CHALMERS
UNIVERSITY OF TECHNOLOGY

Master Thesis

Deep learning based 2D and 3D joint object detection with camera and LIDAR for ADAS and AD

Di Xue
Peizheng Yang

Supervisor:
Hossein Nemati <hossein.nemati@huawei.com>,
Samuel Scheidegger <samuel.scheidegger@huawei.com>

Examiner: Alexandre Graell i Amat <alexandre.graell@chalmers.se>
Advisor: Mohammad Hossein Moghaddam <mh.moghaddam@chalmers.se>

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg
Sweden 2020

Master's thesis in Electrical engineering (EENX30)

Deep learning based 2D and 3D joint object detection with camera and LIDAR for ADAS and AD

Di Xue, Peizheng Yang

Supervisor: Hossein Nemati, Samuel Scheidegger
Examiner: Alexandre Graell i Amat
Advisor: Mohammad Hossein Moghaddam

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2020

Typeset in LaTeX
Gothenburg, Sweden 2020

Abstract

In this thesis, we studied the topic of deep learning based object detection in the context of environment perception for ADAS and AD. The deep learning model we implemented is a joint 2D-3D detector, and the sensors used are camera and LIDAR. The 2D detector outputs 2D bounding boxes with category, location and size on the camera image. The 3D detector takes the information from 2D detection as prior knowledge, localize the object in 3D space and predict corresponding 3D bounding box. In our work, we studied both 2D and 3D detectors. In the 2D detector part, we compared the detection ability of the state-of-art two-stage 2D detector faster R-CNN using two different backbones (VGG16 and ResNet101 FPN) on the NuScenes dataset and found that VGG16 backbone outperforms ResNet101 FPN backbone. We did experiments on how occlusion affects the 2D detectors in the urban traffic scenes, and found that object with occlusion larger than 20% will be difficult to detect.

The 3D detector we implemented is a modified version of Frustum ConvNet which detects the object from the point cloud inside the frustum that is confined by the 2D bounding box. We found that the 3D detector is inherently good at detecting objects with small size.

We also studied the influence of unfavourable weather condition in our 2D and 3D detection task. Surprisingly, the LIDAR performance does not decrease in rain scene or night scene. And the decreased performance of 3D detector in the night scene mainly comes from the decreased performance of 2D detector.

Acknowledgment

First of all, we would like to express our sincere thanks to Huawei Gothenburg Research Center for providing us with this challenging and excellent project. Thanks to Hossein Nemati as our industrial supervisor for his guidance to help us control the overall research . Thanks to Samuel Scheidegger as our mentor in the company for his professional and patient supervision during the whole research process. Also sincerely thanks to Peter Svenningsson as our co-worker for his valuable comments in each meeting and generous technical help. We also want to express our sincere gratitude to the teachers at campus who give us firm support. Thanks to Alexandre Graell i Amat as our academic examiner. Thanks to Andreas Buchberger for the great effort in helping us finding academic examiner. Thanks to Madeleine Persson and Mohammad Hossein Moghaddam for their great help in both academic and administrative tasks.

Secondly, we would like to thank Chalmers University of Technology for its academic resources and environment during our two-year master's program. The courses and projects offered by Chalmers University of Technology have helped us accumulate valuable experience in both theory and practice.

Finally, we want to thank every close friend of us for their kind support during the whole process, thanks for their encouragement and accompany. Thank them for sharing our joy and frustration all the way. None mentioned, none forgotten.

Di Xue, Peizheng Yang, Gothenburg, August 2020

Contents

List of Figures

List of Tables

1
Introduction

This thesis is carried out for the object detection in Advanced driver-assistance systems (ADAS) and Autonomous Driving (AD) research team under Huawei Gothenburg Research Center.

1.1 Background

We are in an era of automation, from industrial production to daily life, more and more automation products are playing an increasingly important role in different positions. Broadly speaking, autonomous robots include all machines that perform behaviors or tasks with a high degree of autonomy. Products such as indoor automatic vacuum cleaners and manufacturing automatic processing robots have been developed to a mature stage and now are widely used in modern households and industries. However, in transportation, one of the most important aspects of daily life, the autonomous level still has great potential to be improved. Passenger cars that are closely related to life are still not available in AD context, and the last-mile delivery nowadays still relies on manual driving. McKinsey [5] has estimated that we would soon inhabit a world where more than 80% of the parcel delivery will have autonomous vehicles (AV) to complete the last mile of delivery. As a part of the autonomous family, AD is an active research area that attracts researchers from different backgrounds to research actively. Considering the room for improvement and massive on-going research, the future of AD is infinite.

AD [6] refers to intelligent vehicles that can completely replace the driver to make decisions in a narrow sense. Broadly speaking, six levels of AD has been defined by The Society of Automotive Engineers (SAE). The second level is also called advanced driver-assistance systems (ADAS) and is the highest level of mature automotive products on the market. The researches on ADAS and AD are often inseparable. They share many technologies and the common purpose is to make driving safer. Most traffic accidents are caused by human factors. To reduce or eliminate unsafe human factors, ADAS and AD need to assist or substitute drivers to make decisions.

A complete autonomous system for AD is illustrated as Figure 1.1. The sensors sample data from environment. The decision making part, consisting of three modules, perception, planning, and control, takes the sensor data as input and decides which actions will be taken. The command of actions are sent to the actuators of the vehicle to execute.

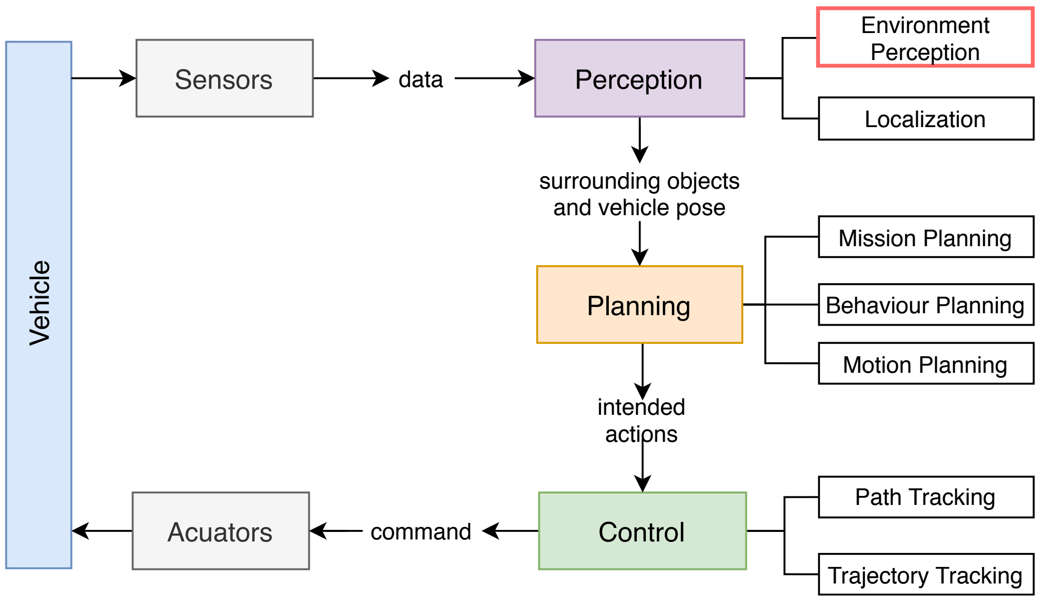


Figure 1.1: Autonomous System, illustration inspired by [1–3]

The sensors that currently accomplish this task are mainly camera, Light detection and ranging sensor (LIDAR), and Radio detection and ranging sensor (radar) [7, 8]. These three sensors have different characteristics. Camera is of low cost and has high resolution but is greatly affected by illumination. LIDAR and radar can directly obtain distance information but the maximum distance is limited compared with a camera. LIDAR is expensive and more weather-sensitive while radar is the most stable one to weather factors but also returns the most sparse point cloud and in most cases no height information. To be more specific, there are 3D radar which can return height information but they are relatively rare and quite expensive. The three sensors are complementary to each other. Some AD dataset [9] install all three types of sensors when collecting data and call this combination the full sensor suite of a real self-driving car.

After the sensor completes the original data collection, a detection algorithm is needed to process the data from sensor and output the detection results. Perception is the first step for decision making. It means the ability of an autonomous system to collect information and extract relevant knowledge from the environment. The relevant knowledge includes two aspects: the semantic meaning of the surrounding environment and the position of the vehicle with respect to the environment. The perceived knowledge provides the evidence for planning, which refers to the process of making purposeful decisions that safely brings the vehicle from a start location to a goal location while avoiding obstacles and optimizing over designed heuristics. The control module converts the intended actions of a vehicle to the real actions. The main purpose of this module is to provide the necessary hardware-level commands so that the actuators can generate the desired motions.

Our work only focuses on the environment perception. To be more specific, our task is object detection which is part of the environment perception. The nature of object detection is to build a learning model that learns the features from the input data. The traditional methods use hand-crafted methods to generate features and use shallow trainable structures [10–14]. In recent years, thanks to the development of computer hardware, object detection using deep learning [15] have become increasingly popular. More and more powerful object detection algorithms have been developed to extract deeper and richer semantics from the original data and achieve the detection effect that traditional methods can not achieve. This thesis also chooses deep learning as the approach to conduct object detection.

Deep learning methods are hungry of large-scale data. The collection and annotation in the autonomous driving usually takes great labour. The open-source database with a considerable amount of data makes the deep learning research much easier. Currently popular published databases include Lyft Level 5 Dataset [16], KITTI dataset [17], ApolloScape Dataset [18] from Baidu, Waymo open dataset [19] and NuScenes dataset [9]. The database that we use in this thesis is NuScenes dataset since it is the newest and the first dataset that carries the full autonomous vehicle sensor suite, which is to say, it contains data from all three main ADAS and AD sensors, i.e., camera, LIDAR and radar.

In this thesis we investigated different networks that utilize one or multiple sensors among camera, LIDAR and radar, which results in a literature review of related work in both 2D and 3D object detection. We did exploration on NuScenes dataset and implemented an object detection architecture utilizing its data and examine the performance.

1.2 Limitations

Since NuScenes dataset is a new dataset, most of the state-of-art object detection algorithms have not been verified on it. Our investigation for appropriate algorithm does not include implementation and verification of every investigated algorithm on NuScenes dataset.

Moreover, although the hyperparameters in our implemented network are manually tuned, a greedy search that goes through every hyperparameter possibility and pushes the performance of the network to the maximum is not involved due to time and resource limitation.

1.3 Thesis Outline

We structure this thesis report to show all the work in the thesis project. Investigation on different object detection methods which results in a literature review and a detailed description for selected algorithm are provided in Chapter 2. Chapter 3 introduces the implementation and experiment details. Chapter 4 shows the quantitative and qualitative results of the algorithms. Chapter 5 gives discussion.

Chapter 6 offers final conclusion and insights for future works.

2
Theory & Related work

Object detection is widely used in ADAS and AD and many other research fields such as biomedical imaging. The definition of object detection is to detect the category and location of each instance giving one environment sample data. Location is commonly represented by a rectangular bounding box that is drawn around the instance and contains every part of it. The input data for object detection can be image, commonly RGB image, from camera, point cloud from LIDAR, point cloud from radar or different data from different sensors fusing together. Object detection can be divided into 2D object detection [14, 20–27] and 3D object detection [4, 28–42]. Notably, both 2D object detection and 3D object detection may leverage data fusion for a better performance. 2D object detection means the bounding box is 2D and drawn on a 2D plane, commonly on an image from camera, while 3D object detection means the bounding box is 3D and drawn in 3D space. Some work [43] uses polylines instead of bounding boxes to better represent the shape.

2.1 2D object detection

Architectures with region proposal stage are called two-stage detector, which means a region is firstly detected as foreground or background and then foreground regions are fed into the later structure to do category-wise classification. On the other hand, one-stage detector does not have region proposal part in its architecture.

2.1.1 Two-stage architecture

Rich feature hierarchies convolutional neural network (R-CNN) [20], the first updated version of R-CNN (fast R-CNN) [21], the second updated version of R-CNN (faster R-CNN) [22] and the third updated version of R-CNN which can do both object detection and semantic segmentation (mask R-CNN) [24] are a series of two-stage objection architecture. We will introduce faster R-CNN in detail.

In the first ten years of 21st century, feature extraction is mainly based on handcrafted techniques such as scale-invariant feature transform (SIFT) [10] and histogram of oriented gradients (HOG) [11], R-CNN [20] is a breakthrough outperforming these by using deep convolutional neural networks (CNN) as feature extractor which shows a stronger feature representation capability.

The architecture of R-CNN composites 3 parts. The first part generates category-independent region proposals which tells the architecture where to look at for objects. The second part is a deep CNN that extracts a fixed-length feature vector from each region proposal. The third part is a set of class-specific linear support vector machines (SVM) which are used for classification. The region proposals are generated using selective search, which intuitively is using the color, texture, composition and hierarchy to segment the image and select regions of interest (ROI). All candidate regions are then warped into the same size in order to be able to feed together into the following CNN. To solve localization error, a simple linear regression model between selective region proposals and corresponding ground truth bounding boxes are trained, which is proved to be very useful in improving performance.

Based on R-CNN, fast R-CNN [21] applies the deep CNN feature extractor on the whole image instead of on each region proposal to reduce repetitive convolutional computation. It also proposes ROI pooling to resize candidate region proposals and puts location regression and category classification together by using two parallel fully connected layers (FC).

Faster R-CNN [22] proposes region proposal network (RPN) and the concept of anchor to replace selective search and generate region proposals. This helps to dramatically reduce the computation cost in region proposal generation and further improves the speed and performance.

Feature pyramid network (FPN) [23] generates a series of feature maps representing different resolutions. FPN expands the scope of feature extractor by using multiple feature maps in parallel. Big size objects are detected from low resolution feature maps while small size objects are detected from high-resolution feature maps. [23] and [30] prove that FPN shows better detection performance in small objects. Mask R-CNN [24] absorbs FPN and proposes ROI align to better localize ROIs in order to improve bounding box localization accuracy. It also add a semantic segmentation branch based on faster R-CNN architecture to do object detection and semantic segmentation simultaneously.

2.1.2 Faster R-CNN

The diagram of faster R-CNN is illustrated in Figure 2.1. Given an input image, the backbone consisting many convolutional layers generates one feature map or multiple feature maps of different scales. Two representative backbones are very deep convolutional networks for large-scale image recognition (VGG) [44] backbone and deep residual neural networks for image recognition (ResNet) [45] backbone. Multiple feature maps can be achieved by adding the feature pyramid structure to the backbone network. RPN takes in the feature map(s) and outputs several proposals. A proposal means a rough bounding box including any object of interest in the detection task and does not tell the specific category of the object. The ROI feature extractor selects the feature inside the proposals from the feature map. The mainstream ROI feature extractors are ROI pooling and ROI align. The selected ROI feature is finally sent to the detection head to classify the category and further refine the localization of each proposal.

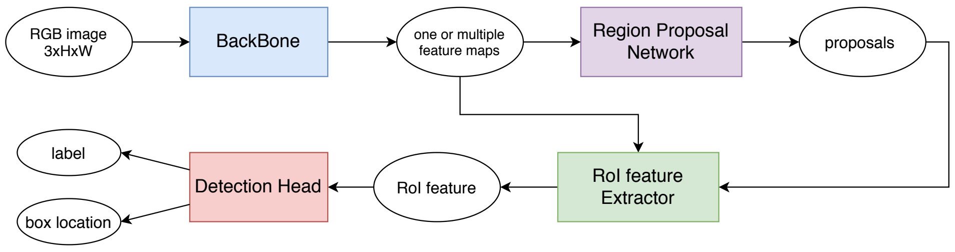


Figure 2.1: faster R-CNN structure

2.1.2.1 VGG backbone

VGG [44] includes a series of same structure but different depth image recognition networks which take an image of size $224 \times 224 \times 3$ as input and predict its category. VGG family includes VGG11, VGG13, VGG16 and VGG19, with different number of weight layers. We take VGG16 as illustration example, as shown in Figure 2.2. A set of cascaded convolutional layers and max-pool layers extracts the features from the input image and finally generates one feature map. All the convolutional kernels are of size 3×3 and the output feature depth, i.e., number of convolutional kernels, are gradually increased from 64 to 512 as the network forwards deeper to get richer semantic information. The VGG backbone that we use in faster R-CNN architecture consists of only the layers before the last max-pooling layer.

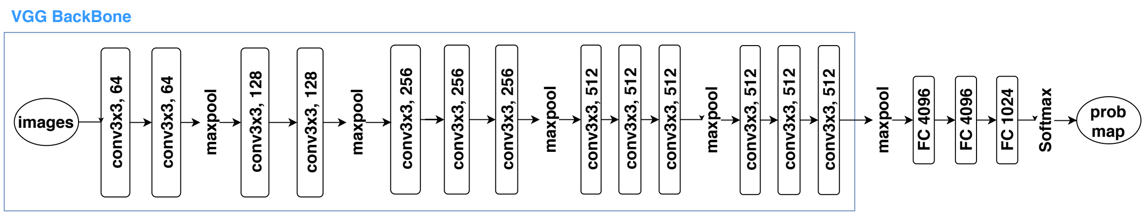


Figure 2.2: VGG16 structure. The layers in the blue box make up the VGG16 backbone.

2.1.2.2 ResNet backbone

ResNet [45] is another family of image recognition networks. The smallest unit block of ResNet which is called residual block, as shown in Figure 2.3, consists of three cascaded convolutional layers and one skip layer. The number of cascaded convolutional layers are usually 2 or 3. The skip layer takes the same input as the cascaded convolutional layers, and it can be either a convolutional kernel of size 1×1 or simply pass the identity map of the input, depending on whether the depth dimension of the input and output of the cascaded convolutional layers match or not. The outputs of both paths are element-wise added as the final output of this basic block.

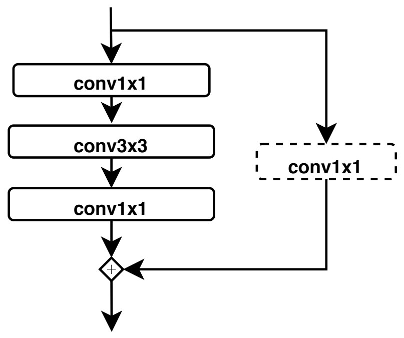


Figure 2.3: Residual block of 3 cascaded convolutional layers

The complete structure of a ResNet is illustrated in Figure 2.4. The initial 7×7 convolutional kernel and max-pooling layer is common for all members in ResNet family. The number of the following residue blocks varies among different ResNets. The network outputs the probability of each category. The ResNet backbone for faster R-CNN only includes the parts within the blue box.

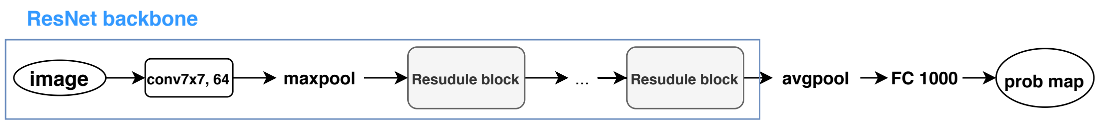


Figure 2.4: ResNet structure. The layers within the blue box make up the ResNet backbone.

2.1.2.3 Feature pyramid network

Feature pyramid network (FPN) [23] is a structure which can be applied on the backbone to generate several feature maps of different resolutions. As illustrated in Figure 2.5, the deepest feature from the backbone, which is denoted as feature map k-1, is the feature map with the coarsest resolution. The coarsest feature map is up-sampled and added with the second deepest feature from the backbone to form the finer feature map which is denoted as feature map k. The other feature maps are formed in the same way.

Using multiple feature maps has the advantage that objects with different sizes can be well represented by these feature maps with different resolutions. As the experiments shown in [23], FPN based faster R-CNN can achieve higher mean average precision (mAP) than the single feature map faster R-CNN.

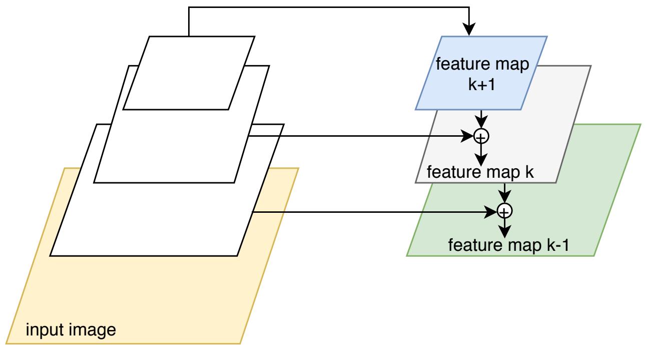


Figure 2.5: Feature pyramid network structure

2.1.2.4 Region proposal network

After the feature map is extracted by the backbone feature extractor, it is sent to region proposal network (RPN). Faster R-CNN proposed the concept of anchor in RPN. Anchor means a set of multiple rectangle boxes with fixed sizes that traverse every pixel on the feature map. From another perspective, anchor borrows the concept of exhaustive method and will hopefully enclose the target object feature representation inside one of the considerable amount of anchors. At the beginning, a 3×3, stride 1 convolution kernel traverses through each pixel on the feature map to perform a semantic space transformation. Then the forward path is divided into two branches. Let k denote the number of anchors. As in Figure 2.6, the left classification branch outputs a tensor of shape $2 \times k$ for each pixel, which represents the probability of background or foreground, i.e., probability of enclosing an object of interest or not, for each anchor. The other localization regression branch outputs a tensor of shape $4 \times k$ for each pixel to predict the variable needed for characterizing the adjusted location of each anchor at each pixel. The adjusted location of each anchor are not directly predicted, instead, the relative offset value from anchor center (x_a, y_a) and anchor width and height (w_a, h_a) are predicted. The localization output is

$$t_x = \frac{x - x_a}{w_a},\ t_y = \frac{y - y_a}{h_a} \tag{2.1a}$$

$$t_w = \log(w/w_a),\ t_h = \log(h/h_a), \tag{2.1b}$$

where (t_x, t_y, t_w, t_h) represent the offset value from anchor center and anchor size respectively.

In RPN the loss function

$$L_{\text{RPN}} = L_{\text{cls}} + L_{\text{reg}} \tag{2.2}$$

is composed by loss from both classification branch and localization branch.

