UTILIZATION OF DENSE DEPTH INFORMATION FOR MONO-VIEW OBJECT DETECTION AND INSTANCE SEGMENTATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÇAĞLAYAN CAN ÇAKIRGÖZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

MAY 2022

Approval of the thesis:

UTILIZATION OF DENSE DEPTH INFORMATION FOR MONO-VIEW OBJECT DETECTION AND INSTANCE SEGMENTATION

submitted by **ÇAĞLAYAN CAN ÇAKIRGÖZ** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. A. Aydın Alatan
Supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Ahmet Oğuz Akyüz
Computer Engineering, METU _____

Prof. Dr. A. Aydın Alatan
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Fatih Kamışlı
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Elif Vural
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Erkut Erdem
Computer Engineering, Hacettepe University _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Çağlayan Can Çakırgöz

Signature :

ABSTRACT

UTILIZATION OF DENSE DEPTH INFORMATION FOR MONO-VIEW OBJECT DETECTION AND INSTANCE SEGMENTATION

Çakırgöz, Çağlayan Can
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. A. Aydın Alatan

MAY 2022, 102 pages

Object detection aims for detecting objects of certain classes in an image by bounding them in rectangular boxes whereas instance segmentation tries to detect objects in pixel level. Deep learning techniques, which have shown great improvements over the last decade, are utilized in these topics as well, and a significant success is achieved against the traditional methods. Similar improvements can be observed in dense depth estimation which deals with deducing dense information of a scene from a single image. Previous works have shown that object detection and instance segmentation performances can be improved by incorporating sensor depth information. This thesis studies whether or not it is possible to have similar improvements when depth information is estimated from images instead of directly provided from sensors. Our research have shown that incorporating estimated depth data results in higher performance in object detection, although it fails in instance segmentation.

Keywords: Object Detection, Instance Segmentation, Convolutional Neural Networks

ÖZ

TEK BAKIŞLI NESNE TESPİTİ VE BÖLÜTLEMESİNDE SIK DERİNLİK BİLGİSİ KULLANIMI

Çakırgöz, Çağlayan Can
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. A. Aydın Alatan

Mayıs 2022 , 102 sayfa

Nesne tespiti; bir görüntüde bulunan, belirli sınıflara ait nesneleri dikdörtgen sınırlar içerisine alarak saptamayı hedefler. Örnek bölütleme ise nesneleri piksel düzeyinde saptamaya çalışır. Geçen on yılda büyük bir gelişme gösteren derin öğrenme tekniklerinden bu konularda da yararlanılmış ve geleneksel yöntemlere karşı önemli başarılar kazanılmıştır. Bir sahnenin derinlik bilgisinin tek görüntü üzerinden çıkarsanmaya çalışıldığı tek bakışlı sık derinlik tahmini konusu da yine derin öğrenme teknikleriyle büyük gelişme göstermiştir. Nesne tespiti ve bölütleme konularında sensörden toplanan derinlik bilgisinin de dahil edilmesinin başarımı artırdığı daha önceki çalışmalarla ortaya konmuştur. Bu tezde de, görüntülerden tahmin edilen derinlik bilgisinin örnek bölütleme ve nesne tespitinde başarım artışını sağlayıp sağlayamayacağı araştırılacaktır. Araştırmaların sonucunda, tahmini derinlik bilgisi kullanarak nesne tespitinde başarım artışı sağlanmış ancak örnek bölütlemede aynı kazanım elde edilememiştir.

Anahtar Kelimeler: Nesne Tespiti, Örnek Bölütleme, Evrişimsel Sinir Ağları

To Efe

ACKNOWLEDGMENTS

First and foremost, I would like to present my special thanks to my supervisor Prof. Dr. A. Aydın Alatan for his unparalleled support he has given me since day one. In the past three years, I have not seen nothing but his kindness and sensibility. When I hit the lowest point in my life, his encouragement helped me get up on my feet. The journey that I began by admiring his intelligence and intellectuality has now come to an end where I ended up admiring his personality even more.

I would like to thank my family members, especially my little sister. She was there for me when everything went all wrong, and supported me through the darkest times of my life. Despite all the concerns she has, I know that the life ahead of us will now be brighter than ever. I would also like to thank my lovely aunts Ayşen Yılmaz and Fatma Yılmaz who have given me so much support, respect and pure love since I was born which makes me feel incredibly lucky and proud to be their son.

I should mention all my friends who have always been there for me. I would like to thank Demet, Kübra and Şevval Tül for their twenty-year-long genuine friendship. It is very fortunate of me to have such friends that I can get into contact and trust whenever I am in need. I do not know how to thank Oğul Can and Yeti Ziya Gürbüz enough for their priceless pieces of advice. They have always been the first address that I called on whenever I ended up in deadlock. I would also like to thank İhsan Emre Üstün for his constant support. After going through all the struggles together, I am glad that we rode out 2020 as of now.

Lastly, I would like to thank my one and only Maya, who revived me and my family with her existence.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
AP	Average Precision
CNN	Convolutional Neural Network
fc	Fully-Connected Layer
FPN	Feature Pyramid Network
mAP	Mean Average Precision
ResNet	Residual Network
RoI	Region of Interest

CHAPTER 1

INTRODUCTION

1.1 Motivation

Object detection problem deals with the classification and bounding box level localization of object instances in an image whereas instance segmentation problem aims at the classification and pixel-wise localization of the same instances. As two fundamental problems of computer vision, they have always been a hot spot for researchers. Thanks to the innovations in hardwares, e.g., GPU, and neural network models, e.g. CNN, a great amount of novel solutions with significant improvements have been proposed in the last decade. For instance, deep learning-based RGB-D object detectors and instance segmentation models have been proposed to fuse both color and depth information through neural networks. It has been shown that detection and segmentation models can be improved by using RGB and depth data together rather than RGB only. However, depth information is not available for most of the time; therefore, RGB-D based models are not practical for real-time applications.

Development of deep neural networks also paved the way for efficient single image depth estimation models. Traditionally, depth information is obtained either directly from sensors, e.g. LIDAR, or through stereo vision using two cameras. With the help of deep neural networks, depth of a scene can now be extracted quite successfully from only one RGB image.

Now that depth information can be retrieved successfully from RGB images, the scarce of depth information can be resolved by using the estimated depth information rather than sensor data, and existing RGB-D models can replace standard RGB-only object detection and instance segmentation models.

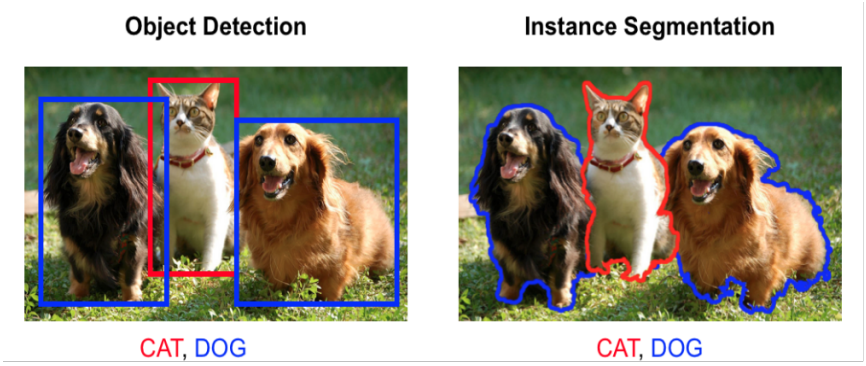


Figure 1.1: Comparison between object detection and instance segmentation.

Figure 1.2: An example for depth estimation from a single RGB image. [45]

1.2 The Scope of the Thesis

We analyze a couple of early fusion techniques for fusing RGB image with its estimated depth mask, and we test the effect of these techniques on the object detection and instance segmentation performances. First, we analyze the concatenation technique in which estimated depth mask is concatenated to the RGB image as the 4^{th} channel. Concatenated RGB-D images are directly given as inputs to state-of-the-art object detection and instance segmentation models. As for the second technique, convolution is used for fusion. A concatenated RGB-D image is first fed to a CNN model which maps this 4-channel input to a 3-channel feature map. Then, this extracted feature map is given as input to state-of-the-art object detection and instance segmentation models. Various state-of-the-art object detection and instance segmentation architectures are modified with the proposed methods, and their effects are discussed by comparing the performances of the modified and unmodified models.

1.3 The Outline

Technical background for object detection, instance segmentation, convolutional neural networks, and feature fusion techniques can be found in Chapter 2. Later in Chapter 3, we present an overview of the works that have been done in both RGB and RGB-D object detection areas. Following that, the literature review of instance segmentation can be found in Chapter 4, and we give the literature review of deep learning-based single image depth estimation in Chapter 5. In Chapter 6, proposed methods and conducted experiments are explained in detail. Finally, we conclude the thesis with summary and future works in Chapter 7.

CHAPTER 2

BACKGROUND

This chapter presents an overview of the object detection problem and the technical background required for the following chapters. Section 2.1 introduces the fundamental visual recognition related computer vision problems. Convolutional neural networks are studied in Section 2.2, followed by explaining backbone architectures in Section 2.3. Finally, section 2.4 discusses the feature fusion techniques.

2.1 Overview of Visual Recognition Problems

The computer vision field involves various fundamental visual recognition problems such as image classification [26], object detection [20], instance segmentation [25], and semantic segmentation [7]. The purpose of image classification is to classify an image according to the object or objects of the same class present in that image. Regardless of the location, area and the number of objects, the whole image is labeled based on the recognized object class. The image given in Fig. 2.1(a) is labeled as cow although there are three cows present in the image. The location and the area of each cow is not known either.

Object detection takes image classification further and in addition to recognition, it also aims localization of each object within the image. Localization is realized by finding the minimum rectangular region that covers the object completely. In the image given in Fig. 2.1(b), each of the three cows are individually recognized and located by their respective rectangular frames.

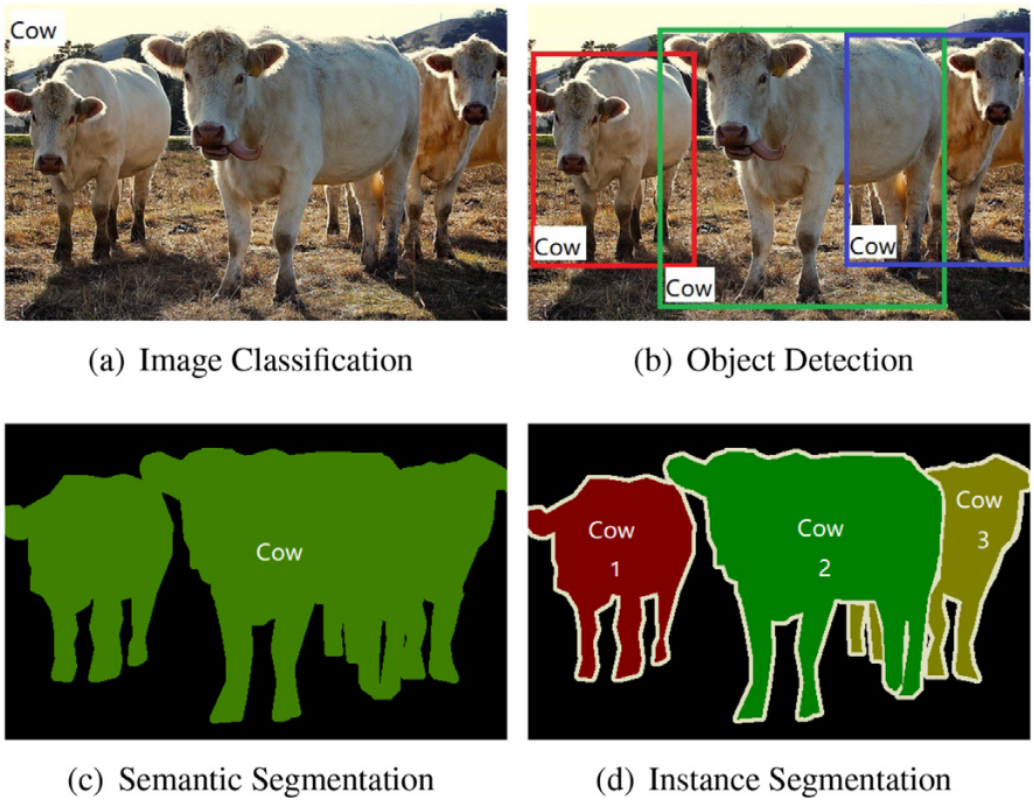


Figure 2.1: Comparison of different computer vision problems. [73]

The objective in semantic segmentation is to predict class labels of each pixel to a class label including background. However, multiple objects of the same class are not differentiated in semantic segmentation. In the image given in Fig. 2.1(c), each pixel of cows are labeled as cow although there is no information about the number of cows and which pixel belongs to which cow.

Instance segmentation tries to answer these questions by fusing object detection with semantic segmentation. Instance segmentation aims to detect each object within the image as in object detection, and then assigns pixels of object regions to class labels as in semantic segmentation. In the image given in Fig. 2.1(d), each cow is detected as in object detection; however, it takes it further and determines the true shape of cows by pixel-level localization instead of rectangular regions.

As the scope of thesis covers only object detection, the primary focus in the following sections is given on its the problem definition of object detection and its performance metrics.

2.2 Convolutional Neural Networks

The history of neural networks can be traced back to 1947 [51]. In their work, Pitts and McCulloch tried to put out a principled approach to solve learning problems. With the introduction of back-propagation algorithm in 1986[57], neural networks became popular in late 1980s and early 1990s. However, they lost their popularity because of limited computation power and scarce of training data. During 2000s, these problems were solved by the publication of large datasets, e.g. ImageNet, and development of parallel computing systems with high performance, e.g. GPU clusters. Thereby, neural networks have become popular once again and remains so even today.

Neural networks with deep layers are referred to as deep neural networks, and convolutional neural networks (CNN) can be regarded as the most representative ones among all deep models [38]. Even though fully connected neural networks can also be used for feature learning and classification tasks, it is impractical for larger inputs such as images because it requires a very large number of neurons to be trained. Convolution instead decreases the number of free parameters which makes it possible to train deeper architectures.

A CNN architecture is built by several different types of layers are used, e.g., convolutional layers, activation layers, pooling layers, etc. convolutional layers have a set of trainable kernels for convolution. A neuron of a convolutional layer is connected to a limited region of the previous layers, which is called as the receptive field of a neuron. A convolutional neuron performs convolution over its receptive field, which is basically the dot product of a convolutional kernel with the input matrix extracted from its receptive field. In this way, all neurons of a convolutional layer form a 2D feature map per a convolutional filter. The whole set of trainable kernels of a convolutional layer forms a 3D feature map, which is output by the convolutional layer. In order to introduce non-linearity to the model, an activation layer applies a non-linear function on the output feature map of a convolutional layer. The commonly used activation functions are rectified linear unit (ReLU), sigmoid, and hyperbolic tangent, which are given in Equation (2.1), Equation (2.2), and Equation (2.3), respectively.

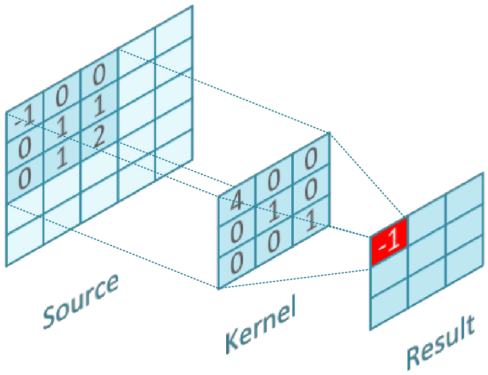


Figure 2.2: 2D convolution with a $3x3$ kernel.

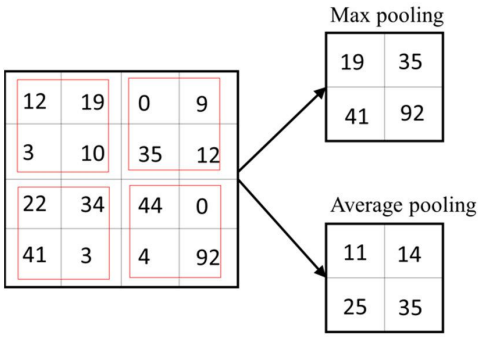


Figure 2.3: Pooling with a $2x2$ kernel and stride=2.

$$ReLU : f(x) = max(0, x) \tag{2.1}$$

$$Sigmoid : \sigma(x) = \frac{1}{1 + e^{x}} \tag{2.2}$$

$$Hyperbolictangent : tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.3}$$

CNN gives more importance to a feature's relative location with respect to the other features than than precise location of features. Moreover, removing these redundant features reduces overfitting, and decreases the number of trainable parameters and computational costs. Thereby, CNNs aim at decreasing the spatial resolution by keeping only the salient features of a feature map. This non-linear down-sampling operation is called as pooling. The most common pooling operations are max-pooling and average-pooling. In max-pooling, the input feature map is divided into grid cells and the maximum element in each grid cell is output. In average-pooling, the input feature map is divided into grid cells and the average of elements in each grid cell is output.

By stacking multiple convolutional, activation, and pooling layers, the body of a CNN is implemented. This section of a CNN acts as a feature extractor, and many deep models use them as their first stage, called backbone of an architecture. For instance, the backbone takes an image as input, processes it through its layers, and

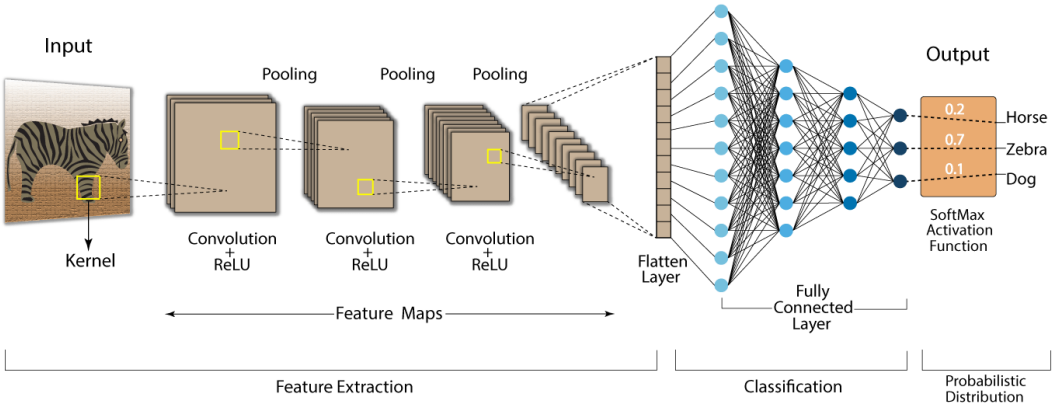


Figure 2.4: A basic CNN architecture for image classification.

outputs a feature map. This feature can be transformed into a feature vector which can be used as an embedding of the whole image in different problems. For classification or regression purposes, the extracted feature vector is fully connected to a loss layer with at least one hidden layer in between. To calculate the loss that should be back-propagated during the training process, and find the classification probabilities or regress unknown variables, the loss layer uses non-linear loss functions such as softmax for predicting a single class of C mutually exclusive classes, sigmoid cross-entropy predicting C independent probability values in [0,1], or Euclidean loss for regressing to real-valued variable in $(-\infty, \infty)$.

2.3 Backbone Architectures

A backbone network of a deep model serves as its feature extractor. As feature extractors of deep models, backbones a play huge role in the performance of deep models, and they should be designed or chosen delicately. Even though backbones can be architecture specific, most of them are actually image classification networks excluding the last fully connected layers. The frequently used backbones are VGG-16 [62], ResNet [26], ResNeXt [41], Inception V2 [31], Inception V3 [65], and Hourglass [37]. For mobile platforms, a lightweight backbone is a must. The commonly used lightweight backbones are MobileNet [28], ShuffleNet [79], and SqueezeNet [30].

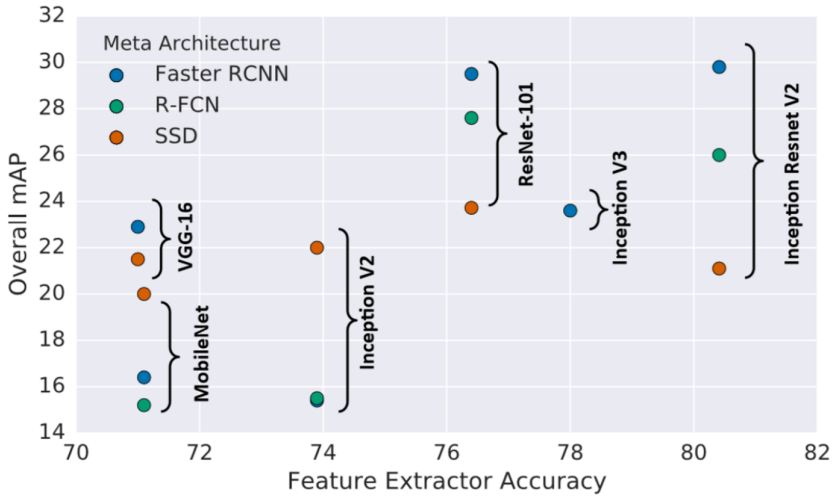


Figure 2.5: A comparison of detection accuracy on MS-COCO dataset between different backbones used in Faster RCNN, R-FCN, and SSD [29].

