# OBJECT DETECTION IN VIDEO SURVEILLANCE USING DEEP LEARNING APPROACHES

**CHENHANG HE**

**MPhil**

**The Hong Kong Polytechnic University**

**2019**

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

# Object Detection in Video Surveillance using Deep Learning Approaches

Chenhang He

A thesis submitted in partial fulfilment of the requirements

for the degree of Master of Philosophy

October 2018

# CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

______________________(Signed)

___________Chenhang He_____(Name of student)

## Abstract

Object detection remains a challenging task in computer vision, not only because an object may appear at different scales with different non-rigid transformations, but also because of the intra-class variations and the variations caused by different viewpoints and illumination. Furthermore, some objects may be deformable, and the pose of an object appears differently in different scenarios. Object detection based on handcrafted features and conventional machine-learning methods has been researched for decades and has a proven performance with real-world applications.

With the rise of deep learning, many classification problems have received a radical enhancement, in terms of accuracy, by using convolutional neural network (CNN), and this has inspired the use of CNN for detecting objects. In this thesis, we will first review some conventional object-detection approaches which use handcraft features and classical machine-learning algorithm, and then will conduct a survey on several CNN-based object-detection approaches. The region-based approaches use a coarsely trained CNN to extract class-agnostic region proposals, and then feed these proposals into a deep CNN for further classification and localization refinement. Proposal-free approaches have also been proposed, in which the CNN is used for both classification and localization. In our experiments, we find that different convolutional layers have different semantic meanings. A good localization requires feature with fine-grained details and high semantic features are better for classification. Therefore, we decouple the classification and localization from a single layer as majority of detector does. As a result, we proposed a fast vehicle detector, with a novel lateral convolutional network, which engages residual learning to enhance the localization but retains a high recall rate. The proposed detector can achieve relatively high accuracy and real-time detection for video surveillance. We have also designed an efficient traffic-sign

classifier for an IEEE competition. Recently, we have also explored 3D object detection based on point clouds. A point cloud, unlike an image, has irregular data structure, which allows to give the same geometric representation with different permutation of the data. Unlike most of the literature, which projects the point cloud data into a Bird's eye view map, and applies a common image-based model, we have proposed a positional imbedded voxel feature encoding layer, which can learn the voxel representation by applying a simple convolutional layer. Experiments have shown that our proposed architecture has great potential for 3D object detection, with purely point cloud data.

# List of Publications

[1] Chenhang He and Kin-Man Lam, “Fast Vehicle Detection with Lateral Convolutional Neural Network,” Proceedings, IEEE International Conference on Acoustics, Speech, and Signal Processing, 2018.

# Acknowledgements

Firstly and foremost, I would like to express my sincere gratitude to my supervisor Prof. Kin-Man Lam for his continuous support of my MPhil study and research, for his patience, motivation and immense knowledge. His guidance helps me in all the time of research and his wisdom of life will stay with me for the rest of my life. My sincere thanks also goes to all the lab members I worked within the past two years: Chun-Cheung Lai, Hailiang Li, Cigdem Turan, and Engr Saad Shakeel Chughtai. I am thankful for their share of knowledge and thoughts at work, and the life we have enjoyed together. Moreover, I feel indebted to many people outside research who always care about me and cheer me up when I feel down. Last but not the least, I would like to thank my parents for supporting me spiritually and financially throughout the research life and my life in general.

# Table of Contents

## Table of Figures

# Chapter 1. Introduction

Object detection is one of the hot computer-vision tasks and has been widely used in many applications, such as Driver Assistance Systems (DAS) and traffic surveillance systems. In this research, we have studied different algorithms for object detection, and also investigated efficient algorithms for traffic-object detection. A traffic-object detection system requires that there is a camera in a vehicle to capture the traffic scenes, and identifies and locates the positions of traffic objects using computer-vision techniques. An intelligent traffic-object detection system can assist automatic driving, collect traffic statistics, and perform traffic scene analysis.

However, traffic-object detection remains a challenging task, because the traffic objects often appear with severe occlusion and can vary rapidly in type, size, and viewpoint. In addition, they are often perceived under bad visual conditions, such as rain, haze, snow, etc., and sometimes suffer from poor camera resolution. This is often considered as a real-world problem, so it is undesirable to use handcrafted features to perform detection. In the past few years, with the development of deep learning technology and big data, the features learned from the data are more robust to various challenging conditions. In this thesis, we apply deep-learning methods to perform traffic-object detection. The first part is on 2D detection, which takes an RGB image as an input and applies a deep neural network to extract hierarchical features. In our research, we have analyzed the semantics meaning from different layers of the network. We hypothesize that deep features have strong semantics, which are good to define the objectness but poor to localize the object. Furthermore, the earlier features contain much more fine-grained details, which can be utilized to achieve good localization. We have developed a real-time traffic-object detector, which can be used in video surveillance for vehicle detection, and a traffic-sign detector, which can perform fine-grained recognition on traffic signs. Then, in the second part of this thesis, we focus on more advanced 3D object detection using point cloud data. This area of research has been widely applied in many computer vision applications, such as autonomous driving, indoor navigation, and augmented reality. The reasons for this are that point clouds contain rich geometric information in the 3D space, which is good to characterize the shape of objects and can be efficiently collected by a commercial acquisition device, such as the laser scanner, LiDAR, as well as binocular cameras. However, it is difficult to adopt a conventional image-based model directly on the point cloud data, due to its irregular data structure and sparsity. To this end, many approaches have dealt with point clouds by first projecting them into a regular 2D domain, and then performing image-based learning techniques. More recent approaches perform the end-to-end learning on the point clouds, by applying a symmetric function to explore its global representation. These approaches show its potential power for handling the point cloud data by learning discriminative features with a moderate amount of model parameters.

This thesis is organized into 6 chapters. In Chapter 2, we will review different object-detection approaches proposed in the past decades. We start with those early approaches, which are based on sliding-window and template-matching methods. Then, we move to those more advanced, selective search approaches. Finally, we introduce recently proposed detection methods based on deep convolutional neural networks.

In Chapter 3, we present our research on fast vehicle detection. In our proposed method, we first explore using different feature layers from a deep residual network to perform vehicle detection. Experiment results show that the high-resolution features from earlier feature layers contain more structural information, which is good to achieve fine-grained localization but yields low recall rates. The low-resolution features in the deep layers contain semantically strong information, which is good for representing the objectness, but too coarse to achieve accurate localization. Therefore, we decouple the localization and objectness prediction from a single layer. Instead, we employ a lateral network that takes the features from earlier layers as its input, and outputs the localization residual. Our proposed detector can achieve fast detection at a rate of 28 frames/s, and a mean average precision (mAP) of 67.25% in the DETRAC vehicle detection benchmark.

In Chapter 4, we address a common issue in detection systems, which is that a poor detector cannot localize objects very well. In this case, the detected object may not be located at the center and may become partially seen in its bounding box. This will decrease the classification rate. To deal with this issue, we present a spatial invariant traffic-sign classifier by utilizing the idea from a spatial transformer network [21].

In Chapter 5, we explore more advanced 3D object-detection methods based on point cloud data collected from a LiDAR scanner. We review some previous approaches, which are based on view-projection, and have implemented the recently proposed VoxelNet [37] method, which can directly learn the voxel-wise feature from point clouds. We further improve VoxelNet by proposing a position embedded voxel feature encoding module and using multi-scale feature learning. The proposed method demonstrates its great potential for real-time 3D object detection.

Finally, in Chapter 6, we will conclude our work and show the plan for our future research.

# Chapter 2. Literature reviews

Object detection has been researched for decades. In the early days, people used handcrafted features to build detection templates, then applied ensemble methods to form cascaded detectors, up until the use of presently emerged deep-learning methods to learn discriminative features for detection. The performance of the detection algorithms has been improving with time. In this chapter, we review these typical object-detection approaches in chronological order.

## 2.1 Template-based detector

The early object detector is based on exhaustive window-searching and model-matching methods. One typical work is by Dalal and Triggs, where they proposed using the histogram of oriented gradients (HOG) feature to detect pedestrians. The work was further enhanced by Wang [2], by combining the local binary pattern (LBP) and HOG feature into a single feature set, and developing a detector that can detect humans with partial occlusion. The object under consideration can be posed with non-rigid transformation. Felzenszwalb et al. [3] proposed a discriminatively trained part-based model (DPM), which uses a HOG detector as the root filter to capture the coarse information from the main body of the object and captures the finer details of other deformable parts with a quadratic model to describe the deformation cost.

## Ensemble-based detector

This template-matching approach has limited performance as objects may vary in terms of scale and aspect ratio. To address these issues, the image pyramid, or ensemble multiple instance templates, has been developed. The Viola-Jones object detector [4] is one of the detection methods based on ensembling multiple templates, and has shown an impressive performance in face detection. In this method, the rectangular Haar features are used to describe different parts of a human face. These features are simple and easy to implement, but they are weak for classification, so the feature pool used is large when performing the sliding window search for an object with different scales. To select effective features, an adaptive boosting algorithm is employed to construct a strong classifier, based on an ensemble of multiple weak classifiers. These strong classifiers are then stacked in a cascaded manner. This strategy is also utilized in [5], where the regionlet descriptor was proposed to capture low-level features in a small region, and these regionlets are organized into small groups with different scales and aspect ratios. Each group of regionlets is viewed as a weak classifier. After sampling millions of weak classifiers, thousands of strong classifiers are built using real boosting, and are arranged in a cascaded manner, such that the all the information collected from the output from a given classifier are used as additional information for the next classifier in the cascade.

The regionlet-based model is normalized to fit the candidate bounding boxes varied in scale and aspect ratio. Those regionlets having greater relatedness are grouped together to describe the possible distribution of the object's grains. To detect a person, regionlets will localize the hand, which is the most informative part of a human body, at various possible locations where it may appear in a specific region.

## 2.2 Sliding window detector

Since AlexNet [6] won the ILSVRC 2012 Challenge, CNN has then become the major approach used for classification. One method of object detection is to perform an exhaustive sliding window search in an image, as shown in **Figure 1**, where the window moves from left to right, and then from top to bottom. To detect different object types at different viewing distances, windows of different sizes and aspect ratios are to be used. The cropped image patches from the sliding window are warped into a fixed size to fit the CNN, which can extract robust high dimensional features from the transformed images. Finally, the CNN-based features are fed to an SVM classifier or a regressor to predict the class probability of the corresponding regions.

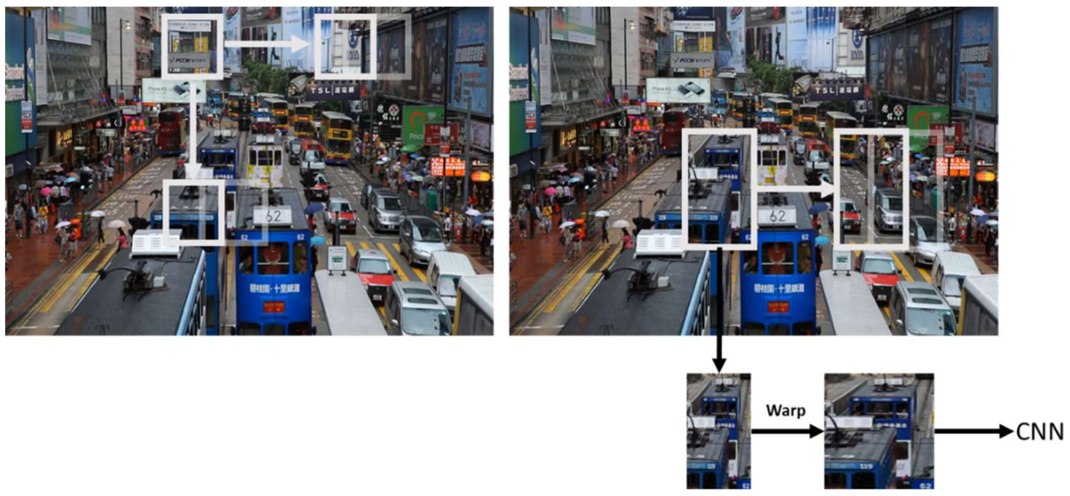


*Figure 1. The sliding window approach: Search windows with different scales and aspect ratios are used to explore the entire image.*

## 2.3 Selective search

The above approaches can achieve remarkable results, but one deficiency is that they are suffering from an exhaustive window search which is computationally intensive. To address this issue, some research focusing on generic object detection based on objectness measure [7] and selective search [8] is proposed. In [7], researchers argue that an object should have at least one of the following distinctive characteristics, the closed boundary in space, a different appearance from its surrounding and uniqueness. To standardize these characteristics, different image cues are proposed, 1) a Multi-scale Saliency (MS) cue to measure the uniqueness characteristic of objects in different scales, 2) a Color Contrast (CC) cue to measure the color discrepancy of a window to its encompassing region, 3) an Edge density (ED) cue to measure the edge compactness near the window borders, 4) a Super-pixels Straddling (SS) cue to measure the close boundary characteristic of objects by taking into account the entireness of super-pixels that lies inside the window and a complementary cue, 5) Location and Size (LS), to evaluate how likely an image window cover an object based on its location and size. These cues are then combined with a Naïve Bayes model so that the posterior of the image window objectness conditioned these cues can be estimated.

In [8], a selective window is searched to extract a subset of possible regions. These regions are supposed to find a blob-like thing by covering an object to their best. They search the object using region merging. As shown in **Figure 2**, they initially over-segment the image into many regions and aggregating the adjacent similar regions based on their texture information. Each aggregating blob is considered to be a possible object region. The aggregation is performed following a bottom-up scheme so that different scales of region proposals can be extracted

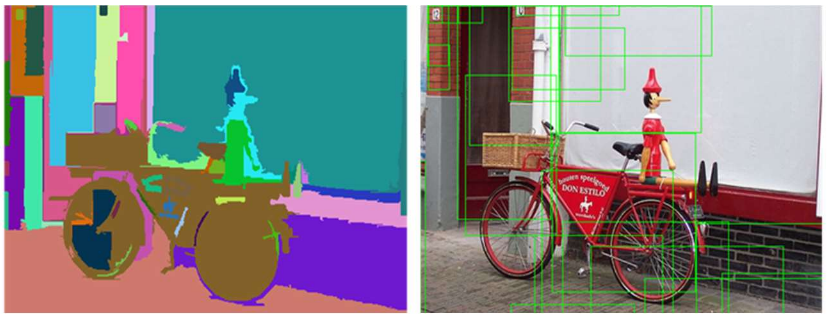


*Figure 2. Left: The super-pixel segmentation. Right: The selective window extracted by each aggregation.*

## 2.4 Region-based detector

In [9]–[11], a region proposal based detection paradigm is developed. In [9], known as a region-based convolutional neural network (R-CNN), they addressed the object detection as a classification problem. Given an image, they first extract around 2000 class-independent region proposals by using selective search. Each region is then warped to feed a fix-sized CNN input and produces a fixed-length feature vector from the CNN output. Based on these feature vectors, they train a linear SVM to classify each region into different object classes. The overview of R-CNN detection pipeline is illustrated in **Figure 3**.

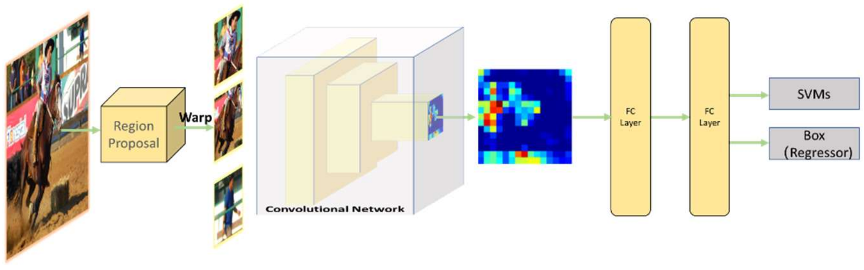


*Figure 3. The pipeline of R-CNN detector.*

The R-CNN has a multi-stage structure. To train the linear SVMs and bounding regressor in R-CNN, the features extracted from CNN are stored in a disk which is space demanding. In detection time, one proposal is fed into the pipeline at a time and the computation could not share among proposals. In [10], the authors propose a fast RCNN framework which outperforms than R-CNN in both time and accuracy. As shown in **Figure 4**, Fast R-CNN first extracts the feature map of the image by passing it through a series of convolutional and pooling layers. Given the feature map of the entire image, a fixed length feature for each proposal is then extracted with an ROI pooling layers. The ROI pooling can calculate the feature for each proposal in a fixed length. The proposal region is first projected to the convolutional feature map so that the feature within that particular region is captured. Then the region feature is sliced into a fixed grid. Within each grid cell, a max-pooling operation is applied to all the feature values in that cell. As the result, for each proposal, a fixed size ROI feature is extracted. The SVMS classifiers in fast RCNN is replaced by two sibling fully connected layers, one for generating classification distribution over a softmax function, one for regressing the bounding box coordinates. In this way, the training is performed in a single stage so that all layers' parameters are updated at a time. The computation among proposal can be shared and no more feature caching is needed in Fast R-CNN. The Fast R-CNN is 10 times faster than R-CNN in training time and 150 times faster for the inference.

