Out-of-Distribution Detection for LiDAR-based 3D Object Detection

by

Van Duong Nguyen

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2021

Author's Declaration

Statement of Contributions

The thesis is based on collaborative work and a paper manuscript (under submission) by the author and Chengjie Huang, Dr. Vahdat Abdelzad, Christopher Gus Mannes, Luke Rowe, Benjamin Therien, Dr. Rick Salay, and Dr. Krzysztof Czarnecki. The author and Chengjie Huang are the main and equal contributors to this work. The following description details the individual contributions of each collaborator.

The definition of out-of-distribution (OOD) detection for object detection is mainly due to the author, Chengjie Huang, Dr. Vahdat Abdelzad, with the author proposing the idea of distinguishing the training distribution and true distribution in the definition.

The main contribution of the author is the method and algorithm to generate OOD objects for LiDAR-based 3D object detectors, as well as the implementation of the method and the generation of the four OOD datasets (Carla, KITTI ignored, KITTI FP, and Waymo OOD) by augmenting KITTI frames. Luke Rowe supported the implementation by selecting and hand labeling the Waymo weird objects, and Benjamin Therien manually labeled the KITTI false positive (FP) objects and identified their occurrence in the KITTI training dataset.

The adaptation of the OOD methods from image classification to object detection, including the feature extraction method for OOD detection, and most of the OOD detection code, including the experimental results, is mainly due to Chengjie Huang. The author wrote the initial version of the code for the OOD method based on the Mahalanobis distance and performed optimizations on the code using normalizing flow. Dr. Vahdat Abdelzad proposed using supervised contrastive learning to improve the quality of feature maps for OOD detection. Chris Gus Mannes implemented the initial version of the contrastive learning code, which was reimplemented for the OOD context by Chengjie Huang.

The author wrote the first draft of the manuscript except for Chapter 6, which was written by Chengjie Huang. The text was edited by Dr. Vahdat Abdelzad and Dr. Krzysztof Czarnecki. Dr. Vahdat Abdelzad, Dr. Krzysztof Czarnecki, and Dr. Rick Salay also provided regular feedback throughout the research.

Abstract

3D object detection is an essential part of automated driving, and deep neural networks (DNNs) have achieved state-of-the-art performance for this task. However, deep models are notorious for assigning high confidence scores to out-of-distribution (OOD) inputs, that is, inputs that are not drawn from the training distribution. Detecting OOD inputs is challenging and essential for the safe deployment of models. OOD detection has been studied extensively for the classification task, but it has not received enough attention for the object detection task, specifically LiDAR-based 3D object detection. In this work, we focus on the detection of OOD inputs for LiDAR-based 3D object detection. We formulate what OOD inputs mean for object detection and propose to adapt several OOD detection methods for object detection. We accomplish this by our proposed feature extraction method. We also propose to use a contrastive loss to improve both the performance of the object detection and OOD detection methods. To evaluate OOD detection methods, we develop a simple but effective technique of generating OOD objects for a given object detection model. Our evaluation based on the KITTI dataset demonstrates that there is an improvement over the baseline. It also shows that different OOD detection methods have biases toward detecting specific OOD objects. It emphasizes the importance of combined OOD detection methods and more research in this direction.

Acknowledgements

I want to thank my advisor Professor Krzysztof Czarnecki, Dr. Vahdat Abdelzad, and Chengjie Huang for the guidance, support, and feedback throughout my studies. Thank Professor Steven Waslander and Professor Mark Crowley, for reviewing my thesis and giving valuable feedback. Last but not least, I also want to thank Dr. Rick Salay, Dr. Sean Sedwards, Dr. Michal Antkiewicz, SunSheng Gu, Matthew Pitropov, Prarthana Bhattacharyya for many valuable discussions and their support.

Dedication

This thesis is dedicated to my respected parents Duc and Niem, Tuan and Huyen and my beloved wife Thao, for their endless love, support, and encouragement. And to my late grandmother and my late uncle—I miss you every single day.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

3D object detection is a crucial component of autonomous vehicles (AVs) allowing them to detect surrounding objects. The state-of-the-art (SOTA) in 3D object detection methods relies on Deep Neural Networks (DNNs) and LiDAR sensing technology [88, 65, 35]. However, DNNs are susceptible to the out-of-distribution (OOD) problem: they can assign high confidence scores to inputs not drawn from the training distribution. The problem poses a safety issue, hindering the deployment of AVs on public roads. For instance, an AV classifying a bike rack or signboard on the roadside as a pedestrian (seen in Figure 1.1) can suddenly apply a brake or another dangerous evasive maneuver. Out-of-distribution (OOD) detection [61] aims to detect such cases.

While a tremendous amount of work attempts to improve SOTA in LiDAR-based 3D object detection, OOD detection in the context of 3D object detection has hardly been explored. Most OOD detection methods are for classification [61, 42, 27, 59, 38, 25, 58, 87, 37, 74, 34, 41, 28] or segmentation tasks [4, 3, 46, 76]. Thus, there is no clear definition for what OOD input means in the context of object detection.

There are several challenges associated with OOD detection for object detection. In classification, all considered classes are clearly defined. In object detection, a clear definition is available solely for the foreground classes, but not the background classes. An OOD input for classification commonly means anything not belonging to the considered classes. By analogy, an OOD input for object detection could be defined as anything not belonging to the foreground and background classes. This definition requires adjustment, however, because even though one can define OOD inputs for foreground classes, there are no objects that are neither in foreground nor background classes, by the very definition of object detection.

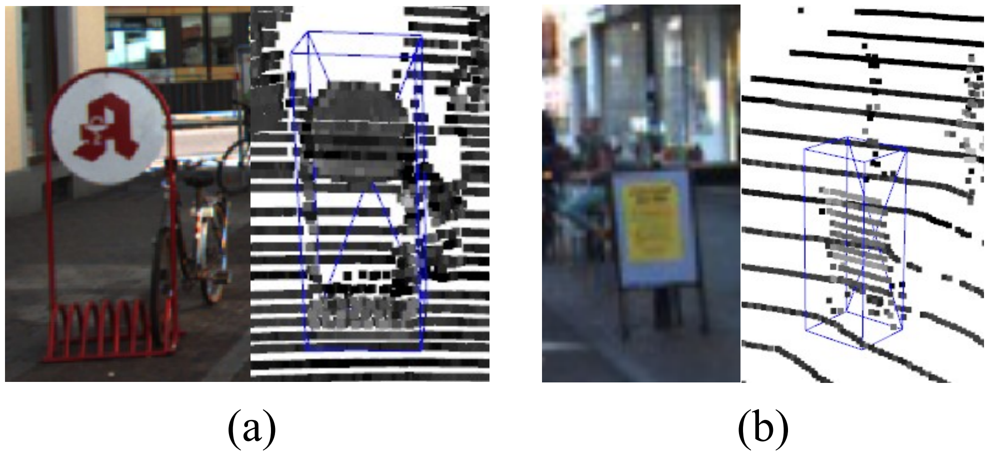


Figure 1.1: Examples of OOD objects detected as foreground objects by PointPillars, a LiDAR-based 3D object detector. The camera images are for visual reference only; the corresponding fragments of the input point clouds are to the right. (a) A bike rack is detected as a pedestrian with 0.86 confidence. (b) A signboard is detected as a pedestrian with 0.74 confidence.

Furthermore, in contrast to image classification, inputs to object detection may contain multiple objects, which can be either foreground or background. This brings an extra challenge for OOD detectors that rely on embeddings, i.e., feature maps. While OOD detectors for image classification commonly use the entire feature map, object detectors may produce an arbitrary number of predictions per input, and thus OOD detection needs to extract features associated with each object prediction individually.

Lastly, OOD detection evaluation requires access to OOD inputs. This is easily achieved for image classification. For example, for the MNIST dataset [36], any image other than MNIST digits can be considered as an OOD input. Obtaining OOD samples for 3D object detection is not as straightforward, however. First, point clouds from different LiDAR sensors can differ significantly, both in terms of beam arrangement and intensity values, which prevents simple mixing of frames from different datasets to vary the scene content. Second, object detection requires scenes with both in-distribution (ID) and OOD objects. Whereas object detection datasets for camera images that are labeled with hundreds of foreground classes exist, 3D LiDAR object detection datasets for autonomous driving are usually limited to a handful of foreground classes (e.g., cars, pedestrians, and cyclists). This makes it challenging to simulate OOD objects by withholding images with certain classes from training.

In this thesis, we tackle the aforementioned challenges by first proposing a definition of OOD inputs for object detection. Our analysis of OOD detection for object detection identifies six types of OOD objects with respect to the foreground classes. We focus on the three of them for which the detector produces detections. We propose multiple detection methods for these OOD types in the context of LiDAR-based 3D object detection, by adapting and extending six OOD methods from image classification. Most importantly, we design a method for extracting features associated with object detections to be used for OOD detection. We also explore the use of a contrastive loss to improve the performance of both the object detector and OOD detection. Finally, as part of the evaluation, we propose a simple yet effective method to generate OOD objects for a given LiDAR-based 3D object detector.

We extensively evaluate the proposed OOD detection methods and OOD object generator on the KITTI dataset [22]. In particular, we compare the OOD detection methods against a simple but strong baseline on the KITTI dataset augmented with diverse real and synthetic OOD objects using the proposed generator. The results show that different OOD detection methods have biases toward detecting specific types of OOD objects. Furthermore, the best practices previously identified for OOD detection in image classification may not hold in object detection.

The main contributions presented in this thesis (with the author mainly contributing to 1 and 3, which enabled 2) are as follows.

1. We formulate the problem of OOD detection for object detection by identifying six types of OOD objects.

2. We design a method to extract features for OOD detection on predicted objects, and use it to adapt OOD methods from image classification to LiDAR-based 3D object detection. Further, we propose the use of a supervised contrastive loss to improve both object and OOD object detection performance.

3. We propose a technique to generate OOD objects for LiDAR-based 3D object detectors and use it to augment the KITTI dataset with OOD objects and extensively evaluate the proposed OOD methods. The evaluation results and setup are expected to aid and stimulate future research on OOD detection for LiDAR-based 3D object detection.

As far as we are aware, we are the first to explore OOD detection for LiDAR-based 3D object detection.

This thesis is organised as follows:

- Chapter 2 covers the necessary background on 3D LiDAR-based object detection and out-of-distribution methods.
- Chapter 3 defines OOD detection in object detection and proposes a technique to generate OOD datasets.
- Chapter 4 proposes a feature extraction method for OOD detection in object detection and a method to investigate the effects of supervised contrastive learning on the quality of feature maps for OOD detection.
- Chapter 5 provides details on the experiments that we perform. It describes the datasets for object detection, training parameters for object detection and OOD detection methods.
- Chapter 6 contains results and discussion.
- Chapter 7 presents the conclusions and future work.

Chapter 2

Background and Related Work

This chapter summarizes existing works on 3D LiDAR-based object detection and OOD detection. Following the summary, it also provides more detailed background knowledge on the OOD detection methods that are adapted and evaluated in this work. Finally, it concludes by describing OOD detection metrics.

2.1 LiDAR-based 3D Object Detection

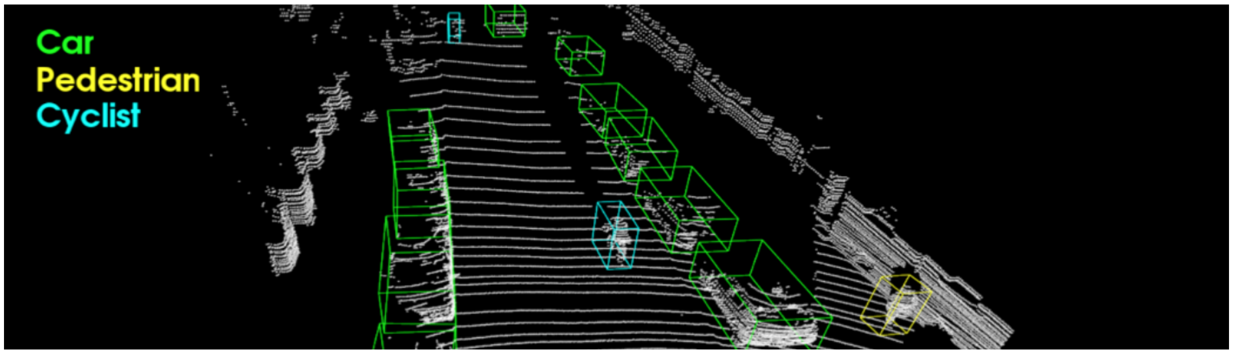


Figure 2.1: An example of LiDAR-based 3D object detection. Detected objects are inside bounding boxes. From VoxelNet [90].

The goal of 3D object detection is to detect objects with their class and 3D location and extent. Existing detectors use images [79, 10, 8, 51], LiDAR point clouds [40, 18, 39, 90, 68, 85, 82, 81, 35, 91, 66, 65, 67, 86], or multi-modal inputs [33, 11, 55, 89]. In this work, we focus on LiDAR-based 3D object detectors because of their superior performance on public datasets. There are different deep architectures based on points [66, 67, 85], voxels [90, 81, 91, 35, 82, 86], or both [65].

In the category of point-based methods, Shi et al. [66] propose a two-stage detector in which the first stage generates bounding box proposals using point cloud segmentation and the second stage focuses on each proposal to aggregate point features for bounding box refinement and classification. Yang et al. [85] present another two-stage detector introducing spherical anchors for bounding box proposals in the first stage and a novel module to aggregate point features in the second stage. Shi et al. [67] construct a graph from a raw point cloud and apply graph neural networks to learn features of each point. Then the features of the points are aggregated for bounding box prediction and classification. Though these methods archive good performance, they are computationally intensive and require a lot of parameter tuning.

In the category of voxel-based methods, detectors [90, 81, 91, 35, 82, 86] divide an input point cloud into voxels and apply PointNet [56] to learn voxel features. The voxel features then go through 3D CNN and 2D CNN for bounding box prediction. These methods have a relatively simple architecture, high performance, and low latency, which are desirable properties for real-time applications such as autonomous driving.

Shi et al. [65] combine both advantages of voxel-based methods and point-based methods and achieve the best performance among 3D LiDAR-based methods. However, again the architecture is complex and has high computational cost.

In this work, we adopt PointPillars [35] because of its high performance and low computational cost, which are essential for resource-constrained and safety-critical systems. PointPillars first voxelizes the input point cloud into vertical voxels called pillars and then extracts pillar features using PointNet [56]. Their bird's-eye-view (BEV) projection is passed to a 2D convolutional backbone, which is followed by a single shot detector (SSD) [45] object detector classification and regression heads (see Figure 4.1).

2.2 Out-of-Distribution Detection

OOD detection has been investigated extensively under terms such as anomaly detection, novelty detection, and open set recognition [61, 83]. While there are subtle differences among the different problem settings (see [83] for a detailed discussion and a unified framework), the main objective is to detect test inputs that are "novel" or unusual in some way with respect to the training dataset. In general, the "novelty" can be characterized in the semantic (i.e., label) space or the sensoric (i.e., input) space. A so-called *semantic shift* occurs when test inputs are drawn from classes that were not present in the training [83]. In contrast, a so-called *covariate shift* occurs when inputs (i.e., the covariates or the independent variables in statistical jargon) are drawn from a different distribution in testing than in training [70]. In the context of OOD, we are interested in detecting inputs that are novel, either because they come from a class different than the classes sampled in training, or because they are rare in the training dataset, i.e., come from a low-density region of the training distribution. Such rare inputs are also called *anomalous* [60]. Note that an anomalous input may be served to a classifier at test time because of covariate shift, or simply by chance even if the training and testing distributions are the same. In this work, we consider both semantic novelty (new label) and input novelty (same label, but an instance that is rare or unrepresented in training).