2.4 Feature Fusion Paradigms

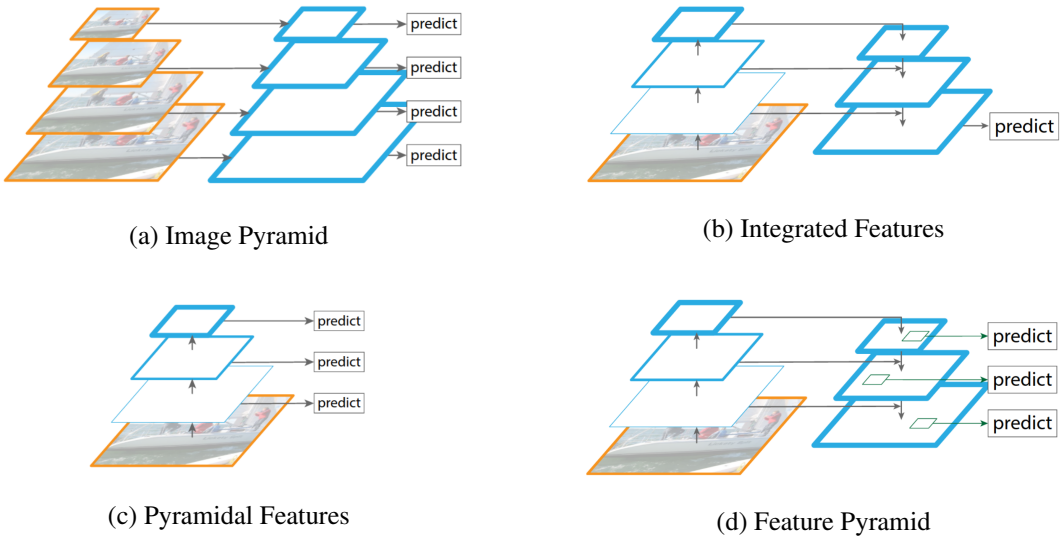
Invariance and equivariance are important properties of image feature representations [81]. The target of object classification is to learn how to interpret high-level semantic information, which requires invariant feature representations. On the other hand, object localization aims at discriminating position and scale changes, which requires equivariant representations. Since object detection consists of both object localization and classification tasks, object detectors are expected to learn both properties. Deep CNN models have multiple convolutional and pooling layers which yields to strong invariance but poor equivariance in deeper layers. Although classification of objects can be realized better with deeper models, localization gets worse. Shallower layers are preferable for localization as they preserve better spatial information such as edges and contours.

To be more precise, detection of objects in different scales and aspect ratios is an arduous work with a single feature map. Shallow layers have smaller receptive fields and higher resolution which is suitable for detecting smaller objects while deep layers have larger receptive fields and semantically richer features which are suitable for detecting larger objects and image classification. This trade-off encourages the fusion of deep and shallow features.

2.4.1 Same-scale Element-wise Operations

Feature fusion can be regarded as element-wise operation between two or more feature maps. Feature maps can be concatenated, summed or multiplied in an element-wise manner. Multiplication may give more importance to certain features which could improve small object detection while concatenation tends to fuse context information better.

2.4.2 Multi-scale Feature Learning



(a) Image Pyramid

(b) Integrated Features

(c) Pyramidal Features

(d) Feature Pyramid

Figure 2.6: Multi-scale feature learning techniques.

2.4.2.1 Image Pyramid

An image is resized to multiple scales and each scale is used for training one of multiple detectors. Detectors trained with larger scales aim at detecting smaller objects while those trained with smaller scales are used for detecting larger objects. During testing, the detection results from all scales are merged on the original scale. However, this approach is ineffective and computationally expensive as it requires training multiple detectors.

2.4.2.2 Integrated Features

Multi-scale feature maps extracted from different layers are fused into one feature map which is used for final prediction. For this, semantically rich but low resolution feature maps are up-sampled to the resolution of spatially rich high resolution feature maps, then any of element-wise operations can be applied on them. The fused feature representation is more descriptive and comprises necessary spatial and semantic information for both localization and classification.

2.4.2.3 Pyramidal Features

Feature maps extracted from different layers have different resolution and semantic information. Then, each of these multi-scale feature maps is fed into different detection heads so that each head is trained for detecting objects in a certain scale.

2.4.2.4 Feature Pyramid Network

A feature pyramid network incorporates integrated features into a prediction pyramid. A low resolution semantically rich feature map is up-sampled to higher resolution, and merged with a spatially richer feature map by summation or concatenation. Thereby, the spatially richer feature map is enhanced semantically which is required for detecting smaller objects. By this way, feature maps of certain shallow layers are all semantically enriched, and a set of multi-scale semantically enriched feature maps is obtained. Finally, each enriched feature map is fed into different detection heads for detecting objects in a certain scale.

CHAPTER 3

OBJECT DETECTION FOR RGB & RGB-D IMAGES

This chapter presents a literature review on object detection. First, it begins with the problem definition of object detection in Section 3.1. Then, object detection performance metrics and datasets are presented in 3.2 and Section 3.3, respectively. Following them, a brief review on the traditional approaches to object detection is given in Section 3.4. Finally, deep learning based object detection models are thoroughly explained in Section 3.5 for RGB images and 3.6 for RGB-D images.

3.1 Introduction

Precise estimation of the concepts and locations of objects contained in each image is essential for a complete image understanding. This task is commonly referred to as object detection [80]. As one of the fundamental problems of computer vision, object detection forms the basis of many other computer vision tasks, such as instance segmentation [11, 23, 24], image captioning [34, 72, 75], object tracking [33], etc.

Object detection involves both object classification and location regression tasks [73]. Therefore, the problem definition for object detection is how a model should be designed so that it finds out the precise positions and class labels of each object instances in an input image [59].

3.2 Performance Metrics

The most common metrics used to measure the performance of object detectors are average precision and mean average precision [50], Before going into details of them, how a predicted bounding box is regarded as a correct detection should be set forth.

Intersection of union (IoU) is a metric that shows how much a predicted bounding box overlaps with a ground truth bounding box. It is based on the Jaccard Index, a coefficient of similarity for two sets of data [32]. It is basically calculated by dividing the area of overlap between the predicted and ground truth bounding boxes by the area of their union.

$$J(B_p, B_g) = IoU = \frac{area(B_p \cap B_g)}{area(B_p \cup B_g} \tag{3.1}$$

where B_p and B_g stand for the predicted and the ground truth bounding boxes, respectively.

A predicted bounding is assigned to a ground truth bounding box if their IoU is above a threshold *t*, and their category labels are the same. A predicted bounding box cannot be assigned to multiple ground truths but a ground truth can be assigned to multiple predictions. Among all predictions assigned to a ground truth, the one with the highest confidence score is labeled as true positive (TP). This means that the detector successfully detects an existing object. The rest of the predictions are labeled as false positive (FP), which means that the detector claims to find a non-existing object. Finally, ground truth bounding boxes that has no assigned predictions are labeled as false negative (FN). This means that an existing object cannot be found by the detector.

A detector's precision (P) tells us what percentage of its predictions are true. It is calculated by dividing the number of correct detections by the number of total detections.

$$P = \frac{TP}{TP + FP} \tag{3.2}$$

A detector's recall (R) shows us what percentage of the existing objects are successfully detected. It is calculated by dividing the number of correct detections by the number of total ground truth objects.

$$R = \frac{TP}{TP + FN} \tag{3.3}$$

A detector's performance over a class of objects is measured by average precision (AP), which is basically the area under the class-based precision-recall curve. To calculate this area, 11-point interpolation is used. First, all detections of a class are sorted in decreasing order based on their confidence scores. Then, starting from the detection with the most confidence score, precision and recall values are calculated at each detection. However, the alternating sequence of TP and FP detections yields to a zigzag-like precision-recall curve which is not monotonically decreasing.

To remedy this problem, interpolated precisions at 11 recall values are used. Finally, the average of these interpolated precision values gives us the AP.

$$AP = \frac{1}{11} \sum_{r \in R} p_{interp}(r) \tag{3.4}$$

where, $R = \{0, 0.1, 0.2, ..., 1.0\}$, and

$$p_{interp}(r) = \max_{r^{'}:r^{'} \geqslant r} p(r^{'}) \tag{3.5}$$

where, p and $r^{'}$ are the calculated precision and recall values.

To measure the overall performance of an object detector over all object classes, mean average precision (mAP) is used, which is basically the average of APs over all classes. To give more emphasis on localization accuracy, MS-COCO introduced mAP@[.5:.95] which averages mAP over different IoU thresholds in [0.5:0.05:0.95]. MS-COCO benchmark uses different notations for the aforementioned metrics: AP50 refers to mAP calculated with IoU threshold at 0.5, AP75 refers to mAP calculated with IoU threshold at 0.75, and AP refers to mAP@[.5:.95].

There are also three AP values measured over different object sizes: APs, APm, and APl. APs is measured over the objects sizes of which are smaller than 32x32 pixels, whereas APl is measured over the objects sizes of which are larger than 96x96 pixels. Finally, APm is measured over the objects sizes of which are larger than 32x32 pixels but smaller than 96x96 pixels.

3.3 Datasets

Since large amount of data is the most important requirement of neural networks, several datasets and benchmarks have been developed over the past two decades such as PASCAL VOC2007 Challenge [15], PASCAL VOC2012 Challenge [16], ImageNet Large Scale Visual Recognition Challenge [58], MS-COCO Challenge [42], etc.

3.3.1 Pascal Visual Object Classes

Pascal Visual Object Classes 2007 (VOC07) is one of the first and most important benchmarks of object detection community. It has approximately 5k images and 12k objects of 20 classes that are common in everyday life such as person, bird, cat, bottle, sofa, etc. Later, in 2012, the dataset was increased to nearly 12k images and 27k objects of the same 20 classes, called as PASCAL VOC12. Today, VOC has become old-fashioned due to the introduction of larger datasets.

3.3.2 ILSVRC

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was held since 2010 until 2017, and included object detection challenge on ImageNet dataset. ILSVRC-14 and ILSVRC-17 have the same training datasets: 476k images and 534k objects of 200 classes. The only difference is that the latter has a larger test set.

3.3.3 MS-COCO

MS-COCO came in sight in 2015 and has become the standard dataset for object detection. It has more small objects, and bounding box annotations include segmentation masks which helps precise localization. MS-COCO has approximately 123k images and 900k objects of 80 classes.

Table 3.1: Commonly used datasets and their statistics.

Dataset	train		validation		trainval		test	
	images	objects	images	objects	images	objects	images	objects
VOC-2007	2,501	6,301	2,510	6,307	5,011	12,608	4,952	14,796
VOC-2012	5,717	13,609	5,823	13,841	11,540	27,450	10,991	-
ILSVRC-2014	456,567	478,807	20,121	55,502	476,688	534,309	40,152	-
ILSVRC-2017	456,567	478,807	20,121	55,502	476,688	534,309	65,500	-
MS-COCO-2015	82,783	604,907	40,504	291,875	123,287	896,782	81,434	-
MS-COCO-2015	118,287	860,001	5,000	36,781	123,287	896,782	40,670	-

3.4 Traditional Object Detectors

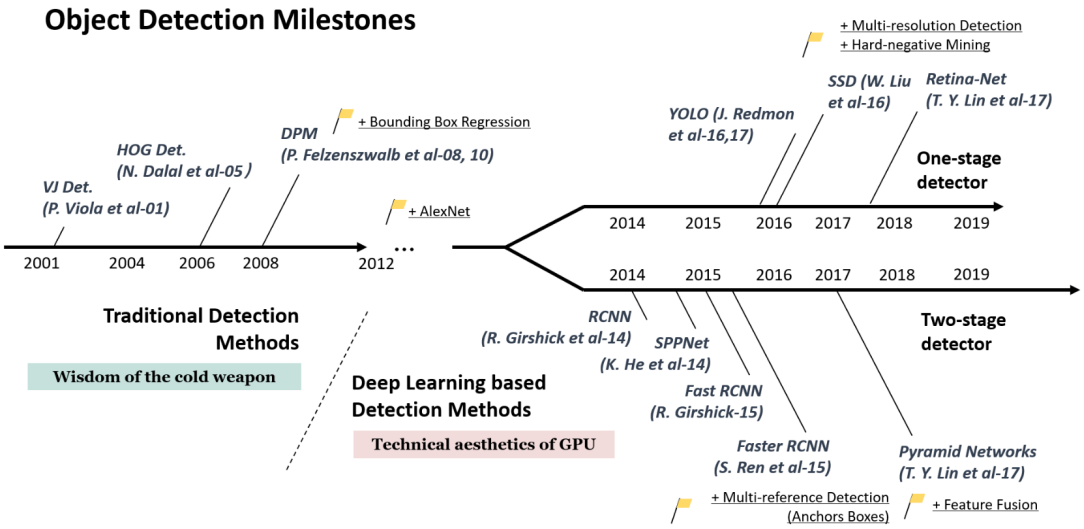


Figure 3.1: The timeline of object detection [81].

The history of object detection can be partitioned into two periods: traditional detection models and deep learning-based detection networks.

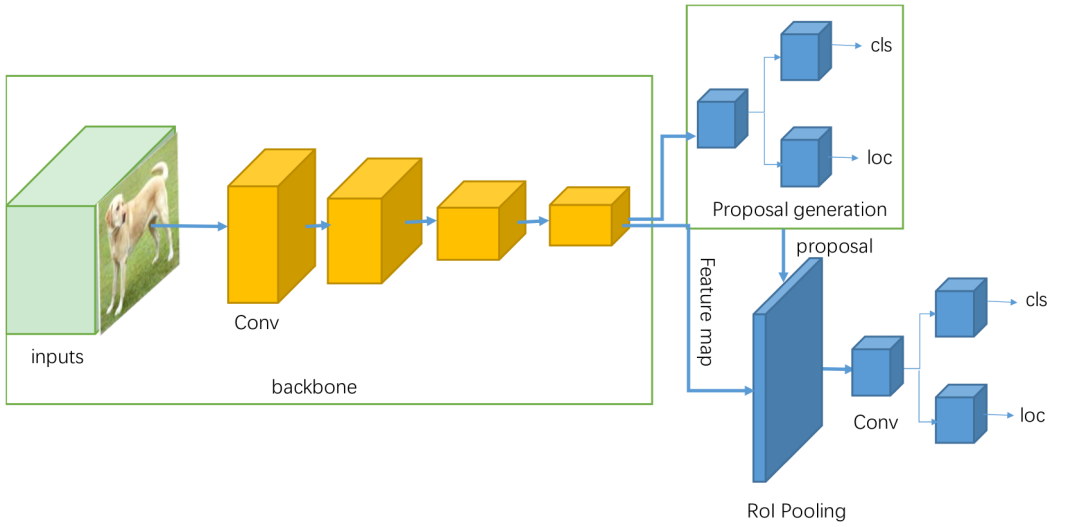
Traditional object detectors comprise of three tasks: generation of region proposals, feature vector extraction, and classification of region proposals. As a first task, possible regions which may contain objects should be detected. These proposed regions are commonly referred to as regions of interest (RoI). To capture these regions, sliding windows are used [67, 1, 12]. However, multiple sliding windows in different scales are required as objects can be found in different scales and aspect ratios. In the second task, feature vectors are extracted from region proposals in order to encapsulate the semantic information found in these region. For this task, low level feature descriptors are utilized, e.g., HOG (Histogram of Gradients) [12], SIFT (Scale Invariant Feature Transform) [40], and DPM (Deformable Part-based Model) [17]. In the last task, the region proposals are labeled to one of the classes based on their feature vectors. To do this, extracted feature vectors are used to train region classifiers such as SVM (Support Vector Machine) [27].

Traditional methods require meticulous design of feature descriptors to get satisfactory results. As one of the groundbreaking object detection algorithms, DPMs won PASCAL VOC challenges in three consecutive years from 2007 to 2009. However, no significant progress was achieved by traditional methods after 2008. There are a couple of prominent reasons behind their limitations [73]. First, low level feature descriptors were mainly used and manually designed. This results in obtaining deficient semantic information from region proposals. Secondly, each stage of object detectors were designed individually. As they were not end-to-end trainable, traditional object detectors would fail in finding the global optimum solution.

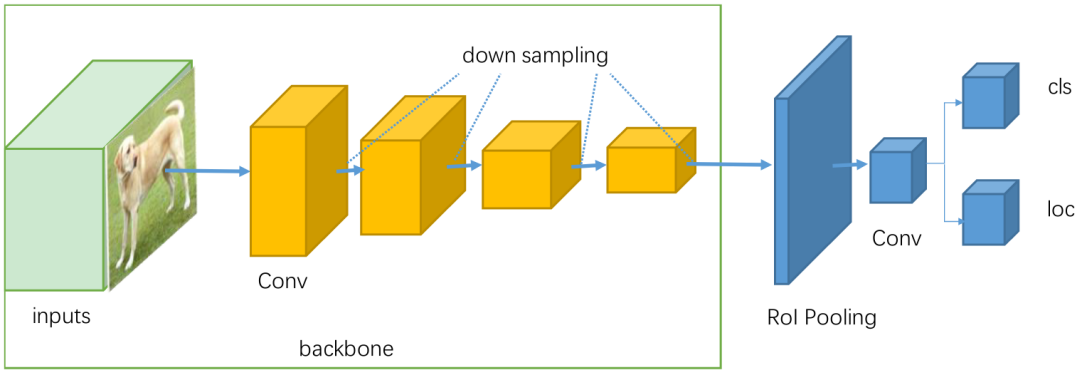
As traditional object detection models suffered from certain limitations, no significant improvement was achieved from 2008 until 2012 when convolutional neural networks appeared.

3.5 Deep Learning-Based Object Detection in RGB

Object detectors can primarily be grouped into two types of models based on the assembly of their architectures: two-stage detectors (Fig. 3.2a) and one-stage detectors (Fig. 3.2b).



(a)



(b)

Figure 3.2: Comparison of deep learning-based object detector models: (a) basic architecture of two-stage object detectors, (b) basic architecture of one-stage object detectors.

3.5.1 Two-Stage Detectors

3.5.1.1 RCNN

RCNN (Region-based Convolutional Neural Network) [36] comprises of three consecutive tasks: generation of region proposals, feature extraction, and classification. Similar to the traditional object detectors, the first stage of RCNN uses selective search for generation of region proposals. In the second stage, proposed regions are first cropped from the input image, resized, and then fed into the backbone to extract the feature vectors of crops. Finally, based on these feature vectors, SVMs classify the objects assumed to be in the image crops, and the coordinates of the region proposals are adjusted by regressors.

Contrary to the traditional methods which use low level feature descriptors, RCNN uses a CNN model (AlexNet [36] trained on ImageNet [13]) for feature extraction from region proposals. Since CNNs have more discriminative expression capability than hand-crafted descriptors, RCNN surpassed all traditional models on VOC-2007, with a great improvement of mAP from 33.7% (DPM-v5 [17]) to 58.5%. However, RCNN cannot be used for real-time applications. RCNN is too slow (47s per test image) as it generates 2000 region proposals per image most of which are redundant, and each of these 2000 region proposals is processed through AlexNet which requires a lot of time.

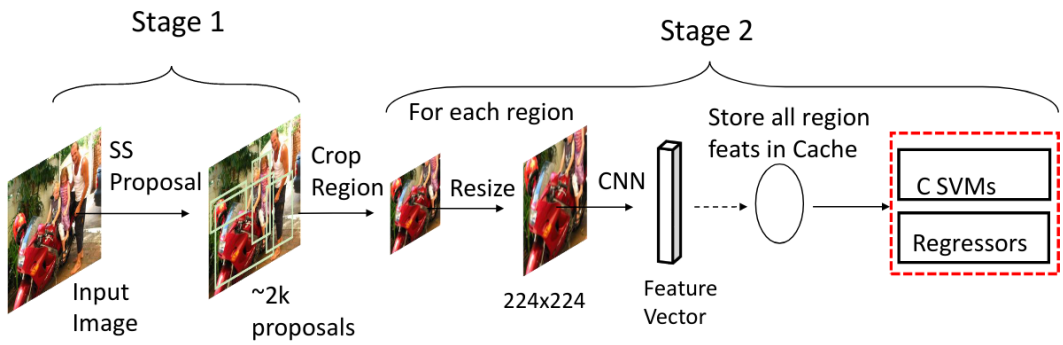


Figure 3.3: Overview of RCNN architecture [73].

3.5.1.2 Fast RCNN

The order of RoI generation and feature extraction tasks of RCNN is swapped in Fast RCNN [19]. As a first task, an image is given as input to the CNN, and at the output, the feature map of the whole image is acquired. Then, region proposals are generated from this feature map by using selective search, which is called as RoI pooling layer. As a result, CNN is used only once instead of 2000 times per image. Consequently, Fast RCNN becomes approximately 20 times faster than RCNN (Table. 3.2). Extracted regions of the convolutional feature map are then fed into a sequence of fully connected layers to obtain the fixed-length feature vector. Finally, this feature vector is fed into two parallel branches: a softmax layer to predict the class label including the background, and a regression layer that outputs four real-valued number for each of classes. These four numbers are used to refine the boundaries of the region proposals. With these modifications, Fast RCNN achieved mAP of 70.0% on VOC-2007, increased from 58.5% of RCNN.

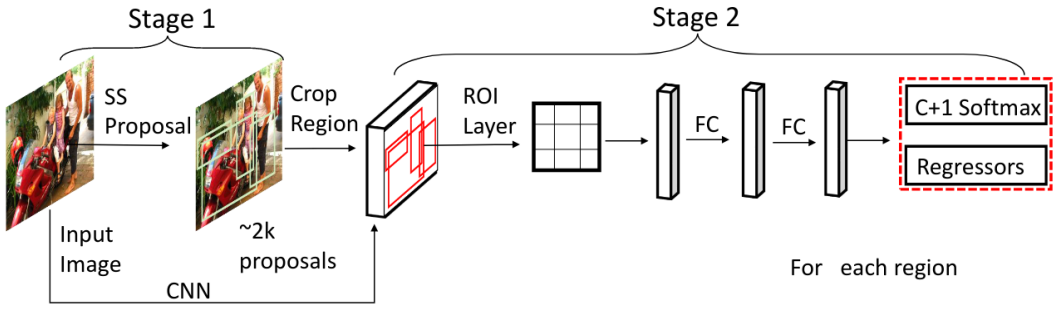


Figure 3.4: Overview of Fast RCNN architecture [73].

