MLOD: A multi-view 3D object detection based on robust feature fusion method

by

Jian Deng

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2019

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis presents Multi-view Labelling Object Detector (MLOD). The detector takes an RGB image and a LIDAR point cloud as input and follows the two-stage object detection framework [8] [32]. A Region Proposal Network (RPN) generates 3D proposals in a Bird's Eye View (BEV) projection of the point cloud. The second stage projects the 3D proposal bounding boxes to the image and BEV feature maps and sends the corresponding map crops to a detection header for classification and bounding-box regression. Unlike other multi-view based methods, the cropped image features are not directly fed to the detection header, but masked by the depth information to filter out parts outside 3D bounding boxes. The fusion of image and BEV features is challenging, as they are derived from different perspectives. We introduce a novel detection header, which provides detection results not just from fusion layer, but also from each sensor channel. Hence the object detector can be trained on data labelled in different views to avoid the degeneration of feature extractors. MLOD achieves state-of-the-art performance on the KITTI 3D object detection benchmark. Most importantly, the evaluation shows that the new header architecture is effective in preventing image feature extractor degeneration.

Acknowledgements

I am deeply indebted and appreciate to my supervisor Professor Krzysztof Czarnecki for his excellent guidance and great encouragement and generous support during my study at the University of Waterloo. I am truly grateful to WISE lab for its support through my master program. I would like to thank many of my friends for their enthusiastic help on my study and living in Ontario. I also want to thank my officemates, Prarthana Bhattacharyya and Venkateshwaran Balasubramanian, for the many happy conversations and laughter we had. Finally, I am greatly indebted to my wife, Menglu Che. Without her constant and selfless love and support, I could not have the opportunity to have and enjoy such joyous and great life in Canada.

Table of Contents

List of Tables vii

List of Figures viii

1 Introduction 1

2 Background 4

2.1 Artificial Neural Networks 4

2.1.1 Stochastic Gradient Descent 5

2.1.2 Dropout 7

2.2 Convolution Neural Networks 8

2.2.1 Selected Convolution Neural Networks Architectures 10

2.3 2D Object Detection 12

2.3.1 Region Based Object Detectors 12

2.3.2 One-Stage Object Detectors 14

2.4 3D Object Detection 16

2.4.1 Image Region Proposal Methods 16

2.4.2 Projection Based Methods 17

2.4.3 Multi-View Based Methods 17

2.5 KITTI 3D Object Detection Evaluation Metrics 18

2.5.1 Intersection over Union 18

2.5.2 Average Precision 18
2.5.3 Average Orientation Similarity 19

3 Multi-view Labelling Object Detector **21**
3.1 The MLOD Architecture 22
3.1.1 BEV Map Preparation 22
3.1.2 Feature Extractor 22
3.1.3 Foreground Mask Layer 23
3.1.4 Multi-view Header 27
3.2 Implementation Details 30
3.2.1 Mini-batch Settings 30
3.2.2 3D Box Encoding 30
3.2.3 Data Augmentation 31

4 Experiments **32**
4.1 KITTI Dataset 32
4.2 Accuracy 33
4.3 Ablation Study 33
4.3.1 Effects of Multi-view Header 33
4.3.2 Effects of Foreground Mask Layer 35
4.4 Qualitative Results 36

5 Conclusion **39**

References **40**

List of Tables

3.1 BEV branch feature extractor layers. 24

3.2 Image branch feature extractor layers. 25

4.1 A comparison of AP_{3D} from MLOD and current state-of-art 3D object detectors on validation set at the moderate difficulty. 32

4.2 A comparison of the performance of MLOD with current state-of-art 3D object detectors 33

4.3 AP_{3D} from MLOD with different λ settings, evaluated on the validation set. Since the image channel lacks depth information, it is difficult to predict the 3D bounding box from it. To facilitate the comparison, results from fusion and image channel use the same 3D bounding boxes. Thus, the shown results reflect only the variation of classification results. 35

4.4 Effects of a foreground mask layer. 35

List of Figures

2.1 An artificial neural network architecture. 5
2.2 Dropout Neural Net Model. Left: An ANN with two hidden layers; Right: The ANN when Dropout is applied. 8
2.3 An example of CNN architecture [48] 9
2.4 Types of Pooling layers. 9
2.5 The architecture of VGG-16 model [46] 11
2.6 The architecture of U-Net model [33] 11
2.7 Recognition problems. (a) Image Object classification, (b) bounding box level object detection [22] 12
2.8 The architecture of fast RCNN model [8] 13
2.9 The architecture of Faster RCNN model [32] 14
2.10 The YOLO model [30] 15
2.11 The architecture of SSD model [23] 15
2.12 The architecture of Frustum PointNet model [27] 16
2.13 The architecture of deep continuous fusion model [20] 17
2.14 The architecture of AVOD model [18] 18

3.1 Architectural diagram of the proposed method 22
3.2 The architecture of the feature pyramid extractor. [18] 23
3.3 Illustration of foreground masking layer procedure: Step 1: calculating the median of nonzero values in each grid; Step 2: obtaining a mask by Equation 3.1 ($d_{min} = 6.8, d_{max} = 9.7$ in this example); Step 3: applying the mask to the feature maps. 26

3.4 A qualitative example of a foreground mask and its application to the original image. The bottom left background and the top left and right background are masked. 26

3.5 The multi-view header architecture diagram 28

3.6 Examples of IoU in different views. The pictures show the projection of 3D bounding boxes (proposals A,B,C in green and the ground truth in red) onto ground plane (BEV) and image. The IoU of proposals B and C is less than 0.3 in BEV, but is larger than 0.7 in image view. Hence proposals B and C are negative in BEV and positive in front view. 29

3.7 A visual comparison between various box encoding method: the 8 corners box encoding method [3], the axis aligned box encoding method [40], and 4 corners encoding method [18] 31

3.8 A visual example of aligning skewed regressed 4 corners. The left figure shows how the reference line is selected along the longest side. The right figure shows how the corners are aligned with respect to the selected line [26]. Note that the figure uses the camera coordinate system convention, where the horizontal plane is represented by the x,z axes. 31

4.1 Examples of the effects of various λ settings. Column A: $\lambda_{sub-cls}/\lambda_{cls} = 0.001$; Column B: $\lambda_{sub-cls}/\lambda_{cls} = 1$. Blue boxes: pedestrians; Yellow boxes: cyclist. 34

4.2 MLOD predictions on sample 000010 36

4.3 MLOD predictions on sample 000357 37

4.4 MLOD predictions on sample 002477 37

4.5 MLOD predictions on sample 003759 38

Chapter 1

Introduction

3D object detection is crucial for a safe and robust mobile robotic system, like an autonomous vehicle. It allows such a system to track and predict the motion of objects by providing classification and localization of physical objects.

Since AlexNet [17] won the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [6], deep learning has changed the computer vision landscape. Object detection, one of the subfields of computer vision, has also experienced tremendous progress in recent years. Object detection is a computer vision task to predict the class and location of objects in images. A large number of object detection algorithms have been proposed based on deep convolutional neural networks (CNNs), which generally fall into one of the two categories [22]:

- Two-stage detection methods include a region proposal network to select the regions of interest for the final detection processes. Thus, the overall pipeline is two-stage. They are also called region-based methods.

- Single-stage detection methods do not rely on region proposal networks. They directly predict the classes and bounding boxes from the input image using a feed-forward CNN.

As it is difficult to obtain sufficient depth information using a mono camera, additional sensors, like stereo cameras or LIDAR, are typically used for 3D perception. LIDAR is a sensor commonly used in autonomous driving to understand 3D structure of the surrounding environment. It measures distance to a target by illuminating the target with

laser light and measuring the reflected light with a sensor. By firing off millions of light beams per second, LIDAR produces a set of 3D points, also referred to as a 3D point cloud, which provides a 3D representation of the environment, including objects such as cars and pedestrians.

Due to the unique characteristics of point cloud data, 2D object detection methods have not transferred well to the detection of 3D objects using LIDAR. LIDAR point cloud data is sparse and heterogeneous (i.e., of different quality) with respect to distance. A LIDAR sensor provides good quality measurement for close objects, but the point cloud of distant object is typically sparse and difficult to interpret. Although CNNs are the state of the art techniques for 2D object detection, they do not perform as well on 3D point cloud as on RGB image data. For example, in the KITTI object detection benchmark [7], a popular 3D object detection dataset in the context of autonomous driving, the state-of-art 2D object detector, RRC [31], reaches an average precision (AP) of 90.23%, 77.86%, and 76.49% for car, pedestrian, and cyclist classes, respectively, on the moderate difficulty level. On the other hand, the current best performing 3D detectors achieve much lower AP for car, pedestrian, and cyclist on moderate level, namely 77.86%[37], 45.61%[47], and 64.68%[47], respectively.

Another challenge of 3D object detection using LIDAR input is the fusion of LIDAR and RGB image information. These two sensors capture different physical attributes of the environment [5]. Moreover the data from camera and LIDAR is derived from different perspectives. Using such multiple sensors, aka input multi-modality, is of high importance in order to provide reliable 3D object detection. Existing 3D object detectors used for autonomous driving fall into one of the three categories based on how they leverage both LIDAR point cloud and RGB image data.

- Image region proposal approaches, like [27] and [50], use image data to generate proposals for point clouds. The final classification and regression of bounding boxes relies only on the selected LIDAR points.

- Projection-based approaches, like [20] and [19], use deep continuous fusion layers, which were first proposed in [20], to 'project' the image feature into a birds-eye view (BEV) map. Then the projected image BEV map and point cloud BEV are jointly fed into a neural network header to predict the object classification and localization.

- Multi-view approaches, like [3] and [18], use one CNN to extract feature maps from the RGB image, and another CNN to extract feature maps from the point cloud BEV

map. Then the multi-view features are fed into the header, which fuses them and predicts the object classification and localization.

In this thesis, I focus on the feature maps fusion problem in the multi-view methods. Several multi-view 3D object detectors with BEV map as input exist [18] [3]. These methods apply CNNs to the BEV map and RGB image data, and fuse the resulting features and use them to detect objects. In these methods, the multi-view detection networks are trained in an end-to-end fashion. During training, object proposals are labeled according to Intersection-over-Union (IoU) in BEV. However the IoU of proposals is different in top-down view and front view, and as a result the labelled data becomes 'noisy' for the image channel. Consequently, the negative samples with high IoU in front view lead to the deterioration of image feature extractor.

I propose a Multi-view Labelling Object Detector (MLOD) to address this problem. The main contribution of this approach is as follows:

- I propose a foreground mask layer, which exploits the projected depth map in front view to select the foreground image features within a 3D bounding box proposal.
- I propose a multi-view detection header which has output not only from fusion layer, but also from each sensor channel. This design enables our detection network to be trained on the samples labeled based on IoU in the view of each channel.

MLOD is evaluated in the KITTI 3D object detection dataset. The multi-view header is shown to significantly improve the performance of the image channel.

This thesis is organized as follows. I first present some background knowledge on deep learning applied to computer vision and an overview of the current 3D object detectors that use both LIDAR point cloud and image data in Chapter 2. Chapter 3 outlines the proposed detection network architecture and the implementation details, followed by the experimental results in Chapter 4. Finally, I conclude in Chapter 5.

Chapter 2

Background

Artificial Neural Networks (ANNs) have been successfully used to solve many complex tasks related to pattern recognition [15] [35], machine translation [36] [21], self-driving cars [1] [10], and so on. In this chapter, I will present some background knowledge on ANNs, and their application to object detection, including 2D image object detection, and 3D object detection with LIDAR input.

2.1 Artificial Neural Networks

ANNs are computational algorithms originally inspired by biological neural systems (see Figure 2.1). An ANN is composed of a collection of connected units or nodes, commonly referred to as neurons. The feed-forward network is a kind of ANN where the connections of output of certain nodes to the input of other nodes are given by a directed weighted graph.