The two types of novelty can be formalized as follows. Let us assume a training dataset $\mathcal{D}$ of $(\mathbf{x}_i, y_i)$ drawn from a joint distribution $P_{X,Y}$, i.e., $(\mathbf{x}_i, y_i) \sim P_{X,Y}$. X and Y are the random variables representing the inputs and outputs, respectively. Further, the support of X and Y is $\mathcal{X} = \mathbb{R}^D$ and $\mathcal{Y} = \{1, ..., C\}$, respectively, where D is the input dimensionality and C is the number of classes. Thus, we have $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. A test sample $(\hat{\mathbf{x}}, \hat{y})$ is OOD through *semantic novelty* iff $\hat{y} \notin \mathcal{Y}$, i.e., the sample's ground-truth class is different than any of the classes in $\mathcal{Y}$. A so-called *simple covariate shift* is defined by the marginal input distribution in training P_X being different from the marginal input distribution in testing P'_X, i.e., $P_X \neq P'_X$, while the labeling of the inputs, i.e., $P(Y|X)$, remains unchanged [70]. A test input $(\hat{\mathbf{x}}, \hat{y})$, with $\hat{y} \in \mathcal{Y}$, is OOD through *anomaly* iff $p_X(\hat{\mathbf{x}}) \leq \tau$, where $\tau \geq 0$ is some threshold such that the probability of $\{\mathbf{x} \in \mathcal{X} | p_X(\mathbf{x}) \leq \tau\}$ is 'sufficiently small' [60], and p_X is the PDF of P_X. Note that $p_X(\hat{\mathbf{x}}) \leq \tau$ can occur because of (i) the label of $\hat{\mathbf{x}}$ is not in $\mathcal{Y}$ (i.e., semantic novelty), or because, despite having a label in $\mathcal{Y}$, $\hat{\mathbf{x}}$ comes from a low-density region of P_X (i.e., novelty of an instance from $\mathcal{Y}$) either (ii) by chance when drawing from P_X or (iii) due to a covariate shift that makes $\mathbf{x}$ less rare in P'_X.

Figure 2.3 shows standard benchmarking datasets for OOD detection in image classification. If a classifier is trained on MNIST hand-written digits dataset (Figure 2.3a), then samples from the street view house numbers (SVHN) dataset (Figure 2.3b) or the CIFAR-10 objects dataset (Figure 2.3c) are considered OOD inputs.

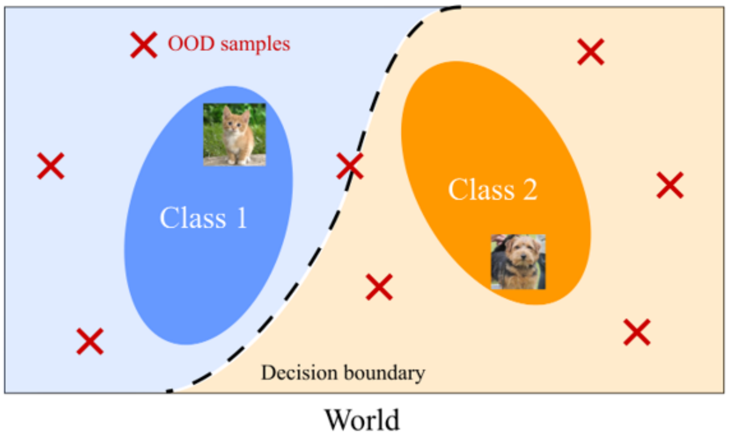


Figure 2.2: An example of input distributions of a cat and dog classifier. OOD samples are not from cat and dog distributions. From Chengjie Huang.

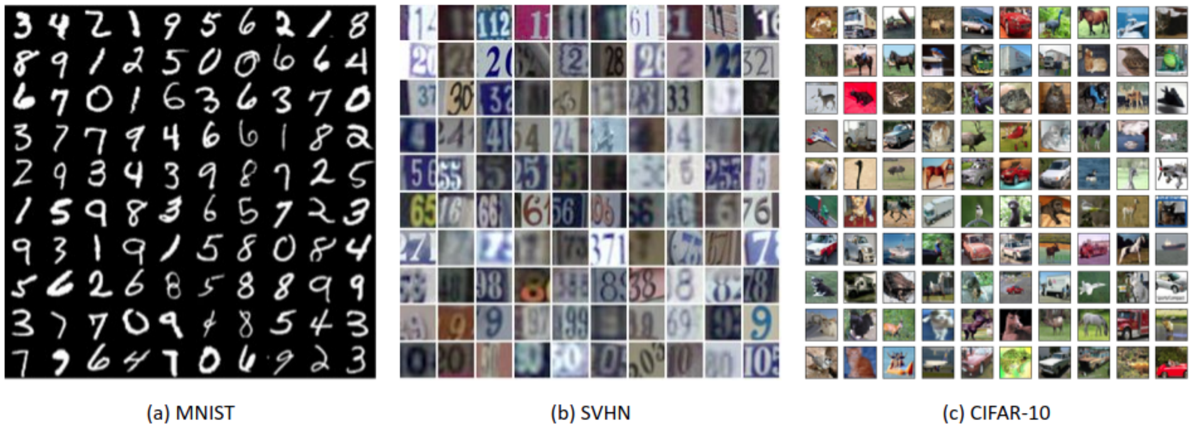


Figure 2.3: Examples of (a) MNIST dataset, (b) SHVN and (c) CIFAR-10. If a classifier is trained for MNIST digits recognition, then inputs from SHVN and CIFAR-10 are OOD.

A majority of works on OOD detection focus on classification [42, 27, 59, 38, 25, 58, 87, 37, 74, 34, 41, 28], with some recent work on image segmentation [4, 3, 6, 46, 76]. The closest to OOD detection for object detection is the work on open set object detection [48, 49, 50, 47]

Hendrick et al. [25] propose a baseline method for detecting OOD inputs for deep image classifiers. It uses max-softmax scores to decide if a sample is an ID or OOD. It assumes that the classifier generally produces a low confidence score for OOD samples. However, works from Guo et al. [24], Nguyen et al. [52], and Goodfellow et al. [23] show that confidence outputs of neural networks are uncalibrated and that neural networks can assign high confidence for arbitrary inputs. ODIN [42] further improves the baseline by applying temperature scaling and input perturbation. Temperature scaling calibrates softmax scores, and input perturbation increases the separation between softmax scores of OOD and ID samples. However, this method requires the availability of OOD samples during training to search for the best temperature constant and perturbation magnitude. Hsu et al. propose Generalized-ODIN [28], removing the need for OOD data for tuning. Lee et al. [37] propose to train jointly a classifier and a GAN network that generates OOD samples for training the classifier. It combines a novel loss named confidence loss with cross-entropy loss. This combination forces the classifier to assign a lower confidence score to OOD samples. Further, Lee et al. [38] use Mahalanobis distance of the input sample to the nearest class-conditional Gaussian distribution estimated from the in-distribution data as the sample's OOD score. This method assumes that the class-conditional distribution of features from the logits layer follows a Gaussian distribution. Furthermore, the authors propose to combine the Mahalanobis distance from multiple layers in the network using logistics regression. The logistic regression model is trained by cross-validation on the validation dataset or adversarial examples generated using FGSM [23].

Uncertainty estimation has also been utilized for detecting OOD samples. The idea is that the samples out of the training distribution should have a higher uncertainty, and in particular epistemic uncertainty [30]. DeVries et al. [14] propose to train a network from scratch with two branches: one for class label prediction and one for confidence prediction. Vyas et al. [74] use ensemble techniques to estimate the confidence of predictions. Lakshminarayanan et al. [34] propose to train an ensemble of several independently trained models for uncertainty prediction. Gal et al. [21] and Li et al. [41] use MC Dropout to estimate uncertainty of predictions.

One-class classifiers such as OC-SVM [62] and SVDD [73] have been successfully applied to OOD detection as well. Recently, Abdelzad et al. [1] has proposed using an OC-SVM trained with features extracted from an optimal layer to detect OOD samples. It shows that features from earlier layers work well for OOD detection in image classification. Bishop [5] suggests that a natural way to detect OOD samples is to estimate the density of ID samples and check if samples are in a low density area. Some works [2, 12, 26, 92, 58] also use generative models [32, 59, 54] to detect OOD samples.

Works in [27, 80] bring up the importance of learning a good representation for OOD detection. A general approach is to learn a good representation and then to design an OOD scoring function for the representation. Applying contrastive learning to self-supervised learning has been shown to improve representation in general [9]. Several works [72, 77, 69] show that contrastive self-supervised representation learning can also improve performance in OOD detection.

Several works in image segmentation [4, 3, 46, 76] have adapted OOD detection methods from image classification for OOD detection at pixel level. They aim to reject the OOD pixels for segmentation and hence improve the segmentation performance. In the automotive domain, Blum et al. [6] create a dataset to benchmark OOD detection for image segmentation. Nitsch [53] evaluates OOD detection for image detection applied to image patches. The OOD detection is trained on the AV datasets KITTI and Nuscenes and tested in ImageNet. Wong et al. [78] propose a method to tackle open set semantic segmentation in the 3D point cloud. The method can recognize and segment known and unknown classes in 3D point clouds.

Finally, Miller et al. have a series of works [48, 49, 50, 47] in open set object detection. These works make use of uncertainty to detect "unknown unknown" objects, defined as not represented in the training dataset and misclassified as foreground objects. Open set recognition is very similar to OOD detection. Our work is different because we aim to detect OOD objects with respect to a foreground class that are detected as foreground. These objects may or may not be represented in the training set. Thus, our OOD definition goes beyond the unknown unknown objects. Overall, most of the works are in the image domain. There is no work addressing OOD detection in 3D LiDAR-based object detection.

2.3 OOD Detection Methods

OOD detection methods aim to design a scoring function f, which maps an input to an OOD score. The function f can be learned from ID training and OOD training samples. Given an input $\mathbf{x}$ with an OOD score $f(\mathbf{x})$, a threshold ζ is determined such that:

$$\begin{cases} f(\mathbf{x}) \geq \zeta \text{ if } \mathbf{x} \text{ is ID} \\ f(\mathbf{x}) < \zeta \text{ if } \mathbf{x} \text{ is OOD} \end{cases} \tag{2.1}$$

This section explains key OOD detection methods that can be easily adopted for existing models: max-softmax, uncertainty estimates, Mahalanobis distance, OC-SVM and normalizing flows. We adapt these methods for object detection and compare them in our experiments. We do not use methods that require OOD training samples because OOD training samples are diverse and hard to generalize in actual deployment.

Max-softmax: In this method [25], the OOD score is the class probability score of the prediction. It has been proposed as a baseline for classification. We also consider this as a baseline method for object detection.

Uncertainty estimates: Aleatoric and epistemic uncertainty estimates can be used as the OOD score [41]. We modified the 3D object detector used in this work to obtain uncertainty estimates via the MC-Dropout method [20]. As a result, our 3D detection models have the capability to produce uncertainty scores.

Mahalanobis distance: This method [38] measures the Mahalanobis distance between a sample and class-conditional Gaussian distributions estimated from the training distribution. The closest distance is used as an OOD score. There are two variants of this method. The first one does not rely on OOD samples, while the second one requires OOD samples for training a linear model. We use the first variant in this work to avoid the bias involved in the second variant:

$$M(\mathbf{x}) = \max_c -\left(f(\mathbf{x}) - \widehat{\mu}_c\right)^\top \widehat{\mathbf{\Sigma}}^{-1} \left(f(\mathbf{x}) - \widehat{\mu}_c\right) \tag{2.2}$$

where $\widehat{\mu}_c$ is empirical class mean and $\widehat{\mathbf{\Sigma}}$ is the covariance of training data.

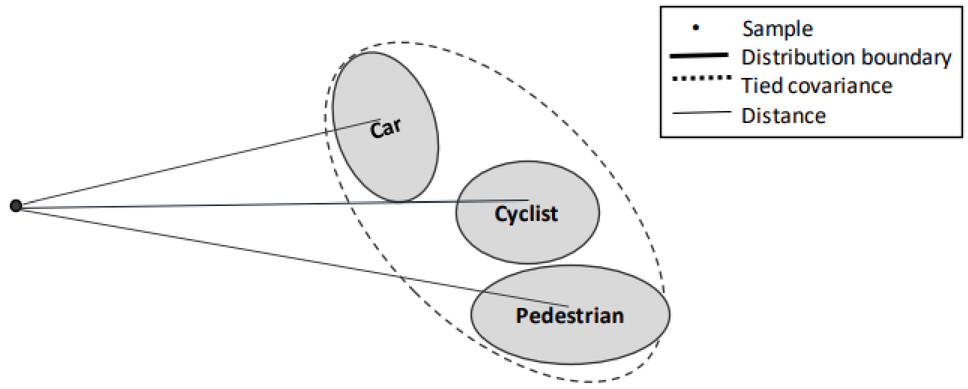


Figure 2.4: Mahalanobis distance.

OC-SVM: This method learns a decision boundary between in-distribution data and the origin in Hilbert space while maximizing the distance between the origin and the decision boundary [62]. The OOD score is the distance from the sample to the decision boundary. An ID sample has a large distance while an OOD sample has a small distance.

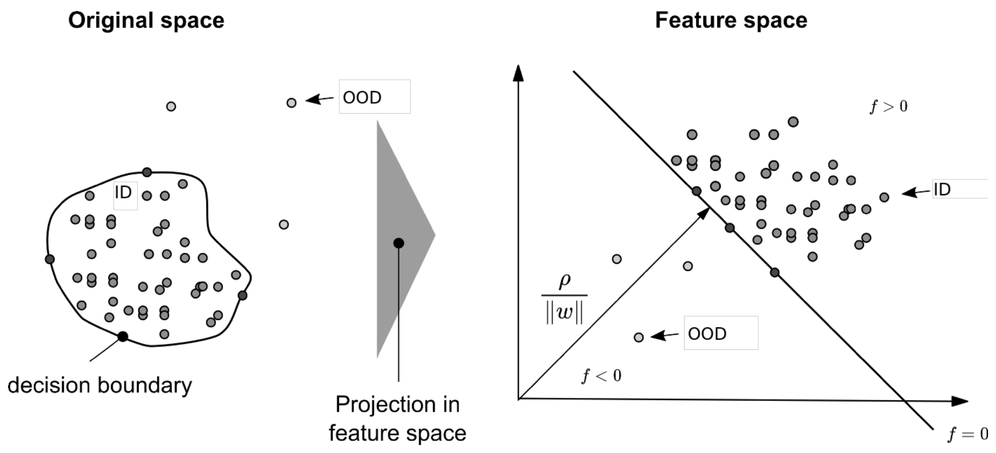


Figure 2.5: OC-SVM. Modified from Dominik Polzer[1].

Normalizing flows: This method is a powerful tool for density estimation [59] because it can output exact log probabilities of the inputs. The method aims to learn a series of differentiable bijections that map complex distributions of observed data to simple distributions of latent variables. Given an observed data variable $\mathbf{x} \in X$, a simple prior probability distribution p_Z (e.g., a Gaussian distribution) on a latent variable $\mathbf{z} \in Z$, and a bijection $f : X \to Z$ (with $g = f^{-1}$), the log-likelihood of an observed sample $\mathbf{x}$ is computed by the change of variable formula:

$$\log\left(p_X(\mathbf{x})\right) = \log\left(p_Z(f(\mathbf{x}))\right) + \log\left(\left|\det\left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^T}\right)\right|\right) \tag{2.3}$$

where $\frac{\partial f(\mathbf{z})}{\partial \mathbf{x}^T}$ is the Jacobian of f at $\mathbf{x}$.

