

Depth-Aware Deep Learning Networks for Object Detection and Image Segmentation

James Dickens

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the
degree of Master of Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

The rise of convolutional neural networks (CNNs) in the context of computer vision has occurred in tandem with the advancement of depth sensing technology. Depth cameras are capable of yielding two-dimensional arrays storing at each pixel the distance from objects and surfaces in a scene from a given sensor, aligned with a regular color image, obtaining so-called RGBD images. Inspired by prior models in the literature, this work develops a suite of RGBD CNN models to tackle the challenging tasks of object detection, instance segmentation, and semantic segmentation. Prominent architectures for object detection and image segmentation are modified to incorporate dual backbone approaches inputting RGB and depth images, combining features from both modalities through the use of novel fusion modules. For each task, the models developed are competitive with state-of-the-art RGBD architectures. In particular, the proposed RGBD object detection approach achieves 53.5% mAP_{50} on the SUN RGBD 19-class object detection benchmark, while the proposed RGBD semantic segmentation architecture yields 69.4% accuracy with respect to the SUN RGBD 37-class semantic segmentation benchmark. An original 13-class RGBD instance segmentation benchmark is introduced for the SUN RGBD dataset, for which the proposed model achieves 38.4% mAP_{50} . Additionally, an original depth-aware panoptic segmentation model is developed, trained, and tested for new benchmarks conceived for the NYUDv2 and SUN RGBD datasets. These benchmarks offer researchers a baseline for the task of RGBD panoptic segmentation on these datasets, where the novel depth-aware model outperforms a comparable RGB counterpart.

Contents

List of Tables	v
List of Figures	vi
1 Introduction	1
1.1 Motivation and Research Inquiry	1
1.2 Research Objectives	5
1.3 Thesis Structure	6
2 Technical Background	8
2.1 Depth Images	8
2.1.1 Repeated Grayscale Encoding	9
2.1.2 Disparity Images	9
2.1.3 HHA Images	10
2.1.4 The Benefits of Depth Images	15
2.2 Computer Vision Tasks And Performance Metrics	18
2.2.1 Object Detection	18
2.2.2 Instance Segmentation	22
2.2.3 Semantic Segmentation	24
2.2.4 Panoptic Segmentation	25
2.3 Residual Networks as a CNN Backbone	28
2.4 Object Detection with Faster R-CNN	30
2.5 Feature Pyramid Networks	33
2.6 Instance Segmentation with Mask R-CNN	36
2.7 Semantic Segmentation with DeepLabv3	39
3 Literature Review	43
3.1 Deep Learning and RGBD Object Detection	43
3.2 Deep Learning and RGBD Semantic Segmentation	52
4 Methodology	65
4.1 Fusion Modules	65
4.2 Proposed Architectures	75
4.2.1 Object Detection	75
4.2.2 Instance Segmentation	78
4.2.3 Semantic Segmentation	80
4.2.4 Panoptic Segmentation	81
5 Experimental Validation	86
5.1 Datasets	86
5.1.1 NYUDv2	87
5.1.2 SUN RGBD	89
5.2 Experiments	91

5.2.1	Object Detection	93
5.2.1.1	Implementation Details	93
5.2.1.2	Results and Analysis	97
5.2.2	Instance Segmentation	103
5.2.2.1	Implementation Details	103
5.2.2.2	Results and Analysis	106
5.2.3	Semantic Segmentation	110
5.2.3.1	Implementation Details	110
5.2.3.2	Results and Analysis	111
5.2.4	Panoptic Segmentation	119
5.2.4.1	Implementation Details	119
5.2.4.2	Results and Analysis	121
5.2.5	Overall Analysis	127
6	Conclusion	128
6.1	Summary and Contributions	128
6.2	Future Research	129
	References	132
	Appendices	140
A	Basic CNN Components	140
A.1	Convolutions	140
A.2	Pooling Layers	143
A.3	Activation Functions	144
A.4	Batch Normalization	145
A.5	Fully Connected Layers	146
B	Category Mapping	147

List of Tables

Table 2.1: Layer details of the ResNet-50 and ResNet-101 models	30
Table 4.1: The number of parameters per fusion module	75
Table 4.2: Resolution sizes per output layer of the DeepLabv3 and standard ResNet-101 backbones	83
Table 5.1: The per-class AP_{50} results on the NYUDv2 19-class object detection benchmark for the baseline and depth-aware Faster R-CNN FPN models trained	97
Table 5.2: The per-class AP_{50} results on the SUN RGBD 19-class object detection bench- mark for the Faster R-CNN FPN models trained	99
Table 5.3: Average recall results for object detection and region proposal generation on the NYUDv2 test dataset for the Faster R-CNN FPN models trained	101
Table 5.4: Average recall results for object detection and region proposal generation on the SUN RGBD test dataset for the Faster R-CNN FPN models trained	101
Table 5.5: Comparing the Faster R-CNN FPN f_{LC} model with state-of-the-art models on the NYUDv2 19-class object detection benchmark	102
Table 5.6: Comparing the Faster R-CNN FPN f_{LC} model with state-of-the-art architec- tures on the SUN RGBD 19-class object detection benchmark	103
Table 5.7: Comparing the inference speed of the Faster R-CNN FPN models trained . . .	103
Table 5.8: The per-class AP_{50} results on the NYUDv2 17-class instance segmentation benchmark for the Mask R-CNN FPN models trained	106
Table 5.9: Comparing the performance of the proposed Mask R-CNN FPN f_{LC} with other works in the literature	108
Table 5.10: The per-class AP_{50} results of the Mask R-CNN FPN models tested on the SUN RGBD 13-class instance segmentation benchmark	109
Table 5.11: Comparing the inference speed of the Mask R-CNN FPN models trained . .	110
Table 5.12: The per-category mIoU of the DeepLabv3 models on the NYUDv2 40-class semantic segmentation benchmark	112
Table 5.13: Performance summary for the DeepLabv3 models on the NYUDv2 40-class semantic segmentation benchmark.	112
Table 5.14: Comparing the DeepLabv3 f_{LC} model performance to state-of-the-art works on the NYUDv2 40-class semantic segmentation benchmark.	115
Table 5.15: Per-category IoU on the SUN RGBD 37-class semantic segmentation bench- mark for the DeepLabv3 models trained	116
Table 5.16: Performance summary for the DeepLabv3 models on the SUN RGBD 37-class semantic segmentation benchmark	116
Table 5.17: Comparing the DeepLabv3 f_{Res} model to other models in the literature on the SUN RGBD 37-class semantic segmentation benchmark	118
Table 5.18: Comparing inference speed of the DeepLabv3 models trained	118
Table 5.19: Per-category panoptic quality on the NYUDv2 30-class benchmark	122
Table 5.20: Per-category panoptic quality on the SUN RGBD 38-class benchmark	125
Table 5.21: Comparing the inference speed of the panoptic segmentation models trained	126

List of Figures

Figure 1.1: An example contour image from the SUN RGBD dataset	3
Figure 2.1: An RGBD image	9
Figure 2.2: A disparity image	10
Figure 2.3: A height above the ground image	13
Figure 2.4: An angle with gravity image	14
Figure 2.5: An example HHA image	14
Figure 2.6: Object boundaries in an RGB image vs. a depth image	15
Figure 2.7: The texture details are abstracted in a depth image	16
Figure 2.8: Scale distortions caused by perspective projection	17
Figure 2.9: The output from an object detection model	19
Figure 2.10: The IoU of two bounding boxes.	19
Figure 2.11: Example output of an instance segmentation model	23
Figure 2.12: A comparison between segmentation types	27
Figure 2.13: An example bottleneck operation without downsampling	30
Figure 2.14: Anchor box centers and example anchor boxes from Faster R-CNN	32
Figure 2.15: A feature pyramid network based on a ResNet backbone	35
Figure 2.16: The mask branch of Mask R-CNN	39
Figure 2.17: The Atrous Spatial Pooling Pyramid of the DeepLabv3 Head	41
Figure 3.1: A simplified overview of the approach of RGBD R-CNN	43
Figure 3.2: The supervision transfer approach of Gupta et al.	47
Figure 3.3: The three backbone RGBD Faster R-CNN approach of Correlated Detection Net	48
Figure 3.4: An overview of the real-time RGBD object detection method of Ophoff et al.	50
Figure 3.5: An overview of the architecture of Eigen and Fergus	54
Figure 3.6: The variants of the FCN architecture	55
Figure 3.7: Unpooling layers	56
Figure 3.8: The LSTM-CF model	57
Figure 3.9: The RefineNet sub-components	58
Figure 3.10: The MMF module and an overview of RDFNet	59
Figure 4.1: Observing noise patterns in a depth image	66
Figure 4.2: Comparing RGB and HHA images	68
Figure 4.3: A very noisy RGB/depth image pair from the SUN RGBD dataset	69
Figure 4.4: The learned concatenation approach	71
Figure 4.5: The f_{Res} operation, the ResNet bottleneck layer, and the MMF module from RDFNet	72
Figure 4.6: The f_{Gated} fusion module, versions 1 and 2.	74
Figure 4.7: A dual backbone multi-modal feature pyramid network	77
Figure 4.8: The proposed depth-aware Faster R-CNN FPN	77
Figure 4.9: The proposed depth-aware Mask R-CNN FPN	79
Figure 4.10: The proposed depth-aware DeepLabv3 architecture	81

Figure 4.11: The modified RGB FPN to be used in the proposed panoptic segmentation architecture	83
Figure 4.12: The modified depth-aware FPN	84
Figure 4.13: An overview of the proposed depth-aware panoptic segmentation architecture	85
Figure 5.1: A noisy depth image and a denoised depth image from the SUN RGBD dataset	90
Figure 5.2: Faster R-CNN FPN f_{LC} object detection output on the NYUDv2 dataset . .	99
Figure 5.3: Faster R-CNN FPN f_{LC} detections on the SUN RGBD dataset	100
Figure 5.4: False positives from the Faster R-CNN FPN f_{LC} model	101
Figure 5.5: Predicted instance masks from the Mask R-CNN FPN f_{LC} model on the NYUDv2 dataset	107
Figure 5.6: A failure case from the Mask R-CNN FPN f_{LC} model on the NYUDv2 dataset	108
Figure 5.7: A comparison of the RGBD DeepLabv3 output to the RGB baseline.	113
Figure 5.8: Failure cases from the DeepLabv3 f_{LC} model on the NYUDv2 dataset	114
Figure 5.9: Qualitative comparison of the model output from the RGB DeepLabv3 baseline to the DeepLabv3 f_{Res} model	117
Figure 5.10: An overlap issue arising from a low quality instance mask	123
Figure 5.11: Depth-Aware Panoptic model output on RGB images from the NYUDv2 dataset	124
Figure 5.12: Predicted masks from the Depth-Aware Panoptic model on the SUN RGBD dataset	126
Figure A.1: A dilated 3×3 filter with dilation tuple $(2, 2)$	142

Chapter 1: Introduction

1.1 Motivation and Research Inquiry

In the late 1960's Seymour Papert, a professor at the famed Massachusetts Institute of Technology (MIT), coordinated a summer project in which students at MIT aimed to develop a system of programs to divide images into regions. The regions were so constructed as to either contain an object, or contain the image background¹. By analyzing shape and surface properties of the objects and structures within a set of images, the ultimate goal was to identify objects by name, matching them with a vocabulary of known objects [1]. Computer science historians often credit this project as one of the initial endeavors of artificial intelligence (AI) within the context of computer vision; what began as an investigative summer project conducted by an ambitious researcher would evolve to become one of the most studied sub-fields of modern AI research. The aim of computer vision is to develop software that can analyze images and videos for many practical uses. These tasks include, among others, object detection and tracking, estimating the 3D pose of humans, and image segmentation.

Industrial applications of computer vision are continuously expanding, largely fueled by the rise of *machine learning* in which researchers use large-scale datasets to train powerful algorithms for each of these tasks, achieving increasingly impressive results. Departing from fixed statistical approaches drawing from the theory of image processing, computer vision researchers have fully embraced *deep learning*, a subset of machine learning inspired by the human brain which makes use of neural networks. Within the machine learning paradigm of *supervised learning*, algorithms are typically trained on a training set of expert-annotated *training examples*, and then tested on a test set wherein the performance of various approaches can be compared using statistical metrics. Computer vision applications that make use of deep learning include robotic navigation, pedestrian detection, facial recognition, and medical imaging.

In this work, the role of a second modality, namely depth imagery, will be explored in order to complement the information provided by a standard color image. A *depth camera*, or *depth sensor* is a piece of hardware that can produce a depth image: a 2D array of pixels where each pixel's numerical value corresponds

¹The third category considered was that of chaos, implying an unclear semantic value.

to the distance from points in a scene to a sensor’s optical center, captured by a sensor array. Aligning the depth frame with a color image, one obtains an RGBD (red-green-blue-depth) image. Following the release of Microsoft’s Kinect v1 in 2010, aided by the subsequent release of the commercial and non-commercial SDK (standard development kit) in 2012 [2], researchers in computer vision and other disciplines were able to make use of depth imagery for the low cost of 150 USD. Previously, the ability to employ depth imaging involved costly stereo camera setups or laser scanners. The original purpose of the Kinect sensor was to use gesture analysis, facial recognition, and skeletal tracking for a gaming experience without controllers or keyboards, possibly with multiple people playing at the same time, used with the Xbox 360 gaming console.

As an unintended consequence, the release of the Kinect v1 camera facilitated new research in disciplines that were adjacent, but not strictly related to the original tasks of gesture recognition and skeletal tracking for gaming applications. Some of the research disciplines taking advantage of the Kinect and the Kinect SDK release were:

- Robotics [3]
- Gait analysis [4]
- Pedestrian detection [5]
- Computer vision research. In particular for the tasks of image classification, object detection and image segmentation, RGBD datasets were created, such as the NYUDv2 dataset [6]

Since the release of the Kinect depth sensor (versions 1 and 2), the market for affordable consumer-grade depth sensors has continued to grow. Different depth sensors use varying techniques in order to obtain depth measurements aligned with an RGB image of the scene, most notably the use of structured light or time-of-flight technologies. The type of depth images used in this thesis will be those obtained from sensors targeted at autonomous robots, to be used in indoor settings with limited range. Range imaging derived from LIDAR (light detection and ranging), which is typically used in outdoor settings, will not be considered.

The focus of modern computer vision research has largely been dominated by RGB images. Despite prominent works in the literature which demonstrate the benefits of the additional information provided by the depth channel, such as the

work of Gupta et al. [7], RGBD-based algorithms have occupied a niche role in the overall research landscape. Hence, the focus of this thesis will be to use modern deep learning techniques to further explore the role of depth in two popular and well researched computer vision tasks within the RGB paradigm. The first such task will be that of categorical object detection, where the aim is to train a deep learning algorithm capable of identifying the location and type of objects within an image, with the type of objects considered belonging to a fixed set of classes.

The second task considered is that of image segmentation. Historically, image segmentation research began with contour detection, where the pixels of a color or grayscale image are partitioned into coherent regions, bound by closed curves and the boundaries of the image. In other words, each pixel is assigned to be either on the boundary of a segment, or the interior of a segment, see Figure 1.1.

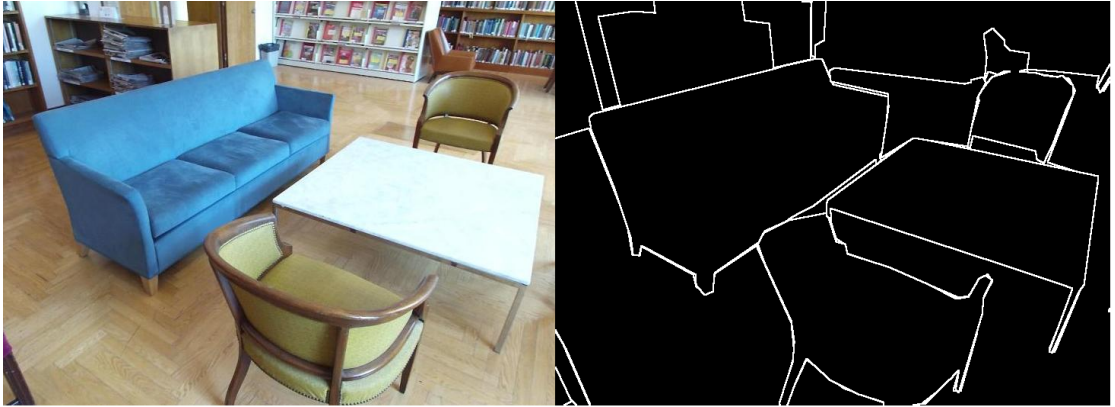


Figure 1.1: Left: An RGB image from the SUN RGBD Dataset [8], Right: A ground truth contour image.

In modern works, researchers have become more ambitious, enhancing the predictive value of the image partitions. In particular, three sub-domains have emerged, each with a rich and competitive body of literature. The first is that of *semantic segmentation*, in which each region is assigned a class label from a fixed set of classes. This approach naturally extends the contour detection goal by asking: what is the meaning of each partition? The second type is *instance segmentation*, which expands upon object detection by segmenting individual object instances, rather than simply localizing them within a rectangular region in the image. The third segmentation type which offers researchers the most challenging test to date, is that of *panoptic segmentation*: the combination of an overall semantic segmentation mask with the segmentation of individual instances.

Indeed, image segmentation of various sorts is a key ingredient to robotics

and autonomous vehicle research. The ultimate goal is to achieve real-time image segmentation to enhance an intelligent agent’s scene understanding capabilities. By deducing the meaning of each pixel in an image of an agent’s field of view, and being able to distinguish between separate entities of the same type, a greater understanding of an entity’s surroundings can be utilized for intelligent decision making.

For each task described, machine learning algorithms typically make use of learned feature extraction, followed by task-specific modules for further processing. The main deep learning tool employed in this work is that of convolutional neural networks (CNNs), the primary apparatus for feature extraction for object detection and image segmentation in modern computer vision research. CNNs, whose recent popularity largely emerged from the impressive success of AlexNet [9] on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [10], are a set of algorithms that make use of convolutional filters to extract discriminative features from images. These features are multifaceted and span image processing cornerstones including edges, corners, and textural information, as well as higher level semantic information such as object structure. Convolutional filters are incorporated into neural network *layers*, for which an overall grouping of layers for feature extraction is referred to as a *convolutional backbone*, or simply a backbone, (see Appendix A for a review of basic CNN components). There exist many different CNN architectures, and their development is rapidly evolving. CNN architectures that make use of RGBD images will be referred to as *depth-aware*, a term borrowed from Wang and Neumann [11].

Exploring depth-aware object detection and segmentation algorithms, a natural set of broad questions emerges in relation to the use of the additional depth channel within the context of existing RGB-based deep learning algorithms:

- What benefit does a depth image offer to complement the information provided by a regular color image?
- Are there features for object detection or image segmentation that are more discriminative for one modality than the other?
- Can a feature detection algorithm using information combined from both modalities improve performance when compared to a single modality?
- Can depth images be used alone with any measure of success?

