DEPTH AWARE RCNN

BY

TIANXI ZHAO

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Assistant Professor Honghui Shi

ABSTRACT

Image object detection networks that depend on region proposal networks (RPN) have achieved state-of-art results. As RPN is trained to share convolutional features with the actual classification layers in the network, features learned by the convolutional backbones may have subtle impact on the RPN. A successful approach comes from RGB-D image object detection, where the convolutional layers learn not just RGB features, but also depth features.

In this thesis, we study the problem of simultaneously localizing objects as well as estimating their depth. We propose to use one backbone network for two tasks and show that multi-task learning with shared weights can have reciprocating benefits. Our experiments show that when combined with depth prediction in the network, the object detection branch in our model outperforms Faster-RCNN on the challenging KITTI detection benchmark and the Cityscapes dataset. Likewise, the performance of our depth prediction branch is slightly better compared with methods using the same depth prediction architecture.

To my parents: My success is impossible without your support and love.

ACKNOWLEDGMENTS

I would like to begin by thanking my thesis advisor, Professor Honghui Shi and Professor Thomas S. Huang, for providing me the opportunity to participate in the project described in this thesis. Their knowledge and practical approach to engineering and research made a lasting impression that I will carry throughout my professional career. Special thanks to Professor Shi for his insight on project selection and experiment design.

This project would also not be possible without the joint efforts from my fellow students and colleagues: Jianbo Jiao and Shengkun Cui. I appreciated our discussion of project ideas and I'm grateful for extremely useful inputs to the design of the project. I am gratefully indebted for Jianbo Jiao's invaluable comments on this thesis. I would also like to thank Shengkun Cui for providing extra computing resources for training and validating our proposed neural networks. His help was essential for acquiring the experimental results presented in this thesis.

I would also like to acknowledge the financial support from the ECE department for providing my teaching assistant position. By working with professor Zbigniew Kalbarcyzk and other TAs, I was able to practice my communication skills and consolidated my knowledge to fundamental computing systems.

I would also like to thank my parents for their sacrifice and support throughout my life. Without their vision and the invaluable opportunities they provided early in my life, I would not have achieved my current success.

Last, but definitely not least, I would like to thank all my friends who were with me throughout my time at the University of Illinois. Thank you Oscar Bi, You Guan, Yu Huang, Sharon Tang, Umberto Ravaioli and many others for making these past seven years some of the best times of my life. I will forever cherish our friendship and wish you all the best of luck in your future endeavors.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Object detection within single images has a long history in computer vision. With the rise of region proposal methods and region-based convolutional neural networks (RCNN) [1], impressive progress has been made on this challenging task. However, the performance of the state-of-art object detection is still not good enough compared to that of average humans. This is because that, in general, only RGB information of images are used in these methods. Humans, on the other hand, in addition to directly selecting regions of interest (RoI) based on the appearance of the scene, perform depth estimation to decide which RoI deserves more attention. Such a process is used on monocular image scenes as well, since humans are adept at monocular depth estimation by using many abstract vision cues to understand the scene of interest [2]. Research on human depth perception has also shown that, when the human brain attempts to understand an image, it tries to both predict scenery depth and recognize objects at the same time. It can be observed that neurons working on each task share signals provided by the visual cortex, and correspondence between the perception from each tasks complement each other to get even more accurate results [3].

A more intuitive example is presented here to show that by estimating depth map of the original RGB input, it is not challenging to use fundamental clustering methods to categorize pixels into different depth ranges. As shown in Figure 1.1, a depth map was estimated using the Monodepth [4] model from the original RGB image. In order to calculate a histogram of pixel count against their estimated depth, we group pixels into bins of size 30cm from 5m to 50m. The depth of each pixel is defined as the distance between the real-world representative of the pixel and the camera that shot the photo. It is obvious from the histogram shown in Figure 1.1 that most pixels fall within 20m from the camera, and there exist two major peaks which may indicate the existence of objects around those depth ranges. By using the Gaussian

