiv preprint arXiv:2112.02205*, 2021.

[80] Y. Xue, J. Mao, M. Niu, H. Xu, M. B. Mi, W. Zhang, X. Wang, and X. Wang. Point2seq: Detecting 3d objects as sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8521–8530, 2022.

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

[82] H. Yang, Z. Liu, X. Wu, W. Wang, W. Qian, X. He, and D. Cai. Graph r-cnn: Towards accurate 3d object detection with semantic-decorated local graph. In *ECCV*, 2022.

[83] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1951–1960, 2019.

[84] M. Ye, S. Xu, and T. Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1631–1640, 2020.

[85] T. Yin, X. Zhou, and P. Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[86] J. Yoo, N. Ahn, and K.-A. Sohn. Rethinking data augmentation for image super-resolution: A comprehensive analysis and a new strategy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8375–8384, 2020.

[87] J. Yoo, H. Lee, I. Chung, G. Seo, and N. Kwak. Training multi-object detector by estimating bounding box distribution for input image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3437–3446, 2021.

[88] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. In *International Conference on Learning Representations*, 2019.

[89] P. Yun, L. Tai, Y. Wang, C. Liu, and M. Liu. Focal loss in 3d object detection. *IEEE Robotics and Automation Letters*, 4(2):1263–1270, 2019.

[90] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019.

[91] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[92] 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*, pages 18953–18962, 2022.

[93] W. Zheng, W. Tang, S. Chen, L. Jiang, and C.-W. Fu. Cia-ssd: Confident iou-aware single-stage object detector from point cloud. *arXiv preprint arXiv:2012.03015*, 2020.

[94] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. Random erasing data augmentation. In *AAAI*, pages 13001–13008, 2020.

[95] X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[96] Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Ouyang, J. Guo, J. Ngiam, and V. Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *Conference on Robot Learning*, pages 923–932. PMLR, 2020.

초 록

자율주행자동차, 로봇의 인식 장비로 많이 활용되고있는 라이다 (LiDAR)는 레이저 펄스를 방출하여 되돌아오는 시간을 계산하여 포인트 클라우드 (point cloud) 형태로 주변 환경을 감지한다. 주변 환경을 감지할때 가장 중요한 부분은 근처에 어떤 객체가 있는지, 어디에 위치해 있는지를 인식하는 것이고 이러한 작업을 수행하기 위해 포인트 클라우드를 활용하는 3차원 객체 검출 기술들이 많이 연구되고 있다.

포인트 클라우드 데이터의 전처리 방법에 따라 매우 다양한 구조의 백본 네트워크 (backbone network) 가 연구되고 있다. 고도화된 백본 네트워크들로 인해 인식 성능에 큰 발전을 이루었지만, 이들의 형태가 크게 다르기 때문에 서로 호환성이 부족하여 연구들의 갈래가 많이 나누어지고 있다. 본 논문에서 풀고자하는 문제는 “파편화된 백본 네트워크의 구조들에 구애받지 않고 3차원 객체 검출기의 성능을 향상시킬 방법이 있는가” 이다. 이를 위해 본 논문에서는 포인트 클라우드 데이터 기반의 3차원 객체 검출 기술을 향상시키는 두 가지 방법을 제안한다.

첫 번째는 3차원 경계 상자 (3D bounding box) 의 구조적인 정보의 활용을 최대화하는 구조 감응형 데이터 증강 (PA-AUG) 기법이다. 3차원 경계 상자 라벨은 객체에 딱 맞게 생성되고 방향값을 포함하기 때문에 상자 내에 객체의 구조 정보를 포함하고 있다. 이를 활용하기 위해 우리는 3차원 경계 상자를 구조 감응형 파티션으로 구분하는 방식을 제안하고, 파티션 수준에서 수행되

는 새로운 방식의 데이터 증강 기법을 제안한다. PA-AUG는 다양한 형태의 3차원 객체 검출기들의 성능을 강인하게 만들어주고, 학습 데이터를 2.5배 증강시키는 만큼의 인식 성능 향상 효과를 보여준다.