- Can the use of both modalities actually *degrade performance* in some cases?
- Can CNNs trained for regular color images be used with any measure of success for depth images as well?

1.2 Research Objectives

The primary goal of this work is to create a suite of depth-aware CNN-based architectures for object detection and image segmentation, ultimately culminating in a panoptic segmentation model, which is novel in two senses. First, the structure of the architecture is novel to this thesis. Secondly, it is the first, or one of the first, evaluations of RGBD-based CNNs for panoptic segmentation in the literature. The results of each model are evaluated per task on the challenging NYUDv2 and SUN RGBD datasets [6, 8]. Comparisons with RGB baseline models of the same architecture as well as prominent works and even state-of-the-art models are made for detailed analysis using standard performance metrics.

Deeply inspired by the work of Gupta et al. [7], discussed in Section 3.1, a hierarchy of sub-goals and tasks is proposed in this thesis that will lead to a final panoptic segmentation architecture consisting of object detection/instance segmentation and semantic segmentation models as building blocks. In particular,

- The Faster R-CNN Feature Pyramid Network (FPN) [12, 13], object detection model is modified to incorporate a dual backbone approach, one backbone input with the RGB modality, and the other input with an HHA-encoded depth image, where the HHA encoding is introduced in Section 2.1.3. As will be discussed concerning the RGBD object detection literature, a neglected approach in RGBD dual backbone approaches for object detection is a learned interaction between the hierarchy of feature maps produced by a CNN. FPNs offer a natural architecture to complement this interaction, and the fusion between feature maps will be tested with three approaches: a learned concatenation operation, a residual approach inspired by the ResNet bottleneck layer [14], and a gated approach inspired by the method proposed by Oyedotun et al. [16]. These will be referred to as *fusion modules*, which have learnable parameters to be trained. Further, the role of the depth modality in the performance of the region proposal network inherent to Faster R-CNN will be evaluated using a single region proposal network, to be contrasted with previous approaches in the literature.

- The resulting depth-aware Faster R-CNN FPN architecture will be extended to an instance segmentation model by adding the mask branch detailed in Mask R-CNN [17]. The same fusion modules are re-tested within the instance segmentation context.
- The DeepLabv3 [18] semantic segmentation architecture will be modified to incorporate a dual backbone approach, experimenting with a late-fusion approach using the same fusion modules.
- The final step is to build and test a dual backbone panoptic architecture by adjusting the DeepLabv3 backbone to be compatible with the Mask R-CNN FPN model. The network will then have network heads for detection, instance segmentation, and semantic segmentation. The predicted instance masks are combined with the predicted semantic segmentation mask using heuristic post-processing to yield a final panoptic mask. The fusion modules employed for the panoptic network will be chosen based on the best performing learned interaction of its sub-components.

Additionally, adjacent research concerns are addressed, namely:

- The benefits of the HHA encoding, a specialized 3-channel image obtained from a depth image introduced by Gupta et al. [7], versus the use of the raw depth images as input to a CNN for feature extraction.
- The performance of the depth modality in each model without the added RGB modality for comparison with the pure RGB counterpart. By evaluating each modality separately, it is possible to make reasonable hypotheses about what the possible advantages of fusing features obtained from each sensory system might be to model performance.

1.3 Thesis Structure

The outline of this thesis is given as follows:

- In Chapter 2, the technical background concerning depth images is presented. Formal definitions for the detection and segmentation tasks are provided and commonly used performance metrics detailed. Technical summaries of Faster R-CNN, Feature Pyramid Networks, Mask R-CNN, and DeepLabv3 architectures are given, with motivations provided for their use.
- In Chapter 3, literature reviews for RGBD object detection and RGBD

semantic segmentation are presented.

- In Chapter 4, the fusion modules are introduced, followed by a description of the proposed architectures used for each of the tasks considered.
- In Chapter 5, an experimental evaluation is conducted, with implementation and training details given, alongside an analysis of the experiment results.
- In Chapter 6, a summary of this work is presented describing its major contributions, followed by a discussion of potential avenues for future research.

Chapter 2: Technical Background

2.1 Depth Images

A central theme in this work is the use of multi-modal computer vision techniques to potentially bolster the performance of algorithms designed for detection and segmentation, when compared to their single modality counterparts. The first modality employed is the use of digital *RGB (red, green, and blue) images* which are stored in memory as a tensor² with 3 axes. The first axis indexes the red, green and blue component of the tensor, and the next two axes index the spatial locations of the tensor, which are the row and column of a pixel's location in an image. For an image of resolution $H \times W$ (height by width) pixels, spatial indices proceed from top to bottom and from left to right; the top left corner of the image is at $(0, 0)$, and the bottom right corner of the image is located at $(H - 1, W - 1)$. Each entry in the RGB tensor is a non-negative integer between 0 and 255, inclusively, for an 8-bit color image. One can think of an image as a discrete sample of the 3 dimensional world, where the reflected light of each object and surface in a scene contributes to the perspective projection of a 3D view to a finite subset of a plane, referred to as the image plane. Each pixel is a sample of the incoming light's frequency distribution at a discrete geometric grid location captured by a sensor.

The second modality will be that of *depth images*. While an RGB image of resolution $H \times W$ can be thought of as a discrete function defined on triples of non-negative integers (using 0-indexing), i.e.

$$I : [0, 2] \times [0, H - 1] \times [0, W - 1] \mapsto [0, 255], \quad (2.1)$$

a depth image of the same resolution can be thought of as a mapping

$$D : [0, H - 1] \times [0, W - 1] \mapsto [d_{min}, d_{max}] \quad (2.2)$$

where d_{min} and d_{max} are the minimum and maximum range of distance from the sensor that can be reliably measured, assuming no noise. Missing depth pixels are assigned a value of 0. The types of depth encoding used in this thesis are described in the remainder of Section 2.1.

²A tensor, in the context of deep learning and computer science, is a multi-dimensional array with a set number of axes and a given *shape*. For example, a color image of resolution 600×600 would have shape $(3, 600, 600)$.

2.1.1 Repeated Grayscale Encoding

In order to view a depth image as a displayable 3-channel image, there are several encoding schemes available. The simplest encoding scheme for a depth image of size $H \times W$ is to first normalize the depth-values, and re-scale them to be integers between $[0, 255]$. This is achieved by the transform:

$$f(d) = \lceil 255 * \frac{(d - d_{min})}{(d_{max} - d_{min})} \rceil \quad (2.3)$$

Afterwards, the single channel values are simply replicated in each of the *RGB* channels of a 3-channel color image. Dark values represent regions in space close to the sensor, and lighter values denote spatial locations further away from the sensor, as in Figure 2.1.



Figure 2.1: An RGBD image obtained from the SUN RGBD dataset [8] with the Kinect v2 depth camera.

2.1.2 Disparity Images

In the context of a stereo camera setup used to compute depth, it is well known that the depth value of a point in 3D space from the horizontal midpoint between both cameras is inversely proportional to the horizontal binocular disparity. With this principle in mind, the computation of disparity images from depth images as in the publicly available code of Gupta et al. [7] will be described.

Suppose we have a depth image $D(x, y)$. First the image's depth values are thresholded so that the minimum depth value is 1 meter, yielding $D'(x, y)$. Next the transformation f from a depth value in meters to a disparity unit in the range $[0, 255]$ is computed as $f(x, y) = \frac{255}{D'(x, y)}$, which is then rounded to the nearest integer. The constant 255 is chosen such that the scaled result will have maximal

disparity value of 255 for a resulting 8-bit grayscale image. See Figure 2.2 for an example disparity image, where objects and surfaces closer to the sensor are mapped to a brighter value, indicating a larger horizontal disparity.



Figure 2.2: An example disparity image from the SUN RGBD Dataset [8]. Left: RGB Image, Middle: Depth Image, Right: Disparity Image.

2.1.3 HHA Images

Gupta et al. [7] propose an encoding of depth images into a 3-channel image, where each channel captures unique information. The resulting image is referred to as an HHA image, consisting of channels:

- Horizontal Disparity, described in Section 2.1.2
- Height Above the Ground
- Angle with a Gravity Vector

In many RGBD object detection and image segmentation works in the literature [11, 20, 21], HHA images have been used as input to CNNs for feature extraction, as opposed to the raw depth images. It was conjectured by Gupta et al. [7] that pixel-wise surface normal and height information would be difficult for a CNN to learn from the raw depth image, and so the HHA image may provide more discriminative information. In this section the full set of steps to obtain an HHA image from a depth image is presented.

The first step in the HHA encoding algorithm is to convert the depth image to a point cloud using the intrinsic and extrinsic camera matrices, which are assumed to accompany the depth image in the dataset used. Considering that we now have a point cloud $P := \{x_i, y_i, z_i\}$, for each $p_i \in P$, a linear least-squares plane-fitting approach is employed to fit a normal to the plane centered around p_i by considering windows of size 3×3 and 5×5 , with appropriate truncation

at the image borders. The equation of a plane with normal $n = (a, b, c)$ is given by $ax + by + cz + d = 0$. Assume that $d = -1$, which is possible since we are not interested in the distance from the plane to the origin, but rather the vector (a, b, c) . Then since the depth values in the point cloud are all non-zero³, we can further simplify the plane equation, for the points in the local neighborhood for which we wish to estimate the plane normal, to $a'\frac{x}{z} + b'\frac{y}{z} + c = \frac{1}{z}$.

For a set of points $\{p_j\}$ in a window around p_i (including p_i), this amounts to solving for $x := [a', b', c]^T$ that minimizes $\|Ax - m\|_2$ where

$$A = \begin{bmatrix} x_0/z_0 & y_0/z_0 & 1 \\ x_1/z_1 & y_1/z_1 & 1 \\ \vdots & \vdots & \vdots \\ x_k/z_k & y_k/z_k & 1 \end{bmatrix} \text{ and } m = \begin{bmatrix} 1/z_0 \\ 1/z_1 \\ \vdots \\ 1/z_k \end{bmatrix} \quad (2.4)$$

Assuming that the 3×3 matrix $A^T A$ is invertible, the linear least squares solution to this minimization problem is achieved as $x = (A^T A)^{-1} A^T m$. Observe that

$$A^T A x = A^T m \implies \begin{bmatrix} \sum (\frac{x_j}{z_j})^2 & \sum \frac{x_j}{z_j} \frac{y_j}{z_j} & \sum \frac{x_j}{z_j} \\ \sum \frac{x_j}{z_j} \frac{y_j}{z_j} & \sum (\frac{y_j}{z_j})^2 & \sum \frac{y_j}{z_j} \\ \sum \frac{x_j}{z_j} & \sum \frac{y_j}{z_j} & |N(p_i)| \end{bmatrix} \begin{bmatrix} a' \\ b' \\ c \end{bmatrix} = \begin{bmatrix} \sum \frac{x_j}{z_j^2} \\ \sum \frac{y_j}{z_j^2} \\ \sum \frac{1}{z_j} \end{bmatrix} \quad (2.5)$$

where $|N(p_i)|$ is the number of points considered in the window centered at p_i . A known linear algebra fact is that for an invertible matrix B ,

$$B^{-1} = \frac{adj(B)}{det(B)} \quad (2.6)$$

where $adj(B)$ is the transpose of the co-factor matrix of B and $det(B)$ is the determinant of B , both of which can be efficiently computed for a 3×3 matrix. Using this fact, we can compute $(A^T A)^{-1}$ and obtain

$$x = (A^T A)^{-1} A^T m = \frac{adj(A^T A)}{det(A^T A)} A^T m \quad (2.7)$$

Then x is the desired normal to the plane in the neighborhood of p_i . The vector x is then normalized⁴. After computing surface normals for each point in the depth image, the normals are reoriented to point outward from the scene.

³In the code, the missing depth values, which do have $z = 0$ are treated as follows: in the matrix A for any z_i that is missing, the corresponding values x_i/z_i and y_i/z_i are set to 0. Despite the fact that the in-painted depth maps do not have missing z values, perhaps the authors did not want to *falsely infer* surface normals from this in-painting process.

⁴In [7] the normals are computed in parallel for each p_i with no assumption of the invertibility of $A^T A$ and without the multiplicative factor $\frac{1}{det(A^T A)}$, after which the normals are normalized.

The next step in the HHA encoding algorithm is to obtain a gravity vector g using the algorithm proposed by Gupta et al. [22]. The gravity vector is used to rotate the points and surface normals of the point cloud, and compute the angle that each rotated surface normal makes with g , as well as derive a height value from each point in the rotated point cloud. The rotated surface normals corresponding to flat horizontal surfaces should be roughly parallel to g , while those rotated normals corresponding to vertical surfaces such as walls should be roughly orthogonal to g . The estimation algorithm proceeds from an initial guess, which is iteratively refined for a fixed number of iterations by a two-step procedure. Set $y_0 = [0, 1, 0]$, the first guess for the gravity vector. Let $d = 45^\circ$.

Step 1

The surface normals $\{n_i\}$ obtained from the plane-fitting algorithm with the 5×5 sized windows are used to determine two sets:

- $M_{||} = \{(n_i | (\theta(n_i, g) < d) \vee (\theta(n_i, g) > 180^\circ - d))\}$
- $M_{\perp} = \{(n_i | (\theta(n_i, g) > 90^\circ - d) \vee (\theta(n_i, g) < 90^\circ + d))\}$, where $\theta(a, b)$ is the angle between vectors a, b in degrees, assuming $0 \leq \theta \leq 180$.

Step 2

Next we wish to solve, using the above sets, for a new gravity vector of length 1 that is maximally parallel to vectors in $M_{||}$ and maximally perpendicular to vectors in $M_{\perp}$, i.e.

$$\min_{g: \|g\|_2=1} \sum_{n \in M_{\perp}} \cos^2(\theta(n, g)) + \sum_{n \in M_{||}} \sin^2(\theta(n, g)) \quad (2.8)$$

Since $\cos^2(\theta) + \sin^2(\theta) = 1 \implies \sin^2(\theta) = 1 - \cos^2(\theta)$ then we can re-write the minimization problem as $\sum_{n \in M_{\perp}} \cos^2(\theta(n, g)) + \sum_{n \in M_{||}} (1 - \cos^2(\theta(n, g)))$ which is the same as minimizing

$$\sum_{n \in M_{\perp}} \cos^2(\theta(n, g)) + \sum_{n \in M_{||}} -\cos^2(\theta(n, g)) = g(N_{\perp}N_{\perp}^T - N_{||}N_{||}^T)g^T \quad (2.9)$$

where $N_{\perp}, N_{||}$ are the vectors in $M_{\perp}, M_{||}$ stacked column-wise. Note that both $N_{||}N_{||}^T$ and $N_{\perp}N_{\perp}^T$ are symmetric matrices, and so is their difference. From the Courant-Fischer theorem from linear algebra, the minimal vector g consists of the eigenvector with the smallest eigenvalue of the matrix $N_{\perp}N_{\perp}^T - N_{||}N_{||}^T$, which can be obtained with an eigensolver.

Steps 1, 2 are repeated 4 more times, where step 1 involves assigning surface

normals to $M_{\perp}, M_{\parallel}$ for the new gravity vector obtained from the previous step. Afterwards d is set to 15° for 5 more iterations of steps 1, 2, yielding a final estimate g , the gravity vector. When $d = 15^\circ$, the sets $M_{\perp}, M_{\parallel}$ are smaller. Typically, as the algorithm progresses, $M_{\perp}$ would contain surface normals obtained from walls, while $M_{\parallel}$ would contain normals obtained from the floor and flat surfaces that are parallel to the gravity vector such as table tops, counters, etc.

To recap, at this point in the HHA encoding algorithm, we have surface normals obtained from plane-fitting using 3×3 and 5×5 windows around each pixel in the point cloud, as well as a gravity vector. A rotation matrix R is then obtained from the rotation of the vector $[0, 1, 0]$ to the gravity vector, after which the surface normals $N_{3 \times 3}$ obtained from the 3×3 sized windows are rotated as in Rn_i^T for each $n_i \in N_{3 \times 3}$, to obtain *gravity aligned surface normals*. Similarly the point cloud values are rotated as in Rp_i^T to obtain P' , the *gravity aligned point cloud*.

The mathematics of mapping depth images to point clouds is such that for y -values less than the image plane principal point's y coordinate in pixel coordinates, the corresponding y value of the point cloud will be negative, inverting the natural height relationship of standard 3D coordinate space photographed with a camera placed at human chest height. For this reason, the y values of P' are multiplied by -1 , and hence by taking the minimal y value of the rotated, scaled P' we obtain the smallest height value h_0 of any point in our point cloud. From here, each point in P' , and hence depth image pixel, can be expressed in terms of its *height above h_0* , the second channel of the HHA encoding, using appropriate scaling to the range $[0, 255]$. In Figure 2.3, as objects and surfaces in the scene have a higher y value in world space coordinates, the image gets whiter, and vice versa.



Figure 2.3: A *height above ground image* obtained from a depth image from the SUN RGBD dataset [8]. Left: RGB Image, Right: Height Above Ground Image.

The final channel in the HHA encoding is the *angle with gravity* channel. Given the rotated surface normals $N'_{3 \times 3}$, for each point $n'_i \in N'_{3 \times 3}$, we compute the

angle θ_i , the *angle with gravity*, as $\theta_i = \arccos(n'_i \cdot g)$ (in degrees), to obtain, after rounding, an integer in the range $[0, 180]$. An example angle with gravity image is shown in Figure 2.4. The top of the chair, package top, and floor, which have surface normals that are parallel to gravity are dark, indicating a small angle with gravity. By contrast, the walls are whiter, indicating a larger surface normal angle with gravity.



Figure 2.4: An *Angle with Gravity Image* obtained from an RGBD image from the SUN RGBD dataset [8]. Left: RGB Image, Right: Angle With Gravity Image.

Concatenating each of these grayscale images, one obtains an HHA image. See Figure 2.5 for an example HHA image, displayed as an RGB image, where the red channel is associated with horizontal disparity, the green channel is associated with height above the ground, and the blue channel is associated with the angle with gravity value. Note how as the horizontal disparity value increases, the image gets redder. As the height of the objects/surfaces increases, the image gets greener. Table/chair tops have less blue content, while walls and the sides of tables/desks have more blue content, demonstrating the angle their surface normals make with the estimated gravity vector.



Figure 2.5: An example HHA image obtained from an RGBD image from the SUN RGBD dataset [8]. Left: RGB Image, Middle: Depth Image, Right: HHA Image.

2.1.4 The Benefits of Depth Images

At this point it is reasonable to ask what the benefit of depth images might be by comparison with RGB images. Indeed, a series of elementary observations will motivate the use of the additional depth channel.

Observation 1: Object boundaries in depth images are invariant to lighting conditions.

In Figure 2.6, the object boundaries of the chair closest to the camera in the RGB image are difficult to distinguish due to a heavy shadow induced by the table, and a low level of ambient lighting in the room. By contrast the chair’s shape and boundary are quite clear in the depth image.