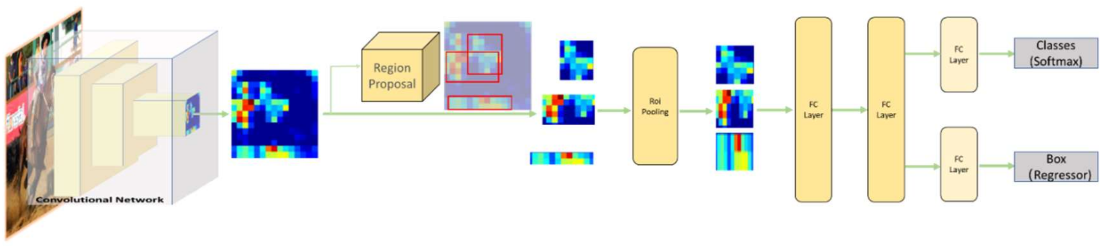


*Figure 4. The pipeline of Fast R-CNN detector.*

In Faster R-CNN[11], as shown in **Figure 5**, this framework is further enhanced in terms of both speed and accuracy, the author used a tiny neural network named region proposal network (RPN) to replace the proposal generation module, which can share the same convolutional feature in Fast R-CNN. They slide the RPN on the feature map and predicts K (number of anchors) region proposals at each position. RPN is an anchor-based detector and its loss is jointly trained with Fast R-CNN. For more details of RPN, we leave it to the section 4.3 since anchor-based methods is another detection paradigm that we want to separately discuss.

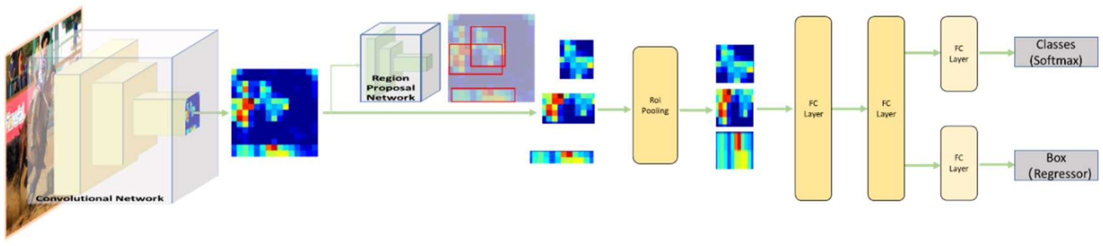


*Figure 5. The pipeline of the Faster R-CNN detector.*

## 2.5 Single-shot detector

Region-based detectors are accurate but at an extreme cost. Faster R-CNN processes 7 frames per second (7 FPS) on the PASCAL VOC 2007 test set. The major time-consuming part is the ROI pooling which processes the region candidate one at a time. Let's take another look at the sliding window detector which use different windows with different scale and aspect ratio to detect the objects with different types. The fatal weakness of the sliding window approach is to use the window as the final bounding box, which required a very large number of windows to cover most of the objects. A more efficient method is to use the predefined prior boxes as initial guesses so that the detector can predict the bounding box refer to these prior boxes, as shown in **Figure 6**.

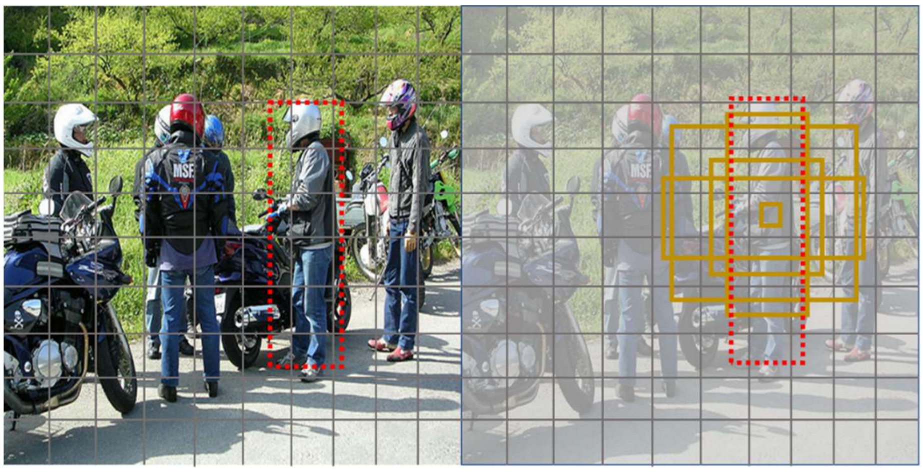


*Figure 6. Instead of searching each position of the image, one-shot detector discretize the image into a grid of cells and perform regression for each cell (left). Using pre-defined anchor boxes with different scales and aspect-ratios to increase the recall rate (right).*

This strategy also adopted in Faster R-CNN, where the region proposal network predicts the class-agnostic bounding boxes in terms of anchor offsets. However, a single-shot detector predicts both the bounding box and the probability of categories. One typical one-shot detector is YOLOs [12,13], which treat object detection tasks as a regression problem. By using a neural network, the coordinates of the bounding box, the confidence of the object contained in the box, and the probabilities of the object are directly predicted from an entire image. YOLO(v1) uses a fully connected layer to directly predict bounding boxes refer to each grid cell. In YOLO(v2), the author removed YOLO(v1)'s fully connected layer and used self-clustered anchor boxes to perform regression. In YOLO(v2), pooling layer is removed to increase the resolution of the convolutional output. The network has changed the input size from 448×448 to 416x416 so that the feature map has only one center. Considering most of the items, especially large ones are more likely to appear in the center of the image, YOLO(v2) uses a convolutional layer with the down-sampling rate of 32 so that the output size is 13 x 13 where the output feature map can encode most of the objects efficiently.

YOLO(v2) uses the anchor boxes to increase the accuracy. If considering only grid cells, the model predicts 98 boxes per image, this is how does in YOLO(V1), but with anchor boxes, each image can have more than 1000 boxes predications. The accuracy of YOLO(V2) is 69.5mAP and the recall is 81%. After the anchor boxes encoding method, the result is 69.2mAP and the recall is 88%, the high recall allows more room for the model to improve. The use of anchor boxes on the YOLO's model starts from the consideration that if the anchor's dimension is manually selected, the model will learn to adapt these anchors. Thus, they use the K-means clustering method to automatically select the best initial anchors to increase IOU scores.

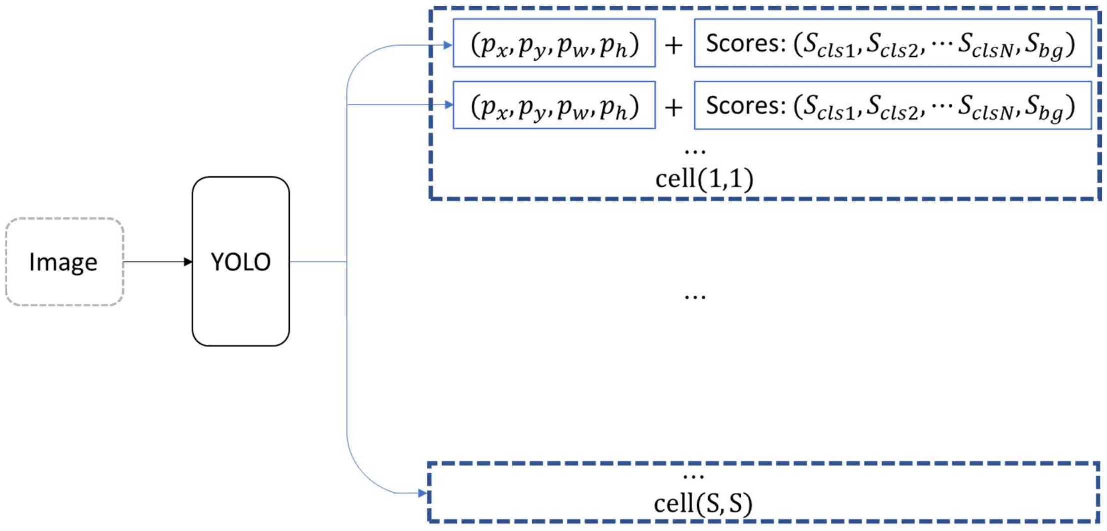

$(p_x, p_y, p_w, p_h)$ + Scores: $(S_{cls1}, S_{cls2}, \cdots S_{clsN}, S_{bg})$
$(p_x, p_y, p_w, p_h)$ + Scores: $(S_{cls1}, S_{cls2}, \cdots S_{clsN}, S_{bg})$
…
cell(1,1)
Image
YOLO
…
…
cell(S, S)

*Figure 7. YOLO perform prediction on both coordinates and categories for each cell.*

YOLO can simultaneously perform a single shot prediction on categories and locations. However, convolutional layers reduce the spatial dimension and resolution. Therefore, the above model can only detect larger targets. To solve the problem, we perform independent target detection from multiple feature maps.

The SSD [12] is another single-shot detector that uses the VGG19 network as a feature extractor (as does CNN used in Faster R-CNN). It uses multiple layers in the convolutional network to detect the objects in multi-scales. In SSD, multiple default boxes (similar to the anchor boxes) with different scales and aspect ratios are selected to generate a series of predictions. There will be some predicted boxes that match the ground truth box (with IOU bigger than 0.5), but at the same time, there are many boxes are not, and the negative boxes are far more than the positive boxes. This can cause imbalances between negative boxes and positive boxes which is difficult to converge during training. Therefore, in this paper, the non-matched boxes are first sorted according to their confidence of matching the default boxes, and then the highest ones are selected as the negative boxes, the number of selection must ensure that the proportion of negatives and positives is 3:1, this also known as hard-negative mining.

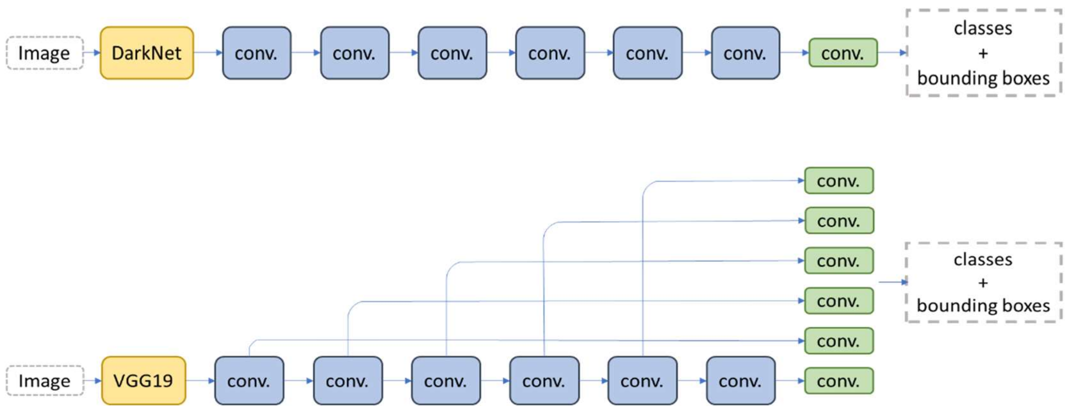


*Figure 8. Up: YOLO perform prediction on the last layer. Bottom: SSD performs detection on multi-layer so more robust to handle objects with various scales.*

## 2.6 Feature pyramid network

Detecting targets at different scales is challenging, especially the detection of small targets. The SSD completes the detection through multiple feature maps. However, the bottom layer will not be selected to perform target detection since their high-resolution feature contains insufficient semantic information which leads to a significant drop in recall. The SSD only uses the upper layer to perform target detection, and therefore has poor detection performance for small objects.

Feature Pyramid Network (FPN) [13], **Figure 9**, is a feature extractor designed to improve accuracy and speed. It replaces feature extractors in detectors such as Faster R-CNN and SSD to generate higher quality feature pyramids. The FPN consists of bottom-up and top-down paths. The bottom-up path is a common convolutional network for feature extraction and the top-down is for feature reconstruction. The spatial resolution decrease from the bottom-up. The semantic meaning in the corresponding reconstruction layer increases. Although the semantics of the reconstruction layer is strong, the position of the target is inaccurate after all up-sampling and down-sampling. Thus, adding a lateral connection between the reconstruction layer and the corresponding feature map can make position detection more accurate. **Figure 9** details the bottom-up and top-down paths. Where P2, P3, P4, and P5 are feature map pyramids for target detection.

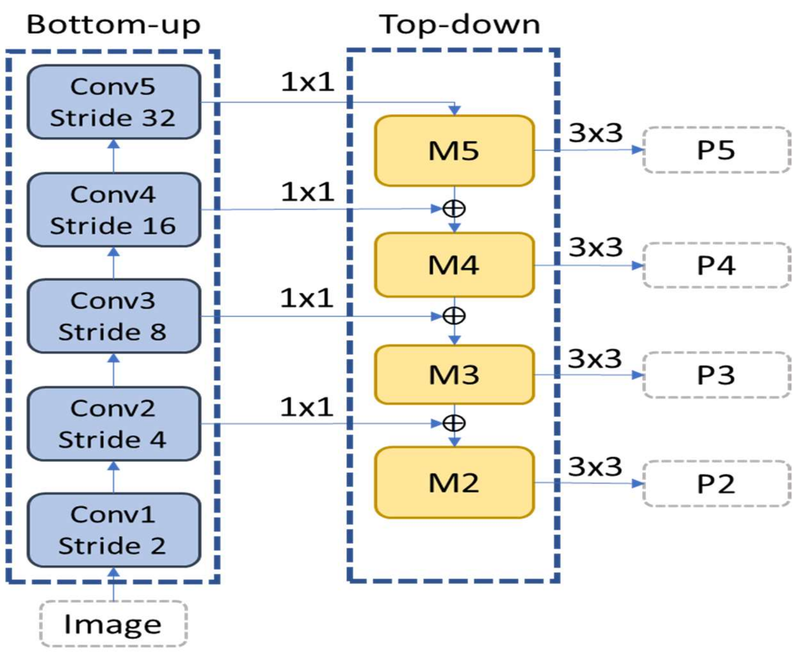


*Figure 9. The features from FPN has a better tradeoff between high semantics and fine-grained details.*

## 2.7 Focal loss and Non-maximum suppression

For most detection algorithms such as SSD and YOLO, the predicted bounding boxes can be divided into positive and negative. When the IOU between the bounding box (obtained from the anchor plus the offset) and the ground truth is greater than a threshold (usually 0.5), the bounding box is a positive sample. If the IOU is less than the lower threshold, the bounding box is a negative sample. In an input image, the proportion of the target is generally much smaller than the proportion of the background, so the predicted boxes are dominated by negative samples, which raises two problems. First, too much negative samples cause too much negative loss, so that the positive loss is submerged, which is not conducive to the goal of convergence. Second, most negative samples are not in the transition area of the foreground and background, the classification is very clear (this easy-to-classify negative is called easy negative), the background class tends to achieve higher confidence value during the training. From another point of view, if the loss of a single sample is small, the corresponding gradient in the back-propagation is also small. Thus, the easy negative sample will have a limited effect on the convergence of the model parameters. One required is the sample that has a large loss, which has a greater impact on the convergence of the parameters, which is a hard positive/negative sample.

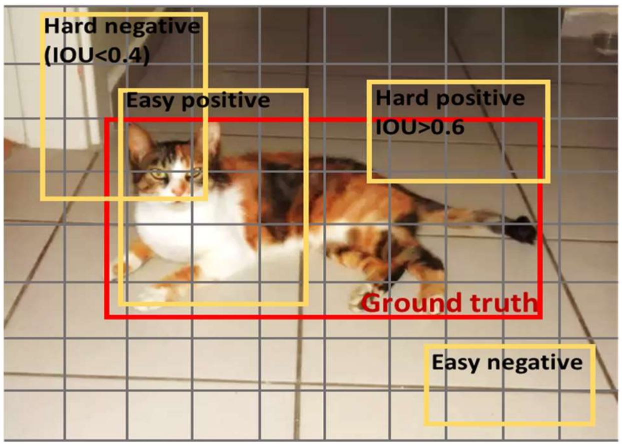


*Figure 10. Different sample contributes different loss during the training.*

As mentioned previously, SSD uses online hard negative mining (OHEM), which is a method of screening easy samples. It sorts the loss and selects the largest loss example to train. This method ensures that the training areas are all hard examples and removes all the easy examples, which is unable to further improve the training accuracy from easy examples. In [14], the author proposes a focal loss, that reframes the stand cross entropy loss function in another formula:

$$\mathrm{FL}(p_t) = -\alpha(1-p_t)^{\gamma}log(p_t) \tag{2.1}$$

where $\mathrm{p_t}$ is the classification probability of different categories, $\gamma$ is a value greater than 0, $\alpha$ is a decimal between [0,1], $\alpha$ and $\gamma$ are at fixed values and do not participate in training. From the expression one can determine whether it is a foreground class or a background class, the larger $\mathrm{p_t}$ is, the smaller the weight $(1 - p_t)^{\gamma}$ is. That is, easy example can be suppressed by a small weight. $\alpha$ is a scaling factor that used to adjust the contribution of positive and negative examples.