Table 3.2: Test speed and performance analysis of RCNN family in terms of mean average precision (mAP) using PASCAL VOC-2012 test dataset.

Model	mAP	Test Speed (sec.)
RCNN	62.4	49
Fast RCNN	68.4	2.3
Faster RCNN	70.4	0.2

3.5.1.3 Faster RCNN

Both RCNN and Fast RCNN use selective search which is non-trainable and time-consuming. Therefore, in Faster RCNN [55], selective search is replaced with a convolutional network, called as region proposal network, and it serves as the attention mechanism of whole architecture. As proposals, k different regions with predefined scale and aspect ratios, called *anchors*, are considered. An n x n window slides over the feature map, and at each sliding-window location, the n x n portion of the feature map is mapped to a 256-d feature vector, which is then fed into two parallel branches: the *cls* layer outputs 2 scores for each of k anchors that indicate the probability of them comprising an object or not, and *reg* layer outputs 4 normalized coordinates for each of k anchors. Consequently, Faster RCNN becomes the first end-to-end trainable deep learning based object detector.

Fast RCNN achieved mAP of 73.2% on VOC07, increased from 70.0% of Fast RCNN. However, the real improvement of Faster RCNN is its speed: Faster RCNN is 10 timer faster than Fast RCNN, and 200 times faster than RCNN (Table 3.2), which makes Faster RCNN the first near-realtime deep learning based object detector as well.

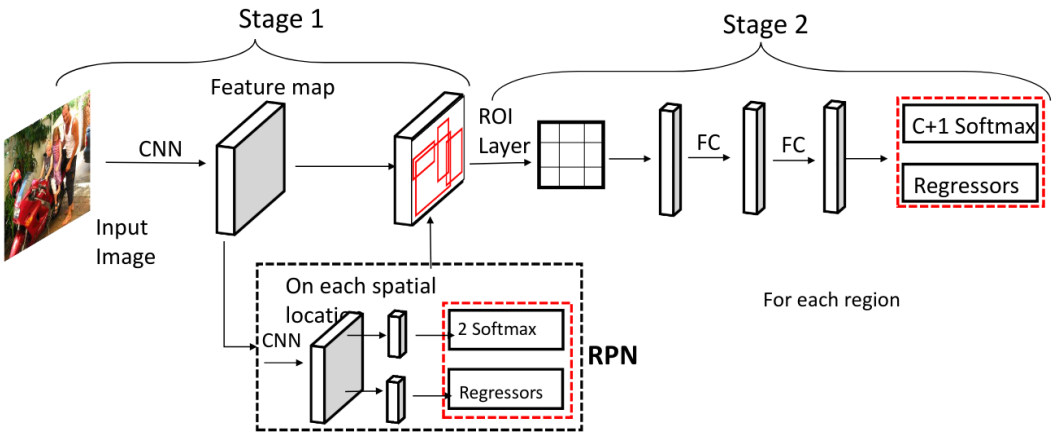


Figure 3.5: Overview of Faster RCNN architecture [73].

3.5.2 One-Stage Detectors

3.5.2.1 YOLO

YOLO (You Look Only Once) [52] was the first real-time one-stage deep learning-based object detector. The idea behind YOLO is that it removes the task of region proposal generation and models object detection as a regression problem.

YOLO divides an image into an *S x S* grid and regards each cell as a region of interest. For each grid cell, YOLO considers *B* bounding boxes, and predicts *x,y,w,h* and a confidence score for each bounding box. (x, y) represents the center of the bounding box relative to the grid center, and (w, h) represents the width and height relative to the image size. Moreover, for each grid cell, *C* class probabilities are predicted. As for the implementation details of YOLO, an input image is first processed through a CNN model, called DarkNet, and extracts a feature vector. Then, the feature vector is fully connected to a feature map of size *S x S x* $(5{*}B + C)$.

The single-stage architecture of YOLO and the lack of region proposal generation makes it faster than RCNN family. YOLO works at 45 fps and achieves an mAP of 63.4% on VOC2007, and 57.9% on VOC2012. However, YOLO has its own drawbacks too. There is a fixed number of bounding boxed per grid cell, and they fail to capture small or crowded objects. Furthermore, only one feature map is utilized, which leads to insufficient generalization and deficient learning of objects at multiple scales or aspect ratios.

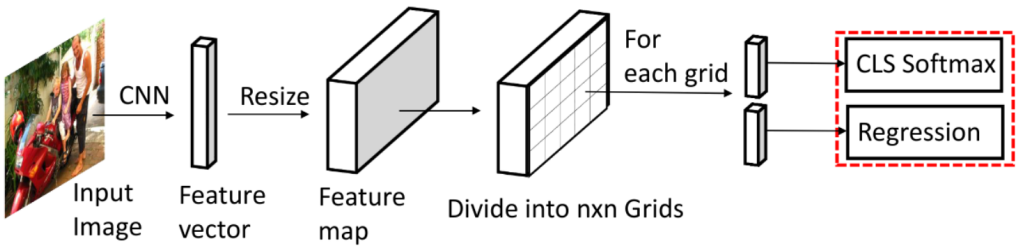


Figure 3.6: Overview of YOLO architecture [52].

3.5.2.2 YOLOv2

YOLO tries to detect the coordinates of the bounding boxes by using fully connected layers on the feature vector extracted from the DarkNet backbone. YOLOv2 [53] removes these fully connected layers and uses hand-picked priors in multiple scale and aspect ratios, called anchor boxes, to predict the true bounding box. Predicting offsets rather than coordinates simplifies the task and allows the network to learn faster. Moreover, using anchor boxes separates the localization and classification tasks which YOLO tries to realize on the same feature map. Meanwhile, YOLOv2 predicts class probabilities and confidence score for each anchor box.

In addition to using anchor boxes, YOLOv2 increases the depth of its backbone to 19, called as DarkNet-19, uses more bounding boxes, and applies batch normalization.

3.5.2.3 YOLOv3

YOLO suffers from ineffective detection of objects at multiple scales or aspect ratios, especially the small ones. This issue keeps YOLO's performance behind Faster RCNN's, and makes it less preferable despite its speed. To remedy this problem, YOLOv3 [54] uses a similar concept to feature pyramid networks and predicts bounding boxes at 3 different scales. To do this, outputs of $79^{th}, 91^{th}$, and 103^{th} layers are used. Each layer is fed into 3 parallel branches, each of which has 3 convolutional layers, and a different stride, i.e., the first branch has a stride of 32, the second one has 16, and the last one has 8. Finally, the output of these branches are passed through kernels of size *1 x 1 x* (*B x* (*5 + C*)) as each grid cell has *B* bounding boxes each of which has *C* class probabilities, 1 objectness score, and 4 coordinate offsets.

In addition to detection at 3 different scales, YOLOv3 also increases the depth of its backbone to 53, called DarkNet-53, to extract semantically richer features. More importantly, YOLOv3 upsamples lower resolution but semantically richer feature maps of later stages, and concatenates them with semantically poor but higher resolution feature maps of earlier stages. This helps YOLOv3 obtain semantically richer and higher resolution feature maps which is needed for detection of smaller objects.a

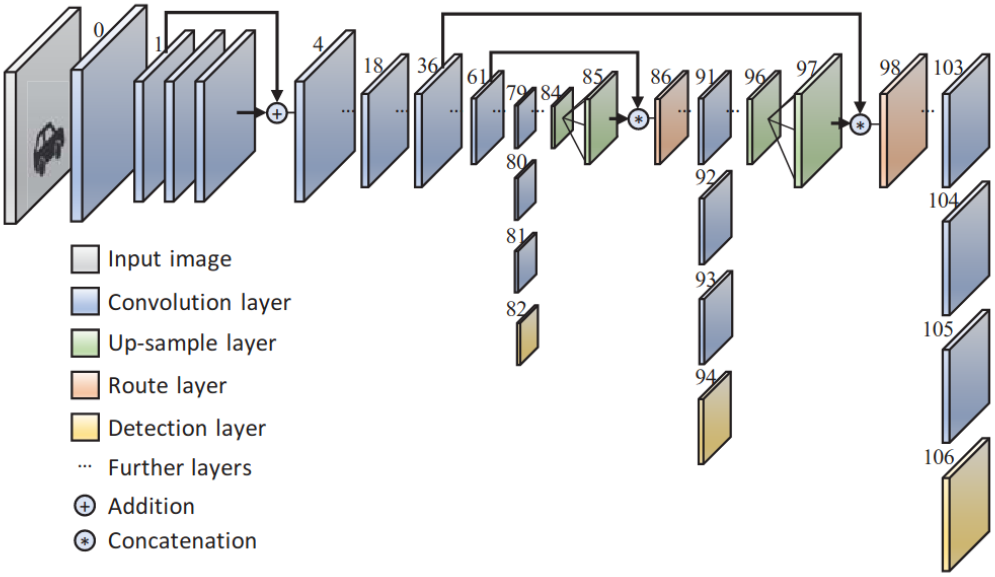


Figure 3.7: Overview of YOLOv3 architecture [8].

Table 3.3: Performance analysis of some state-of-the-art models on different datasets.

Model	VOC-2007 (mAP)	VOC-2012 (mAP)	MS-COCO (AP)	MS-COCO (AP_{50})
RCNN (Alex)	58.5	53.3	-	-
RCNN (VGG)	66.0	62.4	-	-
Fast RCNN	78.9	80.1	19.7	35.9
Faster RCNN	69.9	70.4	21.9	42.7
YOLO	63.4	57.9	-	-
YOLOv2	78.6	73.4	21.6	44.0
YOLOv3	-	-	33.0	57.9

3.6 Deep Learning-Based Object Detection in RGB-D

Traditionally, object detection algorithms would frequently work with RGB images. With low-cost 3D scanners being more available in recent years, a variety of RGB-D object detection datasets and algorithms have been proposed. Similar to the different problems of computer vision, unparalleled levels of performances have also been achieved in RGB-D object detection with the new deep learning techniques and availability of large datasets.

RGB-D object detection models are very similar to RGB based object detectors. They utilize the same pipeline architectures e.g. one-stage or two-stage detectors. However, RGB-D object detection algorithms aim to use different modalities found in the RGB-D images. Since they involve complementary information, e.g. color and depth, RGB-D object detection models require an efficient fusion strategy. These strategies can be grouped into three types: early fusion, late fusion, and deep fusion [71].

3.6.1 Fusion Techniques

3.6.1.1 Early Fusion

The RGB image and the depth map are concatenated to form a 4-channel image, and then this concatenated image is fed as the input to the object detector. Since fusion process is performed outside the object detector, this technique is referred to as early fusion.

3.6.1.2 Late Fusion

The RGB image and the depth map are processed individually in different networks. Just before the final stage, which deals with classification and bounding box regression, the features output by these parallel networks are fused by either concatenation, and this fused feature map is used for the final stage. Since the fusion takes place last, this technique is referred to as late fusion.

In [14], two CNN networks are used for each modality. Both networks are pre-trained on ImageNet that take 3-channel inputs. To be able to use depth maps as input, they are first normalized to lie in the range of [0,255], and then applied by a jet colormap to increase the channel depth from one to three. Both RGB and colored depth map are processed by their own network that has 5 convolutional layers and 2 fc layers. Then the output of the last fc layers of both networks are concatenated, and then fed to the last fc and classification layers.

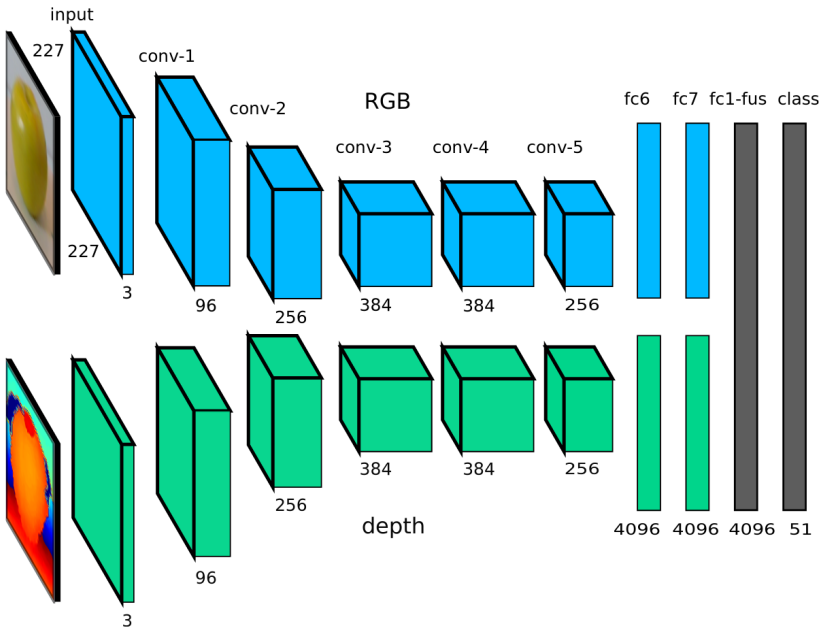
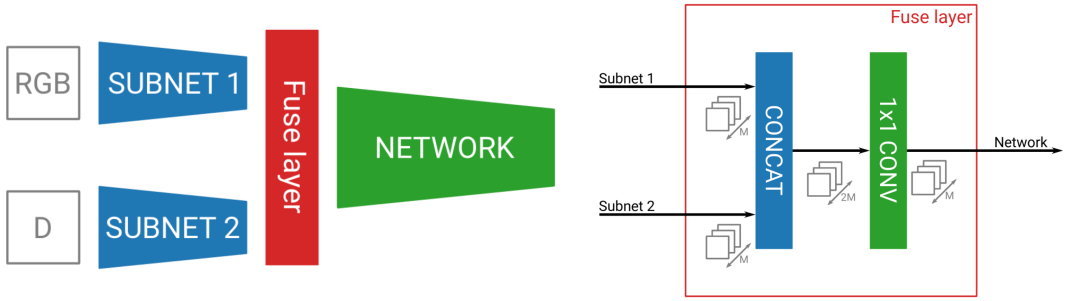


Figure 3.8: Overview of late fusion architecture in [14]. The upper branch takes RGB image as input while the lower branch takes colored depth map. Both branches have seven layers, and the output of 7^{th} layers are fused by a fc layer for classification.

3.6.1.3 Deep Fusion

In deep fusion technique, the color and depth information are independently processed through their own CNN branches. Then, the output of these branches are fused by concatenation and fed to the third CNN branch which prepares the final feature map for classification and bounding box regression.

Ophoff et al. modify YOLOv2 by utilizing deep fusion technique [48]. The first 14 layers of YOLOv2, which has 27 layers in total, are duplicated and used as the two parallel CNN branches Each branch outputs either a depth-based feature map or a color-based one. Then, these output feature maps are concatenated and followed by 1x1 convolution for dimension reduction. The fused feature map is then fed to the shared last 13 layers of YOLOv2.



(a) Modified YOLOv2 architecture.

(b) Fusion layer.

Figure 3.9: Overview of modified YOLOv2 with deep fusion for RGB-D data [48].

3.6.2 Datasets

Compared to the RGB datasets, there are fewer RGB-D datasets as RGB-D images are harder to obtain. One of the most commonly used RGB-D datasets is SUN RGBD [63] and can be regarded as the standard dataset. It contains 10335 real RGB-D images of room scenes. The training and testing datasets contain 5285 and 5050 images, respectively. Ground truth annotations include 2D bounding boxes, 3D bounding boxes, instance and semantic segmentation masks, object orientation, and room layout estimation. More than 800 categories are labeled; however, a base set of 19 classes is commonly used for object detection. Similarly, a base set of 37 classes is often used for instance segmentation.

CHAPTER 4

INSTANCE SEGMENTATION

This chapter presents a literature review on instance segmentation. First, it begins with a brief introduction and problem definition of instance segmentation in Section 4.1. Then, object detection performance metrics and datasets are presented in 4.2 and Section 4.3, respectively. Following them, a brief review on the traditional approaches to instance segmentation is given in Section 4.4. Finally, deep learning based instance segmentation models are thoroughly explained in Section 4.5.

4.1 Introduction

Similar to the generic object detection, instance segmentation deals with classification and localization of objects in an input image. However, contrary to the generic object detection, instance segmentation performs pixel-level localization instead of enclosing the object in a rectangular bounding-box. For this, each pixel is assigned to a unique object of a certain class. Thereby, instance segmentation can be regarded as solving object detection and semantic segmentation problems simultaneously. Since instance segmentation provides the most detailed information about objects, it is highly demanded by different fields, e.g., intelligent driving and medical health [66].

4.2 Performance Metrics

Performance metrics used to assess an instance segmentation model is nearly the same as the metrics used for object detection models (Section 3.2). The only difference is in the calculation of IoU. In object detection, IoU is defined between the detected and

ground-truth bounding boxes. Meanwhile, in instance segmentation, IoU is defined between the detected and ground-truth pixel masks.

4.3 Datasets

As is well known, datasets play a huge role in deep-learning based computer vision tasks. As a result, a quite number of instance segmentation datasets and benchmarks have been developed through years. Along with their object detection datasets (Section 3.3), Pascal VOC and MS-COCO provide datasets for instance segmentation as well. Pascal VOC was the most prominent one until MS-COCO came out. MS-COCO's object detection dataset has pixel-level annotations that can also be used for instance segmentation, which makes it the largest and most commonly used dataset for instance segmentation. Apart from these two datasets, Cityscapes Dataset [9] and the Mapillary Vistas Dataset (MVD) [46] can be mentioned as other frequently used datasets for instance segmentation.

Table 4.1: Commonly used datasets and their statistics.

Dataset	Classes	Train-Val Images	Test Images
VOC-2012	20	11,540	10,991
Cityscapes	8	3,475	1,525
MVD	66	25,000	-
MS-COCO-2015	80	123,287	40,670

4.4 Traditional Instance Segmentation Models

The history of instance segmentation dates back to the traditional image segmentation models. These models partition images into mutually exclusive yet meaningful regions, and pixels in the same region have some certain correlations.

Traditional image segmentation models can be grouped into three types of methods: thresholds [49, 78, 77], edges [10], and clustering [70, 60]. The threshold-based

image segmentation classifies pixels based on different gray-scale thresholds. Pixels that fall into the same gray-scale range are grouped into the same region. This method is convenient for images in which there are apparent gray-scale differences between objects and the background. The edge-based image segmentation detects boundary pixels of an object, and then connects them to form the object's contour. Commonly used edge-based models include Roberts operator [56], Prewitt algorithm [76], Sobel [18], and Canny [6]. Finally, the cluster-based image segmentation tries to merge neighboring pixels with similar features into the same partition. Commonly used models include K-means [3], FCM [47], and SLIC [2].

4.5 Deep Learning-Based Instance Segmentation Models

Instance segmentation technology has made huge progress with the improvements in hardware efficiencies and deep learning techniques. Especially, deep learning-based object detection models have made profound contributions to the advancement in the instance segmentation field.

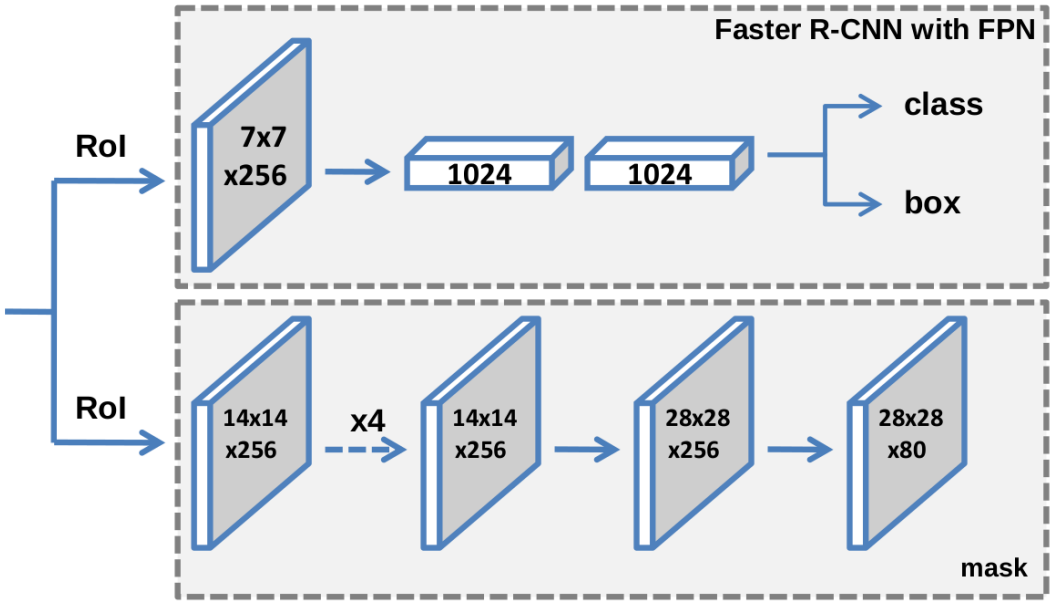


Figure 4.1: Overview of Mask RCNN architecture [25].

4.5.1 Mask RCNN

Mask RCNN [25] is basically an extension to Faster RCNN. The model incorporates a mask branch to Faster RCNN so as to predict segmentation mask of instances. In Faster RCNN, region proposals are only used for classification and regression of bounding box offsets. Mask RCNN feeds these region of interests (RoI) to its segmentation head as well. Segmentation head first performs RoI pooling, i.e., divides RoI into $14x14$ feature map by bilinear interpolation. Bilinear interpolation is preferred because quantization yields to loss of necessary data or addition of uninterested data on the RoI boundaries. Then this feature map is fed through multiple convolution layers and upsampled to size of $28x28$. Finally, $1x1$ convolutional kernel per class is applied to get segmentation mask per class. As MS-COCO has 80 classes, the output of segmentation head has 80 channels in Figure 4.1. The correct segmentation mask is decided based on the classification output of Faster RCNN. According to the classification result, the corresponding class channel in segmentation output is chosen. With all these contributions, Mask RCNN achieved first position in 2016 MS-COCO challenges.