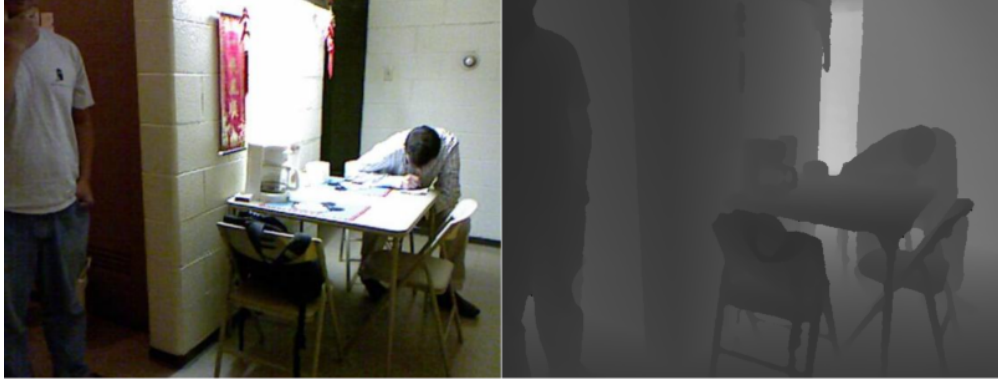


Figure 2.6: A comparison of object boundaries in an RGB image and a depth image from the NYUDv2 dataset [6].

Observation 2: Depth images encode an estimate of the volume of objects.

Suppose we have a bounding box $(x_{min}, y_{min}, x_{max}, y_{max})$ enclosing an object within a depth image where (x_{min}, y_{min}) are the smallest respective x, y pixel coordinates of the box, and (x_{max}, y_{max}) are the largest. A 3D bounding box in world space coordinates can be obtained by converting the depth image to a point cloud, and using the min and max z values of the 3D points whose (x, y) values are mapped from pixels contained within the original bounding box. In turn, this leads to a rough volume estimation of an object, which should remain relatively consistent for objects of a certain class with small variance in volume. Contrast this volume estimation with a 2D bounding box of an object in an RGB image. As the RGB camera draws closer to the object, its bounding box increases in size, and vice versa, creating the illusion that the object is getting larger. By comparison,

the volume estimate obtained from the depth image is *independent of viewpoint or distance from the depth sensor*, assuming the object is within the sensor’s range.

Observation 3: Object detection using depth images does not make use of color or texture information.

This is a double-edged sword. On the one hand, depth images are not fooled by edges in color space that do not correspond to depth edges, which ideally should represent sharp changes in objectness, or a distinction between horizontally adjacent objects. Consider Figure 2.7, a picture of a dog on a toilet. The color and texture information of the dog’s fur and skin, and of the surface of the toilet lid, are completely abstracted in the depth image as rough shapes. If an object’s detection is aided by textural details, this is a detraction. On the other hand, if a rough shape is all that’s needed for an object detection algorithm to successfully identify an object, then the extraneous textural information is discarded in favor of a simpler structural estimation of the object.

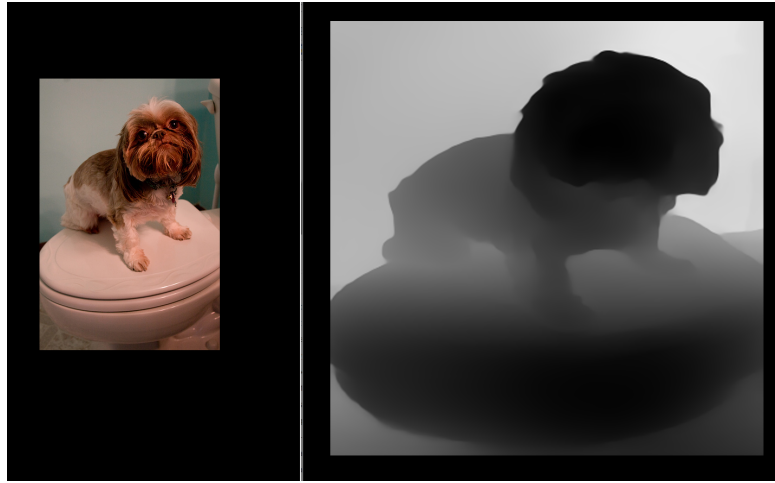


Figure 2.7: A depth estimate of an MS-COCO dataset image [23] obtained from the MIDAS depth estimation network [24].

Observation 4: The height above ground image generated from a point cloud encodes statistical priors about supporting relations.

Objects and surfaces typically occupy certain height ranges within an image, as well as supporting relations. For example, tables tend to support computer monitors, but certainly not the other way around. Ceilings usually occupy the highest height value of any object or surface in a room. It is reasonable to hypothesize

that the height channel of an HHA image can help distinguish which pixels are likely to belong to support and structural classes such as the floor, tables, walls, and the ceiling.

Observation 5: Surface normals encode shape information.

Given a point cloud, surface normals for each pixel can be estimated using a variety of plane fitting algorithms. Consider the angle with gravity channel of an HHA image, described in Section 2.1.3. This channel encodes information derived from surface normals, which can potentially be used to extract discriminative features from object instances that have similar textural information but different surface geometry.

Observation 6: Depth images can help disambiguate scale distortions caused by perspective projection.

Consider Figure 2.8. In this photo, a man appears to be holding up the Leaning Tower of Pisa due to an optical illusion caused by the scaling properties inherent to perspective projection. The man is much closer to the camera than the tower. The tower’s height after projection is roughly scaled to the same size as the man. In a depth image, the pixels associated with the man would receive much smaller values than those associated with the tower, creating an obvious distinction between entities which should be visible with a depth edge.



Figure 2.8: Scale distortions caused by perspective projection. Image obtained from Flickr under a creative commons license [25].

2.2 Computer Vision Tasks And Performance Metrics

In this section, the tasks of object detection, instance segmentation, semantic segmentation and panoptic segmentation are formally defined. In addition, metrics used in this work for model comparison on each task are introduced. These metrics will help show that the depth-aware models introduced in Section 4.2 outperform their pure RGB counterparts.

2.2.1 Object Detection

The task of *object detection* combines two important sub-components:

- *Recognition* or *Classification*, where the goal is to determine if an image, or crop of an image, contains a specific object type or not.
- *Localization*, which consists of giving a mathematical description of the location of an object in an image. The location of an object is given by a rectangular crop of an image, frequently described with pixel coordinates $(x_{min}, y_{min}), (x_{max}, y_{max})$, giving the top left corner and bottom right corner of the crop, referred to as a *bounding box*.

An object detection dataset consists of labeled images $\{I_i\}$ with ground truth annotations $GT_i = \{(B_j, L_j) | 1 \leq j \leq N(I_i)\}$ where (B_j, L_j) are pairs of bounding boxes and class labels, and $N(I_i)$ is the number of annotated objects for image I_i , which varies per image. This annotation type contrasts with the image classification task, which does not localize the object in the image, and for which there is only one salient object per image. The class labels belong to some overall set of $C := \{C_j\}$ classes.

An *object detection model* trains on the training dataset, and outputs a set of predictions for each testing image. Denote this set of predictions on the test set as $M_i = \{(B'_j, L'_j, P_j) | 1 \leq j \leq N'(I_i)\}$ where (B'_j, L'_j) are predicted bounding box and class label pairs with a predicted *confidence value* P_j , and $N'(I_i)$ is the number of predicted boxes for image I_i . Typically only predictions with a confidence value larger than some threshold, commonly 0.5, are chosen as the final model output for a given testing image. See Figure 2.9 for an example detection.

The assessment of the localization accuracy of a predicted bounding box is given by its *Intersection over Union (IoU)* with a ground truth box, in which the ratio of the overlap of the bounding box areas to the union of the box areas is



Figure 2.9: The output from an object detection model input with an image from the MS-COCO dataset [23], showing a bounding box, a confidence score, and a class label.

computed for two boxes B_A, B_C as

$$IoU(B_A, B_C) = \frac{area(B_A \cap B_C)}{area(B_A \cup B_C)} \quad (2.10)$$

The closer the predicted box is to the ground truth box, the higher the IoU value, and vice versa, where the IoU of two bounding boxes is strictly in the range $[0, 1]$. See Figure 2.10 for a visualization.

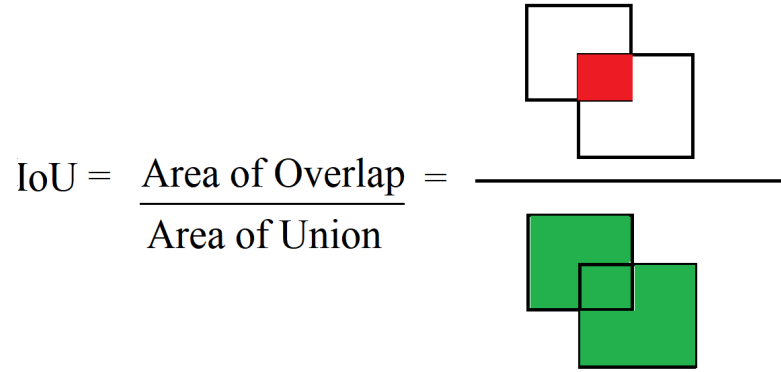


Figure 2.10: The IoU of two bounding boxes.

In order to compute precision and recall, define the notion of false positives/negatives and true positives/negatives in the object detection context as:

- A True Positive (TP) is a correct detection of a ground-truth bounding box. A detection is considered correct if the IoU of the detected box and some ground truth box exceeds a threshold, and the class labels of the two boxes are identical. Detections are considered in decreasing order of confidence scores. A ground truth bounding box can only be detected once; any pre-

dicted bounding box that meets the IoU threshold only for boxes that have already been detected is considered a false positive.

- A False Positive (FP) is an incorrect detection of type:
 - 1. A detection that does not meet an IoU threshold for any ground truth box.
 - 2. A detection that does meet the IoU threshold for at least one box but does not predict the correct class label for any matched ground truth box.
 - 3. A duplicate detection.
- A False Negative (FN) is a ground truth bounding box that has not been detected.
- A True Negative (TN) is not entirely meaningful in this context since object detection models do not detect the background class or the lack of an object.

Ignoring confidence values for the moment, suppose that for G ground truth bounding box/class label pairs, an object detection model outputs N predicted box/class pairs with corresponding TP, FP, FN values, then

$$Precision = P = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}} \quad (2.11)$$

$$Recall = R = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truth boxes}} \quad (2.12)$$

Consider the possible confidence values output by a model, rounded to two decimal points, say $L := \{0.50, 0.51, \dots, 0.98, 0.99\}$, rounding downwards for values in $[0.99, 1)$. Precision and recall can be computed at each confidence value by only considering predictions that have confidence values larger than or equal to a threshold, denoted as a pair (P_c, R_c) for some confidence value c . This results in a set of precision-recall pairs, ordered by decreasing confidence value. The precision-recall pairs are computed for each class separately.

Two popular metrics for object detection are the Pascal VOC 2012 mean average precision or mAP_{50} ⁵, and the MS-COCO mean average precision mAP , which were used to compare object detection models in the Pascal VOC 2012 and MS-COCO 2014 object detection competitions respectively [26].

⁵The Pascal VOC 2012 mAP metric will be written in this thesis as mAP_{50} to distinguish it from the COCO mAP metric, where the former is also sometimes denoted as mAP .

The Pascal VOC mAP_{50} metric fixes the detection IoU threshold to be 0.5, and averages the average precision, denoted as AP_{50} , for each class. Consider the detections for a specific class C_i with corresponding recall-precision pairs $\{(P_c, R_c)\}$. The average precision of the model output for this class is given by an estimate of the area under the precision-recall curve obtained by interpolating between all T precision-recall pairs as:

$$AP_{50}(C_i) := \sum_{t=0}^T (R_{t+1} - R_t) \text{Max}_P(t+1) \quad (2.13)$$

with added endpoints $(P_0, R_0) = (1, 0)$, $(P_{T+1}, R_{T+1}) = (0, 1)$ and

$$\text{Max}_P(t) = \max\{P_{t'} | t' \geq t\} \quad (2.14)$$

Averaging the AP_{50} over all classes yields

$$mAP_{50} = \frac{1}{|C|} \sum_j AP_{50}(C_j) \quad (2.15)$$

The Pascal VOC 2012 metric was criticized for not taking into account models that perform similarly at the fixed IoU threshold of 0.5, but that differ in terms of the geometric precision of their respective predicted bounding boxes. Suppose we have two models that have the same similar recall-precision curves for each class when tested on a dataset, but with one model having much higher average IoU values for each true positive detection. The model with the more accurate localization then clearly has more functional utility, but the mAP_{50} metric would yield roughly the same score for each model. Hence the MS-COCO style mAP addresses this discrepancy by computing the average precision for each class in the same way as the mAP_{50} metric, but averaging over IoU thresholds from the set $Tr := \{0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95\}$, i.e. computing

$$mAP = \frac{1}{|Tr|} \sum_{\tau \in Tr} mAP_{\tau} \quad (2.16)$$

A less common metric used is the MS-COCO style *mean average recall*. In this metric, for each IoU value in Tr , for a fixed max number of detections (typically a value $D \in \{10, 100, 1000\}$), the recall for each class is computed and then averaged. The average recall is then averaged over each IoU value, i.e.

$$mAR^D = \frac{1}{|C|} \frac{1}{|Tr|} \sum_{\tau \in Tr} \sum_{C_i \in C} R_{\tau}^D(C_i) \quad (2.17)$$

where $R_{\tau}^D(C_i)$ is the recall at IoU threshold τ for class C_i using a maximum of D detections.

A class of object detection architectures known as *region-based methods* such as R-CNN [27], Fast R-CNN [28], and Faster R-CNN [12] make use of so-called *region proposals*, which are candidate rectangular regions within the training or testing image that a region proposal algorithm has selected for classification and regression. These region proposals are class agnostic, and are output with a confidence value, after which the architecture classifies the proposals. Hosang et al. [29] observe that region-based models with a higher average recall perform better on object detection tasks.

For a given IoU threshold, the recall of a set of region proposals with respect to a set of ground truth bounding boxes in a testing set is computed as the total number of ground truth boxes with IoU with some region proposal larger than the IoU threshold. The proposals are considered in decreasing order of their confidence scores, and once a ground truth box has been covered by a proposal, it cannot be considered again, to eliminate duplicate region detections. The number of proposals considered is not limited in this metric. For each IoU threshold, the region proposal generation algorithm has a corresponding recall value, generating a set of recall-IoU pairs which can be used to visualize a recall-IoU curve.

Novel to this thesis, the MS-COCO definition of mean average recall is adapted for region proposals by considering all objects to belong to the same class, computing an average of the recalls for IoU thresholds in Tr as

$$AR_{RP} = \frac{1}{|Tr|} \sum_{\tau \in Tr} R_{\tau} \quad (2.18)$$

where R_{τ} is the recall of a region proposal algorithm for some IoU threshold τ . This computation is simpler to compute than the metric presented by Hosang et al. [29], which consists of estimating the area under the continuous recall-IoU curve, and doubling the resultant value.

2.2.2 Instance Segmentation

The task of instance segmentation in some ways extends object detection, where instead of fitting a tight bounding box around an object, the goal is to predict a segmentation mask for each object. The mask is a binary image of the same size as the input image with a value of 1 assigned to pixels that belong to an object

instance, and a value of 0 assigned otherwise. Individual instance masks are commonly visualized by assigning a bright color, say pinkish red or white, to pixels where there is a 1 value in the mask, which is then overlaid over the original image as in Figure 2.11. For visualizing multiple instance masks, a different overlay color is assigned per instance.



Figure 2.11: The output of an instance segmentation model input with an image from the Pascal VOC 2012 dataset [30] consisting of a bounding box, confidence score, class label, and an instance mask overlaid in white over the original color image.

Formally, for a set of images $\{I_i\}$ from an instance segmentation dataset, the ground truth annotation of each image consists of a set of pairs of masks and class labels $GT_i := \{(M_i, C_i) | 1 \leq i \leq |N(I_i)|\}$ where M_i is a binary instance mask for an object instance, C_i is its corresponding class label, and $N(I_i)$ is the number of annotated instances in the image. Crucially, differentiating from semantic segmentation and object detection, instance segmentation has two important components:

- The ground truth binary instance masks in a single image do not overlap, whereas in the object detection task the ground truth bounding boxes may overlap.
- Instances in the same image, like objects in an object detection dataset, may belong to the same class, but are treated separately. For example in the task of semantic segmentation, to be discussed in Section 2.2.3, two different cups would get the same cup class label, with no distinction between the separate instances. By contrast, in instance segmentation, *each individual cup will get its own distinct ground truth mask annotation.*

The metrics used in the literature for instance segmentation make use of mask Intersection over Union (IoU), where for two binary masks M_A, M_B , the intersection area of the two masks is the number of pixel locations in both masks that have a 1 value (the logical *and* operation). The area of the union of two masks is the number of pixel locations that have a 1 in either one mask or both masks (the logical *or* operation). Formally,

$$\text{Mask IoU}(M_A, M_B) = \frac{\text{Area}(M_A \cap M_B)}{\text{Area}(M_A \cup M_B)} \quad (2.19)$$

and indeed the metrics of mAP_{50} , mAP and average recall can be computed identically as in the object detection case by replacing bounding box IoU with mask IoU.

Within the literature, authors of instance segmentation models will often include qualitative comparisons alongside quantitative measurements of mask model performance to examine the aesthetic quality of the predicted instance masks. Masks are judged for qualities such as how accurate the mask boundary is, or the presence of incorrect *splotches* within the mask where a patch of pixels is incorrectly labeled as not belonging to the object.

2.2.3 Semantic Segmentation

For the task of semantic segmentation, the ground truth annotation assigns each pixel in each image a class label from a set of $C := \{C_i\}$ class labels. The goal is to predict the correct class label for each pixel in an image. Kirillov et al. [31] clarify the distinction between *things* vs. *stuff*, describing the full scope of possible semantic segmentation classes. Intuitively, *things* are object classes that have clear boundaries and can have multiple instances, such as animals, chairs, or tables. In the case of indoor settings, *stuff* delineates the background of the image into structural classes such as the floor, walls, the ceiling, etc. In the case of outdoor settings, stuff classes form the image background, encompassing categories such as grass, the sky, trees, etc. Semantic segmentation concerns itself with both things and stuff, but the thing instances are not differentiated from one another.

Four metrics for the comparison of semantic segmentation models employed in this thesis are⁶:

⁶Note that in the computation of each metric, the *unknown* class, typically given the numerical label 0, is ignored. A pixel predicting a non-zero class at the location of an unknown pixel will not be penalized. However if a model predicts the unknown class for a pixel that is not assigned the unknown label, this will be penalized in both the accuracy and IoU metrics.

- The *Pixel Accuracy* metric, denoted as Acc , computed as

$$Acc = \frac{\sum_{C_i \in C} N(\hat{C}_i, C_i)}{s} \quad (2.20)$$

where $N(\hat{C}_i, C_j)$ is the number of pixels of class j predicted to belong to class i , and s is the total number of pixels in the testing set. The value $N(\hat{C}_i, C_i)$ is the number of pixels of class i correctly predicted. Pixel accuracy is the crudest measure of the performance of a semantic segmentation model. Models that correctly classify categories that occupy the majority of the image will perform decently well with this metric regardless of whether or not they categorize the remaining classes correctly.