The f function is carefully designed to be invertible, and its Jacobian must be easy to compute. This function can be learnt by deep neural networks. The objective is minimizing the average of negative log-likelihood over the population of observed samples:

$$L = -\frac{1}{|D|}\sum_{\mathbf{x}\in D} \log\left(p_X(\mathbf{x})\right) \tag{2.4}$$

[1] https://towardsdatascience.com/a-comprehensive-beginners-guide-to-the-diverse-field-of-anomaly-detection-8c818d153995

A well-trained normalizing flows model assigns high log-likelihoods for ID inputs and low log-likelihoods for OOD inputs. Therefore, we can use predicted log-likelihoods as OOD scores. We choose RealNVP [16], an efficient variant of normalizing flows for our work.

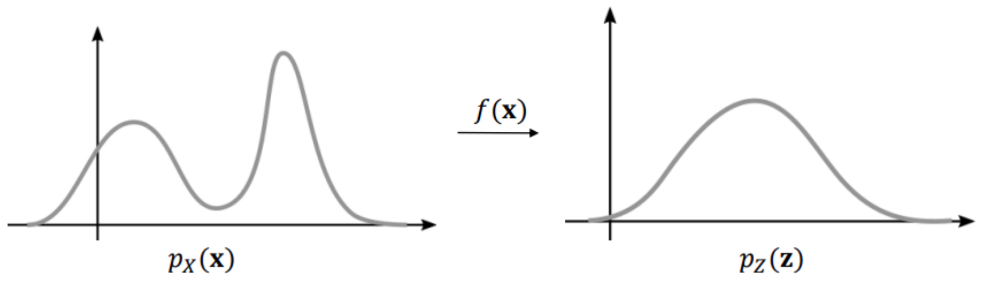


Figure 2.6: Normalizing flows.

2.4 Evaluation metrics

We adopt the evaluation metrics proposed by Hendrycks et al. [25]. AP is a metric for object detection, and AUROC, AUPR, detection error, and FPR @ 95% TPR are metrics for OOD detection.

AP at a certain intersection-over-union (IoU) threshold (e.g., $\text{AP}_{0.5}$ is AP at IoU = 0.5) is a common evaluation metric for object detection. Average Precision is measured by averaging precision at a number of equally spaced recall values ranging from 0 to 1.

AUROC is the measure of the area under the plot of true positive rate (TPR) vs. false positive rate (FPR). It measures the probability that an OOD example is assigned a lower score than an ID example. TPR is computed as $\text{TPR} = \text{TP}/(\text{TP} + \text{FN})$ and FPR is computed as $\text{FPR} = \text{FP}/(\text{FP} + \text{TN})$, where TP, FP, TN, and FN denote True Positive, False Positive, True Negative, and False Negative, respectively.

AUPR is the Area under the Precision-Recall curve. The PR curve is a graph showing the $precision = \text{TP}/(\text{TP} + \text{FP})$ and $recall = \text{TP}/(\text{TP} + \text{FN})$ against each other. The metric AUPR-In and AUPR-Out are the area under the precision-recall curve where ID and OOD samples are specified as positives, respectively.

Detection Error D_e measures the misclassification probability when TPR is 95%. The formula to calculate $D_e = 0.5(1 - \text{TPR}) + 0.5\text{FPR}$, where we assume that both positive and negative examples have equal probability of appearing in the test set.

FPR @ 95% TPR is the probability that an OOD (negative) example is correctly identified when the TPR is 95%.

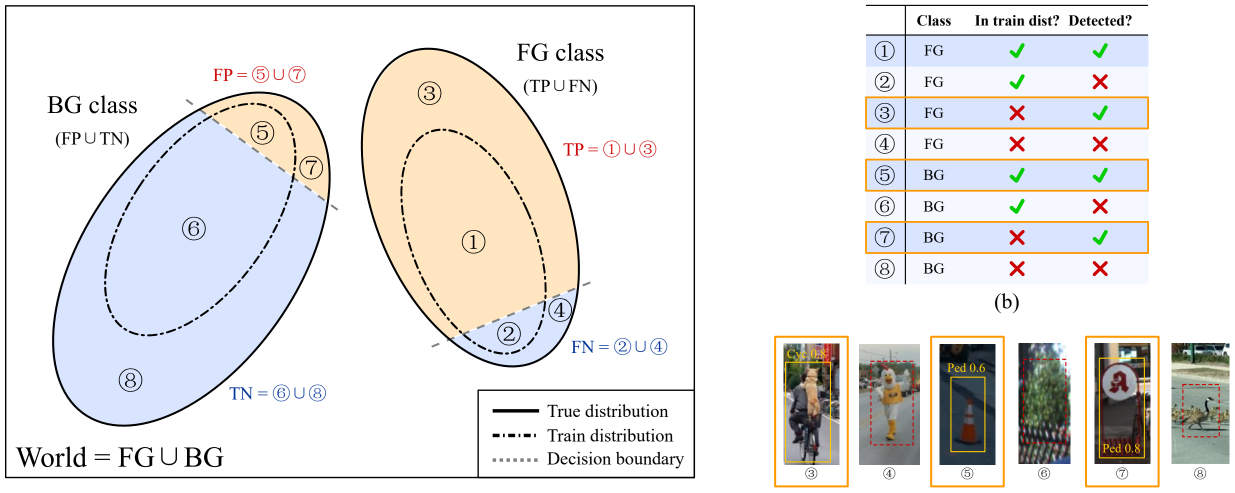
Chapter 3

Out-of-Distribution in Object Detection

This chapter focuses on establishing a definition of out-of-distribution in object detection. Subsequently, it proposes a technique to generate 3D LiDAR OOD datasets for benchmarking OOD detection methods in 3D LiDAR-based object detection. Finally, it shows how OOD datasets are created, and it covers the statistics and characteristics of the created OOD datasets.

3.1 Definition

To define OOD inputs for object detection, we consider the distributions of foreground (FG) and background (BG) objects as illustrated by the Venn diagram in Figure 3.1a. We assume the existence of a true distribution for both FG and BG objects as shown by the solid-line ovals. The training distributions of FG and BG classes are visualized as subsets of the true distributions by the dashed-line ovals to indicate that some FG objects may be underrepresented or absent in the training distributions compared to the true distributions. This distributional shift may have many causes, including biases in how the training data is collected or the evolution of the target deployment environment. A detector trained over the training distributions learns decision boundaries shown by dashed lines. For simplicity, we show only one FG class and one BG class. In object detection, BG is usually very diverse and consists of many BG classes that are implicit in the task. We use these FG and BG distributions and decision boundaries to categorize each sample point (object).



	Class	In train dist?	Detected?
①	FG	✓	✓
②	FG	✓	✗
③	FG	✗	✓
④	FG	✗	✗
⑤	BG	✓	✓
⑥	BG	✓	✗
⑦	BG	✗	✓
⑧	BG	✗	✗

Figure 3.1: Different types of OOD objects for object detection. (a) A visualization of FG and BG object distributions. (b) Classification of objects in object detection. (c) Examples of different types of OOD objects wrt. a FG class (cyclist for ③ and pedestrian for ④-⑧).

If an object is detected and drawn from either the FG training distribution or the unseen part of the FG true distribution, it is called True Positive (TP) (the areas indicated by ① and ③). The detection in ③ is due to generalization. If an object is not detected and drawn from either the FG training distribution or the unseen part of the FG true distribution, it is called False Negative (FN) (the areas indicated by ② and ④). A similar logic can be applied to background classes. If an object is detected and drawn from either the BG training distribution or the unseen part of the BG true distribution, it is called False Positive (FP) (the area indicated by ⑤ and ⑦). If an object is not detected and drawn from either the BG training distribution or the unseen part of the BG true distribution, it is called True Negative (TN), (the area indicated by ⑥ and ⑧).

Figure 3.1b summarizes our categorization of objects in object detection. We consider types ③-⑧ as OOD inputs with respect to the FG class. Objects in ⑤-⑧ are in the BG class, and thus OOD with respect to FG by definition. Objects in ③-④ are truly FG but underrepresented or absent in the FG training distribution. An example of each type is shown in Figure 3.1c. Types ③, ⑤, and ⑦ are objects for which there is a prediction, while there is no prediction for the remaining three types. Furthermore, note that we do not consider objects of one FG class as OOD for another FG class in this work.

Although it is important to detect all OOD types, we specifically focus on OOD types ③, ⑤ and ⑦ for which there are predictions. Type ③ is an unusual foreground object which is detected. Even though it is a TP, such objects can have erratic behavior or they may need to be actively sought for in the field to improve the quality of the training distribution. Types ⑤ and ⑦ are background objects which are classified as foreground objects. Detecting type ④ OOD objects, along with type ② ID objects, is covered in the field of FN detection (e.g., [57, 84]). Thus, we only focus on identifying whether or not a detected object is an OOD input and leave detecting other OOD types as future work.

3.2 OOD Object Generation

We propose a method to generate point clouds with OOD objects that can be used to evaluate OOD methods for LiDAR-based 3D object detection. The method generates such test inputs by inserting synthetic or real OOD objects into real point clouds. The inserted OOD objects must satisfy the criteria of the target types ③, ⑤, and ⑦, i.e., that they are unusual wrt. to the training set or their class must not overlap with the foreground classes, and are detected by the model with a score τ or higher. This is because low confidence predictions can be easily filtered by the object detector using its own score threshold. The resulting point clouds are tailored for a specific model; however, we observe that the synthetic or real OOD objects harvested from existing datasets for this work tend to be misdetected by different models or even detectors trained on KITTI.

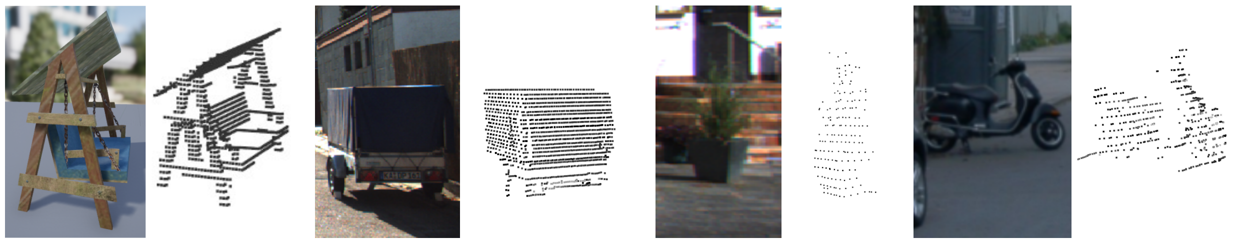

(a) Carla object (swing couch) (b) KITTI Ignored (misc) (c) KITTI FP (potted plant) (d) Waymo object (scooter)

Figure 3.2: Samples of each type of OOD objects intended for OOD dataset creation.

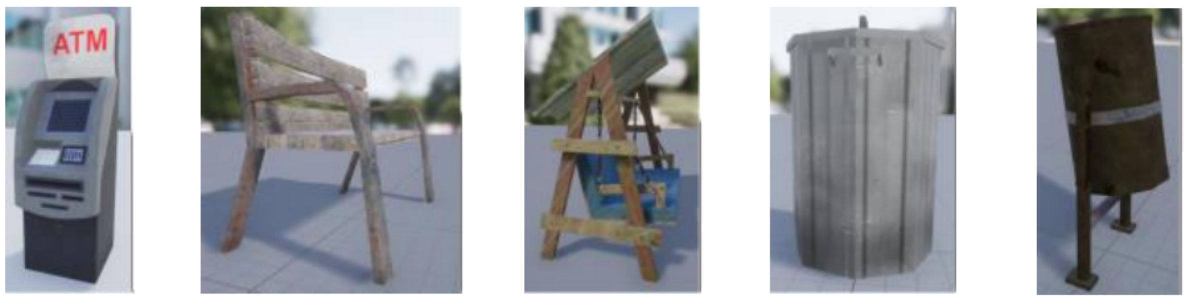


Figure 3.3: Examples of Carla objects - type ⑦ OOD.

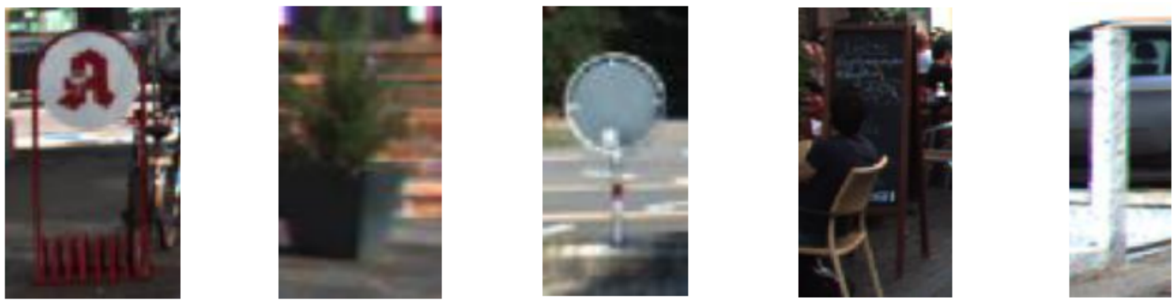


Figure 3.4: Examples of KITTI FP objects. Bike rack is type ⑦ OOD; the remaining objects are type ⑤ OOD. A PointPillars detector trained on KITTI recognizes these objects as Pedestrians.

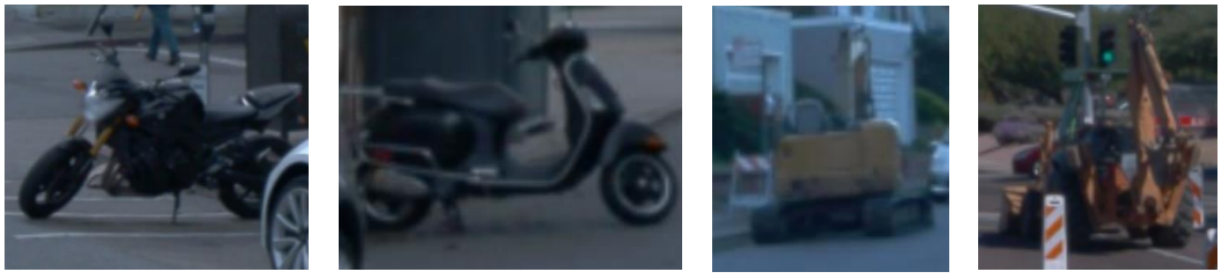

(a) Motorcycle (b) Scooter (c) Excavator (d) Digger

Figure 3.5: Examples of Waymo objects - type ⑦ OOD. A PointPillars detector trained on KITTI recognizes these objects mostly as cars. Some instances of Motorcycle and Scooter are recognized as Cyclist and Pedestrian. Excavator is sometimes recognized as a group of Pedestrians.

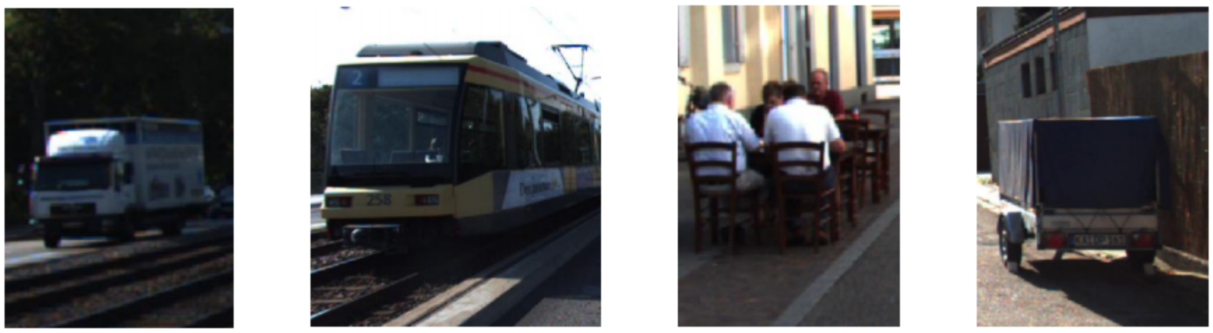

(a) Truck (b) Tram (b) Person sitting (b) Misc. (Trailer)

Figure 3.6: Examples of KITTI ignored objects. These objects are treated as background in training, and therefore can be considered as type ⑤ OOD. A PointPillars detector trained on KITTI recognizes these instances of Truck, Tram, and Trailer typically as Car. Person sitting is sometimes recognized as Pedestrian.