Typically, the detector will make repeated detections for the same target, we use non-maximal suppression (NMS) to removes duplicated detections with lower confidence. Formally, we first rank the detections in a descending sequence according to its confidence. Then, in the first iteration, we select the one with the highest confidence and calculate its intersection over union (IoU) with the others. If any other predictions have the IoU greater than a specific threshold, we remove it from the sequence In the second iteration, we select the prediction with the second highest confidence and apply the removal as does in the first times. We keep this procedure until no predictions can be removed.

## 2.8 Conclusion

This chapter serves as a review of object detection, in which we first introduced the early object detection approaches that use handcraft feature as region descriptor, and some well-known machine learning techniques to classify these regions based on the handcraft features. Later, we reviewed the approaches that apply deep learning, in which two detection paradigms, the region based detector and single shot detector have been introduced. The drawbacks of them are also discussed. In the following chapters, the proposed methods based on the convolutional neural network will be presented.

# Chapter 3. Fast vehicle detector

Vehicle detection is one of the hot computer-vision tasks and has been widely used in many applications, such as Driver Assistance Systems (DAS) and traffic surveillance systems. An intelligent vehicle detection system can assist in automatic driving, collect traffic statistics, and perform traffic scene analysis. However, vehicle detection still remains a challenging task, because vehicles often appear with severe occlusion and are varied in type, size, and viewpoint. In addition, they are often perceived under bad scene conditions, such as rain, haze, snow, etc., and low-light conditions. In this chapter, we introduce our proposed vehicle detector for traffic surveillance.

## 3.1 Overall detection pipeline

The proposed LateralCNN is shown in **Figure 11**, where a deep residual network [15] is used to extract rich feature representations from a given image. Two convolutional layers are employed to the last feature layer to generate 6 feature maps (4 localization maps and 2 objectness maps), which are used to coarsely regress the bounding-box coordinates and the corresponding objectness scores. The features from earlier layers are passed through a lateral network with 1x1 reduction. The lateral network can produce highly resolved feature residuals on the localization maps (red) The final output bounding box is determined by applying non-maximum suppression.

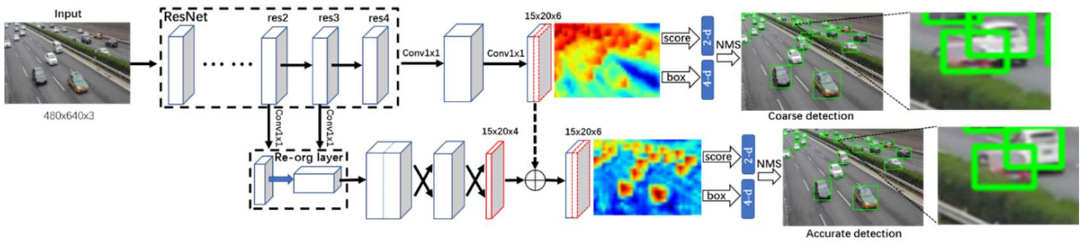
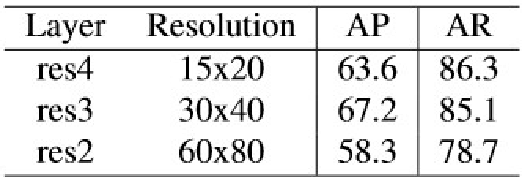


*Figure 11. The upper network performs the coarse detection, while the lower is the lateral network, which generates the fine-grained localization residuals.*

## 3.2 Objectness vs. localization

Features in different scales have different semantical meanings in object detection. To explore which feature layer is efficient for detecting vehicles at a particular size, we use the different residual blocks from a pre-trained 101-layer ResNet to perform detection. We refer to these blocks as res2, res3, and res4 features, and conduct exploratory experiments on the DETRAC [16] dataset. We evaluate the detection performance in terms of the average precision (AP) and the average recall (AR), with IoU threshold of 0.5. From **Table 1**, we observe that the AP increases from res4 to res3, but drops from res3 to res2, and res2 yields an extremely low recall rate. Based on this, we infer that the high-resolution features from early layers, which contain low-level structural information, is less sensitive to semantics. In contrast, the low-resolution deep layers provide a more high-level abstraction of the object, which is semantically strong to represent the objectness but too coarse to give accurate localization, especially for small objects. We argue that there should exist a trade-off between a better localization and a higher recall, so we decouple the objectness and localization prediction from a single layer. As a result, we select the semantically strong res4 feature and employed two convolutional layers to regress the object. This fully convolutional structure maintains relatively higher recall, which allows more room for our model to achieve better accuracy. In the later **section 3.4**, we will show how to utilize the early layers to enhance the localization accuracy.



| Layer | Resolution | AP | AR |
|---|---|---|---|
| res4 | 15x20 | 63.6 | 86.3 |
| res3 | 30x40 | 67.2 | 85.1 |
| res2 | 60x80 | 58.3 | 78.7 |

*Table 1. Average precision (AP) and average recall (AR) are evaluated when using different feature layers to perform detection. The resolution of the input image is 480x640.*

## 3.3 Multi-cell prediction

Similar to YOLO, we sample the image space into a grid, and for each grid cell, the height, width, center offsets relative to the cell, and objectness scores of the bounding box are predicted. However, instead of focusing on only one grid cell where the center of the object lies, as it does in YOLO, we encode the object's bounding-box information into multiple grid cells that overlap with this object. If multiple objects are overlapped with one cell, the object with the highest overlapping area is predicted by that cell. Using multiple cells to perform prediction can largely increase the robustness to object occlusions. **Figure 12** illustrates a case where the two objects are highly overlapped so that their bounding-box centers lie in the same cell. In this case, using the single-cell prediction can detect one object only, while the multi-cell prediction can detect the occluded object by another cell where the object partially lies.

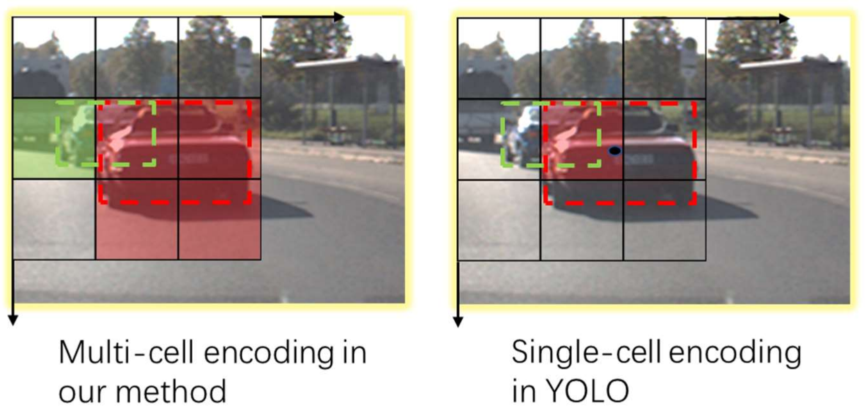

Multi-cell encoding in our method
Single-cell encoding in YOLO

*Figure 12. Two target encoding methods, the colored cells are responsible for predicting the same colored bounding-box during the inference.*

## 3.4 Lateral residual network

Semantically strong features from the deep layers can achieve relatively higher recall, but they are too coarse to detect an object at a small scale. Compared to the deep layers, the high-resolution features from earlier layers can provide much more detailed information about tiny objects and fine-grained representations. Thus, we incorporate these features into our model by passing it through a lateral network. In the lateral network, 1x1 reduction layer is first applied, then a re-organizing layer, which can transform the tensor with the shape of $(height, width, depth)$ into the tensor with the shape of $(\mathrm{height/scale, width/scale, depth \times scale} \times scale)$is employed. The transformed high-resolution features have the equal scale of the low-resolution features, Thus, we can concatenate these features and apply two fully connected layers to predict the localization residual efficiently. With the help of the lateral network, the model can finely resolve the localization map efficiently.

The localization predictions in each cell contain the information of the bounding box coordinates, in terms of the center offset relative to the grid cell, its height, and width.

For simplicity, we use $p_i^{loc}$ to denote the 4-d localization prediction in each grid cell, where $i = 1 \dots N$ is the index of the grid cell. The objectness prediction in each cell is represented by a 2-d foreground-background probability distribution, which is simply denoted by $p_i^{obj}$. Following the multi-cell encoding scheme, we can calculate the localization ground truth, which is denoted by $t_i^{loc}$. We use $c_i$ to indicate the class encoded by each grid cell, which is set to 1 if the cell is assigned to detect the object and set to 0 if the background. For the lateral network, we denote its output by $\delta \mathrm{p}_\mathrm{i}^{\mathrm{loc}}$, which represents the localization residual.

Thus, the network can be trained in two stages. The first stage is to train the coarse detector, with the following loss function:

$$\mathcal{L}_1 = L_{loc}(p, t) + \alpha L_{obj}(p, c). \tag{3.1}$$

The second stage is to train the lateral network to predict the localization residual, with the following loss function:

$$\mathcal{L}_2 = L_{loc}(p + \delta p, t). \tag{3.2}$$

Following Faster RCNN [11], we efficiently implement the two-staged alternative training with an approximate joint training, so that the network can be trained in an end-to-end manner, with the total loss as follows,

$$\mathcal{L} = \mathcal{L}_1 + \beta \mathcal{L}_2. \tag{3.3}$$

$L_{loc}$ is an $\ell_1$ loss, which is defined as:

$$L_{\mathrm{loc}}(\mathrm{p}, \mathrm{t}) = \frac{1}{N} \sum_{i=1}^{N} c_i \, |p_i^{loc} - t_i^{loc}|. \tag{3.4}$$

To solve the foreground-background class-imbalance issue in our detector, a more dedicated focal loss [14] is used with Sigmoid normalization σ. Following [14], $L_{obj}$ can be defined as:

$$L_{obj}(p,c) = -\frac{1}{N}\sum_{i=1}^{N} \omega\,(1-p_i^*)^{\gamma} log(p_i^*), \tag{3.5}$$

Where

$$p_i^* = \begin{cases} \sigma\,(p_i^{obj})\; if\; c_i = 1 \\ 1-\sigma\,(p_i^{obj})\; otherwise. \end{cases} \tag{3.6}$$

α is set to 0.1, which is for balancing the loss contribution between the localization and objectness predictions. We carefully select this hyper-parameter following the inverse-of-gradient rule in multi-task learning.β is set to 0.05, which is determined by cross-validation. ω and γ are the hyper-parameters in the focal loss equation. We empirically set them to 0.5 and 2, respectively, which are the optimized values mentioned in [14].

## 3.5 Experiments

We evaluated the proposed LateralCNN on DETRAC [16], which is a traffic vehicle detection benchmark with 140K captured frames from traffic surveillance. The numbers of training and testing samples are 84K and 56K, respectively, and each sample has a resolution of 960x540. The test set is categorized into different challenge levels, which consider different object sizes and degrees of occlusion, and different scene conditions, such as sunny, cloudy, rainy and night. To train our detector, we resized the training images into the resolution of 640x480 and added jitter to the images. We initialized the feature extractor with a pre-trained 101-layer residual network and initialized the other parts of the network with a zero-mean uniform distribution. Batch normalization is applied to all the convolutional layers. The initial learning rate is 0.001 for the first 100K iterations and then dropped by 50% after every 30K iterations. The network was totally trained for 200K iterations with a mini-batch size of 16, using stochastic gradient descent.

We split the samples into two subsets, 75% for training and 25% for validation. Then, we performed the ablation experiments to evaluate the effectiveness of different components. The first three entries in **Table 2** show the results of the coarse detector, with and without batch normalization (BN), and with either cross-entropy loss (CE) or focal loss (FL). The last two entries show the results of using the lateral network to perform fine-grained detection with different early layers. We found that using batch normalization and focal loss can improve the detection accuracy by 1%~2%, and with the lateral network, the detector can achieve nearly 9% improvement. Although the lateral network introduces the additional runtime, the detection can still achieve 28 frames/s with a moderated GPU.

| Early layer(s) | Use BN? | Loss | mAP | FPS |
|---|---|---|---|---|
| - | no | CE | 68.5 | 35 |
| - | yes | CE | 70.3 | 35 |
| - | yes | FL | 72.2 | 35 |
| res3 | yes | FL | 80.5 | 29 |
| res3+res2 | yes | FL | 81.6 | 28 |

*Table 2. Ablation experiments using different configurations, with the mean average precisions (mAP) and frames-per-second (FPS) reported.*

We also compare LateralCNN with other state-of-the-art methods on the DETRAC test set. We use the precision-recall evaluation protocol to evaluate the performance in terms of average precision, and detection speed, in terms of frames-per-second (FPS). More specifically, we generate different pairs of precision and recall value by changing the score threshold of our detector and plot a Precision-recall (PR) curve. The average precision can then be derived by calculating the area under this PR curve. As shown in **Table 3**, our detector thoroughly outperforms most of the state-of-the-art approaches, like Faster RCNN [11], CompACT [17] and achieves similar accuracy to EB [18] but with a lesser inference time. The reason behind this is that our detector does not rely on the proposal generation. We directly regress the object bounding box from the output.

| Method | Overall | Easy | Medium | Hard | Sunny | Cloudy | Rainy | Night | FPS | GPU |
|---|---|---|---|---|---|---|---|---|---|---|
| RCNN[3] | 48.95 | 59.31 | 54.06 | 39.47 | 59.73 | 39.32 | 39.06 | 67.52 | 0.10 | Tesla K40 |
| Faster RCNN[5] | 58.45 | 82.75 | 63.05 | 44.25 | 66.29 | 69.85 | 45.16 | 62.34 | 11.11 | TitanX |
| CompACT[11] | 53.23 | 64.84 | 58.70 | 43.16 | 63.23 | 46.37 | 44.21 | 71.16 | 0.22 | Tesla K40 |
| EB[12] | **67.96** | **89.65** | 73.12 | **53.64** | **72.42** | 73.93 | 53.40 | **83.73** | 10.00 | TitanX |
| LateralCNN | 67.25 | 89.56 | **73.59** | 51.61 | 69.11 | **74.36** | **55.77** | 78.66 | **28.46** | 1080Ti |

*Table 3. The detection performances on DETRAC, under different challenges and weather conditions.*

**Figure 13** shows the precision-recall curves of our method and the other methods. Note that our detector can achieve better localization accuracy, but a lower recall rate compared to EB. This is because our detector has a regression-based architecture.

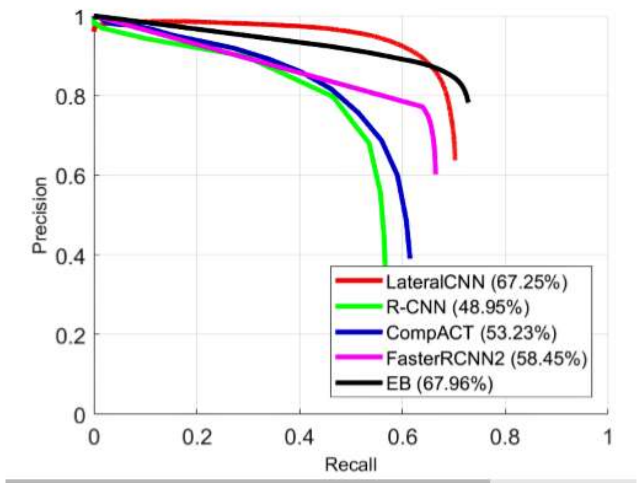


*Figure 13. The precision-recall curves of our proposed method and different state-of-the-art methods tested on the DETRAC test set.*

**Figure 14** shows the qualitative results of our method on the DETRAC test set. Most of the vehicles are at different sizes and viewpoints, and under severe occlusion, but our method can detect them successfully.