- The *Mean Pixel Accuracy* metric, denoted as $mAcc$, which computes the average class-level pixel accuracy as

$$mAcc = \frac{1}{|C|} \sum_{C_i \in C} \frac{N(\hat{C}_i, C_i)}{s_i} \quad (2.21)$$

where s_i is the total number of pixels belonging to class i in the testing set.

- The *Mean IoU* (intersection over union) metric, denoted $mIoU$, which is computed as

$$mIoU = \frac{1}{|C|} \sum_{C_i \in C} \frac{N(\hat{C}_i, C_i)}{s_i + (\sum_{\hat{C}_j \in C} (N(\hat{C}_j, C_i)) - N(\hat{C}_i, C_i))} \quad (2.22)$$

This is the most commonly used metric for semantic segmentation performance.

- The *Frequency Weighted IoU* metric, denoted as $fwIoU$, which computes the IoU per each class weighted by the frequency of the class pixel occurrences within the test dataset, as

$$fwIoU = \sum_{C_i \in C} \frac{s_i}{s} \frac{N(\hat{C}_i, C_i)}{s_i + (\sum_{\hat{C}_j \in C} (N(\hat{C}_j, C_i)) - N(\hat{C}_i, C_i))} \quad (2.23)$$

Similarly to instance segmentation, authors of semantic segmentation models often give qualitative evaluations of their segmentation masks to discuss the aesthetic quality of the model output.

2.2.4 Panoptic Segmentation

The task of panoptic segmentation, introduced by Kirillov et al. [31], unifies the goals of instance segmentation and semantic segmentation, and offers a serious

challenge for modern computer vision research. The word *panoptic* is defined by the Oxford dictionary as “*showing or seeing the whole at one view*” [32].

A panoptic segmentation annotation for each image I_i in a dataset can be formally defined as a set $GT_i := \{M_i\}$, where M_i is a ground truth panoptic mask of the same resolution as I_i . For each pixel coordinate in M_i , say (x, y) of I_i , a ground truth tuple is assigned as (C_i, I_i) where C_i is the class label of the pixel, and I_i is the instance number of the pixel. For background classes, a.k.a *stuff* classes such as walls, the sky, the ground, etc., the instance number of the corresponding pixel ground truth annotation is assigned to be 0. However, for instances that may have multiple distinct units within an image, such as individual people or individual cars, each pixel belonging to each separate unit *gets its own identification number*. The goal is to predict a set of instances as well as the background classes simultaneously, achieving a predicted panoptic mask where each pixel has both a class label and an instance number. Panoptic segmentation differs from instance segmentation in several ways:

- In instance segmentation, each pixel in the image is not necessarily assigned a class number or instance number, whereas every pixel in the panoptic segmentation mask has both a class label and an instance number. Instance segmentation models do not make stuff predictions.
- Instance segmentation allows for the predicted masks to overlap with each other in the computation of the *mAP* metric, whereas each pixel in a predicted panoptic mask has a singular predicted instance number.

In Figure 2.12, observe that in photo a) there are multiple chair and desk instances, as well as a single door instance. Photo b) is created by combining each instance mask, assigning the pixel locations of each instance a distinct color. Any pixel that is not assigned to an instance is visualized with the color black. In the semantic segmentation mask in photo c), the chairs are all assigned the same color (white), as are the desks (gray). In the panoptic segmentation mask in photo d), the individual instances are assigned different ids, and visualized with different colors. In both c) and d), the stuff classes such as the floor (green) and the wall (orange) are assigned ground truth labels. Pixels in the unknown class are shown in auburn red in photos c) and d).

In order to compare panoptic segmentation models, the *Panoptic Quality* metric introduced by Kirillov et al. [31] is computed for each class in the dataset

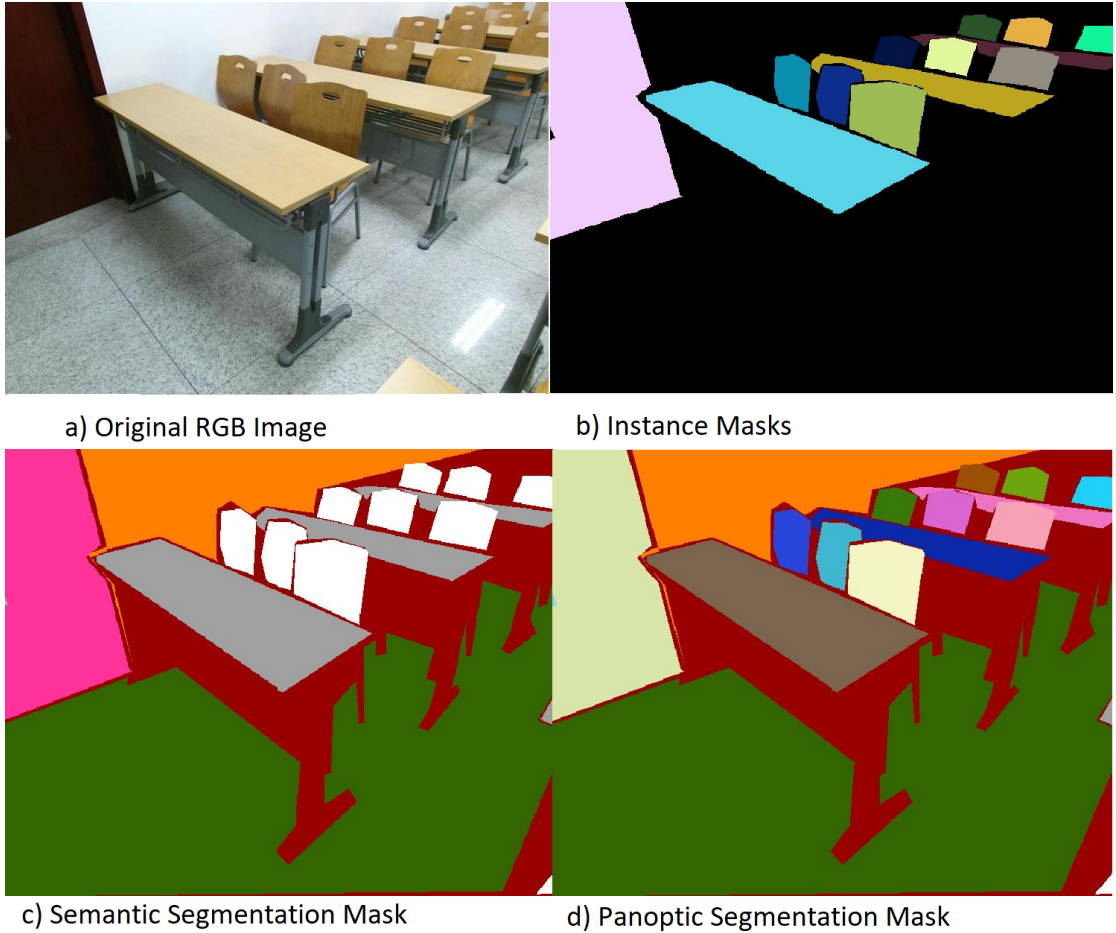


Figure 2.12: A comparison between segmentation types for an image from the SUN RGBD dataset [8].

independently and averaged over classes. Define a region, also referred to as a segment, as a set of pixels with the same class label and instance number. For each image, a set of ground truth segmentation regions are matched with the predicted segmentation regions. Predicted instance regions are matched to ground truth instances, using class labels. The ground truth stuff regions are matched to the predicted stuff regions, again using class labels. A true positive (TP) match is defined for mask IoU between segments greater than 0.5, along with the correct predicted category label. False positives (FP) are defined as unmatched predicted segments; false negatives (FN) consist of unmatched ground truth segments. Kirillov et al. [31] show that given a predicted and ground truth panoptic segmentation of an image, each ground truth segment can have at most one corresponding predicted segment with IoU strictly greater than 0.5 and vice versa. For this reason duplicate matches are not an issue. For a given class, the

panoptic quality for predictions from a model over a test set is defined as

$$PQ = \frac{\sum_{(p,g) \in TP} IoU(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \quad (2.24)$$

where (p, g) are matched regions with p a predicted region and g a ground-truth region. Decomposing

$$PQ = \frac{\sum_{(p,g) \in TP} IoU(p,g)}{|TP|} * \frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \quad (2.25)$$

$$= \frac{\sum_{(p,g) \in TP} IoU(p,g)}{|TP|} * \frac{2}{recall^{-1} + precision^{-1}} \quad (2.26)$$

one can see that the PQ metric is the product of a quality term on the left, where matched segments with higher IoU will yield a greater score, and a recognition term on the right. The recognition term is equivalent to the F_1 score, defined as

$$F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} \quad (2.27)$$

The F_1 score is a frequently used metric in machine learning that balances the weight of recall and precision for measuring model performance on classification tasks.

2.3 Residual Networks as a CNN Backbone

Residual Networks, or *ResNets*, introduced by He et al. [14], are a popular set of deep CNN networks used by many state-of-the-art works in the computer vision literature. Motivated by the success of deep networks such as the VGG architecture [33], early experiments with even deeper networks would expose a commonly encountered issue. Namely, networks with more layers tended to exhibit higher training error than their shallower counterparts. This inability to converge to a lower loss on the training set correlates with decreased performance at test time. To mitigate this issue, residual networks make use of residual connections, also referred to as skip connections. ResNets will be used as the backbones for all models developed in this thesis, in particular the ResNet-50 and ResNet-101 variants. Moreover, the concept of residual learning will be explored as part of a learned fusion approach between tensors output from backbones trained separately on the RGB and depth modalities, discussed in Section 4.1.

Consider a deep neural network architecture with k layers. He et al. [14] argue that there must exist a deeper architecture that exhibits similar training accuracy

for the image classification task, constructed by adding layers that are capable of learning the identity mapping. As a result, such a deeper model should exhibit the same training error as the shallower architecture; hence the development of specialized layers that can learn identity mappings.

In the ResNet-50 and ResNet-101 models, the *bottleneck layer* is developed for this purpose. Bottleneck layers are parameterized by a number of input channels, an intermediate number of feature channels which is smaller than the number of input channels, and a positive integer referred to as the expansion factor, usually set to 4. Additionally, some bottleneck layers perform downsampling. Here the PyTorch implementation is described [15]. For an input tensor x , a sequence of convolutions are performed, while still retaining a reference to the original value of x . Each convolution is followed by batch normalization and ReLU activation. In particular, acting on x , the following steps are computed:

1. A 1×1 convolutional layer of stride 1 that decreases the number of channels of x to the intermediate dimension.
2. A 3×3 convolutional layer with input channels equal to the number of output channels. The stride used is 2 if downsampling is performed, and 1 otherwise.
3. A 1×1 convolution of stride 1 with input planes equal to the intermediate dimension, and output planes equal to $t = \text{expansion} * \text{intermediate}$.
4. If downsampling is performed in 2), then x is downsampled using a 1×1 convolution of stride 2 with out channels equal to t . If downsampling has not been performed, there are two cases to consider. If the number of feature map channels of x is not equal to t , then a 1×1 convolution with stride 1 and t outplanes is performed on x , otherwise x is not modified before the residual connection.
5. Finally the *residual connection*, which allows for the learning of the identity mapping, is performed with an element-wise addition of the output of 3) and x (potentially modified as in 4), followed by ReLU activation. See Figure 2.13 for an example bottleneck operation.

The grouping of outer layers to structure the description of ResNets are labeled ConvX for $X \in \{1, 2, 3, 4, 5\}$, where:

- Conv1 consists of a 7×7 convolution of stride 2 with 64 output channels operat-

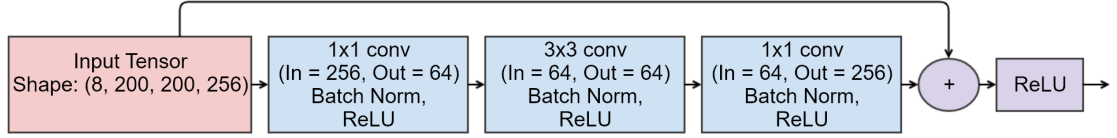


Figure 2.13: An example bottleneck operation without downsampling with 256 input channels, an intermediate dimension of 64, an expansion factor of 4, and a batch size of 8.

ing on a 3-channel input RGB image, followed by batch normalization and ReLU activation, followed by a 3×3 max pooling layer of stride 2.

- Conv2 to Conv5 contain chains of bottleneck layers where downsampling is performed in the first bottleneck layers of Conv3, Conv4, and Conv5. See Table 2.1 for a summary of the ResNet-50 and ResNet-101 architectures, where the convolutions of the bottleneck layers are given in square brackets. Each entry within the square brackets gives the kernel size and the number of output channels of the convolution.

Layer Name	ResNet-50	ResNet-101
Conv1	7×7 convolution, stride 2	7×7 convolution, stride 2
	3×3 max pooling, stride 2	3×3 max pooling, stride 2
Conv2	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
Conv3	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 256 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 256 \end{bmatrix} \times 8$
Conv4	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$
Conv5	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$

Table 2.1: Layer details of the ResNet-50 and ResNet-101 models without the image classification network heads.

2.4 Object Detection with Faster R-CNN

In this thesis, the object detection framework used will be Faster R-CNN [12], chosen for its impressive performance and in order to compare the eventual depth-

aware Faster R-CNN developed in Section 4.2 to other depth-aware R-CNN based object detection models in the literature, discussed in Section 3.1. Object detection datasets can feature many objects in a single image. For instance, the MS-COCO 2014 object detection dataset [23] contains an average of 7.7 object instances per image. In order to identify multiple objects per image, region-based strategies follow the approach of first obtaining a set of candidate *region proposals*, which are rectangular crops of the image. These regions are then classified, with their co-ordinates regressed to form a tight fitting box around an object, in the case of a non-background class prediction.

The fundamental contribution of the Faster R-CNN [12] framework is the development of Region Proposal Networks (RPNs) for use with the Fast R-CNN framework [28], which replace the inefficient and non-learned Selective Search algorithm [34] for region proposals used previously in Fast R-CNN [28] and R-CNN [27]. RPNs input a convolutional feature map and output a set of region proposals, scored by objectness, where a high objectness score is interpreted to be a confident prediction that a given region proposal has an IoU above a given threshold with some ground truth bounding box.

Consider a given CNN backbone, say a ResNet variant, or VGG-16, with network stride s . Without a classification head, the backbone input with an image of size (I_h, I_w) outputs a feature map F of spatial resolution $H \times W$ with M channels where $H = \lfloor I_h/s \rfloor$ and $W = \lfloor I_w/s \rfloor$. We consider k *reference anchors*, composed of rectangular boxes from a set of fixed areas and aspect ratios. These anchor boxes have areas $\{128^2, 256^2, 512^2\}$ and aspect ratios $\{0.5, 1, 1.5\}$, yielding $k = 9$ different anchor shapes. The original image is conceptually tiled with $H * W * k$ anchor boxes centered at $(s * i, s * j)$ for $0 \leq i \leq H$ and $0 \leq j \leq W$, see Figure 2.14.

The RPN network first performs a 3×3 convolution on F with stride 1 and same padding, where the number of output channels is less M . From there two sibling objectness and regression layers perform in parallel:

- A 1×1 convolution of stride 1 with k output channels to give a tensor of shape (H, W, k) . Define the objectness of a given region proposal to be a predicted probability of the proposal containing an object from any non-background class. At spatial location x, y and anchor type k , the value (x, y, k) is interpreted to be the objectness score of this anchor box centered



Figure 2.14: Left: Anchor Box Centers, Right: A set of anchor boxes centered at a specific location.

at $(s * x, s * y)$ in the original image.

- A 1×1 convolution of stride 1 and $4k$ output channels, where we interpret the tensor value at $(x, y, i * k)$, for $1 \leq i \leq 4$, to be the encoding presented in R-CNN [27] of the i 'th encoded value of the regressed bounding box for anchor type k .

Summarizing the details of Faster R-CNN using the approximate joint training scheme, the training of Faster R-CNN proceeds as follows:

- 1) The backbone outputs the feature map F on a training image.
- 2) The RPN gives bounding box regression and objectness predictions for each anchor box, yielding a region proposal at each anchor location. The anchor boxes are matched with the ground truth bounding box with maximal IoU. Anchors with IoU larger than 0.7 are labeled as positive, i.e they contain an object, while anchors with IoU less than 0.3 are considered background boxes and are labeled as negative. During training, a sample of 256 anchor locations are chosen for objectness and regression loss computation, with half the samples being positive anchors and the other half negative. A subset of region proposals are chosen for the next stage based on their objectness scores.
- 3) Using the region proposals output from step 2), non-maximal suppression [27] is applied in order to obtain a refined set of region proposals.
- 4) These region proposals are each projected onto F using the network stride, after which Region of Interest (RoI pooling) [28] is applied to each proposal to get a fixed sized tensor.

5) The tensors output in 4) are input to two fully connected layers, followed by the bounding box regression and classification heads from Fast R-CNN [28], referred to as RoI heads.

6) The RPN and RoI head losses are summed and backpropagated to the relevant backbone, RoI head, and RPN weights. The training loss of Faster R-CNN sums the region proposal network loss with the RoI head loss as

$$L = L_{RPN_{obj}} + L_{RPN_{reg}} + L_{cls} + L_{reg} \quad (2.28)$$

where

- $L_{RPN_{obj}}$ is the cross-entropy loss of the objectness predictions of the sampled anchor boxes, averaged over the number anchors sampled.

- $L_{RPN_{reg}}$ is the anchor box regression loss for the positive labeled anchors, averaged over the number of anchor locations.

- L_{cls} is the cross-entropy loss of the predicted classes of the region proposals, averaged over the number of proposals classified.

- L_{reg} is the regression loss of the region proposals predicted to be labeled with a non-background class, averaged over the number of proposals regressed.

The regression losses $L_{RPN_{reg}}, L_{reg}$ are defined in detail by Ren et al. [12]. At testing time, the regressed region proposals from step 5) which are predicted to belong to a foreground class with confidence value greater than or equal to 0.5 are input to a non-maximal suppression algorithm to obtain the final set of detections, reducing duplicate detections.

After proposing a depth-aware dual backbone modification of the Faster R-CNN framework in Section 4.2.1, the benefits of depth features fused with RGB features using two separate backbones will be assessed, in comparison to a network trained with RGB images only. In particular, the role of depth features in improving the localization and objectness of the region proposals, as well as the classification and regression of those proposals, will be examined.