Under training mode, anchor target creator is responsible for selecting a subset of anchors from the original considerable amount of anchors in order to compute the classification loss and the regression loss. Anchor target creator takes all original anchors and ground-truth bounding boxes as input. It selects some anchors that have an intersection over union (IoU) higher than a threshold with any ground-truth bounding box as positive samples and assigns foreground label 1 to them. It also selects some anchors that have an IoU lower than a threshold with all ground-truth bounding boxes as negative samples and assigns background label 0 to them. Cross-entropy loss is used for classification and smooth $l1$ loss is used for regression. Notably, when computing classification loss, both positive samples and negative samples are taken into account while when computing regression loss, only positive samples contribute and negative samples are ignored.

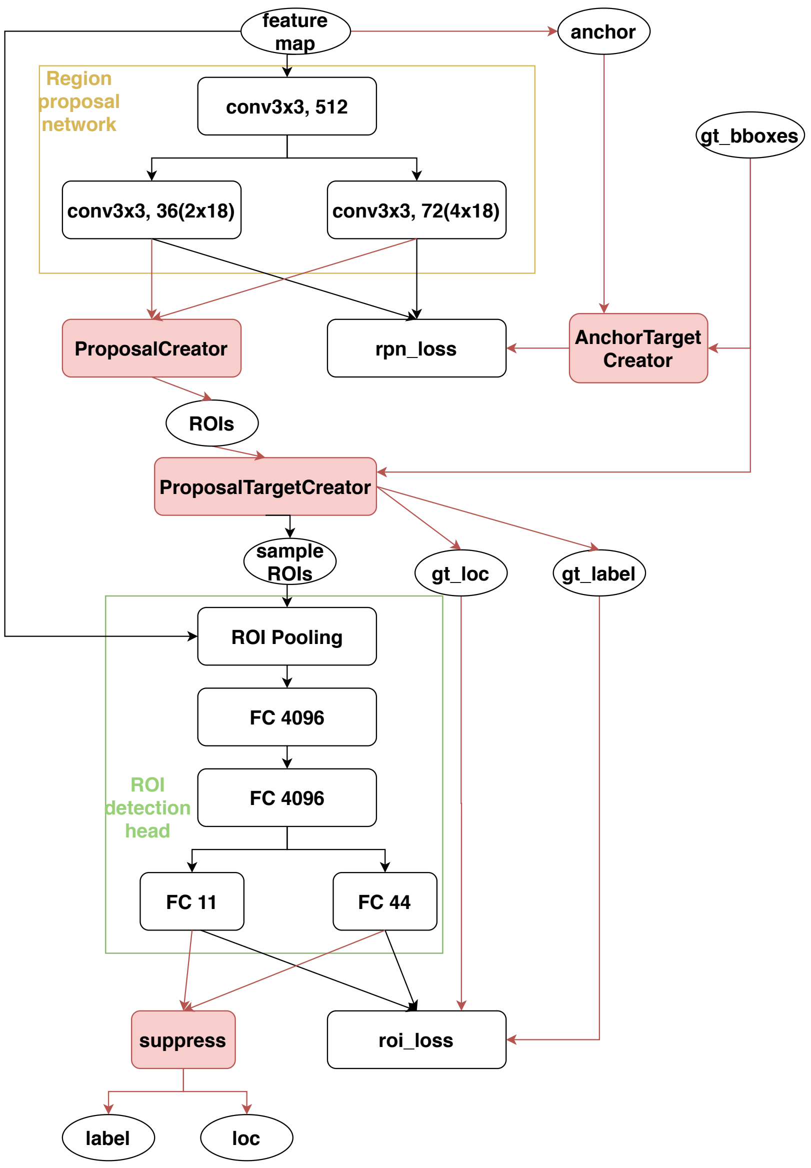


Figure 2.6: RPN, ROI and detection head for the **single** feature map case. The data flow in the red arrow does not have backward propagation. Illustration style refers to [46].

In the case of multi-scale feature maps, they are processed in parallel using the same RPN, as Figure 2.7. In other words, the RPNs chained after each feature map share weights during training. Each feature map will generate its own ROIs.

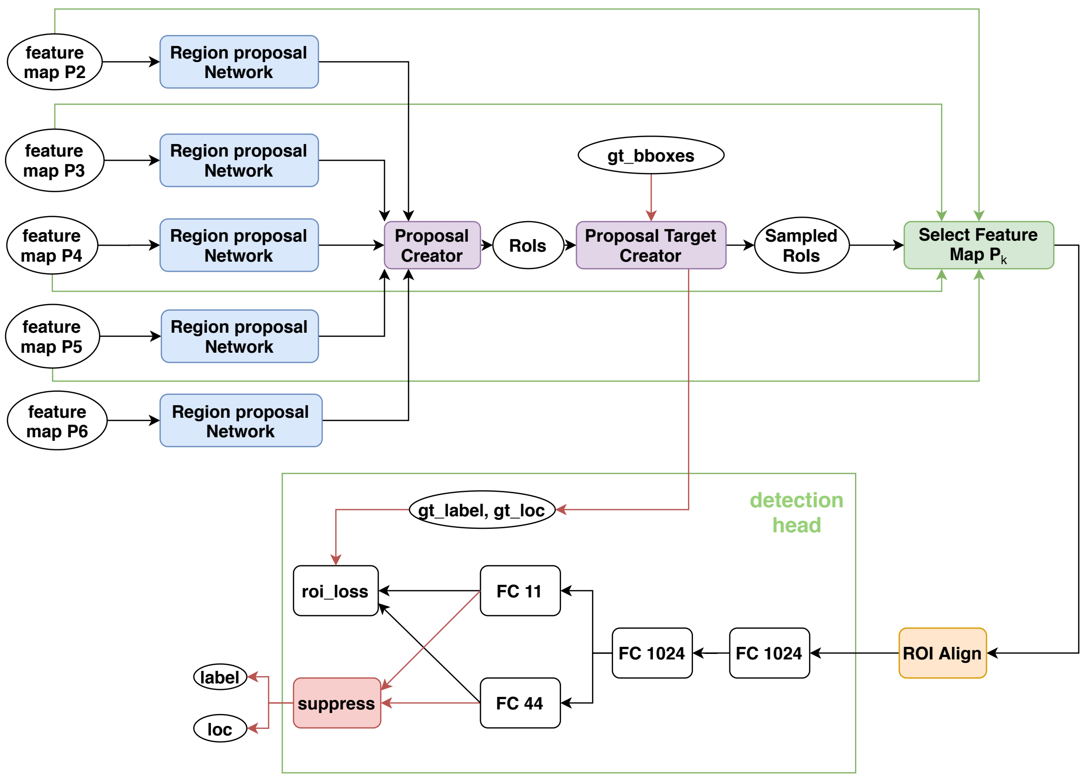


Figure 2.7: RPN, ROI and detection head for the **multiple** feature maps case.

2.1.2.5 Proposal target creator and feature map selection

Under training mode, to help the learning process and also accelerate the training speed, proposal target creator is used to select a subset of the proposals generated from RPN and pass them to the next step. Taken all proposals, i.e., ROIs, and ground-truth bounding boxes as input, proposal target creator selects some proposals that have an IoU higher than $T_{positive}$ with any ground-truth bounding box as positive samples and selects some proposals that have an IoU that equals to or lower than $T_{negative}$ as negative samples. The ratio of positive sample number and negative sample number is set as $1:3$ since there is always more background than foreground in a common image. Under inference mode, there will be no proposal target creator and all proposals generated from RPN are passed to ROI detection head.

For multiple feature maps, proposal target creator selects the ROIs from all the feature maps and outputs the proposals. Given a proposal, assuming the ROI on the image has width equal to w and height equal to h,

$$k = 4 + \log_2(\sqrt{wh}/224) \tag{2.3}$$

gives the relationship between the size of an ROI and the corresponding resolution feature map that it should use, k represents the appropriate level of feature map that should be chosen.

2.1.2.6 ROI feature extractor: ROI pooling

The selected proposals from proposal target creator under training mode or all proposals coming from RPN under inference mode need to be sent to ROI pooling, because each proposal corresponds to an area of feature map with a different size. ROI pooling is used to pool all these areas of different sizes to the same scale in order to enable weights sharing during training. Firstly, the ROIs whose size are originally represented in the image scale, i.e., 563×1000, are scaled to the feature map size, i.e., 36×63, and the float coordinates which are caused by scaling are transformed to integer, as illustrated in Figure 2.8. Next, the feature map inside the integer-size ROI is cropped from the feature map. Given pooled feature is initialized with zeros as the given pooled size $(w_{\text{pooled}}, h_{\text{pooled}})$. Then each patch of the pooled feature retrieves the max value within its neighbourhood from the feature inside the integer-size ROI. The detailed ROI pooling operation is described in Algorithm 1.

In our implementation, the pooled feature of each proposal is set as 7×7. Finally, as shown in Figure 2.6, the pooled ROI features are sent to the detection head, which is composed by two FC and two branches for category classification and predicted bounding boxes localization regression. The loss function of classification and regression is same as in RPN. Cross-entropy loss is used for classification and smooth $l1$ loss is used for regression.In the inference mode, non-maximum suppression (NMS) will be applied for all the predicted bounding boxes to get the final 2D predictions.

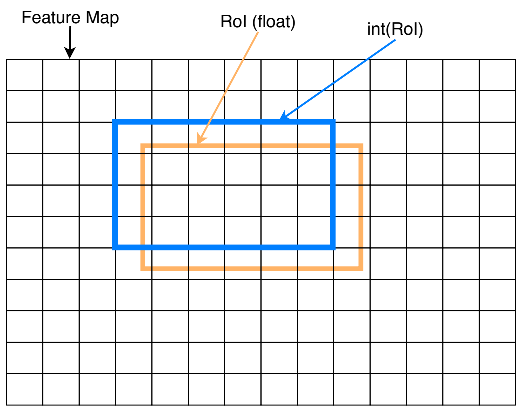


Figure 2.8: ROI pooling first step. The ROIs are scaled to feature map and the features in the integer blue box are used to generate pooled feature.

Algorithm 1 Different size ROI sharing training weights: ROI pooling

```
1: function ROI POOLING($ROI$, $featmap$, $w_{\text{pooled}}$, $h_{\text{pooled}}$)
2:     $x_{min}, y_{min}, x_{max}, y_{max} \leftarrow \text{int}(ROI)$
3:     $R_{\text{cropped}} \leftarrow featmap[y_{min} : y_{max},\ x_{min} : x_{max}]$
4:     $w_{\text{cropped}} \leftarrow x_{max} - x_{min} + 1$
5:     $h_{\text{cropped}} \leftarrow y_{max} - y_{min} + 1$
6:     $s_w,\ s_h \leftarrow w_{\text{cropped}}/w_{\text{pooled}},\ h_{\text{cropped}}/h_{\text{pooled}}$
7:     $\mathbf{P} = \mathbf{0}[h_{\text{cropped}} \times w_{\text{cropped}}]$
8:     for $i = 0 \rightarrow h_{\text{pooled}} - 1$ do
9:         for $j = 0 \rightarrow w_{\text{pooled}} - 1$ do
10:            $x_{\text{min}} \leftarrow js_w$
11:            $x_{\text{max}} \leftarrow (j+1)s_w$
12:            $y_{\text{min}} \leftarrow is_w$
13:            $y_{\text{max}} \leftarrow (i+1)s_w$
14:            if $x_{\text{min}} == x_{\text{max}}$ or $y_{\text{min}} == y_{\text{max}}$ then
15:                continue
16:            else
17:                $\mathbf{P}[i,j] = \max(R_{\text{cropped}}[y_{min} : y_{max},\ x_{min} : x_{max}])$
18:            end if
19:        end for
20:    end for
21: return P
22: end function
```

2.1.2.7 ROI feature extractor: ROI align

Updated and improved from ROI pooling, ROI align [24] neither rounds the float ROI sizes after scaling to feature map to integer, nor round the result of after-scale ROI divided by given pooled size to integer. By eliminating these two approximation step, the error in localization decreases. Given the example of desired pooled feature of shape 2×2 in Figure 2.9, in each bin, there are four sampling points. ROI align computes the value of each sampling point by bilinear interpolation from the surrounding four feature pixels. And the max value of each sampling point is selected as the pooled feature.

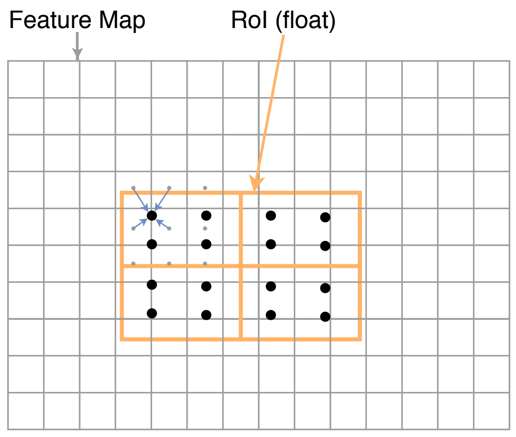


Figure 2.9: ROI align. The gray grid represents each pixel in the feature map. The gray dot represents the center location of each pixel. The figure illustrates the pooled feature of shape 2×2.

2.1.2.8 Detection head

The detection head in faster R-CNN consists of two FC to process the ROI extracted feature. In the VGG16 backbone version, both FC have an output dimension of 4096. The initial weights of the two FC are taken from the same VGG16 pre-trained model which provides the initial weights for the feature extractor. In the ResNet101 FPN backbone version, both FC have an output dimension of 1024, in which the parameters are initialized randomly. At the last step, two branches are diverged to do the category classification and box localization regression, similar as in RPN. The predicted localization variables are the offset of the ROI instead of the direct corner coordinates.

2.1.3 One-stage architecture

YOLO [25] is a representative work of one-stage 2D object detection architecture. The object detection problem is framed as a regression problem here so that one single CNN can be used to predict bounding boxes and class probabilities directly from image.

The pipeline of YOLO is that, an input image is divided into grids, each grid cell predicts bounding boxes and corresponding confidence scores. To be more specific, if no object exists in the cell, the confidence scores should be 0. Otherwise the confidence score is supposed to equal IoU between the predicted bounding box and the ground truth bounding box. At the same time, each grid cell also predicts classification score for each category. [25] also tried to combine YOLO with fast R-CNN and the results show a boost in accuracy because YOLO can distinguish background better. But because fast R-CNN runs much slower than YOLO, this combination makes the detection runs much slower than original YOLO.

The successive version of YOLO, YOLOv2 [26], modifies the backbone, imports the concept of anchor box and uses the offset from anchor boxes to train localization. The number of bounding boxes that each grid cell predicts also increases to enrich prediction results. YOLOv3 [27] supports multi-scale prediction and uses its original darknet-53 as backbone. The softmax classifier has been replaced by multiple independent binary logistic classifier. The performance for small objects has been improved compared to YOLOv2.

Thanks to its one-stage design, YOLO can run really fast at 45 fps in real time. Compared with fast R-CNN, YOLO has lower detection accuracy. YOLO shows that it makes more localization error which is its main source of error, especially in small objects, but it is very good at eliminating false positives on background.

Essentially, one-stage 2D detector provides a significantly faster inference speed at the expense of partial accuracy loss. Currently, only the speed of one-stage detector can be regarded as real time in the context of commercial use. The reason why two-stage 2D detector has an obvious performance accuracy boost is RPN. RPN needs more computation budget but at the same time reduces localization error.

2.2 Point cloud feature learning

The feature extractor in image object detection is usually a CNN. The convolution operation utilizes the fact that the pixels are arranged regularly in the 2D plane. The input feature of image object detection only contains the RGB feature and not include each pixel's position in the 2D plane. However, in point cloud, the naive convolution operation on the input points fail due to the points are not regularly distributed as the 2D image pixel. Thus, a special architecture is needed for the point cloud feature learning.

2.2.1 PointNet feature extraction

PointNet [47] is an architecture specialized for point cloud learning. The input of PointNet is a single point characterized by its spatial coordinates. For LIDAR point, reflective intensity can also be one of the input dimension. The input feature is processed by several FC layers which output a high dimensional tensor with rich feature of each point. The FC layer is shared by each point of the input point cloud. This set of FC layers is called multilayer perceptron (MLP). An element wise max operation will output the max elements of each dimension along all the points. In many point cloud object detection architectures, PointNet is widely used as the backbone network that extracts the features of a pre-confined local region.

2.2.2 Graph-based feature extraction

A graph is composed of a set of vertices and a set of unordered pairs of the edges. Dynamic Graph CNN (DGCNN) [41] is one of the graph-based feature extraction method. This method proposes edge function and edge convolution. The edge function explicitly concatenates the global feature structure and local structure of each point. MLP is also used to get high dimensional feature. The global feature is extracted the same way as PointNet [47], the local feature is captured from K-nearest neighbourhood of each point in the feature space. Note that the the K-nearest neighbourhood is updated from the feature space after each layer of MLP.

2.2.3 Voxel-based feature extraction

Another point cloud learning method is based on voxels. The representative work is VoxelNet [32]. The points are grouped by a set of regularly-aligned 3D cubic voxels, and the feature from each voxel is extracted by MLP layer similar as PointNet. After this, borrowing the idea from 2D convolution, 3D convolution can further extract higher level feature from each voxel with its voxel-level feature.

Generally speaking, PointNet has the highest simplicity and also a strong representation ability. Point-wise MLP as the key component of PointNet is also used in graph-based and voxel-based feature extraction methods. Voxel-based 3D convolutional neural network usually has high computational cost since it adds one dimension to 2D convolution. Graph-based feature extraction utilizes the concept of PointNet in its architecture and becomes more complex. It is inherently powerful to handle irregular data and attracts increasing attention as a newer network in recent years.

2.3 3D object detector using only LIDAR point cloud

Similar to the 2D object detection architecture, 3D object detection architecture can also be divided into one-stage and two-stage. Moreover, like 2D object detector needs to extract features from image pixels, 3D object detector using LIDAR point cloud also needs feature extraction method in Chapter 2.2 to do 3D representation.

LaserNet [34] is a one-stage detector which only takes 3D LIDAR data as input and the format is raw range view data, which is not in Cartesian coordinates system but instead in range-angle system. Especially, for the first time, LaserNet proposes to predict a soft localization which models the distribution of bounding boxes' corners instead of just predicting single fixed values to localize the bounding boxes. LaserNet [34] proves that predicting a probability distribution of 3D boxes is very vital to the performance. The pipeline is that, firstly the range view is arranged into a pseudo image map which has 5 channels that are range, height, intensity, flag, and azimuth angle. Then a fully convolutional neural network is applied to predict classification and soft box estimation for each point. Finally mean shift clustering and adaptive NMS are applied to get the final result. Notably, LaserNet does not consider offset to ground and vehicle height, the 3D bounding boxes only have 4 corners and are essentially bird's eye view (BEV) 2D bounding boxes. The class imbalance problem is handled by using focal loss.

PointPillars [36] is another one-stage detector which only uses LIDAR point cloud data. The pipeline of PointPillars is that, firstly divide the 3D point cloud space into grids in BEV, i.e., infinite length in vertical so that no need to tune cell size in vertical. Secondly count the points inside each pillar and add decorations to make each point have 9 dimensions. Thirdly extract features from pillars and generate a pseudo image which is the feature map of all pillars, then a 2D CNN is applied to proceed the pseudo image and finally the processing result is fed into a SSD [48] detection head.

MEGVII [37] is a voxel-based two-stage detector for 3D object detection. The input is only 3D LIDAR point cloud. The main contribution of MEGVII is its way to deal with imbalance issue on datasets. Firstly a class-balanced sampling strategy handles extreme imbalance issue by duplicating samples of one category according to its fraction among all samples, then randomly selecting and pasting ground-truth boxes of different classes using different magnitude on the ground plane. Secondly a multi-group head network is designed to make categories of similar shapes or sizes could benefit from each other, and categories of different shapes or sizes stop interfere with each other. After the multi-grouping, the class is reduced from 10 to 6.

Sparse-to-Dense (STD) [38] 3D Object Detector is a two-stage detector that utilizes both PointNet and voxelized method, and it generates region proposals in a multi-step fashion. PointNet++ [49] serves as the backbone and generates preliminary class score and features of each point. The spherical anchors are used to generate the initial proposals. The proposal predicted by each anchor is based on the points in the spherical receptive field. NMS is then applied to remove the redundant ones among candidate anchors. With selected anchors and features output from the backbone, 3D points within anchors is fed into a PointNet [47] to predict classification scores, regression offsets and orientations. By adding the offsets and orientation to the predefined anchors, a more refined proposal can be generated. Another NMS is used based on classification scores to eliminate redundant proposals one more time to get the filtered proposals. At the second stage, the feature of each proposal is from voxelization and voxel feature encoding (VFE) layer. The features are fed to two branches, IoU estimation branch and box prediction branch. Box prediction branch generates classification and regression. IoU estimation branch plays a supportive role and predicts the IoU value of each predicted box. The predicted IoU value is multiplied by the classification score to do NMS to get the final bounding boxes.

PointRGCN [39] is a two-stage 3D detector. The first stage RPN takes off-the-shelf methods to generate high-recall proposals. The focus of [39] is its second stage named refine proposals. In the second stage, residual graph convolutional neural network (R-GCN) is used to extract local features from each proposals. Further, to refine all proposals better, all proposals are put in a graph representation where each node is a proposal, EdgeConv [41] is then applied to get global feature. Both local and global feature contributes to the box prediction.

VoteNet [40] is a one-stage detector takes the point cloud of a scene which contains multiple objects and backgrounds and outputs the 3D bounding box of each object. A backbone , which is PointNet++ in [40], has input $N \times 3$, subsamples and learns the feature of the selected points, and outputs $M \times (3 + C)$. Voting module is MLP which learns the xyz offset of each point to its object center. A well-learned voting module will increase the cohesion of the points which belongs to the same object. Then the points after vote layer are clustered and each cluster is meant to represent one object. Each cluster is finally classified and predict bounding box.

Part-aware and Part-aggregation Network [42] is another two-stage architecture. The entire pipeline is mostly based on voxelization. In the part-aware stage, the point cloud is voxelized and fed to a sparse 3D convolutional encoder and decoder network to generate features of each point. In the part-aggregation stage, each 3D proposal is divided into a regular shape of voxels (eg. $14 \times 14 \times 14$). The points with features that lies inside each voxel is voxel-wised maxpooled and voxel-wised average pooled. The output from the pooling is further sent to a sparse convolution layer to get the 3D box prediction.

2.4 3D object detector using both LIDAR and camera

Besides models that only utilize point cloud, there are researches focusing on fusing camera data and LIDAR data together to take advantage of mature on-board digital cameras and off-the-shelf image-based computer vision techniques.