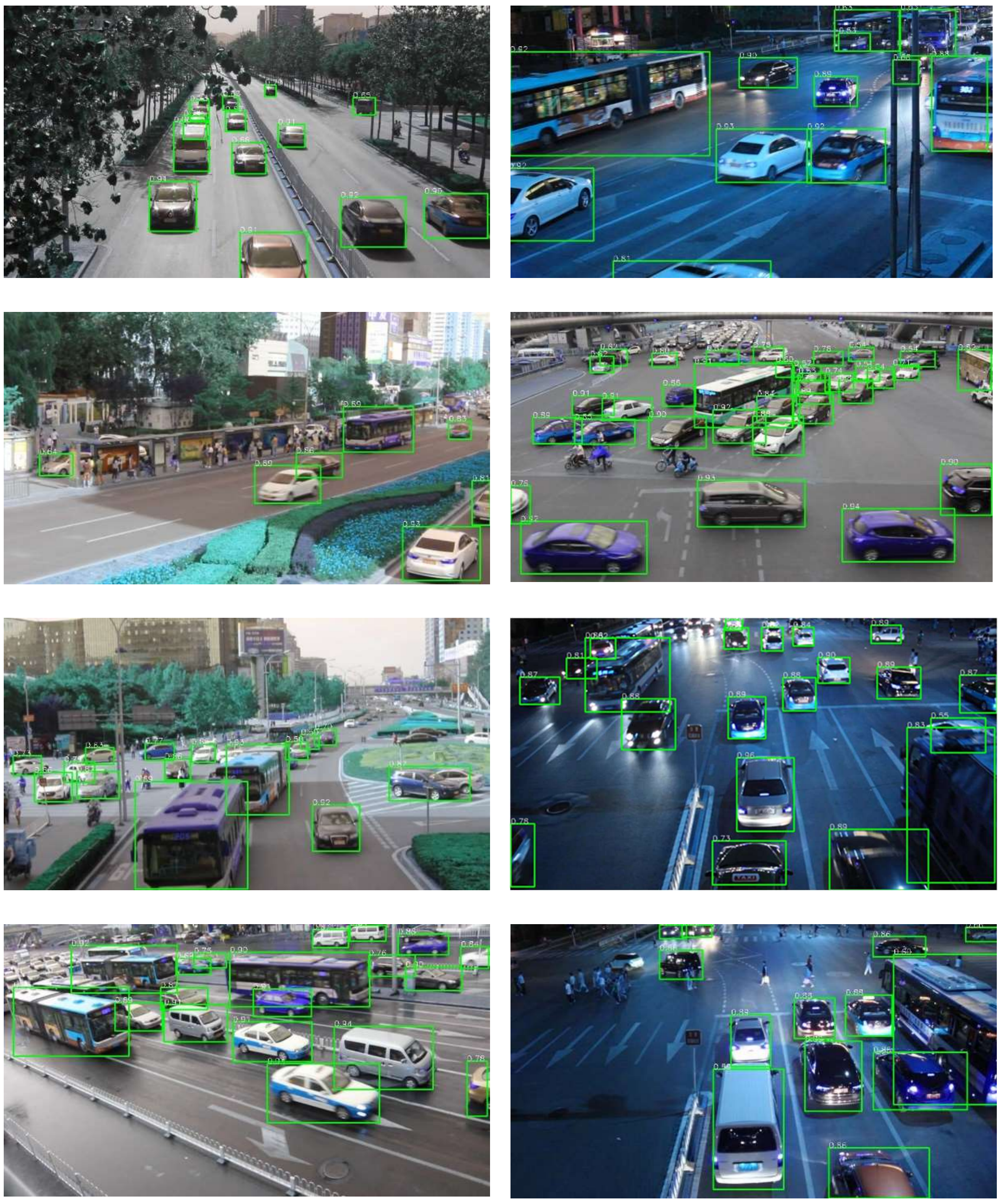


*Figure 14. Qualitative results of the proposed method on the DETRAC test set.*

## 3.6 Conclusion

In this chapter, we have presented a fast vehicle detection approach, which detects objects from coarse to fine. Our coarse detector can achieve relatively high recall, by using semantically strong features. A lateral network is employed to fuse the high-resolution features from earlier layers to achieve fine-grained localization. In our experiments, we show that, with this lateral network, our detector can achieve better detection performance, with an average precision of 67.25% on the DETRAC benchmark. Furthermore, the detector is suitable for real-time applications, which achieves a detection rate of 28 frames/s.

# Chapter 4. Traffic-sign Recognizer

As a common symbol in our daily life, traffic signs provide drivers with the necessary traffic guidance information. Traffic signs are often placed on the top and sides of the road, requiring the driver to shift his gaze to search and identify, which easily distracts the driver. The driver who is unfamiliar with the road conditions will easily lead to traffic accidents, when suddenly seeing a decelerating sign during driving. The effective extraction and identification of road signs can help drivers to obtain the road information on time, which can prevent from accidents. In this chapter, we introduce a robust traffic-sign recognizer, which can be used to classify those badly localized traffic signs.

Badly localized object means that the predicted bounding box is not able to encircle the object tightly or only part of the object appears in the bounding box. In this case, as shown in **Figure 15**, we typically enlarge the detected region with some margin, and then perform classification. Since the traffic sign is not placed well in a search region, which dramatically decreases the detection and classification rate.

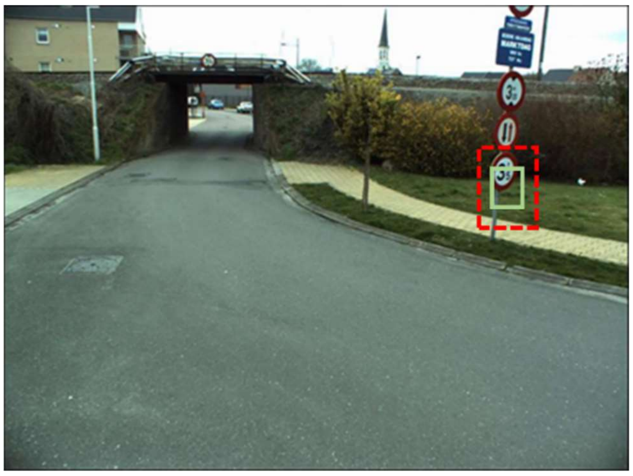


*Figure 15. The effect of a badly located road sign: Green: the bounding box predicted by a poor detector. Red: the enlarged region to be fed into a classifier.*

Using a single convolutional neural network (CNN) to locate road signs may not be a good idea, because the network is insensitive to color and spatial variations. Typically, using image pre-processing techniques, like histogram equalization, whitening, and data augmentation (e.g. flipping, rotation, sheering, etc.), can alleviate the harm, but this is still not enough to achieve human-level accuracy. Furthermore, this will increase the cost for training and introduce extra processing before feeding data to the network. In our proposed algorithm, we have created a translated traffic classification dataset, using the ground-truth bounding-box coordinates in GTSDB [19]. Unlike the normal traffic-sign classification datasets, this dataset is more challenging as the traffic signs are in random positions and some unwanted objects may also appear to intervene them. Figure 16 shows some traffic-sign images from the database. In our method, we use a color attention module and a constraint spatial transformer layer to help our deep neural network to learn color and spatial variations. We demonstrate in our experiments that using this layer, the classifier can achieve better performance without performing any data augmentation.

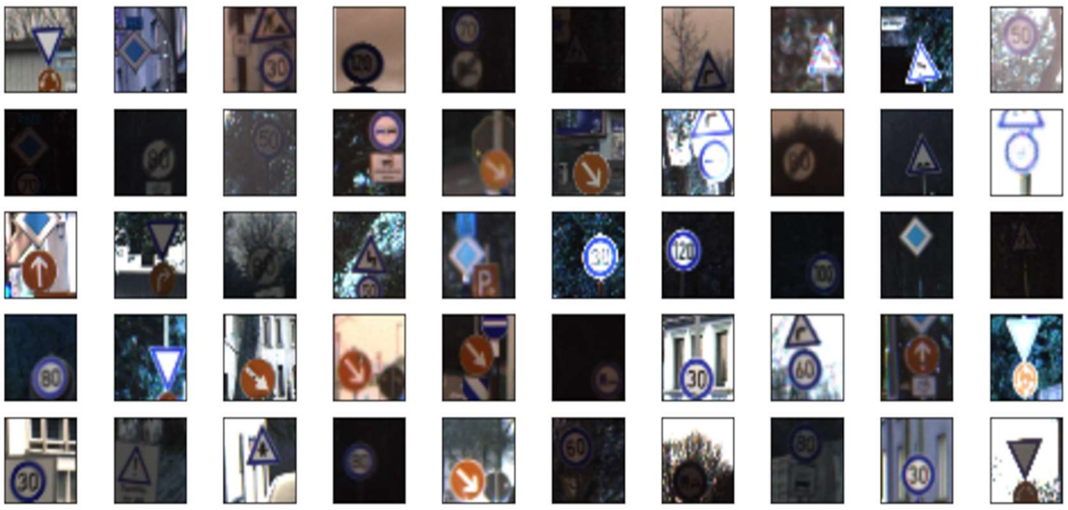


*Figure 16. The translated traffic-sign dataset. Each traffic sign is placed in a random position in each region, and unwanted traffic signs are also involved.*

## 4.1 Model

In [20], it was shown that CNN can achieve different classification results when different color spaces are used. In order to find the optimized color space, we use a channel attention module to impose different weights on different color channels. Meanwhile, we use a spatial transformer network [21] to justify the target location by constraining its learnable affine transformation matrix. We build our classifier with a convolutional neural network, whose configuration is illustrated in **Table 4**.

| Layer | Kernel Size | No. of Channels |
|---|---|---|
| Channel attention | - | 10 |
| Spatial transform | - | 10 |
| Convolution | 3x3 | 16 |
| Max Pooling | - | 16 |
| Convolution | 3x3 | 64 |
| Max Pooling | - | 64 |
| Convolution | 3x3 | 128 |
| Convolution | 3x3 | 64 |
| Max Pooling | - | 64 |
| Fully Conn | - | 1024 |
| Fully Conn | - | 1024 |
| Fully Conn | - | 43 |

*Table 4. The network configuration of a 43-class traffic-sign classifier.*

## 4.2 Channel attention module

Channel information exhibits more discriminative power in traffic-sign recognition. We first apply a 1x1 convolutional kernel to enrich the features by extending a 3-channel RGB image to a multi-channel feature representation. Then, we apply a global pooling to extract the channel-wise feature from these features. After that, a fully connected layer is employed to extract the attention weight from the channel-wise features. The weight will be imposed on the different channels, so that some feature channels will be further activated. For example, the red channel of the "stop" sign and the blue channel of the "speed limit" sign should be paid more attention with. The attention module is differentiable, so it can be inserted into anywhere of the network. We use it as the first layer, because the RGB channels from the input image contain rich information for us to determine the channel weights.

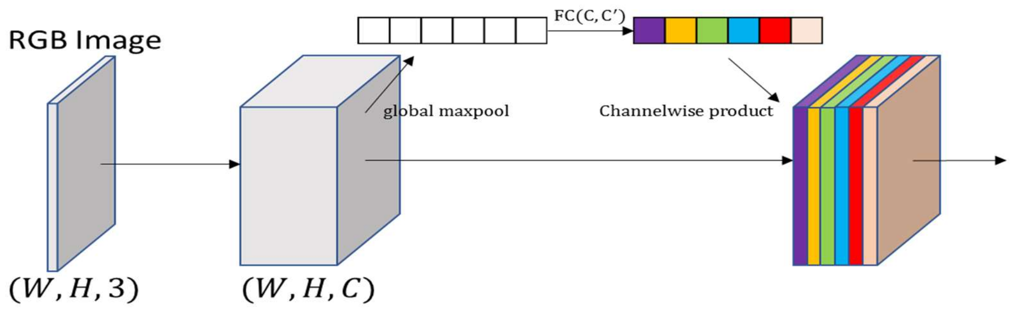


*Figure 17. The channel attention module.*

## 4.3 Spatial transformer layer

The spatial transformer layer [20] can be used to improve the classification accuracy and locate the traffic-sign objects with no supervision. The spatial transformer layer can perform a geometric transformation on the input coordinates by applying a learnable affine transformation matrix. Figure 18 shows the layer, where $T_\theta(G)$

represents the transformation matrix. The spatial transformer layer has been proven to be differentiable. It can be inserted into any position in the network.

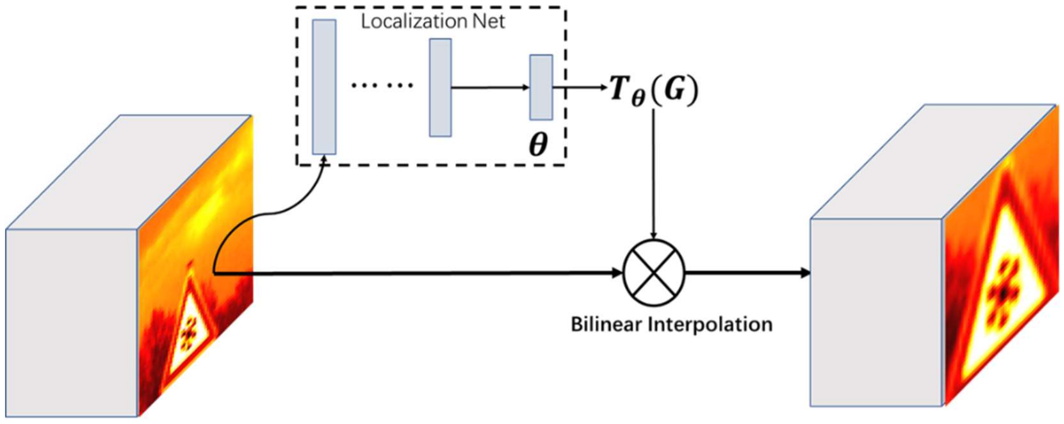


*Figure 18. The spatial Transformer Unit.*

The input feature map U is fed into a localization network and the transformation parameters $\theta$ is predicted. A transformed grid $T_\theta(G)$ is generated to sample the pixels from U. V is the output feature map sampled from $T_\theta(G)$ using bilinear interpolation. Table 5 shows the structure of the localization network.

| Layer | Kernel Size | No. Channels |
|---|---|---|
| Convolution | 7x7 | 16 |
| Max Pooling | - | 16 |
| Convolution | 5x5 | 32 |
| Max Pooling | - | 32 |
| Convolution | 3x3 | 64 |
| Max Pooling | - | 64 |
| Fully Conn + ReLu | - | 128 |
| Fully Conn + ReLu | - | 64 |
| Fully Conn | - | 3 |

*Table 5. The layer information of the localization network.*

The last layer of the localization network produces a 3-D vector $\theta = (\theta_1, \theta_2, \theta_3)$. Then, the following affine transformation matrix is constructed.

$$A_\theta = \begin{bmatrix} \theta_1 & 0 & \theta_2 \\ 0 & \theta_1 & \theta_3 \end{bmatrix} \tag{4.1}$$

$A_\theta$ is a constraint affine transformation matrix parameterized by $\theta$, which only allows isotropic scaling and translation. $\theta_1$ controls the scaling and $(\theta_2, \theta_3)$ control the translation. The affine transformation matrix is applied to the coordinates of the output grid, producing the transformed grid's coordinates.

$$\begin{pmatrix} x_{in} \\ y_{in} \end{pmatrix} = \mathrm{T}_\theta(G) = A \begin{pmatrix} x_{out} \\ y_{out} \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_1 & 0 & \theta_2 \\ 0 & \theta_1 & \theta_3 \end{bmatrix} \begin{pmatrix} x_{out} \\ y_{out} \\ 1 \end{pmatrix} \tag{4.2}$$

where $(x_{in}, y_{in})^T$ is the input coordinates, which represent the sample points of the input feature map, and $(x_{out}, y_{out})^T$ is the output coordinates, which holds the pixel position of output feature map. The output coordinates are used to sample the pixel from the input feature map $I_{in}$ using bilinear interpolation, to generate the output feature map $I_{out}$, as follows:

$$I_{out} = \sum_{x=0}^{X} \sum_{y=0}^{Y} I_{in}(x, y) \times \max(1 - |x_{in} - x|, 0) \times \max(1 - |y_{in} - y|, 0) \tag{4.3}$$

Note that, spatial transformer module is differentiable and can be inserted into anywhere of the network due to the differentiability of the bilinear interpolation, the gradients for each inputs of the spatial transformer module can be derived as the following equations:

$$\frac{\partial I_{out}}{\partial x_{in}} = \sum_{x=0}^{X} \sum_{y=0}^{Y} I_{in}(x, y) \times \max(1 - |y_{in} - y|, 0) \begin{cases} 1 & if\ x \geq x_{in} \\ 0 & \text{if } |x - x_{in}| \geq 1 \\ -1 & if\ x < x_{in} \end{cases} \tag{4.4}$$

$$\frac{\partial I_{out}}{\partial y_{in}} = \sum_{x=0}^{X}\sum_{y=0}^{Y} I_{in}(x,y) \times \max(1 - |x_{in} - x|, 0) \begin{cases} 1 & if\ y \geq y_{in} \\ 0 & if\ \ |y - y_{in}| \geq 1 \\ -1 & if\ y < y_{in} \end{cases} \tag{4.5}$$

$$\frac{\partial I_{out}}{\partial I_{in}(x,y)} = \sum_{x=0}^{X}\sum_{y=0}^{Y} \max(1 - |x_{in} - x|, 0) \times \max(1 - |y_{in} - y|, 0) \tag{4.6}$$

## 4.4 Model analysis

We conducted control experiments by switching off the spatial transform module on the translated dataset, with a enlarging factor of 1.5. We visualize the feature map before the second convolutional layer. As shown in **Figure 19**, the spatial transformer unit can accurately locate the wanted object very well. We have also evaluated the classification accuracy by training our classifier for 2000 epochs. The classification accuracy over the epochs is shown in **Figure 20**.