2.5 Feature Pyramid Networks

Detecting objects at multiple scales proportional to the image size is a fundamental challenge in object detection. Models that perform well on large objects may fail to perform as well with smaller objects, or vice versa. Many object detection and image classification algorithms use multi-scale testing, in which the test image is

rescaled to multiple resolutions, followed by an algorithm to combine the model outputs for each scale to yield an overall prediction. Large image resolutions help to detect smaller objects, while smaller resolutions aid to detect larger objects. The use of multi-scale training is more involved. Approaches have included (prior to Feature Pyramid Networks):

- Using a fixed set of image resolutions for resizing and either training each image at each resolution, or randomly sampling each resolution as in Fast R-CNN’s multi-scale training approach [28].
- Developing different model parameters for each level of an image pyramid, and using the overall features computed to develop detections, as in the Deformable Parts Model [35].

In the context of deep learning, training networks end-to-end on an image pyramid or a fixed set of resolutions is inefficient due to memory constraints for the former and increased training time for the latter. Since the structure of a CNN employs a natural feature hierarchy layer by layer, in which feature maps of different spatial resolutions and feature channels are produced, there exists an *in-network* pyramid that may be exploited. However, as argued by Lin et al. [13], high-resolution maps have low-level features with low semantic value which may harm their contribution to the object detection objective. As a result, Feature Pyramid Networks [13] aim to exploit the inherent multi-scale hierarchy of CNNs with lateral connections, addressing efficiency and performance concerns.

FPNs extend the standard CNN backbone for tasks such as object detection and instance segmentation by outputting feature maps of various spatial resolutions, each with the same number of output channels. The primary contribution in this thesis of FPNs to a depth-aware architecture for object detection is the ability to use two separate backbones for RGB images and depth/HHA images, and to *combine feature maps obtained from similar layers of both backbones* to from a unified architecture where these maps interact. This fusion of the natural hierarchy of features maps inherent to both backbones allows for the spatial and semantic information accumulated at each level to influence one another. Ultimately, the fused feature maps can be input to the same RPN heads and RoI heads from the Faster R-CNN architecture, allowing for a multi-modal Faster R-CNN FPN. The following description of FPNs will be specific to the ResNet architecture, as is the original description provided by Lin et al. [13].

Using a ResNet network, consider the feature maps output by the backbone belonging to $C := \{C_2, C_3, C_4, C_5\}$, where C_1 is not included due to its large memory footprint. Using the notation of Section 2.3, $C_1 = \text{Conv1}$, $C_2 = \text{Conv2}$ etc. The output of the FPN will be a set of feature maps $\{P_2, P_3, P_4, P_5\}$ with the same spatial resolution as the maps in C for each i , however each P_i will have the same number of channels, usually set to 256. Each feature map from C undergoes a 1×1 convolution to reduce its channel dimension to some fixed value, equal for all feature maps (256 in the original implementation), labeled C'_i . Set $P_5 := C'_5$. For each map P_i where $2 \leq i \leq 4$, P_i consists of the element-wise addition of C'_i with an upsampled map (by $2x$) C'_{i+1} , followed by a 3×3 convolution of stride 1 and same padding for anti-aliasing, shown in Figure 2.15. In some implementations, an additional P_6 is included consisting of a stride-two subsampling of P_5 , known as the *Pool* feature map.

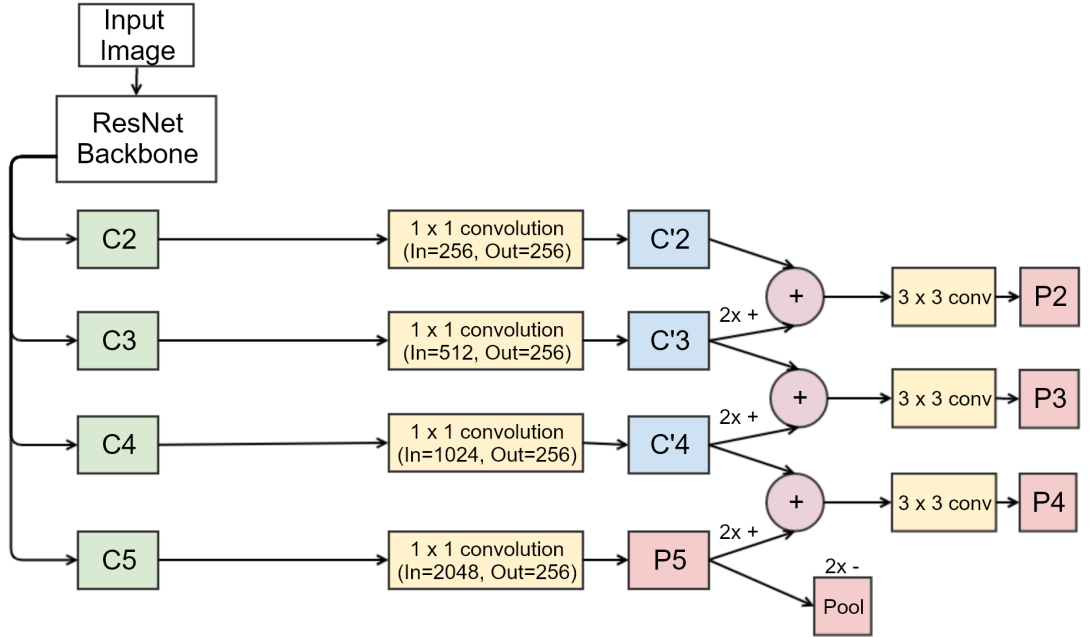


Figure 2.15: A feature pyramid based on a ResNet backbone. $2x -$ denotes a downsampling by a factor of 2 in both the height and width dimensions, $2x +$ denotes upsampling by a factor of 2.

FPNs naturally extend the original Faster R-CNN model by using a single RPN that operates on each feature map, with different anchor areas used for different resolution sizes. For anchors with areas $\{32^2, 64^2, 128^2, 256^2, 512^2\}$, anchors are computed on $\{P_2, P_3, P_4, P_5, P_6\}$ with aspect ratios from $\{0.5, 1, 1.5\}$. As in Faster R-CNN, anchors with an IoU over 0.7 with a ground truth box are assigned a

positive label, and anchors with a max IoU of 0.3 with any ground truth box are given a negative label. For training the RPN, a sample of positive and negative labels among the anchor boxes is chosen for computation of objectness and regression loss. The parameters of the RPN are shared across all feature map levels. The scales of the ground-truth boxes are not explicitly used for assignment to the levels of the pyramid; rather, ground-truth boxes are associated with anchors, which have been assigned to pyramid levels. Viewing the feature pyramid as if it were produced by an image pyramid, the region of interest (RoI) projection from Fast R-CNN [28] is applied to a feature map resolution based on its size. Consider an RoI of width w and height h on the input image, then the assignment to P_k is done by computing the projection level

$$k = \lfloor k_0 + \log_2(\sqrt{wh}/224) \rfloor \quad (2.29)$$

where k_0 is the target level to which an RoI with shape $(w, h) = (224, 224)$, the canonical ImageNet crop size, should be mapped to. The value of k is set to 4 for the ResNet architecture. For each RoI mapped to a level k depending on its size, RoI pooling is used to extract a fixed size from P_k to a tensor of shape $(7, 7, 256)$, where the size of the pooled tensor is a hyperparameter, and 256 is the number of channels for any P_i . This tensor is then input to the standard Faster R-CNN network heads for classification and regression. Indeed, Feature Pyramid Networks extend Faster R-CNN nicely.

2.6 Instance Segmentation with Mask R-CNN

Mask R-CNN [17] is a popular instance segmentation architecture that segments the bounding boxes predicted by the output of Faster R-CNN by adding a parallel *mask-branch*, running alongside the classification and bounding box regression branches. Motivated by the success of fully-convolutional networks for semantic segmentation [36], in which preserving the spatial resolution of the feature maps is key to the success of dense prediction tasks, as well as the success of the Faster R-CNN object detection architecture, the goal of Mask R-CNN is to combine ideas from these two approaches. Despite being introduced in 2017, the Mask R-CNN framework is still a top-performing approach, where the winning entry of the MS COCO 2020 instance segmentation competition MegDetV2 [37] is entirely based off of the Mask R-CNN FPN architecture, adding modifications such as a separate backbone for segmentation and different training and testing protocols. Hence the depth-aware instance segmentation model proposed in Section 4.2.2 will use a

FPN, fusing inputs from separate ResNet backbones trained on RGB and HHA images, in order to develop an RGBD Mask R-CNN FPN model.

Following the approach of Faster R-CNN, in Mask R-CNN a region proposal network is trained. The predicted proposals are projected onto a feature map (or feature map level in the case of the FPN architecture), from which the RoI pooling operation extracts a fixed size tensor to be fed into RoI network heads. These RoI heads predict the class of a set of regions (those regions with high confidence values), and compute bounding box regression to more accurately localize the detection coordinates.

To perform instance segmentation, a mask branch consisting of a fully convolutional network takes as input the RoI pooled tensor, now of size 14×14 as opposed to 7×7 in Faster R-CNN [12]. The fully convolutional mask branch outputs a set of $C \cdot m^2$ segmentation masks, for C classes. Each mask is the output of a sigmoid layer of fixed spatial resolution $m \times m$, and so the result of the mask branch is a tensor of shape (m, m, C) , where m is commonly set to 28. In the PyTorch implementation [38], the ground truth instance masks are first projected to the RoI, after which bilinear interpolation is performed to resize the projection within the RoI to an $m \times m$ instance mask, which is binarized. The mask loss for RoIs corresponding to region proposals with an IoU > 0.5 with some ground truth bounding box, defined on the k -th class, consists of the average binary cross-entropy loss:

$$L_{mask} = \frac{1}{m^2} \sum_{(x,y)} [-\log(M(x,y))G(x,y) - \log(M(x,y)(1 - G(x,y)))] \quad (2.30)$$

where G is the ground truth instance mask projected to the RoI and resized, and M is the predicted probability of a mask pixel in the $m \times m$ mask belonging to an instance, with a value of 1 signifying that it does belong to an instance, and 0 that it does not. The mask loss is only applied to foreground RoIs and only for the k 'th mask, given a ground truth instance from category k . For approximate joint training, where the RPN is trained along with the RoI heads and Mask head, the overall loss is computed as

$$L = L_{RPN_{obj}} + L_{RPN_{reg}} + L_{cls} + L_{reg} + L_{mask} \quad (2.31)$$

where the Faster R-CNN losses $L_{RPN_{obj}}, L_{RPN_{reg}}, L_{cls}, L_{reg}$ are defined in Section 2.4, and L_{Mask} is given by equation 2.30.

During inference, the $m \times m$ instance mask is resized to the size of the regressed RoI using bilinear interpolation, after which a thresholding operation of 0.5 is used to binarize the instance mask, for non-background RoIs. Hence the accuracy of the bounding box regression of the region proposals is paramount for a successful instance mask prediction, since pixels outside the bounding boxes will always receive a prediction value of 0.

A key idea introduced within the work of Mask R-CNN is the *RoI Align* operation, which addresses the quantization inherent in two operations in the Faster R-CNN framework:

- The quantization of image coordinates of RoIs projected to feature map coordinates, depending on the network stride.
- The quantization of the grid computation to divide the projected RoI into a fixed number of cells for the RoI pooling operation.

In order to address both these quantizations, which are not desirable in a segmentation context where spatial details are very important, the authors of Mask R-CNN propose to use bilinear interpolation. In particular, given a network stride s , and a region proposal $[x_1, y_1, x_2, y_2]$, the RoI projection coordinates without quantization are $[x_1/s, y_1/s, x_2/s, y_2/s]$. Then for some fixed grid size, say $h \times w$, the projected region proposal is divided into cells of size $(\frac{y_2 - y_1 + 1}{h} \times \frac{x_2 - x_1 + 1}{w})$. Within each cell, a certain number of *sampled points* are chosen, whose values are computed with bilinear interpolation from the nearest 4 feature map spatial locations for each channel. These values are then averaged, and as a result for C input channels to the RoI Align layer, a tensor of shape (h, w, C) is output, ready for the Mask R-CNN network heads. RoI Align is shown to improve both instance segmentation performance and object detection performance when compared to RoI pooling [17].

For region proposals that have been projected to the corresponding FPN feature map output level according to Equation 2.29, and pooled with the RoI Align operation, a tensor of size $(14, 14, 256)$ is fed into a fully convolutional mask branch. Here the ResNet FPN variant of the Mask R-CNN mask branch is described. The mask branch consists first of four 3×3 convolutional layers with ReLU activation with input channels equal to output channels, set to 256. The output of these layers is then fed into a transposed convolution layer of stride 2 with kernel size 2×2 in order to upsample the tensor to the shape $(28, 28, 256)$.

Next a 1×1 convolution maps this tensor to a reduced dimension of 80 output channels with the same spatial resolution, which is finally convolved to the desired output tensor of shape $(28, 28, C)$, for C classes, using a 1×1 convolution, see Figure 2.16. The output tensor is then input to a sigmoid layer to compute probability vectors for each pixel over the channel index.

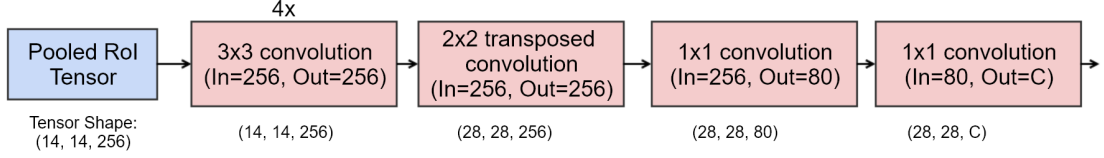


Figure 2.16: The mask branch of Mask R-CNN, to be followed by a sigmoid layer and an upsampling operation.

The challenge of instance segmentation is to clearly outline the boundaries of objects, as well as their inner details. This requires pixel level information. To be explored in the evaluation of the depth-aware Mask R-CNN FPN is whether the features provided by depth images are fine enough to offer complementary information to be fused with RGB features for improved model performance.

2.7 Semantic Segmentation with DeepLabv3

There are two fundamental challenges inherent to using convolutional neural networks for semantic segmentation as noted by Chen et al. [18], namely that:

- The typical architecture for deep convolutional neural networks, such as ResNets or VGG models, reduce the spatial resolution of the feature maps, where fine-grained resolution is crucial for dense pixel-level prediction tasks.
- Segmenting objects at multiple scales without resorting to multi-scale training is difficult.

The *DeepLab* architecture aims to address these issues, where the focus of this section concerns version 3 [18]. Applied to a ResNet-101 backbone, the details of the PyTorch implementation [19] will be examined, since this will be the architecture to be modified for a depth-aware semantic segmentation model proposed in Section 4.2.3.

There are two key components to the DeepLabv3 architecture. First, the use of dilated convolutions and restricted use of downsampling in the backbone. To be concrete, for each 3×3 convolution in blocks Conv4 and Conv5, the dilation factor

is increased from 1 to 2, and the padding is increased from 1 to 2. By changing all convolution strides to be 1 in blocks Conv4 and Conv5, no downsampling is performed, thereby changing the network stride to be 8 as opposed to 32 previously. As a result, for an 800×800 input image, the backbone output's resolution will be of size 100×100 , as opposed to 25×25 for the standard ResNet-101 model, a major increase in resolution.

The second key component of the DeepLabv3 model is the *atrous spatial pooling pyramid*. What follows is a description of the DeepLabv3 network head. The output of the backbone is a tensor of shape $(N, 2048, H, W)$ for $H = \lfloor I_h/8 \rfloor$, and $W = \lfloor I_w/8 \rfloor$ where the original image resolution is (I_h, I_w) , using a batch size of N images. This tensor first undergoes a 1×1 convolution for dimension reduction to 256 output channels, followed by batch normalization and ReLU activation. Label this output tensor F of shape $(N, 256, H, W)$.

Then F is fed through an *atrous spatial pooling pyramid* (ASPP) where 3 parallel dilated 3×3 convolutions with respective dilation factors (12, 14, 36) and padding (12, 24, 36) are applied, resulting in three tensors of shape $(N, 256, H, W)$. The purpose of the pyramid of atrous convolutions is to incorporate context at each pixel from a large window around it, in order to capture multi-scale information without using large image filters or decimating the resulting spatial resolution.

As Chen et al. [18] observe, when the dilation factor of a convolutional filter is too large, the filter degenerates to a simple 1×1 filter. As an alternative, in order to get global context information in a manner similar to a very dilated convolution, a global pooling operation is computed for each channel by averaging the entire spatial resolution of F , resulting in a tensor of shape $(N, 256, 1, 1)$ which is then upsampled using bilinear interpolation to a tensor of shape $(N, 256, H, W)$. The three tensors output from the dilated convolution operations along with the global pooled tensor are concatenated to form a tensor T of shape $(N, 4 * 256, H, W)$, the output of the pyramid. See Figure 2.17 for a visualization of the ASPP.

The tensor T is then reduced channel-wise to 256 channels with a 1×1 convolution, followed by batch normalization, ReLU, and a dropout layer⁷. Finally, a 3×3 convolution followed by batch normalization and ReLU is performed, followed by a 1×1 convolution to map the tensor of shape $(N, 256, H, W)$ to the

⁷Dropout layers zero the input at any tensor component with probability p , where usually $p = 0.5$.

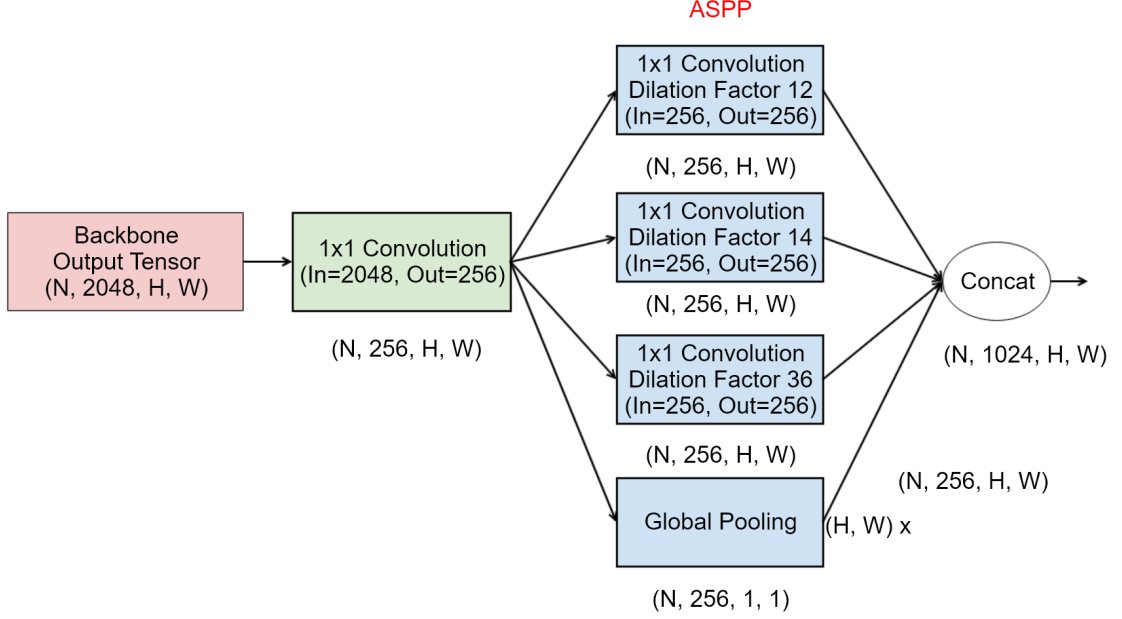


Figure 2.17: The pyramid of atrous convolutions of the DeepLabv3 head. Dilated convolutions and the global pooling operation are displayed in blue, with the resulting tensor shape of each operation placed below. $(H, W) \times$ signifies bilinear upsampling at each channel after the global pooling operation to a spatial resolution of (H, W) .

shape (N, C, H, W) for C classes. The output of the DeepLabv3 head is then upsampled to the original image size, followed by a softmax operation computed across channels at each pixel location, where a negative log-loss is computed during training. This loss is typically averaged over the resolution of the training images. During inference, the predicted class label of each pixel is obtained by taking the maximum value of the tensor across channels.