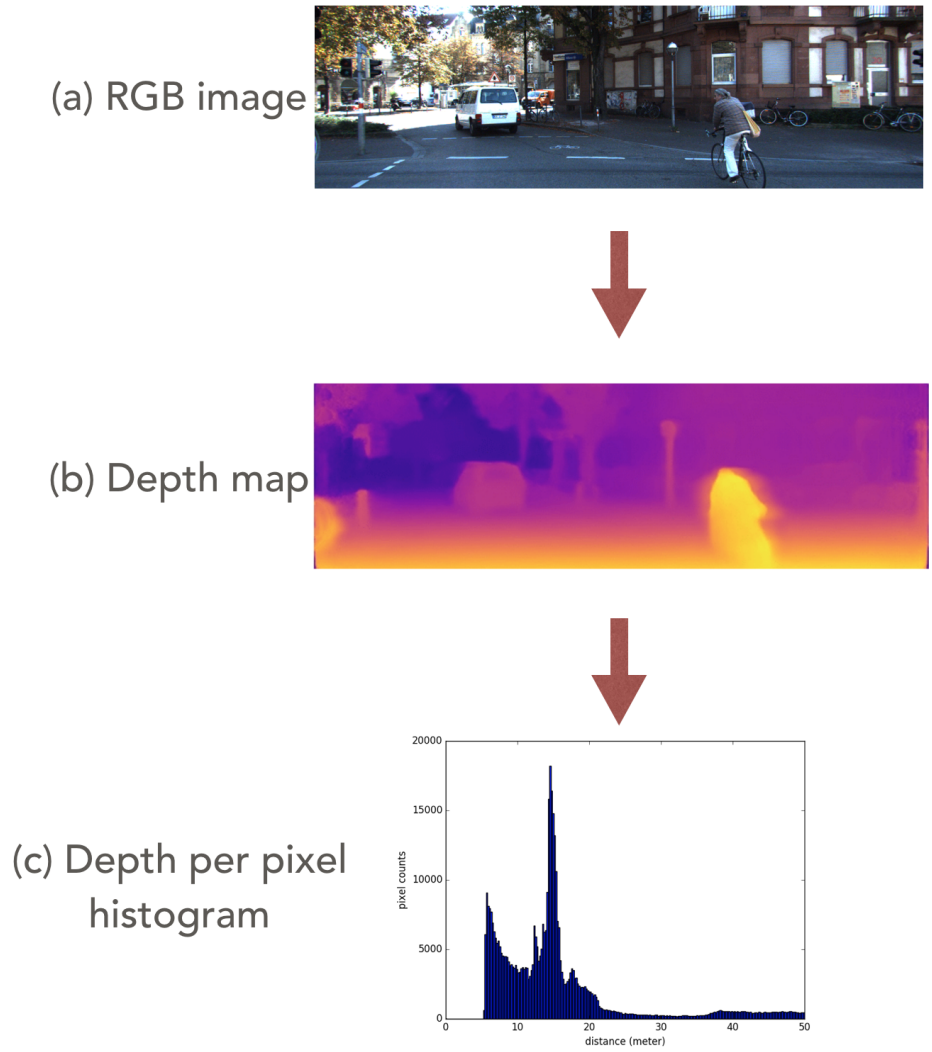

Figure 1.1: The three diagrams from top to bottom are: (a) the original RGB image, (b) the depth estimation generated by Monodepth [4] and (c) histogram that counts pixels against their depth.

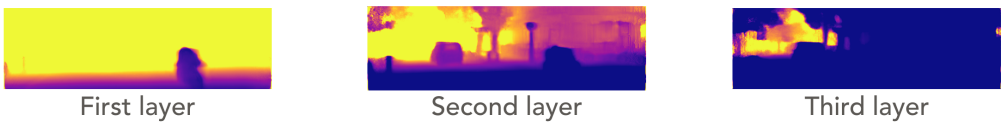

Figure 1.2: Pixels are grouped into three clusters. Starting from the left, we show pixels from 5m to 13m as the first layer, pixels from 14m to 25m as the second layer and pixels from 26m to 50m in the third layer.

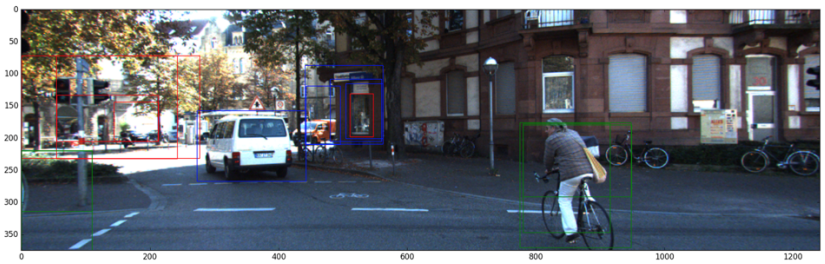

Figure 1.3: The image shows bounding boxes proposed by the selective search algorithm [5] from each depth range in the original image. Green boxes were proposed based on the first layer of depth, blue boxes from the second layer of depth and red boxes from the third layer.

mixture model (GMM), we can cluster the pixels in the given example into three groups. As illustrated in Figure 1.2, each group of pixels was drawn in a different layer. We may observe that the pixels corresponding to the cyclist lie in the first layer, while pixels of the car are in the second layer and the rest of the background pixels are categorized into the third layer. If methods such as selective search [5] and EdgeBoxes [6] algorithms are used to generate regional proposals of object candidates in the separated layers instead of the original images, it would require fewer proposals to cover all objects of interest. Figure 1.3 shows that the top-30 regional proposals, which were generated by the selective search algorithm using the three depth layers, cover all objects of interest. In comparison, it would require ~ 200 regional proposals for the same coverage if the same algorithm is applied to the original RGB image.

However, procedures described in this example require human interaction, such as deciding the best number of clusters of the GMM algorithm. It was popularly known that before the emergence of convolution neural networks (CNN), which were inspired by the principles of the human visual cortex, it was challenging for traditional computer vision methods to identify hand-tuned features for either image object detection or depth estimation. Therefore, it is appealing to conduct research on deep learning approaches which are known to have the potential of learning sophisticated features.

Based on this intuition, related work has been done using RGB-D images

as input for image object detection over the past few years. Utilizing sensors that collect depth information from their environment, such as sonar sensors, the Microsoft Kinect range camera and LiDAR, an increasing number of image detection datasets containing depth ground truth became available for research purposes. However, certain constraints still exist due to the limited quality of depth information provided by the aforementioned sensors.

In general, two major factors need to be considered when acquiring the depth information of objects and environments: **(1) density of depth** (how densely depth labels are assigned to each pixel), and **(2) range of depth** (what is the largest depth that can be measured by the sensor). As for factor **(1)**, sensors like Kinect may provide high density for mapping the depth label to each pixel captured by a camera. Datasets such as the RGB-D object dataset [7] and the NYU V2 dataset [8] are great examples of using Kinect to obtain very dense depth ground truth. However, they are limited by the inability to provide depth information for anything more than short range ($\leq 6m$). On the other hand, there are sensors which are laser-based (e.g. LiDAR) for measuring depth in a very large range ($\geq 30m$). But these sensors are expensive and typically they can only provide sparse depth maps. Therefore, for tasks like providing depth information for self-driving cars where both high density and large range of depth are preferred, solely relying on sensor measurement is insufficient. As a result, learning based methods are becoming a popular approach to obtain depth information.