Given a bank of OOD object scans, such as in Figure 3.2, we randomly pick an OOD object from the bank and a data frame form the validation set of the target dataset such as KITTI and insert the object into the frame by randomly varying the object's azimuth to increase the location diversity. We preserve the object distance to the sensor to keep the point cloud sparsity consistent with its location, and rely on the bank to have object scans at different distances [81]. We also adjust the point intensities to match the distribution of the target frame. We do not consider "shadows" caused by occlusions, since the detector is already exposed to such object insertions as part of the commonly performed ground-truth object augmentation. We also check if the inserted object overlaps with any existing ground-truth objects or predictions in the original data frame or if its score is below τ, in which case the object is rejected; otherwise, the frame with the inserted object is saved. We then repeat the process (see Section 3.3 for more details). Figure 3.7a shows examples of the Carla objects inserted into sample scenes. Figure 3.7b shows that the target model misclassifies the inserted OOD objects, e.g., bench as a car and ATM as a pedestrian.

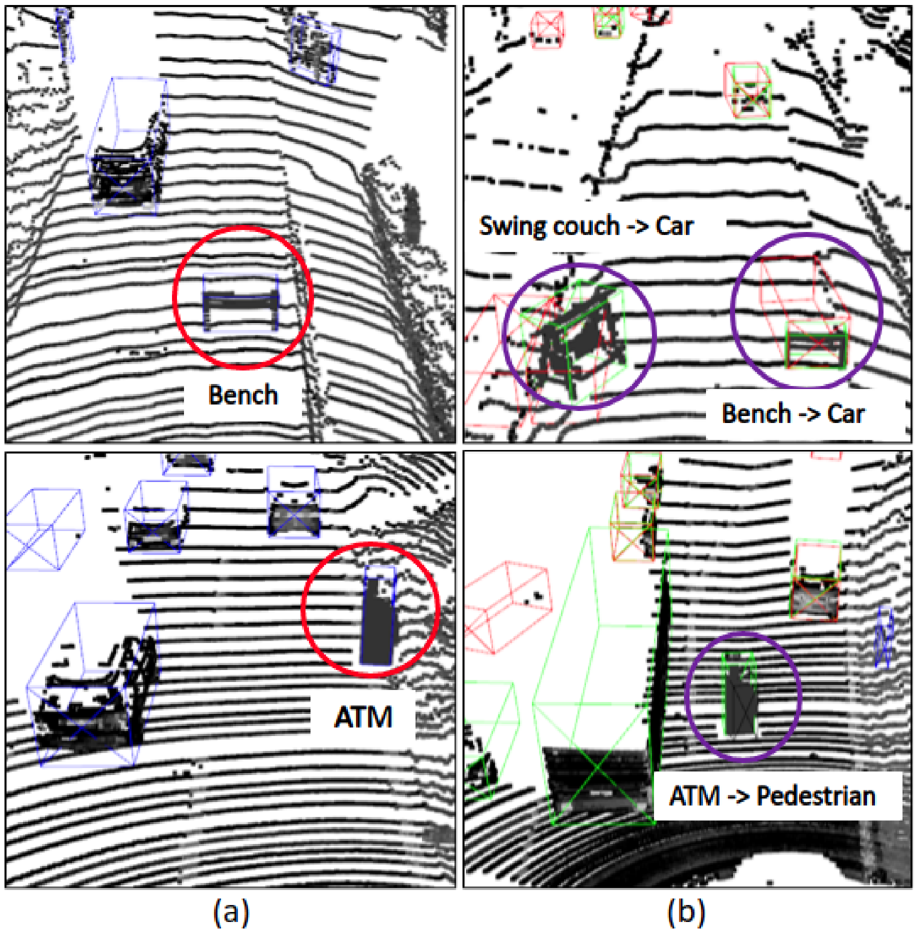


Figure 3.7: Examples of Carla objects inserted into real point clouds. (a) Inserted ATM and Bench (in red circles). (b) Carla objects misclassified as foreground objects (in purple circles), e.g., a Swing couch detected as a Car. Ground truth boxes in green, Car prediction boxes in red, and Pedestrian prediction boxes in black.

Note that the dataset splitting method [50] for generating an OOD dataset for object detection is not applicable in our context. In contrast to camera image datasets for object detection, such as COCO [43] or PASCAL [19], datasets for LiDAR-based 3D object detection in autonomous driving, such as KITTI [22], Waymo [71], and NuScenes [7], contain only a handful of classes. Furthermore, unknown unknown objects in [50] are not guaranteed to be detected by any model.

3.3 OOD Dataset Description

In this section, we present Algorithm 1 to generate an OOD dataset for a model from an OOD object database and an ID dataset. It takes an ID dataset, an OOD object database, an object detection model, and two parameters: maximum number of trials γ_{max} and maximum number of objects ζ_{max} as inputs. The output is an OOD dataset that includes ID objects from the ID dataset and OOD objects from the OOD object database. We also provide a diagram in Figure 3.8 describing the OOD dataset generation process.

Algorithm 1 OOD dataset generation

```
Input: ID dataset ($I$), OOD object database ($O$), detection model ($M$), maximum number of attempt $\gamma_{max}$, maximum
    number of OOD objects ($\zeta_{max}$)
Output: OOD dataset ($I$)
 1: function OODGenerator ($I$,$O$,$M$,$\gamma_{max}$, $\zeta_{max}$ )
 2:     for $c \in classes(O)$ do
 3:         $\zeta \leftarrow 0$
 4:         for $f \in I$ and $\zeta \leq \zeta_{max}$ do
 5:             $P \leftarrow M(f)$                                    ▷ obtain predictions
 6:             $\gamma \leftarrow 0$
 7:             while $\gamma \leq \gamma_{max}$ do
 8:                 $o \leftarrow rand\_obj(c)$                        ▷ select an object $\in c$
 9:                 $a \leftarrow rand\_angle()$                       ▷ select a relative angle
10:                 $f' \leftarrow insert(f, o, a)$                    ▷ add object to frame
11:                 if $IoU(o$, GT$(f)$, $P) \neq 0$ or $o$ is not in the FOV range then
12:                                                        ▷ GT returns ground truth objects
13:                     $\gamma \leftarrow \gamma + 1$
14:                     continue
15:                 end if
16:                 $P' \leftarrow M(f')$                              ▷ obtain predictions
17:                 if $o \notin P'$ then
18:                     $\gamma \leftarrow \gamma + 1$
19:                     continue
20:                 else
21:                     $I \leftarrow update(f', I)$                   ▷ update the dataset
22:                     $\gamma \leftarrow \gamma + 1$
23:                     $\zeta \leftarrow \zeta + 1$
24:                     break
25:                 end if
26:             end while
27:         end for
28:     end for
29:     return $I$
30: end function
```

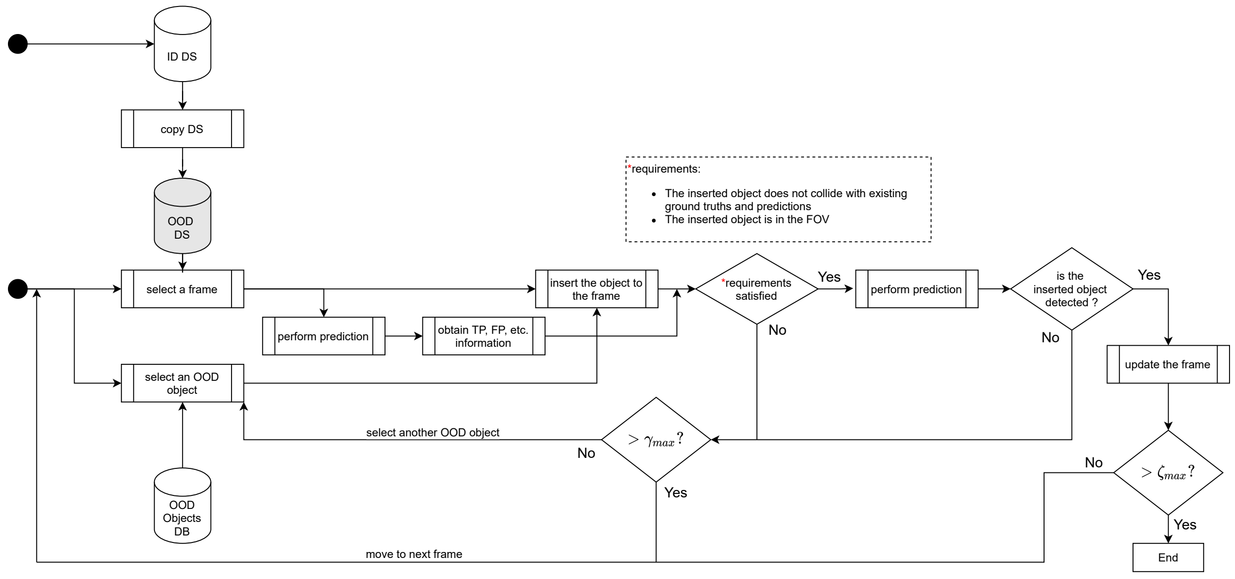

ID DS
copy DS
OOD DS
*requirements:
The inserted object does not collide with existing ground truths and predictions
The inserted object is in the FOV
select a frame
insert the object to the frame
*requirements satisfied
Yes
perform prediction
is the inserted object detected ?
Yes
No
No
perform prediction
obtain TP, FP, etc. information
update the frame
select an OOD object
select another OOD object
$> \gamma_{max}$?
No
Yes
$> \zeta_{max}$?
No
Yes
OOD Objects DB
move to next frame
End

Figure 3.8: OOD dataset generation diagram

We utilize different sources to gather OOD objects to be inserted into the KITTI dataset [22]. For synthetic objects, we use the Carla simulation [17]. We run LiDAR simulation to obtain the point cloud of objects (Figure 3.3) at various angles and distances. For real objects, we use the KITTI ignored objects (Figure 3.6), the KITTI False Positive (FP) objects (Figure 3.4), and weird vehicle objects (Figure 3.5) from the Waymo dataset [71]. For Carla objects, we set the intensity values for the points to the median of the KITTI intensities. For Waymo objects, we transform the original intensities using *tanh* then adjust the intensity values so that the mean and variance under log scale is the same as the KITTI intensities under log scale.

The KITTI FP objects are background objects classified as Pedestrians. We manually label and categorize them into classes such as *potted plant, bike rack, low traffic sign, sidewalk sign*, and *thin sign*. Furthermore, we verify if KITTI FP objects appear in the KITTI training dataset. The Table 3.1 shows that except bike rack, other KITTI FP objects do appear relatively frequently in the KITTI training dataset. Therefore, KITTI FP objects are a good candidate for the type ⑤ OOD objects.

Class	Bike rack	Sidewalk sign	Traffic sign	Potted plant	Thin sign
Quantity	0	625	295	298	60

Table 3.1: The number of times that KITTI FP objects appear in the KITTI training dataset

We manually identify objects among the Waymo vehicle class that are not FG in KITTI, including *motorcycle, scooter, digger, and excavator*. To ease the annotation procedure, we first cluster all collected vehicles in the Waymo validation dataset into either a *main* cluster or an *outlier* cluster using the DBSCAN algorithm and only annotate the vehicles clustered into the *outlier* cluster. We run four separate DBSCAN clusterings on the following four sets of features: $\text{PCA}([l, w, h, l/w, l/h, w/h, lw, lh, wh], \text{ndim} = 2)$, $[l, w]$, $[l, h]$, and $[w, h]$, where l, w, and h correspond to the length, width, and height of the collected vehicle, respectively. We then define the global outlier cluster as $\{\text{vehicles } v | v \text{ is an outlier in at least one clustering}\}$. Next, we manually collect the vehicles in the global outlier cluster that are likely to be OOD objects for the KITTI dataset and assign a class label to each collected out-of-distribution vehicle. Finally, we collect the LiDAR point cloud of each annotated outlier Waymo vehicle across multiple frames (every ten frames) in the scene where the vehicle exists.

In Table 3.2, we list the number of objects per class from each OOD database that we select to create OOD datasets. We select these classes because, based on our experiment, they are likely to be misclassified as foreground objects when we insert them into the KITTI validation dataset. For each class, we insert 300 objects into an OOD dataset. The number of attempted frames is the number of frames the algorithm must go through to insert 300 objects. In each frame, the maximum number of trials is 100. If the number of trials exceeds the maximum number of trials, we abandon the frame and move on to the next frame. Table 3.2 also shows the statistics of inserting objects from each database to create OOD datasets. Injection failures and detection failures are the numbers of failed trials averaged over the number of attempted frames. The higher injection failures, the higher chance that an object overlaps with existing objects in a frame. The higher the detection failures, the lower likelihood that an object is detected when we insert it into a frame. For example, in Carla OOD datasets, bench and swing couch have high injection failures because they are large objects. Thus, they easily overlap with existing objects in a frame. These objects also have high detection failures because a model cannot easily detect them. Furthermore, by comparing the statistics among datasets, we can conclude that it is easier to insert Waymo and KITTI FP objects than to insert Carla objects. It makes sense

			Base model				Contrastive model				
Type	Class	# instances in DB	# injection failures	# detection failures	# attempted frames	# injected objects	# injection failures	# detection failures	# attempted frames	# injected objects	# common OOD
Carla	ATM	10	17.51	43.52	545	300	16.20	40.25	489	300	289.67
	Bench	44	25.97	64.74	2342.67	289.33	22.61	55.70	1229.67	300	318.67
	Swing couch	25	29.37	60.96	2276.33	267.67	24.38	48.01	1631	300	200.67
	Trash can 1	5	18.65	49.22	638.33	300	15.43	39.92	470.67	300	537.67
	Trash can 2	5	4.02	9.38	304.67	300	2.39	5.40	302	300	582
KITTI Ignored	Tram	287	-	-	-	-	-	-	-	-	73.33
	Truck	606	-	-	-	-	-	-	-	-	69.67
	Person sitting	166	-	-	-	-	-	-	-	-	105.33
	Misc	636	-	-	-	-	-	-	-	-	138
KITTI FP	Bike rack	11	0.71	0.04	300	300	0.93	0.09	300	300	599
	Low traffic sign	10	0.33	0.016	300	300	0.22	0.02	300	300	598.67
	Sidewalk sign	33	0.57	0.008	300	300	0.37	0.02	300	300	599
	Potted plant	9	0.89	0.03	300	300	0.69	0.03	300	300	599.33
	Thin sign	3	0.54	0.084	300	300	0.36	0.06	300	300	598.67
Waymo	Motorcycle	456	4.07	0.81	300	300	3.80	0.74	300	300	512
	Scooter	80	9.68	1.39	300	300	8.18	1.01	300	300	385.67
	Digger	49	4.57	1.03	300	300	4.48	1.00	300	300	585.67
	Excavator	44	13.35	3.04	302.33	300	11.67	2.65	300.33	300	564.33

Table 3.2: Statistics for OOD objects. In total, we generate six datasets, one for each model/seed combination. The injection statistics (# injection failures, # detection failures, # attempted frames, # injected objects) and # common OOD are averaged over three seeds.

because LiDAR sensors in both Waymo and KITTI datasets have a similar specification. In contrast, the Carla objects are synthetic data from a simulation. In addition, it is easier to insert Carla objects using the contrastive model than the base model. The observation is opposite for the Waymo and KITTI FP objects.