A neuron takes the output x_i of predecessor neurons with labels 0 to n as its input and computes its output as

$$y = \sigma(\sum_{i=0}^{n} w_i x_i + b),$$

where $\sigma(\cdot)$ is the activation function, b is the bias, and w_i are weights.

If identify function is used as activation function, the whole network becomes a linear system of input since a linear combination of linear models is still a linear system. As many complex tasks can not be addressed by linear models, activation functions are necessarily

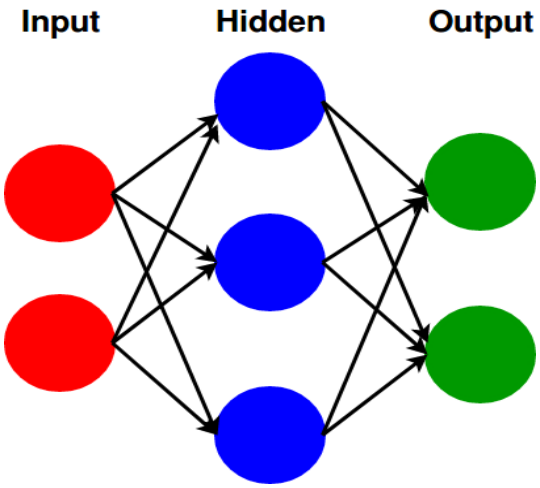


Figure 2.1: An artificial neural network architecture.

nonlinear. It has been proved that a two-layer ANN with non-linear activation function is a universal function approximator [4] [14].

Some popular activation functions are listed as below.

Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$
Tanh	$\sigma(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
Rectified linear unit (ReLU)	$\sigma(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0. \end{cases}$

2.1.1 Stochastic Gradient Descent

There are many ways to train ANNs. Here, we will focus on the first-order optimization algorithms, and we will not discuss algorithms which are not feasible to compute in practice for deep neural networks (NNs).

Gradient descent is one of the most popular algorithms to optimize NNs. For an objective function $L(\theta)$ which is parameterized by $\theta \in \mathcal{R}^d$, gradient descent updates parameters in the opposite direction of the gradient of the objective function $\nabla L(\theta)$ to minimize the objective function. For samples x_i for $i = 0, \ldots, N$, Gradient descent updates parameters

by

$$\theta_t = \theta_{t-1} - \eta \sum_{i=0}^{N} \nabla L(\theta_{t-1}; x_i),$$

where η is the pre-defined learning rate. In many practical tasks, the training data is enormous. Evaluating the sums of all gradients becomes very expensive and thus impractical. Thus, stochastic gradient descent (SGD), which samples a subset of the training set at every step, is used instead:

$$\theta_t = \theta_{t-1} - \eta \sum_{i=0}^{B} \nabla L(\theta_{t-1}; \hat{x}_i),$$

where $\hat{x}_i$ for $i = 0, \ldots, B$ are the randomly selected samples in the training set.

Momentum

SGD becomes very slow when the the surface curves are much more steep in one dimension than in another [42]. Momentum method was introduced in [34] to accelerate SGD by navigating along the relevant direction and softens the oscillations in irrelevant directions. It adds a fraction term τ of the update vector of the past step to the current update vector.

$$\begin{aligned} v_t &= \tau v_{t-1} + \eta \sum_{i=0}^{B} \nabla L(\theta_t; \hat{x}_i), \\ \theta_t &= \theta_{t-1} - v_t \end{aligned}$$

Root Mean Square Propagation

Root Mean Square Propagation (RMSProp) [13] is a method in which a different learning rate is used for every parameter at a time step.

$$\begin{aligned} v_t &= \tau v_{t-1} + (1-\tau) \sum_{i} \left(\nabla L(\theta_t; \hat{x}_i)\right)^2 \\ \theta_t &= \theta_{t-1} - \frac{\eta}{\sqrt{v_t}} \sum_{i} \nabla L(\theta_t; \hat{x}_i) \end{aligned}$$

It is noticed that squaring and square-rooting is done element-wise. The learning rate of RMSProp is adjusted automatically to avoid overshooting.

Adaptive Moment Estimation

Adaptive Moment Estimation (Adam) [16] is another way that computes the adaptive learning rates for each parameter. It does not just calculate exponentially decaying average of past square gradients, but also that of past gradients, similar with Momentum.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) \sum_i \nabla L(\theta_t; \hat{x}_i)$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) \sum_i \left(\nabla L(\theta_t; \hat{x}_i)\right)^2$$

To correct the bias,

$$\hat{m}_t = \frac{m_t}{(1-\beta_1)\sum_{k=1}^{t} \beta_1^{t-k}}$$
$$\hat{v}_t = \frac{v_t}{(1-\beta_2)\sum_{k=1}^{t} \beta_2^{t-k}}$$
$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

2.1.2 Dropout

Since there are millions of parameters in a deep NN, which are much more than the size of data, regularization is necessary. Regularization reduces over-fitting and is often implemented by adding a penalty to the loss function. Srivastava, Nitish, et al [41] proposed another regularization method for NNs, called Dropout.

In the training stage, for each hidden layer, each iteration, any node is either dropped out of the net with probability p, or kept in the net with chance $1-p$. Incoming and outgoing edges to dropped-out nodes are also removed (See Figure 2.2). In the inference phase, all neurons are used, but their activation is reduced by a factor p to keep the expected output of the hidden units consistent with the training.

Dropout can be considered as an ensemble learning method [11]. Dropout produces the original NN into many sub-networks, which are trained independently. During inference, the ensemble output is calculated by as the average of the sub-networks.

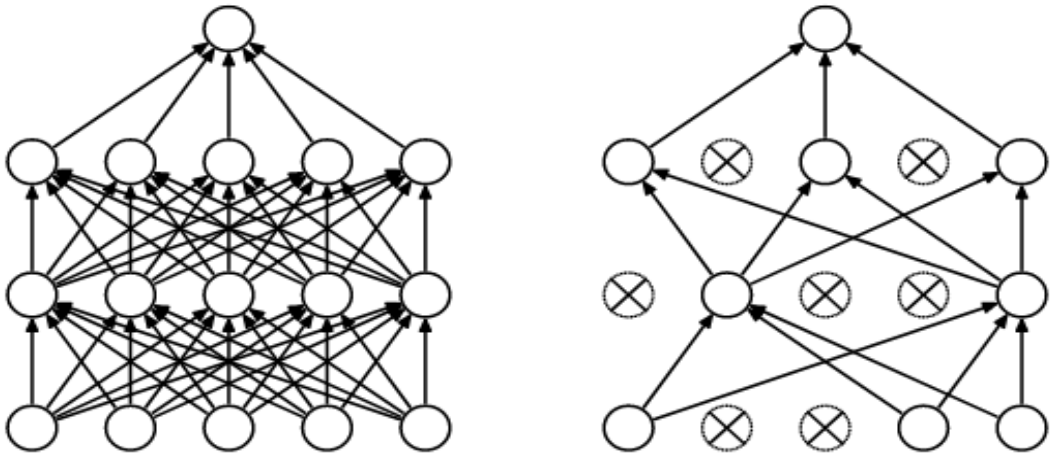


Figure 2.2: Dropout Neural Net Model. Left: An ANN with two hidden layers; Right: The ANN when Dropout is applied.

2.2 Convolution Neural Networks

A convolution neural network (CNN) is a deep learning architecture which is commonly used in computer vision. CNNs were inspired by the organization of the Visual Cortex [25]. Individual neurons respond to stimuli only in a restricted receptive area of the visual field. A collection of such areas overlap to cover the entire visual field.

In general, CNNs are organized as a stack of different convolution layers and pooling layers, which are followed by some fully connected layers (see Figure 2.3).

The convolution operation is an integral that expresses the amount of overlap of one function g as it is reversed and then shifted over another function f:

$$[f * g](t) = \int f(s)g(t-s)ds.$$

For 2D image processing, where data is expressed in the form of 2 dimensional discrete map, the convolution of I with the kernel K is defined as:

$$[I * K](i,j) = \sum_s \sum_t I(s,t)K(i-s, j-t).$$

Convolution Layers are the key component of CNNs. In a convolution layer, input is filtered by a learnable kernel, and then passed to an activation function. Convolution can

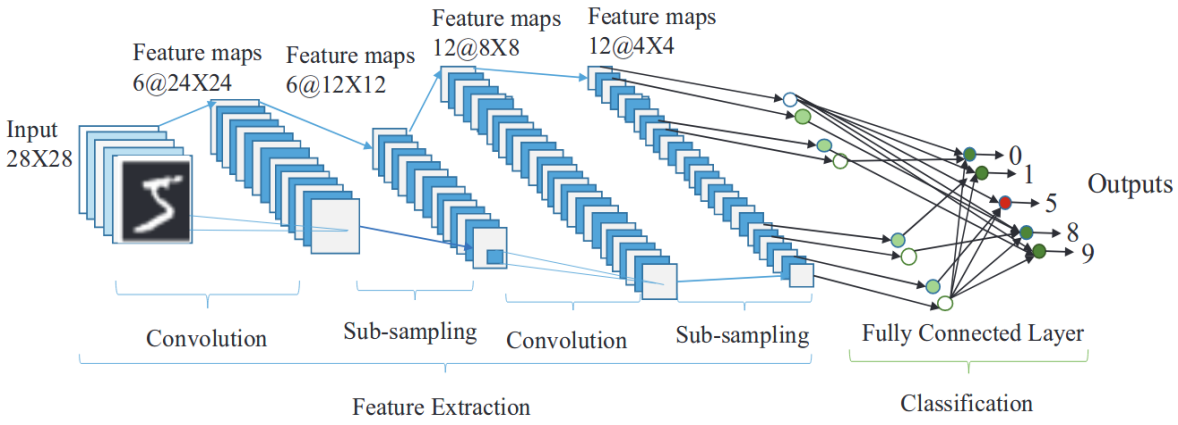


Figure 2.3: An example of CNN architecture [48]

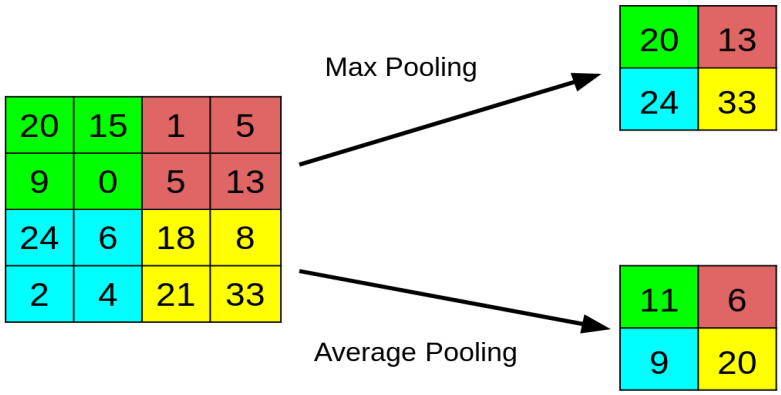


Figure 2.4: Types of Pooling layers.

be viewed as a special type of a linear operation, so the kernel is learned by traditional training algorithm.

Convolution layers have a shared-weights architecture and translation invariance property. Although fully connected layers can be used to learn features from data, it is not practical to apply them to an image. A huge number of nodes would be needed, due to the large size of images. For example, a small image of size 100×100 would need 10^4 number of weight. Thus, it is infeasible to construct deep NNs by stacking fully connected layers to extract higher-level features of image data. A convolution layer uses the same shared weights for each application of the kernel, so that the number of parameters of convolution layers only depends on the kernel size. On the other hand, convolution layers

can successfully extract the spatial and temporal contexts in image data, as it does not require flattening the image data format. If an image represented as a matrix of pixel values is flattened, the spatial dependencies are destroyed by the flatten operation. Every pixel value is treated equally for each node. In a convolution layer, a node only focuses on the nearby pixel values to obtain the local spatial and temporal features.