Taking the above considerations into account, we ask the questions of whether it is possible to design a deep learning model that can learn from both object detection and depth estimation, and internally, if the parameters learned by different tasks contribute to improving the performance of each other. This thesis was inspired by the work of Mask RCNN [9], and proposed a multi-task Depth-aware RCNN (DaRCNN) framework that performs both object detection and depth prediction from a single RGB image input. In this thesis, we will show that through an innovative method of sharing the feature extraction backbone between the two tasks, and integrating depth estimation parameters into the object detection task, the object detection task shows improvements in both recall rate and precision, and outperforms models that only conduct object detection on the KITTI detection benchmark [10]. We will also show that the depth branch weights learned from KITTI datasets can be generalized and improve object detection accuracy on images from

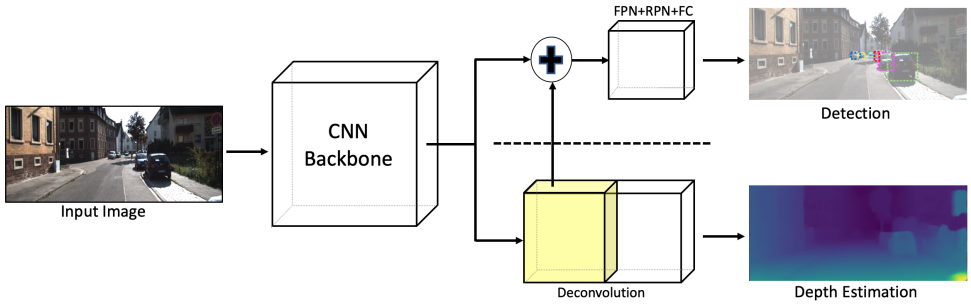

Figure 1.4: Block diagram of Our Depth-aware RCNN (**DaRCNN**) framework for object detection and depth prediction. The input image is fed into CNN backbone structure, and the features extracted are shared between object detection and depth estimation branches. Also, part of the deconvolution layers (as shaded in yellow in the diagram) from the depth estimation branch are fed into the detection branch.

the Cityscapes dataset [11], which has similar scene structures and object classes with the KITTI dataset. A block diagram representing our proposed model is shown in Figure 1.4.

CHAPTER 2

RELATED WORK

2.1 Object Detection

Region-based CNN: There is a large literature that focusing on region-based dense object detection from images. While some of the widely applied object detection methods use region proposal methods (e.g. selective search [5] or EdgeBoxes[6]) to propose region of interest (RoI) in the image in the first step, and then use deep neural network to classify each RoI in bounding boxes. Methods like Faster-RCNN [12] extract features of the image through convolution layers, and then embed a regional proposal network in and end-to-end architecture, and finally use the regional proposal to perform RoIPooling on the extracted feature to perform the final classification (see Figure 2.1). Mask RCNN [9] has built its extension based on the Faster-RCNN's efficient and robust framework. In their paper, He *et al.* proposed to add a fully convolutional network (FCN[13]) branch in parallel to the region proposal and classification branch to output a binary segmentation mask for each RoI. Further, they attempted adding keypoint detection branch to predict human poses. Furthermore, they explored using a feature pyramid network (FPN [14]) as the backbone architecture to obtain feature maps that are semantically strong in different scales. Finally, they innovatively applied the RoIAlign technique to reduce the effect of pixel misalignments caused by the original RoIPooling layers.

RGB-D object detection: Another perspective of image object detection is to integrate depth as a complementary channel to the original RGB image. For the purpose of detection, depth information may come in different forms, such as a point cloud, voxel representation or encoded HHA format [15]. Based on various formats, different approaches are required to perform feature extraction from the inputs. In the case of using point clouds or voxel

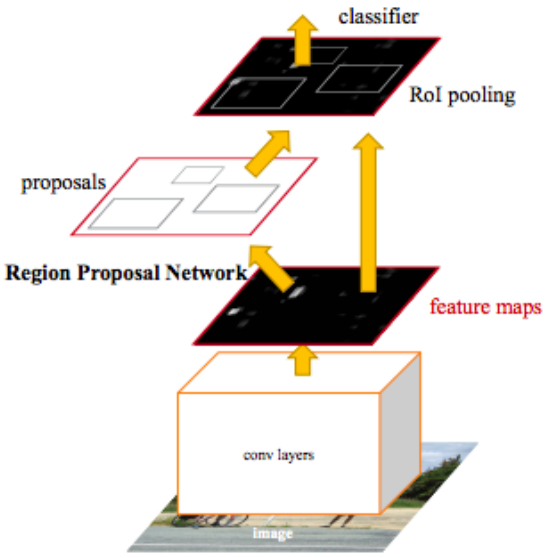

Figure 2.1: An illustration of Faster-RCNN. Image credit: the original paper [12].

representation, where pixels of the original image can be directly mapped to a 3D spatial coordinate in a given environment, methods such as [16, 17, 18] use 3D convolution kernels to perform feature extraction from a 3D spatial input. In other cases, the raw depth data may come in the form of using a single value for each pixel to represent the distance between the physical object and the depth sensor. Such depth information is represented as an extra image channel, or can be further encoded into the HHA format [15], in which it translates the depth map into three channels: embeds height above the ground, horizontal disparity and angle with gravity for each pixel. After encoding the depth information, standard convolutional network such as VGG [19] and ResNet [20] can be applied to these extra channels to extract features. Now with the feature extracted, methods introduced in previous sections can be applied to predict the object bounding box and class. In one of the most recent works, Wang *et al.* [21] further developed the idea of extracting features from depth map by adding depth aware operations in the convolutional network. By integrating checks on depth similarity between pixels in the process of information propagation, they incorporated geometry seamlessly into their CNN.

2.2 Monocular Depth Estimation

Originally, depth prediction from image data relied on multi-view stereo frames, structure from motions, or binocular frames. These methods aim to recognize differences (normally the disparity or keypoint movements) within pairs of consecutive views, and estimate depth through solving triangulation geometry equations. Nevertheless, such methods assume that more than one observation of the environment is available to the estimation model. It has been studied that when a single image of the scenery is provided, average humans perform very well at monocular depth prediction by utilizing cues such as perspective, size of familiar objects, occlusions in the scene, and lighting conditions of object appearance [3].

Many approaches have been presented which try to perform monocular depth estimation on a single-frame image. One of the earliest investigation studied monocular depth prediction by formatting the input image into super-pixels, and inferring depth and 3D orientation of super-pixels through learned plane coefficients combined with Markov Random Fields (MRF) [22]. In addition to this, there exist many related works [23, 24, 25] that depend heavily on hand-tuned features. The major disadvantage of such methods is they do not generate very realistic outputs because those manually tuned features lack global context of the environment. Instead of using hand-tuned features, Liu *et al.* [26], used CNN to extract features and formulated the optimization as a Conditional Random Field (CRF) with variable potentials. But this approach still relies on assumptions, such as similar RGB regions imply also similar depth estimations, and are not always practical in real-world applications.