두 번째는 혼합 밀도 신경망 기반 3차원 객체 검출 (MD3D) 기법이다. MD3D는 가우시간 혼합 모델 (Gaussian Mixture Model) 을 이용해 3차원 경계 상자 회귀 문제를 밀도 예측 방식으로 재정의한 기법이다. 이러한 방식은 기존의 라벨 할당식의 학습 방법들과 달리 포인트 클라우드 전처리 형태에 구애받지 않고 동일한 학습 방식을 적용할 수 있다. 또한 기존 방식 대비 학습에 필요한 하이퍼 파라미터가 현저히 적어서 최적화가 용이하여 인식 성능을 크게 높일 수 있을 뿐만 아니라 간단한 구조로 인해 인식 속도도 빨라지게 된다.

PA-AUG와 MD3D는 모두 백본 네트워크 구조에 상관없이 다양한 3차원 객체 검출기에 공통적으로 사용될 수 있으며 높은 인식 성능 향상을 보여준다. 뿐만 아니라 두 기법은 검출기의 서로 다른 영역에 적용되는 기법이므로 함께 동시에 사용할 수 있고, 함께 사용했을때 인식 성능이 더욱 크게 향상된다.

주요어: 3차원 객체 검출, 라이다, 포인트 클라우드, 데이터 증강, 혼합 밀도 신경망, 자율주행

학번: 2017-23640

감사의 글

박사 학위 취득이라는 한 가지 목표를 바라보고 달려왔던 6년의 석박통합 과정 수학 기간이 마침내 결실을 맺게 되었습니다. 앞만 보고 달려오다 보니 주변을 많이 둘러보지 못 했던 것 같은데, 지난 6년을 돌이켜보니 저 혼자만으로는 절대 불가능했을 것이라는 생각이 절실히 듭니다. 이 글을 통해 부족한 저에게 직·간접적으로 많은 도움을 주신 분들께 감사의 마음을 전합니다.

저는 연구하는데 시간이 남들보다 오래 걸릴 뿐만 아니라, 연구를 하며 실패도 많이 겪어왔습니다. 하지만 지도 교수님이신 곽노준 교수님께서는 언제나 잘하고 있다고 많은 격려를 해주셨습니다. 과제를 수행하며 곤란한 상황에 처했을 때도 발 벗고 나서서 문제를 해결해 주시는 모습에 교수님의 따뜻한 마음을 많이 느낄 수 있었습니다. 그럼에도 지난 학위 기간 동안 교수님께 자랑스러운 제자가 되어드리지는 못한 것 같아 아쉬움이 많이 남습니다. 사회에 나가 더욱 발전하여 교수님께 받은 많은 도움을 돌려드릴 수 있는 제자가 되도록 노력하겠습니다. 또한, 바쁘신 시간 할애하여 저의 학위 논문 심사 기간에 건설적인 피드백을 많이 주신 이교구 교수님, 박재흥 교수님, 이원종 교수님, 이민식 교수님께도 깊은 감사의 인사를 드립니다.

연구실에 처음 입학한 2017년, 연구실 막내로 시작해 훌륭하신 연구실 선배님들께 많은 배움을 얻어 갑니다. 어떤 연구를 해야 할지 감이 전혀 없이 방황하던 막내 시절, 첫 연구 주제를 선정할 때부터 선배님들의 영향을 많이 받았고 덕분에 제가 하고 싶은 연구를 찾아가는데 많은 도움이 된 것 같습니

다. 저에게 언제나 든든하고 자랑스러운 배경인 저희 연구실을 멋지게 일궈내주신 선배님들 정말 감사했습니다. 제가 받은 도움을 후배님들에게 충분히 나눠드렸는지 모르겠습니다. 먼저 사회에 나가 새로운 길을 닦고 있겠습니다. 도움이 필요하면 언제든지 편하게 연락 주시길 바랍니다. 연구가 잘 안되더라도 너무 스트레스 받지 마시고 시원하게 호수 한 바퀴 돌고 와서 털어버리시길 바랍니다. 광교 구석구석 멋진 곳이 많으니 한숨 돌리면서 이곳저곳 탐험해 보시면 생각보다 스트레스 해소에 많은 도움이 될 겁니다. 우수한 후배님들이 많이 계셔서 편한 마음으로 연구실을 맡기고 떠납니다.

저에게 언제나 무한한 신뢰를 보내주시며 서포트해 주시는 부모님, 형에게도 감사의 인사를 드립니다. 가족분들의 지지는 오랜 기간 학생 신분으로 공부를 해온 저에게 계속해서 공부를 이어올 수 있었던 가장 큰 원동력이었던 것 같습니다. 졸업 후에도 자랑스러운 아들이 되어 보답하겠습니다.