A pooling layer slides a window over its input and outputs the maximum or average value within the window (see Figure 2.4). These two types of pooling are called max pooling and average pooling, respectively. Pooling layers are commonly used in CNNs to down-sample feature maps, such that less computational power is required to process them. For instance, an image of size 512×512 is reduced to the size of 256×256 after a 2×2 pooling operation.

2.2.1 Selected Convolution Neural Networks Architectures

In recent years, CNNs have achieved significant success in image classification. After the introduction of CNNs, the top-5 error rate in the ImageNet Large Scale Visual Recognition Challenge [6], a benchmark in object classification, reduced from 26% to less than 3.1% [43].

I present two popular CNNs architectures below. These architectures are not just used for the classification task, but also modified and implemented as backbone networks for other tasks, such as object detection [32] [23] and image semantic segmentation [12] [2].

VGG-Net

VGG is a CNN model proposed by Simonyan et al [39] for object classification problem. It achieves 92.7% top-5 test accuracy in ImageNet. VGG is a CNN constructed by stacking a series of 3×3 convolution layers and max pooling layers (see Figure 2.5).

U-Net

Pooling layers are commonly used in CNNs for object classification problem to reduce the computational volume. But some feature information is lost in the downsampling process. It is unfavorable for tasks with high resolution requirement, like object detection and semantic segmentation. Take VGG-16 CNNs as an example, there are 5 2×2 pooling layers. So an image of size 32×32 is compressed into one pixel in final feature map. Thus, it is impossible to distinguish among the pixels in the 32×32 input image.

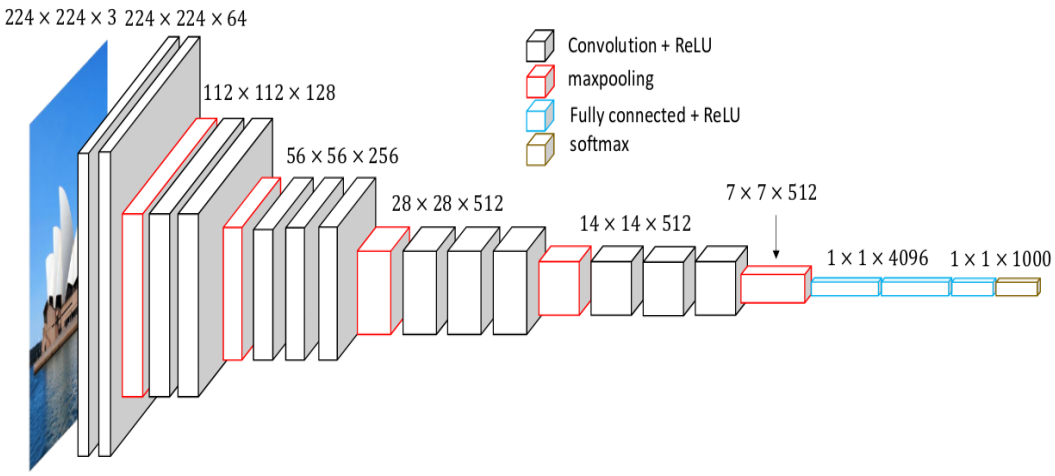


Figure 2.5: The architecture of VGG-16 model [46]

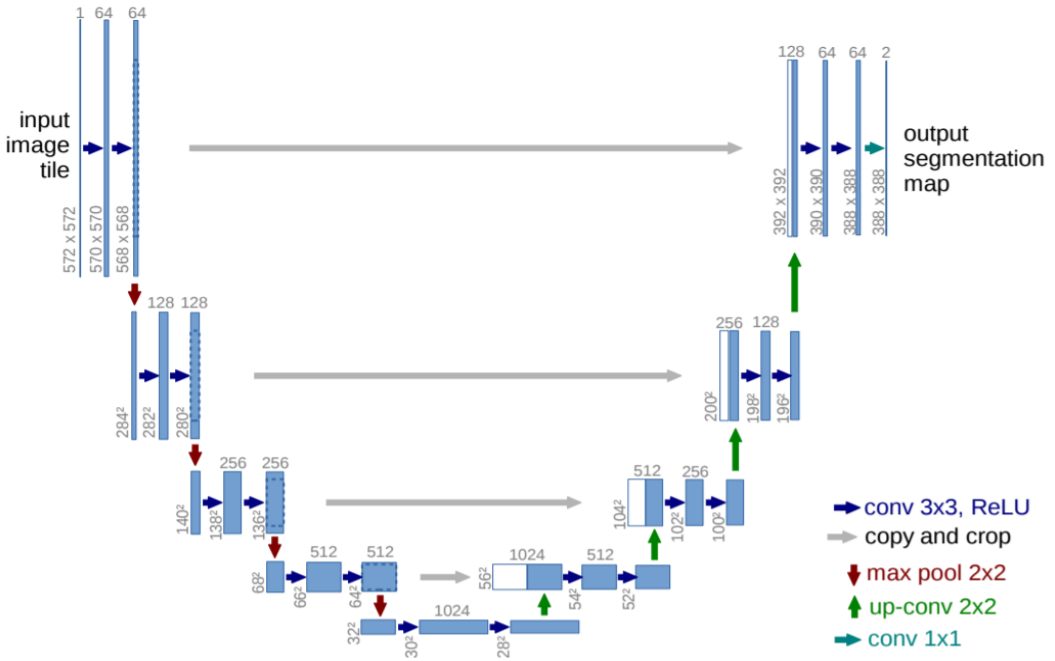


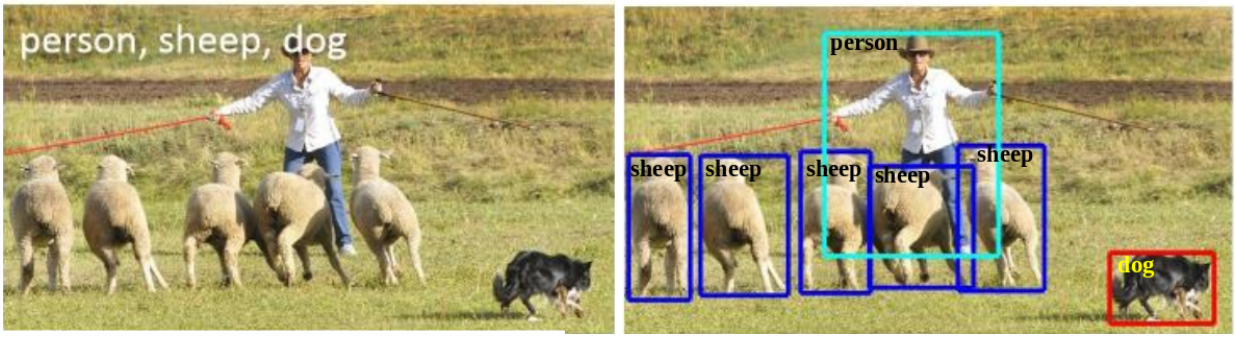
Figure 2.6: The architecture of U-Net model [33]

To overcome the problem, U-Net [33] was developed, originally for biomedical image segmentation. U-Net takes advantage an upsampling operation to propagate context in-

formation to higher resolution layers. The U-Net is composed of a contracting path and an expansive path, which yields a u-shape architecture (see Figure 2.6). The contracting path is a typical CNN which down-samples the feature resolution by pooling layers. The expansive path includes a series of up-convolutions and concatenations. The low resolution features are upsampled through up-convolution, and then concatenated with the corresponding features from contracting path to fuse the information from different levels.

2.3 2D Object Detection

Before discussing 3D object detection, I summarize popular image object detection methods. Object detection is a computer vision task that requires an algorithm to locate instances of certain classes (see Figure 2.7). CNNs based detection methods divide the image into several regions, and classify objects in each region, then combine all results to get the detected objects in the original image.



(a) Object Classification (b) Object Detection

Figure 2.7: Recognition problems. (a) Image Object classification, (b) bounding box level object detection [22]

2.3.1 Region Based Object Detectors

Here I will focus on the Region-Based Convolutional Neural Networks (RCNN) approaches. Since objects in an image have various aspect ratios and spatial locations, a large amount of regions are required to obtain accurate detection results. Hence, region proposal networks are used to select regions, such that much fewer regions are fed into the final classification and localization networks.

Fast RCNN

A typical RCNN uses selective search to extract Region of Interests (RoIs) from an image, followed by a CNN to check whether the regions contain any objects. Hence, such an algorithm needs to run a CNN thousands of times per image.

To speed up RCNN, Ross Girshick proposed a two stage detection method [8], Fast RCNN. Fast RCNN applies a CNN just once per image, and the classification and localization networks take the convolutional feature maps, instead of the original image.

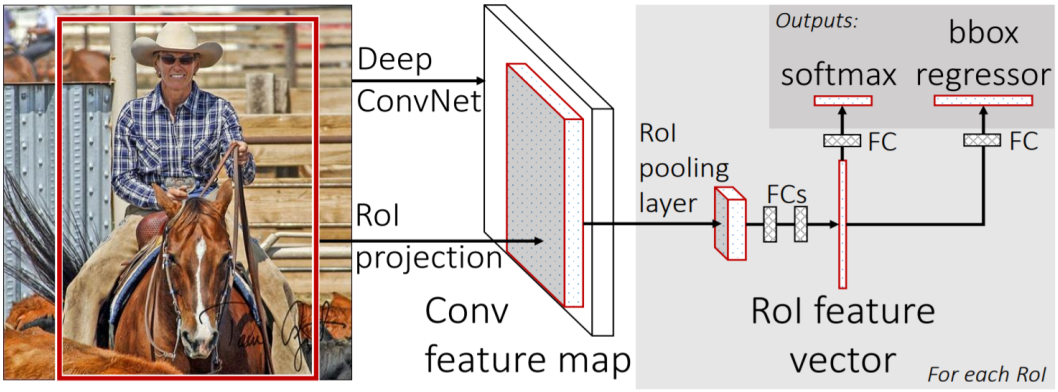


Figure 2.8: The architecture of fast RCNN model [8]

The algorithm is split into three steps below (see Figure 2.8).

1. The algorithm passes an image into a CNN first, and then generates a series of RoIs by using selective search algorithm [44] on the feature maps.

2. An RoI Pooling layer is used to reshape the RoIs into one fixed size.

3. The pooled RoIs are fed into a fully connected layer with softmax output layer to classify the object. Also, a regression layer is used in parallel to refine the coordinates of the bounding boxes.

Faster RCNN

Faster RCNN [32] is a modification of Fast RCNN. Instead of selective search, Faster RCNN uses region proposals network (RPN) to generate RoIs. RPN slides window over

the feature maps. For each window, it generates k anchor boxes with various shapes (see Figure 2.9). The tasks of RPN are to predict the probability that object exists in the ROI, and to adjust the anchors to better fit the object.

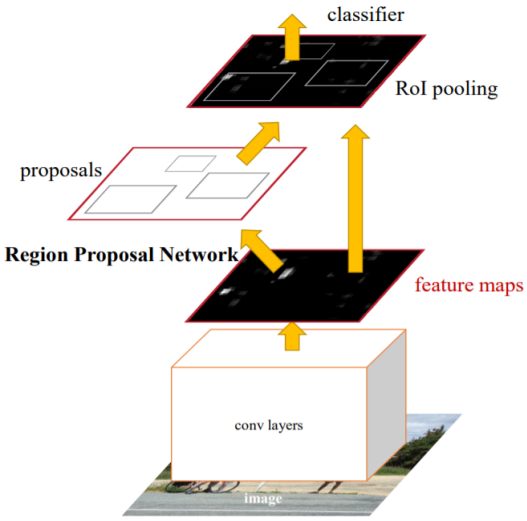


Figure 2.9: The architecture of Faster RCNN model [32]

2.3.2 One-Stage Object Detectors

One-stage approaches refer broadly to architectures that directly predict class probabilities and bounding box offsets from full images with a single feed forward CNN. Since there is no region proposal generation, it is a simpler and faster model architecture.