More recently, a group of related works fully utilize deep learning models for depth prediction. Eigen *et al.* [27] have shown that by training an end-to-end deep neural network using depth map as ground truth, it is possible to produce pixel-wise depth estimation. Unlike other previous related research, methods fully using deep neural networks do not have strong assumptions of the input image. Various works [4, 28, 29] were developed upon such logic and designed networks that follow an encoding-decoding paradigm to perform regression learning. Particularly, our depth estimation branch relies on the work introduced by Laina *et al.* [28], which defines a fully convolutional residual (FCR) up-convolution/up-projection blocks to deconvolve extracted

features into a full resolution depth map.

2.3 Multitask Learning on Depth and Detection

In the work presented by Cao *et al.* [30], the authors proposed a two-stage model for depth prediction and object detection tasks. The first stage contains a deep neural network trained to predict depth from an RGB image, while the second stage combines the predicted depth map with the original RGB image and feeds them into an RGB-D detection network. One disadvantage of such design is that, the two networks in each stage have very limited weights shared, and training on one task would not have effect on the other task. As proven by works such as Mask-RCNN, multitask learning frameworks may benefit from synergistic effects by sharing parameters among different tasks. Research has been attempted to utilize multitask learning to simultaneously perform object detection and image depth prediction from RGB images. In a very recent work presented by Chen *et al.* [31], they have proposed a DSPNet for joint detection, depth estimation and semantic segmentation on driving scene images. In their design, the network produces a single depth value in meter for each detected object, and the model uses a single architecture for both detection and instance-level depth estimation. This means that the final output of their network would not estimate depth for backgrounds (such as roads and buildings) and objects that are not detected by the network. Compared with their method, our model outputs pixel-wise depth estimation, and we utilize two different network architectures that can inference in parallel for the depth estimation and the object detection tasks.

CHAPTER 3

METHODOLOGY

In this chapter, details of the proposed architecture (see Figure 3.1) are explained. For clarity, the convolutional backbone structure used to extract features from the RGB input image are described in Section 3.1, and then the task branch architectures for object detection and depth prediction are presented in Sections 3.2 and 3.3 respectively. Finally, the loss functions applied in the optimization of each task is shown in Section 3.4.

3.1 CNN Backbone

The CNN backbone performs as contraction layers with pyramidal structures. With such a design, features extracted in lower levels have smaller receptive fields and are semantically weaker but come with higher resolution, while features extracted in higher levels have high receptive fields and are semantically stronger but come with lower resolution. In this research, we evaluate our network based on ResNet [20] of 101 layers (ResNet101). Additionally, we evaluate a more effective backbone structure, the feature pyramid network (FPN) [14], which through lateral connections, combines low-resolution features with high-resolution features to obtain a feature pyramid that is semantically strong at different scales.

Specifically, for Resnet101 [32] we denote the features extracted by each stage's last residual block, which are conv1, conv2, conv3, conv4 and conv5 as $\{C_1, C_2, C_3, C_4, C_5\}$. And for the FPN backbone, layers from ResNet101 and the depth estimation branch are used as the bottom-up pathway. We use the same notation as the original paper [14] and denote the final set of feature maps to be $\{P_2, P_3, P_4, P_5\}$, where we do not use P_1 which corresponds to conv1 because of its large memory footprint.

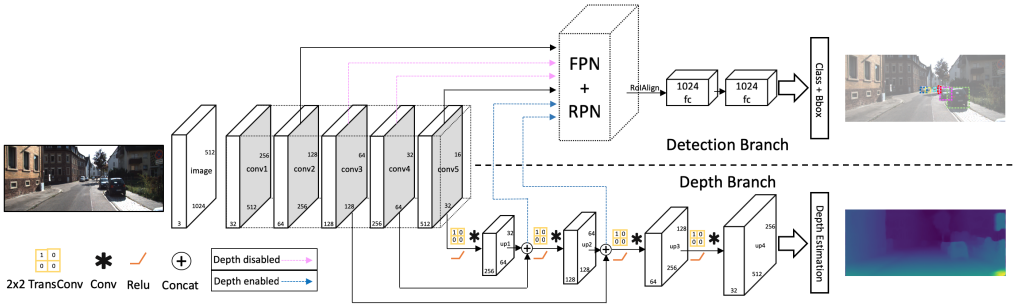

Figure 3.1: **Network architecture.** Our proposed model contains three major parts. The **CNN backbone** encodes the input RGB image, and features extracted here are used by the **object detection branch** and the **depth prediction branch**. The detection branch utilizes FPN [14] and RPN [12] for object detection, and the depth prediction branch uses up-convolution/up-projection blocks [28] to decode features into depth. In addition, we add skip connections to concatenate feature layers $conv3, conv4$ with decoding layers $up1, up2$ respectively in the depth branch, and forward the concatenated layers to the FPN in the detection branch.

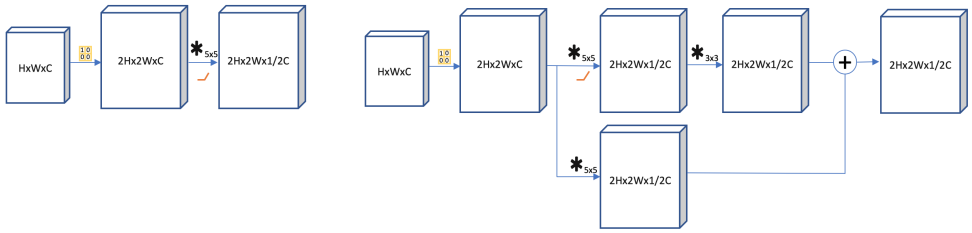

Figure 3.2: **Up-convolution and up-projection blocks.** The diagram shows more details of the referenced (a) up-convolution and (b) up-projection blocks from [28].