In addition to its strong performance, the choice for DeepLabv3 as the semantic segmentation model to be modified is based on a few observations:

- There are prominent works in the literature that use RGBD inputs to earlier versions of modified DeepLab models, but not the third version and so there is a degree of novelty to this usage.
- The atrous spatial pooling pyramid will act on fused features extracted from ResNet-101 backbones trained on RGB and HHA/depth inputs. This pooling of hybrid features from various pixel distances is conjectured in this thesis to have the benefit of *context from depth features at multiple pixel ranges*. HHA features that may correspond to objectness edges, surface normal edges, or height changes from objects and surfaces at different long

distance ranges from a pixel can now inform the predicted semantic labels predicted by the model. This information is then combined with features provided by the color image, which also incorporate multi-scale context.

- The DeepLabv3 model does not make use of post-processing techniques like conditional random fields used in DeepLabv1 [39], for a fully end-to-end solution both in terms of training and inference.

Chapter 3: Literature Review

In this chapter, prominent works related to deep learning approaches to RGBD object detection as well as RGBD semantic segmentation are presented in chronological order. The focus of each model described will be how the depth image features are combined with the RGB features to obtain the model output. Each section is followed by a discussion of what techniques/ideas remain to be explored, and how this thesis will incorporate such ideas.

3.1 Deep Learning and RGBD Object Detection

The RGBD R-CNN model presented by Gupta et al. [7] introduces a framework for scene understanding, tackling the tasks of contour detection, object detection, instance segmentation, and semantic segmentation with a hierarchical approach, for which an overview is shown in Figure 3.1.

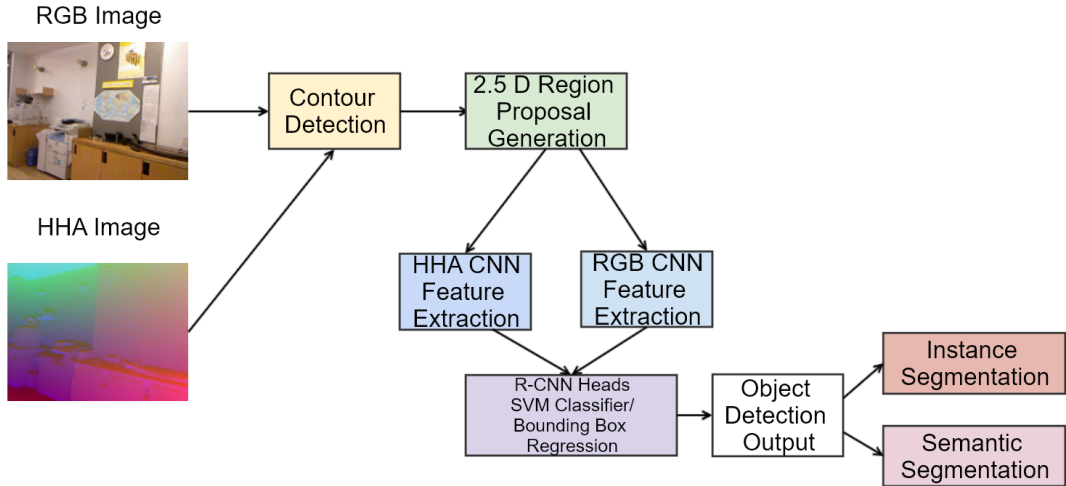


Figure 3.1: A simplified overview of the approach of RGBD R-CNN. Example RGB/HHA image pair taken from the SUN RGBD dataset [8].

For the objective of contour detection, the method of Structured Random Forests [40] is used to obtain a contour map using RGB and depth features engineered by the previous work of Gupta et al. [22]. From these contours, a hierarchy of regions is obtained using the OWT-UCM (Oriented Watershed Transform followed by the Ultrametric Contour Map) method presented by Arbelaez et al. [41]. These regions are used to predict region proposals. Summarized, the approach consists of two steps. First, using the contour map, an oriented watershed transform is

computed yielding an oversegmentation of the image. Secondly, a hierarchical representation of regions merged pairwise from the oversegmented image is obtained. Adjacent regions are merged greedily based on the strength of the contours that separate them (from weak to strong), until no contours remain. From this algorithm a hierarchical region tree is constructed. RGB and depth features computed on tuples of regions from this hierarchy are then input to a random forest classifier to regress an objectness prediction using the Multiscale Combinatorial Grouping approach [42]. Tuples of regions with sufficiently high objectness scores are fit with tightly enclosing bounding boxes. The splitting features used by the random forest regressor derive from both RGB and depth features, and so the resulting region proposals are referred to as *2.5D region proposals*.

For the task of object detection, the R-CNN framework is used [27]. The best reported results stem from a fusion approach from two backbones, both the AlexNet backbone [9], trained with the same fixed 2.5D region proposals. As in R-CNN, the region proposals are warped to a fixed size before being input to the backbone. The first backbone uses the crop of the RGB image, while the second uses the same crop of the HHA-encoded depth image. Each backbone is trained separately with its own network head using the output of layer FC6, the first fully connected layer of AlexNet. Both backbones are initialized with the weights from pre-trained models trained on the ImageNet dataset, and then fine-tuned for object detection on the NYUDv2 dataset [6] with their respective heads. Recall that the network heads of the R-CNN architecture consist of a support vector machine (SVM) for classification, and a bounding box regression layer. Once the backbones have been trained individually, their respective layers are frozen. Using the concatenated output of both backbones, a new R-CNN head is trained on the NYUDv2 dataset. This approach outperforms the equivalent RGB classification framework by 12.8 percentage points using the mAP_{50} metric. Interestingly, the authors report a higher mAP_{50} on a CNN trained with only HHA images than a pure RGB counterpart, a result that will not be reproduced in this work.

Instance segmentation is formulated as a two-class labeling problem on the pixels within each detection window. The goal is to associate a binary pixel mask to each detection produced. During training, each ground truth instance mask is associated with the best object detection bounding box that overlaps the instance mask’s associated bounding box by more than 70%, if such a detection exists. Hand-crafted features from both RGB and depth images are computed

for the pixels within the bounding box, and this *feature crop* is then warped to a 50×50 sized grid, where each location is input to a random forest classifier which outputs the probability that a pixel belongs to an instance. During testing, the top 5000 (or 10000 for the chair class) detections are considered, before confidence value thresholding or non-maximal suppression. For each of these detections, features are computed and warped to obtain a 50×50 confidence map output from the trained random forest. This confidence map is then unwrapped back to the original detection window. These predictions are accumulated over the oversegmented regions obtained in the contour detection stage, referred to as superpixels. Superpixels that have non-empty intersection with an object detection window are thresholded to obtain a final instance mask prediction, allowing the entire superpixel to be considered part of the object mask.

The method for semantic segmentation used expands on Gupta et al.’s previous work [22], where again an oversegmentation is achieved through a watershed transform on the output of a set of oriented contour detectors. Next, a technique referred to as *amodal completion* is employed. One can observe that certain 3D objects when viewed in a 2D image will have parts of their overall structure occluded by other objects, for example an office desk may be occluded by nearby furniture or items on the desk, depending on the viewpoint. The goal of amodal completion is to measure the similarity between superpixel regions, and merge both short-range and long-range regions based on this similarity measure. In this manner an object that is split due to occlusions can be united as a merged set of regions that are not necessarily adjacent in pixel space. The strategy is two-fold:

1. Estimate features for individual superpixels obtained from the hierarchy.
2. Greedily merge superpixels into larger overall regions based on the agreement of the features.

The output of this greedy merging is a set of non-overlapping regions which consists of both long and short range completions of the base superpixels. Superpixels and their amodal completions form a more coarse segmentation of the image, where hand-crafted RGB and depth features computed are extracted from the regions. In addition to these features, category specific features from the object detection stage are considered. Given the detections from the object detection algorithm, the additional features added to each superpixel and its amodal completion consist of the number of overlaps with a feature of a given category, and the accumulated IoU of the overlaps. Using these features, a support vector

machine is trained to classify each superpixel along with its amodal completion.

For each task considered in RGBD R-CNN, machine learning approaches employing RGBD images outperform the comparable RGB counterpart, providing one of the most comprehensive arguments in the literature for the potential of depth-aware methods to bolster computer vision performance.

Following the development of RGBD R-CNN, Gupta et al. [43] investigate techniques to address the discrepancy between large labeled RGB datasets versus smaller labeled datasets for RGBD images. For example, the ImageNet dataset [10], for which many CNN backbones are pre-trained on, consists of over a million labeled images. This is several orders of magnitude larger than any existing RGBD dataset. In order to mitigate this issue the scheme of *supervision transfer* is proposed.

In this approach, for a modality M_d with unlabeled data D_d , the goal is to train a representation of such data by transferring information from a separate modality M_s which has a large labeled set of images D_s and a corresponding K -layered rich representation. It is assumed that there is a relatively large amount of unlabeled pairs (I_s, I_d) . In the context of convolutional neural networks, supervision transfer uses two backbones, which may be different architectures. The RGB backbone is pre-trained either for RGB image classification or RGB object detection, and the depth network is randomly initialized. The backbones considered in this work are the AlexNet and VGG models [9, 33].

Consider a layer number L' of the untrained backbone, and a layer number of the pre-trained backbone L . For image pairs (I_{RGB}, I_{Depth}) the image I_{RGB} is fed through the pre-trained network until being output from layer L to get a tensor I_{RGB}^L , and the depth image is fed through the untrained network to get the output of layer L' as $I_{Depth}^{L'}$. The training loss for the depth network consists of the L_2 norm $\|I_{RGB}^L - I_{Depth}^{L'}\|_2$. For transfer between layers that do not have the same number of channels, an appropriate transformation function between layer outputs is computed (i.e. channel reduction or zero padding for increasing the number of channels). See Figure 3.2 for an overview of this approach.

Supervision transfer from a VGG backbone to an AlexNet backbone is explored where the RGB network is pre-trained either on the ImageNet classification dataset or the MS-COCO object detection dataset [23]. An AlexNet architecture for depth images is then trained with supervision transfer. The unlabeled data

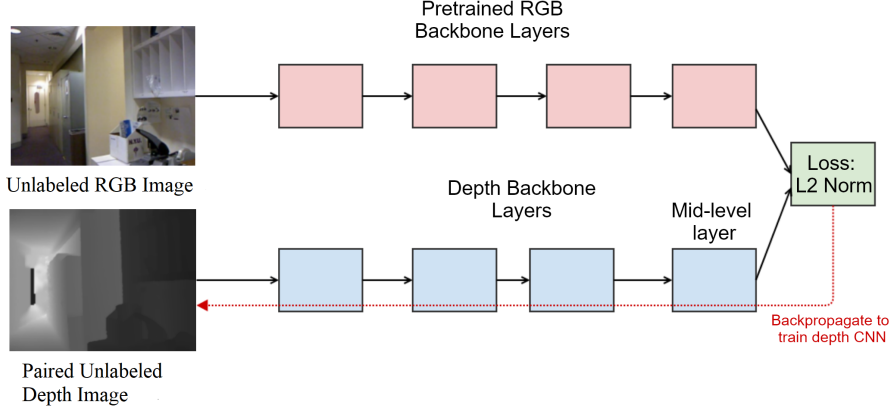


Figure 3.2: The supervision transfer approach of Gupta et al. [43] with an unlabeled RGB/depth image pair from the NYUDv2 dataset [6].

pairs come from the NYUDv2 dataset, of which 10K frame pairs are used⁸.

Next, the VGG and AlexNet backbones are trained with the NYUDv2 training data, with the former being trained on the RGB images, and latter on the depth images. Using these backbones, two networks are trained for object detection using the same region proposals computed from RGBD R-CNN [7], using the approach of Fast R-CNN [28]. During inference, each region proposal’s multi-modal output is computed by averaging class scores from the softmax layer of the Fast R-CNN heads of both modalities. Similarly, the region proposal bounding box regression values are obtained by averaging the regression values from both network heads. Tested on the 19-class object detection benchmark on the NYUDv2 dataset, this approach yields a mAP_{50} of 49.1%, up from 37.2% in RGBD R-CNN [7]. Instance segmentation is also revisited using the approach of RGBD R-CNN, but with objects detected using supervision transfer, improving mAP_{50} from 32.1% to 42.1%.

Xu. et al. [44] extend the Faster R-CNN architecture with a depth backbone and a third CNN structure referred to as the *Correlated Detection Net*. The backbones used for RGB and depth/HHA images are both the AlexNet architecture, with the RGB backbone initialized from ImageNet pre-training, and the depth backbone initialized using the supervision transfer method described by Gupta et al. [43]. The third backbone used for correlated features takes as input the output of Conv2 from both the depth and RGB backbones, and inputs both tensors through Conv3 to Conv5 layers in parallel, with layers Conv3 to Conv5 taken from the AlexNet architecture (see Figure 3.3). The RGB and depth tensors

⁸The NYUDv2 dataset has both labeled and unlabeled RGB/depth frame pairs.

output from Conv5 are then fused in a *correlation* operation, which consists of the Hadamard product (element-wise multiplication), followed by an element-wise square root operation.

$$h_{corr} = \sqrt{RGB_{conv5} \circ Depth_{conv5}} \quad (3.1)$$

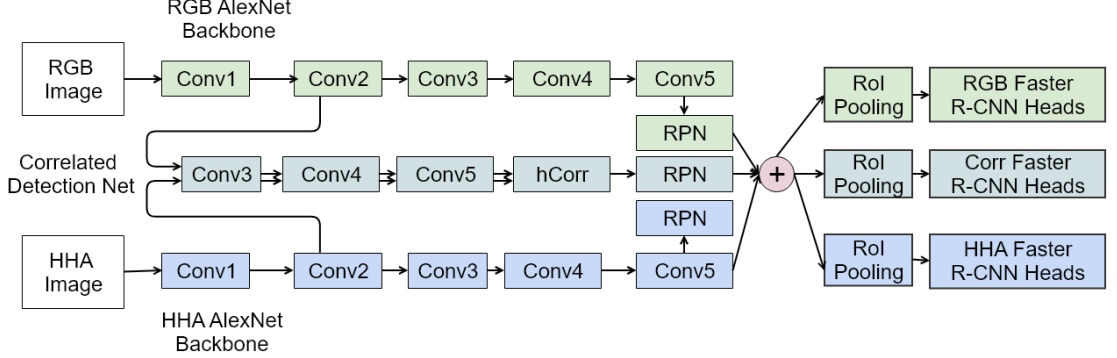


Figure 3.3: The three backbone RGBD Faster R-CNN approach of Xu et al. [44]

Each backbone has its own Region Proposal Network (RPN), and separate RoI heads. For region proposals, the overall anchor box regression and objectness scores are computed as a weighted linear combination of the outputs of each RPN. These region proposals are then used to train the RoI heads of each backbone, and at inference time, for each region proposal the class prediction and regression values are obtained as a weighted linear combination of the predictions of each backbone RoI head. Tested on the NYUDv2 19-class object dataset benchmark, this approach slightly outperforms RGBD R-CNN using supervision transfer, achieving 49.5% mAP_{50} .

Li et al. [20] explore the use of Long Short Term Memory (LSTM) modules [45] and Spatial Transformer Networks (STN) [46] to improve performance on RGBD object detection. This approach is referred to as the cross-modal attentional context algorithm (CMAC). For each RGBD image, the region proposal generation approach of RGB R-CNN [7] is used to pre-compute fixed proposals. Using a VGG-16 model for the RGB image, and an AlexNet architecture for the HHA encoded depth image, two fixed sized tensors of shape (K, K, D) are obtained from each backbone. They are concatenated, and then reduced with a 1×1 convolution to obtain a *global feature tensor* F of shape (K, K, D) . For each region proposal, a fixed tensor of size (S, S, D) is obtained from each backbone

using RoI pooling [28], where each RoI pooled tensor is concatenated and reduced dimension-wise, obtaining a *local feature tensor* of shape (S, S, D) . For classifying the region proposals, an LSTM takes in as input:

- A global context feature vector x_t of shape $(1, D)$, initialized by applying global pooling to the global feature tensor.
- A local feature vector z obtained by applying global pooling to the local feature tensor.
- A hidden state h_t .

At each time step the LSTM learns a weight map α_t of shape (K, K) and computes $x_t = \sum_{(x,y)} \alpha_t(x, y) F(x, y)$ where α_t is used as a filter of size $K \times K$ on each channel of the global feature tensor to obtain x_t , the global context feature vector at time t of shape $(1, D)$. The overall output of the LSTM model is a D -dimensional global context feature vector x_{T+1} , which is further fed in to two fully connected layers to produce feature F_G used for classification. Li et al. refer to this approach as *attention-based global context modeling*, where the idea is to use both the local and global context of a region proposal to aid in classification, compared to a more localized feature extraction used in RoI pooling.

The second key component of the architecture of CMAC is the so-called *fine-grained object parts attention mechanism*. For bounding box regression, a set of spatial transformers are employed to learn discriminative object parts for each region proposal. A spatial transformer mechanism consists of three parts. First a localization network takes the input feature map, and applies a multilayer perceptron that outputs the parameters of the spatial transformation that will be applied to the feature map. The predicted transformation parameters are used to create a sampling grid on the input map using a grid generator. The feature map and the sampling grid are taken as inputs to the sampler, producing the output map sampled from the input at the grid points.

A set of N spatial transformers are used to transform the local feature vector into N tensors of shape $(7, 7, D)$, which are concatenated, and then reduced channel-wise with a 1×1 convolution to produce a tensor of shape $(7, 7, D)$. This tensor is then fed through two fully connected layers to obtain F_L . F_G is concatenated to F_L to be input to a fully connected softmax layer for classification, while F_L is fed into one more fully-connected layer for bounding box regression. Overall,

CMAC achieves 52.3% mAP_{50} on the NYUDv2 dataset, and 47.5% mAP_{50} on the SUN RGBD dataset, both using the 19-class benchmark defined by Gupta et al. [7], outperforming previous works.