YOLO

YOLO (You Only Look Once) was first proposed in [30]. It is a one-stage detector casting object detection as a regression problem from image pixels to spatially separated bounding boxes and associated class probabilities. YOLO divides an image into a $S \times S$ grid. Each grid predicts class probabilities, bounding box locations and confidences scores for those boxes (see Figure 2.10).

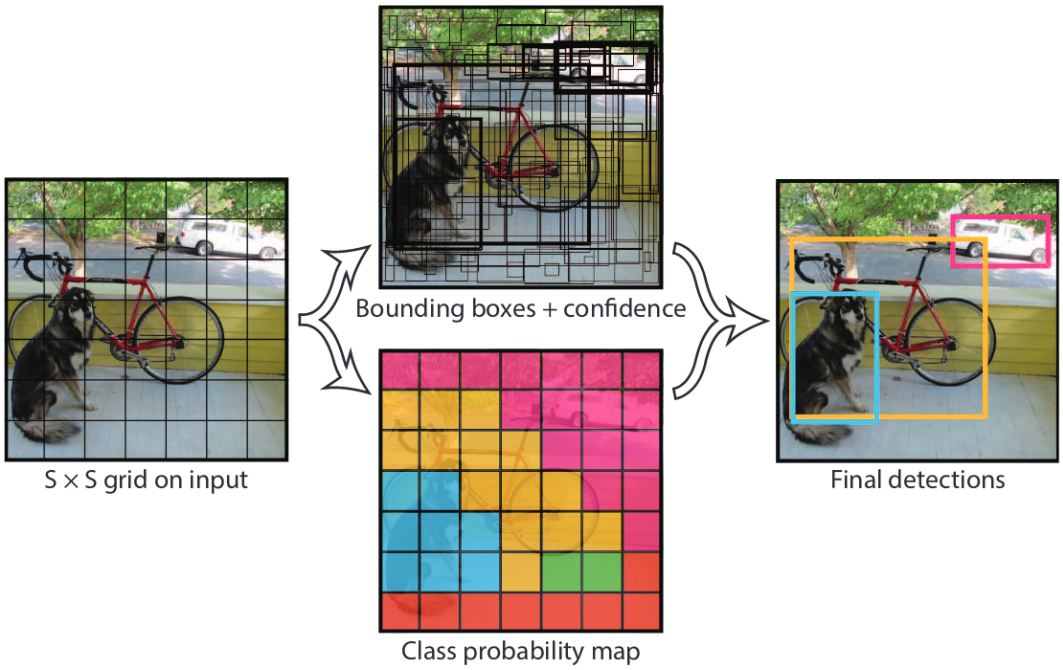


Figure 2.10: The YOLO model [30]

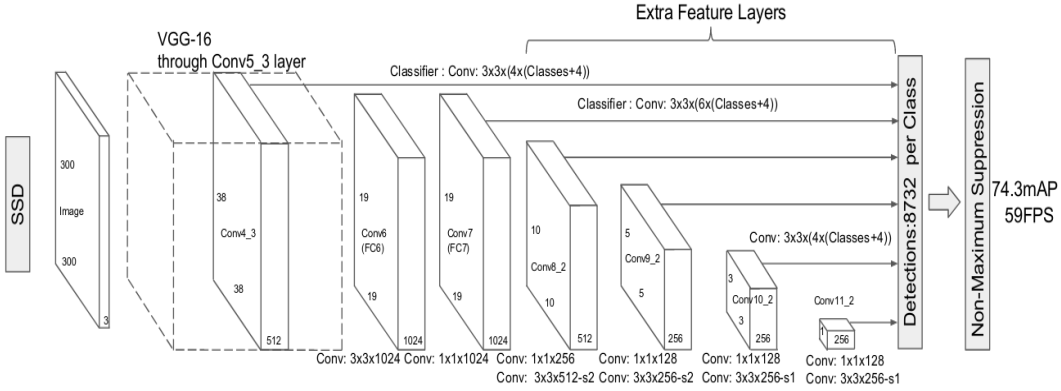


Figure 2.11: The architecture of SSD model [23]

SSD

In order to preserve real-time speed without sacrificing too much detection accuracy, Liu et al. [23] proposed SSD (Single Shot Detector). To speed up the process time, SSD uses convolutional predictors, instead of fully connected layer, for object detection. After ex-

tracting the feature maps, SSD applies convolution filters for each cell to make predictions. Since objects with various scales may appear in the same image, SSD uses multiple layers to detect objects. Shallower layers with higher resolution are used for detecting small objects (see Figure 2.11).

2.4 3D Object Detection

There are many ways to perceive 3D environment, including LIDAR and stereo cameras. In this thesis, I only consider the 3D detection problem with image and LIDAR data input. There are roughly three ways to take advantage of camera and LIDAR for 3D objection detection for autonomous driving scenarios: image region proposal based, projection based and multi-view based methods.

2.4.1 Image Region Proposal Methods

Frustum PointNet (F-PointNet) [27] uses a 2D image detection module to provide 2D bounding boxes as proposals (see Figure 2.12). Then the LIDAR points inside the proposals are cropped and fed into an instance segmentation module using a PointNet [28] to select the positive points. Finally, two additional PointNets predict the bounding box within the selected LIDAR points.

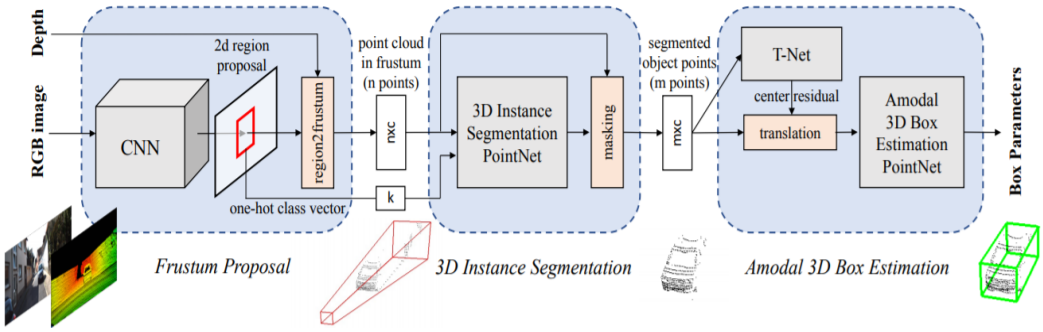


Figure 2.12: The architecture of Frustum PointNet model [27]

IPOD [50] implements a 2D semantic segmentation network to filter out background LIDAR points. Then it classifies and refines 3D bounding boxes on the remaining foreground points.

2.4.2 Projection Based Methods

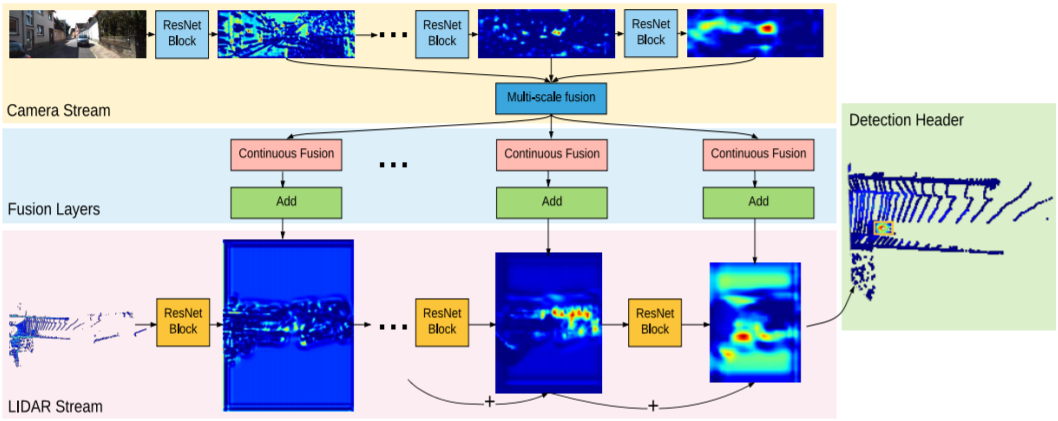


Figure 2.13: The architecture of deep continuous fusion model [20]

Liang et al. [20] proposed a method that projects image features into BEV and fuses them with the convolutional layers of a LIDAR based detector using a continuous fusion layer (see Figure 2.13). The layer creates a continuous BEV feature map where each pixel in BEV contains the corresponding image information. For each BEV pixel, the detector first finds the k nearest LIDAR points on the BEV map, then obtains the image feature at the continuous coordinates by bilinear interpolation. The interpolated image features and geometry offsets are feed into a multilayer perceptron (MLP). Then the deep continuous fusion networks fuse multi-sensor information by the summation of BEV features and output from MLP.

2.4.3 Multi-View Based Methods

MV3D [3] and AVOD [18] are examples of multi-view based two-stage detectors. The multi-view based methods merge features from BEV map and RGB image to predict the 3D bounding boxes. MV3D uses only BEV maps in its RPN to generate proposals, and AVOD uses both BEV and image views. When small instances (like pedestrians and cyclists) on BEV maps are down-sampled by pooling layers, object features are compressed into one pixel in the final feature map, which is insufficient for the second stage detection. Hence, AVOD-FPN improved MV3D by using pyramid convolution structure in BEV/image feature extractors. In AVOD, features are merged in the refinement phase (see Figure 2.14).

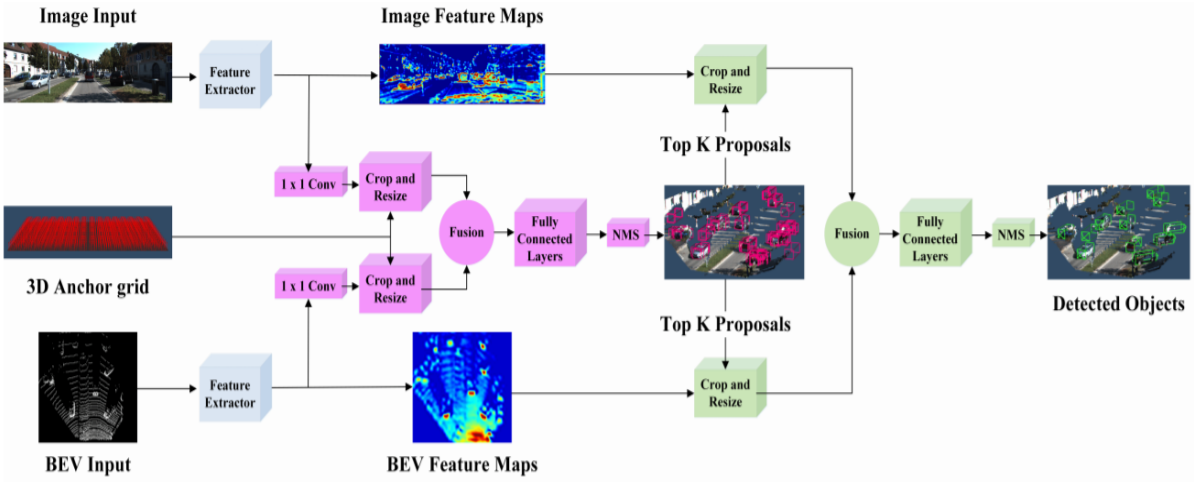


Figure 2.14: The architecture of AVOD model [18]

2.5 KITTI 3D Object Detection Evaluation Metrics

In this section, I discuss some common metrics used to evaluate 3D object detectors for autonomous vehicles.