3.2 Depth Prediction Branch

Recent approaches of using deep neural network for monocular image depth prediction have shown significant progress. Among those methods, many have designed their networks that conform to the encoder-decoder paradigm [27, 33, 4, 28, 29]. In our design, the CNN backbone which extracts image features plays the role of encoder, and we explore two simple architectures introduced in [28] for decoding as illustrated in Figure 3.2.

The first decoding architecture we explore is the up-convolution [28] structure. The up-convolution layers are divided into several blocks that share the same layer sequence. In each up-convolution block, the input feature maps are fed into the unpooling layer as described in [13], and is implemented with a 1×1 transpose convolution operation with stride equal to 2. This unpooling layer doubles the size of input along both width and height. Then, the unpooling layer is followed by a 5×5 convolution layer and a ReLU activation layer to generate the output.

The second decoding architecture we evaluate is the up-projection networks as described in [28]. Again, this architecture is divided into blocks. The only difference between up-projection blocks and the up-convolution blocks is that, up-projection blocks have *projection* residual connection from input to output.

The depth branch of our final design stacks four of either up-convolution or up-projection blocks. In this thesis, we denote the output layers from each decoding blocks to be $\{U_1, U_2, U_3, U_4\}$. The final output size is 16x of the scale of feature maps generated at C_5/P_5. Also, in addition to the original design proposed in [28], we introduced skip connections from ResNet output to the decoding layers. In our design, the inputs to generate U_2 and U_3 are $U_1 + C_4$ and $U_2 + C_3$ respectively (see Figure 3.1).

3.3 Object Detection Branch

Our object detection branch follows the two-stage procedure used in the Faster-RCNN detector [32]. In the first stage, a region proposal network (RPN) is applied to propose bounding boxes which contain candidate objects. Then for the next stage, the detection network uses RoIAlign which was

proposed in Mask RCNN [9] to extract features from each bounding box, and finally it performs classification and bounding box regression. Because there exist a large number of small objects in the KITTI detection challenge dataset [10], choosing RoIAlign instead of RoIPooling helps to reduce misalignment when generating the final feature map for classification. For comprehensive comparisons between Mask/Faster-RCNN and other detection frameworks, please refer to [34, 9, 35]. The major difference in our design is that unlike Mask-RCNN, whose FPN [14] is built based on only ResNet features, our model may use both ResNet layers and depth decoding layers to build the feature pyramid. When depth branch is disabled, the FPN in the detection branch are fed with features from ResNet only. However, when depth branch is enabled, P_4 is generated from $\{P_5, U_1 + C_4\}$ and P_3 is generated from $\{P_4, U_2 + C_3\}$. The second design allows high semantic features trained for decoding depth to interact with the object detection task, which is similar to add features extracted from the depth channel to the original RGB features.

3.4 Loss Function

As for training our model, two sets of loss functions are applied. For the detection branch, the loss is calculated on each sampled RoI as $L_{RoI} = L_{cls} + L_{box}$, which are identical to the classification loss L_{cls} and bounding box loss L_{box} defined in [34].

On the other hand, the regression loss L_{depth} is defined as the squared $l2$ norm between prediction $\hat{y}$ and y: ground truth:

$$L_{depth} = ||\hat{y} - y||_2^2 \tag{3.1}$$

This loss function is reported to generate good results in [28] for the depth prediction branch architecture.

In addition to using L2 loss for the depth estimation training, there exists a more robust berHu [36] loss which can be used for the depth estimation learning task. The berHu loss is defined as

$$x_i = \hat{y}_i - y_i \tag{3.2}$$

$$c = \frac{1}{5} \max_i |x_i| \tag{3.3}$$

$$\mathcal{B}(x_i) = \begin{cases} |x_i|, & \text{if } |x_i| \leq c \\ \frac{x_i^2 + c^2}{2c}, & \text{otherwise} \end{cases} \tag{3.4}$$

In general, the berHu loss is a reversed L1-smooth loss, and is more sensitive to small losses. We would like to further experiment with this loss for more improvements in the future.

CHAPTER 4

EXPERIMENTS

In this thesis, the effectiveness of both object detection and depth prediction branches are evaluated, and the proposed model is compared with various single-task methods focusing either on object detection or depth estimation. Because the training of the proposed network requires ground truth of both object labels and depth of the scenery, existing image datasets, such as [37, 38, 11] which only focus on image object detection, or datasets like [8] which only focus on scenery depth prediction, are not suitable for training our model. As an alternative, we evaluate our methods using the challenging KITTI object detection dataset [10].

4.1 Network Configuration

In our network configuration, we resize the input images so that their longer edge is set to 1024 pixels, and their shorter edge is first resized to preserve the input image aspect ratio, and then padded with 0 so the final input image has a resolution of 1024×512 pixels. One issue we observe with this setup is that, even though the padding is necessary for the contraction layers in the backbone to work, it introduces garbage features. Although this would not cause a big issue for object detection training, as only regions proposed containing objects will be examined, those unnecessary features lead to meaningless depth prediction.

Therefore, in order to remove the effect of padding zeros for predicting depth, the final output from the depth estimation branch is cropped back to the original image racial, and the loss is only calculated from pixels that have non zero depth ground truth. With the input image having resolution 1024×512, the resolution of output image from depth estimation branch is 512×256, and after cropping, the final predicted depth map's resolution is

512×160 to follow the original aspect ratio of raw KITTI images.

4.2 Training

Dataset. The KITTI detection benchmark [10] provides 7,481 training images and 7,518 testing images. For our analysis and evaluation on the task of object detection, we follow the advice from [39] to split the provided training images into 3,682 images and 3,799 images for training and validation respectively. During training, images were randomly flipped horizontally for data augmentation. For the depth prediction branch evaluation, we no longer use the image provided in the detection benchmark, but focus our training on the dataset for the depth prediction benchmark. We notice that although the depth prediction dataset provides sparse depth ground truth generated by LiDAR sensors, the provided labels are not dense enough to train our model. Fortunately, training images released in this benchmark were captured from the left camera mounted on a car, and each left image can be paired with an image captured from the right camera as a stereo image pair. In this way, we were able to obtain disparity maps for training images. By using the camera calibration that came along with the benchmark, and align with the sparse depth ground truth, we reconstruct a pixel-level dense depth map for each training and validation image.