2.4.1 Multimodal/multiView feature fusion

One type of the detectors fuses the features from different sensor modalities together. For example, in MV3D [28], a 3D proposal is generated by using only LIDAR BEV feature. The 3D proposal is projected to LIDAR BEV, LIDAR front view and front camera view respectively, and the features inside the projected proposals of one 3D proposal are concatenated together as a fused feature of the 3D proposal. In AVOD [30], the same fusion method is used to extract the feature from front camera and LIDAR BEV of a given pre-defined 3D anchor box. Another example is LaserNet++ [35], which projects each LIDAR point to the front camera image. A CNN extracts the image features. The camera features that lie at the location where LIDAR points are projected to are concatenated with the LIDAR features.

2.4.2 Camera detector as prior information

Frustum PointNet (F-PointNet) [31] is a novel 3D object detection network that directly operates on point cloud. It directly takes off-the-shelf 2D object detectors and uses results from them, i.e., 2D bounding boxes and corresponding classifications, as prior region proposals. The 2D bounding boxes in image are then projected into 3D space as a frustum and LIDAR point cloud inside each projected frustum is recorded. Notably, each frustum is set to contain exactly one object of interest whose detection points are selected by instance segmentation. 3D bounding box estimation is conducted using discrete and fixed anchor candidates after object of interest is segmented. A novel regularization loss called corner loss [31] is used to jointly optimize bounding boxes' location and orientation.

Inspired by [31, 32], another novel 3D object detection architecture named Frustum ConvNet (F-ConvNet) [4] is proposed. Same as F-PointNet, an off-the-shelf 2D detector is used to provide frustums of LIDAR points. However, the performance of F-PointNet is limited because it does not use end-to-end learning to estimate oriented boxes and the final estimation relies on too few foreground points which themselves are possibly segmented wrongly. F-ConvNet resolved these two issues.

F-ConvNet has three main contributions. It proposes a novel grouping mechanism named sliding frustum, as well as the concept of multi-resolution frustum and a refinement method after the 3D bounding boxes have already been generated to deal with the inaccurate 2D proposal issue. Compared with MV3D, AVOD and F-PointNet, F-ConvNet has outbreak progress in hard detection task. The key design of F-ConvNet is to aggregate point-wise features inside each frustum at the early stage as a frustum-wise feature vector. The imbalance issue of foreground and background is dealt with focal loss. F-ConvNet will be introduced in detail in Chapter 2.4.3.

2.4.3 Frustum ConvNet

Frustum ConvNet (F-ConvNet) [4] is an end-to-end structure that utilizes the 2D detection results as prior information for the 3D detector. Given a 2D detection bounding box on an image and the detected category, F-ConvNet takes the LIDAR points that can be projected inside the 2D bounding box, and the one-hot vector that represents the detected 2D category as input. The output is the predicted center, size and orientation of an 3D bounding box. The assumption is that only one object of interest is in each frustum, as in Figure 2.10.

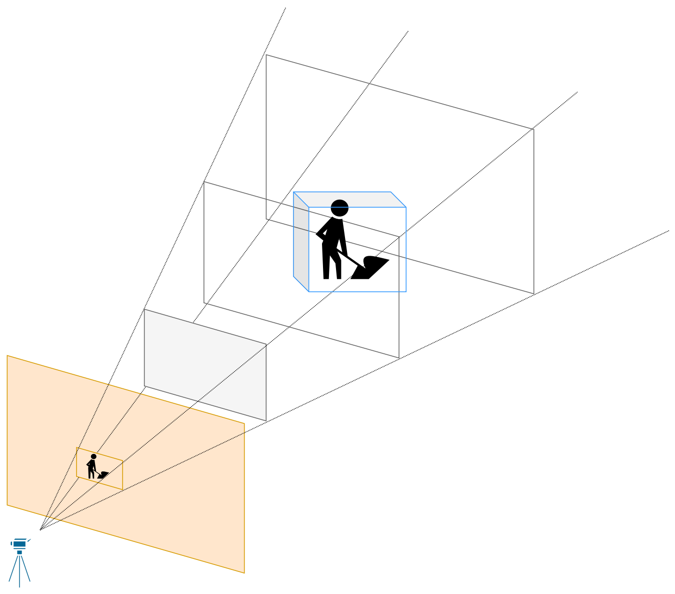


Figure 2.10: Illustration of Frustum ConvNet

Frustum ConvNet is composed by three parts, PointNet backbone as feature extractor, fully convolutional layer for feature aggregation and detection head that is responsible for classification and regression. The entire 2D-3D joint detection pipeline and each part of Frustum ConvNet will be introduced in this chapter.

2.4.3.1 Entire detection pipeline

The entire pipeline utilize three sensors on a vehicle, camera, LIDAR and Inertial Measurement Unit (IMU). Given a single timestamp, camera and LIDAR provide not only the captured data, i.e., 2D image and 3D point cloud, but also their pose information at this timestamp and the intrinsic matrix for the camera. The sensor modalities of the data used is shown in Table 2.1. The data flow is shown in Figure 2.11. The point cloud from the LIDAR is in LIDAR coordinate system, and it is transformed to the camera coordinate system first.

Table 2.1: The data used and its source sensor

sensor	data provided
camera	image, camera pose, camera intrinsic
LIDAR	point cloud, LIDAR pose
IMU	ego pose

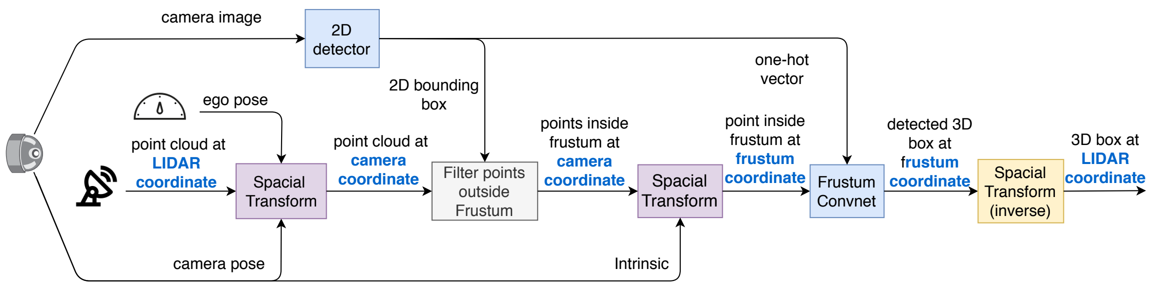

camera image
2D detector
2D bounding box
one-hot vector
ego pose
point cloud at LIDAR coordinate
Spacial Transform
point cloud at camera coordinate
Filter points outside Frustum
points inside frustum at camera coordinate
Spacial Transform
point inside frustum at frustum coordinate
Frustum Convnet
detected 3D box at frustum coordinate
Spacial Transform (inverse)
3D box at LIDAR coordinate
camera pose
Intrinsic

Figure 2.11: Entire 2D-3D joint detection pipeline work flow

Next, given the 2D bounding box and the camera intrinsic, only the 3D points that can be projected inside the 2D bounding box will be kept. Let $\mathbf{x} = (x, y, z)^T$ denote the point at the camera coordinate, $\mathbf{u} = (x_{2\text{D}}, y_{2\text{D}}, 1)^T$ denote the homogeneous point on the camera plane, and $\mathbb{K}$ denote the camera intrinsic matrix. The camera projection equation is

$$\lambda \begin{pmatrix} x_{2\text{D}} \\ y_{2\text{D}} \\ 1 \end{pmatrix} = \mathbb{K}\mathbf{x}. \tag{2.4}$$

The projected $(x_{2\text{D}}, y_{2\text{D}})$ outside the image canvas will be removed.

Next, the remaining points at camera coordinate will be rotated to frustum coordinate, as in Figure 2.12. The z axis of frustum coordinate system points to the center of the 3D object at BEV, as Figure 2.12 indicates. The point cloud at frustum coordinate system will be sent to the 3D detector to generate predicted 3D boxes which are also at the frustum coordinate system. Finally, each predicted 3D bounding box will be transformed back to the LIDAR coordinate system.

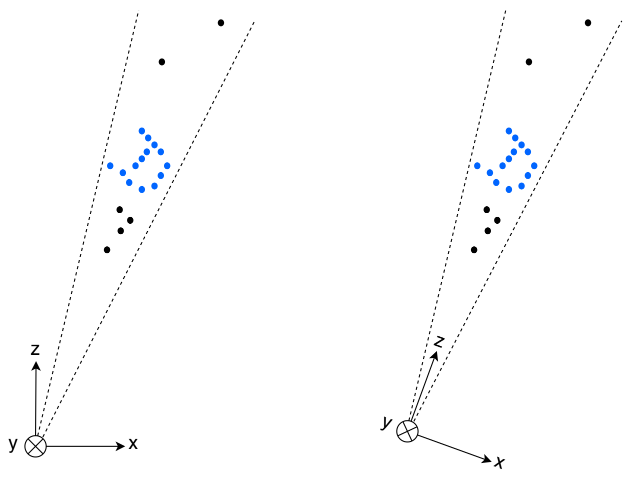


Figure 2.12: The rotation of the point cloud from camera coordinate to frustum coordinate. Left: camera coordinate, Right: frustum coordinate. Blue dots represent the point cloud on the object of interest.

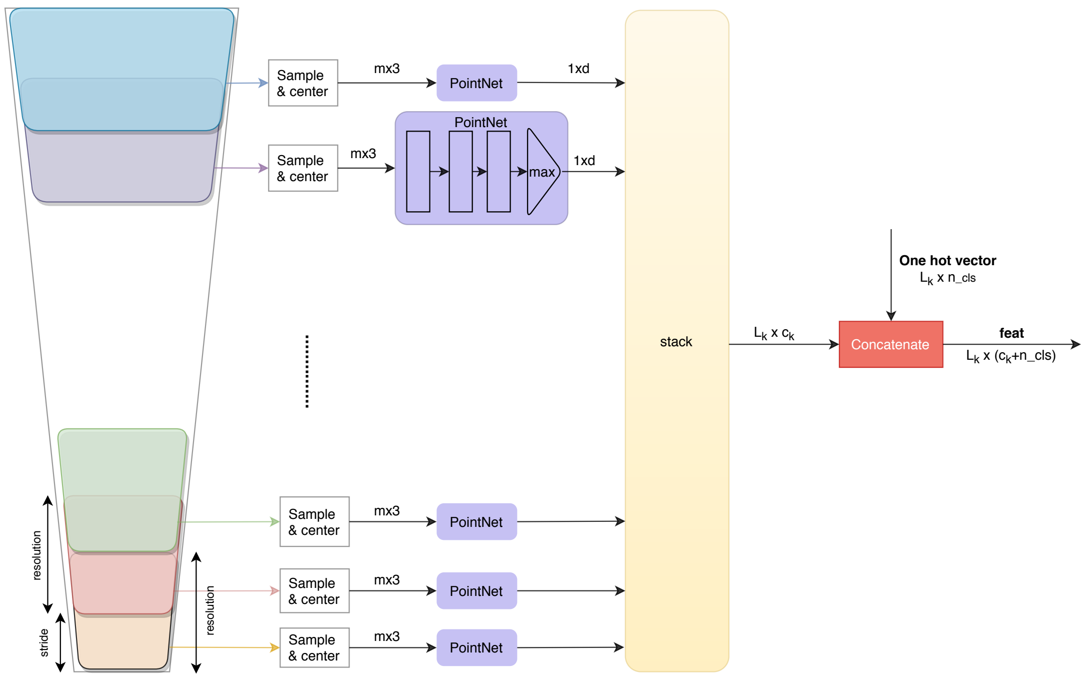


Figure 2.13: PointNet backbone illustration for a single resolution. The PointNet module is shared by each slice under the single resolution. The one-hot vector from 2D detection is concatenated in the last step.

2.4.3.2 Multi resolution PointNet backbone

The PointNet backbone as in Figure 2.13 is responsible for extracting feature in multiple resolutions. Firstly, we consider a single resolution case. Given a 2D detection, a fixed detection range $D_{\max}$, and a pair of resolution and stride (r_k, d_k), a sequence of overlapping frustum slices are generated to group the points in the frustum view. $D_{\max}$ determines the maximum detection distance from the origin of frustum coordinate, and objects lie outside $D_{\max}$ will not be detected. The length of each slice and the sliding stride are d_k and s_k respectively. In our implementation, $d_k = 2s_k$. The number of slices L_k is determined by $\frac{D_{\max}}{s_k}$. The points in each slices of the same resolution are randomly sampled and the coordinates are centered by subtracting the centroid of each slice. These sampled group of points are used as the input of a shared PointNet. The feature of each slice is obtained after a max operation after 3 MLP layers, and later stacked together with the order from close to far. Next, the stacked feature with shape $L_k \times d_k$ will concatenate a one-hot vector which represents the category detected by the 2D detector. The final output shape of a single resolution d_k is $L_k \times (d_k + n_{\text{cls}})$.

In order to extract to the features at different scales, more than one d_k are applied and each d_k has its own PointNet as feature extractor. Thus, s_k and L_k vary with resolution scales.

2.4.3.3 Fully convolutional layer

The illustration of a fully convolutional layer taking 4 features as input is in Figure 2.14. The fully convolutional layer takes the multi-resolution features concatenated with one-hot vector from the PointNet Backbone as input, and fuses these features in a cascaded structure. Convolution and deconvolution are used in this cascaded structure. The idea is that the more detailed feature should be aggregated at the earlier stage. The purpose of deconvolution layer is to make sure that the feature aggregated at all the early and late stage will have the same length $\tilde{L}$. The feature after deconvolution will be concatenated at the feature dimension as the enriched feature for the detection head.

We name each $i \in \{0, 1, ..., \tilde{L} - 1\}$ as virtual slice since the feature belongs to each virtual slice is not strictly from points in any explicitly separated slices, instead, it contains the feature close to the center of each virtual slice.

Each virtual slice has a ground truth binary label of foreground or background. In training mode, if the virtual slice center lies inside the γ_{min} times 3D ground-truth bounding box size scope, the virtual slice is assigned as foreground, if the virtual slice center lies in $(\gamma_{\text{min}}, \gamma_{\text{max}})$ times of the 3D ground-truth bounding box, the slice will be assigned as ignored, which means it will not contribute to the loss function. Virtual slice center lies outside the 3D ground-truth bounding box will be assigned as background.

2.4.3.4 Detection head

The detection head has two tasks. The first is to classify each virtual slice as foreground or background, i.e., whether the object of interest lies within this virtual slice. The output is a probability map of $\tilde{L} \times 2$.

The second task, i.e., 3D bounding box localization regression, will output all the parameters that can determine a 3D bounding box, including the center offset $\mathbf{\Delta x_c} = (\Delta x_c, \Delta y_c, \Delta z_c)$ in respect to corresponding slice centroid, orientation bin label $O_j \in \{0, ..., n_{\text{angle}} - 1\}$, orientation residual δ_j, size label S_u $(u \in \{0, ..., n_{\text{size}} - 1\})$ and size residual $(\sigma_u^w, \sigma_u^l, \sigma_u^h)$. n_{angle} and n_{size} are the predefined number of orientation bins and bounding box sizes. The illustration of the detection head following fully convolutional layer is in Figure 2.14.

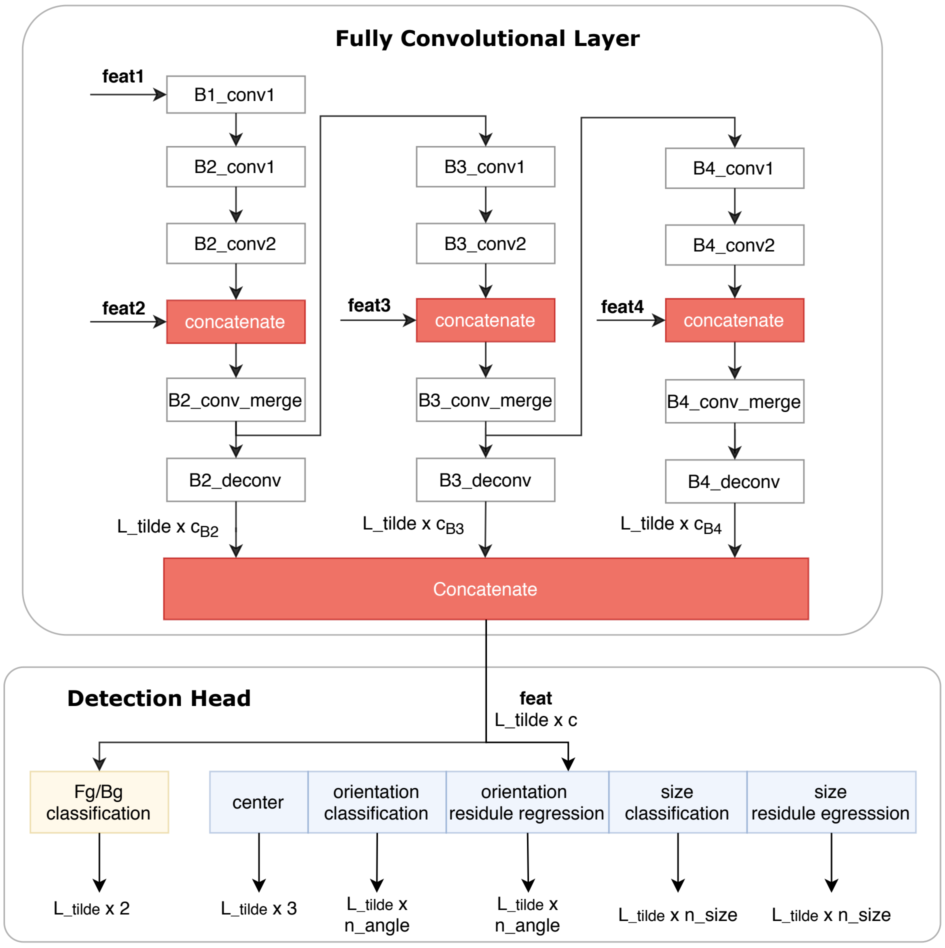


Figure 2.14: The structure of fully convolutional layer and detection head in Frustum ConvNet [4].

The detection head will classify which orientation and which size the input belongs to, and predict the residual value from the reference angle and size template. A predicted bounding box is parameterized as $(\mathbf{x_c}, w, l, h, \theta)$, where $\mathbf{x_c}$ is the box center representing (x_c, y_c, z_c), w, l, h is the width, length and height and θ is the orientation. The bounding box parameters can be obtained by the following equations

$$\mathbf{x_c} = \Delta\mathbf{x_c} + \mathbf{x_c}^{fg} \tag{2.5}$$
$$\theta = \theta_j + \delta_j \tag{2.6}$$
$$w = w_u(1 + \sigma_u^w) \tag{2.7}$$
$$l = l_u(1 + \sigma_u^l) \tag{2.8}$$
$$h = h_u(1 + \sigma_u^h), \tag{2.9}$$

where $\Delta\mathbf{x_c}$, δ_j, σ_u^w, σ_u^l, σ_u^h are five values directly output from the detection head. $\mathbf{x_c}^{fg}$ is the center of the some virtual slice which is classified as foreground. O_j is the predicted label of this virtual slice, and θ_j is the central degree of the bin label O_j. S_u is the predicted size label of this virtual slice, and w_u, l_u, h_u are the average width, length and height of S_u. Note that each slice can generate a 3D prediction, but we only keep the 3D prediction of the virtual slice with the highest foreground probability.

2.4.3.5 Loss function

The total loss function is

$$L = L_{\text{cls}} + \lambda_{\text{box}} \left(L_{\text{center}} + L_{\text{angle}} + L_{\text{size}} + L_{\text{corner}} \right), \tag{2.10}$$

where λ_{box} is a parameter to balance the loss of classification and the loss related to the bounding box.

L_{cls} represents the loss of foreground/background classification. The formula of L_{cls} is

$$L_{\text{cls}} = \sum_{\text{t}} -\alpha_{\text{t}}(1-p_{\text{t}})^{\gamma}\log(p_{\text{t}}), \qquad \text{t} \in \{\text{fg, bg}\}. \tag{2.11}$$

Considered the imbalance of foreground and background, focal loss [50] is used. p_{t} is the correct probability of foreground or background. For example, if a slice is labeled as foreground and has a probability in background at 0.8, then 0.2 is its correct probability of foreground.

L_{center} is the huber loss for the center regression with the following form

$$L_{\text{center}} = \begin{cases} \frac{1}{2}(\delta a d - \delta d^*)^2 & |\delta d - \delta d^*| \leq \epsilon \\ \epsilon|\delta d - \delta d^*| - \frac{1}{2}\epsilon^2 & \text{otherwise,} \end{cases} \tag{2.12}$$

where $\Delta d = ||\Delta \mathbf{x_c}||_2$ is a scalar that represents the length of the center offset $\Delta \mathbf{x_c}$, $\Delta d^* = ||\Delta \mathbf{x_c}^*||_2$ represents distance of the ground-truth center offset, and ϵ is hyper-parameter needs to be manually set.

L_{angle} is the loss of orientation, including the orientation classification loss and orientation residual regression loss. As defined by NuScenes official, traffic cone does not contribute to the orientation loss due to its symmetry. The classification loss uses cross-entropy loss, and residual loss uses huber loss. Similar to orientation loss, size loss is also composed by classification loss and residual regression loss.

Corner loss [31] is the sum of distance between predicted corners and ground truth corners, denoted as P_i^*, $i \in \{1, ..., 8\}$. It has the following formula

$$L_{\text{corner}} = \min(\sum_{i=1}^{8} ||P_i - P_i^*||_2, \sum_{i=1}^{8} ||P_i - P_i^{**}||_2). \tag{2.13}$$

Note that to avoid the huge penalty when the predicted box orientation has a flipped heading with respect to the ground truth box, we also computed the corner loss from the flipped ground-truth box corners, denote as P_i^{**}, $i \in \{1, ..., 8\}$. The smaller loss is the one to be optimized.

2.5 Literature review summary

Object detection can be divided into one-stage object detection and two-stage object detection. Two-stage detector has RPN which can be further divided into anchor-based proposal generation and anchor-free proposal generation. Generally speaking, two-stage methods have higher performance than one-stage methods but execute slower. Similarly, anchor-based proposal generation tends to have higher proposal recall compared with anchor-free proposal generation but is computationally more expensive.

Feature extraction is an important part in object detection. CNN is widely used as the feature extractor. For 3D object detection, the methods need to deal with the irregularly distributed 3D points in euclidean space. The methods can be categorised as PointNet based, graph-based and voxel-based.