2.5.1 Intersection over Union

Intersection over Union (IoU) is measures the overlap between predicted bounding box and the ground truth one. It is calculated as follows

$$\text{IoU} = \frac{\text{area of } B_{pred} \cap B_{gt}}{\text{area of } B_{pred} \cup B_{gt}},$$

where B_{pred} and B_{gt} are the predicted bounding box and ground truth bounding box, respectively. Usually, a predicted bounding box is said to match with a ground true one, if its IoU is larger than some threshold number. For example, KITTI benchmark [7] requires IoUs to be larger than 0.7 for cars, and 0.5 for pedestrians and cyclists.

2.5.2 Average Precision

Average Precision (AP) is a popular metric to measure the accuracy of object detectors. It is the average precision value over recall numbers from 0 to 1.

- Precision measures the accuracy of the detection. It is the ratio of positives in the predictions. It is also referred to as positive predictive value.

- Recall measures the ability of detecting all objects. It is the ratio of detected objects in all ground true objects. It is also referred to as the true positive rate.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

where TP, FP, and FN denote the number of true positive, false positive, and false negative samples, respectively.

By ranking the scores of predictions, we can generate the precision-recall curve $p(r)$. In general, AP is the area under the precision-recall curve.

$$AP = \int_0^1 p(r)dr$$

Practically, AP is calculated by the mean of interpolated precision at 11 equally-spaced recall levels.

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

where

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

Unlike the original zigzag precision-recall curve $p(r)$, its interpolated version $p_{interp}(r)$ is a monotonically decreasing function. Thus, the calculated AP is less sensitive to the small variation in ranking of examples. Since AP penalizes the approaches that detect only a subset of ground-true objects with high precision, a detector should have high precision at all levels of recall to obtain a high AP score. Detections are assigned to ground truth labels based on the largest IoU. Multiple detections of the same object are considered as false positives.

2.5.3 Average Orientation Similarity

Average Orientation Similarity (AOS) was first proposed in [7] to assess the performance of jointly detecting objects and estimating their 3D orientation. It is defined as:

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

where s is the orientation similarity at recall level r. It is a normalized variant of the cosine similarity.

$$s(r) = \frac{1}{|D(r)|} \sum_{i \in D(r)} \frac{1 + \cos \Delta_\theta^{(i)}}{2} \delta_i.$$

Here $D(r)$ denotes the set of detections at recall level r. $\Delta_\theta^{(i)}$ is the difference in angle between predicted and ground-truth orientation of detection i. $\delta_i = 1$ if detection i is assigned to a ground-truth label, and $\delta_i = 0$ if it has not been assigned. Thus, the event that an object is detected multiple time is penalized by δ_i .

Ku et al. [18] proposed Average Heading Similarity (AHS) to evaluate 3D detection results. It is the AOS, but evaluated using 3D IoU and global orientation angle instead of 2D IoU and observation angle.

Chapter 3

Multi-view Labelling Object Detector

The existing multi-view based approaches tend to rely more on BEV map input rather than RGB images. Two-stage methods usually select the top k proposals to feed into the detection header networks. Due to the different viewpoints, the IoUs with ground-true box in BEV and image view, respectively, are also different. Hence some proposals are labelled as negative samples in BEV, but should be treated as positive in image view (see Fig. 3.6). Since the positive/negative samples in the current multi-view neural network architectures are assigned based on the IoU in BEV, some positive samples in image view are labelled as negative ones, and thus the image feature extractor is trained on the 'noisy' labels. This problem weakens the performance of the image channel. Therefore the existing multi-view 3D object detectors tend to fail to leverage the image information, and only concentrate on the BEV map.

The proposed two-stage neural network architecture is presented in Figure 3.1. BEV map and RGB image are fed into two convolution neural networks to obtain features. For computational efficiency, we only use the BEV features in RPN to generate 3D proposals. Based on the depth information of the proposals, image features outside 3D proposals are masked by a foreground mask layer. Then the masked image feature map and the BEV feature map are cropped and passed to multi-view header to provide the final classification, localization, and orientation results.

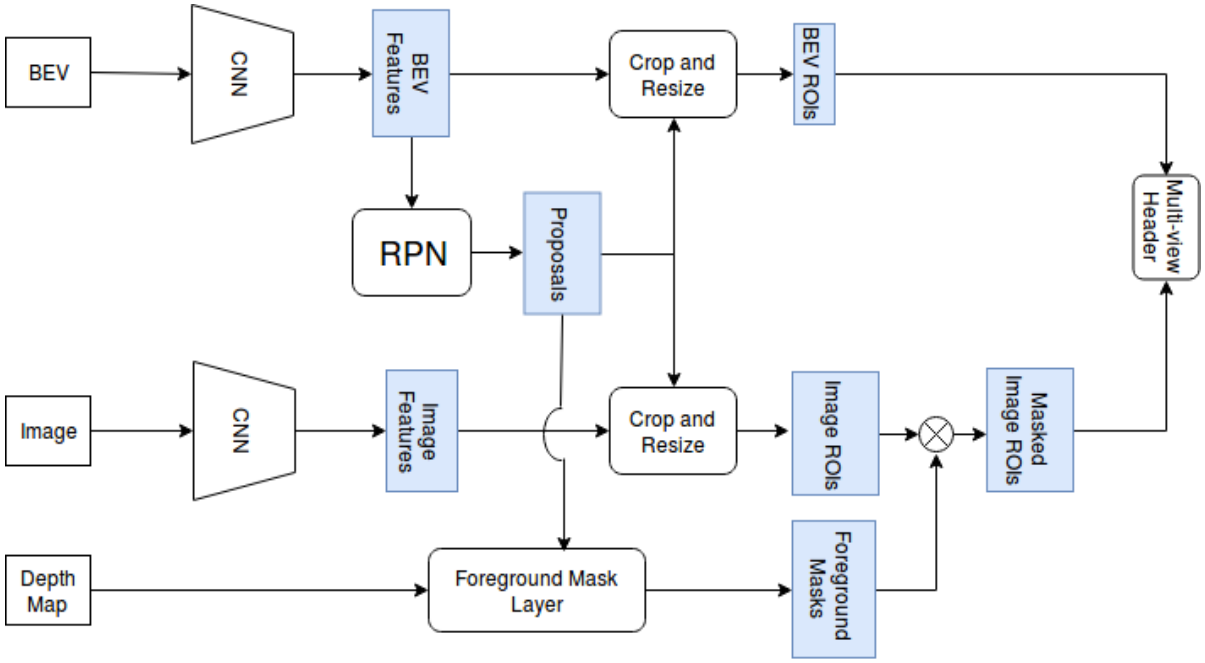


Figure 3.1: Architectural diagram of the proposed method

3.1 The MLOD Architecture

3.1.1 BEV Map Preparation

Similar to [3] [18], the six-channel BEV map input is a 2D grid with 0.1 meter resolution, which includes five height channels and a single density channel. The BEV only includes points which are inside the domain $[-40, 40] \times [0, 70]$ meters, along the x- and z-axis respectively. Hence, the point cloud data is encoded into a BEV map of size $800 \times 700 \times 6$. The point cloud is divided into 5 equal slices between $[0, 2.5]$ meters along the normal of the ground plane, and each slice produces a height channel with each grid cell representing the maximum height of points in that cell. The density map is the logarithm of number of points in a cell, N, and capped at 1.0, i.e. $\min(1.0, \log(N+1)/\log 16)$.

3.1.2 Feature Extractor

We adopt the U-Net [33] structure from [18] as BEV and image feature extractors. The encoder part for the BEV feature extractor is a VGG-like CNN [39], but with CNN layers

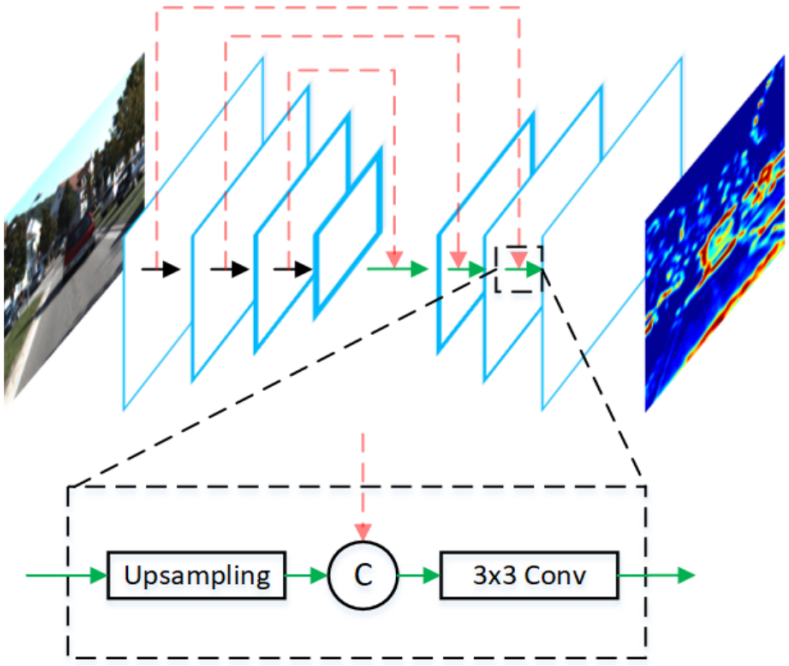


Figure 3.2: The architecture of the feature pyramid extractor. [18]

only up to conv-4 and only half of the channels (see Table 3.1). The encoder part of the image feature extractor is a VGG16 CNN up to conv-5 layer (see Table 3.2). In the decoder part, both feature extractors use the conv-transpose operation to up-sample the feature maps. The up-sampled feature maps are fused with corresponding features from encoder via concatenation (see Figure 3.2).

3.1.3 Foreground Mask Layer

To correctly capture the image features of the object inside the proposed 3D bounding box, we introduce a foreground masking layer to filter out the foreground features.

In order to identify the foreground and background of images, the depth information of each pixel is necessary. But due to the sparsity of the LIDAR point cloud, most of the depth information in the image plane is unknown. Recently, several approaches were proposed to complete the depth map, e.g., [45], [24]. Unfortunately, they typically have high GPU memory usage, and thus are not suitable for our implementation. Instead we introduce a light-weight method to take advantage of the sparse depth information.

Number of Layers	Number of Channels	Operations	Kernel
2	32	conv1	3×3
1	-	max pooling	2×2
2	64	conv2	3×3
1	-	max pooling	2×2
3	128	conv3	3×3
1	-	max pooling	2×2
3	256	conv4	3×3
1	128	upconv3	3×3
1	64	pyramid fusion3	3×3
1	64	upconv4	3×3
1	32	pyramid fusion2	3×3
1	32	upconv5	3×3
1	32	pyramid fusion1	3×3
1	32	upconv5	3×3

Table 3.1: BEV branch feature extractor layers.

Number of Layers	Number of Channels	Operations	Kernel
2	64	conv1	3×3
1	-	max pooling	2×2
2	128	conv2	3×3
1	-	max pooling	2×2
3	256	conv3	3×3
1	-	max pooling	2×2
3	512	conv4	3×3
1	-	max pooling	2×2
3	512	conv5	3×3
1	512	upconv3	3×3
1	256	pyramid fusion3	3×3
1	256	upconv4	3×3
1	128	pyramid fusion2	3×3

Table 3.2: Image branch feature extractor layers.