4.5.2 PANet

Path aggregation network (PANet) [43] takes Mask RCNN further by introducing bottom-up path augmentation and fully connected fusion. The overall architecture of PANet is similar to Mask RCNN. It uses ResNet with FPN as its baseline. The multi-scale feature maps extracted from different FPN layers are processed through its bottom-up augmented path. Then, as in Mask RCNN, RoI pooling is performed on these processed feature maps, and extracted RoI features are fed into both detection and segmentation branches. In both branches, features are fused halfway to leverage semantic and spatial information as much as possible. As a result of these contributions, PANet achieved the first position in 2017 MS-COCO challenge.

As the novel contribution of PANet, bottom-up path augmentation is used right after the backbone ResNet-FPN. The backbone generates feature levels $\{P_2, P_3, P_4, P_5\}$.

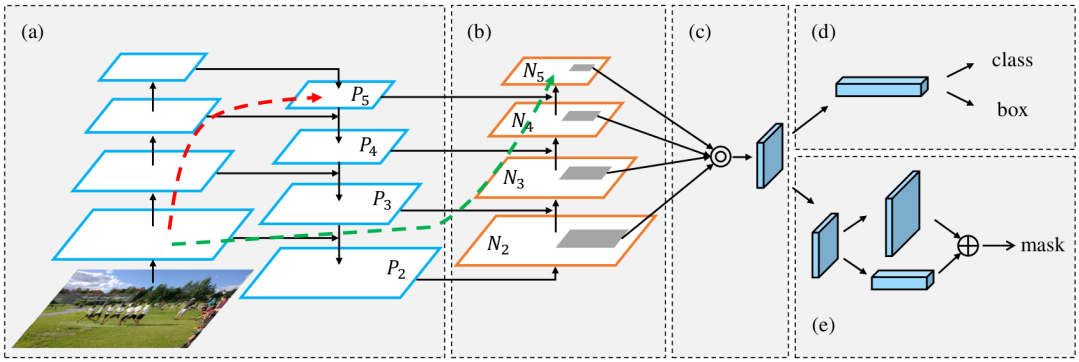
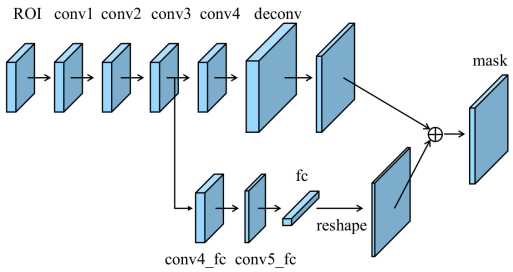
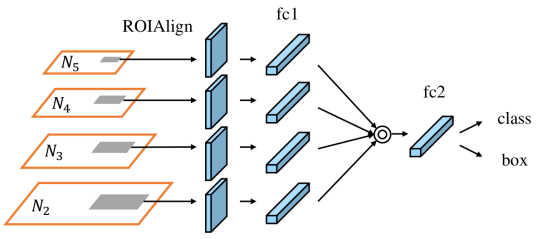


Figure 4.2: Overview of PANet architecture [43]. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Bounding box branch. (e) Mask branch.

The proposed augmented path begins with the lowest level P_2, and each feature level goes through a $3x3$ convolution with stride 2 to match its resolution with the higher feature level so that they can be summed and processed by another $3x3$ convolutional layer which outputs the fused feature map of that layer. By repeating this bottom-up augmentation strategy, fused feature levels $\{N_2, N_3, N_4, N_5\}$ are obtained. The reason to incorporate this network is that it enhances the localization ability of FPN hierarchy by propagating low-level features from lower layers to higher layers. Without bottom-up augmentation, FPN only propagates high-level semantically rich features from higher layers to lower layers.



(a) Mask prediction branch.



(b) Adaptive feature pooling.

Figure 4.3: Details of mask prediction branch and adaptive feature pooling of PANet.

As in Mask RCNN, RoI pooling is performed on each of fused feature maps, and they are fed into both detection and segmentation branches. In detection branch, each extracted feature map is first processed through a *fc* layer, and then summed. This enables the network to adapt features. The fused feature is then processed through one *fc* layer for bounding box regression and one *fc* layer for obtaining class probabilities. In segmentation branch, each extracted feature map is first processed through an individual convolutional layer, and then the outputs of these layers are summed. The fused feature map is processed through two more convolutional layers and then fed into two parallel sub-networks. The first sub-network applies one convolutional and one deconvolutional layer, and predicts a binary pixel-level mask for each class independently, which decouples the segmentation and classification tasks as in Mask RCNN. The second sub-network applies two *fc* layers, and predicts class-agnostic foreground/background mask. The outputs of both sub-networks are summed to obtain the final segmentation mask.

4.5.3 SOLO

As is known, two objects in an image have either different centroids or different scales. Using this prior information, SOLO [68] introduces the notion of instance categories and aims for directly predicting segmentation masks instead of detect-then-segment procedure. As a result, SOLO can be trained in an end-to-end fashion using only mask annotations. Unlike previous instance segmentation models which perform classification, bounding-box-level detection, and segmentation, SOLO performs only classification and segmentation tasks. With its simple yet effective model, SOLO reaches the same performance of Mask RCNN.

SOLO basically divides an input image into a grid of *SxS* cells. The grid cell where the center of an object is located is responsible for classification and segmentation of that object. Hence, each grid cell handles only one instance. To distinguish objects with different sizes, SOLO uses FPN backbone and utilizes multi-scale feature maps.

SOLO first processes the input image through its FPN backbone, and obtains multi-scale feature maps. These feature maps are then fed into the prediction heads. There is one prediction head responsible for each scale of feature map, and each prediction

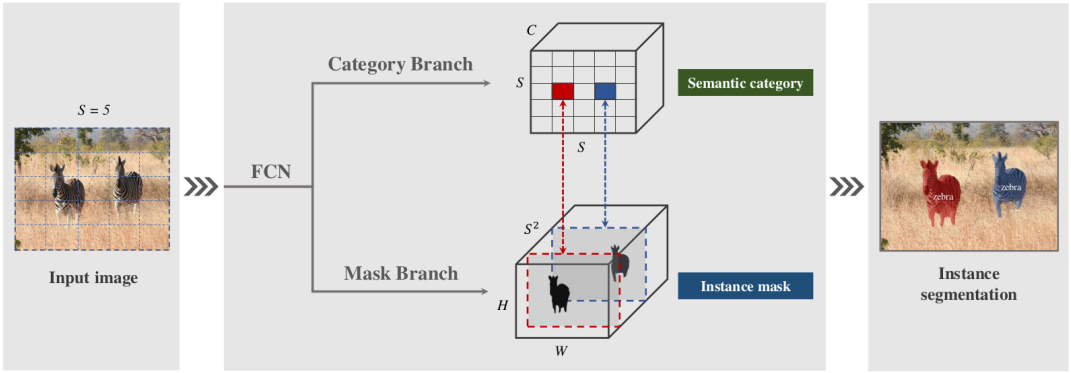


Figure 4.4: Overview of SOLO architecture [68].

head has two parallel branches, one for classification and one for segmentation. The classification branch predicts *C* class probabilities for each grid cell, i.e. the output tensor is of size *SxSxC*. By this way, all class probabilities are conditioned on grid cells. The segmentation branch predicts segmentation mask for each grid cell, i.e. the output tensor is of size $WxHxS^2$, where W and H stand for the original image width and height, respectively. Consequently, a one-to-one correspondence between classification and class-agnostic segmentation is achieved. Finally, all grid results are compiled and NMS is used to filter duplicated detections.

Before feeding to the prediction heads, feature maps extracted from multiple FPN layers are actually concatenated with normalized coordinates of corresponding pixels in order to incorporate the spatial functionality to the network. The reason behind the explicitly incorporating spatial information is that convolutional kernels are spatially invariant to some degree whereas SOLO needs to be spatially variant because segmentation masks are conditioned on the grid cells. If the original feature map is of size *HxWxD*, the concatenated feature map is of size *HxWx(D+2)*, where the last two channels are *(x,y)* coordinates of the corresponding pixel, normalized to [-1,1].

4.5.4 SOLOv2

SOLOv2 [69] takes SOLO further by introducing a couple of modifications to its segmentation branch. The first modification is that SOLOv2 fuses multi-scale feature outputs of FPN backbone into a unified feature map. The feature map extracted from

the deepest FPN level, P5, is concatenated with normalized pixel coordinates. By applying a series of convolutions, up-samplings, and element-wise summation with the rest of output feature maps of FPN, a single feature representation is obtained, and fed into the prediction head.

SOLOv2 modifies the prediction head by using dynamic convolution in its segmentation branch. SOLO outputs a segmentation tensor of size $HxWxS^2$. This output tensor is very large and computationally inefficient. SOLO prepares segmentation mask for all grid cells as if there were at least one object in each of these cells. Therefore, SOLOv2 aims for performing segmentation for only valid cells, which requires dynamic convolution. For this, segmentation branch is decoupled into two sub-branches: kernel branch and feature branch. Feature branch performs convolution on the input feature map extracted from FPN, and obtains the final feature map of size *WxHxE*, where E stands for the channel depth, W and H stand for the original image width and height, respectively. Kernel branch performs convolution on the very same feature map of FPN, and outputs segmentation kernels of each grid cell. The output of kernel branch is of size *SxSxD*, where D is the channel depth and corresponds to the segmentation kernel size, i.e., if the segmentation kernel size of a grid cell is *kxk*, then $D = k^2E$.

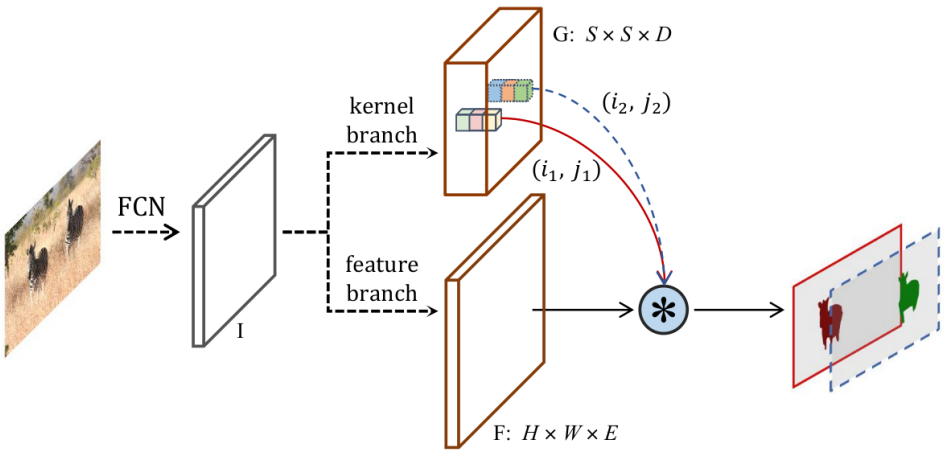


Figure 4.5: Overview of SOLOv2 architecture [69].

In short, the valid grid cells are decided based on the class probabilities computed by the classification branch. Then, segmentation kernels of the valid cells are extracted from the kernel branch. By performing convolution using these segmentation kernels on the feature map output by the feature branch, segmentation mask of each valid cell is obtained.

Model	AP (%)	Year
Mask RCNN	37.1	2017
PANet	40.0	2018
SOLO	37.8	2020
SOLOv2	39.7	2020

Table 4.2: Performance analysis of some state-of-the-art instance segmentation models on MS-COCO test dataset.

CHAPTER 5

DEEP LEARNING-BASED SINGLE IMAGE DEPTH ESTIMATION

This chapter presents a literature review on single image depth estimation. First, it begins with a brief introduction and problem definition of monocular depth estimation in Section 5.1. Then, object detection performance metrics are explained in 5.2. Finally, deep learning-based single image depth estimation models are given in Section 5.3.

5.1 Introduction

As one of the most fundamental problems in scene understanding, depth estimation is the task of predicting the spatial structure of a scene based on the appearance of objects in images. There are many applications such as synthetic object insertion in computer graphics [35], synthetic depth of field in computational photography [4], grasping in robotics [39], using depth as a cue in human body pose estimation [61], robot assisted surgery [64], and automatic 2D to 3D conversion in film [74].

Successful depth estimation techniques have been developed using structure from motion, shape-from-X, binocular and multi-view stereo; however, most of these methods require multiple observations of the interested scene, e.g., multiple viewpoints or multiple observations under different lighting conditions [21]. To tackle this problem, many researchers have focused on monocular depth estimation which aims for predicting dense map for a given single RGB image. What makes monocular depth estimation challenging is that there is an infinite number of 3D scenes that yield to the same 2D image, i.e. there is a one-to-many mapping between RGB images and depth maps [44]. Human visual system utilizes several monocular cues to overcome

this ambiguity such as occlusion, perspective, size, texture gradient, motion parallax etc. This phenomenon that humans use prior knowledge and statistics for monocular depth estimation has encouraged many researchers to exploit machine learning and deep learning techniques to advance depth estimation models.

5.2 Performance Metrics

Given a predicted depth image and the corresponding ground truth, with $\hat{d}_p$ and d_p denoting the estimated and ground-truth depths respectively at pixel p, and T being the total number of pixels for which there exist both valid ground truth and predicted depth [5],

- Absolute Relative Error

$$\frac{1}{T}\sum_{p}\frac{|d_p - \hat{d}_p|}{d_p} \tag{5.1}$$

- Linear Root Mean Square Error (RMSE)

$$\sqrt{\frac{1}{T}\sum_{p}(d_p - \hat{d}_p)^2} \tag{5.2}$$

- Log-Scale Invariant RMSE

$$\frac{1}{T}\sum_{p}(\log \hat{d}_p - \log d_p + \alpha(\hat{d}_p, d_p))^2 \tag{5.3}$$

  where $\alpha(\hat{d}_p, d_p)$ addresses scale alignment.

- Accuracy Under A Threshold

$$\max(\frac{\hat{d}_p}{d_p}, \frac{d_p}{\hat{d}_p}) = \delta < th \tag{5.4}$$

  where th is a predefined threshold.

5.3 Deep Learning-Based Single Image Depth Estimation Models

5.3.1 Monodepth

Prior to Monodepth [21], most of the learning-based approaches considered monocular depth estimation as a supervised regression problem. This requires a large quantity of labeled data; however, obtaining quality depth data from a variety of environments is burdensome. To remedy this problem, Monodepth removes the necessity of ground truth depth data by replacing it with binocular stereo images, and poses depth estimation as an image reconstruction problem. To be more precise, Monodepth has access to I^l and I^r during training, the left and right RGB images captured simultaneously from a calibrated stereo pair. Instead of directly estimating the depth map, Monodepth tries to find the dense correspondence field d^r that can be utilized to reconstruct the right image from the left pair, i.e. $\tilde{I^r} = I^l(d^r)$, where $\tilde{I^r}$ corresponds to the reconstructed right image. If the input images are rectified, d becomes the image disparity that Monodepth learns to estimate. Given the baseline b and the camera focal length f, the depth can be estimated from the estimated disparity, $\hat{d} = bf/d$.

Monodepth network is basically built upon an encoder-decoder architecture. The decoder stage has skip connections from the encoder's activation blocks to fuse feature maps for higher resolution details. Monodepth outputs predicted disparities at four different scales, and then reconstructs images with backward mapping using a bilinear sampler. The network is trained to estimate the disparity maps for both views by sampling the opposite input images so that consistency between both disparity maps can be enforced for more accurate results. However, disparity maps in both cases are still generated from only the left image I^l for the sake of monocular depth estimation. During the test time, image reconstruction branches and right image pipelines are removed, and only the left disparity map output will be used in depth estimation of the input (left) image.

As disparity predictions are performed at four different scales, the loss function C is formed as the sum of losses at each output scale, $C = \sum_{s=1}^{4} C_s$. A loss module at each scale computes C_s as weighted combination of three terms,

$$C_s = \alpha_{ap}(C^l_{ap} + C^r_{ap}) + \alpha_{ds}(C^l_{ds} + C^r_{ds}) + \alpha_{lr}(C^l_{lr} + C^r_{lr}) \tag{5.5}$$

These terms are,

- The appearance matching loss, C_{ap}, enforces the reconstructed image to look similar to the original image,

$$C^l_{ap} = \frac{1}{N}\sum_{i,j} \alpha \frac{1 - SSIM(I^l_{ij}, \tilde{I}^l_{ij})}{2} + (1-\alpha)||I^l_{ij} - \tilde{I}^l_{ij}|| \tag{5.6}$$

  where I^l_{ij} is the input image, $\tilde{I}^l_{ij}$ is the reconstructed image, and N is the number of pixels.

- The disparity smoothness loss, C_{ds}, encourages disparities to be locally smooth with L_1 penalty on the disparity gradients ∂d,

$$C^l_{ds} = \frac{1}{N}\sum_{i,j} |\partial_x d^l_{ij}| e^{-||\partial_x I^l_{ij}||} + |\partial_y d^l_{ij}| e^{-||\partial_y I^l_{ij}||} \tag{5.7}$$

- The left-right disparity consistency loss, C_{lr}, enforces the left-view disparity map to be equal to the projected right-view disparity map,

$$C^l_{lr} = \frac{1}{N}\sum_{i,j} |d^l_{ij} - d^r_{ij+d^l_{ij}}| \tag{5.8}$$

Finally, during inference, Monodepth estimates the disparity at the finest scale that has the same resolution as the input image. By using the camera baseline and focal length, disparity map can be converted to the depth map.

5.3.2 Monodepth2

In order to close the gap between the monocular and binocular depth estimation networks, Monodepth2 [22] takes the idea of Monodepth further by introducing several innovations. Similar to Monodepth, Monodepth2 is also trained using image reconstruction as the supervisory signal. Self-supervision can be performed using synchronized stereo pairs or monocular videos. In self-supervised stereo training, disparities between the image pairs are predicted. On the other hand, in self-supervised monocular training, consecutive temporal video frames are used for training. As a result,

in addition to the disparities, the camera pose between the consecutive frames should be estimated too. The camera pose is only required for training, and not used during inference.

The overall architecture of Monodepth is very similar to Monodepth. Monodepth2 uses U-Net for its depth network, which is basically an encoder-decoder model. Given the input image I_t, the depth network estimates the depth map D_t. As one of the introduced innovations, a 4-layer CNN is used as the pose network which predicts the pose between a pair of frames. Similar to the Monodepth, loss function is composed of three terms. Given that I_t is the input image, $I_{t'}$ is the image pair, $T_{t\to t'}$ is the pose of $I_{t'}$ with respect to I_t, and D_p is the dense depth map,

- Photometric reprojection error, L_p,

$$L_p = \sum_{t'} pe(I_t, I_{t'\to t}), \tag{5.9}$$

$$\text{and} \qquad I_{t'\to t} = I_{t'}\left\langle proj(D_t, T_{t\to t'}, K)\right\rangle \tag{5.10}$$

  where $pe()$ is the photometric reconstruction error, e.g., L1 distance in pixel space, $proj()$ is the resulting 2D coordinates of the projected depth D_t in $T_t^{'}$, and $\left\langle\right\rangle$ is the bilinear sampling operator.

- Photometric error function, $pe()$,

$$pe(I_a, I_b) = \frac{\alpha}{2}(1 - SSIM(I_a, I_b)) + (1-\alpha)||I_a - I_b|| \tag{5.11}$$

- Edge-aware smoothness error, L_s,

$$L_s = |\partial_x, d_t^*|e^{-|\partial_x I_t|} + |\partial_y, d_t^*|e^{-|\partial_y I_t|} \tag{5.12}$$

  where $d_t^* = d_t/\overline{d_t}$ is the mean-normalized inverse depth.

For monocular training, two temporally adjacent frames to I_t are used as image pairs for the computation of reprojection error, i.e., $I_{t'} \in \{I_{t-1}, I_{t+1}\}$. Previous self-supervised monocular depth estimation models would average these errors to obtain the final reprojection error; however, this fails in cases where some pixels are occluded in an adjacent image and not occluded in the other because the occluded pixels

cause a high reprojection error. To tackle this problem, Monodepth2 uses the minimum of reprojection error at each pixel. Therefore, the final per-pixel reprojection loss becomes,

$$L_p = \min_{t'} pe(I_t, I_{t' \to t}) \tag{5.13}$$

Another contribution of Monodepth2 is a simple auto-masking method which filters out pixels whose appearance does not change from one frame to another. The reason behind this is that self-supervised monocular training assumes a static scene and a moving camera, and the performance is highly degraded when this assumption is not met. Monodepth2 applies a per-pixel binary mask $\mu \in \{0, 1\}$ to the loss. μ includes pixels where the reprojection error of the warped image $I_{t' \to t}$ is lower than that of the original image pair $I_t^{'}$, i.e.,

$$\mu = \big[\min_{t}^{'} pe(I_t, I_{t' \to t}) < \min_{t}^{'} pe(I_t, I_{t'}) \big] \tag{5.14}$$

where $[]$ is the Iverson bracket.

The final contribution of Monodepth2 is in its multi-scale estimation. During training, Monodepth performs multi-scale depth prediction and calculates the photometric reprojection error on images at the resolution of each decode layer. However, this results in holes in large low-texture regions and texture-copy artifacts. To remedy this problem, Monodepth2 upsamples the lower resolution depth maps to the input image resolution, and then calculate the losses. This also enforces the multi-scale depth maps to learn how to reconstruct the high resolution input target images as accurately as possible.