Some 3D object detection architectures use only LIDAR point cloud as input data, while the others use multi-modal sensor. The sensor fusion method has two levels. The stronger level is directly concatenating features from different sensor. The weaker level is to use information from one sensor as prior information and data from another sensor as the main source for detection.

Class imbalance may exist in some datasets, proper sampling strategies and focal loss help to alleviate the negative influence of data imbalance.

3
Implementation & Experiments

In our work, we chose 2D+3D pipeline as our detection algorithm. The 2D detector provides the detected category and the 2D location on the camera plane. The 3D detector further utilizes these as prior information and localize the bounding box in 3D space.

We utilized this type of pipeline and choose camera and LIDAR as our environment perception sensors. Camera was chosen because the 2D detection technique based on it is widely researched and used in many aspects besides AD, including medical imaging, astronomy, streaming media, etc. [51–53]. With a solid research foundation, many state-of-art 2D detectors have been repeatedly verified in different context to ensure their reliability. Therefore it is reasonable to take an off-the-shelf 2D detector to narrow down search areas by feeding its result into a 3D detector for further detection.

In this thesis we chose faster R-CNN [22] as the 2D object detection model to replicate since it is an algorithm with meticulous logic targeting at high accuracy and has been iterated and improved several times by its inventors. Meanwhile, it still shows vitality in the latest literature [4, 52].

LIDAR was selected as the input data for 3D detector because of its high resolution and dense pointcloud. According to the exploration on NuScenes, LIDAR has generated point cloud on more objects than radar. Thus we chose LIDAR in order to make the final detection performance better.

The 3D object detection model that we chose to replicate was Frustum ConvNet [4] because it was a recent network that theoretically absorbed the advantages of VoxelNet [32] and F-PointNet [31], as described in Chapter 2, and has achieved good results on KITTI dataset [17].

In this chapter, we will present the implementation details of 2D and 3D detectors, and the corresponding experiments we did with them.

3.1 2D detector's architecture details

We implemented two versions of faster R-CNN as 2D detector: faster R-CNN with 16-layer version VGG as backbone (VGG16 backbone) and faster R-CNN with 101-

layer version ResNet based feature pyramid network as backbone (ResNet101 FPN backbone). NuScenes dataset provides images at the size of $3 \times 900 \times 1600$. We re-scale all images into $3 \times 563 \times 1000$ in the dataloader before sending them to both 2D detectors.

3.1.1 VGG16 backbone faster R-CNN

The configuration and output structure of each layer in VGG16 backbone is illustrated in Table 3.1. In this version of faster R-CNN, ROI pooling is used as ROI feature extractor.

Table 3.1: Parameters of the VGG16 backbone

Layer Type	kernel size, stride, padding	num kernel	output shape ($depth \times h \times w$)
conv	$3 \times 3, 2, 1$	64	$64 \times 563 \times 1000$
conv	$3 \times 3, 2, 1$	64	$64 \times 563 \times 1000$
max-pool	$2 \times 2, 2, 0$	/	$64 \times 282 \times 500$
conv	$3 \times 3, 2, 1$	128	$128 \times 282 \times 500$
conv	$3 \times 3, 2, 1$	128	$128 \times 282 \times 500$
max-pool	$2 \times 2, 2, 0$	/	$128 \times 141 \times 250$
conv	$3 \times 3, 2, 1$	256	$256 \times 141 \times 250$
conv	$3 \times 3, 2, 1$	256	$256 \times 141 \times 250$
conv	$3 \times 3, 2, 1$	256	$256 \times 141 \times 250$
max-pool	$2 \times 2, 2, 0$	/	$256 \times 71 \times 125$
conv	$3 \times 3, 2, 1$	512	$512 \times 71 \times 125$
conv	$3 \times 3, 2, 1$	512	$512 \times 71 \times 125$
conv	$3 \times 3, 2, 1$	512	$512 \times 71 \times 125$
max-pool	$2 \times 2, 2, 0$	/	$512 \times 36 \times 63$
conv	$3 \times 3, 2, 1$	512	$512 \times 36 \times 63$
conv	$3 \times 3, 2, 1$	512	$512 \times 36 \times 63$
conv	$3 \times 3, 2, 1$	512	$512 \times 36 \times 63$
max-pool	$2 \times 2, 2, 0$	/	$512 \times 36 \times 63$

3.1.2 ResNet101 FPN backbone faster R-CNN

We built the FPN backbone structure based on ResNet101, as illustrated in Figure 3.1. Pre-trained ResNet101 model on ImageNet [54] was used for initialization.

ResNet101 consists of five cascaded units, conv1, conv2_x, conv3_x, conv4_x and conv5_x. Conv1 is an ordinary 7×7 conv layer. All the other units are composed of several cascaded residual blocks. The tensors output by conv2_x to conv5_x represent the features from a fine resolution to a coarse resolution. These tensors will be processed further in the feature pyramid structure to generate their corresponding feature map.

In the feature pyramid structure, the tensors output by conv2_x to conv5_x are processed by one convolutional layer to get the same depth dimension. Then the output from conv$k$_x ($k = 3, 4, 5$) is up-sampled and element-wise added with the output from conv($k-1$)_x. Feature map P5 will be simply down-sampled by max-pooling to get the feature map P6. All of the feature maps will be used by RPN while feature map P6 will not be passed to ROI detection head.

In the ResNet101 FPN version, we chose ROI align as the ROI feature extractor.

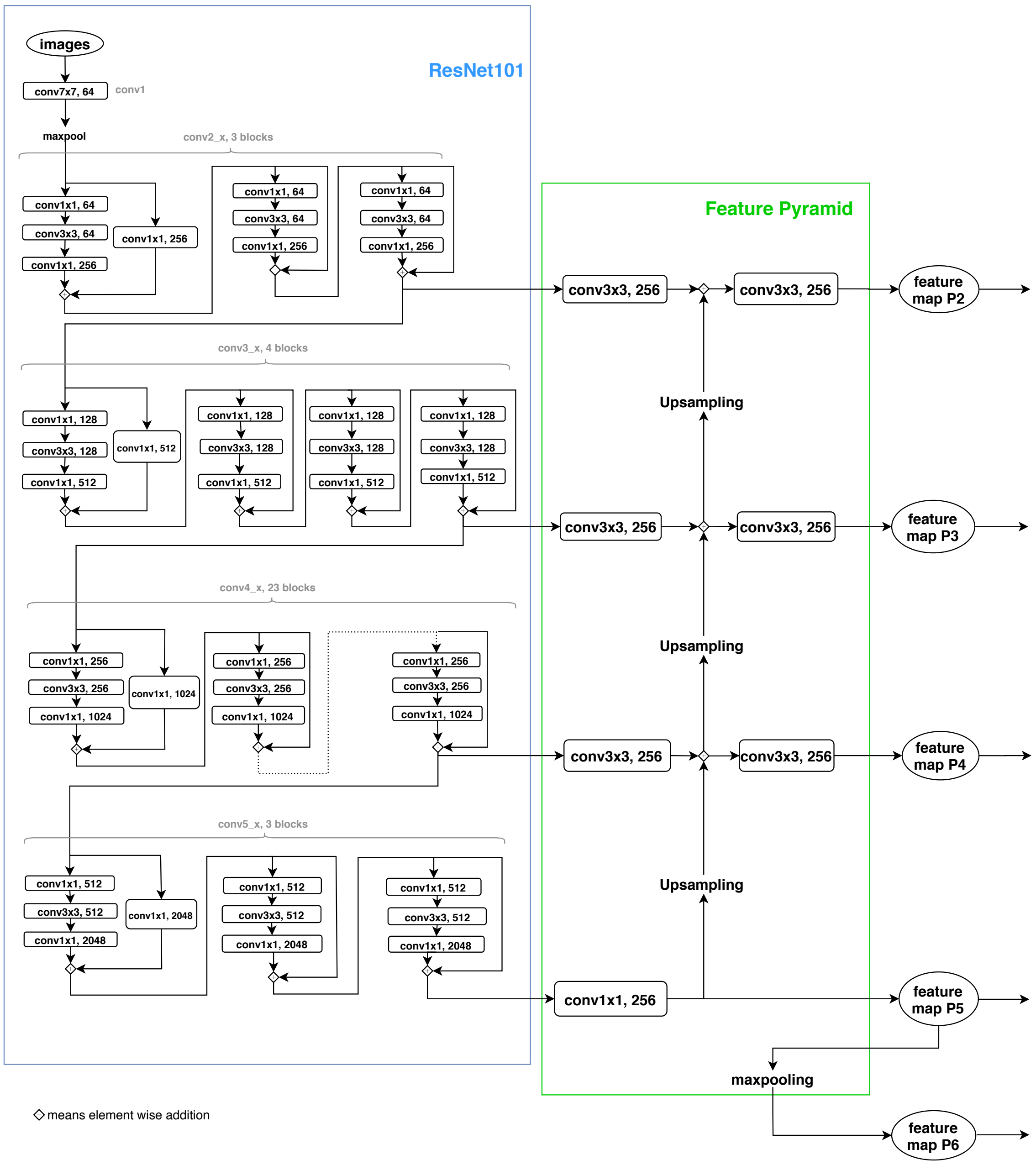


Figure 3.1: ResNet101 FPN backbone structure

3.2 3D detector's implementation details

The detailed parameters of our implementation of Frustum ConvNet is introduced in this chapter.

3.2.1 Multi resolution PointNet backbone

We set the detecting range $D_{\max} = 50\,\mathrm{m}$, and extract features of 4 resolutions. The stride s_k, frustum resolution d_k, MLP layer and feature shape of each resolution is presented in Table 3.2.

Table 3.2: Configuration of multi-resolution PointNet backbone, unit for strides and resolutions is meter.

feat_k	stride s_k	resolution d_k	feat shape $L_k \times (c_k + n_{\mathrm{cls}})$	MLP
feat1	0.25	0.5	$200 \times (128 + 10)$	[64, 64, 128]
feat2	0.5	1.0	$100 \times (128 + 10)$	[64, 64, 128]
feat3	1.0	2.0	$50 \times (256 + 10)$	[128, 128, 256]
feat4	2.0	4.0	$25 \times (512 + 10)$	[256, 256, 512]

3.2.2 Fully convolutional layer

The detailed parameters of the fully convolutional layer is in Table 3.3. The number of foreground categories $n_{\mathrm{cls}} = 10$, according to the NuScenes dataset. The threshold value $\gamma_{\min} = 0.5$ and $\gamma_{\max} = 1.0$. The number of virtual slices $\tilde{L} = 100$.

Table 3.3: The parameters of fully convolutional layer

Module Name	**nc_in**	**length_in**	**nc_out**	**length_out**	**kernel, stride, padding**
B1_conv1	138	200	128	200	3, 1, 1
B2_conv1	128	200	128	100	3, 2, 1
B2_conv2	128	100	128	100	3, 1, 1
B2_convmerge	128+138	100	128	100	1, 1, 0
B3_conv1	128	100	256	50	3, 2, 1
B3_conv2	256	50	256	50	3, 1, 1
B3_convmerge	256+266	100	256	50	1, 1, 0
B4_conv1	256	50	512	25	3, 2, 1
B4_conv2	512	25	512	25	3, 1, 1
B4_convmerge	512+522	25	512	25	1, 1, 0
B2_deconv	128	100	256	100	1, 1, 0
B3_deconv	256	50	256	100	1, 1, 0
B4_deconv	512	25	256	100	1, 1, 0

3.2.3 Detection head

Note that what we did here is different from the original paper [4]. In [4], the category classification is conducted in the 3D detector by setting the probability map as $\tilde{L} \times (n_{\text{cls}} + 1)$. However, according our experiments, we found that the training loss of 3D detector does not go down if we output 11 probability map for NuScenes dataset. To make this branch work, we made the modification and only predict the foreground/background probability instead of a more detailed category classification. The slice with the highest foreground probability, which is also known as detection score, is considered as the slice where the object lies.

$n_{\text{angle}} = 12$ orientation bins and their range are defined in Table 3.4. We set the each foreground category a predefined bounding box size, as in Table 3.5.

Table 3.4: Frustum ConvNet detection head orientation label definition

orientation bin	angle range
0	$[-\pi, -\frac{10}{12}\pi)$
1	$[-\frac{10}{12}\pi, -\frac{8}{12}\pi)$
2	$[-\frac{8}{12}\pi, -\frac{6}{12}\pi)$
3	$[-\frac{6}{12}\pi, -\frac{4}{12}\pi)$
4	$[-\frac{4}{12}\pi, -\frac{2}{12}\pi)$
5	$[-\frac{2}{12}\pi, 0)$
6	$[0, \frac{2}{12}\pi)$
7	$[\frac{2}{12}\pi, \frac{4}{12}\pi)$
8	$[\frac{4}{12}\pi, \frac{6}{12}\pi)$
9	$[\frac{6}{12}\pi, \frac{8}{12}\pi)$
10	$[\frac{8}{12}\pi, \frac{10}{12}\pi)$
11	$[\frac{10}{12}\pi, \pi)$

Table 3.5: Mean width, length, height of each detection category by calculating through all ground-truth in NuScenes dataset

category name	(w_u, l_u, h_u)
barrier	(2.3, 0.61, 1.1)
bicycle	(0.64, 1.80, 1.40)
bus	(3.0, 11.0, 3.8)
cars	(1.91, 4.63, 1.71)
construction vehicle	(2.6, 5.6, 2.4)
motorcycle	(0.68, 2.0, 1.5)
pedestrian	(0.68, 0.74, 1.8)
traffic cone	(0.47, 0.45, 0.78)
trailer	(2.3, 10.0, 3.7)
truck	(2.4, 6.5, 2.6)

3.2.4 Loss function

λ_{box} in Eq 2.10 is set as 0.01 to manually balance the magnitude difference between different losses in our case. The focal loss weight in Eq 2.11 is set as $\alpha_{\text{bg}} = 0.75$, $\alpha_{\text{bg}} = 0.25$, which keeps the same with [4]. The hyper-parameter in 2.12 is 3.0.

3.3 Training settings

3.3.1 The training settings of faster R-CNN

The training details are listed in Table 3.6.

Table 3.6: Training configuration of both faster R-CNN with different backbones.

backbone	batch size	optimizer	lr	epoch	pretrained
VGG16	6	SGD	0.001	14	caffe-VGG16
ResNet101 FPN	6	SGD	0.001	14	PyTorch-ResNet101

3.3.2 The training settings of Frustum ConvNet

Before training, the ground-truth data are wrapped for the F-ConvNet. The wrapped data includes: camera channel, category string, 2D bounding box, 3D bounding box, camera pose, camera intrinsic, ego pose, LIDAR point cloud. The 2D bounding box are projected to the image canvas from the 3D bounding box, and if the projected 2D box intersects the boundary of the image canvas, then the projected 2D bounding box is cropped within the image canvas.

The training data is screened by the following criterion. The object with 3D bounding box includes 0 LIDAR points, or the object that has visibility level lower than "3", or the projected 2D bounding box have an intersection with the image less than 20% will be discarded from the training dataset. After the screening, the remaining number of sampling annotation is 68.15% of all the annotations in the whole dataset.

The training configurations are as Table 3.7. The model was only trained for two epochs due to time limitation.

Table 3.7: Training configuration of Frustum ConvNet.

batch size	epoch	optimizer	lr	betas	eps
35	2	ADAM	1.0×10^{-4}	$(0.9, 0.999)$	1.0×10^{-4}

3.4 Evaluation metrics

The average precision (AP) [55] is calculated from precision and recall. Taking true positive (TP), false positive (FP), false negative (FN) as input,

$$\text{precision} = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}} \tag{3.1}$$

$$\text{recall} = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground-truths}}. \tag{3.2}$$

For 2D detection part, whether a detection is decided as a correct detection, i.e., TP, or a wrong detection, i.e., FP, is by comparing the IoU between predicted bounding boxes and ground-truth bounding boxes. In our case, IoU ≥ 0.5 is regarded as TP. Further AP calculation follows PASCAL VOC standard [56]. The mean AP among all categories is named mean average precision (mAP).

For 3D detection part, TP or FP is calculated by comparing the distance between the center of predicted bounding boxes and ground-truth bounding boxes. The further AP computation uses NuScenes standard [9].

NuScenes dataset defines six metrics, but in this thesis we only focus on the following four metrics when doing 3D detection evaluation. They are one precision metric and three error metrics:

1. AP: average precision.
2. ATE: Euclidean center distance in 2D in meters.
3. ASE: Calculated as 1 - IoU after aligning centers and orientation.
4. AOE: Smallest yaw angle difference between prediction and ground-truth in radian. Orientation error is evaluated at 360 degree for all classes except barriers where it is only evaluated at 180 degrees. Orientation errors for cones are ignored in any condition due to its symmetry.

Note that when indicating the mean results on all categories, the above metrics are renamed as mAP, mATE, mASE, mAOE respectively.

3.5 2D-3D joint detection process

We evaluate the joint 2D-3D detector by the steps illustrated in Figure 3.2. After the 2D detection procedure is finished, the predicted 2D bounding boxes with a score higher than S_{eval} are sent to 2D detection evaluation metric to calculate the AP for 2D detection network part. Meanwhile, the predicted 2D bounding boxes with a score higher than S_{to3D} are selected as the prior information input to the 3D detector. Each 2D detection that is fed into F-ConvNet generates its corresponding 3D detection. Since a single object in 3D is likely to be visible in more than one camera image, thus several 2D detection results which belong to the same object might have their corresponding 3D detection bounding boxes overlapping. Considering this, we perform category-wise NMS among the 3D predictions to eliminate the overlapping issue. The NMS IoU is set in the context of BEV. The final step is to calculate the NuScenes evaluation metrics on the post NMS 3D detection results using NuScenes evaluation standard. Note that we also calculate the NuScenes evaluation metrics based on the ground-truth 2D bounding boxes annotations in order to get the quantitative performance of a benchmark 3D detector regardless of the performance of 2D detector.

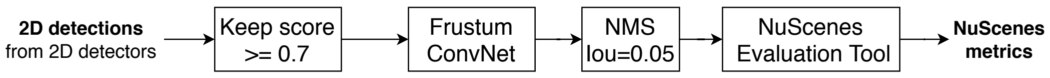


Figure 3.2: The evaluation process of 3D detector.

3.6 Experiments design

3.6.1 Experiments on 2D detector

The NuScenes dataset contains three different weather and lightning conditions, namely normal scene, rain scene and night scene. For 2D detectors, we calculated the mAP, AP of each category at the whole validation set, and also each scene separately. We not only wanted to see the 2D detectors' performance on each category, but also to see how the different scene can influence the detection results.

NuScenes dataset defined 4 visibility levels to describe the occlusion level of an object. The definition table is shown in Table 3.8. We evaluated the detection performance of two 2D detectors under different visibility levels separately to see how the occlusion affect the performance of 2D detector.

Table 3.8: Definition of visibility level in NuScenes dataset

Visibility code	Definition
"1"	Fraction of whole object's pixels visible in all the images collected from 6 different cameras is between 0 and 40%
"2"	Fraction of whole object's pixels visible in all the images collected from 6 different cameras is between 40 and 60%
"3"	Fraction of whole object's pixels visible in all the images collected from 6 different cameras is between 60 and 80%
"4"	Fraction of whole object's pixels visible in all the images collected from 6 different cameras is between 80 and 100%

3.6.2 Experiments on 3D detector

Our experiments on 3D detectors have two parts. We first used the ground-truth 2D detections as input for 3D detector to calculate the mAP, mATE, mASE, mAOE and AP, ATE, ASE, AOE of each category at the whole validation scene and each scene separately. By calculating the NuScenes metrics on ground-truth 2D detections, we can know the inherent characteristics of 3D detector, without influence by the 2D detector. We were also interested in how our 3D detector performs different under different scenes.

We also did the same experiments given the input as the 2D detection results from two 2D detectors. By comparing the benchmark results and 2D-3D joint detection results, we are able to see how the 2D detections influence the results of 3D detections.

4
Results

This chapter will show the 2D object detection results from two 2D detectors, i.e., faster R-CNN with two different backbones, 3D object detection results using ground truth 2D bounding boxes and 2D bounding boxes coming from 2D detectors.

4.1 2D detection results

We implemented two faster R-CNN models using different backbones, namely VGG16 and ResNet101 FPN, as described in chapter 2 and 3. Both models are trained and validated on NuScenes dataset v1.0-trainval version. 2D bounding boxes ground-truth is obtained by projecting 3D bounding boxes annotations onto each camera plane. This section will show the performance of the two 2D detectors under different conditions.

4.1.1 Quantitative results

4.1.1.1 Scene agnostic, visibility agnostic results

The mAP of VGG16 backbone faster R-CNN is obviously higher than that of ResNet101 FPN backbone faster R-CNN, as shown in Table 4.1. When we list all AP of each category in Table 4.2, it is found that the AP in each category of ResNet101 FPN backbone faster R-CNN is relatively uniformly lower than that of VGG16 backbone faster R-CNN, which proves that VGG16 backbone has a better performance than ResNet101 FPN backbone in this context by being able to detect more accurately in each category instead of for example by having extremely unbalanced AP.

Table 4.1: mAP of faster R-CNN with VGG16 backbone and ResNet101 FPN backbone

Backbone	mAP
VGG16	0.441
ResNet101 FPN	0.3254

Table 4.2: AP for each category of both 2D detectors

Category	barrier	bicycle	bus	car	const. vehicle
VGG16	0.442	0.392	0.62	0.64	0.204
ResNet101 FPN	0.339	0.263	0.515	0.54	0.129
Category	motorcycle	pedestrian	traffic cone	trailer	truck
VGG16	0.429	0.448	0.476	0.29	0.467
ResNet101 FPN	0.245	0.347	0.309	0.218	0.349

4.1.1.2 Scene Aware, visibility agnostic results

Considering that camera is sensitive to illumination, we evaluate the performance of our 2D detector under normal scene, rain scene and night scene separately. Table 4.3 shows that the values of normal scene mAP and rain scene mAP are really close while night scene mAP suffers an obvious decrease.