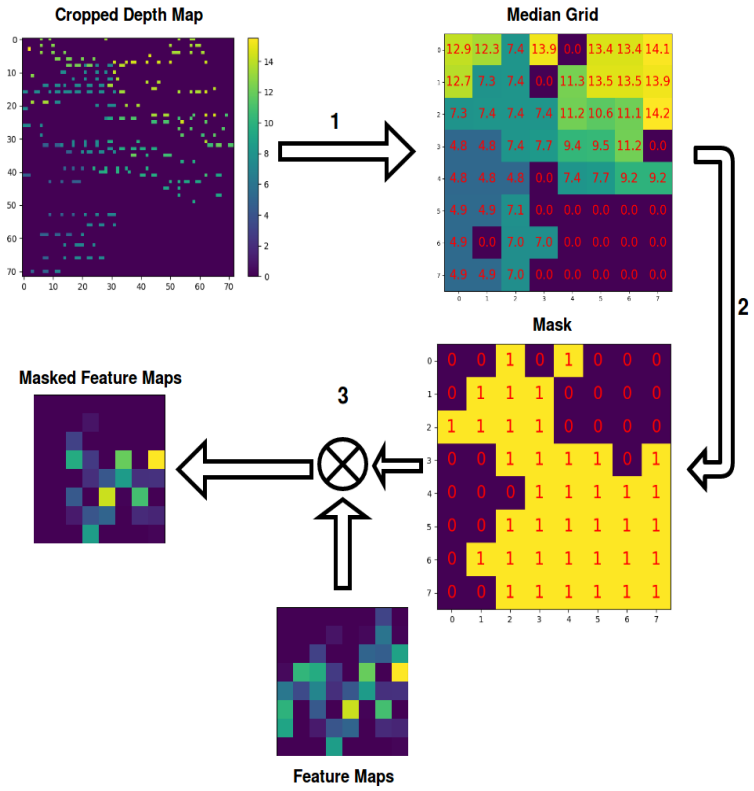


Figure 3.3: Illustration of foreground masking layer procedure: Step 1: calculating the median of nonzero values in each grid; Step 2: obtaining a mask by Equation 3.1 ($d_{min} = 6.8, d_{max} = 9.7$ in this example); Step 3: applying the mask to the feature maps.

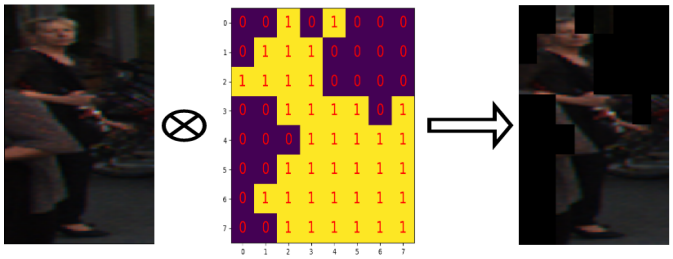


Figure 3.4: A qualitative example of a foreground mask and its application to the original image. The bottom left background and the top left and right background are masked.

Figure 3.3 presents the procedure of the foreground masking layer. First, the layer crops

and resizes the (sparse) depth map using front-view 2D bounding boxes, projected from the 3D proposals. For computational convenience, the resized depth map is n times the $k \times k$ size of the cropped image feature map. Since the depth information is discontinuous in front view, we use nearest neighbour interpolation algorithm to obtain the resized depth map. Then the $nk \times nk$ depth map is split equally into a $k \times k$ grid. Thus each grid cell represents the depth information of the corresponding pixel in the $k \times k$ image feature map. The layer calculates the median m_{ij} of the nonzero depth values in each grid cell, as zero value means no LIDAR point information for this pixel. Note that all depth values in a grid cell may be zero, due to the sparsity of point cloud. Since far objects have fewer projected LIDAR points, some parts of these objects do not have any depth information. Thus, to preserve the image features that are inside the 3D bounding box or have no depth information, we set the foreground mask as

$$Mask_{ij} = \begin{cases} 1 & \text{if } m_{ij} \in [d_{min} - \epsilon_1, d_{max} + \epsilon_1] \cup [0, \epsilon_2] \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

where d_{max} and d_{min} are the maximum and minimum depth value of a 3D bounding box, respectively. ϵ_1 and ϵ_2 are small buffers to absorb the uncertainty of 3D proposals and point cloud.

3.1.4 Multi-view Header

In the current multi-view 3D object detection methods, the labels of proposals are assigned based on the IoU in BEV. But the IoU in front view can be significantly different than that in BEV. Fig. 3.6 shows an example that a 3D bounding box is assigned to a negative label, but has IoU > 0.7 in image view. When object detectors are trained on labels assigned based on IoU only in BEV, the performance of (front-view) image channel is degraded.

We propose a multi-view detection header to avoid the decay of RGB image features. Figure 3.5 shows the header network structure. The key idea is to add an extra output layer to each channel before the (Concat) fusion layer. Each of the two outputs feeds into a corresponding sub-output loss. Each sub-output loss is calculated using the labels assigned by IoU in the corresponding channel's view, i.e.

$$\begin{aligned} L_{sub-cls} = & \frac{1}{N} \sum_i L_{cls}(y_i^{img}, \hat{y}_i^{img}) \\ & + \frac{1}{N} \sum_i L_{cls}(y_i^{bev}, \hat{y}_i^{bev}) \end{aligned}$$

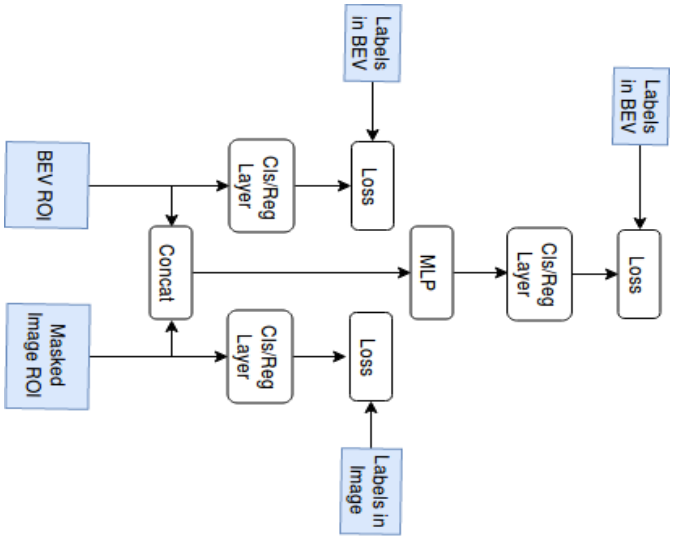


Figure 3.5: The multi-view header architecture diagram

$$L_{sub-reg} = \frac{1}{N_p^{img}} \sum_i I[\hat{y}_i^{img} > 0] L_{reg}(s_i^{img}, \hat{s}_i^{img}) + \frac{1}{N_p^{bev}} \sum_i I[\hat{y}_i^{bev} > 0] L_{reg}(s_i^{bev}, \hat{s}_i^{bev}).$$

$I[\cdot > 0]$ is the indicator function to select the positive proposals. N, N_p^{img} and N_p^{bev} are the number of total samples, positive samples in image view and BEV, respectively. y_i^{img} and y_i^{bev} are the classification scores for proposal i obtaied from image and BEV branch, respectively, and $\hat{y}_i^{img}$ and $\hat{y}_i^{bev}$ are the corresponding ground-truth labels. The predicted corner offsets for each branch are s_i^{img} and s_i^{bev}, and the corresponding ground truth lables are $\hat{s}_i^{img}$ and $\hat{s}_i^{bev}$.

We use a multi-task loss to train our network. The loss function of the detection

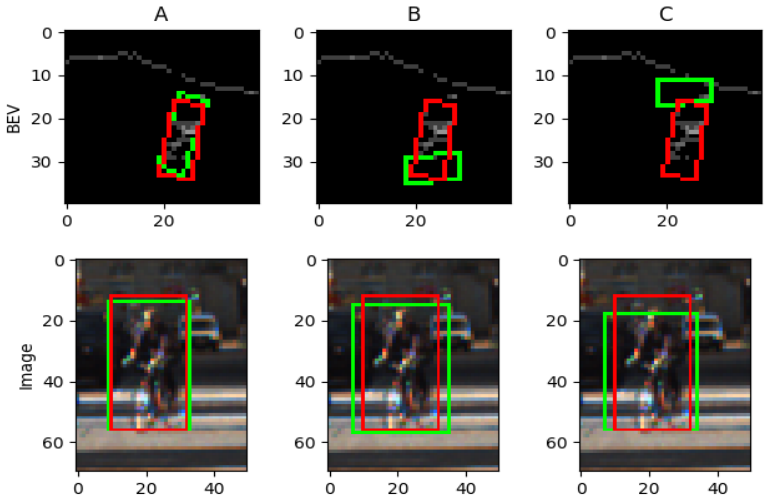


Figure 3.6: Examples of IoU in different views. The pictures show the projection of 3D bounding boxes (proposals A,B,C in green and the ground truth in red) onto ground plane (BEV) and image. The IoU of proposals B and C is less than 0.3 in BEV, but is larger than 0.7 in image view. Hence proposals B and C are negative in BEV and positive in front view.

network is defined by Eq. 3.2,

$$
\begin{aligned}
L = & \frac{\lambda_{cls}}{N} \sum_i L_{cls}(y_i^{fusion}, \hat{y}_i^{bev}) \\
& + \frac{\lambda_{reg}}{N_p^{bev}} \sum_i I[\hat{y}_i^{bev} > 0] L_{reg}(s_i^{fusion}, \hat{s}_i^{bev}) \\
& + \frac{\lambda_{ang}}{N_p^{bev}} \sum_i I[\hat{y}_i^{bev} > 0] L_{ang}(a_i^{fusion}, \hat{a}_i^{bev}) \\
& + \lambda_{sub-cls} L_{sub-cls} + \lambda_{sub-reg} L_{sub-reg}.
\end{aligned}
\tag{3.2}
$$

We use smooth L1 loss for 3D bounding box offset and orientation rotation regression, and cross-entropy loss for classification. λ are the hyperparameters to balance the different loss terms. The sub-output losses can be considered as a kind of regularization on the network.

3.2 Implementation Details

KITTI benchmark [7] uses different IoU thresholds for the car class (>0.7) and the pedestrian and cyclist classes (>0.5). Hence, following [18], we train two networks, one for cars, another for pedestrians and cyclists. The RPN network and the detection header are trained jointly using mini-batches with 1024 ROIs. We use ADAM [16] optimizer with an exponentially decayed learning rate initialized to 0.0001. For the car network, we apply a decay factor of 0.1 every 100K iterations. For the pedestrian and cyclist network, we apply a decay factor of 0.5 every 20K iterations. Image feature extractor loads pre-trained ImageNet [6] weights. The weights of the BEV feature extractor are initialized by Xavier uniform initializer [9].

3.2.1 Mini-batch Settings

A car proposal is marked as positive in top-down/front view if the BEV/image IoU with a ground truth object is larger than 0.65/0.7, respectively. It is marked negative if its BEV/image IoU is less than 0.55/0.5, respectively. A positive pedestrian or cyclist proposal has at least 0.45/0.6 IoU in BEV/image view, respectively. A negative sample has no more than 0.4/0.4 IoU in BEV/image view, respectively. For mini-batches, we first select 1024 samples consisting of both positive ROIs and negative ROIs with highest RPN scores in top-down view, then pick ROIs which are positive or negative in front view.

3.2.2 3D Box Encoding

There are many ways to encode 3D boxes (e.g., [18], [3], [40]). To reduce the number of parameters and keep physical restrictions, we follow the encoding method from [18], where the 3D bounding box is represented as four corners on X-Y plane and the top and bottom corner height offsets from the ground plane (see Figure 3.7). In regression stage, the final bounding box prediction is determined by aligning the quadrilateral, which is formed by the network output of 4 skewed corners. Figure 3.8 shows the alignment process. First, two segments joining the midpoints of opposite sides of the quadrilateral are calculated. Then the predicted bounding box is the minimum rectangle covering the quadrilateral, and aligned with the longest segment. It is calculated by choosing the minimum and maximum corner values along each axis.