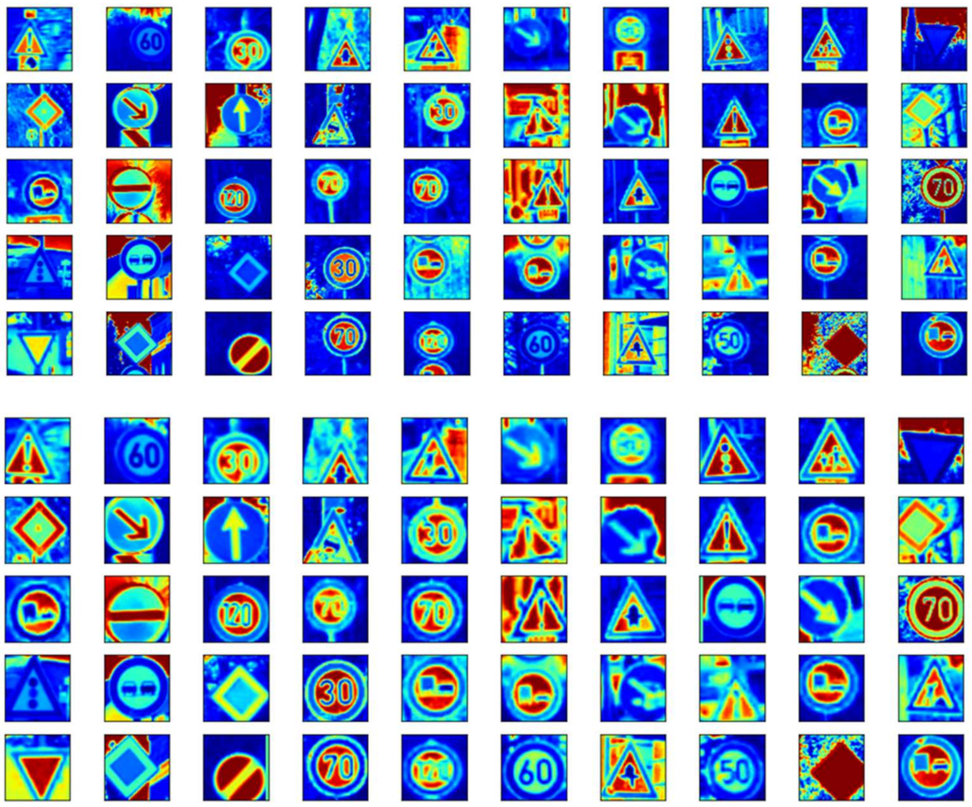


*Figure 19. Up: The feature maps without using the spatial transformer layer.*

*Bottom: The feature maps using the spatial transformer layer.*

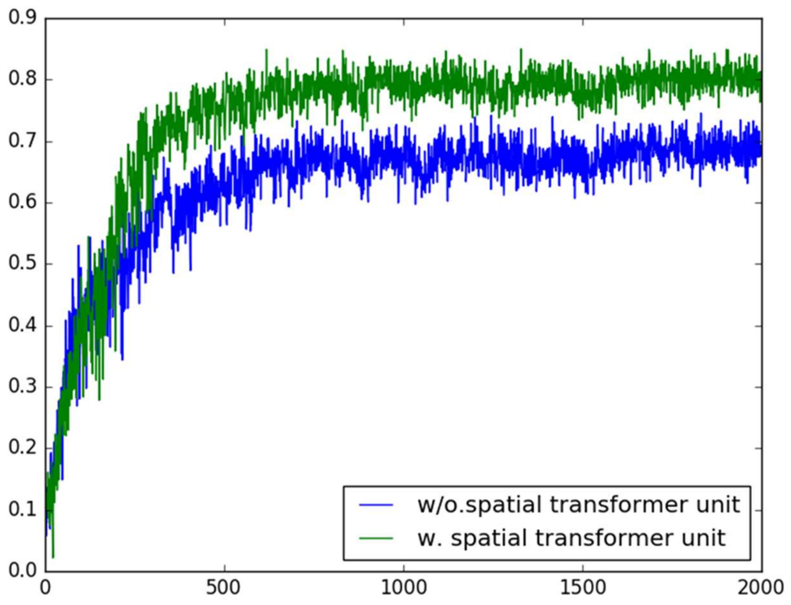


*Figure 20. The classification accuracy on the translated GTSRB.*

We also perform the ablation study by switching off the channel attention module and the spatial transformer unit separately and together. **Table 6** shows how the different modules impact the overall performance on the translated GTSDB dataset.

| Channel Attention | Spatial Transformer | Accuracy |
|---|---|---|
| ✗ | ✗ | 92.53% |
| ✗ | ✓ | 98.72% |
| ✓ | ✓ | 99.21% |

*Table 6. Evaluation of the spatial transformer (ST) layer and channel attention module in terms of classification accuracy on translated GTSDB.*

## 4.5 VIP Cup Competition

We formed a team to participate in the IEEE SPS Video and Image Processing (VIP) Cup competition last year. The team was composed of two undergraduates from our Department, and the other two undergraduates from UTS, and one postgraduate. I was the postgraduate member. My responsibility was to guide and instruct the undergraduate team members to develop algorithms for the competition. The VIP Cup competition was organized by IEEE Signal Processing Society, and the challenging topic was traffic-sign detection, under bad weather conditions. In the competition, we were required to detect and recognize traffic signs from video sequences, under different severe conditions, such as rain, haze, snow, blur, and darkness. We tackled this challenge, by proposing a two-stage detector. In the first stage, we applied a single-shot network to detect class-agnostic traffic signs' region, which was fast and able to achieve high recall. The network is a variant of YOLO detector which output layer is modified to generate two-class (foreground and background) probability.

In the second stage, we applied our proposed traffic-sign recognizer to classify and re-localize these candidate regions. Considering the traffic signs' classes are highly imbalanced across the dataset, we train our region detector and classifier separately to ensure the system achieves high recall. Figures 21 shows the precision-recall curves of the class-agnostic region detector and the final classifier on the VIP Cup's benchmark.

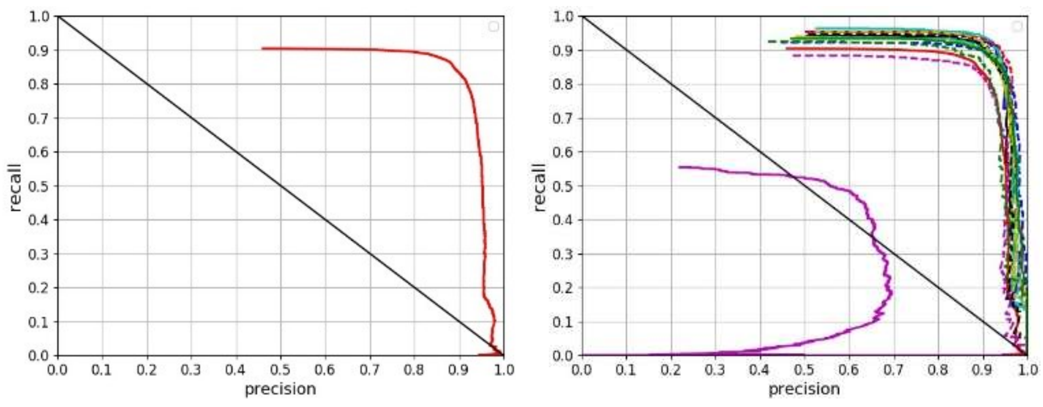


*Figure 21. The precision-recall curves of class-agnostic and class-specific detection tasks.*

Our proposed model won the first runner-up in the final competition. There were more than 250 teams from 28 countries participated in the competition. For more details, please refer to https://ghassanalregib.com/vip-cup/.

## 4.6 Conclusion

In this chapter, we have introduced a robust traffic-sign classifier based on a convolutional neural network. The proposed classifier contains a spatial transformer module, which can adjust the position of those badly located traffic signs, and a channel-attention module, which can boost the performance in recognizing traffic-sign objects with salient colors. The proposed classifier shows its effectiveness in real-world environments.

# Chapter 5. 3D Object Detector

Existing object detection frameworks, such as Faster-RCNN, SSD, YOLO, etc., have demonstrated remarkable performance in 2D object detection. However, in many applications like robotics, autonomous driving system, 2D bounding-box is insufficient and provides ambiguities when reasoning about 3D pose of an object, occlusion and depth. Therefore high-quality 3D bounding box is required. In 3D object detection, it is necessary to estimate the 3D statistics for each object, by leveraging different sensor modalities. Apart from RGB camera, some stereo devices like structured light sensors, LiDAR and binocular camera are typically used.

In this chapter, we will first present a 3D object detection approach based on point cloud data. Point cloud data is a set of points that carry the rich amount of information. Each point is presented as a 4D vector in terms of the *X*, *Y*, and *Z* coordinates and the intensity value (reflectance of the light). We first describe the coordinate system and the bounding box parameters used in the representation. A point cloud is captured by using a LiDAR device, which is mounted on the top of a vehicle. The X, Y, and Z coordinates are shown in **Figure 22**, where the X, Y, and Z axes are spanned along the distance, breadth, and height, respectively, of the searching space of the LiDAR sensor. Usually, $(\mathrm{x}, \mathrm{y}, \mathrm{z}, \mathrm{h}, \mathrm{w}, \mathrm{l}, \theta)$ is used to define a 3D object bounding box, where $\mathrm{h}, \mathrm{w},$ and $\mathrm{l}$ are the size of the bounding box along Z, Y, and X axes to represent the height, width, and length, respectively, of the bounding box; $(\mathrm{x}, \mathrm{y}, \mathrm{z})$ is the location of the centroid of the bottom plane; and $\theta \in [-180°, 180°]$ is the yaw angle defined on the X-Y plane, which represents the pose of the bounding box.

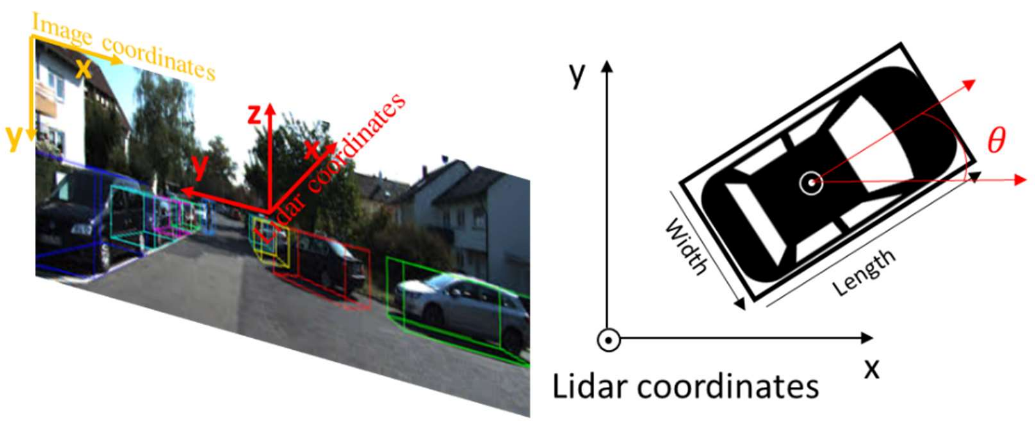
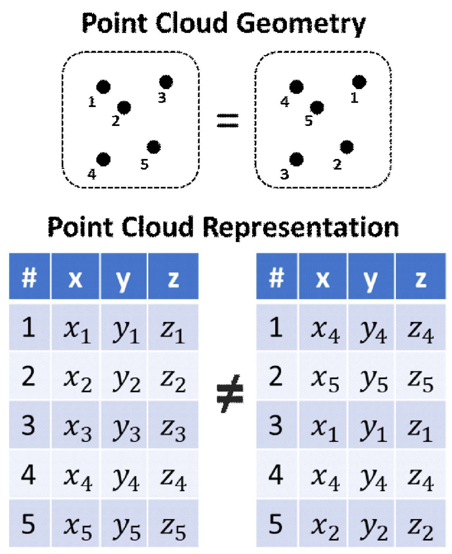


*Figure 22. The coordinate system from the camera's and lidar's perspectives.*

However, it is hard to apply a deep neural network to learn and analyze geometric data, like point clouds. The reasons for this are, first, a point cloud is typically sparse. Second, the point cloud data is a set of point coordinates, which has irregular data structure. The geometry representation of point cloud data is invariant to its data permutations. As shown in **Figure 23**, the geometry representation is unchanged after re-arranging the order of its data entries. In the past few years, methods based on 2D projection and voxel-wise feature learning have been proposed.



| # | x | y | z |
|---|---|---|---|
| 1 | $x_1$ | $y_1$ | $z_1$ |
| 2 | $x_2$ | $y_2$ | $z_2$ |
| 3 | $x_3$ | $y_3$ | $z_3$ |
| 4 | $x_4$ | $y_4$ | $z_4$ |
| 5 | $x_5$ | $y_5$ | $z_5$ |

| # | x | y | z |
|---|---|---|---|
| 1 | $x_4$ | $y_4$ | $z_4$ |
| 2 | $x_5$ | $y_5$ | $z_5$ |
| 3 | $x_1$ | $y_1$ | $z_1$ |
| 4 | $x_4$ | $y_4$ | $z_4$ |
| 5 | $x_2$ | $y_2$ | $z_2$ |

*Figure 23. The point cloud data has geometry representation that is invariant to the data permutations.*

## 5.1 Image-based 3D detection

Similar to the models used for 2D object detection, the methods in [22]–[25], based on monocular images, have employed deep neural networks to extract features with more semantics. [20] proposed a cascaded framework to simultaneously predict the bounding box, part localization, and part visibility of vehicles. Sochor et al. [21] applied the auxiliary rasterized bounding box as a geometry prior and combined the encoded view-points to achieve fine-grained 3D localization on vehicles. Mousavian et al. [22] harnessed the geometric constraints to estimate the orientation of the 3D bounding box detected. However, these methods are not straightforward and are less accurate, because of their lack of depth information. Works [26]–[28], based on RGB-D images, have shown their potential and superiority over those methods based on 2D models. [26] exploited stereo imaging to estimate the depth-encoded features, such as object size priors and ground planes, and generated high-quality 3D bounding box proposals. [27] proposed a 3D Region Proposal Network, which directly acts on the 3D scene. Similar to Faster RCNN, it performs a 3D sliding window search at two scales to generate anchor-related 3D object proposals.

## 5.2 Point-cloud-based 3D detection

Recently, with the remarkable progress of sensing technology, how machines are seeing the world has evolved from 2D optical imaging to 3D scanning methods. The 3D point cloud data, obtained by a visible light scanner (LiDAR), has shown its great advantages in 3D object detection, because point cloud data is convenient to obtain, and is robust to the changes of light and occlusion, as well as being less sensitive to textures. However, processing the point cloud data is challenging, due to its large data volume, irregular structure, and vulnerability to noise. In addition, point cloud data is sparse and unevenly distributed, which makes it easy to miss the object in detection.

How to represent the point cloud data and what kind of in-depth network structure should be used to detect 3D objects is still an open issue.

Currently, most of the existing methods project a 3D point cloud into a 2D image or convert it to a volumetric mesh by quantization before applying the data to a convolutional neural network (CNN). Li et al. [29] are the first to propose transforming point cloud data into Bird's-eye-view (BEV) images, where each pixel represents a grid cell of the horizontal plane. Each pixel is intentionally encoded into different channels in terms of the height, intensity, and density of the points. The same strategy has also applied in [30], where a much simpler conversion from point cloud data to Bird's-eye-viewed RGB map is explored. The Bird's-eye view projection transforms the irregular point cloud data into the Cartesian space. Such transformation allows the use of conventional well-designed and pre-trained 2D CNN models to extract rich representations, and has shown a significant performance improvement over the previous image-based detection approaches, especially for occluded objects.

However, point cloud data still suffer from poor resolution, and the sparse input provides less information about the objects' shapes. Consequently, it is hard to generate object proposals, especially for those far away from the sensors. As a result, some works [31]–[33] explored to combine the high-resolution image source and the corresponding point cloud to perform the detection. Chen at al. [31] proposed a multi-view three-dimensional (MV3D) framework by encoding the point cloud in compact multi-view representations, Bird's-eye view, and frontal view, as shown in **Figure 24**. The network consists of two sub-networks, one is an object proposal network, which generates a 3D candidate box for each object from the Bird's-eye view representation, while the other one is a fusion network, which combines the regional features from the different views and the RGB image. In [33], RGB images are used to generate 2D proposals, which are then lifted to a 3D frustum proposal. The point clouds within the frustum are then fed into a PointNet [34]-based network for further regression and classification. One drawback of [33] is the missing of 2D proposal generation. This will result in the missed detection of some 3D objects. This issue is addressed and resolved in [32], where a multimodal fusion region proposal network was developed, which fuses the high-density image input and the occlusion-free Bird's-eye view input to generate the region proposals.