Ophoff et al. [47] explore real-time RGBD object detection, modifying an RGB real-time object detection architecture to input RGBD images. Although the Faster R-CNN framework has very strong performance, its inference time is relatively slow, making it impractical for use in real-time applications. Hence a series of *single shot* methods such as YOLO (You Only Look Once) [48] and SSD [49] (the Single Shot MultiBox Detector) emerged in the object detection literature, attempting to increase the frame rate at inference time, while still maintaining good performance. In general, these works re-frame the object detection problem as a regression problem and dispense with the region proposal stage of the detection pipeline.

Ophoff et al. modify the YOLOv2 architecture [50] to make use of two sub-networks, each of which consist of the first N layers of YOLOv2, of which there are 27 layers in total. For an RGBD object detection dataset, an RGB YOLOv2 and depth YOLOv2 are trained, both using weights initialized from convolutional backbones pre-trained on ImageNet, up to layer 23 of the overall YOLOv2 architecture. The remaining layers in both sub-networks are randomly initialized.

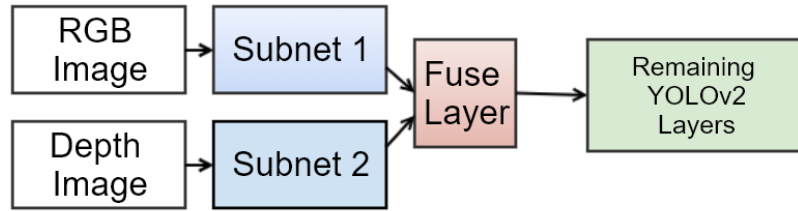


Figure 3.4: An overview of the real-time RGBD object detection method of Ophoff et al. [47].

After both the RGB and depth images are fed through their respective sub-networks, a fusion layer consisting of a concatenation operation followed by a 1×1 convolution for dimension reduction outputs the fused tensor, which is then fed into the remaining $27-N$ layers of the YOLOv2 model, see Figure 3.4. Since there are 27 layers, there are 28 different possibilities to choose which layer to fuse the information from both sub-networks, depending on which layer is chosen for the

fusion (including early fusion before network input, and late fusion at network output). For training the dual-stream fusion network, RGB and depth sub-networks are initialized with the RGB and depth pre-trained backbone weights layers until either the fusion layer or the 23rd layer, whichever comes first. If the fusion layer occurs before layer 23, then the layers in between the fusion layer and layer 23 are initialized with the RGB pre-trained weights. Otherwise the layers after 23, including after the fusion layer, are randomly initialized.

Brute force testing is used to find the optimal layer for fusion, depending on the results of tests on multiple datasets. For each dataset, the depth infused two-stream approach improves performance over the pure RGB YOLOv2 counterpart. The evaluation datasets chosen depart from the typical focus on the NYUDv2 and SUN RGBD object detection benchmarks. In particular, the datasets tested on were:

- The EPFL Pedestrian Dataset [citeEPFL](#) (Ecole Polytechnique Federale de Lausanne, Switzerland) which consists of around 5000 RGBD images of persons captured with a Kinect v2.
- The KITTI dataset [52], which consists of around 7500 stereo RGB images, captured from a car driving around a city, with annotated boxes for pedestrians, cyclists, and cars. This dataset is now one of the canonical datasets for object detection models engineered to be used with autonomous vehicles.
- A smaller dataset from an industrial context known as the GD Screws dataset [53], provided by the Life Cycle Engineering lab of KU Leuven. The depth images in this dataset are acquired using a sheet-of-light scanner, providing highly accurate depth maps, where the goal is to detect small screws located on pieces of electronics.

Overall, despite minor differences in the optimal fusion layers observed for each dataset tested, mid-level fusion around layer 14 demonstrated the best performance.

In light of the review of the RGBD object detection literature presented in this section, some themes emerge for unexplored avenues that will be exploited in this work:

- The use of a dual backbone multi-modal Feature Pyramid Network to take advantage of fusion operations between multiple backbone layers. In each

work described in this section, fusion occurs with *a single feature map* from both modalities.

- The extension of a depth-aware Faster R-CNN that combines features from the RGB and depth images with a single region proposal network and single set of network heads. This would contrast to the three RPNs and network heads used by Xu et al. [44]. This approach would also differ from the Fast R-CNN approach of Gupta et al. [43], which averages classification and bounding box regression values for pre-computed region proposals from the RoI heads for each backbone.
- More exploration of different learned approaches to combine the features from the RGB and depth backbones, rather than simply concatenating the features as in RGB R-CNN [7], or multiplying them element-wise as in the Correlated Detection Net [44].
- The use of more recent CNN backbones such as ResNets [14] for more powerful feature representations from both the RGB and depth/HHA images.

3.2 Deep Learning and RGBD Semantic Segmentation

Much of the published research utilizing RGBD images in computer vision focuses on semantic segmentation. Couprie et al. [54] employed one of the earliest semantic segmentation architectures for RGBD images based on convolutional neural networks. The RGB and depth images are treated as separate images, but the outputs of the network on both images are used to predict a segmentation mask. With each image, a Laplacian Pyramid P_l is computed, which consists of the difference images of adjacent levels in the well-known Gaussian pyramid structure G_l , i.e

$$P_l(x, y) = \max(0, G_l(x, y) - G'_{l+1}(x, y)) \quad (3.2)$$

where G'_{l+1} is the upsampled image at the layer $l + 1$ of the Gaussian pyramid. It is upsampled to match the spatial resolution of G_l , using bilinear interpolation. For a Gaussian pyramid of l layers, difference images are computed for the first $l - 1$ levels, yielding the Laplacian pyramid of $l - 1$ levels. The Laplacian pyramid is used to obtain multi-scale predictions.

Laplacian pyramids are obtained from both the RGB and depth images. Each image in each pyramid is fed through the convolution neural network described by Farabet et al. [55] to obtain a set of tensors of different spatial resolutions. These

tensors are upsampled with bilinear interpolation to match the finest resolution output by the CNN on the Laplacian pyramid. These feature maps are then input into a pixel-wise classifier consisting of 2 fully connected layers of input size 1024, and output consisting of a class label prediction for each spatial location in the input tensors. In parallel, the input RGB image is segmented into class-agnostic regions using a graph-based image segmentation approach presented by Felzenszwalb and Huttenlocher [56], decomposing the image into superpixels, used to smooth the output segmentation. The final predicted labeling is obtained by an aggregation of the prediction tensors across superpixels.

Eigen and Fergus [57] utilize a multi-scale convolutional neural network for three separate tasks: depth prediction, surface normal estimation, and semantic segmentation of RGBD images. The same general network structure is employed for each task, but a different loss function and input type are used for training. For semantic segmentation, RGBD images along with estimated surface normals are input to the network, which consists of three stages. Each stage has a corresponding output scale, which increases per stage. The first stage uses a standard AlexNet or VGG-16 backbone pre-trained on ImageNet, followed by two fully connected layers with an output of 64 channels and a spatial resolution $\frac{1}{16}$ the size of the original image. This is then followed by a bilinear interpolation operation to increase the spatial resolution to $\frac{1}{4}$ times the size of the input image. The first stage only takes the RGB image (or image batch) as input. The second stage has two inputs, which are concatenated:

- The output of the first stage.
- The concatenated RGB, depth, and surface normal images⁹, each convolved with their own respective set of resolution-reducing filters.

The second stage consists of a series of convolutions that preserve resolution. The third stage takes as input the upsampled output of the previous stage, concatenated with the RGB, depth, and surface normal images convolved at an even finer stride. This stage consists of a series of resolution preserving convolutions, followed by a softmax layer to be trained with a cross-entropy loss function. See Figure 3.5 for an overview. The output of scale 3 is upsampled to match the original image

⁹Surface normals estimated at each spatial location in a depth image can be treated as a surface normal image where each channel corresponds to one of the x, y, z components of the normal vector, scaled to $[0, 255]$.

resolution to compute the loss. As in most semantic segmentation architectures, the predicted segmentation mask for an image of resolution (H, W) for C semantic labels will be a tensor of shape (H, W, C) , where for each pixel location (x, y) , the semantic label with the highest probability is chosen as the predicted pixel label. This network architecture outperforms the semantic segmentation approach of RGBD R-CNN [7] on the 40-class NYUDv2 benchmark introduced by Gupta et al. [22], achieving 34.1% mean IoU.

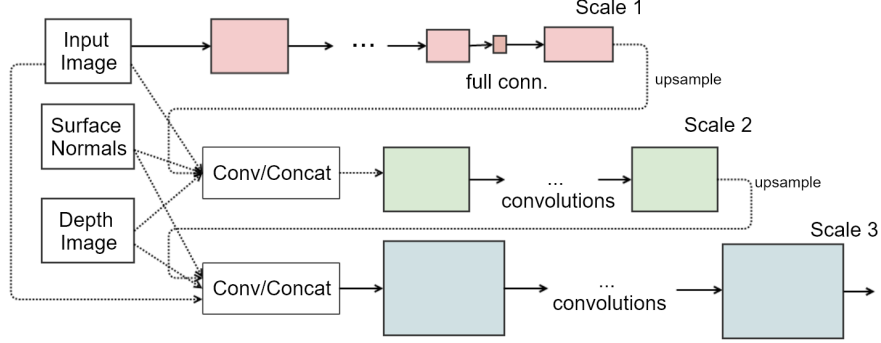


Figure 3.5: An overview of the architecture of Eigen and Fergus [57].

In one of the most cited works in the RGB semantic segmentation literature, Long et al. build a fully convolutional network (FCN) [36] by modifying ImageNet classification architectures, and show impressive performance and efficiency gains on the Pascal VOC 2011-2012, NYUDv2, and the Sift Flow datasets. A VGG-16 [33] network is modified by converting its fully connected layers to convolutional layers. Since VGG-16 has a network stride of 32, the output tensor before the fully connected layers, i.e. layer Pool 5, on the canonical ImageNet input shape $(244, 244, 3)$ will be a tensor of shape $(7, 7, 512)$. The fully connected layers are replaced by 1×1 convolutions, resulting in a tensor of shape $(7, 7, C)$, for C classes. Using transposed convolution, this tensor is upsampled $32x$ to the original resolution of the input image, and a softmax operation is computed at each spatial location. A per-pixel cross-entropy loss is computed, averaged over the image resolution size.

In order to refine the precision of the output, two additional variations of the FCN exist, as shown in Figure 3.6. The variations of the FCN architecture are:

- FCN-32s, the network already discussed.
- FCN-16s. In this variation, the network output of FCN-32 is upsampled

by a factor of 2 in each dimension, while the output of Pool4 is reduced channel-wise to the number classes, and the results are summed, before a 16x upsampling is computed to match the input image resolution.

- FCN-8s. Similar to FCN-16, the network output of FCN-16 is upscaled by a factor of 2 in each dimension, while the output of Pool3 is reduced channel-wise to the number of classes, after which the results are summed. This result is then upsampled by a factor of 8 to match the input image resolution.

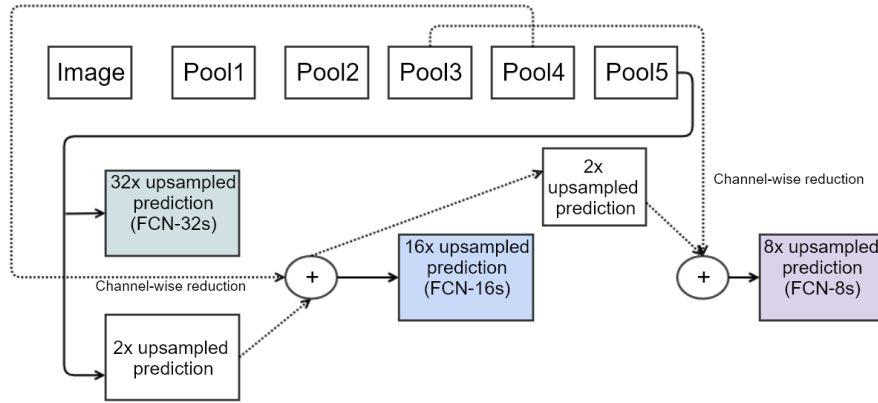


Figure 3.6: The variants of the FCN architecture with a VGG-16 backbone.

Relevant to RGBD semantic segmentation, Long et al. use two FCN-16s trained on RGB and HHA images respectively and use a two-stream approach of a simple addition operation between network predictions. The probabilities per pixel are added for each class in the prediction tensors, where at inference time the highest scoring class per pixel is chosen as the predicted semantic label. This approach yielded a mean IoU of 34.0% on the NYUDv2 40-class benchmark.

Hazirbas et al. develop FuseNet [58], an encoder-decoder architecture for semantic segmentation of RGBD images. Two VGG-16 backbones are utilized to encode the RGBD feature representation, one for the RGB image, and one for the depth image (or HHA encoding). The outputs of similar convolutional layers are fused using element-wise summation, the results of which are input to the next layer of the RGB backbone. In order to decode the RGBD features, the method of DeConvNet [59] is employed which makes use of so-called *unpooling layers*. For a given pooling layer in the RGB backbone, the spatial location of the maximum used in the max pooling of kernel size 2×2 is memorized for each image input at each channel. The decoder network has the same convolution and ReLU operations

as the encoder, with different learned weights, but the pooling layers are replaced with unpooling layers. The unpooling layer maps each spatial location (x, y) at each channel c of the input tensor I to a 4×4 sized square with the memorized location of the maximum element in the corresponding max pooling location set to $I(x, y, c)$, and other elements set to 0. This increases the width and height of the spatial resolution of the tensor by a factor of 2. An example unpooling operation is shown in Figure 3.7, simplified with a $(4, 4)$ input with a single channel.

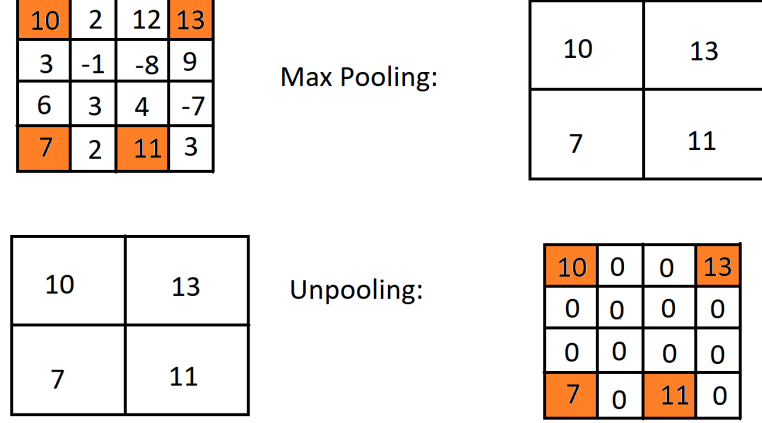


Figure 3.7: An example of an unpooling operation using the memorized maximum locations used in a max pooling operation with kernel size 2×2 and stride tuple $(2, 2)$.

The authors of FuseNet experiment with sparse fusion and dense fusion. In sparse fusion, only the output of the convolutional block before each pooling operation undergoes fusion, whereas in dense fusion, fusion occurs after each convolutional block. A surprising result is the improved performance of the latter, indicating that too much interaction between modalities can degrade performance. FuseNet achieves 37.2% mean IoU on the SUN RGBD dataset 37-class benchmark [8], and was not tested on the NYUDv2 dataset.

Li et al. [60] explore the use of multiple LSTM modules alongside convolutional layers in order to encode short-range and long-range spatial dependencies with their LSTM-CF architecture (Long Short-Term Memorized Context Fusion) for RGBD semantic segmentation. In this approach, the depth image is encoded as an HHA image, and fed into three randomly initialized convolutional layers, each with max pooling, to extract a feature tensor. This tensor is then fed into two LSTMs, which operate vertically on the input, in the upwards and downwards directions. Each LSTM takes in the depth feature map F of shape (w, h, c) , and outputs at each time step a tensor of size (w, h, d) . Denote the c -dimensional

vector at spatial location (i, j) for tensor F as $F(i, j)$. The hidden states are computed for the upward and downward LSTMs as

$$t_{i,j}^u = LSTM(t_{i,j-1}^u, F_{i,j}), \quad j = 2, \dots, h \quad (3.3)$$

and

$$t_{i,j}^d = LSTM(t_{i,j+1}^d, F_{i,j}), \quad j = h - 1, \dots, 1 \quad (3.4)$$

with states $t_{i,1}^u, t_{i,h}^d$ randomly initialized.

The output of both LSTMs on each $(F_{i,j}, t_{i,j})$ pair are dependent on the *vertical context* surrounding each spatial location, across channels. In parallel, the RGB image is fed through a modified DeepLabv1 [39] backbone which concatenates outputs from various layers with upsampling and downsampling operations before being fed to a similar bi-directional LSTM for vertical context.

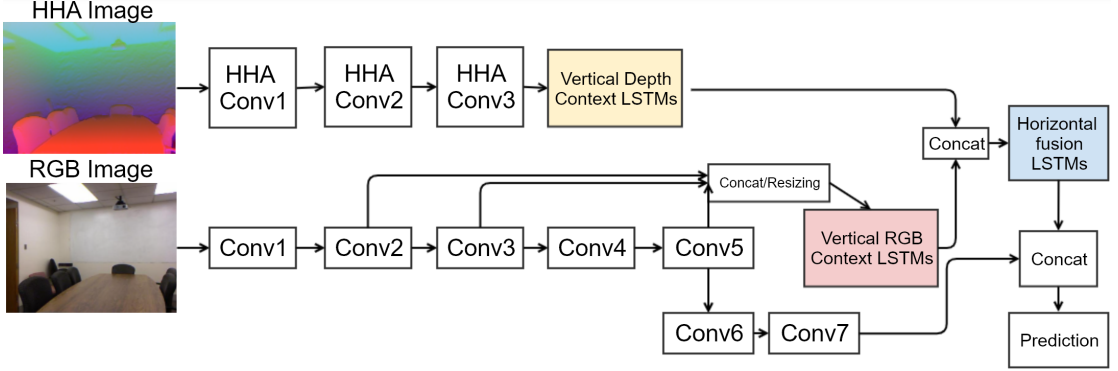


Figure 3.8: The LSTM-CF model [60]. Example RGB/HHA image pair taken from the SUN RGBD dataset [8].

The result of both LSTMs for the HHA and RGB images are concatenated, and fed into a horizontal version of the LSTM previously described. Hence the output of each spatial location from the horizontal LSTMs depend on the features of the input along a horizontal line, stemming from both modalities. This output is then concatenated with the overall RGB DeepLabv1 backbone output, convolved, upsampled to the original image resolution, and fed into a softmax layer. See Figure 3.8 for a visualization. LSTM-CF achieves mIoU of 49.4% on the NYUDv2 40-class semantic segmentation benchmark, and mIoU of 48.1% on the SUN RGBD dataset 37-class benchmark.