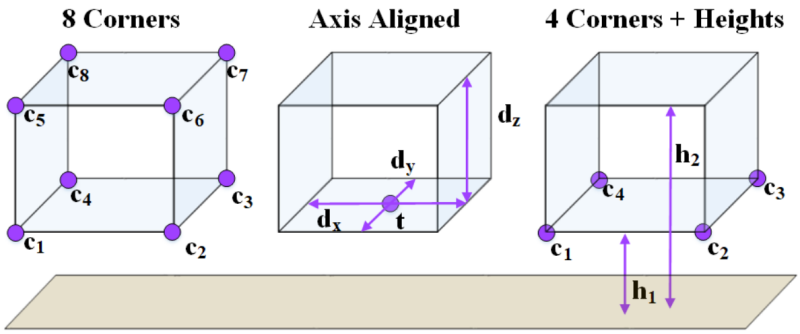


Figure 3.7: A visual comparison between various box encoding method: the 8 corners box encoding method [3], the axis aligned box encoding method [40], and 4 corners encoding method [18]

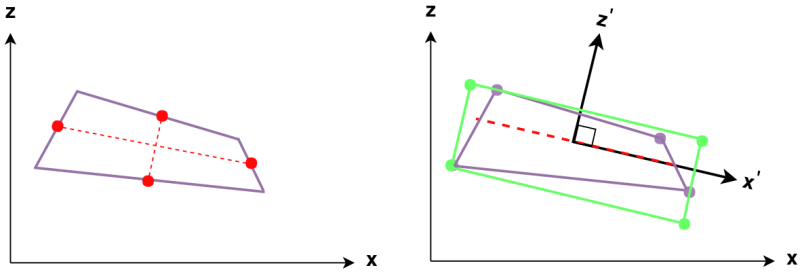


Figure 3.8: A visual example of aligning skewed regressed 4 corners. The left figure shows how the reference line is selected along the longest side. The right figure shows how the corners are aligned with respect to the selected line [26]. Note that the figure uses the camera coordinate system convention, where the horizontal plane is represented by the x,z axes.

3.2.3 Data Augmentation

Data augmentation is an important technique for increasing the number of training instances and reducing overfitting. Two augmentation methods, flipping and PCA jittering [17], are implemented in our network training. The point clouds and images are flipped along the x-axis. PCA jittering alters the intensities of the RGB channels in training images. PCA decomposition is applied to the set of RGB pixel values of the whole set of training images. Then Gaussian random noise is added to the principle components of images.

Chapter 4

Experiments

4.1 KITTI Dataset

We evaluate MLOD on the 3D detection tasks for cars, pedestrians, and cyclists of the KITTI Object Detection Benchmark [7]. The data was recorded on a modified Volkswagen Passat B6, which used Velodyne HDL-64E Laser scanner to collect LIDAR point cloud data, and 2 Point Grey Flea 2 colour cameras for image RGB data.

The 3D object detection dataset of KITTI contains $7,481$ training frames and $7,518$ testing frames. The frames contain target-class objects categorized into into three difficulty levels: easy (E), moderate (M), and hard (H), based on the occlusion level, maximum truncation and minimum bounding box height. Since no official validation set is provided, the labelled $7,481$ frames are split into a training set and a validation set at $1:1$ ratio, similar to [18] and [3].

Method	Cars	Pedestrians	Cyclist
MV3D [3]	72.4	-	-
AVOD [18]	**74.4**	58.8	49.7
Ours	74.1	**63.9**	**54.6**

Table 4.1: A comparison of AP_{3D} from MLOD and current state-of-art 3D object detectors on validation set at the moderate difficulty.

		$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
Method	Class	E	M	H	E	M	H
AOVD-FPN[18]	Car	**81.94**	**71.88**	**66.38**	88.53	83.79	**77.90**
MV3D [3]		71.09	62.35	55.12	86.02	76.90	68.49
F-PointNets [28]		81.20	70.39	62.19	**88.70**	**84.00**	75.33
MLOD		72.24	64.20	57.20	85.95	77.86	76.93
AOVD-FPN[18]	Pedestrian	50.80	42.81	**40.88**	**58.75**	**51.05**	**47.54**
MV3D [3]		-	-	-	-	-	-
F-PointNets [28]		**51.21**	**44.89**	40.23	58.09	50.22	47.20
MLOD		48.26	40.97	35.74	52.24	44.40	43.24
AOVD-FPN[18]	Cyclist	64.00	52.18	46.61	68.09	57.48	50.77
MV3D [3]		-	-	-	-	-	-
F-PointNets [28]		**71.96**	**56.77**	**50.39**	**75.38**	**61.96**	**54.68**
MLOD		67.66	49.89	42.23	69.68	58.21	50.14

Table 4.2: A comparison of the performance of MLOD with current state-of-art 3D object detectors

4.2 Accuracy

To evaluate the performance of MLOD, we present the Average Precision (AP) results over the validation set and the KITTI test set in Table 4.1 and 4.2, respectively. MLOD outperforms two other state-of-the-art multi-view object detectors on the validation set. However, our method perfroms worse than AVOD on the KITTI test set. This may caused by the different ground planes used in MLOD and AVOD. The evaluation shows that our method can reach the current state of art result, however.

4.3 Ablation Study

4.3.1 Effects of Multi-view Header

To evaluate the effects of the multi-view header, we compare the AP(%) of MLOD with different $\lambda_{sub-cls}$ settings in Table 4.3 on the validation set. When $\lambda_{sub-cls}/\lambda_{cls} = 0.001$, the fusion channel, with BEV labelled samples, dominates the network training, such that the sub-channel losses are ignorable. The multi-view header is shown to provide significant

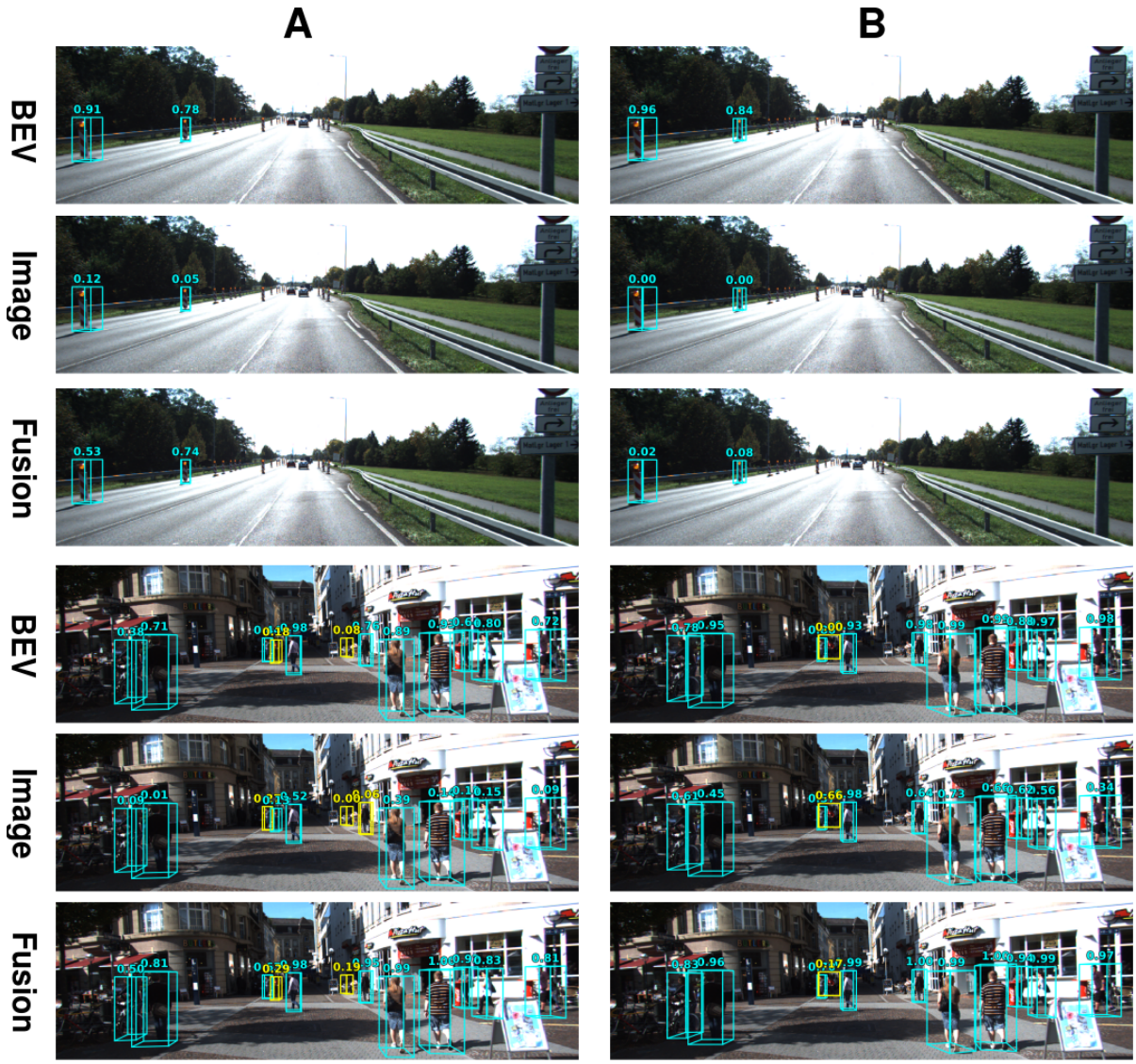


Figure 4.1: Examples of the effects of various λ settings. Column A: $\lambda_{sub-cls}/\lambda_{cls} = 0.001$; Column B: $\lambda_{sub-cls}/\lambda_{cls} = 1$. Blue boxes: pedestrians; Yellow boxes: cyclist.

performance gains for image channel, ranging from 5% to 20%, however. The final detection AP achieves an increase of 6.7%, 5.2% and 4.5% in AP for Pedestrians Easy, Moderate, and Hard classes, respectively. Figure 4.1 shows an example of the effects of multi-view header. Note when $\lambda_{sub-cls}/\lambda_{cls} = 0.001$, the image channel fails to correctly classify any

object detection from LIDAR BEV, and it assigns score of 0.0 to true or false positives from LIDAR BEV. Since the image channel information is not reliable, the fusion channel just takes advantage of the BEV features to provide the final prediction. On other hand, when $\lambda_{sub-cls}/\lambda_{cls} = 1$, the image channel correctly assigns score of 0.0 to the pedestrian false positives from LIDAR BEV, such that the fusion channel notices the false positives from LIDAR BEV, by observing the image channel information.

		Pedestrians			Cyclist		
$\frac{\lambda_{sub-cls}}{\lambda_{cls}}$	Branch	E	M	H	E	M	H
0.001	Fusion	65.2	58.7	51.7	71.5	53.6	47.5
	Image	53.3	47.3	41.5	39.5	23.6	22.8
1	Fusion	71.9	63.9	56.2	73.5	54.6	52.7
	Image	59.4	52.8	49.7	59.2	40.2	38.5

Table 4.3: AP_{3D} from MLOD with different λ settings, evaluated on the validation set. Since the image channel lacks depth information, it is difficult to predict the 3D bounding box from it. To facilitate the comparison, results from fusion and image channel use the same 3D bounding boxes. Thus, the shown results reflect only the variation of classification results.

4.3.2 Effects of Foreground Mask Layer

Table 4.4 shows how the mask component affects the performance of MLOD.