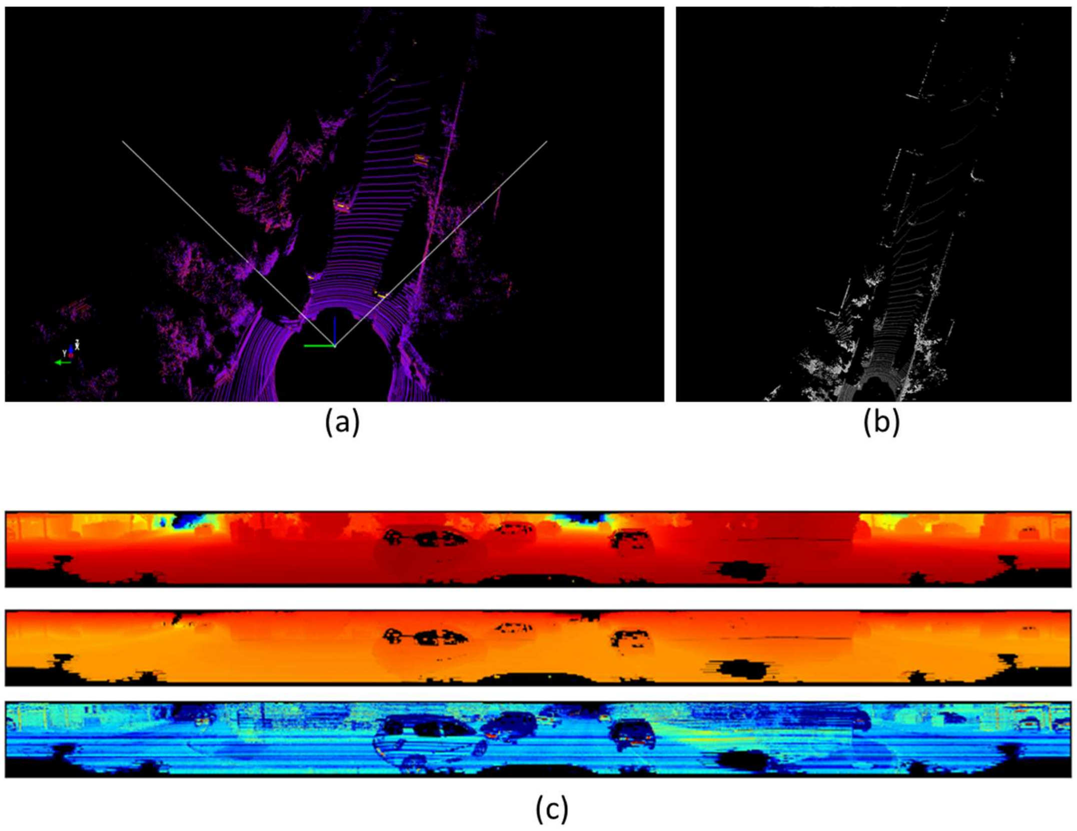


*Figure 24. (a) Point cloud data, (b) the Bird's-eye view map, and (c) the frontal view map which includes the 'depth', 'height', and 'intensity' channels.*

## 5.3 End-to-end learning on Point Cloud

Data representation based on point cloud projection tends to weaken the pattern and affect the invariability of the point cloud data. Recently, some papers have proposed to operate directly on the point cloud, instead of transforming the data into other formats. Wang and Posner [35] presented a novel point-centric voting scheme, which exploit the sparsity of point cloud data to achieve efficient sliding window search.

Another important work is VoxelNet [36], which performs feature extraction and bounding-box prediction into a unified framework. VoxelNet first slices the point cloud into equally spaced 3D sub-regions, known as voxels, and converts a set of points within each voxel into a single feature via a voxel feature encoding (VFE) layer. This layer extract a fix-sized feature for each voxel which is invariant to the point permutation. The point cloud is hence transformed to a volumetric representation. After that, a convolutional middle layer (CML) is applied to perform dimension reduction and outputs a 2D representation which encodes the feature representation in Bird's-eye view. Finally, a region proposal network (RPN) is adapted to perform bounding-box prediction. This work abandons the traditional feature engineering process and multi-view projection, which provides a generic end-to-end learning paradigm on point clouds and achieves higher recall and state-of-the-art performance in the KITTI [37] benchmark. However, VoxelNet is a memory-consuming model. The voxel feature encoding (VFE) module is a stack of sequential vanilla PointNet's layers. Each layer will first transform each point into a high-dimensional feature by a sharable multi-layer perceptron, and then applies a symmetric function, such as max pooling to all the features. The pooled feature of each point is concatenated back to each points and fed to the next layer. We follow the VoxelNet structure, and have proposed a more efficient network, namely VoxelCNN.

## 5.4 VoxelCNN (proposed)

The overall framework is shown in **Figure 25**, which consists of three modules: 1) a voxel feature extraction module, 2) a feature pyramid network, which can leverage the voxel features from multiple scales to perform detection from coarse to fine, and

3) a bounding-box regression layer, which predicts bounding boxes based on the offset towards the predefined anchor boxes. The following will illustrate these modules in detail.

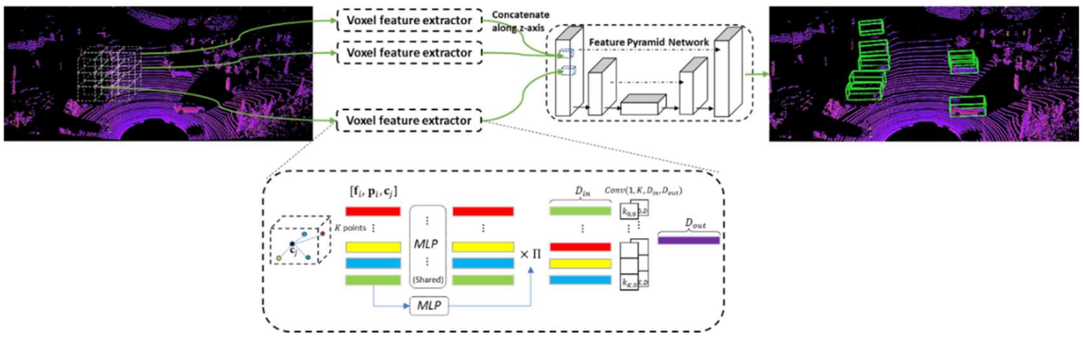


*Figure 25. The proposed detection pipeline.*

We consider the points within the ranges (0m, 72m), (-40m, 40m), and (-2m, 1.2m) along the X, Y, and Z axes, and discretize the point clouds into a voxel grid. Each voxel is a 3D subspace, which contains a variable number of points because the LiDAR points are typically sparse and distributed unevenly. Thus, we randomly sample $K$ points for each voxel if the number of points in a voxel is greater than $K$, and pad zero-valued points if the number is smaller than $K$, to make sure the voxels are non-empty. In our experiments, we discretise the point clouds with the resolutions of 0.2m, 0.2m, and 0.4m along the X, Y, and Z axes, respectively. Therefore, for each example, we generate a sparse 4D tensor, which has the shape of $(352{,}400{,}10, K)$.

### 5.4.1 Position imbedded voxel feature encoding

For each voxel, we propose a position embedded voxel feature encoding layer. The structure of this layer is designed based on these considerations: 1) enforce the permutation canonicalization, 2) better capture the local structure of the data, and 3) with the global position embedding. To clearly illustrate our algorithm, we represent a LiDAR point $\boldsymbol{p}$ into a $(\boldsymbol{p_i}, \boldsymbol{f_i})$ form, where $\boldsymbol{p_i} = [x_i, y_i, z_i] \in R^3$ and $\boldsymbol{f_i} =$

$[f_i, r_i, g_i, b_i] \in R^4$ are the position and intensity feature of the points, respectively. To explore the spatial correlation within a local region, we also incorporate the centre position of the $j^{th}$ voxel $\boldsymbol{c_j}$. We first transform the points to the local coordinates by subtracting each point by the centre position. Then, we concatenate the intensity feature with the local positional feature and the centred positional feature, which contains the positional information from both the local and holistic views. The position embedded feature can represent different local structures, and has different receptive fields for voxels with different scales. This variability makes the feature map carry different semantics, which allows us to use a feature pyramid to enrich the presentation (see Section 5.4.3). We follow [36] to apply a sharable multi-layer perceptron to map the feature to a higher dimensional space. This greatly reduces the module's sensitivity to the permutation of the inputs. Instead of applying a symmetric function, as [32] and [34] do, as inspired by the spatial transformer network [21], we learn a $K \times K$ permutation transformation matrix $\Pi$, such that after transformation, the points with different permutation can potentially achieve invariant representation in another dimensional space. Then, we aggregate the information across points by applying a $1 \times K$ convolutional layer. The above-mentioned process can be summarized as the following algorithm, known as Position Embedded Voxel Feature Encoding (PEVFE) module.

Since the point-cloud points are sparsely distributed across space, most of the voxels contain no points. Therefore, we can simply perform the voxel-wise feature extraction, because our voxel feature encoding layer is sharable across voxels, and for each voxel, the multi-layer perceptron are also shareable across the points. Thus, the algorithm in **Table 7** can be parallelized in the point level by using GPU. We first store the voxel index for each point in a table, then we perform the point-wise feature extraction (up to line 5). To aggregate the features from the same voxel, we apply the $Conv(1 \times K, D_{in}, D_{out})$ convolutional kernel across all the voxels to calculate the voxel-wise feature, where $1 \times K$ is the kernel size, and $D_{in}, D_{out}$ are the number of input channels and output channels respectively. The aggregated voxel features are then indexed to the dense feature volume based on the pre-stored voxel indices. This fast GPU implementation is shown in **Figure 26**.

---

**Algorithm 5.1**: Position Embedded Voxel Feature Encoding (PEVFE) Module

---

```
Input: positional feature: $\boldsymbol{P} = (\boldsymbol{p_0}, \boldsymbol{p_1}, \cdots\cdots, \boldsymbol{p_{K-1}})^{\boldsymbol{T}}, intensity\ feature: \boldsymbol{F} = (\boldsymbol{f_0}, \boldsymbol{f_1}, \cdots\cdots, \boldsymbol{f_{K-1}})^{\boldsymbol{T}}$, and centroid of voxel $c_j$
Output: encoded voxel feature: $\boldsymbol{y}$
    1. Local coordinates transform: $\widehat{\boldsymbol{P}} = \boldsymbol{P} - c_j$
    2. Position embedding: $\widehat{\boldsymbol{F}} = [\boldsymbol{F}, \widehat{\boldsymbol{P}}, c_j]$    $(\widehat{\boldsymbol{F}} \in \mathbb{R}^{\boldsymbol{K} \times \boldsymbol{7}})$
    3. Project to high dimensional feature space: $\widetilde{\boldsymbol{F}} = MLP(\widehat{\boldsymbol{F}})$   $(\widetilde{\boldsymbol{F}} \in \mathbb{R}^{\boldsymbol{K} \times \boldsymbol{D_1}})$
    4. Learn the permutation transformation matrix: $\Pi = MLP(\widehat{\boldsymbol{F}})$
    5. Permutation invariance features: $\boldsymbol{F}^* = \Pi \times \widetilde{\boldsymbol{F}}$
    6. Feature aggregation: $\boldsymbol{y} = \boldsymbol{Conv}(\boldsymbol{1} \times \boldsymbol{K}, \boldsymbol{D_1}, \boldsymbol{D_2})(\boldsymbol{F}^*)$    $(\boldsymbol{y} \in \mathbb{R}^{\boldsymbol{D_2}})$
```

---

*Table 7.The algorithm implemented in PEVFE module.*

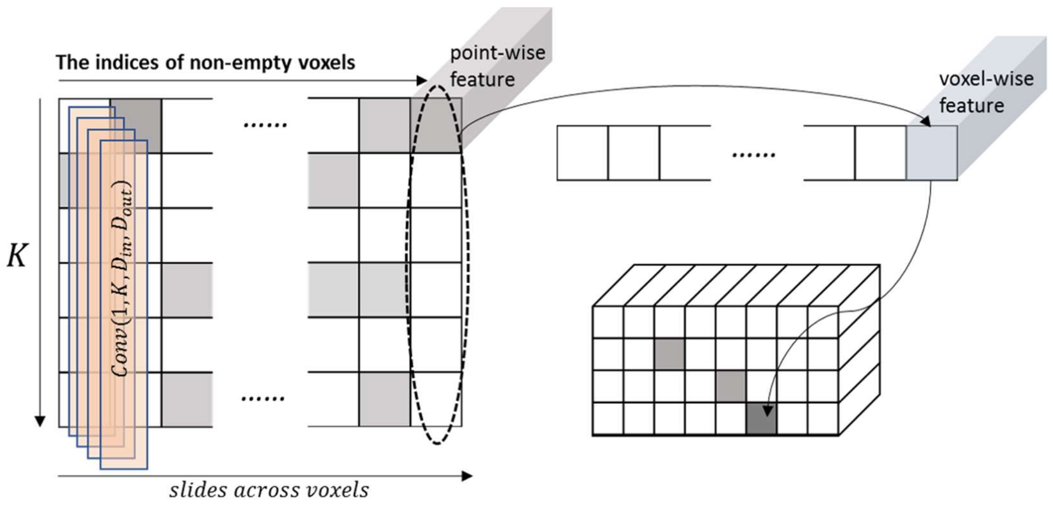


*Figure 26. Fast GPU implementation of the voxel encoding layer.*

### 5.4.2 Bounding-box regression

We simply adopt the anchor-based regression strategy to predict the 3D bounding-box offset towards each anchor. We start from the Bird's-eye view and divide the searching space into a grid with three different resolutions, which are compatible with the three scaled output feature maps. We encode the ground-truth bounding box to the target feature map by calculating the overlapping area between the ground-truth bounding box and the anchor boxes. Since the point clouds are usually sparse, most of the anchor boxes contain no points, those anchors overlapped with the ground-truth objects are far less than those are not. Therefore, to avoid overfitting the problem, we categorize the anchors into either positive, negative or "ignored", with the following rule: 1) An anchor box having an overlapping (IoU) area with one of the ground-truth boxes, with IoU larger than 0.65, is declared positive. 2) For a ground-truth box, the corresponding anchor box with the largest overlapping area is considered to be positive. 3) The anchors in a cell, which contain no points, are declared "ignored". 4) An anchor with the overlapping area smaller than 0.4 is also declared "ignored". 5) An anchor not classified into any of the above four cases is considered to be negative.

Consider a ground-truth bounding box with parameters $(x_b, y_b, z_b, h_b, w_b, l_b, \theta_b)$, and the positive anchor at the ith position $(x_c^i, y_c^i)$, the target output $\boldsymbol{t_i} = (x_t^i, y_t^i, z_t^i, h_t^i, w_t^i, l_t^i, \tau_t^i, \eta_t^i) \in R^8$ at this position should satisfy the following:

$$x_b = \sigma(x_t^i) + x_c^i,$$

$$y_b = \sigma(y_t^i) + y_c^i,$$

$$z_b = \sigma(z_t^i) + z_{velo},$$

$$h_b = h_a e^{h_t^i},$$

$$w_b = w_a e^{w_t^i},$$

$$l_b = l_a e^{l_t^i}, \text{ and}$$

$$\theta_b = \arctan \frac{\tanh(\tau_t^i)}{\tanh(\eta_t^i)}.$$

"tanh" is used to normalize the output so that it lies in $[-1,1]$ and $\sigma$ is the sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}},$$

and $z_{velo}$ is the z-value in the Velodyne coordinate of the ground plane, which equals -1.7. Thus, we define the following loss functions, a localization loss, which is applied to positive anchors only, and a classification loss, which calculates the cross-entropy loss over all the positive anchors and negative anchors. The localization loss is defined as follows:

$$\mathcal{L}_{\ell o c} = \frac{1}{N_{pos}} \sum_i \text{SmoothL1}(\boldsymbol{p_i^{loc}}, \boldsymbol{t_i}),$$

where SmoothL1 is the smooth $\ell 1$ function [12], $\mathrm{i}$ is the index of the positive cells, and $\mathrm{N_{pos}}$ is the number of positive anchors. The classification loss is defined as follows:

$$\mathcal{L}_{cls}^{pos} = \frac{1}{\mathrm{N_{pos}}} \sum_{\mathrm{i}} \mathrm{BCE}\left(\boldsymbol{p}_i^{cls}, 1\right),$$

$$\mathcal{L}_{cls}^{neg} = \frac{1}{\mathrm{N_{neg}}} \sum_{\mathrm{j}} \mathrm{BCE}\left(\boldsymbol{p}_j^{cls}, 0\right)$$

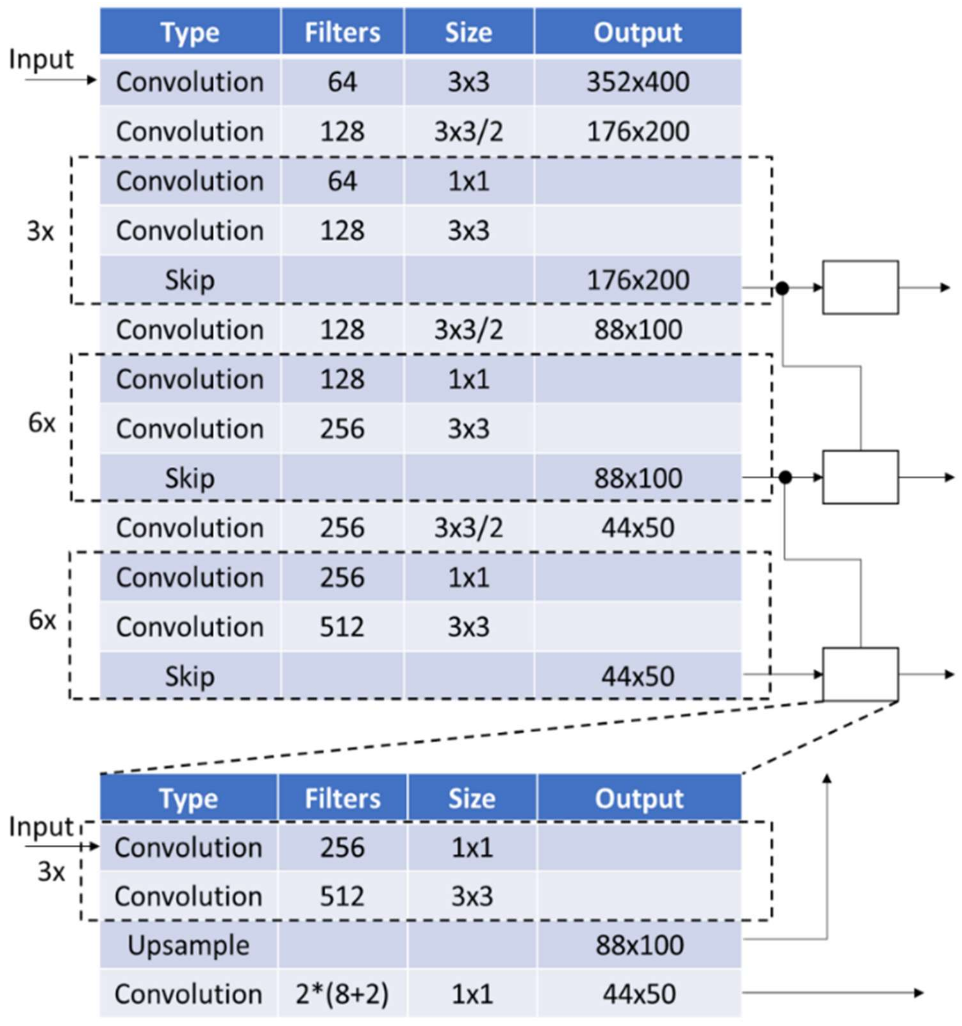
where BCE stands for binary cross entropy loss, $\mathrm{i}$ and $\mathrm{j}$ are the indices of the positive and negative anchors, respectively. Note that, in our scenario, the positive and negative anchors are highly imbalanced. Thus, we perform the hard-negative mining by sorting the negative anchors in terms of the classification loss in descending order. Then, we select the top $\mathrm{N_{neg}}$ negative anchors so that the ratio of $\mathrm{N_{pos}}:\mathrm{N_{neg}}$ equals 1:3. The loss functions can be jointly optimized as follows:

$$\mathcal{L} = \alpha \mathcal{L}_{loc} + \beta \mathcal{L}_{cls}^{pos} + \gamma \mathcal{L}_{cls}^{neg},$$

where the weighting factors $(\alpha, \beta, \gamma) = (1.5, 5.0, 0.5)$, which are determined empirically so that the losses will almost lie in the same range.