Training Parameters. Our model is trainable using only one Nvidia 1080TI GPU with a reasonable amount of time, and we choose to use a batch size of two training image based on the given GPU memory. The training of our model contains three stages in total to fully utilize different datasets for each task branch, albeit these two branches can work in parallel to generate results at inference time. The training procedure of our models can be specified using the following steps: (1) train only the depth prediction branch on the KITTI depth prediction dataset until L_{depth} converges, (2) train the object detection branch with the depth branch disabled on KITTI 2D detection dataset until $L_{detection}$ converges, and (3) train with both branches enabled using a 2D detection dataset until $L_{depth} + L_{detection}$ converge. In stage (1), we use initial learning rate of 0.005, and weight decay of 10% after every 5k iterations. In stage (2), we follow the training procedure introduced in [34]. Here we use the KITTI evaluation standard that, in order to be con-

Table 4.1: Results of depth prediction between our method and benchmark. For the reported evaluations RMSE, lower is better, where as for the accuracies $\delta_i \leq 1.25^i$, higher is better.

KITTI Train/Val Depth				
Method	RMSE	δ_1	δ_2	δ_3
Eigen *et al.*[27]	7.391	0.563	0.724	0.837
Monodepth [4]	4.471	0.635	0.789	0.872
FCR(up-conv) [28]	1.267	0.721	0.870	0.958
FCR(up-proj) [28]	1.011	0.758	0.906	0.964
DaRCNN(up-conv)	1.186	0.801	0.936	0.974
DaRCNN(up-proj)	**0.922**	**0.828**	**0.948**	**0.986**

sidered as a positive sample, it requires the intersection of union (IoU) ≥ 0.7 for cars, and IoU ≥ 0.5 for pedestrians and cyclists. We start our training with pretrained ResNet101 weights on the COCO dataset [37]. The shared CNN backbone is updated in stages (1) and (3), while the task branches' weights are updated only when used for inference. The learning rate of the detection branch is initialized at 0.001. We use the momentum 0.9 and the weight decay 0.0001. And in the final stage, the two branches are enabled simultaneously using an initial learning rate of 0.0005 with same momentum and weight decay used in stage (2). The final model is obtained after 200K iterations from all three stages, and it takes approximately 17 hours on a single 1080TI GPU.

4.3 KITTI Depth Estimation Results

As the depth prediction branch in our DaRCNN is built based on the fully convolutional residual architecture (FCR) proposed in [28], we mainly compare our model with the original FCR model, and several other methods that were used as comparison reported in [28] in Table 4.1. Some of the results shown in this table are different from the number reported in their original paper. This is due to the choice of dataset used for evaluation. In those works, they have used the Eigen split on the KITTI raw dataset as described in [27], which uses 687 test images and uses 22,600 images for training, which is very different from the new train/validation split provided by the KITTI

benchmark. From the comparisons in Table 4.1, we observe that when the detection branch of our DaRCNN is disabled during training and inference, the performance of this model is very close to the FCR implementation with the same CNN backbone structure. This confirms that our depth prediction branch can achieve the same performance as indicated by [28]. In comparison, when we co-trained our depth branch with the detection branch, our model achieved higher accuracy. This illustrates that using a shared backbone structure in our network design would not have any adverse effect on the depth prediction branch.

4.4 KITTI Detection Results

We evaluate and compare our DaRCNN detection results on both the KITTI detection test set and our own train/validation split. The comparison results on train/validation split is in Tables 4.2, 4.3 and 4.4, and the results on submitted KITTI test detection is shown in Table 4.5. Both instantiations of our model (DaRCNN with up-conv or up-proj) outperform the baseline model (Faster-RCNN + FPN [12, 14]) which our models are built based on.

Also, comparison of detection results between Faster-RCNN, DaRCNN with up-projection and the depth prediction of DaRCNN are visualized in Figure 4.1. By going through the validation image outputs, we noticed that DaRCNN performs better detection on objects that are comparatively farther away from the camera. These objects, such as pedestrians and cyclists that are far away cover only very small regions of pixels in the original image, and could be mistakenly recognized as background. But our visualization of the depth prediction shows different depth on these regions compared to the background. This verifies that the encoded features learned by depth branch are activated on such objects. By sharing these features with RPN, it is more likely to result in region proposals of these projects for the final classification network. A comparison of recall rate of each object category is shown in Table 4.6. Both of our models show a higher recall rate compared to the baseline Faster-RCNN, especially in the pedestrian and cyclist categories.

Table 4.2: Car detection results between different detection methods on our train/validation set under three difficulties: easy, medium (med) and hard.

KITTI Train/Val Car Detection (AP%)				
Methods	Backbone	easy	med	hard
Faster-RCNN	101	87.83	86.29	76.27
Faster-RCNN	101+FPN	88.74	87.21	77.64
DaRCNN(up-conv)	101+FPN	90.16	89.02	78.72
DaRCNN(up-proj)	101+FPN	**90.47**	**89.52**	**80.26**

Table 4.3: Pedestrian detection results between different detection methods on our train/validation set under three difficulties: easy, medium (med) and hard.

KITTI Train/Val Pedestrian Detection (AP%)				
Methods	Backbone	easy	med	hard
Faster-RCNN	101	76.27	75.85	63.91
Faster-RCNN	101+FPN	77.64	77.48	66.17
DaRCNN(up-conv)	101+FPN	80.91	69.27	58.83
DaRCNN(up-proj)	101+FPN	**90.47**	**89.52**	**80.26**

Table 4.4: Cyclist detection results between different detection methods on our train/validation set under three difficulties: easy, medium (med) and hard.

KITTI Train/Val Cyclist Detection (AP%)				
Methods	Backbone	easy	med	hard
Faster-RCNN	101	72.82	58.73	57.47
Faster-RCNN	101+FPN	74.82	59.09	59.93
DaRCNN(up-conv)	101+FPN	77.27	65.72	**62.96**
DaRCNN(up-proj)	101+FPN	**77.91**	**68.20**	62.79

Table 4.5: Detection results of medium level objects on the KITTI Test set.