	Pedestrians			Cyclist		
	E	M	H	E	M	H
With masks	**71.9**	**63.9**	**56.2**	73.5	**54.6**	**52.7**
W/o masks	69.1	61.4	53.6	**74.1**	54.2	52.5

Table 4.4: Effects of a foreground mask layer.

4.4 Qualitative Results

Some qualitative results in 3D and image space are presented in Figure 4.2 - 4.5.

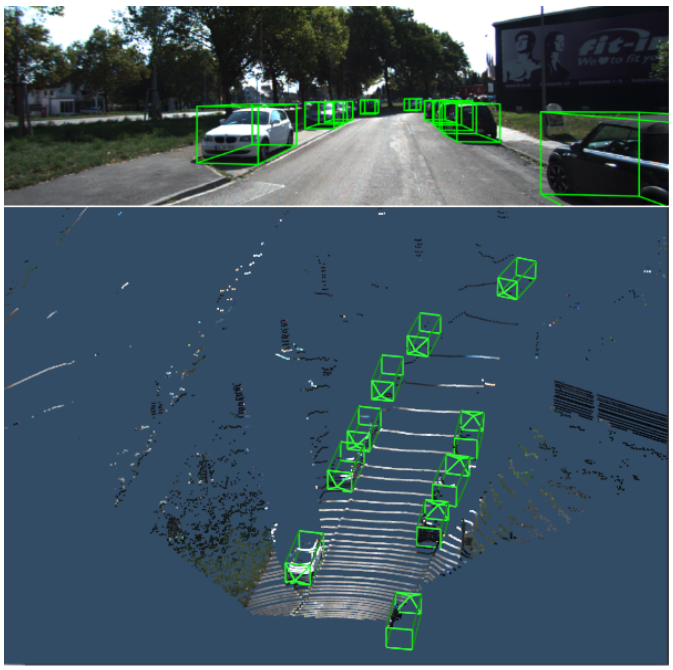


Figure 4.2: MLOD predictions on sample 000010

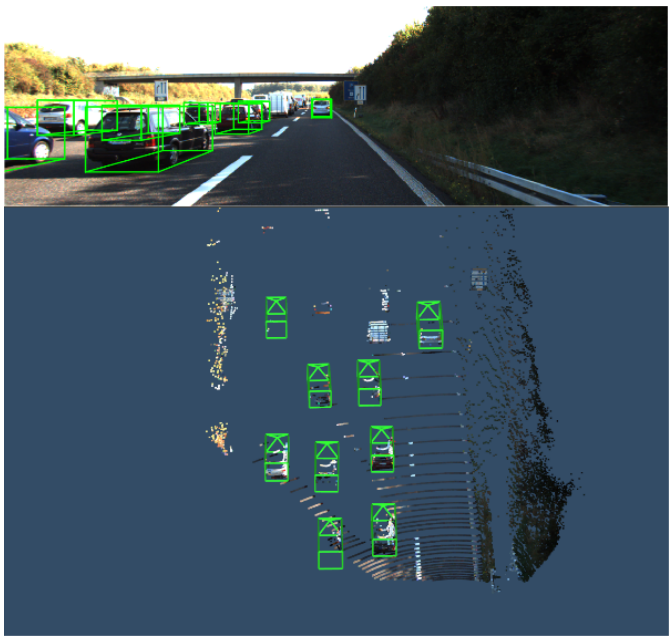


Figure 4.3: MLOD predictions on sample 000357

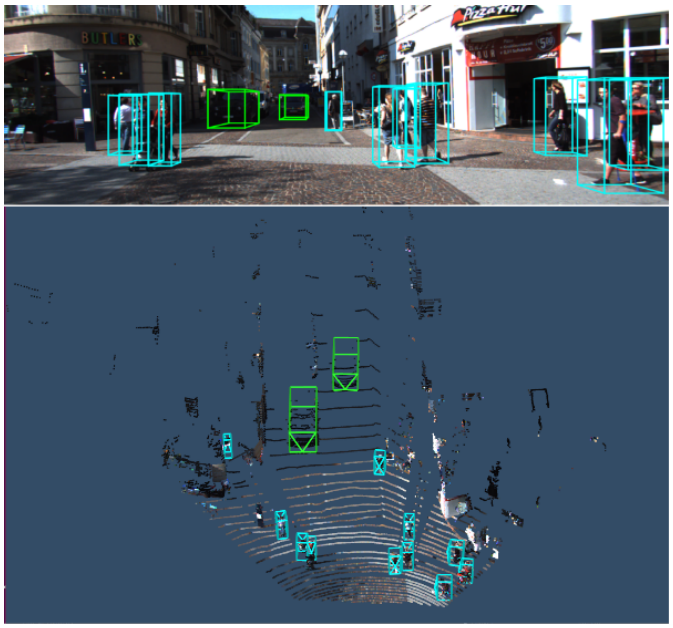


Figure 4.4: MLOD predictions on sample 002477

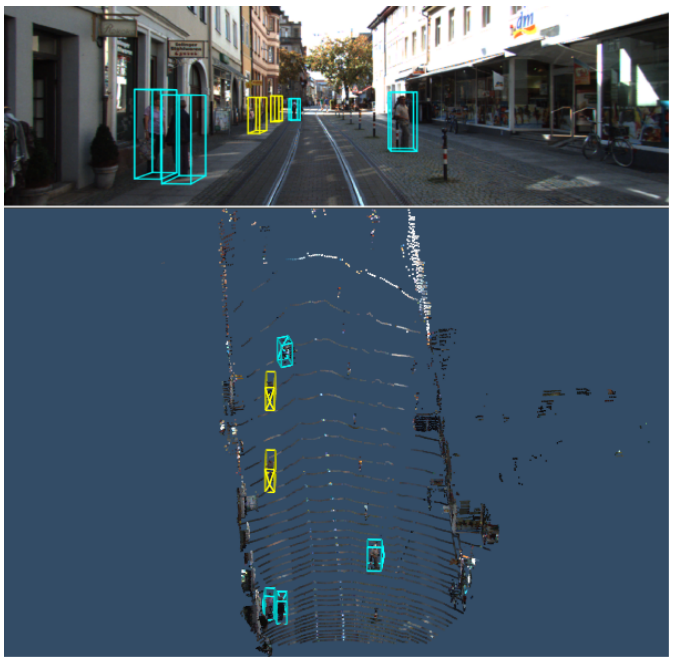


Figure 4.5: MLOD predictions on sample 003759

Chapter 5

Conclusion

Camera and LIDAR sensors fusion is one of the most challenging problem in 3D environment perception for for autonomous driving scenarios, as the RGB image and LIDAR point cloud data represent different physical characteristics of instances. They are different in many ways, such as in data format and geometric alignment.

In this thesis, a multi-view based 3D object detection model is proposed. In order to obtain the image features inside 3D bounding box proposals, a foreground mask layer is introduced. This layer uses LIDAR depth points to remove the background information in the image feature maps.

Like other multi-view based 3D object detector, such as MV3D[3] and AVOD[18], point cloud is encoded into the form of BEV maps in this method. Since BEV maps have different viewpoint compared to image data, the proposed detector is trained with the multi-view rather than BEV-only labelled data to prevent the decay of the image channel, and is shown to provide better classification and localization outcomes as a result. Evaluated on KITTI detection dataset, our method achieves state of the art benchmark performance.

Our detection method relies on the ground prior, which is considered as a plane. However, it is often inaccurate for vertically curved roads [49]. 3D object detector with BEV input can be further improved by non-plane ground estimation, which leverages the geometry of the ground to provide better 3D location. Another idea for future work is to use uncertainty in the depth predictions to accomplish LIDAR and image fusion [45] [29]. The depth completion methods use confidence/uncertainty masks to handle mixed LiDAR signals near foreground boundaries due to occlusion, and combine estimates from the color image. This idea could be extended to 3D object detection, and potentially provide more accurate depth estimation at the edges of instances.

References

[1] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[3] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[4] George Cybenko. Approximations by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:183–192, 1989.

[5] Varuna De Silva, Jamie Roche, and Ahmet Kondoz. Robust fusion of lidar and wide-angle camera data for autonomous mobile robots. *Sensors*, 18(8):2730, 2018.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[8] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[9] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[10] Alexandru Gurghian, Tejaswi Koduri, Smita V Bailur, Kyle J Carey, and Vidya N Murali. Deeplanes: End-to-end lane position estimation using deep neural networksa. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 38–45, 2016.

[11] Kazuyuki Hara, Daisuke Saitoh, and Hayaru Shouno. Analysis of dropout learning regarded as ensemble learning. In *International Conference on Artificial Neural Networks*, pages 72–79. Springer, 2016.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[13] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14:8, 2012.

[14] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[15] Anil K Jain, Jianchang Mao, and KM Mohiuddin. Artificial neural networks: A tutorial. *Computer*, (3):31–44, 1996.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

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

[19] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019.

[20] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 641–656, 2018.

[21] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[22] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*, 2018.

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[24] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. Self-supervised sparse-to-dense: self-supervised depth completion from lidar and monocular camera. *arXiv preprint arXiv:1807.00275*, 2018.

[25] Masakazu Matsugu, Katsuhiko Mori, Yusuke Mitari, and Yuji Kaneda. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16(5-6):555–559, 2003.

[26] Melissa Farinaz Mozifian. Real-time 3d object detection for autonomous driving. Master's thesis, University of Waterloo, 2018.

[27] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum PointNets for 3d object detection from RGB-D data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[28] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[29] Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc Pollefeys. Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3313–3322, 2019.

[30] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[31] Jimmy Ren, Xiaohao Chen, Jianbo Liu, Wenxiu Sun, Jiahao Pang, Qiong Yan, Yu-Wing Tai, and Li Xu. Accurate single stage detector using recurrent rolling convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5420–5428, 2017.

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[34] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[35] Sandhya Samarasinghe. *Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition*. Auerbach publications, 2016.

[36] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[37] Shaoshuai Shi, Zhe Wang, Xiaogang Wang, and Hongsheng Li. Part-aˆ 2 net: 3d part-aware and aggregation neural network for object detection from point cloud. *arXiv preprint arXiv:1907.03670*, 2019.

[38] Kiwoo Shin, Youngwook Paul Kwon, and Masayoshi Tomizuka. Roarnet: A robust 3d object detection based on region approximation refinement. *arXiv preprint arXiv:1811.03818*, 2018.

[39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[40] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in RGB-D images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 808–816, 2016.

[41] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[42] Richard Sutton. Two problems with back propagation and other steepest descent learning procedures for networks. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986*, pages 823–832, 1986.

[43] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[44] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[45] Wouter Van Gansbeke, Davy Neven, Bert De Brabandere, and Luc Van Gool. Sparse and noisy LiDAR completion with RGB guidance and uncertainty. *arXiv preprint arXiv:1902.05356*, 2019.

[46] Suhang Wang, Yilin Wang, Jiliang Tang, Kai Shu, Suhas Ranganath, and Huan Liu. What your images reveal: Exploiting visual contents for point-of-interest recommendation. In *Proceedings of the 26th International Conference on World Wide Web*, pages 391–400. International World Wide Web Conferences Steering Committee, 2017.

[47] Zhixin Wang and Kui Jia. Frustum ConvNet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. *arXiv preprint arXiv:1903.01864*, 2019.

[48] Chris Yakopcic, Md Zahangir Alom, and Tarek M Taha. Memristor crossbar deep network implementation based on a convolutional neural network. In *2016 International joint conference on neural networks (IJCNN)*, pages 963–970. IEEE, 2016.

[49] Bin Yang, Ming Liang, and Raquel Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In *Conference on Robot Learning*, pages 146–155, 2018.

[50] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. IPOD: Intensive point-based object detector for point cloud. *arXiv preprint arXiv:1812.05276*, 2018.