### 5.4.3 Feature pyramid network

We use the feature pyramid network [13] as our backbone network because it can leverage the feature with different scales and semantics, and it demonstrates superior performance in object detection. We build a feature pyramid network (FPN), as shown in **Figure 27,** to learn hierarchical representations from the voxel features. FPN is built by stacking a number of convolutional blocks, each of which uses sequential $1 \times 1$ and $3 \times 3$ convolutional layers, and is then followed by a batch normalization and ReLU activation. We also include skip-connections after each block, which allows the model to be deeper. The last layer predicts the encoded bounding box and classification probability. In our experiments, we predict two boxes, each of which contains 8 regression parameters and 2 classification parameters in terms of foreground/background probabilities, so the output is a $\mathrm{W} \times \mathrm{H} \times [2 * (8 + 2)]$ tensor. We also take the features from earlier layers and concatenate them with the bottom layers via an up-sampling kernel. The following diagram shows the network in detail.



| Type | Filters | Size | Output |
|---|---|---|---|
| Convolution | 64 | 3x3 | 352x400 |
| Convolution | 128 | 3x3/2 | 176x200 |
| Convolution | 64 | 1x1 | |
| Convolution | 128 | 3x3 | |
| Skip | | | 176x200 |
| Convolution | 128 | 3x3/2 | 88x100 |
| Convolution | 128 | 1x1 | |
| Convolution | 256 | 3x3 | |
| Skip | | | 88x100 |
| Convolution | 256 | 3x3/2 | 44x50 |
| Convolution | 256 | 1x1 | |
| Convolution | 512 | 3x3 | |
| Skip | | | 44x50 |

| Type | Filters | Size | Output |
|---|---|---|---|
| Convolution | 256 | 1x1 | |
| Convolution | 512 | 3x3 | |
| Upsample | | | 88x100 |
| Convolution | 2*(8+2) | 1x1 | 44x50 |

*Figure 27. The proposed feature pyramid network.*

### 5.4.4 Data augmentation

Data augmentation is crucial in our training because only a limited number of training samples are available. An existing pre-trained model is used and fine-tuned with the training samples in order to avoid overfitting. The data is augmented randomly using one of the following ways.

1) Flip the point cloud and the ground-truth bounding box horizontally and vertically.

2) Add jitter to all the points within each ground-truth bounding box.

3) Apply a global scaling to all the points and ground-truth boxes.

4) Apply a global rotation to all the points and ground-truth boxes.

### 5.4.5 Performance evaluation

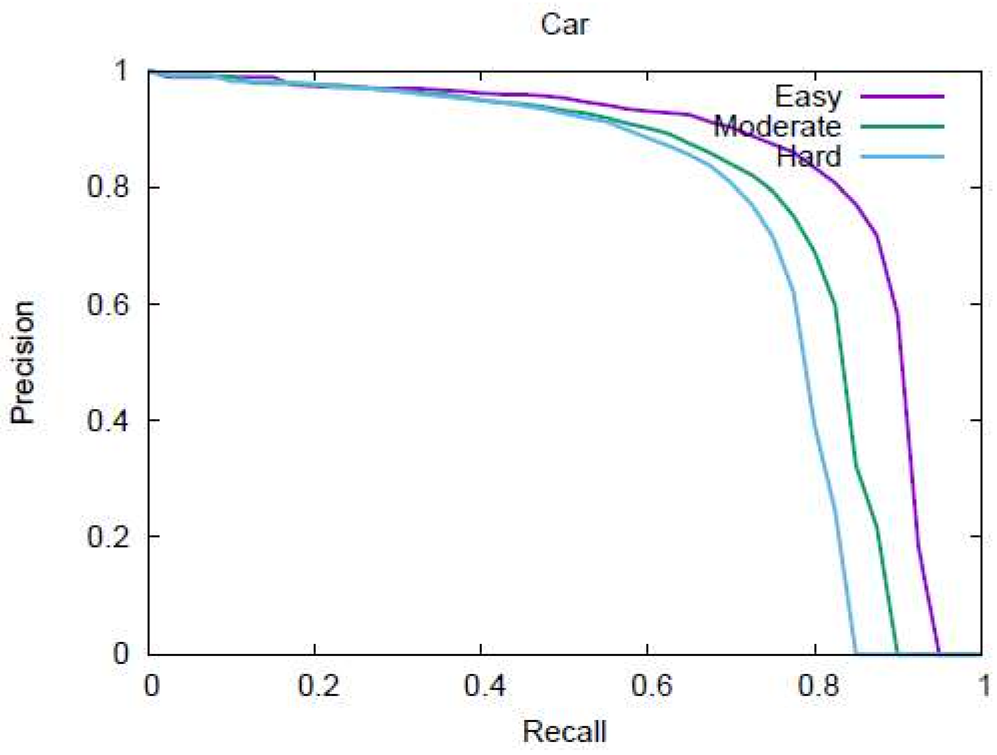
We evaluate our proposed detector on the KITTI [37] 3D object detection benchmark. This dataset contains 7,481 training examples, including RGB/grayscale images from four cameras and point clouds. The objects include vehicles, pedestrians, and cyclists. The evaluation is categorized into three different levels: easy, moderate, and hard, based on the object size and the amount of occlusion, as well as whether the object is truncated. We conducted our experiments on detecting vehicle objects, and we split the training data into a training and a validation set, with a ratio of about 1:1. This results in having 3,712 samples for training and 3,769 samples for validation.

To analyze the proposed feature-encoding module, we chose the binary voxel feature [29] as the baseline to compare with our proposed detector. Since our detector outputs are 3D bounding-boxes, we reformat them to BEV bounding-box in 2D by dropping the height value. We evaluate the performances of BEV detection and 3D detection based on the PASCAL criteria. Specifically, we consider each detection as a hit/miss based on the overlapping area/volume between the predicted bounding box and the ground-truth box. For cars, this bounding-box overlapping area/volume is required to exceed 0.7. Then we can calculate multiple precision-recall (PR) values in different confidence levels and plot a PR curve. The detection performance can then be evaluated in terms of the area under this curve, known as average precision. The network is trained for 100 epochs, with a learning rate of 0.001 using the Adam optimizer. To remove duplicated bounding boxes, we perform 3D non-maximum suppression.

**Table 8** shows the detection performance in terms of the mean average precision. As seen from the table, our proposed voxel encoding module can learn features more robust than the binary voxel features and PointNet feature. However, the robustness also depends on the number of points sampled within a voxel. In our method, we sample only 35 points for each non-empty voxel, which follows the same setting in VoxelNet [36]. To visualize the precision-recall trade-off of the proposed detector, the PR curve is plotted, as shown in **Figure 28**.

| **Benchmark** | **Method** | **Easy** | **Moderate** | **Hard** |
|---|---|---|---|---|
| Car (3D Detection) | 3DFCN | 42.31% | 39.52% | 32.81% |
| | VoxelNet* | 53.43% | 48.78% | 48.06% |
| | Proposed | **63.46%** | **57.90%** | **56.06%** |
| Car (Bird's Eye View) | 3DFCN | 74.29% | 68.31% | 62.92% |
| | VoxelNet* | **85.41%** | **83.16%** | 77.10% |
| | Proposed | 83.42% | 82.61% | **80.47%** |

*Table 8. Mean average precision on two detection tasks. * indicates the unofficial results which are based on the implementation [https://github.com/qianguih/voxelnet](https://github.com/qianguih/voxelnet)*



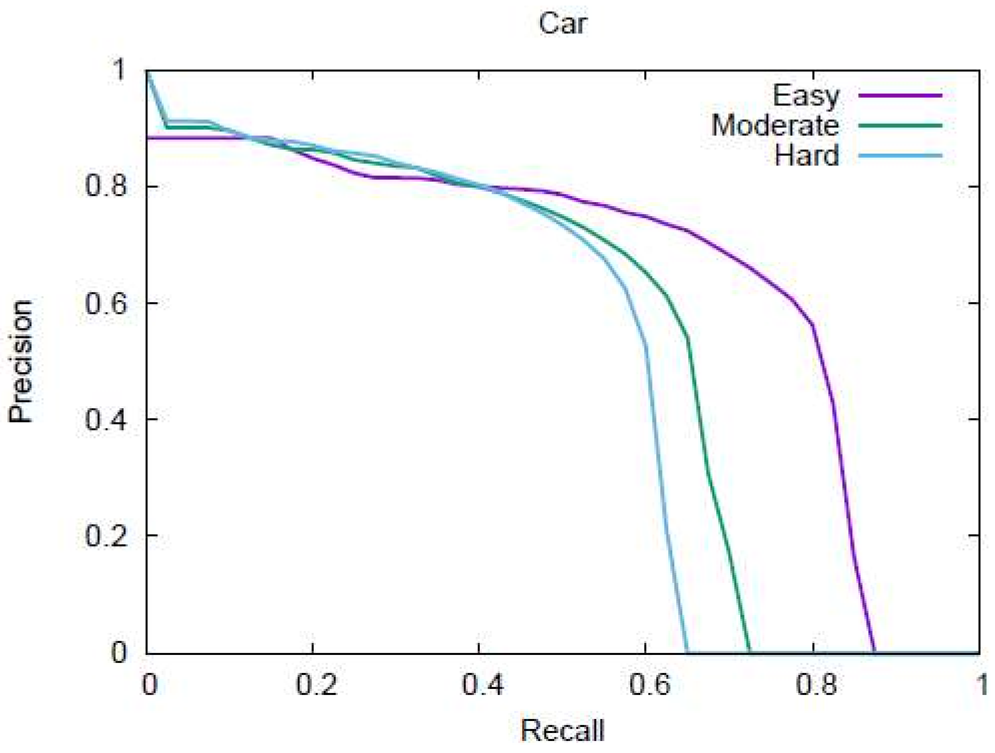
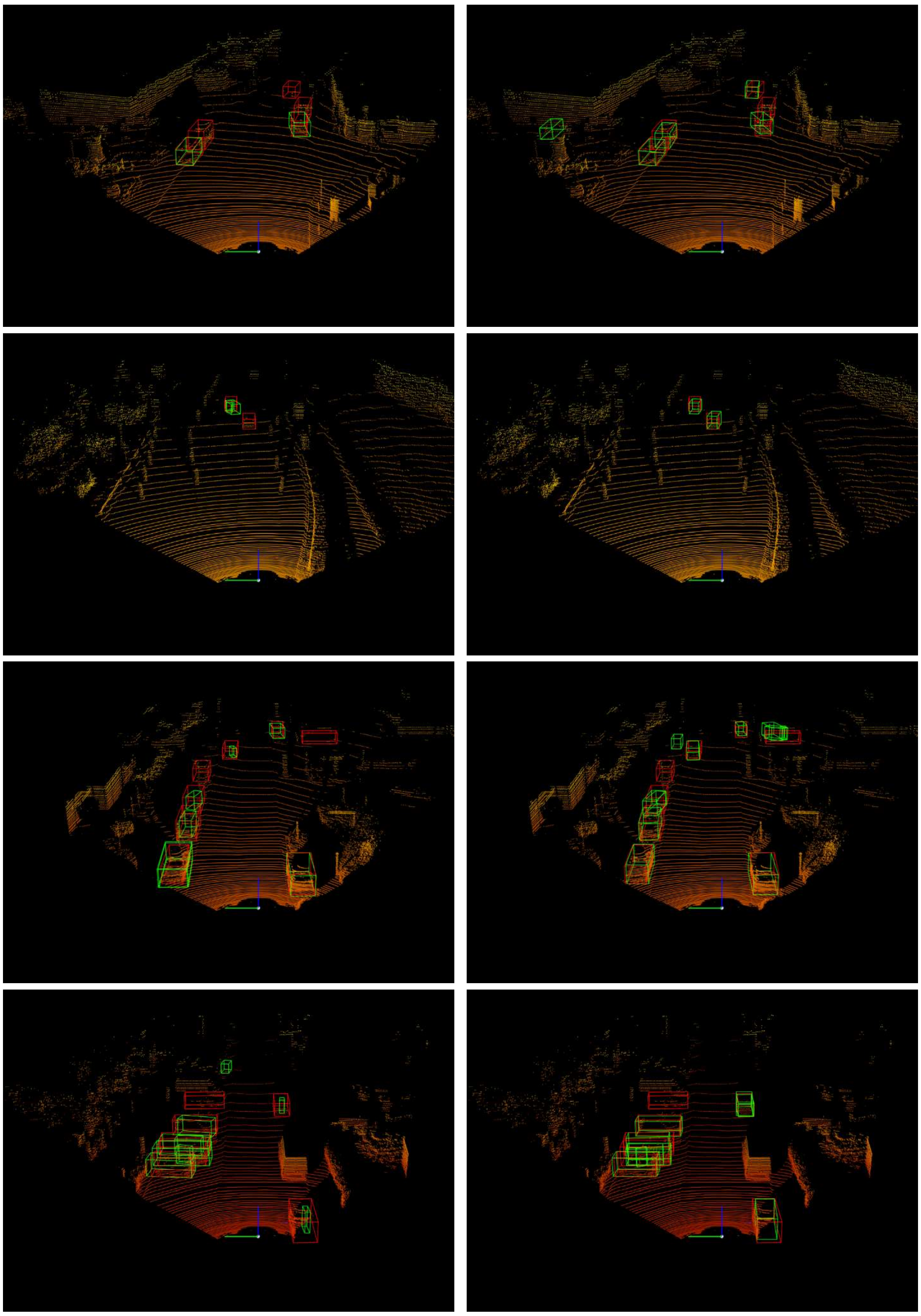


*Figure 28. The precision and recall in the Bird's-eye view detection (up) and 3D detection tasks (bottom).*