Table 4.3: mAP for normal, rain and night scene of both 2D detectors

Backbone	Normal mAP	Rain mAP	Night mAP
VGG16	0.443	0.44	0.337
ResNet101 FPN	0.329	0.322	0.219

Table 4.4: AP for each category in normal, rain and night scene

Normal scene	barrier	bicycle	bus	car	const. vehicle
VGG16	0.426	0.406	0.629	0.633	0.206
ResNet101 FPN	0.326	0.264	0.524	0.531	0.133
Normal scene	motorcycle	pedestrian	traffic cone	trailer	truck
VGG16	0.442	0.453	0.483	0.278	0.473
ResNet101 FPN	0.262	0.351	0.316	0.222	0.357
Rain scene	barrier	bicycle	bus	car	const. vehicle
VGG16	0.493	0.409	0.581	0.656	0.191
ResNet101 FPN	0.421	0.269	0.467	0.542	0.148
Rain scene	motorcycle	pedestrian	traffic cone	trailer	truck
VGG16	0.458	0.356	0.459	0.339	0.458
ResNet101 FPN	0.268	0.274	0.282	0.212	0.333
Night scene	barrier	bicycle	bus	car	const. vehicle
VGG16	0.299	0.287	nan	0.631	nan
ResNet101 FPN	0.19	0.195	nan	0.544	nan
Night scene	motorcycle	pedestrian	traffic cone	trailer	truck
VGG16	0.34	0.348	0.072	nan	0.385
ResNet101 FPN	0.141	0.256	0.002	nan	0.207

According to Table 4.4, compared with the normal scene, generally category AP in the rain scene has no significant difference. For some categories, AP is slightly higher in normal scene and for others, AP is slightly higher in rain scene. But the difference is minor. For night scene, both VGG16 and ResNet101 FPN backbone faster R-CNN models do not have AP results, i.e., the result is not a number (nan), for bus, construction vehicle and trailer because there is no corresponding ground truth in NuScenes dataset. Most categories in night scene have a much lower AP than that in the normal scene or rain scene. For traffic cone, the decrease is dramatic. Notably for car, the AP does not show obvious fluctuation among normal, rain and night scene.

4.1.1.3 Scene agnostic, visibility aware results

For 2D object detection based on camera, occlusion is part of the main issues. If an object is highly occluded in an image, which means the camera can not really see the object, the detection will be severely affected.

Table 4.5: mAP for different visibility of both 2D detectors

backbone	“1” mAP	“2” mAP	“3” mAP	“4” mAP
VGG16	0.046	0.067	0.153	0.491
ResNet101 FPN	0.026	0.036	0.1	0.368

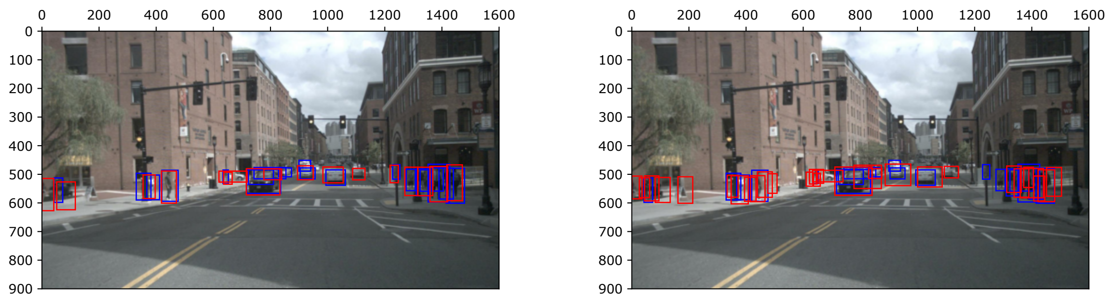
From Table 4.5 it is observed that visibility loss, i.e. occlusion or truncation, has a significant impact on both VGG16 and ResNet101 FPN backbone faster R-CNN models. Surprisingly, even with visibility code “3”, which means more than half of an object is visible, the mAP is dramatically lower than that of visibility code “4”. Table 4.6 shows the AP of each category under different visibility. From visibility code “1” to visibility code “2”, AP of all categories for both VGG16 and ResNet101 FPN backbone faster R-CNN models increase slightly but are still too low to give very meaningful detection results. In visibility code “3”, ResNet101 FPN backbone has a slight higher AP in car and traffic cone although in other categories it lags behind with a certain gap, but VGG16 backbone overtakes ResNet101 FPN backbone completely in visibility code “4” in all categories by a large margin.

Table 4.6: AP for each category in different visibility

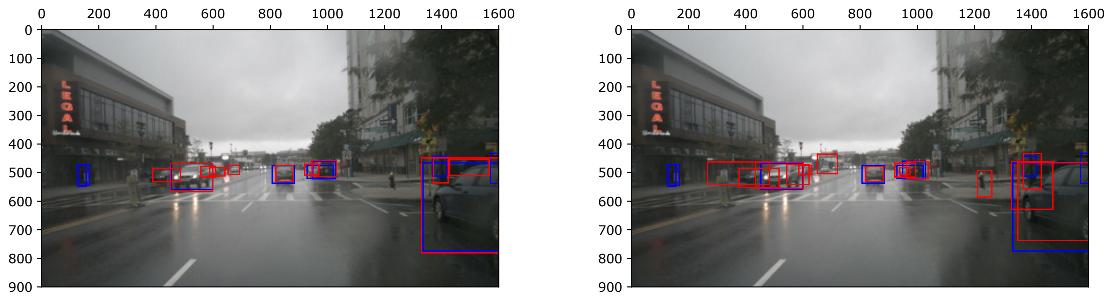
Visibility "1"	barrier	bicycle	bus	car	const. vehicle
VGG16	0.02	0.112	0.009	0.061	0.026
ResNet101 FPN	0.015	0.021	0.007	0.045	0.007
Visibility "1"	motorcycle	pedestrian	traffic cone	trailer	truck
VGG16	0.027	0.04	0.009	0.138	0.019
ResNet101 FPN	0.014	0.04	0.004	0.09	0.012
Visibility "2"	barrier	bicycle	bus	car	const. vehicle
VGG16	0.029	0.134	0.099	0.104	0.04
ResNet101 FPN	0.02	0.029	0.055	0.074	0.019
Visibility "2"	motorcycle	pedestrian	traffic cone	trailer	truck
VGG16	0.033	0.101	0.016	0.063	0.05
ResNet101 FPN	0.011	0.064	0.008	0.054	0.03
Visibility "3"	barrier	bicycle	bus	car	const. vehicle
VGG16	0.166	0.233	0.26	0.174	0.092
ResNet101 FPN	0.08	0.075	0.179	0.205	0.052
Visibility "3"	motorcycle	pedestrian	traffic cone	trailer	truck
VGG16	0.145	0.112	0.033	0.154	0.163
ResNet101 FPN	0.057	0.081	0.056	0.12	0.094
Visibility "4"	barrier	bicycle	bus	car	const. vehicle
VGG16	0.481	0.485	0.619	0.686	0.245
ResNet101 FPN	0.377	0.323	0.531	0.618	0.161
Visibility "4"	motorcycle	pedestrian	traffic cone	trailer	truck
VGG16	0.509	0.526	0.54	0.3	0.522
ResNet101 FPN	0.274	0.414	0.354	0.218	0.408

4.1.2 Qualitative results

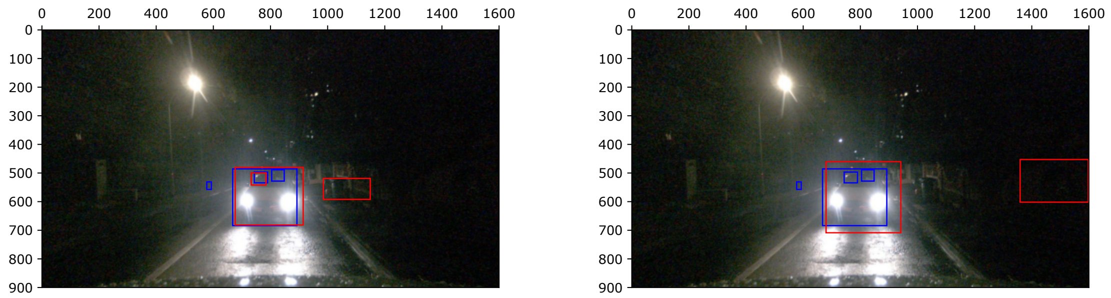
Figure 4.1 shows some qualitative detection results in different scenes. Intuitively, ResNet101 FPN backbone faster R-CNN is making predictions more aggressively than VGG16 backbone faster R-CNN, which is also proved by its much larger prediction results file. It also generates more false positives than VGG16 backbone faster R-CNN. As we can see, in normal scene and rain scene, the number of prediction bounding boxes in the right column are significantly much more than those in the left column. In night scene, considering the fact that the objects moving around ego vehicle are significantly fewer, the number of both ground truth bounding boxes and predictions is smaller. It is surprising to see how precise the predictions are in the left column of last line even under such a dark condition.



(a) normal scene



(b) rain scene



(c) night scene

Figure 4.1: faster R-CNN 2D bounding boxes output illustration for different weather and lightning conditions, only showing bounding boxes with score >=0.7. Left column: VGG16 backbone, Right column: ResNet101 FPN backbone. Blue rectangle: ground truth, Red rectangle: detection box.

4.2 Benchmark 3D detection results

The benchmark detection result represents the upper-bound performance of our 3D detector. Since the benchmark results are calculated based on the ground-truth 2D detection, and the 2D detection from the real 2D detector cannot surpass the ground truth. These results are presented in Table 4.7 to 4.9.

4.2.1 Category agnostic, scene agnostic results

Table 4.7: Benchmark 3D detector results: all scenes, all categories

Benchmark			
mAP	mATE	mASE	mAOE
0.5955	0.4421	0.3982	1.0006

4.2.2 Category aware, scene agnostic results

Table 4.8: AP, ATE, ASE, AOE on each category

	Benchmark				
Category	car	truck	bus	trailer	const.vehicle
AP	0.725	0.584	0.475	0.191	0.355
ATE	0.271	0.479	0.885	0.984	0.678
ASE	0.316	0.423	0.553	0.529	0.515
AOE	0.381	0.631	0.755	1.492	1.397
Category	pedestrain	motorcycle	bicycle	traffic cone	barrier
AP	0.774	0.628	0.681	0.835	0.707
ATE	0.133	0.352	0.263	0.117	0.259
ASE	0.306	0.363	0.338	0.310	0.329
AOE	1.421	1.189	1.652	/	0.086

In Table 4.8, given the ground-truth 2D detection as input, the AP varies with the category. Car, pedestrian and traffic cone are the top three categories in AP, while the lowest AP is achieved on trailer. The ATE is the lowest at traffic cone and highest on trailer. The AOE of barrier is the lowest while bicycle has the highest AOE.

4.2.3 Category agnostic, scene aware results

From Table 4.9, we noticed that the mAP in rain and night scene are higher than that in normal scene. While mATE, mASE and mAOE do not show a specific pattern related to the scene condition.

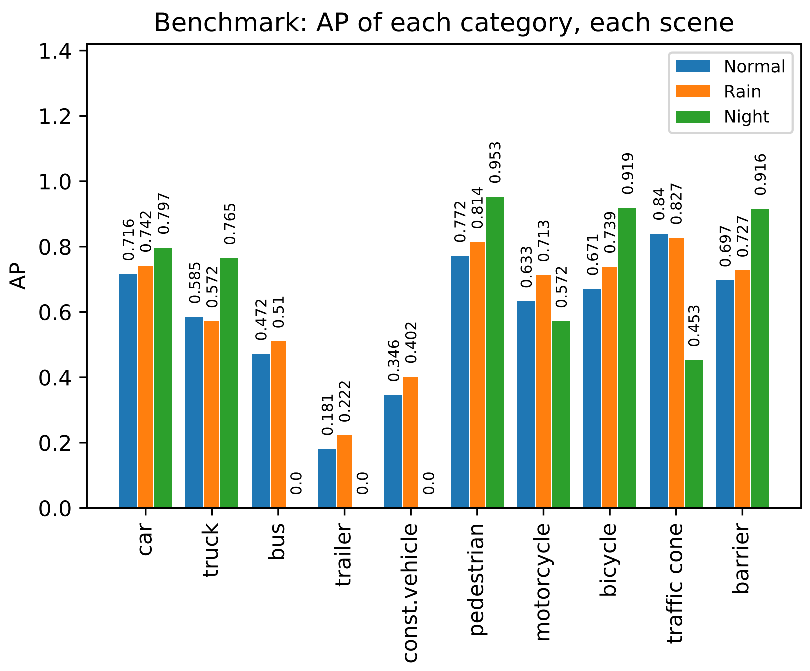
Table 4.9: mAP, mATE, mASE, mAOE on normal, rain, night scenes

Benchmark				
scene	mAP	mATE	mASE	mAOE
normal*	0.7020	0.2656	0.3390	0.8770
rain*	0.7334	0.2514	0.3600	0.8780
night*	0.7679	0.2956	0.3141	1.0357

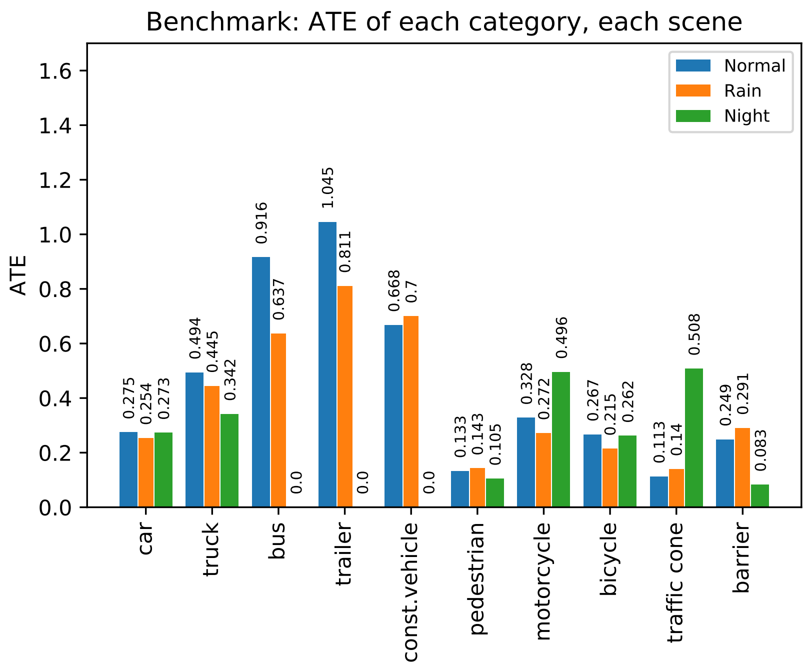
Since there is no ground-truth bus, trailer and construction vehicle in night scene, the mAP, mATE, mASE values in Table 4.9 are calculated only on 7 categories (car, truck, pedestrian, motorcycle, bicycle, traffic cone, barrier). The mAOE is calculated on 6 categories (car, truck, pedestrian, motorcycle, bicycle, barrier), since mAOE of traffic cone is ignored by the NuScenes standard. All the scenes evaluated on a subset of all the 10 categories are marked with *.

4.2.4 Category aware, scene aware results

The AP, ATE, ASE and AOE of each category at each scene are illustrated in Figure 4.2. It is interesting that for truck, pedestrian, bicycle and barrier, the AP in the night is significantly higher than that in the other two scenes, while the traffic cone's AP in the night is much lower than that in the other two scenes. Also, we noticed that the ATE of bus and trailer is significantly higher in normal scenes than the rain scene.

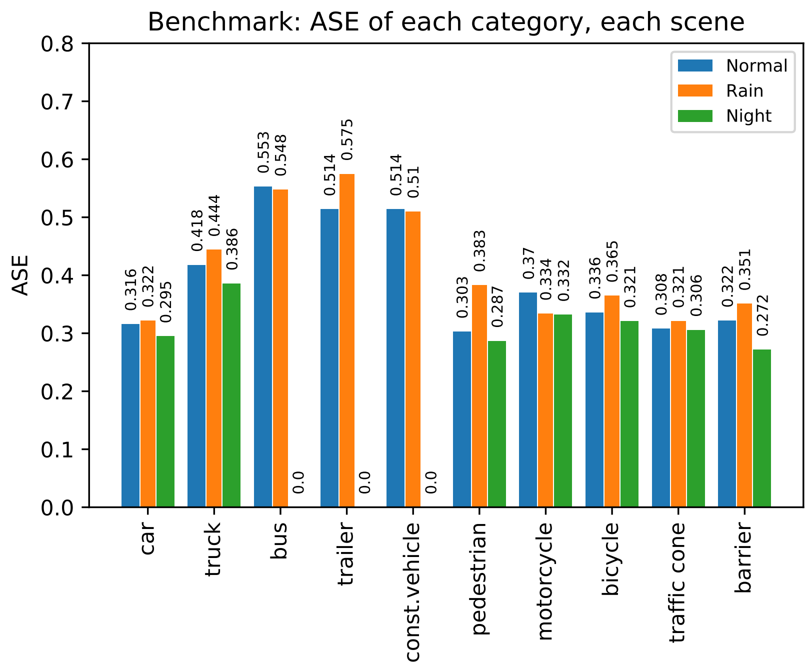


(a) AP

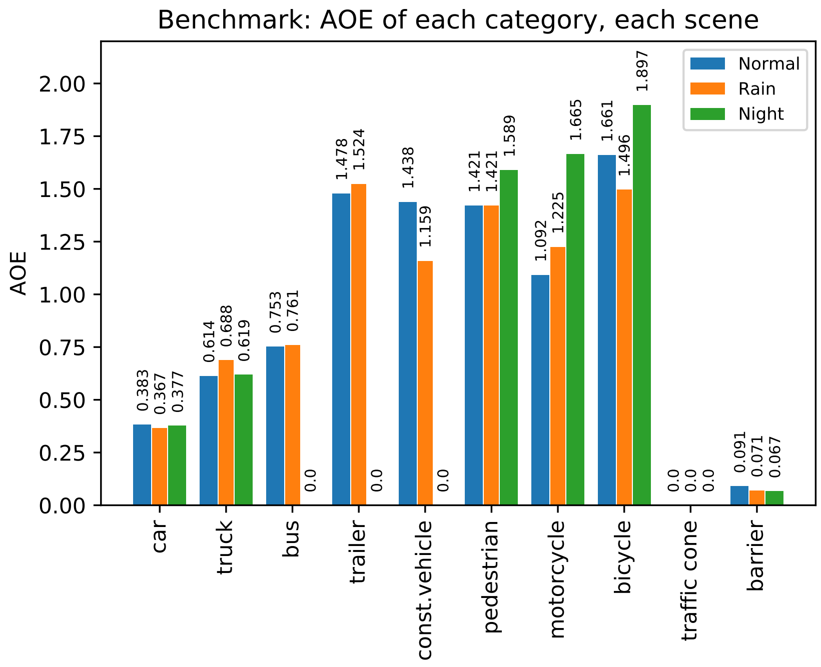


(b) ATE

Figure 4.2: Benchmark AP, ATE, ASE, AOE for each category, each scene.



(c) ASE



(d) AOE

Figure 4.2: Benchmark AP, ATE, ASE, AOE for each category, each scene.

4.3 2D-3D joint detection results

In this section, the 2D detection results from two faster R-CNN models are fed to F-ConvNet. All evaluation metrics used in this section are the same as in benchmark results.

4.3.1 Quantitative results

The quantitative results are presented in a similar way as the benchmark 3D detection results. They are listed in Table 4.10 to 4.12 and Figure 4.3 to 4.6.

4.3.1.1 Category agnostic, scene agnostic results

From the overall detection results in Table 4.10, the mAP of VGG16 backbone faster R-CNN is around 50% higher than ResNet101 FPN backbone faster R-CNN. The error metrics of VGG16 backbone faster R-CNN are also lower than those of ResNet101 FPN backbone.

Table 4.10: The four NuScenes metrics of 2D-3D joint detection results on all scenes, all categories.

2D detector backbone	mAP	mATE	mASE	mAOE
VGG16	0.3649	0.4832	0.4116	0.9715
ResNet101 FPN	0.2589	0.5826	0.4292	1.0425

4.3.1.2 Category aware, scene agnostic results

According to Table 4.11, the mAP of car, pestrain and traffic cone are the top three in both 2D detectors, while trailer and construction vehicle achieve the poorest mAP.

Table 4.11: AP, ATE, ASE, AOE on all categories for both faster R-CNN with different backbones

		VGG16			
category	car	truck	bus	trailer	const.vehicle
AP	0.609	0.307	0.346	0.057	0.118
ATE	0.302	0.559	0.892	0.990	0.800
ASE	0.315	0.421	0.567	0.515	0.557
AOE	0.381	0.616	0.672	1.424	1.417
category	pedestrain	motorcycle	bicycle	traffic cone	barrier
AP	0.519	0.369	0.336	0.581	0.408
ATE	0.168	0.328	0.263	0.165	0.365
ASE	0.333	0.362	0.331	0.366	0.351
AOE	1.405	1.101	1.640	/	0.087
		ResNet101 FPN			
category	car	truck	bus	trailer	const.vehicle
AP	0.525	0.208	0.222	0.023	0.068
ATE	0.356	0.712	0.981	0.998	0.853
ASE	0.311	0.447	0.567	0.519	0.581
AOE	0.463	0.750	0.882	1.356	1.381
category	pedestrain	motorcycle	bicycle	traffic cone	barrier
AP	0.434	0.090	0.198	0.491	0.332
ATE	0.215	0.490	0.401	0.276	0.542
ASE	0.341	0.379	0.369	0.406	0.372
AOE	1.445	1.354	1.651	/	0.100

4.3.1.3 Category agnostic, scene aware results

As shown in Table 4.12, for both 2D detectors, the mAP of night scene is significantly lower than that in normal and rain. mASE and mAOE do not show visible connection to the scene for both faster R-CNN while mATE also does not show significant scene aware difference for VGG16 backbone faster R-CNN.

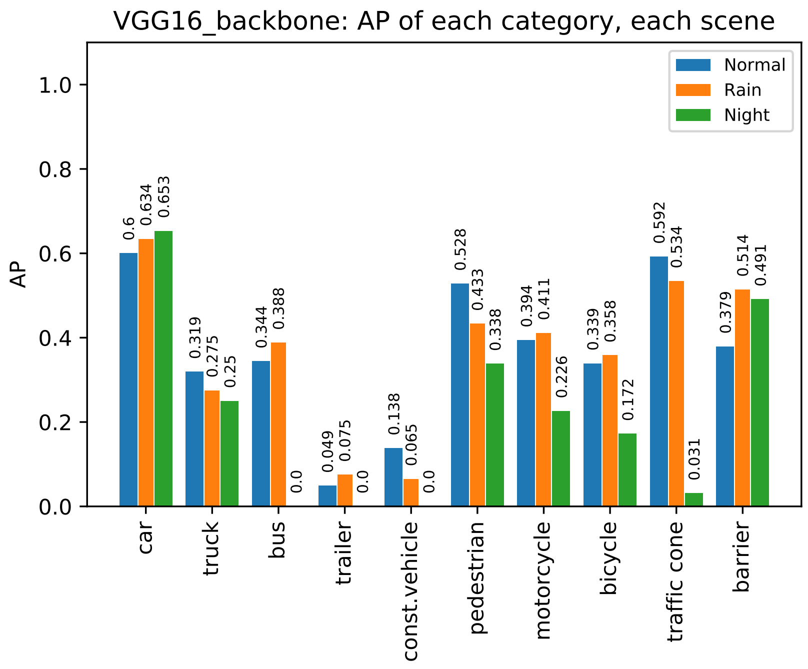
Table 4.12: AP, ATE, ASE, AOE on normal, rain, night scenes of joint detection for both versions of 2D detectors.