By collecting objects from different sources we ensure that our inserted objects are diverse. Figure 3.9 shows the cumulative distribution of softmax scores, predictive entropy, aleatoric entropy, and mutual information of the inserted objects (i.e., OOD) and foreground objects (i.e., ID). The figure shows that our dataset covers objects with various ranges of uncertainties, sofmax predicted scores and predictive entropy. The detection threshold τ is set to 0.3 for all models.

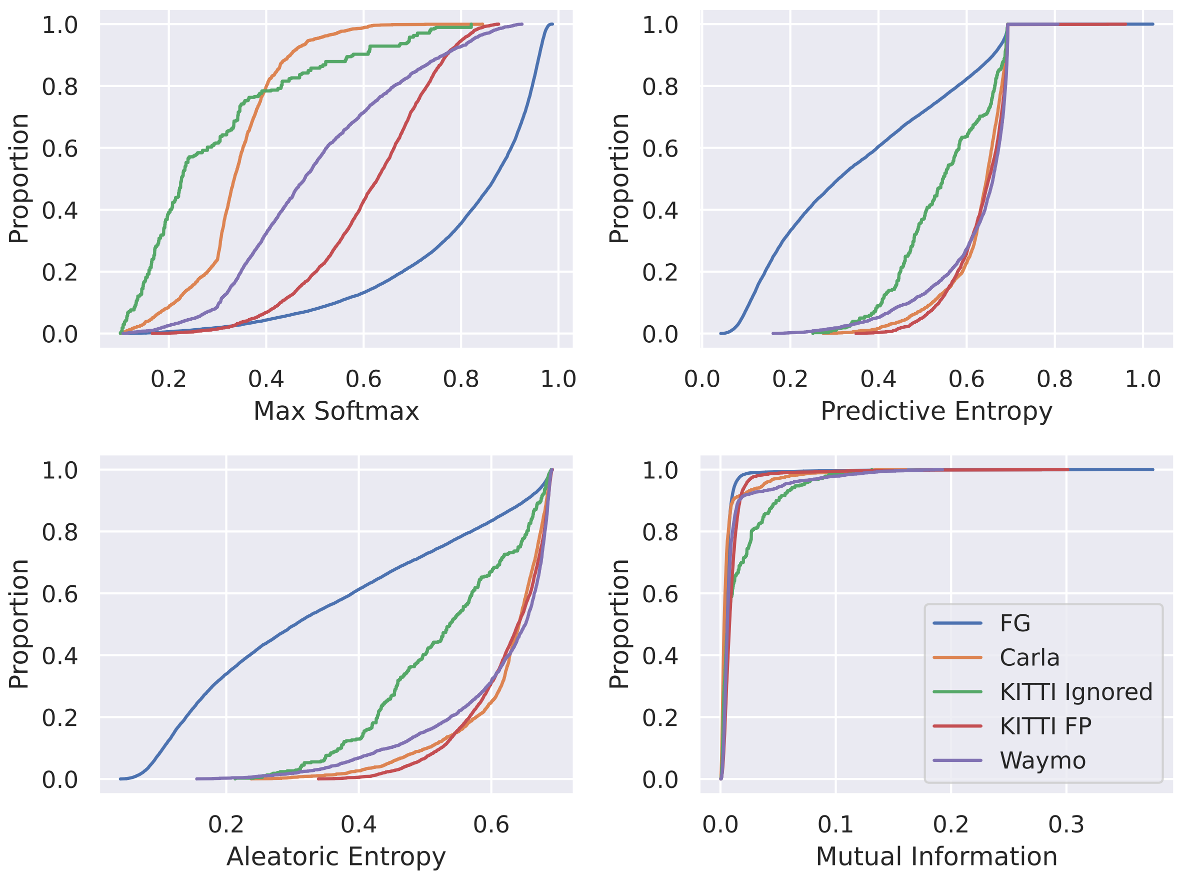


Figure 3.9: The CDFs of max-softmax scores, predictive uncertainty, aleatoric uncertainty and epistemic uncertainty of OOD object datasets.

Chapter 4

OOD Detection for 3D Object Detection

This chapter proposes a feature extraction method for OOD detection in object detection and a method to investigate the effects of supervised contrastive learning on the quality of feature maps for OOD detection.

4.1 Feature Extraction

Feature extraction for the object detection task is different from classification since each input can produce an arbitrary number of predictions. Thus, using the entire feature map may inadvertently include features from other predictions or environmental objects. Instead, we identify a single feature vector within each feature map for each prediction.

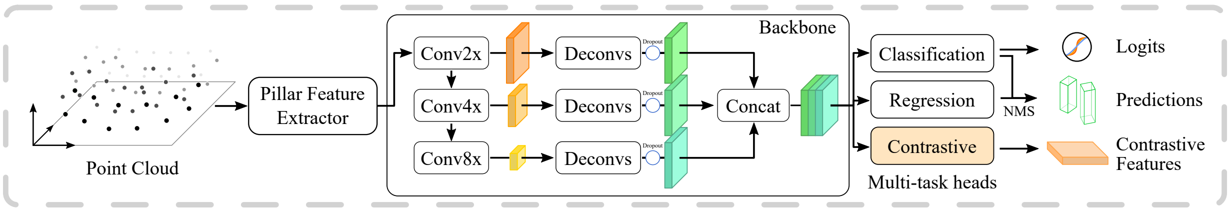


Figure 4.1: The PointPillars architecture extended with dropout and a contrastive-loss head.

In this work, we use the single-stage anchor-based 3D object detector PointPillars [35] (Figure 4.1). In the PointPillars-style architectures [45], a set of anchor boxes is predefined for each BEV pixel location in the final backbone feature map (after concatenation in Figure 4.1). During training, labels are assigned to the anchor boxes based on the overlap with FG objects. We then use the pixels in the final feature map with positive anchors as associated feature vectors, used for training the OOD detection methods. It is possible for one object to correspond to multiple feature vectors during training. We treat them as independent training samples and do not perform any aggregation. During testing, we identify the anchor from which the final prediction is regressed and use its corresponding pixel in the final feature map as the feature vector for the prediction.

In addition to the final feature map, we also extract features from three intermediate layers after each convolution block in the backbone (conv2x, conv4x, and conv8x in Figure 4.1) as well as the feature map after contrastive mapping (will be discussed in Section 4.2). For feature maps that are smaller than the final feature map where the anchor boxes are defined, we apply nearest neighbors upsampling and use the aforementioned method to extract feature vectors for training and testing.

4.2 Contrastive Learning

Contrastive learning is a popular approach that allows the model to learn better feature embeddings. It has been used in OOD detection and has shown good results in both OOD detection and downstream tasks [69, 72, 77, 63].

We incorporate a supervised contrastive loss into the training of our object detectors to investigate its effectiveness on OOD detection for 3D object detection [31]. We add an additional contrastive head after the backbone layer that produces a contrastive feature map. The supervised contrastive loss is applied to the features in the contrastive feature map shown in Figure 4.1. We train the contrastive loss objective jointly with the classification and regression losses. The labels for contrastive head is consistent with the classification head target. Due to the large number of background pixels that do not generate any predictions, we subsample a small portion of background feature vectors according to the predicted foreground probability from the classification head. In other words, we consider background feature vectors that are most likely to generate foreground predictions. This technique is known as hard negative mining [45].

Chapter 5

Experimental Setup

This chapter focuses on the details of the experiments, such as datasets, methods, and training parameters for the 3D detection models and OOD detection methods.

5.1 Dataset

We use the KITTI dataset [22], which has 7481 point cloud frames with 3D annotated bounding boxes. We split the official dataset into training and validation datasets (3712 and 3769 samples, respectively) [35]. We use the training dataset for training object detectors and extracting feature maps for the OOD detection methods. We use the validation dataset for calculating metrics. Figure 5.1 shows an example from the KITTI dataset.

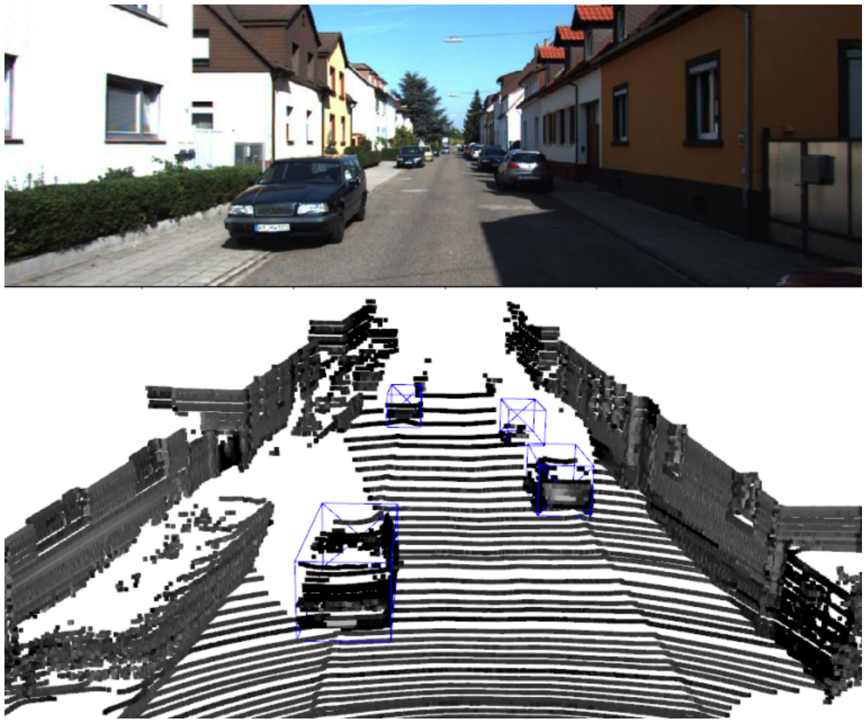


Figure 5.1: An example from KITTI dataset. Ground truth objects are inside bounding boxes.

5.2 Object detection models

We train two object detectors based on the PointPillars architecture (Figure 4.1) for detecting three foreground classes: *Car, Pedestrian*, and *Cyclist*. The first model, referred to as the base model, is a modified version of the standard PointPillars that allows for uncertainty estimation. We add one dropout layer after each deconvolution block in the backbone. The base model also outputs softmax distribution instead of individual sigmoid scores.

The second model, referred to as the contrastive model, is the base model extended with supervised contrastive loss [31]. We add six additional 1x1 convolution layers after the backbone to produce a contrastive feature map with 64 channels. Both models are trained for 120 epochs with a batch size of 6. We follow the OpenPCDet's recommendations for other hyperparameters [64]. Each model has been trained three times with randomly

Model	Car	Pedestrian	Cyclist	mAP
Base	78.38	48.72	62.75	63.28
Contrastive	**79.10**	**50.96**	**64.47**	**64.72**
Δ	+0.72	+2.24	+1.72	+1.44

Table 5.1: The AP of the base and contrastive models on moderate difficulty samples of the KITTI validation dataset averaged over three runs.

initialized weights. The average precision of these models on the KITTI validation dataset is shown in Table 5.1. We can observe a consistent improvement over all classes. We consider investigating different models and datasets as one of our future works due to the high computational cost of training LiDAR-based 3D object detectors.

5.3 OOD evaluation datasets

We use four OOD datasets: Carla, KITTI ignored, KITTI FP, and Waymo OOD object datasets, which are generated and described in Section 3.3 to benchmark OOD methods.

5.4 OOD detection methods

We benchmark five OOD methods: Max-softmax, uncertainty estimates, Mahalanobis distance, OC-SVM and normalizing flows (RealNVP). For object detection model each model (base and contrastive), we extract features of conv2x, con4x, con8x, backbone, and contrastive layers to train OC-SVM, Mahalanobis distance, and normalizing flows. For the OC-SVM method, we train one OC-SVM with SGD per FG class and use the highest score as the OOD score. We set $\nu = 0.01$, $\gamma = 2.0$ and train with a batch size of 64 for 5 epochs. For Mahalanobis, we apply online mean/covariance update with a batch size of 64 for 5 epochs. RealNVP is trained with a batch size of 8 for 2320 steps. For uncertainty estimates based on MC dropout, the dropout probability is set to 0.5.

Chapter 6

Results and Discussion

This chapter presents quantitative and qualitative results of our experiments, along with a discussion of insights. It concludes with a discussion of the impact of OOD detection on object detection performance when using OOD detection to reject an object detection (i.e., object detection with a rejection option).

6.1 Quantitative results

To evaluate each OOD detection method and properly compare the base and contrastive models, we report results for the common objects that both models detect (Tables 6.1 and 6.2). We apply class-balanced resampling during the evaluation to remove any potential biases. For Mahalanobis distance, OC-SVM, and normalizing flows, we report the results of the layer with the best overall performance for the base model. We first use AU-ROC and FPR at 95% TPR as our primary metrics to make the results easier to follow. The full results with other metrics and layers are in following paragraphs. In total, we repeat dataset generation, model training, and evaluation process three times and average the results.

Overall, normalizing flows with backbone features and max-softmax achieve the best OOD detection performance (Table 6.1). The performance of OC-SVM and epistemic uncertainty (mutual information) via MC-dropout are significantly lower than other methods, suggesting that they may not be suitable for the OOD detection in the 3D object detection context.

Supervised contrastive learning can improve the OOD detection performance in 3D object detection for certain OOD detection methods. In particular, max-softmax, uncertainty estimations, and Mahalanobis distance benefit the most from the contrastive learning.

The best performing layer for OOD detection can vary for the different OOD detection methods. In our experiments, the backbone features are optimal for both Mahalanobis distance and normalizing flows, whereas OC-SVM works best with conv8x features. Using logits with Mahalanobis as proposed by [38] is out-performed by the backbone features by a large margin. The same result is observed for the OC-SVM method [1], which shows that early layers provide a better separation between ID and OOD objects with OC-SVM in image classification. This suggests that the optimal layer for each method can depend on the specific model architecture or feature map distribution. Existing results for image classification tasks may not transfer to other tasks with specialized architectures such as LiDAR-based 3D object detection.

The performance of each OOD detection method can vary significantly for different types of OOD objects. For instance, normalizing flows has 9% higher AUROC and 20% lower FPR compared to max-softmax for KITTI FP objects (Table 6.2c) and has comparable results for Carla and Waymo objects (Tables 6.2a and 6.2d). However, it becomes much worse than max-softmax for KITTI ignored objects with 10% difference in both AUROC and FPR (Table 6.2b). This can also be observed for other OOD methods. This variation can be due to the noticeable differences in uncertainty and max-softmax probability between different OOD objects (Figure 3.9). The main observations are: i) the max-softmax scores and uncertainty estimations of the OOD objects can vary across different OOD types, which affects the performance of these methods, ii) Mahalanobis distance applied to the backbone layer is relatively stable across different types of OOD. However, for KITTI FP objects with both high max-softmax score and high uncertainty, its performance becomes significantly worse, and iii) for normalizing flows with backbone layer, OOD objects with higher aleatoric and lower epistemic uncertainty (Carla and KITTI FP) have better performance.

In our experiments, we do not have a single OOD detection method that works well for all types of OOD objects. For Carla and KITTI ignored objects, max-softmax is the optimal OOD detection method. For KITTI FP objects, normalizing flows outperforms other methods by a large margin. For Waymo objects, Mahalanobis distance and max-softmax are the best for base and contrastive model, respectively. This shows that depending on the type of OOD objects and their characteristics, the optimal OOD detection method could be different. A similar observation in the classification context is also noted by Kaur et al. [29], who suggest that a combination of multiple OOD detection methods may be needed to cover different types of OOD. We leave in-depth investigation and evaluation of combined OOD detection methods for future work.