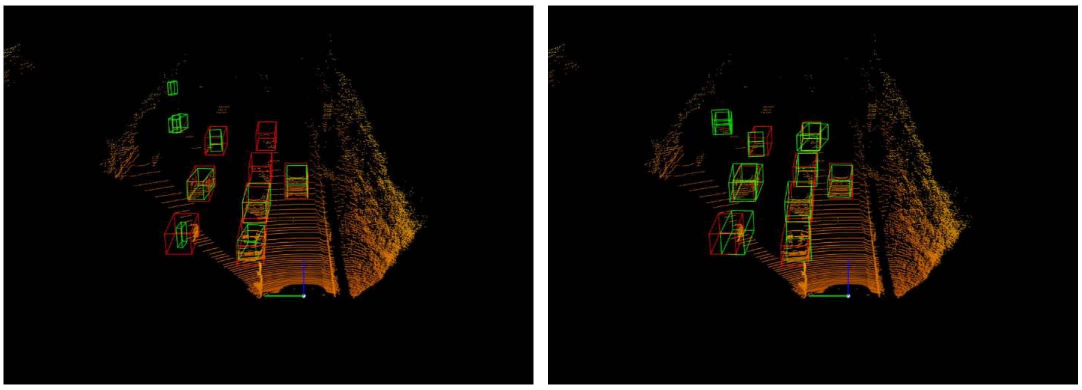
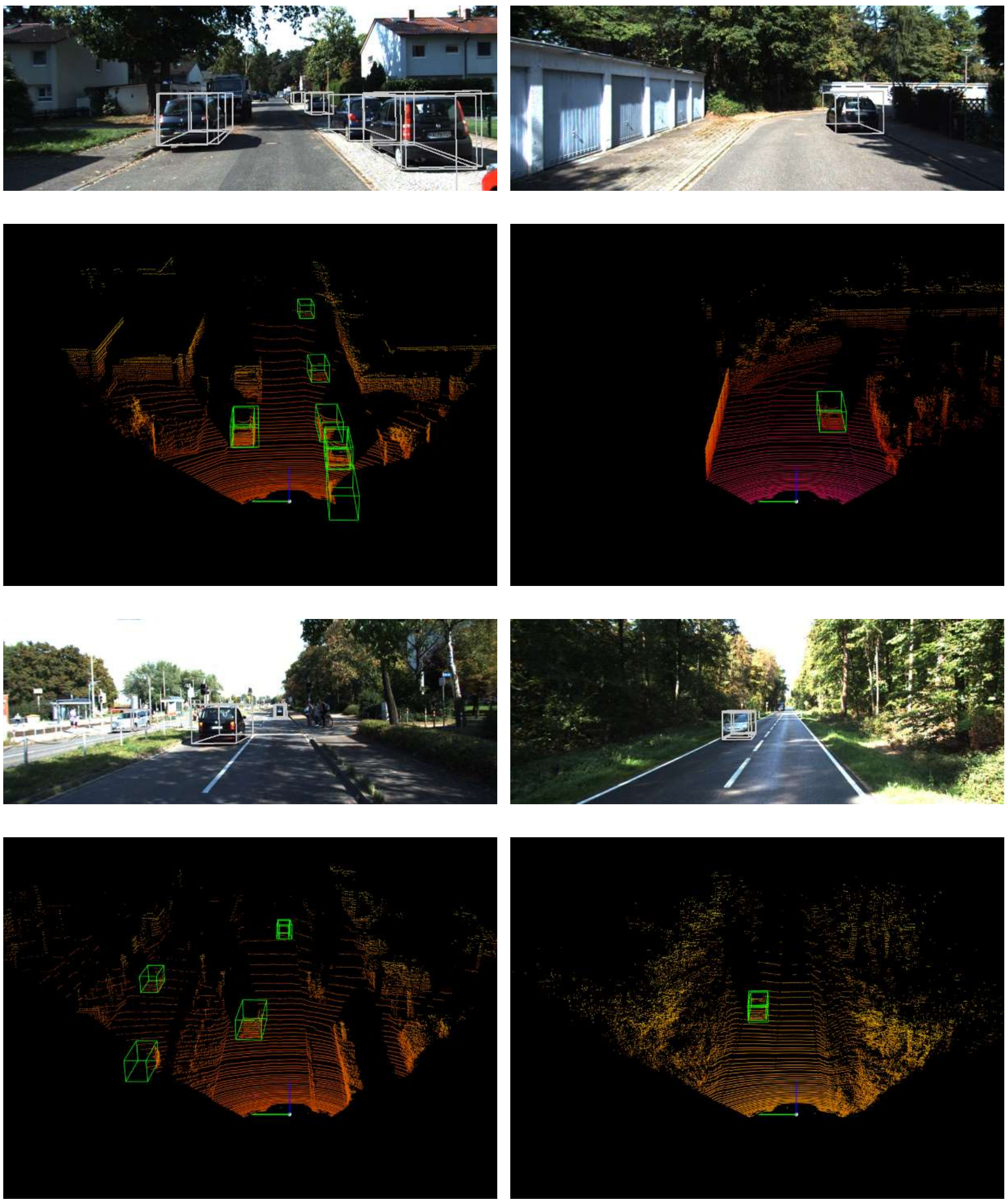


*Figure 29. The detection result between VeloFCN(left)[29] and our proposed VoxelCNN(right). The green boxes are prediction boxes and the red ones are ground-truth boxes. Note that our proposed detector can achieve less missing rate and better localization in 3D detection task.*

**Figure 29** shows the qualitative results of the VeloFCN *[29]* and our proposed VoxelCNN detector. Note that, our approaches can achieve better performance in both classification and localization.

**Figure 30** shows the qualitative results of our method on the validation set. Most of the vehicles can be detected successfully even under a severe occlusion. Occlusion is a common characteristic of LiDAR data. However, in some cases where the objects are far from the LiDAR source, the detection may be heavily interfered with. The reason behind this is the point cloud data, which is at a farther distance, tends to be sparser.



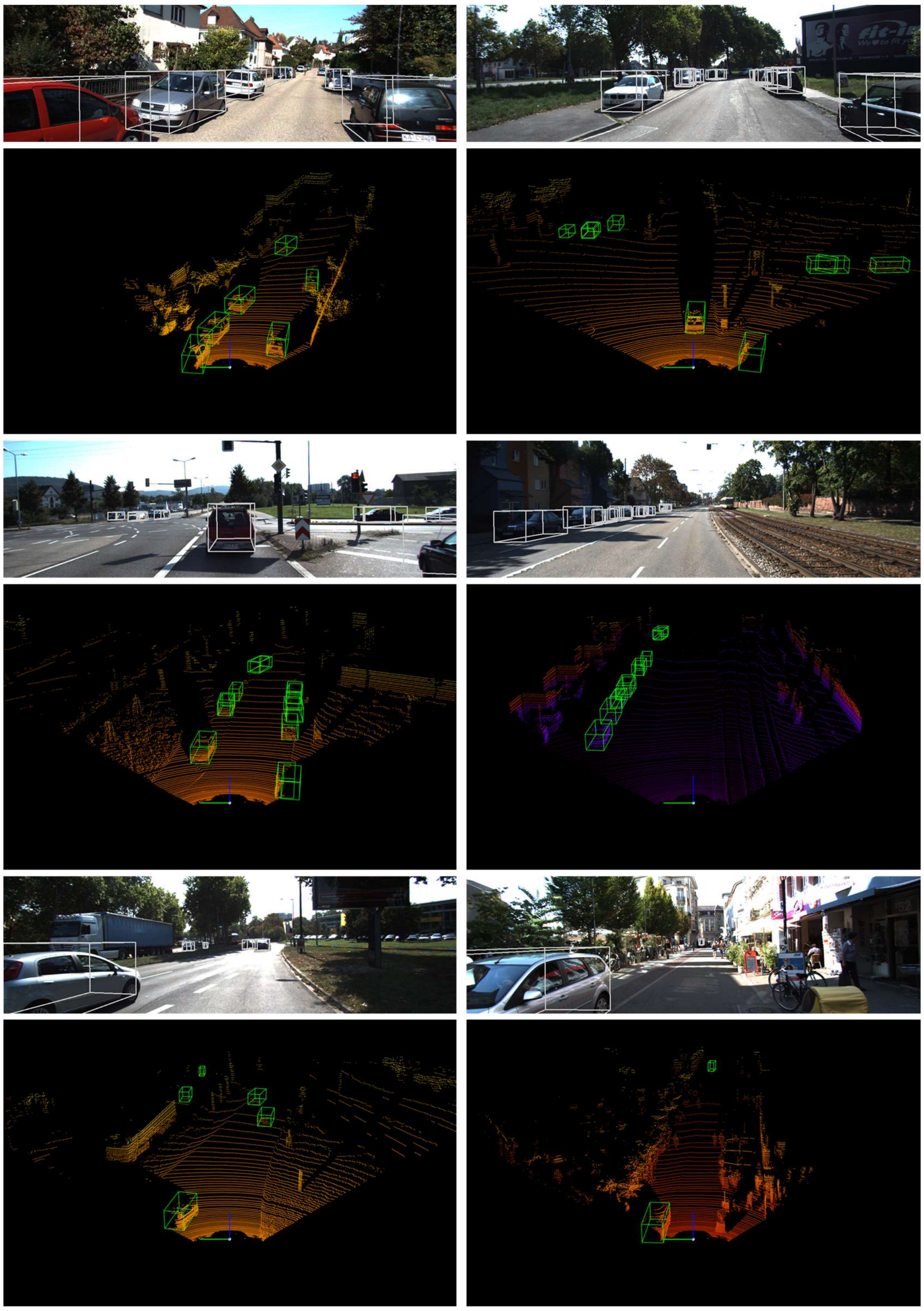


*Figure 30. The qualitative results on the KITTI validation set.*

## 5.5 Conclusion

In this chapter, we first introduced the 3D object detection based on point clouds data by its challenges, the point clouds data is typically sparse and has a non-Euclidian data structure. We then reviewed the early approaches which are about projecting the point clouds into a Brid's eye view map and applying the conventional image-based technique to perform detection. We then introduced the recently proposed voxel feature leaning which can efficiently perform detection in an end-to-end manner. We also demonstrated a proposed detector, which performs voxel feature learning with a novel permutation transformation technique. The proposed detector also resorted multi-scale features by a feature pyramid network and showed its potential power in 3D vehicle detection task.

# Chapter 6. Conclusion

In this thesis, we have overviewed different object-detection approaches proposed in the computer-vision community. In Chapter 2, we introduced the conventional methods in object detection, which applied handcrafted features, like histogram of oriented gradients (HoG), local binary pattern (LBP), regionlet, etc., and used machine learning based classifiers, such as adaptive boosting, support vector machine (SVM), etc. to determine the objectness of sampled regions extracted by using the sliding window search. We have also introduced the approaches based on deep learning and big data. In this kind of approaches, researchers started to train deep neural networks for extracting deep features using an extremely large amount of data. It has turned out that the deep features are more robust to different environments, such as weather conditions, occlusion, lighting changes, etc. Furthermore, the deep features are better to characterize different objects, in terms of different variations such as viewpoints, scales and aspect ratios. Instead of using the sliding-window search, researchers tended to selectively generate region proposals by some image prior, or directly use deep features to coarsely predict the possible region proposals. We have also summarized two different paradigms based on deep-learning approaches. The first one is the one-shot detector, which directly regresses the object position and the class from the same feature layer. This detector can achieve fast detection. In this paradigm, some prior boxes (default boxes) are used and regression is performed on each of these boxes. A box was considered into positive, negative and ignored based on the intersection over union (IoU) with respect to the ground-truth boxes. A large number of prior boxes will also cause class imbalance problem which makes the neural network hard to train so that affect the recall of the one-shot detector. Some training tricks like online hard negative mining (OHNM) and *focal loss* can be used to stabilize the training. Compare to the single-shot detector, the region-based detector can achieve relatively higher recall. The region-based detector performs detection in two stage, the first stage is about generating region proposals using a class-agnostic one-shot detector and the second stage is about performing classification on these proposals. However, the two-stage detection manner makes it has relatively slow.

In Chapter 3, we present our proposed lateral convolutional neural network and we applied it to the vehicle detection task. In this model, we argue that different layers from deep neural network exhibit different semantics. For the earlier layer, the fine-grained details are better to perform good localization, but low semantics will affect the objectness prediction. Thus, we decouple the localization regression and objectness prediction from a single layer. Instead, we use an earlier layer to regress the localization residual and apply a new bounding box encoding scheme so that the detector is robust to handle the occluded object. The proposed network shows its potential on the video surveillance for vehicle detection and can achieve almost real-time with a moderate GPU.

In Chapter 4, we develop a traffic sign recognition system. We applied the aforementioned detector to detect the traffic sign and applied another convolutional neural network to perform classification. We consider the scenario the traffic sign may not be well localized from our detector and developed a classifier which is robust to the spatial shifted from the object. The classifier is built by sequential of convolutional layers and one channel-attention layer and one spatial attention module. Experiments show that with the spatial attention module, our classifier can further enhance the recognition rate. The channel attention layer also shows its potential when detecting the objects in salient color.

In Chapter 5, we investigate the 3D object detection based on the point clouds. We first reviewed the latest technology on point-cloud-based detection which projects the point cloud into different views and performs the image-based method. Some research focuses on multi-source fusion which integrating the features from both point cloud data and RGB image. Inspired by VoxelNet, we proposed a "position embedded voxel feature encoding" module which can directly learn the features from the scattered point. This module tried to solve the permutation invariant problem existed in point set feature learning. In which, we learn a permutation canonicalize matrix to transform the point set into a high-dimensional domain and apply a convolution operator to extract features from the point set. We also utilize the feature pyramid network to perform multi-scale detection and propose a novel 3D bounding box encoding scheme which can predict the oriented 3D bounding box with 8 regression parameters.

In point-cloud-based detection problem, the sparsity and the large capacity of the point cloud data still a tough problem. Besides this, point set lies in an irregular domain which is hard to extract feature with a fix configured convolutional kernel. In our future research, we plan to solve these two issues. For the first issue, our thought is to perform sparse to dense hallucination by using some image cues. For the second issues, we will keep track of the current research on point set classification problem and explore more on point set representation methods.

# References

[1] *N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), 2005, vol. 1, pp. 886–893 vol. 1.*

[2] *X. Wang, T. X. Han, and S. Yan, “An HOG-LBP human detector with partial occlusion handling,” in 2009 IEEE 12th International Conference on Computer Vision, 2009, pp. 32–39.*

[3] *P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object Detection with Discriminatively Trained Part-Based Models,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.*

[4] *P. Viola and M. J. Jones, “Robust Real-Time Face Detection,” International Journal of Computer Vision, vol. 57, no. 2, pp. 137–154, May 2004.*

[5] *X. Wang, M. Yang, S. Zhu, and Y. Lin, “Regionlets for Generic Object Detection,” presented at the Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 17–24.*

[6] *A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.*

[7] *B. Alexe, T. Deselaers, and V. Ferrari, “Measuring the Objectness of Image Windows,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 11, pp. 2189–2202, Nov. 2012.*

[8] *K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders, “Segmentation as selective search for object recognition,” in 2011 International Conference on Computer Vision, 2011, pp. 1879–1886.*

[9] *R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” in Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC, USA, 2014, pp. 580–587.*

[10] R. Girshick, "Fast R-CNN," in Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Washington, DC, USA, 2015, pp. 1440–1448.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in Advances in Neural Information Processing Systems 28, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99.

[12] W. Liu et al., "SSD: Single Shot MultiBox Detector," in Computer Vision – ECCV 2016, 2016, pp. 21–37.

[13] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 936–944.

[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," arXiv:1708.02002 [cs], Aug. 2017.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[16] L. Wen et al., "UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking," arXiv:1511.04136 [cs], Nov. 2015.

[17] Z. Cai, M. Saberian, and N. Vasconcelos, "Learning Complexity-Aware Cascades for Deep Pedestrian Detection," presented at the Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3361–3369.

[18] L. Wang, Y. Lu, H. Wang, Y. Zheng, H. Ye, and X. Xue, "Evolving boxes for fast vehicle detection," in 2017 IEEE International Conference on Multimedia and Expo (ICME), Hong Kong, Hong Kong, 2017, pp. 1135–1140.

[19] "Detection of traffic signs in real-world images: The German traffic sign detection benchmark - IEEE Conference Publication." [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6706807. [Accessed: 04-Oct-2018].

[20] "Systematic evaluation of convolution neural network advances on the Imagenet - ScienceDirect." [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314217300814. [Accessed: 17-Aug-2018].

*[21] M. Jaderberg, K. Simonyan, A. Zisserman, and koray kavukcuoglu, "Spatial Transformer Networks," in Advances in Neural Information Processing Systems 28, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2017–2025.*

*[22] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau, "Deep MANTA: A Coarse-to-fine Many-Task Network for joint 2D and 3D vehicle analysis from monocular image," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2040–2049.*

*[23] J. Sochor, A. Herout, and J. Havel, "Boxcars: 3d boxes as cnn input for improved fine-grained vehicle recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 3006–3015.*

*[24] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká, "3d bounding box estimation using deep learning and geometry," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 5632–5640.*

*[25] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1521–1529.*

*[26] X. Chen et al., "3D Object Proposals for Accurate Object Class Detection," in Advances in Neural Information Processing Systems (NIPS), 2015.*

*[27] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 808–816.*

*[28] Z. Deng and L. J. Latecki, "Amodal detection of 3d objects: Inferring 3d bounding boxes from 2d ones in rgb-depth images," in Conference on Computer Vision and Pattern Recognition (CVPR), 2017, vol. 2, p. 2.*

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

*[30] M. Simon, S. Milz, K. Amende, and H.-M. Gross, "Complex-YOLO: Real-time 3D Object Detection on Point Clouds," arXiv preprint arXiv:1803.06199, 2018.*

*[31] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, vol. 1, p. 3.*

*[32] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3D Proposal Generation and Object Detection from View Aggregation," arXiv preprint arXiv:1712.02294, 2017.*

*[33] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D Object Detection from RGB-D Data," arXiv preprint arXiv:1711.08488, 2017.*

*[34] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, vol. 1, p. 4.*

*[35] D. Z. Wang and I. Posner, "Voting for Voting in Online Point Cloud Object Detection," in Proceedings of Robotics: Science and Systems, Rome, Italy, 2015.*

*[36] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," arXiv preprint arXiv:1711.06396, 2017.*

*[37] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.*