VGG16				
scene	mAP	mATE	mASE	mAOE
normal*	0.4501	0.3036	0.3520	0.8555
rain*	0.4513	0.3027	0.3739	0.9262
night*	0.3087	0.4106	0.3186	1.0240
ResNet101 FPN				
scene	mAP	mATE	mASE	mAOE
normal*	0.3351	0.4247	0.3743	0.9463
rain*	0.2716	0.4240	0.3891	1.0267
night*	0.1986	0.4449	0.3357	1.0900

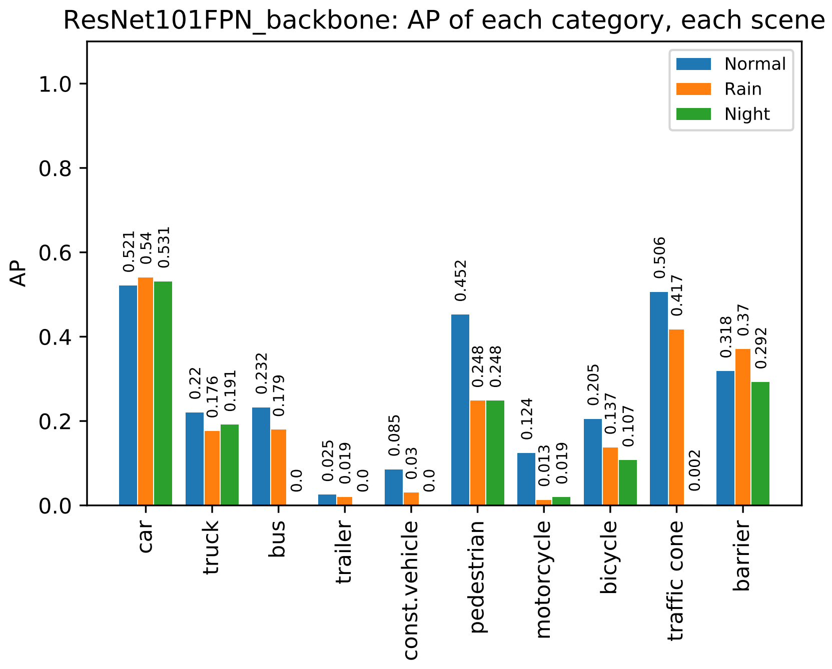
Since there does not exist ground-truth bus, trailer and construction vehicle in night scene, the mAP, mATE, mASE values in Table 4.12 are calculated only on 7 categories (car, truck, pedestrian, motorcycle, bicycle, traffic cone, barrier). The mAOE is calculated on 6 categories (car, truck, pedestrian, motorcycle, bicycle, barrier), since mAOE of traffic cone is ignored by the NuScenes standards. All the scenes evaluated only on a subset of all the 10 categories are marked with *.

4.3.1.4 Category aware, scene aware results

The AP, ATE, ASE, AOE of each category and each scene are shown in Figure 4.3 to 4.6.

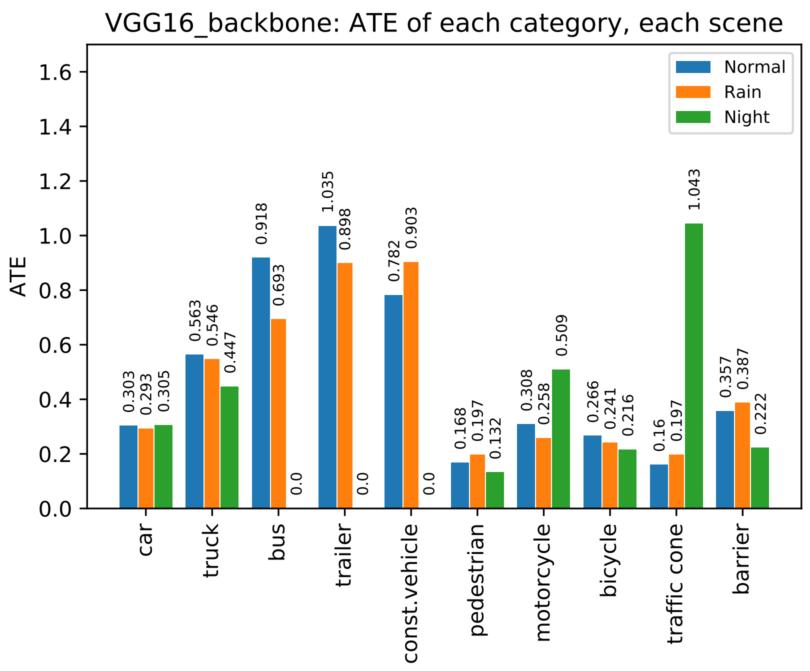


(a) VGG16

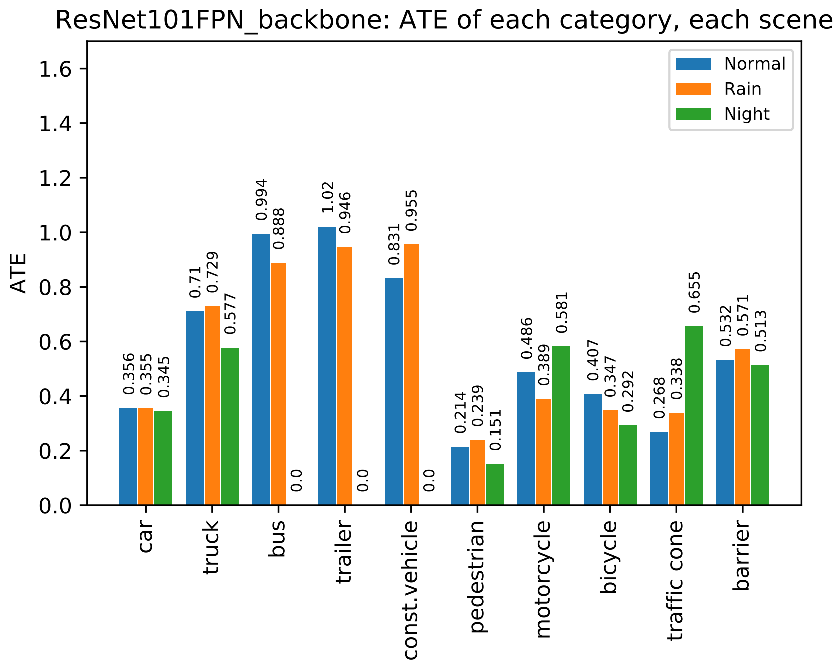


(b) ResNet101 FPN

Figure 4.3: AP for each category, each scene. Top: VGG16 backbone faster R-CNN, Bottom: ResNet101 FPN backbone faster R-CNN.

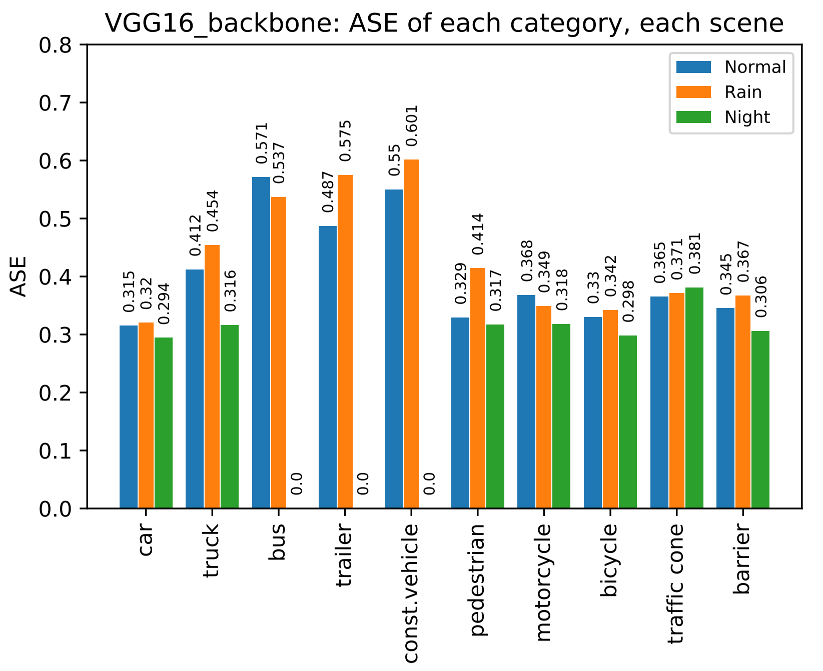


(c) VGG16

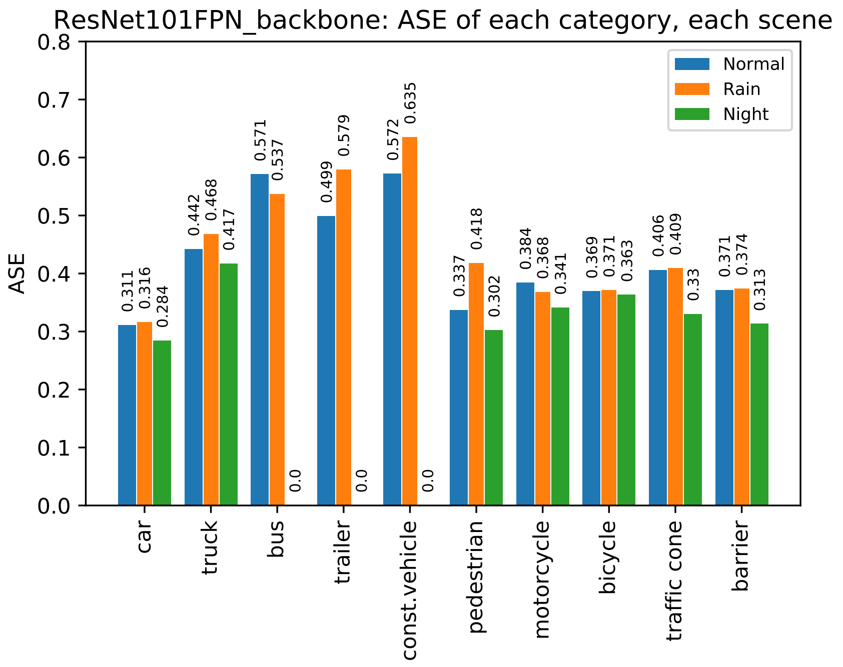


(d) ResNet101 FPN

Figure 4.4: ATE for each category, each scene. Top: VGG16 backbone faster R-CNN, Bottom: ResNet101 FPN backbone faster R-CNN.

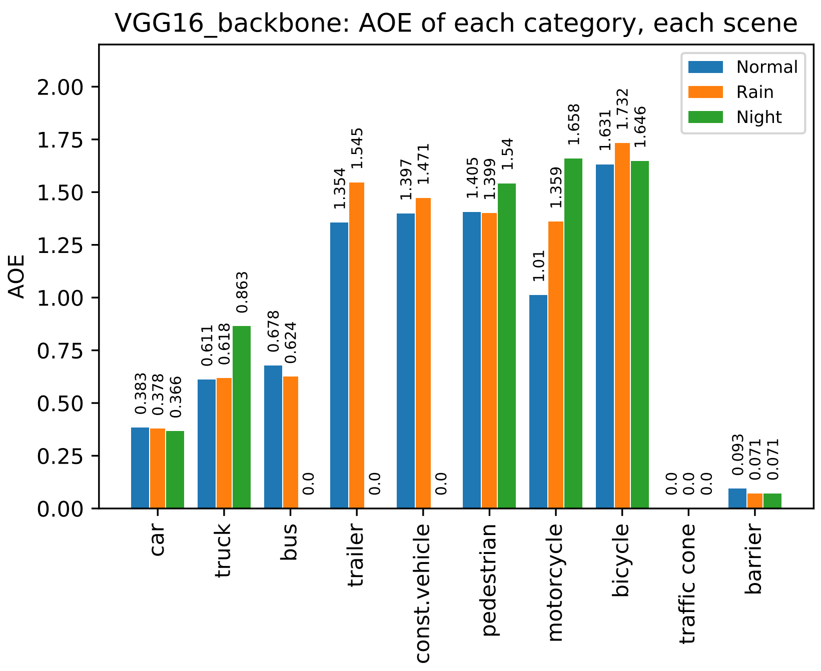


(a) VGG16

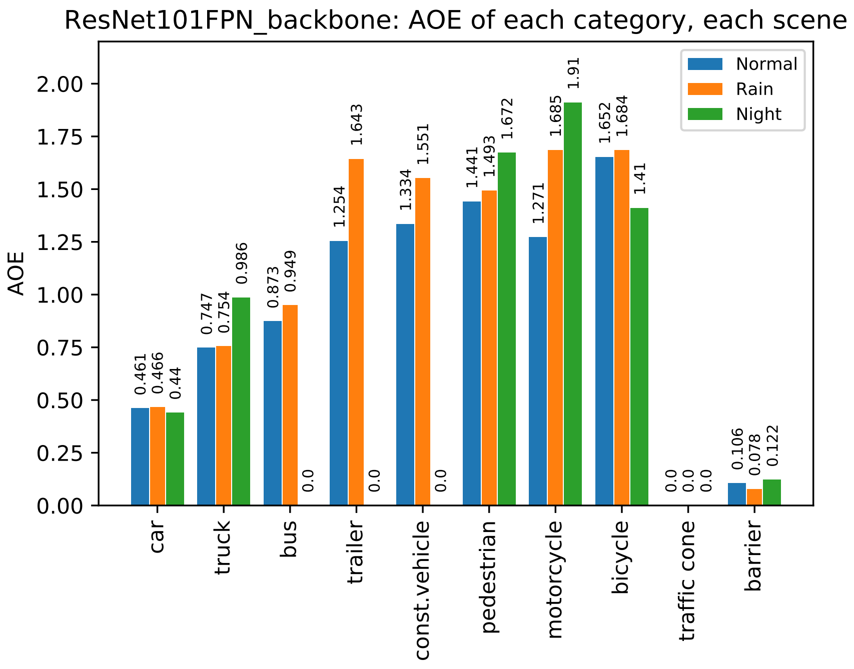


(b) ResNet101 FPN

Figure 4.5: ASE for each category, each scene. Top: VGG16 backbone faster R-CNN, Bottom: ResNet101 FPN backbone faster R-CNN.



(a) VGG16

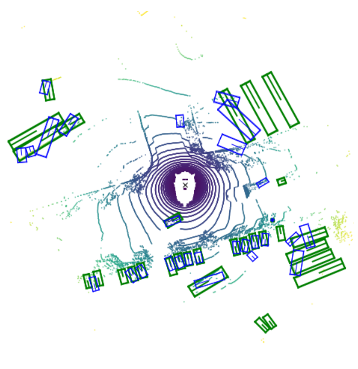


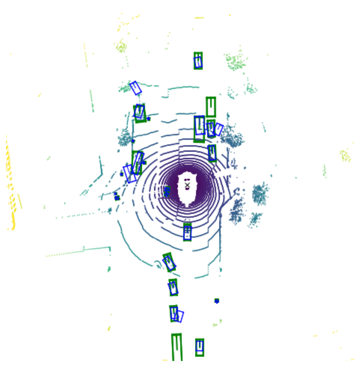
(b) ResNet101 FPN

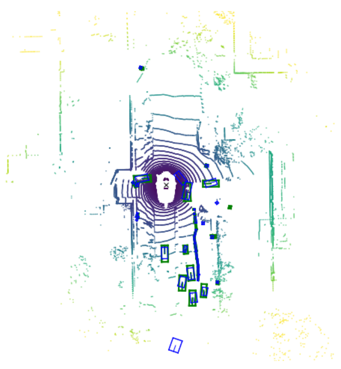
Figure 4.6: AOE for each category, each scene. Top: VGG16 backbone faster R-CNN, Bottom: ResNet101 FPN backbone faster R-CNN.

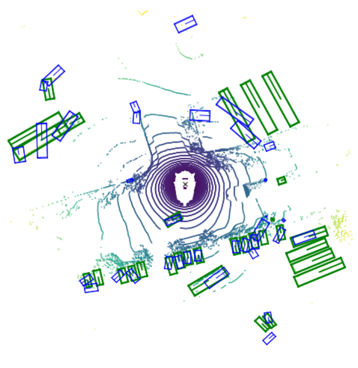
4.3.2 Qualitative results

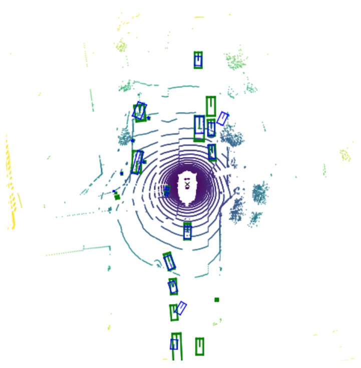
The BEV detection illustrations of 3D detector in normal scene, rain scene and night scene are presented in Figure 4.7, 4.8 and 4.9 respectively. For each scene, three samples of LIDAR pointcloud are plotted, and the 2D detection result from the benchmark, VGG16 backbone faster R-CNN and ResNet101 FPN backbone faster R-CNN are applied as the input of 3D detector.

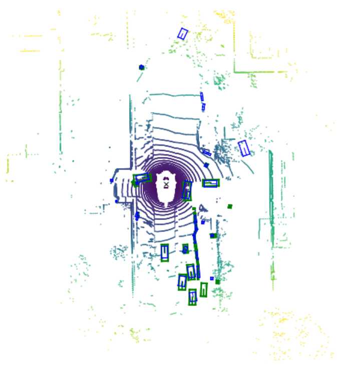

(a) Benchmark

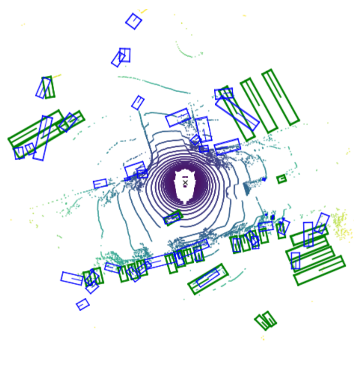

(b) Benchmark

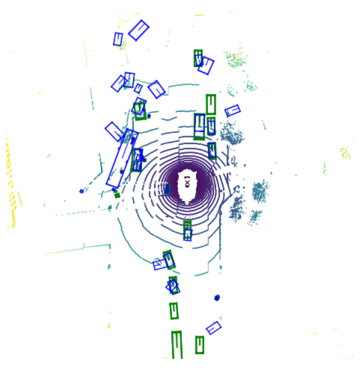

(c) Benchmark


(d) VGG16


(e) VGG16


(f) VGG16


(g) ResNet101 FPN


(h) ResNet101 FPN

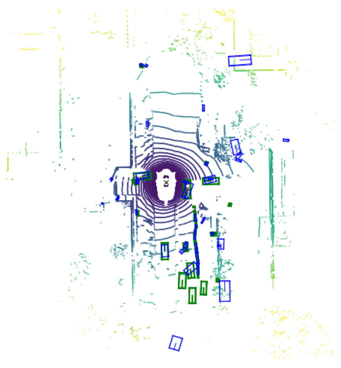
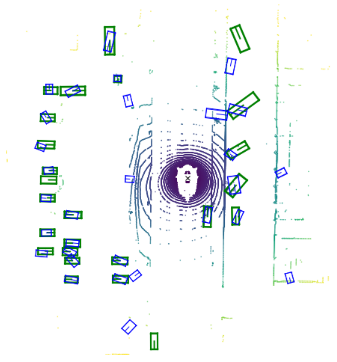
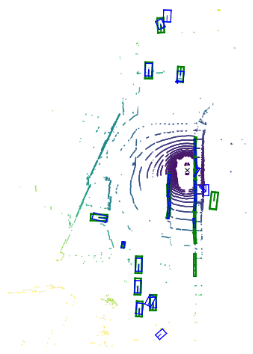
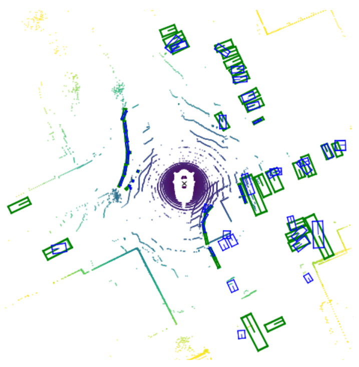
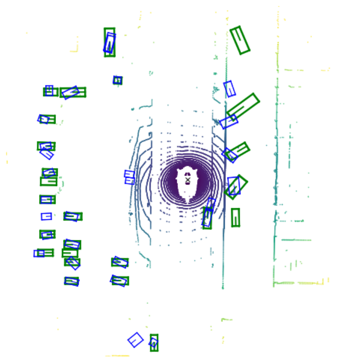

(i) ResNet101 FPN

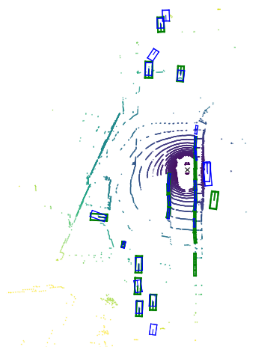
Figure 4.7: The qualitative results of normal scene. Row 1: Benchmark, Row 2: VGG16 backbone faster R-CNN, Row 3: ResNet101 FPN backbone faster R-CNN.