CHAPTER 6

PROPOSED METHODS

6.1 Motivation

Previous works have shown that object detection performances can be improved by incorporating depth information obtained from sensors such as LIDAR. However, neither RGB-D object detection nor instance segmentation is practical because of a couple of reasons:

- **Scarcity of labeled data:** Deep learning-based RGB object detection and instance segmentation algorithms have shown great improvements in recent years. Yet, these models are highly dependent on the training data. The quantity and quality of datasets can be seen as the bottleneck of deep learning-based models. To work with RGB images, not only there are plenty of benchmarks and datasets but also one can prepare their own dataset even just by using their cellphone cameras. However, contrary to the RGB images, depth data is expensive and difficult to acquire. There are only a few datasets available, and it is rather impractical to prepare high quality custom-made depth datasets.
- **Impractical application:** Even though a comprehensive RGB-D dataset can be obtained for offline training, depth information should still be supplied to the detection or segmentation model during inference. This requires the use of an RGB-D camera or a depth sensor along with an RGB camera all of which ideally have the same sensor characteristics as the cameras used to prepare the training dataset. Due to the high prices of depth sensors and their arduous synchronization with RGB sensors, RGB-D object detection and instance segmentation algorithms do not receive much attention from industrial applications.

6.2 Proposed Idea

In order to benefit from their superior performances, we aim to eliminate the drawbacks of RGB-D models by replacing sensor-based depth information with the estimated monocular depth maps. The possible advantages of incorporating monocular depth estimation into object detection and instance segmentation are:

- Any RGB dataset can be converted to an RGB-D training dataset.
- Since depth of the scene is estimated from the RGB data, they are both inherently synchronized.
- Single image depth estimation models do not require an extra depth sensor neither in training nor in inference.
- Any platform which uses an RGB-based object detection or instance segmentation model can be easily replaced with an RGB-D based model.

6.3 Proposed Models

The state-of-the-art monocular depth estimation, object detection and instance segmentation models are modified such that detection and segmentation models take as input both RGB images and depth maps. The proposed models are comprised of two stages. The first stage is where depth estimation takes place, and Monodepth2 is used to extract a 1-channel depth map from an input RGB image. We consider using two types of depth maps: grayscale or RGB. To obtain an RGB depth map, Jet colormap is applied to the extracted 1-channel depth map.

In the second stage, the input RGB image and the corresponding depth map are fed to the fusion module. We propose two types of fusion modules: concatenation and convolution. The output of the fusion module is then supplied to the second stage where either object detection or instance segmentation takes place. SOLOv2 and Mask RCNN are used for the instance segmentation stage, and Faster RCNN is used for object detection stage.

Both of the proposed fusion modules exploit the early fusion technique. Thus, there are no parallel branches for different modalities. Color and depth information are processed together through the layers of the aforementioned architectures. Since the proposed models use early fusion technique, there is no need to modify RGB-based object detection and instance segmentation architectures except for their first layers.

6.3.1 Fusion by Concatenation

A 3-channel RGB image is concatenated with either 1-channel depth map to form a 4-channel image or 3-channel depth map to form a 6-channel image. The fused image is then fed as input to either an object detection or an instance segmentation architecture. Since these models are originally developed for RGB images, convolutional filters in their input layers accept inputs with 3-channel only. As a result, their first layers should be modified such that they can convolute 4-channel and 6-channel input images.

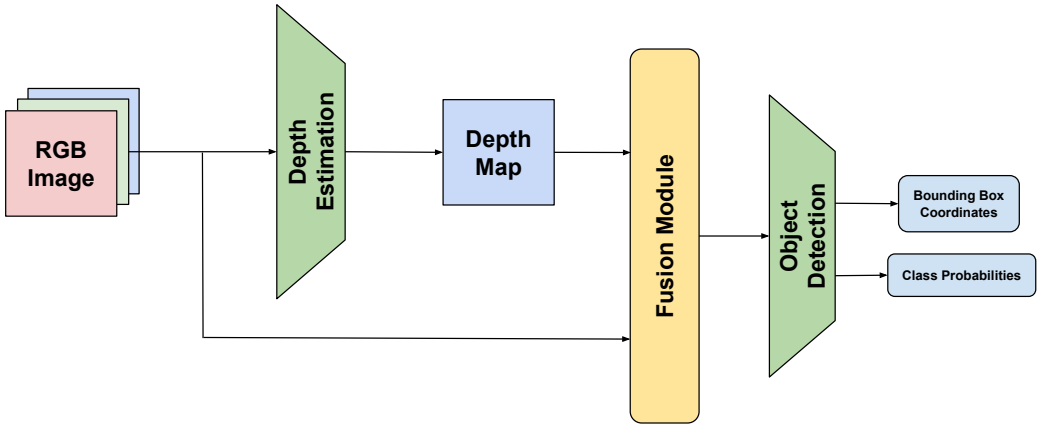


Figure 6.1: Proposed RGB-D object detection model.

6.3.2 Convolutional Fusion

Similar to the fusion by concatenation, a 3-channel RGB image is concatenated with either 1-channel depth map to form a 4-channel image or 3-channel depth map to form a 6-channel image. The fused image is then processed through a couple of convolutional layers. The first convolutional layer increases the channel size to 8 while the second layer decreases it to 3. Both layers preserve the original width and height of the input RGB image. The purpose of this mini-head network is to let the model learn how it can fuse the color and depth information in the most effective way when early fusion strategy is applied.

The output of the mini-head is then fed to either an object detection or an instance segmentation architecture. Since the output of the mini-head has 3 channels in depth, it can be directly supplied to the first layer of the following object detection or instance segmentation architecture without any further modification.

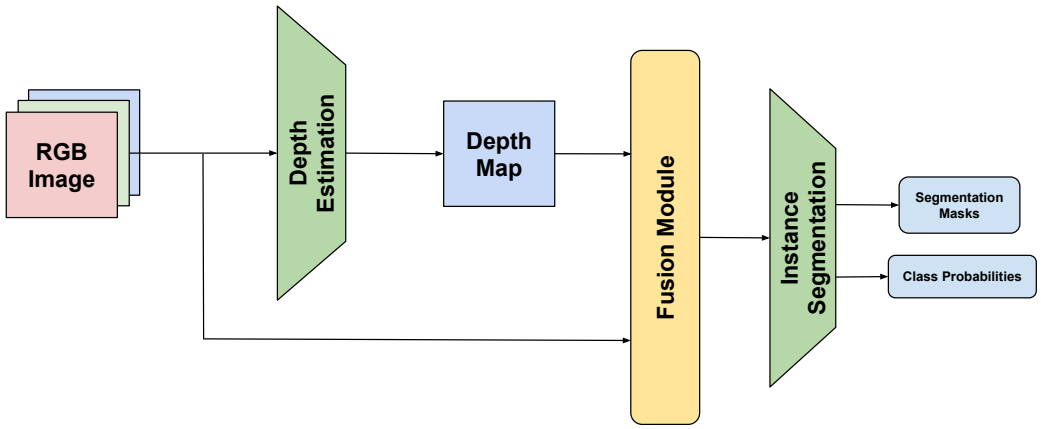


Figure 6.2: Proposed RGB-D instance segmentation model.

6.4 Datasets

The proposed models are trained and tested on SUN RGBD and our subset of MS-COCO Detection 2017 dataset. The annotations of SUN RGBD and MS-COCO 2017 have both bounding box coordinates and pixel-level segmentation masks; therefore,

both datasets can be used for object detection and instance segmentation tasks. However, MS-COCO 2017 is a very large dataset: it contains 120k images for training and 5k images for validation. Considering the capacity of available hardware in hand, training the proposed models on the whole dataset takes too much time e.g. 27 days on a single GPU of NVIDIA GeForce RTX 2080Ti. To accelerate the training process, a subset of MS-COCO Detection 2017 training dataset is used. The subset is prepared by sampling random 12k training images, corresponding to 10% of the original dataset. The random sampling is repeated until the subset has the same class distribution as the original dataset's. The original validation and test sets are kept since they are already small, and they do not affect the duration of the training process.

6.5 Experiments

To be able to measure the effect of proposed methods, a variety of experiments have been conducted using our MS-COCO subset. First, object detection and instance segmentation models are trained and tested with only RGB images. Their results make up the baseline for each proposed model, and they are referred to as RGB Image Only models. Likewise, these models are also trained and tested with only RGB depth maps. Their results indicate the worst-case scenario for each proposed model, and they are referred to as RGB Depth Only models. Then, to measure the effect of concatenating fusion module, two series of experiments are conducted. In the first series, models are trained and tested with RGB images and grayscale depth maps, and they are referred to as 4-channel Input models. In the second series, models are trained and tested with RGB images and RGB depth maps, and they are referred to as 6-channel Input models. Similarly, to measure the effect of convolutional fusion module, two series of experiments are executed. In the first series, models are trained and tested with RGB images and grayscale depth maps, and they are referred to as 4-to-3 Mapping models. In the second series, models are trained and tested with RGB images and RGB depth maps, and they are referred to as 6-to-3 Mapping models.

In order to assess how efficiently the *estimated* depth information is exploited by the proposed methods, the same experiments are repeated using SUN RGBD dataset. Since SUN RGBD contains *true* depth data along with RGB images, their results

make up the best-case scenario for each proposed model and allow us to compare one model with another in the most accurate way possible. Moreover, and most importantly, by comparing the performances of models trained and tested on our MS-COCO subset and SUN RGBD, we can deduce if *estimated* depth data gives as much information as *true* depth data does for instance segmentation and object detection tasks.

In all experiments, Monodepth2 is used for depth estimation stage. It is not included in the training process but used offline in order to accelerate the training. Before training, all depth maps are obtained by Monodepth2 with pre-trained weights *mono+stereo_640x192* [22]. In the second stage, SOLOv2 and Mask RCNN are used for instance segmentation tasks whereas Faster RCNN is used for object detection. Moreover, object detection head of Mask RCNN is also used for crosschecking Faster RCNN performance.

As backbone, all models use ResNet pre-trained on ImageNet. Hyperparameters are kept constant for the same instance segmentation or object detection architecture. Base learning rate is 0.01 for all SOLOv2 and Mask R-CNN models, and 0.02 for all Faster RCNN models. Batch size is kept 4 for all models. Number of iterations changes according to the training dataset because the SUN RGBD dataset has less than half of the images our MS-COCO subset has and this causes much earlier overfitting when SUN RGBD is used. Therefore, all models are trained for 270k iterations when our MS-COCO dataset is used, and for 90k iterations when SUN RGBD is used. To train the models with convolutional modules, ResNet weights are kept frozen during the first 5k iterations so that convolutional filters can be optimized much faster. Apart from this, all weights are updated during the whole training process of all models. Experiments are executed with 1 NVIDIA GeForce RTX 2080Ti.

6.5.1 Test Results

The test results of the aforementioned experiments are given from Table 6.1 to Table 6.8. We used MS-COCO notations for the metrics (see Section 3.2). Furthermore, the category-based AP results can be found in Appendix A.

Table 6.1: Object detection test results on SUN RGBD: Faster RCNN.

Model	AP (%)	AP50 (%)	AP75 (%)	APs (%)	APm (%)	APl (%)
RGB Image Only	16.702	**27.578**	17.663	**3.213**	**11.833**	23.021
4-channel Input	**16.936**	27.528	**18.036**	2.476	11.196	**23.913**
4-to-3 Mapping	16.138	26.046	17.217	2.437	10.160	22.944

Table 6.2: Object detection test results on MS-COCO subset: Faster RCNN.

Model	AP (%)	AP50 (%)	AP75 (%)	APs (%)	APm (%)	APl (%)
RGB Depth Only	5.944	11.438	5.364	0.435	3.222	12.595
RGB Image Only	23.215	38.712	24.495	10.852	24.062	31.396
4-channel Input	**25.482**	40.712	**27.544**	11.133	**25.896**	**35.181**
6-channel Input	25.269	**41.669**	26.736	**11.888**	25.403	34.431
4-to-3 Mapping	22.258	37.153	23.265	8.917	22.281	31.434
6-to-3 Mapping	23.013	39.673	23.801	10.073	23.624	31.838

Table 6.3: Object detection test results on SUN RGBD: Mask RCNN.

Model	AP (%)	AP50 (%)	AP75 (%)	APs (%)	APm (%)	APl (%)
RGB Image Only	17.052	**28.093**	17.835	**3.162**	**12.267**	23.481
4-channel Input	**17.207**	27.980	**18.178**	2.816	11.560	**24.194**
4-to-3 Mapping	16.492	26.695	17.597	2.262	10.188	23.416

Table 6.4: Object detection test results on MS-COCO subset: Mask RCNN.

Model	AP (%)	AP50 (%)	AP75 (%)	APs (%)	APm (%)	APl (%)
RGB Depth Only	7.188	13.648	6.798	0.644	4.373	14.449
RGB Image Only	25.365	**41.472**	27.099	**13.089**	25.866	33.739
4-channel Input	24.876	40.385	26.433	11.163	25.092	34.060
6-channel Input	**25.628**	40.911	**27.540**	11.759	**25.976**	**35.478**
4-to-3 Mapping	22.854	37.787	24.298	11.094	22.672	31.850
6-to-3 Mapping	23.488	38.749	24.933	9.722	23.778	33.043

Table 6.5: Instance segmentation test results on SUN RGBD: Mask RCNN.

Model	AP (%)	AP50 (%)	AP75 (%)	APs (%)	APm (%)	APl (%)
RGB Image Only	**14.215**	**26.571**	13.572	**1.545**	**8.536**	20.329
4-channel Input	14.210	26.482	**13.921**	1.281	7.741	**20.762**
4-to-3 Mapping	13.473	24.989	13.179	0.902	6.785	19.875

Table 6.6: Instance segmentation test results on MS-COCO subset: Mask RCNN.

Model	AP (%)	AP50 (%)	AP75 (%)	APs (%)	APm (%)	APl (%)
RGB Depth Only	6.892	12.293	6.810	0.340	3.689	15.377
RGB Image Only	**23.609**	**39.191**	**24.869**	**9.926**	**23.792**	34.958
4-channel Input	22.968	37.950	24.205	8.138	22.754	35.229
6-channel Input	23.307	38.480	24.420	8.662	23.614	**35.607**
4-to-3 Mapping	21.454	35.594	22.523	8.141	20.935	33.398
6-to-3 Mapping	21.686	36.363	22.375	6.986	22.125	33.550

Table 6.7: Instance segmentation test results on SUN RGBD: SOLOv2.

Model	AP (%)	AP50 (%)	AP75 (%)	APs (%)	APm (%)	APl (%)
RGB Image Only	**16.894**	**30.995**	**16.285**	**1.583**	**10.180**	**23.703**
4-channel Input	14.358	26.660	13.769	0.874	7.571	20.762
4-to-3 Mapping	16.458	29.687	16.243	1.057	8.769	23.607

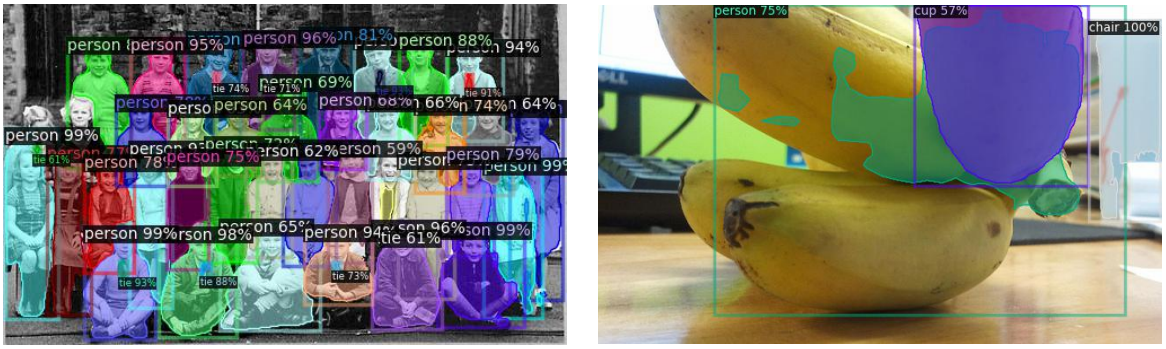
Table 6.8: Instance segmentation test results on MS-COCO subset: SOLOv2.

Model	AP (%)	AP50 (%)	AP75 (%)	APs (%)	APm (%)	APl (%)
RGB Depth Only	5.740	10.503	5.486	0.179	2.903	11.981
RGB Image Only	**23.888**	**38.565**	**25.098**	**8.006**	**25.164**	35.043
4-channel Input	21.454	36.655	21.702	6.492	22.407	32.683
6-channel Input	22.501	37.678	22.378	7.129	23.119	33.788
4-to-3 Mapping	22.363	36.881	22.897	6.737	23.102	**35.524**
6-to-3 Mapping	22.447	37.448	23.198	7.641	23.661	35.221

6.5.2 Discussion

6.5.2.1 Instance Segmentation

In instance segmentation experiments which are conducted with our MS-COCO subset, RGB Image Only models are consistently superior to the proposed methods. Original SOLOv2 trained with only RGB images achieves an AP of 23.888% while the 4-channel, 6-channel, 4-to-3 Mapping and 6-to-3 Mapping models achieve an AP of 21.454%, 22.501%, 22.363% and 22.447%, respectively. Likewise, original Mask RCNN trained with only RGB images achieves an AP of 23.609% while the 4-channel, 6-channel, 4-to-3 Mapping and 6-to-3 Mapping models achieve an AP of 22.968%, 23.307%, 21.454% and 21.686%, respectively. With both SOLOv2 and Mask RCNN architectures, 6-channel Input models exceed the performance of 4-Channel Input models. Similarly, 6-to-3 Mapping models yield better results than 4-to-3 Mapping models. This can be explained by the fact that the original SOLOv2 and Mask RCNN architectures are tuned for RGB data, and the pre-trained ResNet weights are obtained from RGB images only. Therefore, it is expected that these systems respond more positively when depth data is given in RGB format rather than in grayscale.



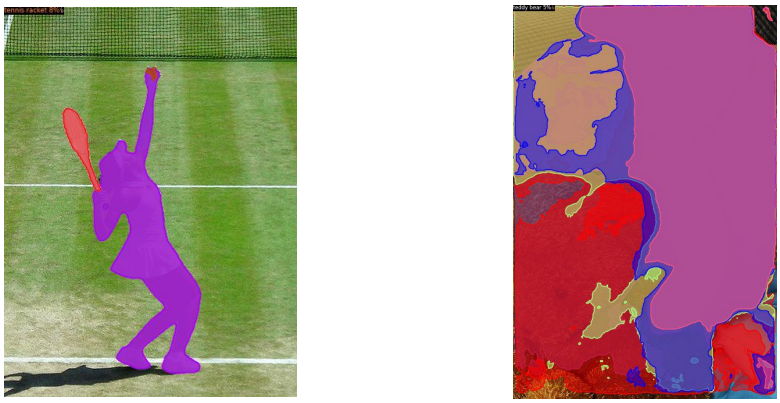
(a) Typical successful case.

(b) Typical failure case.

Figure 6.3: Inference results of the best proposed Mask RCNN model: 6-channel Input.

Even though the RGB Image Only models get the highest AP scores, the obtained results are very close to one another, independent of the original architecture and fusion module. This shows us that deep learning-based instance segmentation models intrinsically estimates and exploits depth information through their convolutional layers. As a result, explicitly feeding the estimated depth maps to these architectures did not yield any improvement.

To support this argument, we can refer to the results of instance segmentation experiments conducted with SUN RGBD dataset. RGB Image Only models are once again consistently superior to the proposed methods. Original SOLOv2 trained with only RGB images achieves an AP of 16.894% while the 4-channel and 4-to-3 Mapping models achieve an AP of 14.358% and 16.458%, respectively. Likewise, original Mask RCNN trained with only RGB images achieves an AP of 14.215% while the 4-channel and 4-to-3 Mapping models achieve an AP of 14.210% and 13.473%, respectively. Since SUN RGBD contains true depth data, these results can show us if there is a problem with the estimated depth data or the proposed methods cannot exploit any type of depth information. Since the results are consistent regardless of the dataset, we can conclude that deep learning-based instance segmentation models intrinsically estimate and exploit depth information, and any additional depth information is not necessary.



(a) Typical successful case.

(b) Typical failure case.