OOD Method	Layer	AUROC ↑ Base	AUROC ↑ Ctrst	AUROC ↑ Δ	FPR @ 95 TPR ↓ Base	FPR @ 95 TPR ↓ Ctrst	FPR @ 95 TPR ↓ Δ
Max Softmax	**-**	89.95	**89.97**	**0.02**	33.83	**31.86**	**-1.97**
Predictive Entropy	**-**	83.49	84.80	**1.32**	39.60	34.54	**-5.05**
Aleatoric Entropy	**-**	83.09	84.65	**1.56**	41.77	36.41	**-5.37**
Mutual Information	**-**	60.48	60.70	**0.23**	97.07	96.51	**-0.56**
Mahalanobis	**backbone**	83.77	84.56	**0.80**	51.01	43.92	**-7.10**
OCSVM	**conv8x**	65.44	62.91	-2.53	66.20	70.31	4.11
RealNVP	**backbone**	**90.28**	88.30	-1.98	**26.72**	33.14	6.42

Table 6.1: The OOD detection results for all OOD objects such that each detected by both models. For each metric, we underline the best performing OOD method. We also show the performance difference Δ between contrastive model and base model, with bold numbers indicating contrastive model has better performance. All results are averaged over three sets of experiments.

OOD Method	Layer	AUROC ↑ Base	Ctrst	Δ	FPR @ 95 TPR ↓ Base	Ctrst	Δ
Max Softmax	**-**	**<u>96.04</u>**	**<u>96.15</u>**	**0.11**	**<u>10.39</u>**	**<u>10.23</u>**	**-0.16**
Predictive Entropy	**-**	85.34	87.12	**1.78**	35.55	30.10	**-5.45**
Aleatoric Entropy	**-**	85.30	87.37	**2.06**	37.87	30.76	**-7.11**
Mutual Information	**-**	45.04	43.18	-1.86	98.75	98.66	**-0.09**
Mahalanobis	**backbone**	88.34	89.88	**1.54**	37.49	35.78	**-1.71**
OCSVM	**conv8x**	66.49	63.12	-3.37	63.49	70.01	6.53
RealNVP	**backbone**	93.63	92.50	-1.13	19.90	22.80	2.90

(a) Carla

OOD Method	Layer	AUROC ↑ Base	Ctrst	Δ	FPR @ 95 TPR ↓ Base	Ctrst	Δ
Max Softmax	**-**	**<u>95.58</u>**	**<u>96.06</u>**	**0.48**	**<u>21.23</u>**	**<u>20.80</u>**	**-0.43**
Predictive Entropy	**-**	78.65	80.41	**1.76**	41.66	36.10	**-5.55**
Aleatoric Entropy	**-**	78.24	80.03	**1.79**	43.93	38.59	**-5.33**
Mutual Information	**-**	52.09	55.94	**3.86**	99.13	98.58	**-0.55**
Mahalanobis	**backbone**	86.25	84.09	-2.16	54.35	50.86	**-3.48**
OCSVM	**conv8x**	66.95	63.67	-3.28	62.58	69.03	6.45
RealNVP	**backbone**	85.81	84.02	-1.79	31.17	32.49	1.31

(b) KITTI Ignored

OOD Method	Layer	AUROC ↑ Base	Ctrst	Δ	FPR @ 95 TPR ↓ Base	Ctrst	Δ
Max Softmax	**-**	82.84	82.87	**0.03**	36.41	34.05	**-2.36**
Predictive Entropy	**-**	86.92	86.76	-0.16	29.69	27.96	**-1.73**
Aleatoric Entropy	**-**	86.76	86.77	**0.02**	30.09	28.24	**-1.86**
Mutual Information	**-**	70.62	72.47	**1.86**	85.78	82.66	**-3.12**
Mahalanobis	**backbone**	72.48	80.60	**8.11**	59.52	38.05	**-21.47**
OCSVM	**conv8x**	63.22	62.59	-0.63	72.03	70.91	**-1.11**
RealNVP	**backbone**	**<u>91.71</u>**	**<u>91.81</u>**	**0.10**	**<u>16.25</u>**	**<u>15.44</u>**	**-0.81**

(c) KITTI FP

OOD Method	Layer	AUROC ↑ Base	Ctrst	Δ	FPR @ 95 TPR ↓ Base	Ctrst	Δ
Max Softmax	**-**	83.18	**<u>83.34</u>**	**0.16**	52.17	**<u>45.80</u>**	**-6.36**
Predictive Entropy	**-**	80.64	81.23	**0.58**	53.69	47.50	**-6.19**
Aleatoric Entropy	**-**	80.01	80.90	**0.88**	53.97	47.62	**-6.36**
Mutual Information	**-**	67.53	66.41	-1.12	90.14	86.23	**-3.91**
Mahalanobis	**backbone**	**<u>88.30</u>**	82.21	-6.09	43.53	51.34	7.80
OCSVM	**conv8x**	64.74	62.20	-2.55	67.81	71.51	3.70
RealNVP	**backbone**	85.44	79.56	-5.88	**<u>33.03</u>**	46.01	12.98

(d) Waymo

Table 6.2: The OOD detection results for OOD object datasets.

Table 6.3 shows the additional results with all evaluation metrics and additional layers for all common OOD objects (i.e., OOD objects that are detected by both the base model and the contrastive model for a given run) for all OOD datasets. Results separated for each OOD dataset are in Tabs. 6.4 to 6.7. We highlight two additional observations.

For Mahalanobis distance and normalizing flows, backbone layer provides best overall performance. However, the optimal layer can vary for different type of OOD objects. For instance, for KITTI ignored and Waymo objects (Tables 6.5 and 6.7), using normalizing flows with conv4x features gives better results than with backbone features, whereas for KITTI FP objects (Table 6.6), backbone layer has 20% higher AUROC and AUPR compared to conv4x layer. For Mahalanobis distance, the variation is smaller. Conv2x layer provides better performance for a few metrics than backbone layer for Carla and KITTI FP objects (Tabs. 6.4 and 6.6).

For the contrastive model, we also utilize the contrastive layer features (labeled 'ctrst' in the table) to train Mahalanobis distance, OC-SVM, and normalizing flows. The contrastive layer does not provide a consistent overall improvement for any of the methods over all types of OOD objects (Table 6.3). Interestingly, for KITTI ignored objects (Table 6.5), the contrastive layer is the best performing layer for all three methods, outperforming other layers by a large margin. This suggests that the contrastive features are highly biased towards certain types of OOD objects. It would be interesting to investigate whether or not this issue also exists for image-based object detection and classification.

		AUROC			AUPR (ID)			AUPR (OOD)			FPR @ 95% TPR			Detection Err.		
OOD Method	**Layer**	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ
Max Softmax	**-**	89.95	**89.97**	**0.02**	91.46	**91.85**	**0.38**	**88.01**	**87.66**	-0.35	33.83	**31.86**	**-1.97**	19.33	**18.32**	**-1.00**
Predictive Entropy	**-**	83.49	84.80	**1.32**	87.50	88.94	**1.44**	76.48	77.56	**1.08**	39.60	34.54	**-5.05**	22.19	19.65	**-2.54**
Aleatoric Entropy	**-**	83.09	84.65	**1.56**	87.03	88.67	**1.64**	75.93	77.75	**1.82**	41.77	36.41	**-5.37**	23.29	20.57	**-2.71**
Mutual Information	**-**	60.48	60.70	**0.23**	54.10	54.79	**0.69**	64.48	63.87	-0.61	97.07	96.51	**-0.56**	49.98	49.83	**-0.15**
Mahalanobis	**logits**	66.55	67.76	**1.21**	59.28	59.04	-0.25	71.47	72.36	**0.88**	93.81	95.31	1.51	49.16	49.70	0.54
	backbone	**83.77**	**84.56**	**0.80**	**85.30**	**86.73**	**1.43**	**81.65**	**81.93**	**0.28**	**51.01**	**43.92**	**-7.10**	**27.89**	**24.34**	**-3.55**
	conv2x	79.37	79.48	**0.11**	78.46	77.89	-0.56	77.93	78.78	**0.85**	70.36	72.44	2.07	37.55	38.55	1.00
	conv4x	66.94	69.12	**2.18**	64.28	66.95	**2.67**	66.12	67.91	**1.79**	87.43	83.99	**-3.44**	46.11	44.41	**-1.70**
	conv8x	62.71	68.68	**5.97**	60.76	69.02	**8.26**	59.53	64.91	**5.38**	90.34	79.83	**-10.51**	47.55	42.32	**-5.23**
	ctrst	-	74.89	-	-	68.37	-	-	76.07	-	-	87.09	-	-	45.91	-
OCSVM	**logits**	52.53	46.45	-6.08	51.69	45.98	-5.71	57.43	54.11	-3.32	93.75	97.78	4.03	48.16	49.96	1.79
	conv2x	60.28	61.47	**1.19**	68.82	69.42	**0.60**	75.62	75.94	**0.32**	79.80	80.28	0.48	42.25	42.53	0.28
	conv4x	51.38	52.51	**1.13**	49.92	50.93	**1.02**	70.07	70.58	**0.51**	97.95	97.94	**-0.01**	49.97	49.95	**-0.02**
	conv8x	**65.44**	**62.91**	-2.53	**78.76**	**79.93**	**1.17**	**78.83**	**78.54**	-0.29	**66.20**	**70.31**	4.11	**35.31**	**37.17**	1.86
	ctrst	-	58.58	-	-	49.88	-	-	70.75	-	-	98.15	-	-	49.99	-
RealNVP	**logits**	63.84	61.46	-2.38	61.93	60.33	-1.59	63.24	60.20	-3.04	88.02	89.11	1.09	46.38	46.83	0.46
	backbone	**90.28**	88.30	-1.98	**92.67**	**90.89**	-1.78	**86.93**	**84.72**	-2.21	**26.72**	33.14	6.42	**15.72**	18.97	3.25
	conv2x	78.09	83.64	**5.56**	76.87	84.42	**7.56**	78.58	81.57	**2.99**	66.04	51.86	**-14.18**	35.34	28.33	**-7.01**
	conv4x	87.56	**88.41**	**0.85**	88.52	89.66	**1.15**	86.61	86.27	-0.33	48.26	41.69	**-6.57**	26.52	23.16	**-3.36**
	ctrst	-	87.32	-	-	89.38	-	-	83.19	-	-	**32.95**	-	-	**18.89**	-

Table 6.3: Results for all common OOD objects detected by both base and contrastive models. For Mahalanobis distance, OC-SVM, and RealNVP, we bold the best performing layer for each method. The best performing OOD method/layer for each metric is underlined. We also report the difference Δ between contrastive model and base model, with bold numbers indicating contrastive model has better performance. All results are averaged over three sets of experiments.

		AUROC			AUPR (ID)			AUPR (OOD)			FPR @ 95% TPR			Detection Err.		
OOD Method	**Layer**	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ
Max Softmax	**-**	**96.04**	**96.15**	**0.11**	**97.25**	**97.35**	**0.11**	**93.46**	**93.45**	-0.02	**10.39**	**10.23**	**-0.16**	**7.39**	**7.28**	**-0.11**
Predictive Entropy	**-**	85.34	87.12	**1.78**	89.42	91.06	**1.64**	77.72	79.29	**1.58**	35.55	30.10	**-5.45**	20.15	17.22	**-2.92**
Aleatoric Entropy	**-**	85.30	87.37	**2.06**	89.07	90.98	**1.91**	77.99	80.25	**2.26**	37.87	30.76	**-7.11**	21.34	17.69	**-3.65**
Mutual Information	**-**	45.04	43.18	-1.86	44.13	43.54	-0.59	53.30	50.95	-2.35	98.75	98.66	**-0.09**	49.99	49.99	-0.00
Mahalanobis	**logits**	83.51	83.40	-0.11	81.04	80.53	-0.51	84.63	83.31	-1.32	66.26	69.77	3.51	35.55	37.31	1.75
	backbone	**88.34**	**89.88**	**1.54**	**89.99**	**90.86**	**0.87**	**86.44**	88.35	**1.90**	**37.49**	35.78	**-1.71**	**21.16**	20.29	**-0.87**
	conv2x	85.82	88.80	**2.98**	84.44	85.90	**1.45**	85.10	**89.61**	**4.50**	53.82	58.01	4.19	29.35	31.44	2.09
	conv4x	72.57	80.11	**7.53**	73.83	81.42	**7.60**	68.86	77.10	**8.24**	72.85	60.14	**-12.71**	38.83	32.49	**-6.34**
	conv8x	67.08	71.82	**4.75**	71.95	75.88	**3.93**	58.73	63.58	**4.85**	70.17	63.25	**-6.91**	37.49	34.02	**-3.47**
	ctrst	-	89.81	-	-	90.19	-	-	85.08	-	-	**35.07**	-	-	**19.95**	-
OC-SVM	**logits**	54.26	48.41	-5.85	51.69	47.02	-4.67	58.74	56.45	-2.29	96.38	97.57	1.19	49.53	49.84	0.31
	conv2x	63.61	64.82	**1.21**	77.11	76.76	-0.34	78.39	78.60	**0.22**	68.39	**66.22**	**-2.17**	36.37	**35.43**	**-0.94**
	conv4x	61.50	63.16	**1.66**	74.71	76.27	**1.56**	77.47	78.16	**0.69**	77.32	71.20	**-6.12**	40.91	37.92	**-2.99**
	conv8x	**66.49**	63.12	-3.37	**82.29**	**80.74**	-1.55	**79.78**	78.74	-1.04	**63.49**	70.01	6.53	**33.61**	36.92	3.30
	ctrst	-	**72.38**	-	-	62.21	-	-	**79.69**	-	-	93.43	-	-	48.82	-
RealNVP	**logits**	70.81	70.26	-0.55	74.09	73.57	-0.52	66.90	65.64	-1.26	68.84	69.44	0.60	36.82	37.10	0.28
	backbone	93.63	92.50	-1.13	95.16	94.17	-0.99	90.82	89.22	-1.59	19.90	22.80	2.90	12.11	13.75	1.64
	conv2x	92.07	93.59	**1.51**	92.03	93.05	**1.03**	91.43	92.66	**1.22**	24.79	21.81	**-2.98**	14.79	13.32	**-1.47**
	conv4x	**94.65**	**95.79**	**1.13**	**95.92**	**97.01**	**1.09**	**92.25**	**92.78**	**0.53**	**17.85**	**12.24**	**-5.61**	**11.12**	**8.22**	**-2.90**
	conv8x	88.96	84.02	-4.94	92.13	87.70	-4.42	80.85	76.57	-4.28	26.83	39.55	12.71	15.81	22.15	6.33
	ctrst	-	92.44	-	-	94.78	-	-	87.81	-	-	16.66	-	-	10.52	-