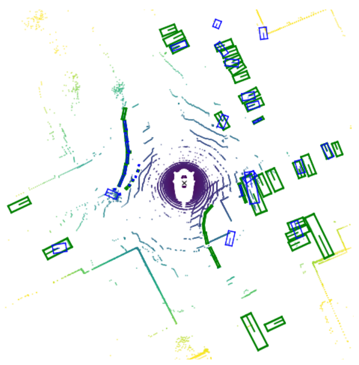

(a) Benchmark

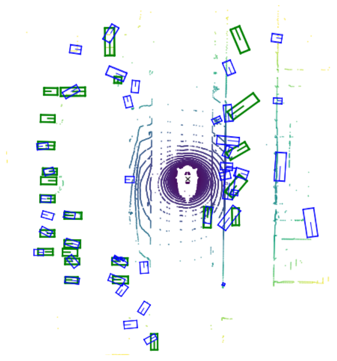

(b) Benchmark

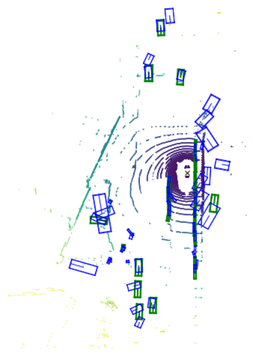

(c) Benchmark


(d) VGG16


(e) VGG16


(f) VGG16


(g) ResNet101 FPN


(h) ResNet101 FPN

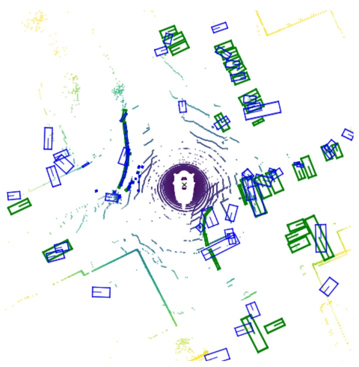
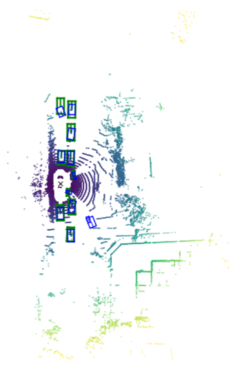
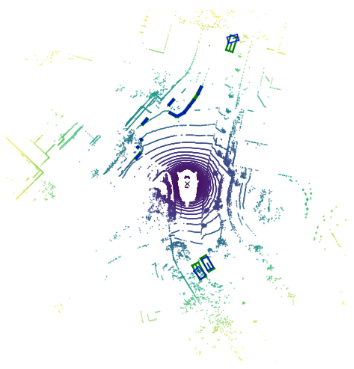
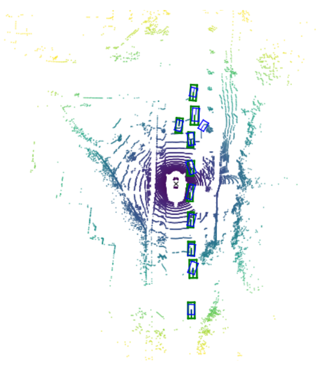
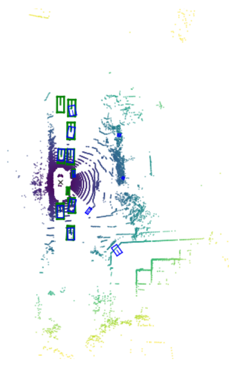

(i) ResNet101 FPN

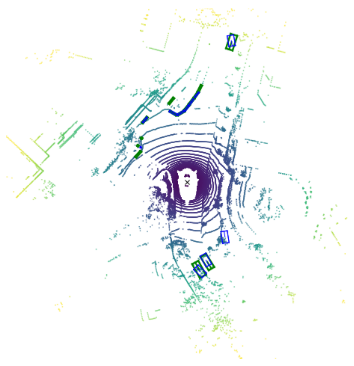
Figure 4.8: The qualitative results of rain scene. Row 1: Benchmark, Row 2: VGG16 backbone faster R-CNN, Row 3: ResNet101 FPN backbone faster R-CNN.

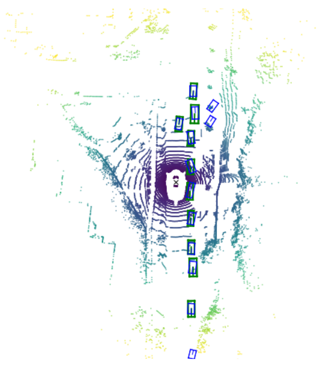

(a) Benchmark

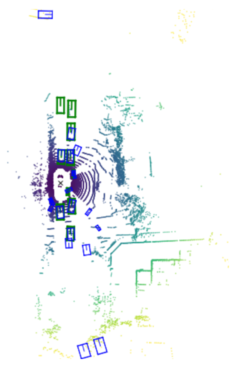

(b) Benchmark

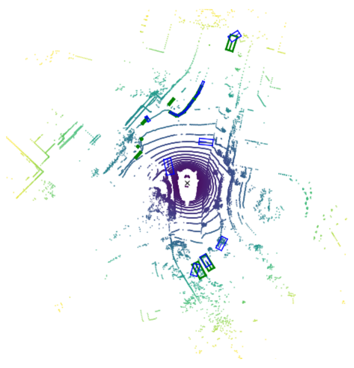

(c) Benchmark


(d) VGG16


(e) VGG16


(f) VGG16


(g) ResNet101 FPN


(h) ResNet101 FPN

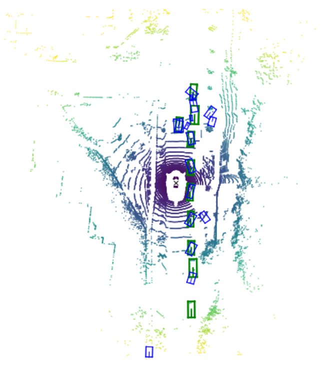

(i) ResNet101 FPN

Figure 4.9: The qualitative results of night scene. Row 1: Benchmark, Row 2: VGG16 backbone faster R-CNN, Row 3: ResNet101 FPN backbone faster R-CNN

5 Discussion

In this chapter a detailed analysis and discussion of the results from the last chapter will be performed.

5.1 The performance of 2D detector

As shown in Chapter 4, VGG16 backbone faster R-CNN outperforms ResNet101 FPN backbone faster R-CNN by a large margin and is selected as the better 2D detector in our experiment on NuScenes dataset. However, on PASCAL VOC [56] dataset and COCO [57] dataset on which VGG16 backbone faster R-CNN and ResNet101 FPN backbone faster R-CNN were originally developed, ResNet101 FPN backbone performs better than VGG16 backbone. We here argue that since VGG16 is much simpler and more straightforward than ResNet101 and the number of convolutional layers in VGG16 is much smaller than ResNet101, it needs much more hand-crafted hyperparameter tuning for ResNet101 FPN backbone faster R-CNN to make it perform well. Due to the lack of time and resources, we didn't manage to do a greedy search to fine tune ResNet101 FPN backbone faster R-CNN and reveal its maximum capability on NuScenes dataset, which probably caused its unsatisfying performance. On the other hand, this comparison result reveals the robustness of VGG16 backbone faster R-CNN, considering that its good performance without much tuning on NuScenes dataset which has very different content from PASCAL VOC or COCO.

Furthermore, the pre-trained initial parameters of VGG16 are taken from caffe library, while the pre-trained initial parameters of ResNet101 are taken from torchvision library. It is noticed that at the beginning of the training process, the loss value imbalance issue is significantly larger for ResNet101 FPN backbone faster R-CNN, which in other words, the results from four loss functions are on different magnitudes. The regression loss for RPN is quite small and the regression loss for ROI head is even smaller. The small value of ROI regression loss is probably caused by the usage of ROI align since it aims to eliminate localization error. We therefore have to manually add weights to different losses to fill the magnitude gap among them so that the backward propagation is not dominated by one type of loss. This proves that the pre-trained VGG16 from caffe library provides a better initialization which makes the following training easier.

On the other hand, we used the mean and standard deviation (std) value from ImageNet to normalize all images in NuScenes as pre-processing. This pre-processing aims to eliminate extreme differences among images and cooperate with the feature extractor whose initialization is taken from ImageNet pre-trained model. As mentioned above, the VGG16 and ResNet101 pre-trained models are taken from different libraries so the pre-processing of two faster R-CNN models are also different. We argue that the pre-processing method is also one of the possible reasons that affect the final performance. Considering that all images of NuScenes dataset are taken by the same one suite of cameras, which is different from the condition in ImageNet, a new pre-processing method, for example using the mean and std value calculated from NuScenes instead of ImageNet, may also help to improve the detection performance.

As observed in Table 4.2 to 4.6, car has the highest AP under almost every condition because it is the category with an exclusively high sample number. Thanks to the huge amount of data of category car, the training process managed to learn its feature better than other categories. On the other hand, the exclusively high number of cars causes imbalance issue. To mitigate this issue and let other categories with much fewer samples have similar good results, one way is to use category balancing pre-processing technique to reduce the imbalance severity, such as in [37]. Interestingly, with a relatively small number of samples, bus achieves the second best place in detection performance and even reaches a higher AP than car in visibility code “3”. One possible reason is that bus shares some features that are similar to car, which causes that the learning of car also helps the the learning of bus. Furthermore, since trailer has similar size and more samples compared with bus but its AP is significantly lower, we consider that the good detection performance on bus is probably because of its regular shape as a closed cubic. On the other hand, trailer and construction vehicle have a medium amount of samples but they result as the two categories with lowest AP. Because they both have open irregular structure shape.

The mAP in rain scene is really close to the mAP in normal scene. The AP per category further proves that camera is not affected by rainy weather, at least in NuScenes dataset. From the qualitative results we can have an intuition that the definition of rain scene in NuScenes does not include a storm when the sky is completely black. The rain does not cause a catastrophic affect on the illumination condition, so the camera can still operate as well as in normal weather. In contrast, the illumination condition in night scene get significantly worse. Therefore the camera suffers a huge detection performance decrease. We implemented low-illumination image enhancement algorithm [58] for the inference time of our 2D detector to make night scene images subjectively look more like taken in normal scenes, but this was found not helpful for improving the performance since current low-illumination image enhancement algorithms are still quite limited and far from good enough. Another naive probable way to improve the performance decay at night is to project LIDAR point-cloud to the camera plane when it is night scene, since LIDAR is not affected by night condition and can serve as a supplement to the original unclear images taken at night. Interestingly, the detection performance for car is not impacted by night scene. One reason is that all cars are supposed to turn on headlights when driving at night, which turns them into light sources. Meanwhile, the location and brightness of the headlights are relatively standard, which gives different cars regular features. It may be noticed that truck should have similar conditions with car since it is also another standard type of vehicle. But the detection performance of truck at night is not quite satisfying. We believe that it is because there are approximately 15 times more samples of car in night scene than those of truck. A significant larger amount of data helps the detection network learn the features of car much better than truck.

It is observed that visibility loss which is mainly caused by occlusion has a fatal impact on the 2D object detection performance. There is another visibility loss cause named truncation, but nowadays since the AD perception sensor suite covers 360 degrees, stitching the images from every camera together will alleviate truncation problem. For occlusion issue, considering that two or more objects have to be really close to each other to cause occlusion, we were intuitively expecting the more aggressive prediction behaviour of ResNet101 FPN backbone faster R-CNN would help on this occlusion issue, which turns out not the case. To be more specific, we were expecting the denser prediction bounding boxes of ResNet101 FPN backbone faster R-CNN may hopefully cover the objects that are highly occluded by another object in the front and result in a better performance in low visibility code. However, the results show that the performance of ResNet101 FPN backbone faster R-CNN is inferior to VGG16 backbone faster R-CNN in all directions and it does not have interesting advantage for alleviating occlusion. One explanation could be that although an occluded object is enclosed by a bounding box, the corresponding classification is not right, which makes the bounding box meaningless. Another possible way to improve the performance under occlusion condition is to use video sequence as input instead of conducting frame-wise object detection, since in ADAS and AD context most occlusion is temporary because the ego vehicle continues moving. However, video sequence as input has gone beyond the scope of the capability of faster R-CNN and other 2D detector implementation is needed to verify this assumption.

5.2 The benchmark performance of 3D detector

5.2.1 AP, ATE, ASE, AOE in all scenes

Figure 5.1 illustrates the scatter plot of AP with the ATE, ASE and AOE. We also calculated the Pearson correlation coefficient (PCC) between AP and each of ATE, ASE and AOE respectively as the following equation

$$\eta_{\text{AP, AxE}} = \frac{cov(\text{AP, AxE})}{\sqrt{var(\text{AP}) \cdot var(\text{AxE})}}, \tag{5.1}$$

where AxE represents one of ATE, ASE and AOE. The numerical results are $\eta_{\text{AP, ATE}} = -0.9421$, $\eta_{\text{AP, ASE}} = -0.9074$, $\eta_{\text{AP, AOE}} = -0.4943$. (AP, ATE) and (AP, ASE) of benchmark 3D detection are the pairs with strong linear correlation, while (AP, AOE) does not show very strong correlation.

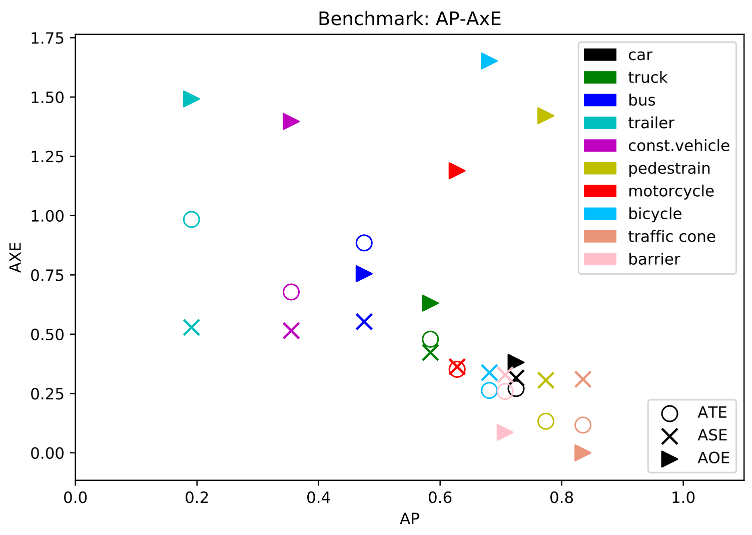


Figure 5.1: Scatter plot of the NuScenes evaluation metrics of benchmark 3D detection results. In the plot, x-axis denotes AP of each category, while the y-axis denote the error metrics, i.e., ATE, ASE and AOE.

We also plotted the the proportion of each valid category among all the valid categories in Figure 5.2. It is found that the proportion of one category is not a key factor to determine the AP of that category. Instead, AP and volume are highly correlated with each other, with $\eta_{\text{AP, volume}} = -0.81$, as Figure 5.3 illustrated. Thus, we can conclude that the category with smaller average volume is likely to have a higher AP. This finding can be explained intuitively. The more accurate the 2D detection results a more accurate 3D detection. Recall that the 2D bounding box we used to generate the frustum view is the intersection of 2D projection and camera canvas. The 2D projection of the large objects that are close to the ego vehicle is less likely to be fully included in the camera canvas, which results the inaccurate frustum, while the smaller object are more likely to be fully included in the camera image. We infer that the camera with larger view scope can provide more accurate frustum view for the 3D detector.

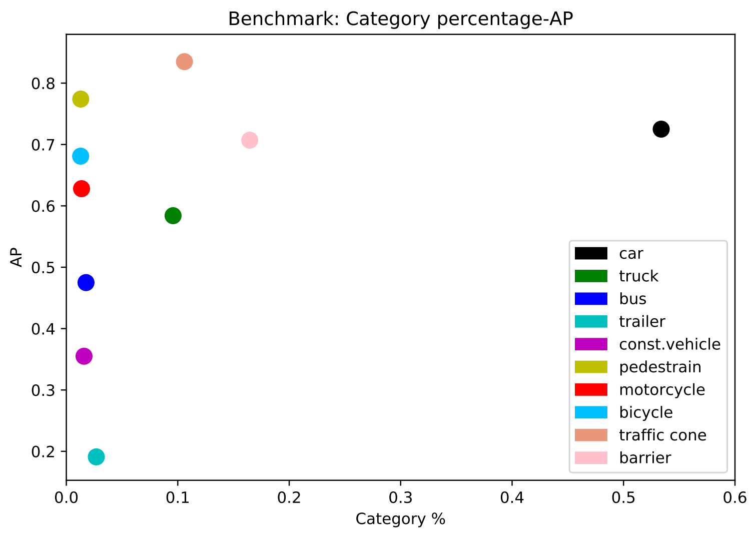


Figure 5.2: Scatter plot of the category percentage and each category's AP of benchmark 3D detection results. The AP does not show a clear relation to the category percentage.

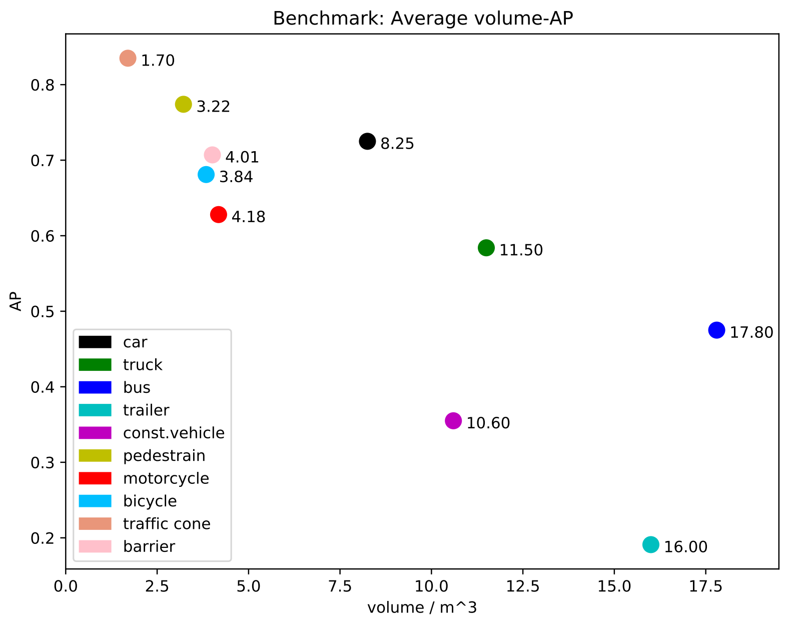


Figure 5.3: Scatter plot of the category average volume and each category's AP of benchmark 3D detection results. The category with larger size is likely to have a lower AP.

5.2.2 AP in different scenes

To our surprise, from Table 4.9, the mAP of the rain scene and night scene outperform that of normal scene. However, we do not know where the difference comes from. The possible reasons for the difference could be

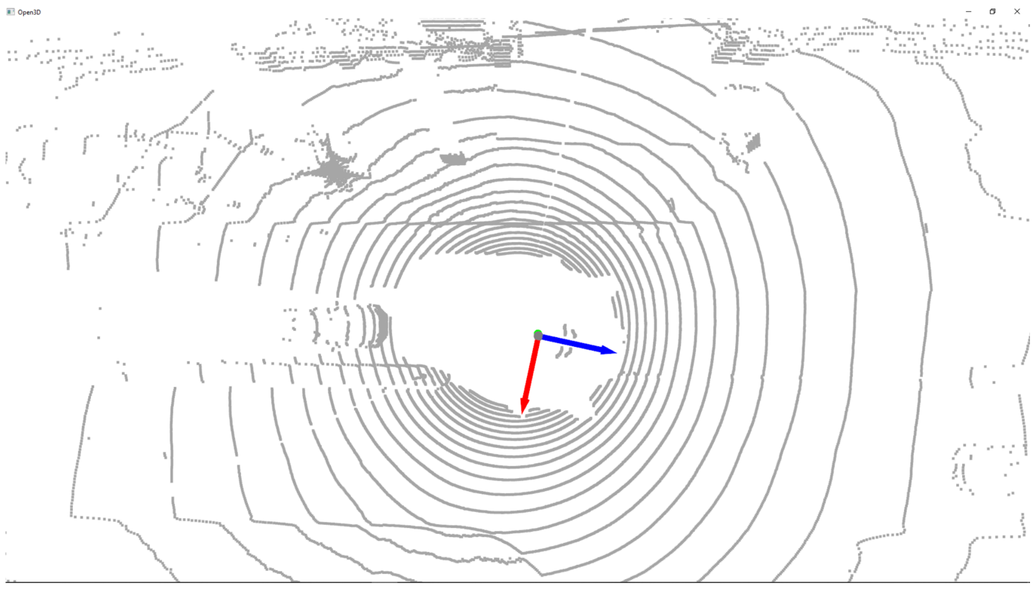
1. biased samples in the night and rain scene,

2. the LIDAR pointcloud data does show systematic difference in night and rain scene compared to normal scene.

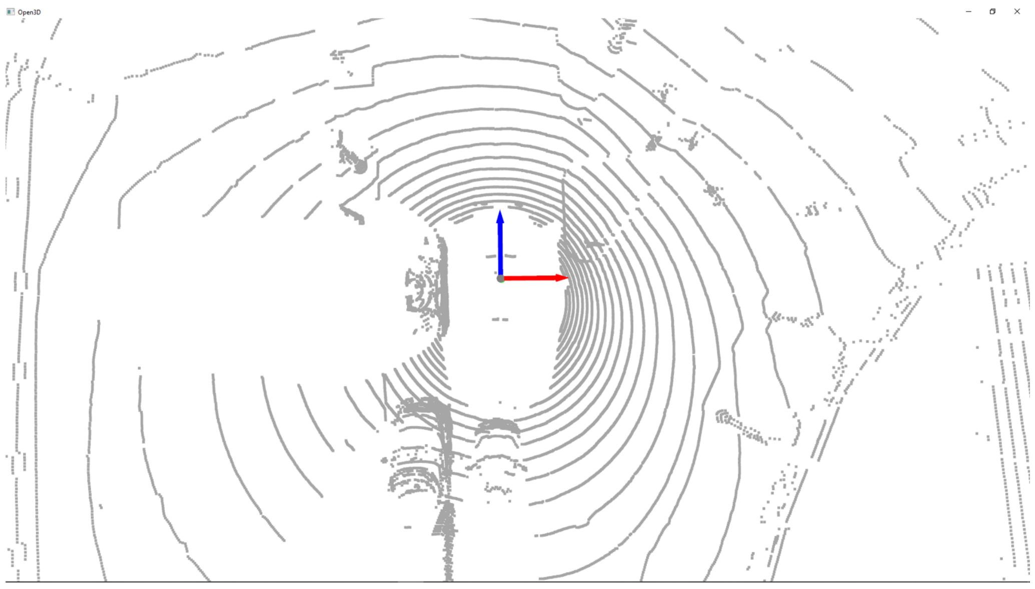
At first, we thought about the mAP difference between normal scene and night scene. Since the only difference between night scene and normal scene is the illumination level and LIDAR unit does not require ambient light to operate, we believe that there is no difference for LIDAR to operate in normal or night scene. Thus, the second assumption is excluded. Given the number of night scenes (14) is much smaller than the normal scenes (112), the first assumption could explain the mAP difference.