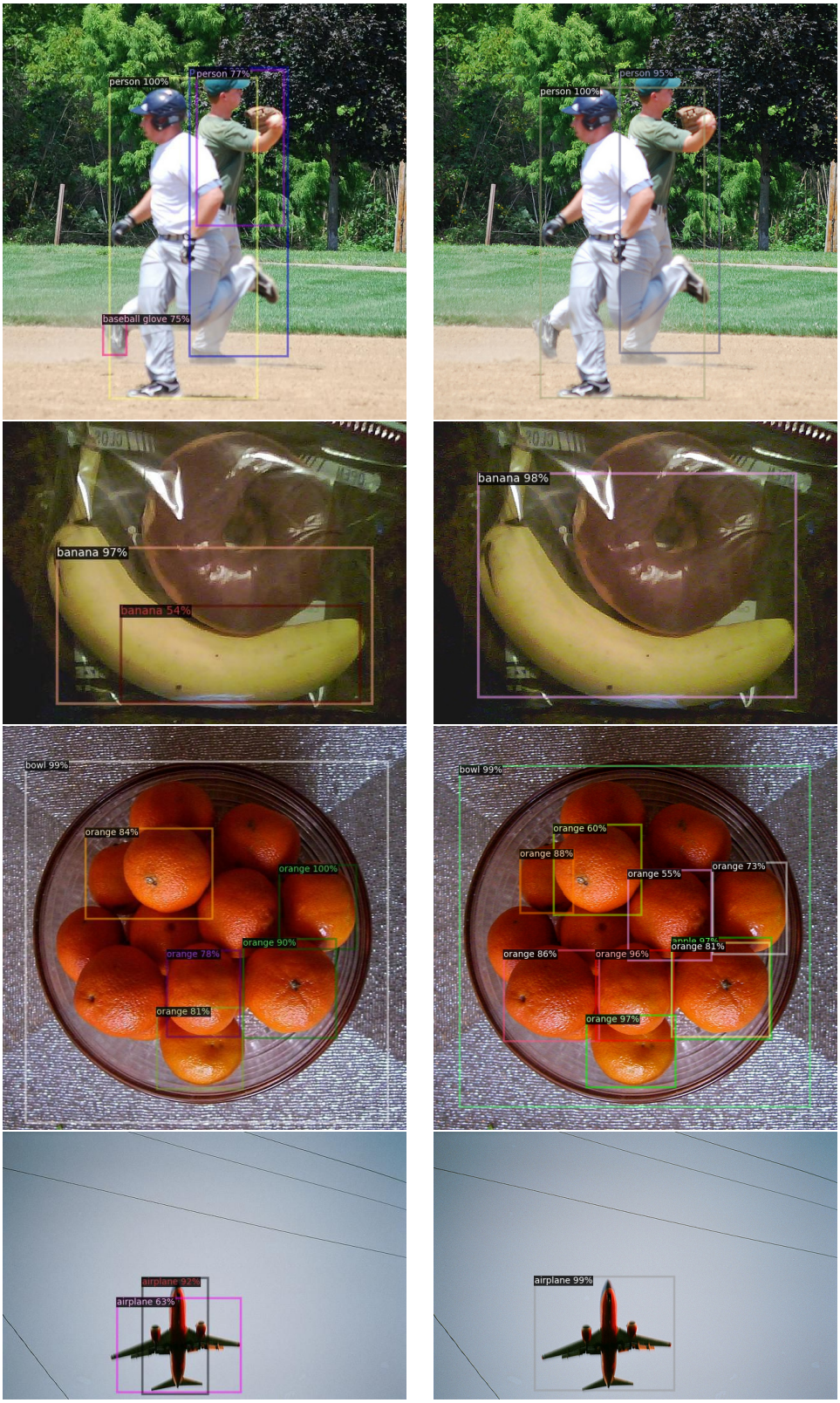
Figure 6.4: Inference results of the best proposed SOLOv2 model: 6-channel Input.

The lower AP results of the proposed models can be explained by the fact that the fused RGB and depth data is first processed through the ResNet backbone which is pre-trained on ImageNet dataset, and this dataset comprises only RGB images. Thus, it is natural to observe a small drop in performance when the RGB-trained ResNet weights face with depth data that is already extracted in the following layers and has nothing to bring forth.

6.5.2.2 Object Detection

In object detection experiments which are conducted with our MS-COCO subset, concatenating fusion models are consistently superior to the other methods. For Faster RCNN architecture, 4-channel model achieves an AP of 25.482% and 6-channel model achieves an AP of 25.269% while the RGB Image Only, 4-to-3 Mapping and 6-to-3 Mapping models achieve an AP of 23.215%, 22.258% and 23.013%, respectively. The performance of 4-channel and 6-channel Input models are very close to each other and the difference between them is that 6-channel Input model is better at detecting small objects with APs of 11.888% while the 4-channel Input model is better at detecting medium-size and large objects with APm and APl of 25.896% and 35.181%, respectively. Apart from this, Mask RCNN not only outputs segmentation masks but also bounding box coordinates, i.e. Mask RCNN involves Faster RCNN as a part of its architecture. Therefore, the object detection performance of Mask RCNN can also be used for crosschecking the sole Faster RCNN experiments. Once again, 6-channel Input model is superior to the other methods. 6-channel model achieves an AP of 25.628% whereas the RGB Image Only, 4-channel Input, 4-to-3 Mapping and 6-to-3 Mapping models achieve an AP of 25.365%, 24.876%, 22.854% and 23.488%, respectively.

As in instance segmentation experiments, 6-to-3 Mapping models yield better results than 4-to-3 Mapping models. Similarly, 6-channel Input models either exceed or equal to 4-Channel Input models. Once again, this can be explained by the fact that the original Faster RCNN is tuned for RGB data, and the pre-trained ResNet weights are obtained from RGB images only. Thus, it is expected that these systems respond more positively when depth data is given in RGB format rather than in grayscale.



(a) "RGB Image Only" Model Results

(b) "4-channel Input" Model Results

Figure 6.5: Comparison between some selected visual results of RGB Image Only and 4-channel Input models of Faster RCNN.

(a) Typical successful case.

(b) Typical failure case.

Figure 6.6: Inference results of the best Faster RCNN model: 4-channel Input.

Contrary to the instance segmentation architectures, object detection architectures can benefit from depth information and improve their performances. To support this argument, we can refer to the results of object detection experiments conducted with SUN RGBD dataset. Models with concatenating module are once again consistently superior to the RGB Image Only and convolutional fusion methods. 4-channel Input model Faster RCNN achieves an AP of 16.936% while the RGB Image Only and 4-to-3 Mapping models achieve an AP of 16.702% and 16.138%, respectively. Likewise, 4-channel Input model of Mask RCNN's object detection head achieves an AP of 17.207% while the RGB Image Only and 4-to-3 Mapping models achieve an AP of 17.052% and 16.492%, respectively. Since SUN RGBD contains true depth data, these results can show us if the proposed methods can exploit any type of additional depth information. As the results are consistent regardless of the dataset, we can conclude that any kind of depth data improves the object detection performance and the existing deep learning-based object detection models do not exploit it intrinsically. Therefore, by explicitly supplying estimated depth information, we could boost their performances.

CHAPTER 7

CONCLUSIONS & FUTURE WORKS

7.1 Conclusions

In this thesis, we studied whether or not it is possible to achieve improvements in performances when estimated dense-depth map is incorporated to the object detection and instance segmentation architectures. To be able to exploit both depth and color information, we proposed two fusion modules based on early fusion technique: fusion by concatenation and convolutional fusion. We then integrated these fusion modules to various RGB object detection and instance segmentation architectures, and tested them if any significant improvement is achieved. In experiments, we used only Monodepth2 for monocular depth estimation.

For object detection task, the fusion modules are incorporated into Faster RCNN. Fusion by concatenation models trained on MS-COCO dataset outperformed the RGB Image Only models. For Faster RCNN architecture, 4-channel model achieves an AP of 25.482% and 6-channel model achieves an AP of 25.269% while the RGB Image Only, 4-to-3 Mapping and 6-to-3 Mapping models achieve AP of 23.215%, 22.258% and 23.013%, respectively. The same tendency is observed when the models are trained on SUN-RGBD datasets, i.e., depth data is not estimated but obtained from sensors. 4-channel Input model Faster RCNN achieves an AP of 16.936% while the RGB Image Only and 4-to-3 Mapping models achieve AP of 16.702% and 16.138%, respectively. This tendency indicates that the existing deep learning-based object detection models do not exploit depth information intrinsically and thus explicitly using estimated depth information can boost their performances.

As for instance segmentation task, the proposed fusion modules are incorporated into

SOLOv2 and Mask RCNN architectures. Unlike object detection models, none of the modified instance segmentation models trained on MS-COCO surpassed the RGB Image Only models. SOLOv2 trained with only RGB images achieved an AP of 23.888% while the 4-channel, 6-channel, 4-to-3 Mapping and 6-to-3 Mapping models achieved AP of 21.454%, 22.501%, 22.363% and 22.447%, respectively. Likewise, Mask RCNN trained with only RGB images achieved an AP of 23.609% while the 4-channel, 6-channel, 4-to-3 Mapping and 6-to-3 Mapping models achieved AP of 22.968%, 23.307%, 21.454% and 21.686%, respectively. The same tendency is observed when the models are trained on SUN-RGBD datasets, i.e., depth data is not estimated but obtained from sensors. SOLOv2 trained with only RGB images achieves an AP of 16.894% while the 4-channel and 4-to-3 Mapping models achieve AP of 14.358% and 16.458%, respectively. Likewise, Mask RCNN trained with only RGB images achieves an AP of 14.215% while the 4-channel and 4-to-3 Mapping models achieve AP of 14.210% and 13.473%, respectively. This tendency indicates that deep learning-based instance segmentation models intrinsically estimates and exploits depth information, and explicitly using them do not provide any improvement.

To sum up, the conducted experiments show that state-of-the-art deep learning-based instance segmentation models extract and exploit the monocular dense-depth information from an input image, and do not require to be explicitly fed with such input data. However, state-of-the-art deep learning-based object detection models do not implicitly perform depth estimation and supplying any kind of depth data, e.g. estimated or acquired from a sensor, helps them improve their performances.

7.2 Future Works

In this thesis, only early fusion-based fusion modules are proposed and tested. As future work, late fusion or deep fusion-based fusion modules can be studied. As for the late fusion technique, two parallel networks of the same model can be used to train on RGB images and depth maps independently, and the output features of these branches can be fused for classification and localization layers. Likewise, as for the deep fusion technique, two parallel networks can be used to train on RGB images and depth maps independently, and the output features of these branches can

be fused and fed to a third CNN which also handles localization and classification at its last layer. Furthermore, more complex functions can be used in fusion module, e.g., element-wise multiplication followed by square-root operation, or depth map can be integrated into an attention mechanism which enhances certain features in certain regions and yield to better performances. Last but not least, multi-task learning can be employed instead of learning one specific task such as object detection or instance segmentation. Depth estimation, object detection, and instance segmentation tasks are not completely independent of each other, and they can benefit from the high performance of the other. Therefore, one may achieve an overall higher performance in each of these tasks when a common architecture is trained to learn how to carry out these tasks simultaneously instead of focusing on only one of them.

APPENDIX A

CATEGORY-BASED TEST RESULTS

A.1 Object Detection Test Results on SUN RGBD: Faster RCNN

Table A.1: Model: RGB Image Only

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
wall	20.026	floor	30.361	cabinet	15.687	bed	24.406
chair	25.796	sofa	23.204	table	20.460	door	20.150
window	13.126	bookshelf	20.821	picture	27.166	counter	17.980
blinds	17.984	desk	10.484	shelves	0.977	curtain	18.666
dresser	23.439	pillow	21.926	mirror	12.193	floormat	4.280
clothes	3.745	ceiling	23.742	books	5.495	fridge	16.003
tv	16.491	paper	6.659	towel	5.807	showercurtain	8.726
box	5.026	whiteboard	28.721	person	7.249	nightstand	29.305
toilet	30.815	sink	21.747	lamp	20.477	bathtub	15.570
bag	3.248						

Table A.2: Model: 4-channel Input

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
wall	19.903	floor	31.369	cabinet	16.530	bed	27.006
chair	25.567	sofa	23.842	table	21.021	door	17.704
window	10.927	bookshelf	19.441	picture	24.364	counter	19.004
blinds	16.637	desk	12.086	shelves	0.790	curtain	20.855
dresser	20.409	pillow	24.674	mirror	12.959	floormat	6.969
clothes	4.699	ceiling	18.797	books	4.709	fridge	20.343
tv	17.762	paper	6.858	towel	6.856	showercurtain	7.958
box	6.090	whiteboard	26.433	person	5.492	nightstand	28.245
toilet	31.330	sink	25.175	lamp	19.456	bathtub	21.185
bag	3.198						

Table A.3: Model: 4-to-3 Mapping

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
wall	18.459	floor	29.474	cabinet	15.276	bed	26.661
chair	24.470	sofa	22.728	table	20.054	door	16.884
window	9.703	bookshelf	20.177	picture	22.798	counter	18.582
blinds	16.026	desk	11.396	shelves	0.948	curtain	18.132
dresser	19.319	pillow	23.903	mirror	11.983	floormat	4.305
clothes	3.346	ceiling	18.549	books	4.510	fridge	20.092
tv	16.888	paper	6.156	towel	7.914	showercurtain	9.558
box	4.121	whiteboard	27.488	person	6.188	nightstand	25.553
toilet	32.236	sink	23.884	lamp	18.649	bathtub	17.614
bag	3.082						

A.2 Object Detection Test Results on MS-COCO Subset: Faster RCNN

Table A.4: Model: RGB Depth Only

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	18.860	bicycle	2.518	car	3.672	motorcycle	7.876
airplane	16.508	bus	22.418	train	20.657	truck	3.390
boat	0.725	trafficlight	0.852	firehydrant	26.831	stopsign	28.335
parkingmeter	10.625	bench	3.898	bird	3.945	cat	13.249
dog	6.537	horse	13.484	sheep	2.661	cow	9.082
elephant	15.934	bear	15.753	zebra	13.716	giraffe	21.807
backpack	0.378	umbrella	5.487	handbag	0.083	tie	0.549
suitcase	0.771	frisbee	1.579	skis	0.430	snowboard	0.267
sportsball	2.591	kite	1.477	baseballbat	1.791	baseballglove	1.673
skateboard	1.676	surfboard	3.223	tennisracket	3.826	bottle	2.548
wineglass	2.822	cup	4.979	fork	0.645	knife	0.034
spoon	0.814	bowl	4.443	banana	0.930	apple	0.321
sandwich	3.881	orange	3.232	broccoli	0.597	carrot	0.030
hotdog	1.223	pizza	9.316	donut	5.676	cake	2.488
chair	2.388	couch	9.543	pottedplant	1.139	bed	13.944
diningtable	12.316	toilet	19.585	tv	8.813	laptop	13.071
mouse	3.092	remote	0.031	keyboard	2.548	cellphone	0.963
microwave	1.790	oven	2.732	toaster	0.891	sink	3.602
refrigerator	9.801	book	0.127	clock	1.418	vase	4.415
scissors	0.268	teddybear	9.164	hairdrier	0.000	toothbrush	0.796

Table A.5: Model: RGB Image Only

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	43.594	bicycle	19.820	car	32.716	motorcycle	27.510
airplane	42.012	bus	48.167	train	42.792	truck	18.537
boat	12.623	trafficlight	18.575	firehydrant	48.694	stopsign	53.377
parkingmeter	22.996	bench	11.600	bird	19.544	cat	38.120
dog	32.715	horse	38.411	sheep	29.747	cow	32.013
elephant	41.915	bear	48.630	zebra	54.978	giraffe	54.506
backpack	6.537	umbrella	17.291	handbag	4.520	tie	18.674
suitcase	10.974	frisbee	47.880	skis	12.128	snowboard	14.634
sportsball	34.842	kite	29.720	baseballbat	14.172	baseballglove	18.746
skateboard	28.562	surfboard	19.705	tennisracket	28.677	bottle	23.158
wineglass	20.631	cup	25.509	fork	10.977	knife	5.109
spoon	3.071	bowl	25.722	banana	10.004	apple	4.543
sandwich	19.039	orange	15.316	broccoli	11.125	carrot	8.950
hotdog	12.923	pizza	38.473	donut	19.275	cake	15.231
chair	11.222	couch	22.716	pottedplant	11.183	bed	27.587
diningtable	16.673	toilet	41.809	tv	38.732	laptop	40.235
mouse	41.498	remote	12.729	keyboard	32.352	cellphone	18.405
microwave	33.584	oven	15.697	toaster	7.996	sink	19.943
refrigerator	23.895	book	6.866	clock	38.559	vase	18.384
scissors	3.691	teddybear	26.381	hairdrier	0.192	toothbrush	4.266

Table A.6: Model: 4-channel Input

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	43.692	bicycle	17.268	car	33.164	motorcycle	29.124
airplane	43.635	bus	47.359	train	44.601	truck	19.531
boat	14.543	trafficlight	18.088	firehydrant	50.027	stopsign	55.492
parkingmeter	27.292	bench	13.379	bird	16.722	cat	44.121
dog	40.639	horse	39.769	sheep	30.971	cow	33.197
elephant	42.709	bear	51.758	zebra	54.498	giraffe	52.118
backpack	7.128	umbrella	16.398	handbag	6.783	tie	23.187
suitcase	14.316	frisbee	47.073	skis	12.099	snowboard	16.330
sportsball	35.322	kite	24.874	baseballbat	14.287	baseballglove	22.142
skateboard	32.857	surfboard	20.405	tennisracket	29.801	bottle	21.777
wineglass	21.685	cup	26.255	fork	13.606	knife	4.636
spoon	4.072	bowl	25.179	banana	10.065	apple	3.854
sandwich	20.219	orange	17.370	broccoli	11.131	carrot	9.696
hotdog	14.329	pizza	39.604	donut	21.091	cake	16.793
chair	12.647	couch	23.705	pottedplant	11.595	bed	29.493
diningtable	18.526	toilet	43.510	tv	36.734	laptop	43.075
mouse	42.416	remote	13.806	keyboard	34.225	cellphone	20.519
microwave	33.608	oven	17.258	toaster	26.403	sink	22.344
refrigerator	31.999	book	5.799	clock	37.233	vase	18.110
scissors	10.797	teddybear	28.928	hairdrier	0.000	toothbrush	7.752

Table A.7: Model: 6-channel Input

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	44.363	bicycle	17.325	car	32.911	motorcycle	28.832
airplane	44.964	bus	47.721	train	43.180	truck	20.684
boat	14.361	trafficlight	19.483	firehydrant	49.693	stopsign	54.041
parkingmeter	28.838	bench	13.247	bird	18.317	cat	40.757
dog	35.332	horse	39.534	sheep	29.938	cow	32.728
elephant	43.362	bear	46.946	zebra	53.478	giraffe	54.597
backpack	6.206	umbrella	19.250	handbag	5.595	tie	22.158
suitcase	12.424	frisbee	45.255	skis	10.736	snowboard	19.251
sportsball	35.095	kite	24.475	baseballbat	14.585	baseballglove	20.748
skateboard	30.699	surfboard	20.377	tennisracket	29.311	bottle	23.174
wineglass	20.579	cup	26.888	fork	12.492	knife	4.533
spoon	2.976	bowl	26.227	banana	11.595	apple	5.129
sandwich	21.598	orange	19.116	broccoli	13.579	carrot	10.318
hotdog	16.678	pizza	38.817	donut	21.346	cake	18.525
chair	12.827	couch	24.012	pottedplant	11.542	bed	30.739
diningtable	19.105	toilet	41.258	tv	39.620	laptop	40.987
mouse	43.499	remote	12.978	keyboard	33.230	cellphone	20.234
microwave	33.033	oven	17.239	toaster	21.485	sink	24.129
refrigerator	29.535	book	6.488	clock	38.027	vase	17.590
scissors	8.772	teddybear	26.042	hairdrier	0.000	toothbrush	4.779

Table A.8: Model: 4-to-3 Mapping

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	40.649	bicycle	17.590	car	29.981	motorcycle	25.301
airplane	41.735	bus	46.944	train	39.034	truck	17.509
boat	11.857	trafficlight	15.850	firehydrant	47.499	stopsign	53.220
parkingmeter	23.837	bench	10.989	bird	16.426	cat	41.349
dog	33.975	horse	35.017	sheep	24.028	cow	29.400
elephant	37.149	bear	47.462	zebra	47.665	giraffe	49.608
backpack	6.750	umbrella	16.336	handbag	4.215	tie	20.105
suitcase	8.795	frisbee	37.645	skis	9.230	snowboard	11.449
sportsball	28.170	kite	22.445	baseballbat	12.357	baseballglove	16.960
skateboard	29.186	surfboard	16.812	tennisracket	26.288	bottle	20.393
wineglass	19.881	cup	23.569	fork	9.066	knife	4.300
spoon	2.558	bowl	24.643	banana	8.633	apple	3.630
sandwich	14.419	orange	12.910	broccoli	9.527	carrot	5.150
hotdog	10.515	pizza	34.170	donut	15.846	cake	12.587
chair	10.817	couch	21.862	pottedplant	7.801	bed	30.439
diningtable	17.920	toilet	39.278	tv	35.047	laptop	38.791
mouse	42.736	remote	9.904	keyboard	34.065	cellphone	16.925
microwave	30.934	oven	14.981	toaster	9.505	sink	19.958
refrigerator	27.402	book	5.632	clock	35.427	vase	14.868
scissors	8.162	teddybear	23.190	hairdrier	0.000	toothbrush	2.394

Table A.9: Model: 6-to-3 Mapping

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	41.818	bicycle	17.904	car	30.500	motorcycle	25.650
airplane	40.605	bus	47.375	train	38.741	truck	17.011
boat	12.720	trafficlight	16.349	firehydrant	46.620	stopsign	53.841
parkingmeter	23.134	bench	11.392	bird	19.187	cat	39.985
dog	34.040	horse	36.883	sheep	28.135	cow	33.571
elephant	42.111	bear	44.671	zebra	50.826	giraffe	52.727
backpack	6.097	umbrella	18.323	handbag	3.338	tie	20.824
suitcase	11.015	frisbee	39.055	skis	9.675	snowboard	12.527
sportsball	25.814	kite	20.124	baseballbat	12.651	baseballglove	18.746
skateboard	27.962	surfboard	19.460	tennisracket	26.592	bottle	22.172
wineglass	19.564	cup	25.664	fork	9.810	knife	2.977
spoon	2.569	bowl	25.952	banana	8.402	apple	6.570
sandwich	15.037	orange	14.404	broccoli	10.742	carrot	6.234
hotdog	12.289	pizza	33.769	donut	19.323	cake	14.696
chair	12.526	couch	20.490	pottedplant	7.966	bed	28.953
diningtable	18.521	toilet	39.685	tv	35.771	laptop	41.228
mouse	42.695	remote	10.946	keyboard	32.224	cellphone	18.512
microwave	32.314	oven	15.727	toaster	10.594	sink	19.765
refrigerator	26.058	book	5.940	clock	37.301	vase	17.001
scissors	10.662	teddybear	25.762	hairdrier	0.000	toothbrush	2.242