Table 6.4: Results for Carla objects

		AUROC			AUPR (ID)			AUPR (OOD)			FPR @ 95% TPR			Detection Err.		
OOD Method	**Layer**	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ
Max Softmax	**-**	**95.38**	**96.06**	**0.48**	**95.80**	**96.35**	**0.55**	**95.32**	**95.77**	**0.45**	**21.23**	**20.80**	**-0.43**	**12.34**	**12.27**	**-0.08**
Predictive Entropy	**-**	78.65	80.41	**1.76**	84.83	86.59	**1.76**	69.75	71.44	**1.69**	41.66	36.10	**-5.55**	22.74	20.01	**-2.73**
Aleatoric Entropy	**-**	78.24	80.03	**1.79**	84.31	86.11	**1.80**	69.29	71.30	**2.01**	43.93	38.59	**-5.33**	24.00	21.26	**-2.73**
Mutual Information	**-**	52.09	55.94	**3.86**	47.53	50.16	**2.63**	60.33	63.24	**2.91**	99.13	98.58	**-0.55**	49.95	49.96	0.01
Mahalanobis	**logits**	74.12	78.08	**3.96**	68.50	75.29	**6.79**	73.93	75.74	**1.80**	88.34	78.49	**-9.85**	45.92	41.11	**-4.82**
	backbone	**86.25**	84.09	-2.16	**86.07**	85.46	-0.61	**85.98**	83.46	-2.52	54.35	50.86	**-3.48**	**28.80**	27.18	**-1.62**
	conv2x	63.77	59.44	-4.33	62.04	59.80	-2.25	65.70	58.77	-6.93	90.77	88.91	**-1.86**	46.81	46.23	**-0.58**
	conv4x	72.09	69.64	-2.44	70.67	67.15	-3.52	72.15	67.54	-4.62	78.82	84.29	5.47	41.25	43.94	2.68
	conv8x	85.02	89.98	**4.96**	84.94	**91.84**	**6.90**	80.52	85.98	**5.47**	**55.34**	32.50	**-22.84**	29.25	18.32	**-10.93**
	ctrst	-	**93.76**	-	-	91.28	-	-	**93.09**	-	-	**29.12**	-	-	**16.06**	-
OCSVM	**logits**	51.71	59.31	**7.60**	51.22	61.06	**9.84**	51.36	57.29	**5.93**	95.39	78.44	**-16.95**	49.11	41.15	**-7.97**
	conv2x	57.65	57.45	-0.20	61.82	61.07	-0.74	73.69	72.69	-1.01	90.26	88.26	**-2.00**	46.62	46.17	**-0.46**
	conv4x	63.44	64.52	**1.08**	80.73	80.64	-0.09	78.76	79.12	**0.37**	69.34	67.22	**-2.11**	36.61	35.52	**-1.08**
	conv8x	**66.95**	63.67	-3.28	**82.44**	81.83	-0.60	**80.00**	79.03	-0.98	**62.58**	69.03	6.45	**33.10**	36.33	3.23
	ctrst	-	**93.13**	-	-	**92.67**	-	-	**94.06**	-	-	**25.98**	-	-	**15.02**	-
RealNVP	**logits**	80.52	76.22	-4.29	79.61	77.48	-2.12	79.82	73.18	-6.64	67.14	67.79	0.64	35.56	35.83	0.28
	backbone	85.81	84.02	-1.79	90.13	89.09	-1.04	77.69	74.08	-3.61	31.17	32.49	1.31	17.49	18.18	0.69
	conv2x	77.66	76.60	-1.06	75.72	76.39	**0.67**	76.83	73.24	-3.59	70.63	71.81	1.18	37.33	37.73	0.40
	conv4x	**92.30**	91.79	-0.52	**92.84**	92.93	**0.09**	**90.32**	88.48	-1.84	**27.39**	34.80	7.42	**15.63**	19.24	3.61
	conv8x	90.10	87.13	-2.98	91.97	91.08	-0.89	85.62	80.26	-5.37	34.80	28.58	**-6.22**	19.29	16.23	**-3.06**
	ctrst	-	**94.79**	-	-	**95.23**	-	-	**92.62**	-	-	**23.55**	-	-	**13.62**	-

Table 6.5: Results for KITTI ignored objects

		AUROC			AUPR (ID)			AUPR (OOD)			FPR @ 95% TPR			Detection Err.		
OOD Method	**Layer**	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ
Max Softmax	**-**	82.84	82.87	**0.03**	88.06	88.36	**0.30**	72.43	72.06	-0.37	36.41	34.05	**-2.36**	20.64	19.40	**-1.23**
Predictive Entropy	**-**	86.92	86.76	-0.16	90.83	90.93	**0.11**	80.34	79.32	-1.02	29.69	27.96	**-1.73**	17.22	16.23	**-0.99**
Aleatoric Entropy	**-**	86.76	86.77	**0.02**	90.69	90.90	**0.20**	79.87	79.48	-0.39	30.09	28.24	**-1.86**	17.44	16.37	**-1.07**
Mutual Information	**-**	70.62	72.47	**1.86**	66.79	69.55	**2.76**	68.34	69.49	**1.15**	85.78	82.66	**-3.12**	45.37	43.82	**-1.55**
Mahalanobis	**logits**	47.34	49.66	**2.32**	46.27	46.62	**0.34**	52.99	55.72	**2.73**	97.17	97.66	0.49	49.97	49.98	0.01
	backbone	72.48	**80.60**	**8.11**	78.11	**86.52**	**8.41**	64.03	70.88	**6.85**	59.52	**38.05**	**-21.47**	32.24	**21.43**	**-10.80**
	conv2x	**76.18**	79.94	**3.76**	**81.64**	85.00	**3.36**	**67.34**	**72.18**	**4.84**	**53.12**	44.72	**-8.40**	**29.04**	24.81	**-4.22**
	conv4x	47.69	55.70	**8.01**	48.95	57.91	**8.96**	47.58	51.88	**4.30**	95.06	87.64	**-7.42**	49.65	46.26	**-3.38**
	conv8x	42.91	51.91	**9.01**	45.67	55.59	**9.92**	44.45	49.29	**4.85**	96.30	89.28	**-7.02**	49.66	47.03	**-2.63**
	ctrst		50.02	-	-	52.86	-	-	49.30	-	-	90.70	-	-	46.57	-
OC-SVM	**logits**	54.72	35.60	-19.13	61.71	52.18	-9.53	63.23	50.06	-13.16	**67.89**	**70.61**	2.71	**34.76**	**35.81**	1.06
	conv2x	54.35	57.45	**3.10**	61.80	65.80	**4.00**	70.37	72.34	**1.97**	82.22	78.88	**-3.34**	43.36	41.56	**-1.80**
	conv4x	23.16	24.00	**0.84**	33.50	33.82	**0.33**	46.59	47.40	**0.81**	99.29	99.26	**-0.03**	50.00	49.99	-0.00
	conv8x	**63.22**	**62.59**	-0.63	**73.98**	**79.64**	**5.66**	**77.07**	**78.42**	**1.35**	72.03	70.91	**-1.11**	38.34	37.48	**-0.87**
	ctrst	-	10.30	-	-	32.18	-	-	32.49	-	-	99.40	-	-	50.00	-
RealNVP	**logits**	45.35	43.23	-2.11	48.57	48.00	-0.57	46.32	44.59	-1.73	94.23	94.18	**-0.05**	48.17	48.88	0.71
	backbone	91.71	91.81	**0.10**	94.69	94.79	**0.09**	84.77	85.44	**0.67**	16.25	15.44	**-0.81**	9.58	8.99	**-0.59**
	conv2x	60.67	77.25	**16.58**	68.26	83.51	**15.25**	57.28	68.29	**11.01**	65.80	46.29	**-19.51**	34.96	25.48	**-9.48**
	conv4x	70.46	74.13	**3.66**	77.83	80.92	**3.09**	62.13	65.78	**3.65**	57.30	48.94	**-8.36**	31.00	26.36	**-4.64**
	conv8x	77.71	78.96	**1.25**	84.86	83.63	-1.23	66.30	71.48	**5.18**	40.53	40.06	**-0.47**	22.53	21.62	**-0.92**
	ctrst	-	80.91	-	-	88.48	-	-	66.52	-	-	27.81	-	-	14.51	-

Table 6.6: Results for KITTI FP objects

		AUROC			AUPR (ID)			AUPR (OOD)			FPR @ 95% TPR			Detection Err.		
OOD Method	**Layer**	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ	Base	Ctrst	Δ
Max Softmax	**-**	83.18	83.34	**0.16**	85.41	86.51	**1.10**	78.72	78.23	-0.49	52.17	45.80	**-6.36**	28.53	25.35	**-3.18**
Predictive Entropy	**-**	80.64	81.23	**0.58**	83.86	85.29	**1.43**	74.48	74.49	**0.01**	53.69	47.50	**-6.19**	29.29	26.19	**-3.10**
Aleatoric Entropy	**-**	80.01	80.90	**0.88**	83.45	85.05	**1.61**	73.14	73.96	**0.82**	53.97	47.62	**-6.36**	29.44	26.27	**-3.17**
Mutual Information	**-**	67.53	66.41	-1.12	63.31	63.70	**0.39**	67.16	65.15	-2.01	90.14	86.23	**-3.91**	47.50	45.56	**-1.93**
Mahalanobis	**logits**	52.45	52.37	-0.09	49.40	47.05	-2.36	58.55	62.28	**3.73**	96.29	98.21	1.93	49.73	49.98	0.25
	backbone	**88.30**	**82.21**	-6.09	**88.59**	**83.27**	-5.32	**85.96**	79.16	-6.80	**43.53**	**51.34**	7.80	**24.22**	**28.11**	3.89
	conv2x	85.96	81.80	-4.16	85.04	80.29	-4.75	83.49	**79.77**	-3.72	54.93	66.69	11.76	29.94	35.81	5.88
	conv4x	73.61	69.04	-4.57	70.75	64.27	-6.48	72.48	69.60	-2.88	81.35	89.39	8.04	43.10	47.11	4.00
	conv8x	69.90	71.70	**1.80**	71.88	73.19	**1.31**	63.76	65.78	**2.01**	75.88	73.17	**-2.71**	40.39	39.04	**-1.36**
	ctrst	-	70.37	-	-	64.18	-	-	69.45	-	-	92.54	-	-	48.44	-
OCSVM	**logits**	48.13	55.08	**6.96**	50.58	53.44	**2.86**	47.96	56.23	**8.28**	90.63	91.87	1.24	46.93	47.87	0.93
	conv2x	63.66	64.51	**0.85**	77.92	76.58	-1.34	**78.49**	**78.42**	-0.07	68.43	**66.47**	**-1.96**	36.33	**35.62**	**-0.72**
	conv4x	62.36	62.95	**0.59**	**78.15**	76.19	-1.96	78.21	78.08	-0.13	71.18	69.82	**-1.36**	37.65	37.27	**-0.39**
	conv8x	**64.74**	62.20	-2.55	78.02	**78.00**	-0.02	78.35	78.07	-0.28	**67.81**	71.51	3.70	**36.17**	37.94	1.77
	ctrst	-	**76.53**	-	-	72.76	-	-	77.67	-	-	87.44	-	-	45.66	-
RealNVP	**logits**	59.93	55.35	-4.59	62.04	53.92	-8.12	57.74	54.51	-3.23	84.17	93.10	8.93	44.51	48.59	4.08
	backbone	85.44	79.56	-5.88	89.46	84.60	-4.86	80.73	74.04	-6.69	33.03	46.01	12.98	18.92	25.43	6.51
	conv2x	84.62	82.29	-2.33	85.65	83.18	-2.47	82.17	79.84	-2.34	51.38	57.98	6.61	28.14	31.46	3.32
	conv4x	91.45	88.93	-2.53	92.77	89.76	-3.01	88.86	86.86	-2.00	31.47	44.45	12.98	18.18	24.68	6.50
	conv8x	78.69	74.58	-4.10	82.43	79.87	-2.55	71.27	66.54	-4.74	54.13	56.44	2.30	29.52	30.67	1.15
	ctrst	-	79.28	-	-	79.32	-	-	74.74	-	-	65.70	-	-	35.31	-

Table 6.7: Results for Waymo objects

6.2 Qualitative results

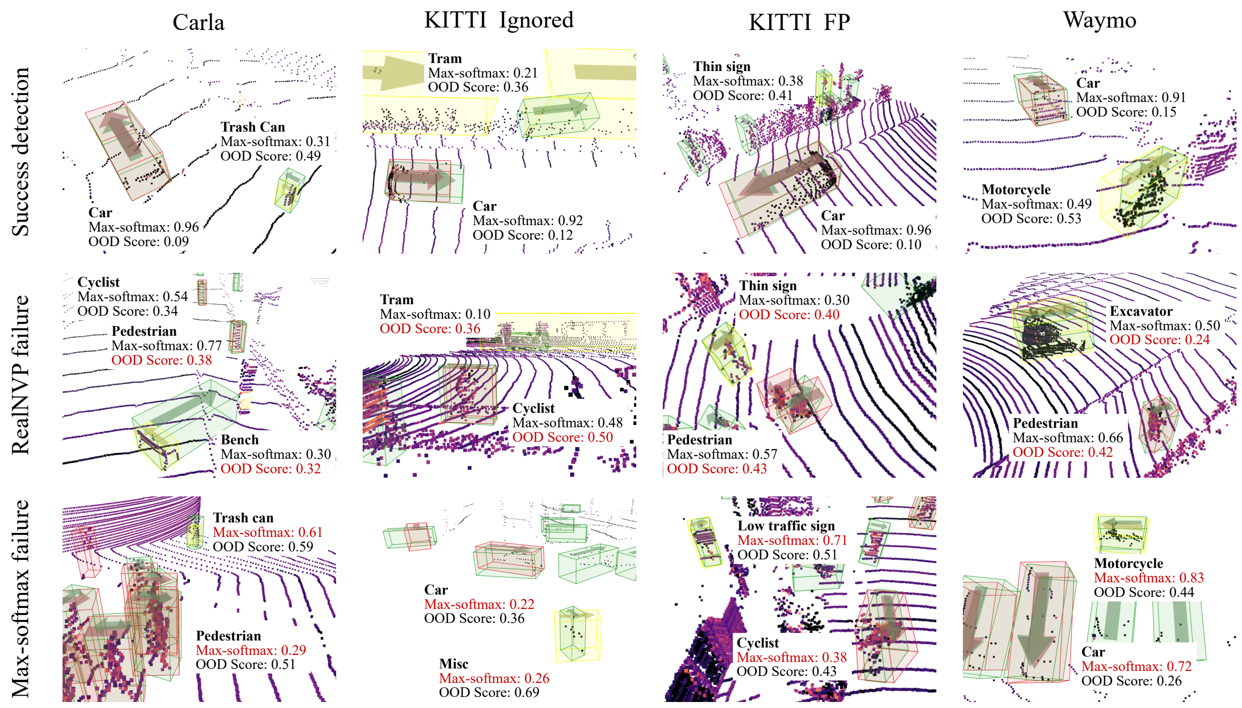
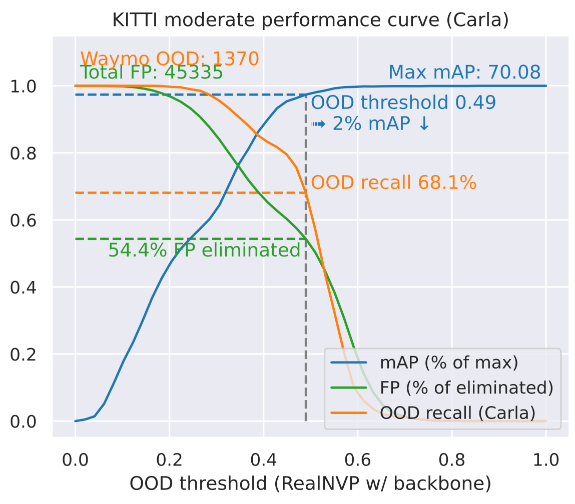


Figure 6.1: Qualitative results for OOD detection with max-softmax and OOD scores from normalizing flows (RealNVP) with backbone layer. Red boxes represents ground truth ID objects, yellow boxes are OOD objects, and green boxes are predictions from the base model. The text labels for RealNVP and max-softmax failures are red.

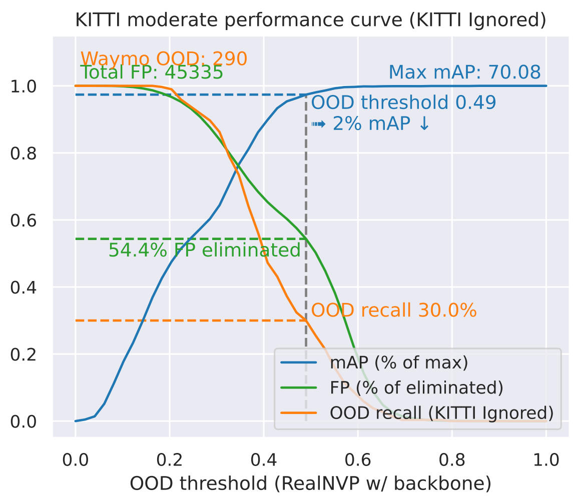
Figure 6.1 shows the qualitative results for OOD detection with max-softmax score and normalizing flows OOD detection methods. We use the base model without contrastive learning and normalizing flows with the backbone layer, which has the best overall performance for the base model (Table 6.3). For each type of OOD objects, we show three examples: successful detection, normalizing flows (RealNVP) failure, and max-softmax failure. Successful detections are examples where the OOD object is assigned low max-softmax score and high OOD score; RealNVP failures are examples where the OOD score from the normalizing flows for the OOD object is lower than ID objects; Max-softmax failures are examples where the max-softmax score for OOD object is higher than ID objects.