Then we thought about the mAP difference between normal scene and rainy scene. The first assumption could be one of the reasons, since there are only 24 scenes in the rain scene, which are fewer than the number of normal scenes. Now we examine the second assumption. Four pictures of LIDAR detected road surface are plotted in Figure 5.4. It is easy to notice that the LIDAR scan of road surface in the normal scene looks like a smooth curve, but in the rain scene the points of road surface are less smooth. This difference is an evidence to support assumption two, that the difference of LIDAR pointcloud data between normal and rain scene does exist and can be perceived by human eyes. We can infer that the difference also exists not only exist in the road surface, but also in the some of the other objects.

Another question is, does this difference in LIDAR points influence the detection ability of 3D detector? This is a question we cannot answer yet. According the mAP of normal scene 0.7020 and of rain scene 0.7334, it is risky to conclude that the rainy scene helps the 3D detector to achieve higher mAP, since the samples in rain scenes can be biased towards the easier side for the 3D detector. To answer this question rigorously, we might need two frames of LIDAR pointcloud with identical set of objects and surroundings except the difference in rain or not rain, and see the mAP of the 3D detector these two frames. However, such LIDAR frames this are unrealistic to get.

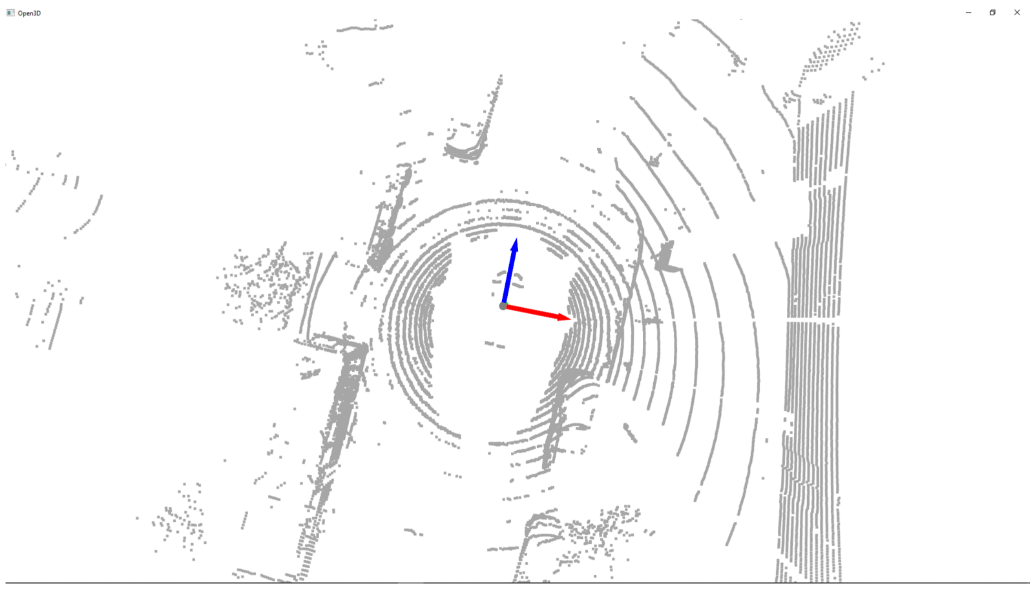


(a) normal scene

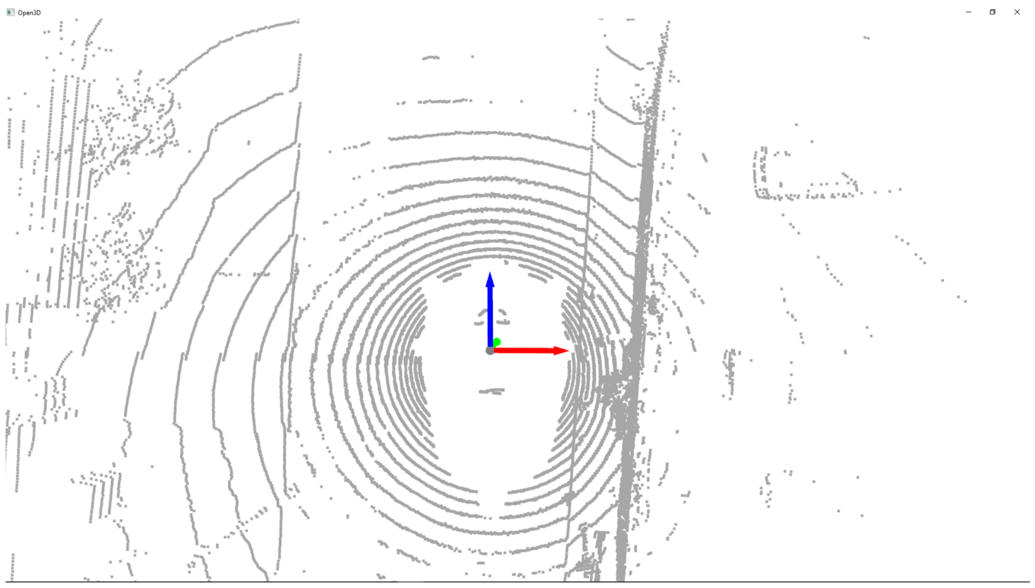


(b) normal scene

Figure 5.4: LIDAR points of road surface in normal and rain scene.



(c) rain scene



(d) rain scene

Figure 5.4: LIDAR points of road surface in normal and rain scene.

5.3 Joint 2D-3D detection performance

In this section, we aims to discuss how the AP of joint 2D-3D detection is influenced by the 2D detection. Note that we stated in Chapter 3.4, all the 2D detection from VGG16 backbone faster R-CNN and ResNet101 FPN backbone faster R-CNN that serve as the input for F-ConvNet are selected by a threshold at 0.7. However, the evaluation for the 2D detector itself in Chapter 4.1 takes all 2D bounding boxes with a score higher than 0.05 into account. Thus, here we evaluate again the performance of 2D detection with the score threshold 0.7 in order to give better explanations for the following 3D detection result. The results are shown in Table 5.1.

Table 5.1: AP and mAP of two faster R-CNN with different backbones under different scenes, evaluation score threshold 0.7. Since the bus, trailer and construction vehicle do not exist in the night scene, all the mAPs in the last row do not count in these three categories, for the convenience of comparison.

	VGG16 backbone				**ResNet101FPN backbone**			
	all	normal	rain	night	all	normal	rain	night
car	0.601	0.599	0.607	0.608	0.494	0.494	0.497	0.494
truck	0.398	0.399	0.334	0.270	0.274	0.302	0.250	0.184
bus*	0.532	0.534	0.523	nan	0.425	0.436	0.373	nan
trailer*	0.227	0.225	0.288	nan	0.131	0.127	0.146	nan
const.veh*	0.152	0.192	0.146	nan	0.107	0.109	0.117	nan
pedestrian	0.417	0.420	0.322	0.339	0.303	0.306	0.239	0.250
motorcycle	0.346	0.351	0.411	0.313	0.219	0.219	0.194	0.073
bicycle	0.343	0.343	0.342	0.253	0.237	0.244	0.175	0.145
traffic cone	0.477	0.480	0.405	0.062	0.294	0.298	0.276	0.008
barrier	0.393	0.383	0.429	0.254	0.292	0.287	0.297	0.188
mAP (without *)	0.425	0.425	0.407	0.300	0.302	0.307	0.276	0.192

5.3.1 AP, ATE, ASE, AOE in all scenes

From Table 4.7 and 4.10, it is easy to see that the 2D detection from faster R-CNN yields higher AP of 3D detection and lower ATE by a large margin. While the ASE and AOE does not show such relation as AP or ATE.

We also compared how much the AP of each category drops from benchmark 3D AP in the 2D-3D joint detection results. Let "AP decay" denote how much AP of the joint detection has dropped from the AP of the benchmark detection, i.e.,

$$\text{AP decay} = 1 - \frac{\text{AP}_{\text{joint}}}{\text{AP}_{\text{benchmark}}}. \tag{5.2}$$

A scatter plot of the AP decay and the AP of 2D detection from Table 5.1 is illustrated as Figure 5.5. The PCC is -0.958 for VGG16 backbone faster R-CNN and -0.849 for ResNet101 FPN backbone faster R-CNN. Thus, we can conclude that the detection category with higher AP in 2D detection are likely to have less drop of AP in the 3D detection from the benchmark results.

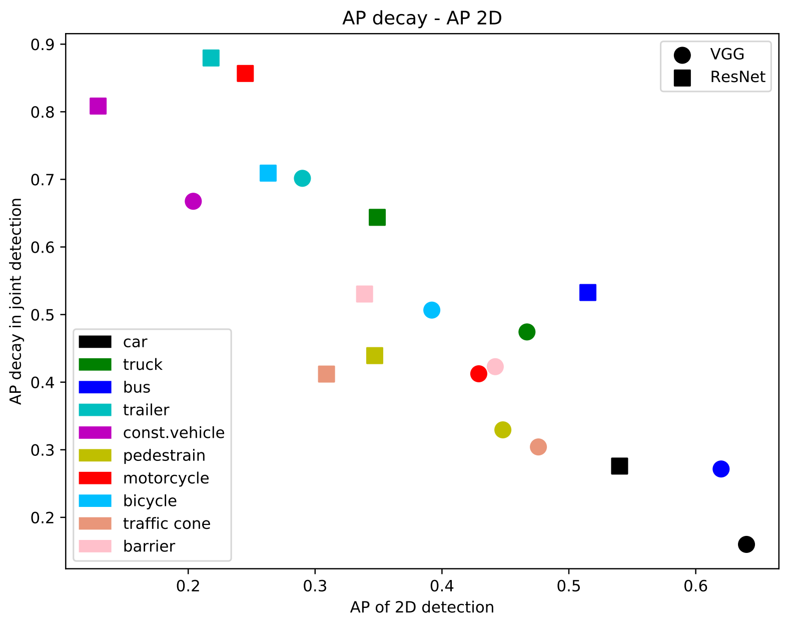


Figure 5.5: Scatter plot of AP decay in joint detection and 2D AP for each category. The AP decay describes how much AP has dropped in joint detection from the benchmark 3D detection. AP decay is highly negatively correlated with the 2D AP for both 2D detectors.

5.3.2 AP in different scenes

From Table 4.12, we find that 2D mAP in night is significantly lower than that in normal and rain scene, for both 2D detectors. We also calculate the AP decay under either 2D detector, which is displayed in Table 5.2. The conclusion that the AP decay of each category is highly negatively correlated with the AP of 2D detection also holds for different scenes. The strong dependence on the 2D detector could be a risk of this joint 2D-3D detection algorithm, especially in the night scene, where the bicycle, pedestrian are not easy to be detected.

Table 5.2: Pearson coefficient of AP decay and 2D AP (score threshold 0.7) of each scene. * means only 7 categories are used to calculated the coefficient.

faster R-CNN backbone	VGG16	ResNet101 FPN
all	-0.958494	-0.849487
normal*	-0.904145	-0.82928
rain*	-0.944948	-0.888039
night*	-0.894487	-0.971984

6
Conclusion & Future Work

6.1 Conclusion

In this thesis, we explored NuScenes dataset, which is a newly released AD dataset and conducted a thorough analysis. We did an extensive literature review on the deep learning method for 2D and 3D object detection. We implemented and trained the joint 2D-3D object detector, which uses the 2D detection as the prior information for 3D detector, for our NuScenes object detection task. We also conducted an analysis and discussion of the characteristics of our object detector.

Two versions of faster R-CNN as 2D detector with different backbones were implemented for the 2D detection. According to our experiments, the VGG16 backbone faster R-CNN outperforms the ResNet101 FPN backbone faster R-CNN by a large margin. We also found that the occlusion has an unfavourable impact on the detection ability.

The 3D detector we implemented is Frustum ConvNet. The 2D detection's category and bounding box is the prior information for the Frustum ConvNet. To be able to train up to 10 categories on this architecture, we modified the detection head from the original Frustum ConvNet.

Frustum ConvNet has different inherent detection ability on different categories. The Frustum ConvNet we implemented did best at traffic cone and the worst on trailer, if we use AP as the indicator for detection ability. We found that the AP of each category is correlated with the mean size of this category. The smaller category is more likely to achieve higher AP.

The performance of 2D-3D detector on some category is determined by the 3D detector's inherent detection ability on this category together with the 2D detector's ability on this category. But generally speaking, the category with higher AP in 2D detection has less AP drop percentage in AP of joint 2D-3D detection. A successful 2D detector is the necessary condition of a successful joint 2D-3D detector.

We also tested the performance of 2D and 3D detector on different scenes. For either 2D detector, the detection ability is much weaker in the night scene, except the car category. The ability for normal scene and raining scene is very similar. We also noticed that Frustum ConvNet benchmark results in rainy and night scenes are similar and even slightly better than the normal scene. We can conclude that LIDAR performs equally well among normal, rain and night scene in the context of using our 3D detector and the NuScenes dataset.

The joint 2D-3D detection performance in night scene is reduced because of the limitation of camera. Thus, careful attention should be paid to reliability of camera sensor in the low light environment when the ADAS algorithm is developed for commercial purpose.

6.2 Future Work

Due to the time and resource limitation, the 3D detector is only trained two epochs. It is expected that more epochs will increase the performance of 3D detector.

For the same limitation, we did not evaluate the joint 2D-3D detection performance using 2D predictions with score over 0.05. Since the 2D detector evaluation using all predicted bounding boxes with a score higher than 0.05 gives a better result than using 0.7, and NMS will be applied after F-ConvNet to eliminate the 3D bounding boxes overlapping issue, we believe that feeding more 2D predictions as prior information to F-ConvNet can improve the performance. However, more 2D prior information makes the processing time significantly longer, therefore how to make the computation more efficient and decrease the inference time is also another future work direction.

In our thesis, we only have one model of 3D detection using PointNet as feature extractor. More variations of the feature extractor could be explored in the future so that we can see how the feature extractor will influence the performance of 3D detector.

NuScenes is very imbalanced in the valid categories. Some work such as [37] uses pre-processing technique on the dataset to reduce the imbalance issue. We can also try such technique to further improve the performance.

Finally, since NuScenes also provides Radar data, the pipeline of our work enables the LIDAR data that we use be transferred to Radar data, and the pioneer work [59] shows the potential of using PointNet on radar data to conduct object detection, we believe our work will have some interesting results once transferred on radar data.

Bibliography

[1] Nakhaeinia D, Tang SH, Noor SM, Motlagh O. A review of control architectures for autonomous navigation of mobile robots. International Journal of the Physical Sciences. 2011;6(2):169–174.

[2] Zhu H, Yuen KV, Mihaylova L, Leung H. Overview of environment perception for intelligent vehicles. IEEE Transactions on Intelligent Transportation Systems. 2017;18(10):2584–2601.

[3] González D, Pérez J, Milanés V, Nashashibi F. A review of motion planning techniques for automated vehicles. IEEE Transactions on Intelligent Transportation Systems. 2015;17(4):1135–1145.

[4] Wang Z, Jia K. Frustum convnet: Sliding frustums to aggregate local pointwise features for amodal 3d object detection. arXiv preprint arXiv:190301864. 2019;.

[5] Schroder J, Heid B, Neuhaus F, Kasser M, Klink C, Tatomir S. Fast forwarding last-mile delivery–implications for the ecosystem. Travel, Transport and Logistics and Advanced Industries, McKinsey & Company, July. 2018;.

[6] Rosenzweig J, Bartl M. A review and analysis of literature on autonomous driving. E-Journal Making-of Innovation. 2015;p. 1–57.

[7] Yurtsever E, Lambert J, Carballo A, Takeda K. A survey of autonomous driving: Common practices and emerging technologies. IEEE Access. 2020;8:58443–58469.

[8] Ilas C. Electronic sensing technologies for autonomous ground vehicles: A review. In: 2013 8TH INTERNATIONAL SYMPOSIUM ON ADVANCED TOPICS IN ELECTRICAL ENGINEERING (ATEE). IEEE; 2013. p. 1–6.

[9] Caesar H, Bankiti V, Lang AH, Vora S, Liong VE, Xu Q, et al. nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:190311027. 2019;.

[10] Lowe DG. Distinctive image features from scale-invariant keypoints. International journal of computer vision. 2004;60(2):91–110.

[11] Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). vol. 1. IEEE; 2005. p. 886–893.

[12] Lienhart R, Maydt J. An extended set of haar-like features for rapid object detection. In: Proceedings. international conference on image processing. vol. 1. IEEE; 2002. p. I–I.

[13] Barnich O, Van Droogenbroeck M. ViBe: a powerful random technique to estimate the background in video sequences. In: 2009 IEEE international conference on acoustics, speech and signal processing. IEEE; 2009. p. 945–948.

[14] Beaupré DA, Bilodeau GA, Saunier N. Improving multiple object tracking with optical flow and edge preprocessing. arXiv preprint arXiv:180109646. 2018;.

[15] Guo Y, Liu Y, Oerlemans A, Lao S, Wu S, Lew MS. Deep learning for visual understanding: A review. Neurocomputing. 2016;187:27–48.

[16] Kesten R, Usman M, Houston J, Pandya T, Nadhamuni K, Ferreira A, et al.. Lyft level 5 av dataset 2019; 2019.

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

[18] Huang X, Cheng X, Geng Q, Cao B, Zhou D, Wang P, et al. The apolloscape dataset for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops; 2018. p. 954–960.

[19] Sun P, Kretzschmar H, Dotiwalla X, Chouard A, Patnaik V, Tsui P, et al. Scalability in perception for autonomous driving: Waymo open dataset. arXiv. 2019;p. arXiv–1912.

[20] Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2014. p. 580–587.

[21] Girshick R. Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision; 2015. p. 1440–1448.

[22] Ren S, He K, Girshick R, Sun J. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems; 2015. p. 91–99.

[23] Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S. Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 2117–2125.

[24] He K, Gkioxari G, Dollár P, Girshick R. Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision; 2017. p. 2961–2969.

[25] Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 779–788.

[26] Redmon J, Farhadi A. YOLO9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 7263–7271.

[27] Redmon J, Farhadi A. Yolov3: An incremental improvement. arXiv preprint arXiv:180402767. 2018;.

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

[29] Chen X, Kundu K, Zhang Z, Ma H, Fidler S, Urtasun R. Monocular 3d object detection for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2016. p. 2147–2156.

[30] Ku J, Mozifian M, Lee J, Harakeh A, Waslander SL. Joint 3d proposal generation and object detection from view aggregation. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE; 2018. p. 1–8.

[31] Qi CR, Liu W, Wu C, Su H, Guibas LJ. Frustum pointnets for 3d object detection from rgb-d data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2018. p. 918–927.

[32] Zhou Y, Tuzel O. Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2018. p. 4490–4499.

[33] Shi S, Guo C, Jiang L, Wang Z, Shi J, Wang X, et al. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. arXiv preprint arXiv:191213192. 2019;.

[34] Meyer GP, Laddha A, Kee E, Vallespi-Gonzalez C, Wellington CK. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2019. p. 12677–12686.

[35] Meyer GP, Charland J, Hegde D, Laddha A, Vallespi-Gonzalez C. Sensor fusion for joint 3D object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops; 2019. p. 0–0.

[36] Lang AH, Vora S, Caesar H, Zhou L, Yang J, Beijbom O. Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2019. p. 12697–12705.

[37] Zhu B, Jiang Z, Zhou X, Li Z, Yu G. Class-balanced Grouping and Sampling for Point Cloud 3D Object Detection. arXiv preprint arXiv:190809492. 2019;.

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

[39] Zarzar J, Giancola S, Ghanem B. PointRGCN: Graph Convolution Networks for 3D Vehicles Detection Refinement. arXiv preprint arXiv:191112236. 2019;.

[40] Qi CR, Litany O, He K, Guibas LJ. Deep hough voting for 3d object detection in point clouds. In: Proceedings of the IEEE International Conference on Computer Vision; 2019. p. 9277–9286.

[41] Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM. Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG). 2019;38(5):1–12.

[42] Shi S, Wang Z, Shi J, Wang X, Li H. From Points to Parts: 3D Object Detection from Point Cloud with Part-aware and Part-aggregation Network. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2020;.

[43] Kraemer S, Stiller C, Bouzouraa ME. LiDAR-Based Object Tracking and Shape Estimation Using Polylines and Free-Space Information. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE; 2018. p. 4515–4522.

[44] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:14091556. 2014;.

[45] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 770–778.

[46] Chen Y. Learning Faster R-CNN from the perspective of programming implementation (with minimalist implementation);. `https://zhuanlan.zhihu.com/p/32404424` Accessed March 10, 2020. [EB/OL].

[47] Qi CR, Su H, Mo K, Guibas LJ. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 652–660.

[48] Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, et al. Ssd: Single shot multibox detector. In: European conference on computer vision. Springer; 2016. p. 21–37.

[49] Qi CR, Yi L, Su H, Guibas LJ. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems; 2017. p. 5099–5108.

[50] Lin TY, Goyal P, Girshick R, He K, Dollár P. Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision; 2017. p. 2980–2988.

[51] Tofa KN, Ahmed F, Shakil A, et al. Inappropriate scene detection in a video stream. BRAC University; 2017.

[52] Jin A, Yeung S, Jopling J, Krause J, Azagury D, Milstein A, et al. Tool detection and operative skill assessment in surgical videos using region-based convolutional neural networks. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE; 2018. p. 691–699.

[53] González RE, Munoz RP, Hernández CA. Galaxy detection and identification using deep learning and data augmentation. Astronomy and computing. 2018;25:103–109.

[54] Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. Ieee; 2009. p. 248–255.

[55] Padilla R, Netto SL, da Silva EAB. A Survey on Performance Metrics for Object-Detection Algorithms. In: 2020 International Conference on Systems, Signals and Image Processing (IWSSIP); 2020. p. 237–242.

[56] Everingham M, Van Gool L, Williams CK, Winn J, Zisserman A. The pascal visual object classes (voc) challenge. International journal of computer vision. 2010;88(2):303–338.

[57] Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, et al. Microsoft coco: Common objects in context. In: European conference on computer vision. Springer; 2014. p. 740–755.

[58] Shi Z, mei Zhu M, Guo B, Zhao M, Zhang C. Nighttime low illumination image enhancement with single image using bright/dark channel prior. EURASIP Journal on Image and Video Processing. 2018;2018(1):13.

[59] Danzer A, Griebel T, Bach M, Dietmayer K. 2d car detection in radar data with pointnets. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC). IEEE; 2019. p. 61–66.