A.3 Object Detection Test Results on SUN RGBD: Mask RCNN

Table A.10: Model: RGB Image Only

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
wall	20.387	floor	30.920	cabinet	16.678	bed	25.084
chair	26.188	sofa	23.617	table	21.164	door	21.047
window	13.189	bookshelf	19.984	picture	27.262	counter	18.273
blinds	19.381	desk	10.312	shelves	0.811	curtain	19.928
dresser	25.240	pillow	22.013	mirror	12.171	floormat	4.549
clothes	4.193	ceiling	24.218	books	5.724	fridge	16.352
tv	17.934	paper	6.518	towel	6.634	showercurtain	7.083
box	5.513	whiteboard	28.448	person	7.272	nightstand	29.724
toilet	29.790	sink	21.725	lamp	21.405	bathtub	16.226
bag	3.964						

Table A.11: Model: 4-channel Input

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
wall	20.241	floor	31.735	cabinet	16.984	bed	27.215
chair	25.954	sofa	24.259	table	21.530	door	18.485
window	11.431	bookshelf	20.664	picture	24.223	counter	18.546
blinds	17.294	desk	11.882	shelves	0.975	curtain	21.039
dresser	20.493	pillow	24.468	mirror	14.740	floormat	5.321
clothes	5.150	ceiling	20.158	books	5.094	fridge	17.946
tv	17.065	paper	7.014	towel	6.032	showercurtain	9.963
box	6.021	whiteboard	27.061	person	6.259	nightstand	28.382
toilet	33.314	sink	24.667	lamp	20.101	bathtub	21.282
bag	3.666						

Table A.12: Model: 4-to-3 Mapping

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
wall	19.435	floor	29.923	cabinet	15.471	bed	27.185
chair	24.529	sofa	22.544	table	20.116	door	16.764
window	9.890	bookshelf	19.910	picture	22.958	counter	17.600
blinds	15.481	desk	10.555	shelves	0.591	curtain	18.090
dresser	20.863	pillow	23.592	mirror	12.406	floormat	3.962
clothes	3.836	ceiling	17.864	books	4.372	fridge	17.650
tv	18.077	paper	6.190	towel	7.910	showercurtain	17.643
box	4.656	whiteboard	27.796	person	6.667	nightstand	27.434
toilet	31.710	sink	25.143	lamp	18.193	bathtub	19.354
bag	3.849						

A.4 Object Detection Test Results on MS-COCO Subset: Mask RCNN

Table A.13: Model: RGB Depth Only

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	20.921	bicycle	3.297	car	4.737	motorcycle	8.488
airplane	17.966	bus	25.287	train	23.690	truck	4.815
boat	1.180	trafficlight	0.956	firehydrant	24.711	stopsign	28.456
parkingmeter	10.824	bench	4.782	bird	4.069	cat	18.222
dog	10.246	horse	14.467	sheep	4.276	cow	10.749
elephant	14.706	bear	17.933	zebra	16.685	giraffe	24.690
backpack	0.181	umbrella	6.180	handbag	0.322	tie	1.204
suitcase	1.869	frisbee	3.632	skis	1.022	snowboard	0.871
sportsball	2.553	kite	2.371	baseballbat	3.666	baseballglove	1.642
skateboard	3.089	surfboard	4.768	tennisracket	6.030	bottle	3.567
wineglass	3.446	cup	6.310	fork	0.520	knife	0.098
spoon	0.072	bowl	6.516	banana	1.130	apple	0.789
sandwich	5.781	orange	4.389	broccoli	1.185	carrot	0.025
hotdog	1.320	pizza	11.780	donut	7.686	cake	2.823
chair	3.542	couch	10.303	pottedplant	0.841	bed	20.193
diningtable	12.993	toilet	24.954	tv	11.906	laptop	16.707
mouse	4.777	remote	0.064	keyboard	5.270	cellphone	1.456
microwave	5.634	oven	3.545	toaster	0.216	sink	6.170
refrigerator	11.676	book	0.427	clock	2.254	vase	3.905
scissors	1.099	teddybear	13.697	hairdrier	0.000	toothbrush	0.405

Table A.14: Model: RGB Image Only

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	45.273	bicycle	18.461	car	33.672	motorcycle	28.759
airplane	43.304	bus	50.684	train	46.417	truck	19.261
boat	13.568	trafficlight	17.623	firehydrant	48.828	stopsign	55.307
parkingmeter	27.150	bench	12.813	bird	20.623	cat	39.825
dog	33.682	horse	39.329	sheep	30.600	cow	32.782
elephant	45.366	bear	47.807	zebra	54.513	giraffe	54.839
backpack	6.752	umbrella	19.551	handbag	5.546	tie	19.784
suitcase	11.152	frisbee	46.858	skis	12.824	snowboard	14.526
sportsball	35.857	kite	31.329	baseballbat	14.676	baseballglove	21.043
skateboard	31.495	surfboard	22.516	tennisracket	27.257	bottle	25.834
wineglass	22.813	cup	27.900	fork	11.501	knife	5.010
spoon	4.546	bowl	26.210	banana	9.777	apple	4.019
sandwich	21.288	orange	17.081	broccoli	11.231	carrot	9.758
hotdog	13.328	pizza	40.865	donut	19.668	cake	15.992
chair	13.068	couch	26.285	pottedplant	11.778	bed	30.115
diningtable	16.900	toilet	42.473	tv	39.710	laptop	42.593
mouse	42.572	remote	15.843	keyboard	31.511	cellphone	20.715
microwave	35.022	oven	17.030	toaster	19.211	sink	22.387
refrigerator	28.752	book	6.904	clock	37.636	vase	20.052
scissors	4.989	teddybear	28.602	hairdrier	0.696	toothbrush	5.874

Table A.15: Model: 4-channel Input

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	45.285	bicycle	17.734	car	33.843	motorcycle	29.286
airplane	44.833	bus	49.698	train	46.273	truck	20.593
boat	15.060	trafficlight	19.114	firehydrant	51.606	stopsign	56.685
parkingmeter	24.723	bench	14.927	bird	17.468	cat	41.563
dog	35.219	horse	36.669	sheep	31.526	cow	31.014
elephant	43.160	bear	47.304	zebra	53.597	giraffe	55.139
backpack	5.821	umbrella	18.891	handbag	4.836	tie	22.354
suitcase	9.262	frisbee	45.515	skis	11.208	snowboard	16.604
sportsball	33.431	kite	24.596	baseballbat	13.565	baseballglove	20.305
skateboard	32.196	surfboard	19.696	tennisracket	28.119	bottle	23.384
wineglass	21.464	cup	26.040	fork	14.157	knife	5.214
spoon	3.574	bowl	25.524	banana	11.352	apple	3.836
sandwich	18.862	orange	17.932	broccoli	13.480	carrot	9.101
hotdog	15.073	pizza	38.356	donut	15.914	cake	15.545
chair	13.164	couch	25.831	pottedplant	10.567	bed	29.460
diningtable	18.514	toilet	45.321	tv	38.217	laptop	41.781
mouse	43.205	remote	13.340	keyboard	32.757	cellphone	17.745
microwave	28.941	oven	17.659	toaster	12.707	sink	20.509
refrigerator	30.934	book	6.158	clock	36.312	vase	19.575
scissors	8.868	teddybear	26.064	hairdrier	0.000	toothbrush	4.896

Table A.16: Model: 6-channel Input

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	44.533	bicycle	17.936	car	33.609	motorcycle	30.074
airplane	46.428	bus	51.008	train	46.480	truck	20.556
boat	15.208	trafficlight	17.838	firehydrant	50.521	stopsign	53.911
parkingmeter	27.888	bench	13.900	bird	18.398	cat	44.972
dog	40.034	horse	39.059	sheep	31.087	cow	31.935
elephant	44.031	bear	47.132	zebra	54.478	giraffe	51.562
backpack	8.272	umbrella	18.432	handbag	6.536	tie	22.050
suitcase	13.593	frisbee	47.341	skis	12.738	snowboard	20.125
sportsball	36.574	kite	26.137	baseballbat	14.206	baseballglove	21.771
skateboard	31.238	surfboard	21.003	tennisracket	27.701	bottle	22.166
wineglass	21.721	cup	27.000	fork	14.056	knife	5.711
spoon	4.436	bowl	25.693	banana	9.233	apple	3.635
sandwich	20.630	orange	18.419	broccoli	12.399	carrot	10.592
hotdog	15.667	pizza	39.052	donut	20.127	cake	14.734
chair	12.652	couch	22.888	pottedplant	10.308	bed	29.808
diningtable	17.903	toilet	40.883	tv	37.287	laptop	43.416
mouse	43.221	remote	15.504	keyboard	35.329	cellphone	20.259
microwave	33.021	oven	17.321	toaster	20.238	sink	22.689
refrigerator	34.266	book	6.296	clock	34.609	vase	18.725
scissors	13.348	teddybear	27.022	hairdrier	0.000	toothbrush	7.680

Table A.17: Model: 4-to-3 Mapping

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	43.161	bicycle	18.152	car	30.826	motorcycle	28.591
airplane	43.648	bus	46.129	train	43.705	truck	19.350
boat	12.653	trafficlight	15.581	firehydrant	49.785	stopsign	52.382
parkingmeter	26.449	bench	10.638	bird	17.244	cat	40.435
dog	33.454	horse	35.085	sheep	27.707	cow	31.169
elephant	38.924	bear	44.197	zebra	50.785	giraffe	52.228
backpack	5.224	umbrella	17.308	handbag	4.861	tie	19.689
suitcase	9.830	frisbee	38.736	skis	9.471	snowboard	11.237
sportsball	28.368	kite	22.609	baseballbat	11.669	baseballglove	17.841
skateboard	26.568	surfboard	17.398	tennisracket	25.637	bottle	21.746
wineglass	21.169	cup	26.397	fork	9.791	knife	3.712
spoon	2.404	bowl	27.566	banana	5.776	apple	3.248
sandwich	14.552	orange	12.088	broccoli	10.295	carrot	6.480
hotdog	12.029	pizza	35.306	donut	14.626	cake	12.769
chair	12.341	couch	24.284	pottedplant	8.555	bed	28.356
diningtable	18.197	toilet	43.195	tv	37.138	laptop	41.095
mouse	41.987	remote	11.286	keyboard	32.802	cellphone	16.473
microwave	27.840	oven	16.425	toaster	10.662	sink	21.262
refrigerator	29.625	book	5.319	clock	35.882	vase	16.879
scissors	3.641	teddybear	24.781	hairdrier	0.125	toothbrush	1.508

Table A.18: Model: 6-to-3 Mapping

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	42.150	bicycle	19.134	car	30.945	motorcycle	27.975
airplane	44.127	bus	50.004	train	39.752	truck	17.602
boat	13.592	trafficlight	16.356	firehydrant	47.414	stopsign	52.683
parkingmeter	27.055	bench	12.414	bird	19.597	cat	43.310
dog	33.896	horse	38.654	sheep	26.667	cow	30.022
elephant	42.221	bear	48.051	zebra	51.019	giraffe	53.625
backpack	6.106	umbrella	16.765	handbag	4.780	tie	19.708
suitcase	10.350	frisbee	43.642	skis	9.506	snowboard	15.835
sportsball	27.712	kite	22.178	baseballbat	11.944	baseballglove	17.666
skateboard	28.901	surfboard	18.308	tennisracket	26.732	bottle	20.888
wineglass	21.870	cup	25.144	fork	12.031	knife	4.955
spoon	2.324	bowl	25.842	banana	7.693	apple	4.523
sandwich	15.996	orange	15.135	broccoli	11.290	carrot	5.757
hotdog	11.339	pizza	35.578	donut	18.727	cake	14.232
chair	11.968	couch	22.627	pottedplant	8.517	bed	25.868
diningtable	18.407	toilet	40.949	tv	38.065	laptop	42.416
mouse	42.495	remote	11.388	keyboard	32.348	cellphone	19.926
microwave	32.873	oven	14.797	toaster	8.534	sink	20.086
refrigerator	30.949	book	6.149	clock	36.175	vase	14.881
scissors	7.746	teddybear	26.014	hairdrier	0.000	toothbrush	4.148

A.5 Instance Segmentation Test Results on SUN RGBD: Mask RCNN

Table A.19: Model: RGB Image Only

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
wall	18.719	floor	30.601	cabinet	13.998	bed	19.147
chair	18.533	sofa	16.610	table	11.044	door	17.025
window	11.754	bookshelf	15.495	picture	27.471	counter	10.187
blinds	19.651	desk	6.642	shelves	0.366	curtain	20.081
dresser	21.184	pillow	18.916	mirror	9.478	floormat	3.654
clothes	3.657	ceiling	20.739	books	5.087	fridge	14.400
tv	17.445	paper	5.303	towel	6.227	showercurtain	8.205
box	4.502	whiteboard	25.778	person	5.332	nightstand	23.719
toilet	24.904	sink	18.205	lamp	14.363	bathtub	13.816
bag	3.727						

Table A.20: Model: 4-channel Input

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
wall	18.215	floor	31.018	cabinet	14.834	bed	21.075
chair	18.447	sofa	17.088	table	11.666	door	14.240
window	9.956	bookshelf	16.242	picture	24.066	counter	10.088
blinds	17.479	desk	7.811	shelves	0.473	curtain	20.115
dresser	17.238	pillow	20.854	mirror	10.973	floormat	3.123
clothes	3.963	ceiling	17.262	books	4.402	fridge	15.917
tv	16.884	paper	5.742	towel	5.942	showercurtain	10.515
box	4.997	whiteboard	24.325	person	5.633	nightstand	22.755
toilet	27.881	sink	20.207	lamp	13.814	bathtub	17.024
bag	3.514						

Table A.21: Model: 4-to-3 Mapping

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
wall	17.159	floor	29.287	cabinet	12.975	bed	20.576
chair	17.409	sofa	15.958	table	10.514	door	12.647
window	8.659	bookshelf	15.104	picture	22.868	counter	8.773
blinds	15.564	desk	6.581	shelves	0.163	curtain	17.612
dresser	16.808	pillow	20.032	mirror	8.818	floormat	3.151
clothes	3.016	ceiling	15.182	books	3.893	fridge	15.151
tv	18.103	paper	4.892	towel	7.180	showercurtain	17.055
box	3.926	whiteboard	25.167	person	5.650	nightstand	21.340
toilet	27.060	sink	20.403	lamp	11.904	bathtub	14.553
bag	3.348						

A.6 Instance Segmentation Test Results on MS-COCO Subset: Mask RCNN

Table A.22: Model: RGB Depth Only

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	16.381	bicycle	1.603	car	4.346	motorcycle	6.383
airplane	17.385	bus	27.744	train	27.406	truck	4.995
boat	1.284	trafficlight	1.272	firehydrant	26.556	stopsign	30.524
parkingmeter	14.584	bench	3.190	bird	4.332	cat	20.953
dog	10.160	horse	9.218	sheep	3.818	cow	9.327
elephant	13.240	bear	19.697	zebra	10.017	giraffe	17.754
backpack	0.064	umbrella	8.935	handbag	0.243	tie	0.461
suitcase	2.216	frisbee	4.068	skis	0.000	snowboard	0.892
sportsball	2.798	kite	1.884	baseballbat	1.779	baseballglove	1.616
skateboard	0.739	surfboard	3.253	tennisracket	8.409	bottle	3.682
wineglass	2.531	cup	6.773	fork	0.048	knife	0.082
spoon	0.025	bowl	6.478	banana	0.812	apple	0.895
sandwich	6.716	orange	4.882	broccoli	1.363	carrot	0.024
hotdog	0.597	pizza	11.878	donut	9.383	cake	3.162
chair	1.806	couch	7.740	pottedplant	0.812	bed	13.987
diningtable	6.124	toilet	27.720	tv	12.793	laptop	18.782
mouse	5.228	remote	0.018	keyboard	4.746	cellphone	1.731
microwave	6.364	oven	3.848	toaster	0.243	sink	6.687
refrigerator	12.506	book	0.370	clock	2.479	vase	4.226
scissors	0.155	teddybear	12.906	hairdrier	0.000	toothbrush	1.208

Table A.23: Model: RGB Image Only

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	39.181	bicycle	10.194	car	31.444	motorcycle	20.506
airplane	34.987	bus	50.502	train	47.840	truck	19.572
boat	11.385	trafficlight	17.606	firehydrant	49.071	stopsign	57.654
parkingmeter	30.558	bench	8.832	bird	17.698	cat	45.304
dog	34.553	horse	28.662	sheep	26.554	cow	28.212
elephant	41.084	bear	48.239	zebra	46.640	giraffe	40.880
backpack	6.377	umbrella	27.285	handbag	6.416	tie	20.084
suitcase	11.918	frisbee	47.772	skis	0.760	snowboard	7.125
sportsball	36.305	kite	23.342	baseballbat	13.453	baseballglove	23.678
skateboard	15.679	surfboard	18.792	tennisracket	37.915	bottle	25.216
wineglass	19.726	cup	28.388	fork	4.575	knife	3.427
spoon	2.541	bowl	25.176	banana	7.513	apple	3.989
sandwich	22.904	orange	17.644	broccoli	11.061	carrot	9.095
hotdog	11.730	pizza	40.241	donut	19.663	cake	16.884
chair	8.071	couch	21.682	pottedplant	10.782	bed	21.585
diningtable	7.504	toilet	43.216	tv	42.298	laptop	43.750
mouse	44.505	remote	15.389	keyboard	31.088	cellphone	21.347
microwave	37.123	oven	15.890	toaster	21.381	sink	20.838
refrigerator	31.248	book	3.756	clock	38.368	vase	19.448
scissors	3.932	teddybear	26.277	hairdrier	0.743	toothbrush	4.636

Table A.24: Model: 4-channel Input

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	38.679	bicycle	10.335	car	31.597	motorcycle	20.815
airplane	36.047	bus	50.660	train	48.146	truck	19.851
boat	12.357	trafficlight	18.598	firehydrant	52.262	stopsign	56.527
parkingmeter	26.498	bench	9.976	bird	15.420	cat	46.769
dog	34.547	horse	27.025	sheep	26.795	cow	27.177
elephant	39.500	bear	48.170	zebra	45.392	giraffe	39.685
backpack	5.201	umbrella	26.614	handbag	5.589	tie	20.863
suitcase	10.541	frisbee	46.419	skis	0.827	snowboard	9.353
sportsball	33.910	kite	18.187	baseballbat	11.538	baseballglove	23.466
skateboard	15.624	surfboard	16.802	tennisracket	37.558	bottle	22.827
wineglass	19.961	cup	26.843	fork	5.791	knife	3.364
spoon	2.278	bowl	23.393	banana	8.517	apple	3.675
sandwich	19.777	orange	17.906	broccoli	12.599	carrot	8.541
hotdog	13.870	pizza	37.601	donut	16.356	cake	16.782
chair	7.957	couch	20.987	pottedplant	9.995	bed	22.593
diningtable	8.300	toilet	46.239	tv	40.501	laptop	43.087
mouse	43.620	remote	13.928	keyboard	31.133	cellphone	20.032
microwave	29.880	oven	16.669	toaster	10.242	sink	19.756
refrigerator	33.404	book	3.063	clock	36.859	vase	18.939
scissors	6.330	teddybear	24.153	hairdrier	0.283	toothbrush	4.090

Table A.25: Model: 6-channel Input

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	37.556	bicycle	10.438	car	30.799	motorcycle	22.111
airplane	34.685	bus	49.920	train	47.514	truck	20.698
boat	12.763	trafficlight	17.284	firehydrant	49.462	stopsign	52.980
parkingmeter	29.578	bench	9.308	bird	15.665	cat	47.777
dog	39.041	horse	27.947	sheep	26.641	cow	26.812
elephant	39.494	bear	48.397	zebra	45.166	giraffe	38.134
backpack	7.426	umbrella	27.300	handbag	6.489	tie	20.474
suitcase	14.735	frisbee	47.314	skis	0.893	snowboard	10.405
sportsball	36.188	kite	17.917	baseballbat	14.468	baseballglove	24.627
skateboard	15.280	surfboard	18.132	tennisracket	37.153	bottle	21.328
wineglass	19.400	cup	27.510	fork	6.187	knife	3.939
spoon	2.363	bowl	24.012	banana	6.679	apple	3.514
sandwich	20.931	orange	17.567	broccoli	12.299	carrot	9.875
hotdog	14.694	pizza	38.715	donut	20.853	cake	15.458
chair	7.522	couch	18.625	pottedplant	8.922	bed	20.904
diningtable	6.973	toilet	42.625	tv	38.994	laptop	43.075
mouse	42.960	remote	16.345	keyboard	33.675	cellphone	20.514
microwave	31.958	oven	15.786	toaster	22.139	sink	21.164
refrigerator	34.941	book	3.353	clock	35.126	vase	18.093
scissors	8.007	teddybear	23.613	hairdrier	0.000	toothbrush	4.980