KITTI Test medium objects results (AP%)			
Method	Car	Pedestrian	Cyclist
Faster-RCNN-FPN	86.83	66.92	63.82
DaRCNN(up-proj)	**89.56**	**67.72**	**67.39**

Table 4.6: Results of medium level objects recall rate. The depth prediction branch participates in the training, and improvements on the recall rate of the final detected object can be observed.

KITTI Train/Val Medium Object Recall (%)			
Method	Car	Pedestrian	Cyclist
Faster-RCNN-FPN	88.62	60.45	55.74
DaRCNN(up-conv)	89.21	64.23	60.82
DaRCNN(up-proj)	**90.76**	**64.82**	**61.04**

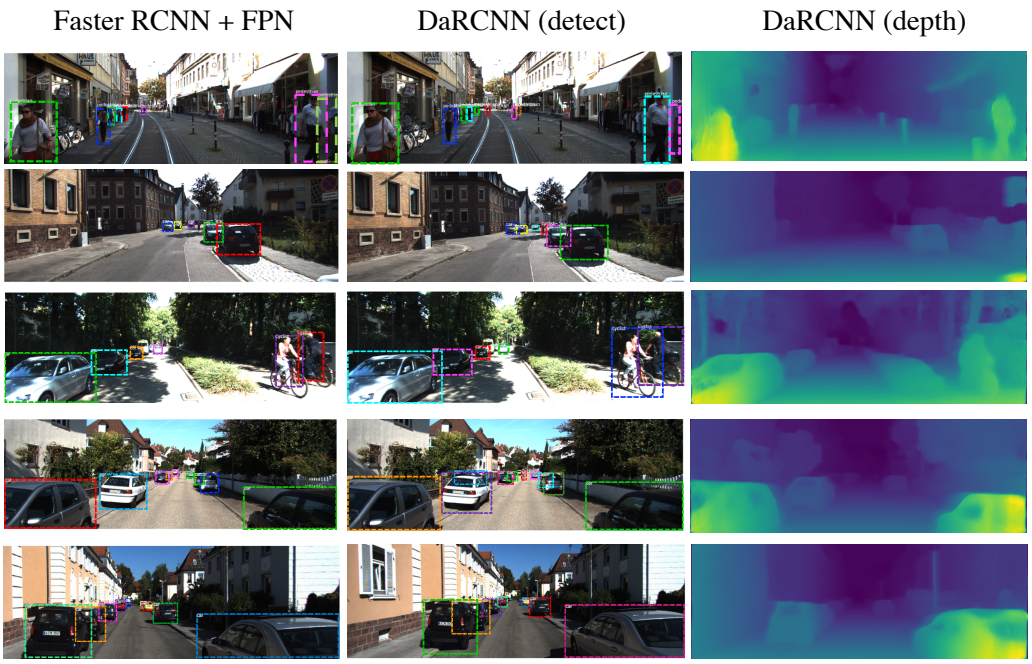

Figure 4.1: The left two columns show some detection results between Faster-RCNN + FCN [12, 14] *vs.* DaRCNN (with up-projection) on KITTI validation images. It is notable that our detection results can better recognize objects which are relatively further away from the camera. The third column shows the visualization of depth estimation by DaRCNN. The depth maps shown above are resized and interpolated to the same scale of input images.

Table 4.7: Detection results for car, rider and person on the Cityscapes dataset. The DaRCNN with depth uses up-projection for deconvolution.

Cityscapes Object Detection (AP%)				
Methods	Depth	Car	Rider	Person
DSPNet [31]	✓	59.1	**37.7**	34.9
DaRCNN	×	58.8	36.4	34.5
DaRCNN	✓	**59.7**	37.1	**35.2**

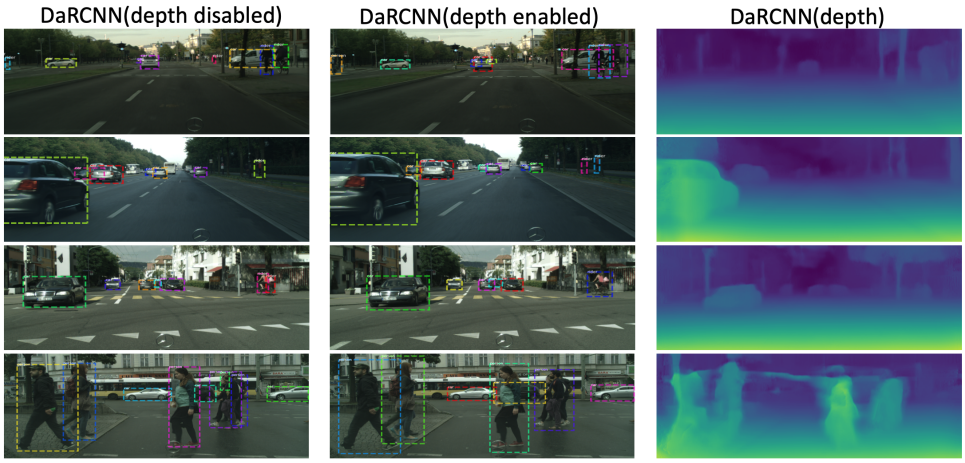

Figure 4.2: In the first three rows, DaRCNN with depth enabled performs slightly better than with depth disabled. DaRCNN without depth enabled mistakenly classified the rider in example (row) 3 into two riders, while with depth enabled, our model mistakenly classified several persons in example (row) 4 into a single bounding box.

4.5 Cityscapes Detection Results

To further show that the learned depth weights can be generalized to driving scene in other datasets, we fine-tuned our model on the Cityscapes dataset [11] without updating depth branch weights, and evaluated on classes similar to those used in our KITTI evaluation with depth branch enabled. As shown in Table 4.7, with the depth weights learned from the KITTI dataset, it improves the performance of detection branch by an average $\sim 0.8\%$ on each category.