6.3 Performance impact of OOD detection

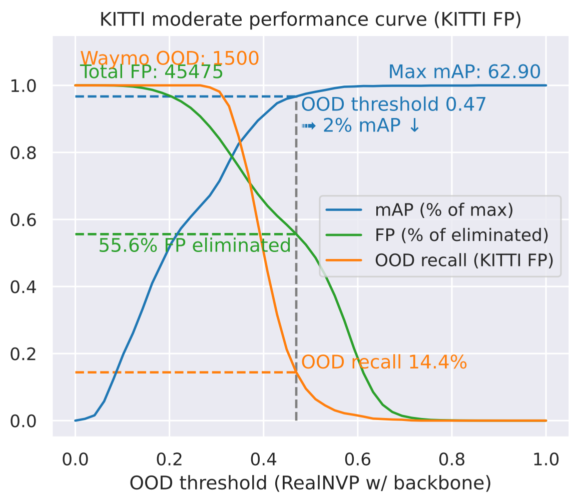
To demonstrate how OOD detection would impact the object detector's performance during deployment, we evaluate the mAP performance of the base model on KITTI moderate objects after removing predictions with OOD scores higher than a threshold. This is done across multiple OOD score thresholds. We choose normalizing flows (RealNVP) with the backbone layer as our OOD detector, which has the best overall performance for the base model. We use one set of experiments with the dataset augmented with Carla, KITTI FP, KITTI Ignored, and Waymo objects. Figure 6.2 shows the mAP, number of FP, and number of OOD across multiple OOD score thresholds.



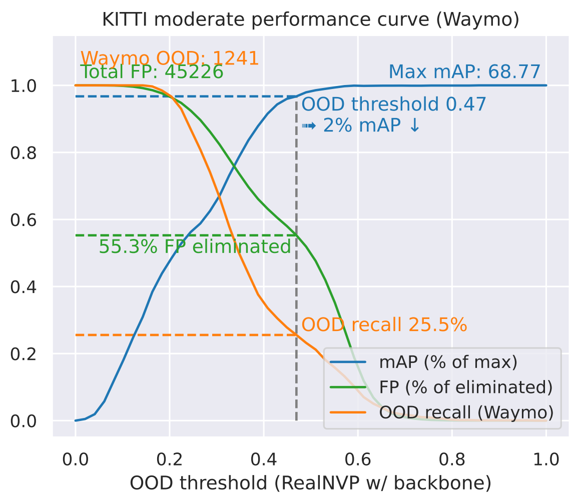
(a) Carla



(b) KITTI Ignored



(c) KITTI FP



(d) Waymo

Figure 6.2: Performance impact of OOD detection.

With a lower OOD score threshold, more predictions are marked as OOD objects, which naturally eliminates more FP and OOD objects but introduces more FN at the same time, leading to a decrease in mAP performance. From Figure 6.2, we see that with 2% mAP performance decrease, over 50% of the FP can be eliminated. However, the OOD recall varies significantly for different types of OOD objects. Overall, 35.3% of the objects labeled as OOD can be successfully identified, which shows that the OOD detection method is a practical addition to a 3D object detector in deployment.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Deep LiDAR-based 3D object detectors play an important role in robotic vision, and making them robust to OOD objects is key for assuring the safety of such systems. Although OOD detection has been defined and investigated extensively for classification, it has not been explored for LiDAR-based 3D object detection. In this thesis, we define different types of OOD samples for object detection and adapt the state-of-the-art OOD detection methods from image classification to LiDAR-based 3D object detection. In order to use OOD detection methods that rely on intermediate layers, we also propose a method for extracting feature embeddings for the detected objects. To enable the evaluation of the OOD detection methods, we propose a simple yet effective method to generate OOD objects for LiDAR-based 3D object detectors. We evaluate the OOD detection methods on the KITTI dataset augmented with a diverse set of real and synthetic OOD objects, revealing a nuanced landscape of how the current OOD detection methods perform in the context of LiDAR-based 3D object detection. The results demonstrate that each method is biased toward detecting certain types of OOD objects. Furthermore, the best practices proposed for image classification, such as selecting features of specific layers for OOD detection, may not transfer to object detection. We hope that our OOD dataset generation and evaluation results will stimulate further research into effective OOD detection for LiDAR-based 3D object detection.

7.2 Limitations and Future Work

Our OOD datasets have a few limitations. First of all, we do not collect type ③ OOD objects, which are unusual foreground objects. This is challenging, because we need to define to what extent a foreground object is considered unusual. Also, unusual cars are rare by definition and typically absent in the existing datasets for autonomous driving. Furthermore, Carla simulated objects use a fixed intensity (which matches the median from the target KITTI frame). It is possible that due to the fixed intensity, the softmax scores of Carla OOD objects are low, and therefore, the Max-softmax method works very well for these objects. Finally, compared to current datasets in autonomous driving, the KITTI dataset is small and has limitations in the labeling process. For example, there are a lot of foreground objects in the dataset, such as cars and pedestrians, that are unlabeled. Also, there are inconsistencies in labeling vans and cars.

One shortcoming of the adaptation of the OOD detection methods in this thesis is the inability to distinguish between type ⑤ and type ⑦ OOD objects. For example, the methods cannot differentiate between an FP from background objects in a BG training distribution and those in an unseen part of BG true distribution. In addition, we do not benchmark the methods on a type ③ OOD dataset due to the dataset limitation mentioned above.

Further, since type ④ OOD have no detections, we have no way to detect them using the presented methods. Detecting type ④ OOD objects, along with type ② ID objects, is covered in the field of FN detection (e.g., [57, 84]).

As future work, we suggest extending our work by evaluating the OOD detection and object generation methods over other automotive datasets. For instance, we can insert the OOD objects into the Waymo dataset, which is larger and with higher quality annotations than the KITTI dataset. Also, we should define and collect unusual foreground objects—type ③ OOD—to benchmark the methods. Furthermore, we should make sure that the intensity of the OOD objects matches the intensity of inserted datasets. For simulated objects, this may require learning intensity distributions conditioned on surface properties from real data and applying them in simulation. We can also explore methods to distinguish between type ⑤ and type ⑦ OOD, by training these OOD detection methods on the background classes. We are also interested in understanding better the characteristics of OOD methods, why a contrastive loss has a different effect on the different methods and developing a combined OOD detection method that is not biased toward specific types of OOD objects.

References

[1] Vahdat Abdelzad, Krzysztof Czarnecki, Rick Salay, Taylor Denounden, Sachin Vernekar, and Buu Phan. Detecting out-of-distribution inputs in deep neural networks using an early-layer output. *arXiv preprint arXiv:1910.10307*, 2019.

[2] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

[3] Matt Angus, Krzysztof Czarnecki, and Rick Salay. Efficacy of pixel-level OOD detection for semantic segmentation. *arXiv preprint arXiv:1911.02897*, 2019.

[4] Petra Bevandić, Ivan Krešo, Marin Oršić, and Siniša Šegvić. Discriminative out-of-distribution detection for semantic segmentation. *arXiv preprint arXiv:1808.07703*, 2018.

[5] Christopher M Bishop. Novelty detection and neural network validation. *IEE Proceedings-Vision, Image and Signal processing*, 141(4):217–222, 1994.

[6] Hermann Blum, Paul-Edouard Sarlin, Juan Nieto, Roland Siegwart, and Cesar Cadena. Fishyscapes: A benchmark for safe semantic segmentation in autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019.

[7] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar 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.

[8] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teuliere, and Thierry 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.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pages 1597–1607. PMLR, 2020.

[10] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016.

[11] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian 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] Hyunsun Choi, Eric Jang, and Alexander A Alemi. WAIC, but why? generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

[13] Taylor Denouden. An application of out-of-distribution detection for two-stage object detection networks. Master's thesis, University of Waterloo, 2020.

[14] Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.

[15] Akshay Dhamija, Manuel Gunther, Jonathan Ventura, and Terrance Boult. The overlooked elephant of object detection: Open set. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1021–1030, 2020.

[16] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *The International Conference on Learning Representations*, 2016.

[17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Conference on Robot Learning*, pages 1–16. PMLR, 2017.

[18] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar 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. IEEE, 2017.

[19] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.

[20] Di Feng, Lars Rosenbaum, and Klaus 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.

[21] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

[22] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[23] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *The International Conference on Learning Representations*, 2015.

[24] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

[25] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *The International Conference on Learning Representations*, 2016.

[26] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *The International Conference on Learning Representations*, 2019.

[27] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *Conference on Neural Information Processing Systems*, 2019.

[28] Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. Generalized ODIN: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10951–10960, 2020.

[29] Ramneet Kaur, Susmit Jha, Anirban Roy, Oleg Sokolsky, and Insup Lee. Are all outliers alike? on understanding the diversity of outliers for detecting oods. *arXiv preprint arXiv:2103.12628*, 2021.

[30] Alex Kendall and Yarin Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5574–5584. Curran Associates, Inc., 2017.

[31] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

[32] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *The International Conference on Learning Representations*, 2014.

[33] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[34] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Conference on Neural Information Processing Systems 2017*, 2016.

[35] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar 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.

[36] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[37] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *The International Conference on Learning Representations*, 2018.

[38] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in Neural Information Processing Systems*, 31, 2018.

[39] Bo 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. IEEE, 2017.

[40] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. *Robotics: Science and Systems XII*, 2016.

[41] Yingzhen Li and Yarin Gal. Dropout inference in bayesian neural networks with alpha-divergences. In *International Conference on Machine Learning*, pages 2052–2061. PMLR, 2017.

[42] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *The International Conference on Learning Representations*, 2018.

[43] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[44] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE conference on Conference on Computer Vision and Pattern Recognition*, pages 806–814, 2015.

[45] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[46] Nicolas Marchal, Charlotte Moraldo, Hermann Blum, Roland Siegwart, Cesar Cadena, and Abel Gawel. Learning densities in feature space for reliable segmentation of indoor scenes. *IEEE Robotics and Automation Letters*, 5(2):1032–1038, 2020.

[47] Dimity Miller. *Epistemic uncertainty estimation for object detection in open-set conditions*. PhD thesis, Queensland University of Technology, 2021.

[48] Dimity Miller, Lachlan Nicholson, Feras Dayoub, and Niko Sünderhauf. Dropout sampling for robust object detection in open-set conditions. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3243–3249. IEEE, 2018.

[49] Dimity Miller, Niko Sünderhauf, Michael Milford, and Feras Dayoub. Class anchor clustering: a distance-based loss for training open set classifiers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV 2021)*, 2021.

[50] Dimity Miller, Niko Sünderhauf, Michael Milford, and Feras Dayoub. Uncertainty for identifying open-set errors in visual object detection. *arXiv preprint arXiv:2104.01328*, 2021.

[51] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana 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.

[52] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[53] Julia Nitsch, Masha Itkina, Ransalu Senanayake, Juan Nieto, Max Schmidt, Roland Siegwart, Mykel J Kochenderfer, and Cesar Cadena. Out-of-distribution detection for automotive perception. *Intelligent Transportation Systems Conference (ITSC)*, 2021.

[54] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with PixelCNN decoders. *Conference on Neural Information Processing Systems*, 2016.

[55] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas 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.

[56] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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.

[57] Quazi Marufur Rahman, Niko Sünderhauf, and Feras Dayoub. Did you miss the sign? a false negative alarm system for traffic sign detectors. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3748–3753. IEEE, 2019.

[58] Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark A DePristo, Joshua V Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. *Conference on Neural Information Processing Systems*, 2019.

[59] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[60] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 2021.

[61] Mohammadreza Salehi, Hossein Mirzaei, Dan Hendrycks, Yixuan Li, Mohammad Hossein Rohban, and Mohammad Sabokrou. A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges. *arXiv preprint arXiv:2110.14051*, 2021.

[62] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[63] Vikash Sehwag, Mung Chiang, and Prateek Mittal. SSD: A unified framework for self-supervised outlier detection. *arXiv preprint arXiv:2103.12051*, 2021.

[64] Shaoshuai Shi. OpenPCDet. https://github.com/open-mmlab/OpenPCDet, 2018.

[65] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li Pv-rcnn. Point-voxel feature set abstraction for 3d object detection. 2020 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10526–10535, 2020.

[66] Shaoshuai Shi, Xiaogang Wang, and Hongsheng 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.

[67] Weijing Shi and Raj 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.

[68] Martin Simon, Stefan Milz, Karl Amende, and Horst-Michael Gross. Complex-YOLO: Real-time 3d object detection on point clouds. *arXiv preprint arXiv:1803.06199*, 2018.

[69] Kihyuk Sohn, Chun-Liang Li, Jinsung Yoon, Minho Jin, and Tomas Pfister. Learning and evaluating representations for deep one-class classification. *The International Conference on Learning Representations*, 2021.

[70] Amos J Storkey. When training and test sets are different: characterising learning transfer. In *In Dataset Shift in Machine Learning*, pages 3–28. MIT Press, 2009.

[71] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

[72] Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. CSI: Novelty detection via contrastive learning on distributionally shifted instances. *Conference on Neural Information Processing Systems*, 2020.

[73] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[74] Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 550–564, 2018.

[75] Lorenz Wellhausen, René Ranftl, and Marco Hutter. Safe robot navigation via multi-modal anomaly detection. *IEEE Robotics and Automation Letters*, 5(2):1326–1333, 2020.

[76] David Williams, Matthew Gadd, Daniele De Martini, and Paul Newman. Fool me once: Robust selective segmentation via out-of-distribution detection with contrastive learning. *IEEE International Conference on Robotics and Automation*, 2021.

[77] Jim Winkens, Rudy Bunel, Abhijit Guha Roy, Robert Stanforth, Vivek Natarajan, Joseph R Ledsam, Patricia MacWilliams, Pushmeet Kohli, Alan Karthikesalingam, Simon Kohl, et al. Contrastive training for improved out-of-distribution detection. *arXiv preprint arXiv:2007.05566*, 2020.

[78] Kelvin Wong, Shenlong Wang, Mengye Ren, Ming Liang, and Raquel Urtasun. Identifying unknown instances for autonomous driving. In *Conference on Robot Learning*, pages 384–393. PMLR, 2020.

[79] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio 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.

[80] Zhisheng Xiao, Qing Yan, and Yali Amit. Do we really need to learn representations from in-domain data for outlier detection? *arXiv preprint arXiv:2105.09270*, 2021.

[81] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[82] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[83] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

[84] Qinghua Yang, Hui Chen, Zhe Chen, and Junzhe Su. Introspective false negative prediction for black-box object detectors in autonomous driving. *Sensors*, 21(8):2819, 2021.

[85] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1951–1960, 2019.

[86] Tianwei Yin, Xingyi Zhou, and Philipp 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.

[87] Hongjie Zhang, Ang Li, Jie Guo, and Yanwen Guo. Hybrid models for open set recognition. In *European Conference on Computer Vision*, pages 102–117. Springer, 2020.

[88] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu. SE-SSD: Self-ensembling single-stage object detector from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14494–14503, 2021.

[89] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay 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.

[90] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[91] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019.

[92] Ev Zisselman and Aviv Tamar. Deep residual flow for out of distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13994–14003, 2020.