Table A.26: Model: 4-to-3 Mapping

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	36.788	bicycle	10.304	car	28.689	motorcycle	20.886
airplane	35.672	bus	47.284	train	46.698	truck	19.849
boat	11.076	trafficlight	15.054	firehydrant	50.999	stopsign	54.586
parkingmeter	28.533	bench	7.236	bird	14.999	cat	45.090
dog	32.799	horse	24.857	sheep	24.588	cow	26.890
elephant	36.736	bear	45.742	zebra	43.289	giraffe	38.426
backpack	5.270	umbrella	24.882	handbag	5.793	tie	18.958
suitcase	11.316	frisbee	38.526	skis	0.509	snowboard	5.880
sportsball	28.928	kite	15.943	baseballbat	10.648	baseballglove	19.674
skateboard	12.887	surfboard	14.190	tennisracket	35.943	bottle	22.193
wineglass	19.251	cup	27.111	fork	4.431	knife	2.936
spoon	1.393	bowl	25.286	banana	4.830	apple	3.279
sandwich	16.663	orange	12.406	broccoli	9.623	carrot	5.956
hotdog	10.663	pizza	35.325	donut	15.296	cake	13.734
chair	7.812	couch	20.231	pottedplant	8.251	bed	21.948
diningtable	9.454	toilet	45.711	tv	38.534	laptop	41.104
mouse	43.418	remote	11.330	keyboard	30.838	cellphone	17.338
microwave	30.662	oven	17.777	toaster	11.714	sink	20.593
refrigerator	31.455	book	2.484	clock	37.955	vase	16.103
scissors	1.702	teddybear	21.746	hairdrier	0.124	toothbrush	1.280

Table A.27: Model: 6-to-3 Mapping

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	35.910	bicycle	10.834	car	28.312	motorcycle	20.145
airplane	35.978	bus	50.827	train	41.977	truck	17.744
boat	10.684	trafficlight	15.517	firehydrant	47.435	stopsign	53.979
parkingmeter	29.954	bench	8.208	bird	17.522	cat	47.205
dog	34.285	horse	26.513	sheep	23.033	cow	26.113
elephant	39.020	bear	49.040	zebra	43.042	giraffe	38.633
backpack	5.371	umbrella	23.349	handbag	4.777	tie	20.029
suitcase	10.401	frisbee	43.411	skis	0.465	snowboard	7.847
sportsball	27.999	kite	16.588	baseballbat	11.988	baseballglove	20.249
skateboard	13.995	surfboard	15.988	tennisracket	36.231	bottle	20.930
wineglass	20.356	cup	25.694	fork	4.588	knife	3.347
spoon	1.100	bowl	24.074	banana	5.809	apple	4.401
sandwich	17.481	orange	15.504	broccoli	10.650	carrot	5.663
hotdog	10.640	pizza	35.185	donut	19.348	cake	14.309
chair	7.770	couch	17.447	pottedplant	7.859	bed	17.158
diningtable	7.204	toilet	42.335	tv	39.295	laptop	43.962
mouse	42.340	remote	13.767	keyboard	31.911	cellphone	20.592
microwave	33.905	oven	16.173	toaster	8.704	sink	17.882
refrigerator	32.065	book	3.061	clock	36.475	vase	13.985
scissors	3.961	teddybear	22.948	hairdrier	0.043	toothbrush	4.381

A.7 Instance Segmentation Test Results on SUN RGBD: SOLOv2

Table A.28: Model: RGB Image Only

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
wall	23.641	floor	35.346	cabinet	16.331	bed	23.248
chair	20.937	sofa	21.601	table	17.009	door	19.373
window	13.232	bookshelf	15.847	picture	27.775	counter	14.906
blinds	23.421	desk	6.734	shelves	0.359	curtain	24.884
dresser	25.503	pillow	20.472	mirror	10.963	floormat	5.896
clothes	3.049	ceiling	25.425	books	4.146	fridge	17.114
tv	21.090	paper	4.906	towel	9.082	showercurtain	14.874
box	5.385	whiteboard	26.299	person	8.497	nightstand	26.359
toilet	29.946	sink	22.580	lamp	17.798	bathtub	16.759
bag	4.295						

Table A.29: Model: 4-channel Input

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
wall	22.880	floor	35.697	cabinet	13.221	bed	23.973
chair	18.470	sofa	19.436	table	15.834	door	13.210
window	8.936	bookshelf	12.444	picture	24.573	counter	13.190
blinds	16.088	desk	6.427	shelves	0.444	curtain	17.958
dresser	18.659	pillow	19.341	mirror	10.786	floormat	6.457
clothes	2.275	ceiling	20.565	books	3.314	fridge	14.231
tv	19.215	paper	3.823	towel	6.732	showercurtain	11.488
box	3.421	whiteboard	22.285	person	3.046	nightstand	22.571
toilet	28.071	sink	19.574	lamp	13.079	bathtub	16.818
bag	2.724						

Table A.30: Model: 4-to-3 Mapping

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
wall	23.249	floor	35.653	cabinet	15.317	bed	25.673
chair	19.963	sofa	20.774	table	16.918	door	15.616
window	8.851	bookshelf	14.936	picture	25.483	counter	15.590
blinds	20.456	desk	6.786	shelves	0.420	curtain	22.174
dresser	20.554	pillow	22.729	mirror	13.242	floormat	5.339
clothes	3.125	ceiling	20.572	books	3.702	fridge	17.554
tv	22.340	paper	5.106	towel	9.435	showercurtain	23.474
box	5.052	whiteboard	24.910	person	4.221	nightstand	23.288
toilet	31.335	sink	23.254	lamp	16.878	bathtub	21.469
bag	3.490						

A.8 Instance Segmentation Test Results on MS-COCO Subset: SOLOv2

Table A.31: Model: RGB Depth Only

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	14.710	bicycle	1.462	car	3.644	motorcycle	6.629
airplane	16.267	bus	26.674	train	25.959	truck	5.438
boat	0.720	trafficlight	1.203	firehydrant	26.687	stopsign	26.319
parkingmeter	10.252	bench	2.693	bird	3.343	cat	17.229
dog	5.919	horse	5.268	sheep	2.048	cow	5.474
elephant	13.975	bear	16.860	zebra	11.288	giraffe	15.299
backpack	0.022	umbrella	7.852	handbag	0.127	tie	0.534
suitcase	1.065	frisbee	2.482	skis	0.000	snowboard	0.454
sportsball	2.464	kite	1.605	baseballbat	2.169	baseballglove	1.663
skateboard	0.631	surfboard	1.034	tennisracket	8.142	bottle	2.988
wineglass	1.910	cup	5.998	fork	0.014	knife	0.006
spoon	0.183	bowl	4.063	banana	0.949	apple	0.978
sandwich	3.388	orange	3.828	broccoli	0.353	carrot	0.042
hotdog	1.622	pizza	9.958	donut	4.940	cake	2.568
chair	1.729	couch	10.261	pottedplant	0.265	bed	13.208
diningtable	4.452	toilet	25.767	tv	11.975	laptop	14.345
mouse	3.057	remote	0.085	keyboard	2.348	cellphone	2.152
microwave	1.523	oven	2.259	toaster	1.069	sink	3.939
refrigerator	8.750	book	0.121	clock	1.053	vase	2.921
scissors	0.324	teddybear	7.791	hairdrier	0.396	toothbrush	0.000

Table A.32: Model: RGB Image Only

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	38.589	bicycle	12.708	car	29.487	motorcycle	23.067
airplane	39.582	bus	52.147	train	48.890	truck	20.246
boat	12.529	trafficlight	16.209	firehydrant	48.951	stopsign	49.359
parkingmeter	21.162	bench	7.853	bird	17.445	cat	53.056
dog	39.124	horse	29.536	sheep	29.220	cow	29.667
elephant	49.196	bear	50.717	zebra	52.816	giraffe	50.646
backpack	6.600	umbrella	27.628	handbag	4.407	tie	17.065
suitcase	14.191	frisbee	45.892	skis	2.624	snowboard	8.077
sportsball	30.559	kite	22.261	baseballbat	13.455	baseballglove	21.190
skateboard	19.756	surfboard	14.200	tennisracket	39.535	bottle	20.719
wineglass	19.309	cup	25.134	fork	6.823	knife	3.827
spoon	2.612	bowl	21.824	banana	9.413	apple	3.821
sandwich	18.444	orange	20.155	broccoli	13.135	carrot	10.697
hotdog	12.241	pizza	37.014	donut	20.871	cake	13.259
chair	9.611	couch	22.563	pottedplant	10.999	bed	22.922
diningtable	8.362	toilet	48.438	tv	43.350	laptop	44.553
mouse	43.037	remote	12.311	keyboard	31.521	cellphone	20.718
microwave	38.446	oven	17.354	toaster	17.371	sink	21.475
refrigerator	31.960	book	1.643	clock	38.421	vase	19.000
scissors	4.413	teddybear	30.187	hairdrier	0.057	toothbrush	3.363

Table A.33: Model: 4-channel Input

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	33.988	bicycle	10.186	car	24.466	motorcycle	19.101
airplane	37.383	bus	46.595	train	45.233	truck	18.325
boat	9.953	trafficlight	13.251	firehydrant	44.581	stopsign	52.197
parkingmeter	22.556	bench	6.260	bird	16.012	cat	50.365
dog	39.032	horse	27.320	sheep	24.123	cow	26.860
elephant	43.926	bear	52.978	zebra	47.180	giraffe	45.333
backpack	5.364	umbrella	24.684	handbag	3.121	tie	12.943
suitcase	10.692	frisbee	44.522	skis	1.171	snowboard	5.847
sportsball	28.372	kite	18.094	baseballbat	11.674	baseballglove	22.036
skateboard	17.243	surfboard	12.145	tennisracket	38.233	bottle	16.932
wineglass	15.983	cup	21.455	fork	3.644	knife	2.725
spoon	2.400	bowl	17.349	banana	7.222	apple	4.921
sandwich	22.809	orange	16.656	broccoli	11.163	carrot	6.582
hotdog	9.201	pizza	33.645	donut	19.044	cake	13.783
chair	7.318	couch	19.073	pottedplant	8.080	bed	20.675
diningtable	8.490	toilet	45.611	tv	40.930	laptop	39.331
mouse	39.980	remote	10.790	keyboard	31.244	cellphone	16.888
microwave	31.653	oven	14.729	toaster	8.294	sink	20.271
refrigerator	26.550	book	1.324	clock	34.567	vase	15.326
scissors	5.070	teddybear	24.732	hairdrier	0.781	toothbrush	5.736

Table A.34: Model: 6-channel Input

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	34.743	bicycle	9.965	car	24.763	motorcycle	20.375
airplane	36.808	bus	47.504	train	46.167	truck	18.765
boat	10.437	trafficlight	14.522	firehydrant	45.869	stopsign	52.559
parkingmeter	22.396	bench	5.522	bird	16.497	cat	49.754
dog	39.521	horse	27.51	sheep	23.521	cow	27.201
elephant	43.378	bear	54.37	zebra	47.116	giraffe	45.376
backpack	6.055	umbrella	26.147	handbag	3.33	tie	13.362
suitcase	10.913	frisbee	45.007	skis	2.304	snowboard	6.503
sportsball	29.713	kite	18.403	baseballbat	12.665	baseballglove	23.131
skateboard	16.916	surfboard	12.014	tennisracket	38.341	bottle	16.452
wineglass	16.059	cup	22.889	fork	4.511	knife	3.414
spoon	3.759	bowl	17.048	banana	8.651	apple	4.228
sandwich	23.394	orange	16.801	broccoli	10.659	carrot	7.865
hotdog	10.641	pizza	33.022	donut	19.588	cake	13.137
chair	6.835	couch	19.818	pottedplant	7.581	bed	20.661
diningtable	7.912	toilet	46.192	tv	41.343	laptop	39.43
mouse	40.76	remote	10.755	keyboard	32.021	cellphone	18.227
microwave	31.671	oven	14.808	toaster	9.221	sink	21.503
refrigerator	26.967	book	2.309	clock	35.442	vase	15.057
scissors	5.941	teddybear	25.704	hairdrier	0.508	toothbrush	5.785

Table A.35: Model: 4-to-3 Mapping

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	36.134	bicycle	12.339	car	27.129	motorcycle	21.153
airplane	39.509	bus	51.236	train	49.906	truck	20.046
boat	9.858	trafficlight	13.164	firehydrant	50.614	stopsign	51.051
parkingmeter	21.298	bench	8.089	bird	16.372	cat	49.343
dog	35.723	horse	27.613	sheep	28.565	cow	28.073
elephant	45.918	bear	53.265	zebra	49.153	giraffe	45.258
backpack	5.745	umbrella	27.095	handbag	4.223	tie	16.911
suitcase	11.775	frisbee	39.798	skis	1.834	snowboard	7.470
sportsball	28.236	kite	17.592	baseballbat	13.799	baseballglove	16.498
skateboard	17.933	surfboard	12.450	tennisracket	40.142	bottle	19.089
wineglass	17.746	cup	25.035	fork	5.729	knife	3.352
spoon	2.217	bowl	21.434	banana	7.054	apple	4.010
sandwich	19.443	orange	16.741	broccoli	11.209	carrot	4.985
hotdog	7.937	pizza	32.487	donut	20.099	cake	13.969
chair	9.544	couch	22.612	pottedplant	8.408	bed	25.596
diningtable	9.551	toilet	48.405	tv	38.704	laptop	42.707
mouse	48.285	remote	8.114	keyboard	32.000	cellphone	16.634
microwave	28.224	oven	17.870	toaster	9.594	sink	21.965
refrigerator	31.333	book	1.498	clock	35.479	vase	16.158
scissors	2.526	teddybear	26.165	hairdrier	0.000	toothbrush	2.795

Table A.36: Model: 6-to-3 Mapping

Class	AP (%)	Class	AP (%)	Class	AP (%)	Class	AP (%)
person	37.404	bicycle	11.809	car	27.049	motorcycle	21.233
airplane	40.255	bus	51.118	train	50.012	truck	19.871
boat	9.239	trafficlight	13.003	firehydrant	51.619	stopsign	50.442
parkingmeter	21.7	bench	9.292	bird	17.16	cat	48.944
dog	36.887	horse	28.798	sheep	28.207	cow	29.151
elephant	46.462	bear	52.592	zebra	48.495	giraffe	46.103
backpack	5.043	umbrella	26.993	handbag	4.265	tie	16.91
suitcase	11.394	frisbee	40.614	skis	2.394	snowboard	8.005
sportsball	28.307	kite	18.468	baseballbat	14.87	baseballglove	17.377
skateboard	18.69	surfboard	12.027	tennisracket	41.535	bottle	19.867
wineglass	17.527	cup	24.496	fork	5.051	knife	2.864
spoon	1.596	bowl	22.329	banana	6.575	apple	3.703
sandwich	20.515	orange	17.338	broccoli	12.448	carrot	4.834
hotdog	9.353	pizza	33.521	donut	21.268	cake	15.073
chair	10.96	couch	23.974	pottedplant	9.454	bed	25.503
diningtable	9.824	toilet	48.933	tv	38.222	laptop	43.633
mouse	48.032	remote	7.469	keyboard	31.697	cellphone	17.621
microwave	27.53	oven	18.69	toaster	10.742	sink	23.196
refrigerator	31.756	book	2.411	clock	36.071	vase	16.049
scissors	2.124	teddybear	25.469	hairdrier	-0.52	toothbrush	2.191

REFERENCES

[1] *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), with CD-ROM, 8-14 December 2001, Kauai, HI, USA*. IEEE Computer Society, 2001.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[3] M. W. Ayech and D. Ziou. Segmentation of terahertz imaging using k-means clustering based on ranked set sampling. *Expert Systems with Applications*, 42(6):2959–2974, 2015.

[4] J. T. Barron, A. Adams, Y. Shih, and C. Hernández. Fast bilateral-space stereo for synthetic defocus. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4466–4474, 2015.

[5] C. Cadena, Y. Latif, and I. D. Reid. Measuring the performance of single image depth estimation methods. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4150–4157, 2016.

[6] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[7] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

[8] J. Choi, D. Chun, H. Kim, and H.-J. Lee. Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 502–511, 2019.

[9] 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*, abs/1604.01685, 2016.

[10] F. Da and H. Zhang. Sub-pixel edge detection based on an improved moment. *Image and Vision Computing*, 28(12):1645–1658, 2010.

[11] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. *CoRR*, abs/1512.04412, 2015.

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

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[14] A. Eitel, J. T. Springenberg, L. Spinello, M. A. Riedmiller, and W. Burgard. Multimodal deep learning for robust RGB-D object recognition. *CoRR*, abs/1507.06821, 2015.

[15] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[16] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[17] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[18] W. Gao, X. Zhang, L. Yang, and H. Liu. An improved sobel edge detection. In *2010 3rd International Conference on Computer Science and Information Technology*, volume 5, pages 67–71, 2010.

[19] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[20] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[21] C. Godard, O. M. Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *CoRR*, abs/1609.03677, 2016.

[22] C. Godard, O. M. Aodha, and G. J. Brostow. Digging into self-supervised monocular depth estimation. *CoRR*, abs/1806.01260, 2018.

[23] B. Hariharan, P. Arbelaez, R. B. Girshick, and J. Malik. Simultaneous detection and segmentation. *CoRR*, abs/1407.1808, 2014.

[24] B. Hariharan, P. A. Arbeláez, R. B. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. *CoRR*, abs/1411.5752, 2014.

[25] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[27] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.

[28] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[29] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, abs/1611.10012, 2016.

[30] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016.

[31] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[32] P. Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[33] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang. T-CNN: tubelets with convolutional neural networks for object detection from videos. *CoRR*, abs/1604.02532, 2016.

[34] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306, 2014.

[35] K. Karsch, K. Sunkavalli, S. Hadap, N. Carr, H. Jin, R. Fonte, and M. Sittig. Automatic scene inference for 3d object compositing. *CoRR*, abs/1912.12297, 2019.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012.

[37] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. *CoRR*, abs/1808.01244, 2018.

[38] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[39] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[40] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Proceedings. International Conference on Image Processing*, volume 1, pages I–I, 2002.

[41] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[42] 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*, abs/1405.0312, 2014.

[43] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. *CoRR*, abs/1803.01534, 2018.

[44] A. Mertan, D. J. Duff, and G. Unal. Single image depth estimation: An overview. *CoRR*, abs/2104.06456, 2021.

[45] S. M. H. Miangoleh, S. Dille, L. Mai, S. Paris, and Y. Aksoy. Boosting monocular depth estimation models to high-resolution via content-adaptive multi-resolution merging. 2021.

[46] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5000–5009, 2017.

[47] J. Noordam, W. van den Broek, and L. Buydens. Geometrically guided fuzzy c-means clustering for multivariate image segmentation. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 1, pages 462–465 vol.1, 2000.

[48] T. Ophoff, K. Van Beeck, and T. Goedemé. Exploring rgb+depth fusion for real-time object detection. *Sensors*, 19(4), 2019.

[49] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[50] R. Padilla, S. L. Netto, and E. A. B. da Silva. A survey on performance metrics for object-detection algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242, 2020.

[51] W. Pitts and W. S. McCulloch. How we know universals; the perception of auditory and visual forms. *The Bulletin of mathematical biophysics*, 93:127–47, 1947.

[52] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[53] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[54] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[55] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[56] A. Rosenfeld. The max roberts operator is a hueckel-type edge detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(1):101–103, 1981.

[57] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[58] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[59] V. Sharma and R. N. Mir. A comprehensive and systematic look up into deep learning based object detection techniques: A review. *Comput. Sci. Rev.*, 38:100301, 2020.

[60] Y. Shi, Z. Chen, Z. Qi, F. Meng, and L. Cui. A novel clustering-based image segmentation via density peaks algorithm with mid-level feature. *Neural Computing and Applications*, 28:29–39, 2016.

[61] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Commun. ACM*, 56(1):116–124, jan 2013.

[62] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[63] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, 2015.

[64] D. Stoyanov, M. V. Scarzanella, P. Pratt, and G.-Z. Yang. Real-time stereo reconstruction in robotically assisted minimally invasive surgery. *Medical image*

computing and computer-assisted intervention : MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention, 13 Pt 1:275–82, 2010.

[65] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[66] D. Tian, Y. Han, B. Wang, T. Guan, H. Gu, and W. Wei. Review of object instance segmentation based on deep learning. *Journal of Electronic Imaging*, 31(4):1 – 18, 2021.

[67] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. *2009 IEEE 12th International Conference on Computer Vision*, pages 606–613, 2009.

[68] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li. SOLO: segmenting objects by locations. *CoRR*, abs/1912.04488, 2019.

[69] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen. Solov2: Dynamic, faster and stronger. *CoRR*, abs/2003.10152, 2020.

[70] Z. Wang and M. Yang. A fast clustering algorithm in image segmentation. In *2010 2nd International Conference on Computer Engineering and Technology*, volume 6, pages V6–592–V6–594, 2010.

[71] I. R. Ward, H. Laga, and M. Bennamoun. RGB-D image-based object detection: from traditional methods to deep learning techniques. *CoRR*, abs/1907.09236, 2019.

[72] Q. Wu, C. Shen, A. van den Hengel, P. Wang, and A. R. Dick. Image captioning and visual question answering based on attributes and their related external knowledge. *CoRR*, abs/1603.02814, 2016.

[73] X. Wu, D. Sahoo, and S. C. H. Hoi. Recent advances in deep learning for object detection. *CoRR*, abs/1908.03673, 2019.

[74] J. Xie, R. B. Girshick, and A. Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. *CoRR*, abs/1604.03650, 2016.

[75] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.

[76] L. Yang, X. Wu, D. Zhao, H. Li, and J. Zhai. An improved prewitt algorithm for edge detection based on noised image. In *2011 4th International Congress on Image and Signal Processing*, volume 3, pages 1197–1200, 2011.

[77] H. Ye and S. Yan. Double threshold image segmentation algorithm based on adaptive filtering. In *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 1008–1011, 2017.

[78] J.-C. Yen, F.-J. Chang, and S. Chang. A new criterion for automatic multilevel thresholding. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 4 3:370–8, 1995.

[79] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CoRR*, abs/1707.01083, 2017.

[80] Z.-Q. Zhao, P. Zheng, S. tao Xu, and X. Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30:3212–3232, 2019.

[81] Z. Zou, Z. Shi, Y. Guo, and J. Ye. Object detection in 20 years: A survey. *CoRR*, abs/1905.05055, 2019.