Some examples of DaRCNN with and without depth branch enabled are shown in Figure 4.2. It is noticeable that the detection branch gets influence

by the shared depth weights as it tends to group objects clustered together with very high occlusion into one bounding box. This leads to the correct detection on the single rider in example 3, while the model without depth enabled treats part of the back seat of the bike as another rider. However, such behavior may also leads to mistakes such as grouping several person with occlusion into one bounding box in example 4.

CHAPTER 5

CONCLUSION

In this thesis, we presented a conceptually simple and flexible Depth Aware RCNN (DaRCNN), which adds depth prediction as an extension to Faster-RCNN [12]. In our experiments, we adopted Faster-RCNN network architecture as the regional based detection branch, and used a fully convolutional residual (FCR) [28] network with skip connections from the ResNet [20] for our depth estimation branch. By allowing these two branches to share features extracted from both the CNN backbone and part of the decoding layers, we have shown that the object detection branch in our model outperforms Faster-RCNN on the KITTI detection benchmark, and our depth estimation branch achieves similar performance compared to the original FCR architecture. Moreover, we showed that the learned trained depth estimation weights can improve object detection accuracy on driving scenes from Cityscapes datasets. We hope that our proposed multitask learning method provides a new perspective which combines object detection and scenery depth estimation, and a framework that future research in object detection and depth prediction can help to improve.

As for our future work, we plan to focus on designing more sophisticated and efficient network architectures for both object detection and depth prediction branches. We will further confirm that our framework is robust and can achieve better performance if new architectures are used.

REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, Jan. 2016. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2015.2437384

[2] W. Wen and S. Khatibi, "Towards measuring of depth perception from monocular shadow technique with application in a classical painting," *Journal of Computers*, vol. 11, 2016.

[3] I. Howard, *Perceiving in Depth, Volume 1: Basic Mechanisms*, Oxford University Press, 2012.

[4] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," *CoRR*, vol. abs/1609.03677, 2016. [Online]. Available: http://arxiv.org/abs/1609.03677

[5] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, 2013. [Online]. Available: http://www.huppelen.nl/publications/selectiveSearchDraft.pdf

[6] L. Zitnick and P. Dollar, "Edge boxes: Locating object proposals from edges," in *ECCV*, September 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/edge-boxes-locating-object-proposals-from-edges/

[7] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1817–1824.

[8] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *ECCV*, 2012.

[9] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870

[10] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[11] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes dataset for semantic urban scene understanding," *CoRR*, vol. abs/1604.01685, 2016. [Online]. Available: http://arxiv.org/abs/1604.01685

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN : Towards real-time object detection with region proposal networks," in *Advances in Beural Information Processing Systems*, 2015, pp. 91–99.

[13] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014. [Online]. Available: http://arxiv.org/abs/1411.4038

[14] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016. [Online]. Available: http://arxiv.org/abs/1612.03144

[15] S. Gupta, R. B. Girshick, P. Arbelaez, and J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," *CoRR*, vol. abs/1407.5736, 2014. [Online]. Available: http://arxiv.org/abs/1407.5736

[16] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," *CoRR*, vol. abs/1711.06396, 2017. [Online]. Available: http://arxiv.org/abs/1711.06396

[17] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," *CoRR*, vol. abs/1608.04236, 2016. [Online]. Available: http://arxiv.org/abs/1608.04236

[18] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *CoRR*, vol. abs/1612.00593, 2016. [Online]. Available: http://arxiv.org/abs/1612.00593

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[21] W. Wang and U. Neumann, "Depth-aware CNN for RGB-D segmentation," *CoRR*, vol. abs/1803.06791, 2018. [Online]. Available: http://arxiv.org/abs/1803.06791

[22] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 824–840, May 2009.

[23] D. Hoiem, A. A. Efros, and M. Hebert, "Geometric context from a single image," in *10th IEEE International Conference on Computer Vision (ICCV 2005), 17-20 October 2005, Beijing, China*, 2005. [Online]. Available: https://doi.org/10.1109/ICCV.2005.107 pp. 654–661.

[24] B. Liu, S. Gould, and D. Koller, "Single image depth estimation from predicted semantic labels," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010, pp. 1253–1260.

[25] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. C. Platt, Eds. MIT Press, 2006, pp. 1161–1168. [Online]. Available: http://papers.nips.cc/paper/2921-learning-depth-from-single-monocular-images.pdf

[26] F. Liu, C. Shen, G. Lin, and I. D. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *CoRR*, vol. abs/1502.07411, 2015. [Online]. Available: http://arxiv.org/abs/1502.07411

[27] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *CoRR*, vol. abs/1406.2283, 2014. [Online]. Available: http://arxiv.org/abs/1406.2283

[28] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," *CoRR*, vol. abs/1606.00373, 2016. [Online]. Available: http://arxiv.org/abs/1606.00373

[29] F. Ma, G. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera," *arXiv preprint arXiv:1807.00275*, vol. abs/1807.00275, 2018. [Online]. Available: https://arxiv.org/pdf/1807.00275

[30] Y. Cao, C. Shen, and H. T. Shen, "Exploiting depth from single monocular images for object detection and semantic segmentation," *CoRR*, vol. abs/1610.01706, 2016. [Online]. Available: http://arxiv.org/abs/1610.01706

[31] L. Chen, Z. Yang, J. Ma, and Z. Luo, "Driving scene perception network: Real-time joint detection, depth estimation and semantic segmentation," *CoRR*, vol. abs/1803.03778, 2018. [Online]. Available: http://arxiv.org/abs/1803.03778

[32] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 379–387.

[33] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," *arXiv preprint arXiv:1809.02165*, vol. abs/1806.02446, 2018. [Online]. Available: https://arxiv.org/pdf/1806.02446

[34] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[35] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikaine, "Deep learning for generic object detection: A survey," *arXiv preprint arXiv:1809.02165*, 2018. [Online]. Available: https://arxiv.org/pdf/1809.02165.pdf

[36] A. B. Owen, "A robust hybrid of lasso and ridge regression," Tech. Rep., 2006.

[37] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[38] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, June 2010.

[39] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3d voxel patterns for object category recognition," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2015.