Lee et al. [61] extend the RefineNet [62] semantic segmentation architecture to use a dual backbone structure, fusing features from RGB and HHA images extracted from separate ResNet backbones to develop the RDFNet model. A

4-cascaded single-scale RefineNet network is modified with a novel multi-modal feature fusion network. The RefineNet model takes the conv2, conv3, conv4, and conv5 layer outputs from the ResNet backbone, and uses multi-path refinement to incorporate features from each layer using RefineNet-X modules, where X is the layer number. These modules consist of a Residual Convolution Unit (RCU)¹⁰ followed by multi-resolution fusion and chained residual pooling modules, see Figure 3.8 for a visualization of these components.

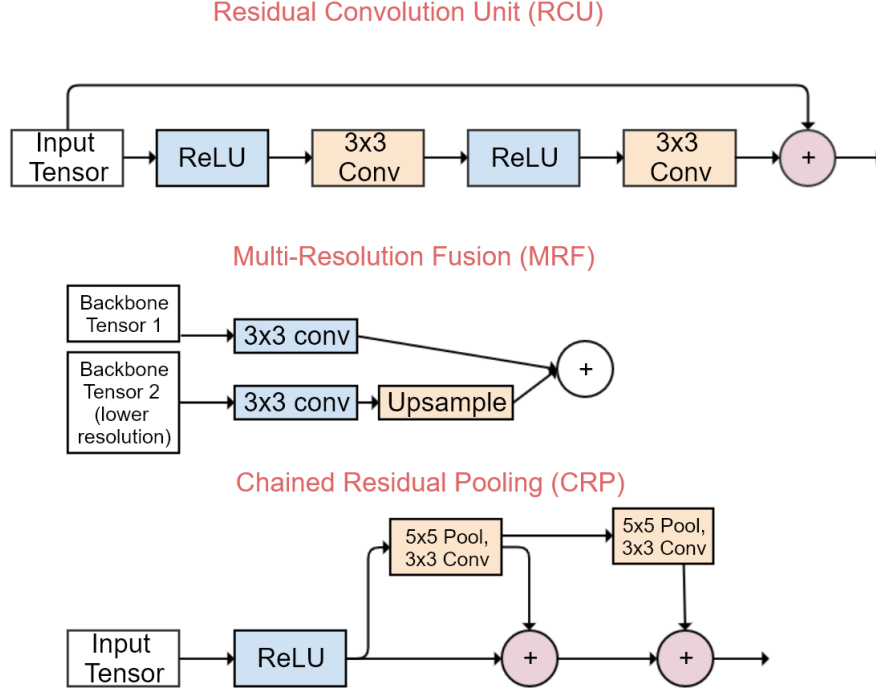


Figure 3.9: The RefineNet sub-components [62].

In the cascaded variant, RefineNet- K takes as input conv $K + 1$ and the output of RefineNet- $(K + 1)$ ¹¹; the multi-resolution fusion module takes the output of two RCU units applied in parallel to each input, and first applies 3×3 convolutions to each, reducing the number of channels to a similar reduced value. The coarser feature map is then upsampled to match the spatial resolution of the finer resolution feature map, before adding the two feature maps element-wise. Afterwards, the chained residual pooling layer takes the output of this multi-resolution fusion and twice applies: a 5×5 convolution, followed by a 3×3 convolution, followed by a residual connection. The high-resolution output of the RefineNet-1 module is then input to two RCU units and reduced channel-wise to the number of classes

¹⁰The Residual Convolution Units employ channel reduction in the first 3×3 convolution, reducing channels to 512 for RefineNet-4 and 256 for the remaining RefineNet modules.

¹¹There is only one input for RefineNet-4 which consists of Conv4.

in the dataset. The result is fed to a dense soft-max layer followed by bilinear interpolation to match the original image size.

The multi-modal feature (MMFNet- X) networks introduced in RDFNet take as inputs the outputs of Conv X from each backbone for each modality, where $X \in \{2, 3, 4, 5\}$. Each input is first reduced channel-wise, and then fed through two separate RCU units, followed by 3×3 convolutions on each modality's feature tensors. Next, the fusion operation consists of an element-wise addition followed by ReLU activation and a residual pooling operation is performed, see Figure 3.10. Hence RDFNet extends RefineNet to RGBD images achieving 50.1% mIoU on the NYUDv2 40-class benchmark, and 47.7% mIoU on the 37-class SUN RGBD benchmark.

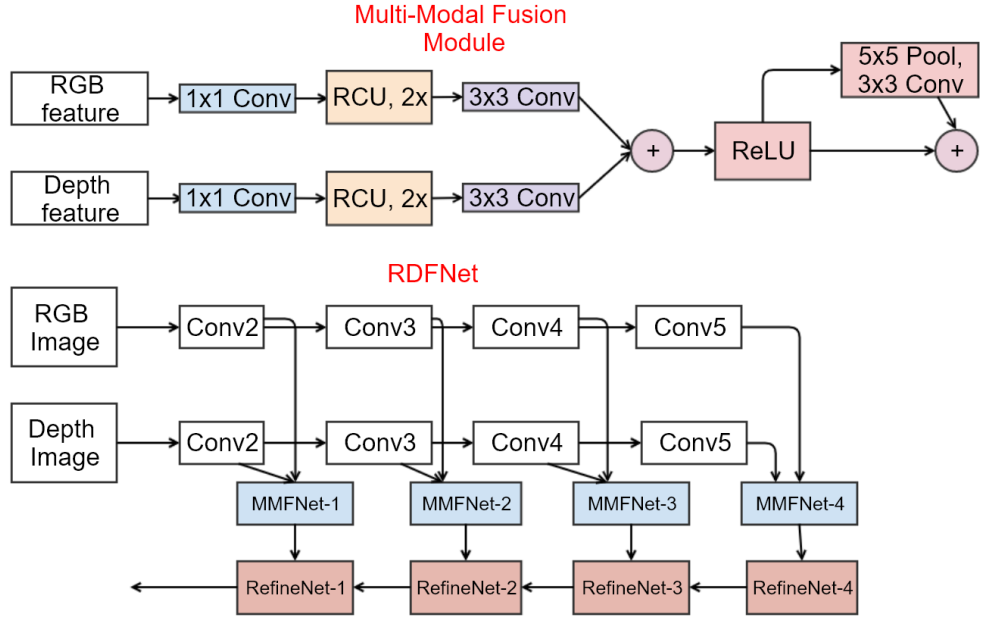


Figure 3.10: The MMF module (top) and an overview of RDFNet (bottom) [61].

Lin et al. [63] use a CNN with multiple prediction branches and introduce the notion of *scene resolution* and *context-aware receptive fields*, developing CFN, the Cascaded Feature Network. As the name suggests, the term scene-resolution refers to the resolution of the objects and surfaces within the scene. There is a correlation between scene-resolution and depth in the sense that high scene resolution corresponds to objects closer to the sensor and lower scene resolution corresponds in general to objects and surfaces further away from the sensor. The goal is to decompose the segmentation task into separate predictions for sets of pixels occupying different depth ranges, and combine the results intelligently.

Given an RGB image, a CNN is used to compute a convolutional feature map F . Using a range of scene resolutions, i.e. depth pixel ranges, image pixels are assigned to one of K different branches. Each branch corresponds to a *scale*, for which the input image is decomposed into a set of superpixels obtained from the method of Structured Random Forests [40]. The earlier the branch number, the lower the scene resolution, and the coarser the decomposition to superpixels. The concept of a *context-aware receptive field* extends the normal notion of the receptive field of a feature map location by aggregating convolutional features of a pixel over its superpixel and adjacent superpixels. For the k 'th branch level, with scale s_k , the input image is subdivided into superpixels $\{S_{i_k}\}$, which are then projected onto the feature map F using the network stride. A context-aware feature map M_k of the same shape as F is obtained such that $M_k(c, x, y)$ consists of the sum of:

- 1) The sum of values of F over the spatial locations corresponding to the projected superpixel that (x, y) belongs to, computed for each channel.
- 2) The average-weighted sum of the values of F belonging to adjacent superpixels, where the average is taken over the size of the adjacent superpixel (again computed for each channel).

Each branch then attends to a feature map $F_k = F_{k-1} + M_k$ with $F_0(x, y, c)$ simply consisting of item 1) above computed with superpixels $\{S_{i_1}\}$. Each F_i is fed to a prediction branch, where the softmax loss of F_i is computed only for pixels in the range of the i 'th scene-resolution or superpixel scale. The sum of the softmax losses for each F_i is then the overall loss to be backpropagated. The final prediction for pixel value (x, y) with scene resolution value j , consists of the max prediction across classes for the upsampled $F_j(x, y)$ (upsampled to the original image resolution). CFN achieves 47.7% mIoU on the NYUDv2 40-class benchmark and 48.1% mIoU on the SUN RGBD 37-class benchmark.

Qi et al. [21] make use of a 3D Graph Neural Network (3D-GNN) to compute RGBD semantic segmentation by reasoning about an RGBD image using its 2D appearance and 3D geometric information. Graph neural networks use an iterative process known as information propagation, where for a node v in a graph, its neighbors propagate contextual information to v depending on their feature values, their proximity in the graph, and the time step of the iterative algorithm.

A point cloud $P = \{(x_i, y_i, z_i)\}$ is first computed from the depth image.

The RGB image is then input to a CNN specialized for semantic segmentation which consists of a VGG-16 architecture with dilated convolutions and reduced downsampling, to obtain a feature map F of shape (H, W, D) . Median pooling is computed on each 2D array of respective (x, y, z) values from the point cloud to obtain a reduced point cloud P' with $H \times W$ points, which can be viewed as tensor of shape $(H, W, 3)$ ¹². Associate the vector at each location (x, y) in the reduced point cloud P' with the (x, y) position in the feature map F . Using the K-Nearest Neighbors algorithm, 64 neighbors per spatial location (x, y) in the feature map are obtained based on the nearest neighbors of the paired point cloud value. Using these neighbors, a directed graph is obtained, where a node v has a directed arc to a node v' if v' is one of the 64 nearest neighbors of v . The propagation process at time step t for vertex v with point cloud coordinate (x, y, z) is computed as

$$m_v^t = \frac{1}{|N(v)|} \sum_{n \in N(v)} g(h_n^t) \quad (3.5)$$

$$h_v^{t+1} = M(h_v^t, m_v^t) \quad (3.6)$$

where m_v^t is the message that node v receives at time step t . This is computed by the average sum of outputs of g input with the hidden states of the neighbors of v , where g is a multilayer perceptron (MLP). The hidden state of a node v at time $t+1$ is updated with an updating function M that takes as input the current hidden state h_t and message m_v^t , where the initial state h_v^0 is the D dimensional vector $F(x, y)$. The update function M can either be another MLP or a LSTM module. After T time steps, for each node, a probability vector v of size C for C classes is computed by an MLP that takes the concatenation of h_v^T and h_v^0 as input, or $M([h_v^T, h_v^0])$. Note that this prediction is formed with both the output of the CNN and the output of the hidden state obtained by the propagation algorithm in order to benefit from the discriminative power of the CNN along with the contextual information obtained from the graph. The entire network is trained end-to-end using the backpropagation through time (BPTT) algorithm [64], achieving 43.1% mIoU on the NYUDv2 test set, and 45.9% mIoU on the SUN RGBD 37-class benchmark.

Wang and Neumann [11] develop a specialized convolution and pooling operation referred to as *depth-aware convolution* and *depth-aware pooling* for the

¹²The reduced point cloud implementation is not elaborated on in the original paper, but can be seen from the PyTorch implementation available at https://github.com/yanx27/3DGNN_pytorch.

modification of a standard CNN backbone. These operators are borne out of the intuition that pixels within a certain kernel window with similar depths should have more impact on the output than pixels with dissimilar depth values. The advantage of this approach is the flexibility to modify any backbone with no additional parameters added and minimal overhead, i.e. to dispense with the need for a second backbone to compute depth features. In spite of this, Wang and Neumann still experiment with fusing features from an HHA trained semantic segmentation architecture with the depth-aware CNN features computed from the RGB values to boost performance. This is achieved by summing the predictions of both RGB and HHA networks, similarly to the approach of Long et al. [36].

Supposing we have an input tensor I with one channel for simplicity. For a single output channel, the depth-aware convolution operation is defined for I at spatial location $p_0 = (x_0, y_0)$ and rectangular filter R of size (k, k) with weights $w(x, y)$ as

$$y(p_0) = \sum_{i=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} \sum_{j=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} w(i, j) \cdot F_D((x_0, y_0), (x_0 + i, y_0 + j)) \cdot I(x_0 + i, y_0 + j) \quad (3.7)$$

with depth similarity term F_D defined as

$$F_D(p_1, p_2) = e^{-\alpha |D(p_1) - D(p_2)|} \quad (3.8)$$

where $D(p_i)$ is the depth of pixel p_i and α is a model hyperparameter. In order to use depth-aware convolutions at any layer of a CNN backbone, the depth image is re-scaled using average pooling to match the resolution of the feature map being convolved. Note that as the absolute difference of depth value between p_i and p_j approaches 0, the depth similarity term approaches 1. When the absolute difference increases, the depth similarity term approaches 0. The depth-aware average pooling operation, a kind of modified average pooling, is given by

$$y(p_0) = \frac{1}{M} \sum_{i=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} \sum_{j=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} F_D((x_0, y_0), (x_0 + i, y_0 + j)) \cdot I(x_0 + i, y_0 + j) \quad (3.9)$$

where

$$M = \sum_{i=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} \sum_{j=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} F_D((x_0, y_0), (x_0 + i, y_0 + j)) \quad (3.10)$$

Wang and Neumann modify the backbone of a DeepLabv1 semantic segmentation architecture with these depth-aware operators and achieve mIoU of 44.7% on the

NYUDv2 40-class benchmark and 42.0% on the SUN RGBD 37-class benchmark respectively.

Taking into account the review of the RGBD semantic segmentation literature described in this section, as well as the review of RGBD object detection models in Section 3.1, a set of observations can motivate further inquiry and experimentation. The most obvious point may be that the majority of RGBD works concern semantic segmentation alone, without any investigation of instance level segmentation methods, with the exception of Gupta et al.’s works [7, 43]. Due in part to the lack of a standardized benchmark in the literature, alternate forms of segmentation such as instance segmentation are neglected in this body of research. Hence in Section 4.2.2, the popular Mask R-CNN architecture will be adapted to make use of depth features extracted from a second backbone. Moreover, a novel 13-class benchmark is introduced on the SUN RGBD dataset to test the approach with multiple datasets where Gupta et al. only test on the NYUDv2 dataset.

In addition, recent segmentation research for RGB images has expanded to a new formulation of segmentation, combining instance segmentation and semantic segmentation. In 2018, panoptic segmentation was added to the list of tasks for the MS-COCO competition, in order to encourage researchers to study things and stuff in a unified approach [65], fully replacing the semantic segmentation “Stuff” challenge in 2020. Since then, many prominent works such as DETR [66] and SoloV2 [67] have included panoptic segmentation algorithms as part of an overall detection or segmentation framework. Models such as EfficientPS [68] have treated panoptic segmentation as the exclusive focus of a proposed algorithm. Given that RGBD research has applications in robotics, where robots need to be able to distinguish instances, it would be reasonable to assume that more detailed segmentation approaches would be of benefit. At present, there is no prominent work in the RGBD literature that has developed a depth-aware panoptic segmentation architecture. This thesis will present an RGBD panoptic segmentation architecture in Section 4.2.4. A simple method for combining ground truth instance masks and semantic segmentation masks to achieve a panoptic mask for images in the NYUDv2 and SUN RGBD datasets is given in Section 5.2.4.1 to develop RGBD panoptic segmentation benchmarks. Similar to the RGBD object detection literature, there lacks a diversity of fusion approaches from features collected in a dual backbone (RGB and depth) setup, with the notable exception of RDFNet [61]. Moreover, it is worth examining whether RGB features should be given a kind of

structural precedence over depth/HHA features in this fusion process, since the former may be more discriminative.

Chapter 4: Methodology

In this chapter, novel depth-aware architectures for each task described in Section 2.2 are presented in detail, along with a discussion of the motivation for their structures, including how they differ from existing approaches. Each model has a corresponding RGB baseline counterpart, which will be used for comparison of single modality models, i.e. architectures trained and tested with RGB images or depth/HHA images input alone, vs. multi-modal RGBD networks. The object detection, instance segmentation and semantic segmentation models are used as building blocks for a final panoptic segmentation model. Although panoptic segmentation subsumes these tasks, by experimentally validating the best fusion approach for each of these sub-goals, the best overall fusion approach can be deduced, possibly making use of different modules for instance segmentation than for semantic segmentation. A description of the fusion approaches to be used precedes the descriptions of the proposed architectures.

4.1 Fusion Modules

Most of the RGBD models discussed in the literature review chapter can be subdivided into two broad categories:

- Geometric-centered models, in which the absolute depth values or depth value ranges are used to structure the feature extraction, such as 3D-GNN [21], and CFN [63].
- Image-centered models, in which features from RGB and depth/HHA images are combined to obtain a final prediction, such as RDFNet [61], or FuseNet [58].

The models proposed in this chapter are exclusively image-centered, focusing on the interaction between RGB and depth/HHA features extracted from convolutional neural networks. Geometric-centered models that make use of absolute depth values to obtain a three dimensional representation, while theoretically interesting, suffer from the issue that depth images are in general quite noisy. The sources of sensor noise and lack of precise output, being outside of the scope of this work, result in non-trivial inaccuracies in the 3D representation. In the image-centered approach using learned fusion, depth image noise is better accounted for with the inherent smoothing, denoising, and generalizing ability of learned con-

volitional filters. The NYUDv2 dataset provides depth images that have been denoised using a 2D in-painting algorithm. The SUN RGBD dataset uses a depth map improvement algorithm that makes use of nearby frames in a temporal neighborhood of the frame considered, however the resulting 3D representation is not entirely accurate; hence neither dataset can be relied upon for an accurate point cloud representation without a robust 3D denoising algorithm. However, as indoor depth sensing technology improves, this may be less of an issue in the future if sensor noise becomes less of a concern. It is conjectured in this thesis that inaccurate depth values in a depth image, although sometimes inconsistent, *can actually be predictive of certain types of objects or surfaces that are prone to noise* such as mirrors, windows, and doors. Such noise patterns can actually still preserve a kind of structural integrity for objects and surfaces. Consider Figure 4.1, where the windows in the top right corner are assigned an erroneous depth value of 0, whereas the shape of the windows is mostly left intact. The same can be said about the glass pane in the door frame.



Figure 4.1: Observing noise patterns in a depth image from the SUN RGBD dataset [8].

Another common example of sensor noise derives from the range of the cameras themselves. Pixels corresponding to objects and surfaces outside of the camera’s range are assigned a depth value of 0, which in indoor settings is often related to open doors, long hallways, and ceilings. A point cloud generated from such a depth map without a 3D denoising algorithm would be inaccurate, and could lead to false predictions, while the removal of noisy pixels results in an overly sparse representation. While the HHA encoding algorithm has to account for noisy pixels with regards to surface normal computation, the disparity channel uses estimated depth values for noisy pixels, where the result is a full-resolution, coherent image. Indeed approaches making use of 3D geometry have to deal with either reduced resolution or inaccurate point clouds. Image-centered models using denoised depth

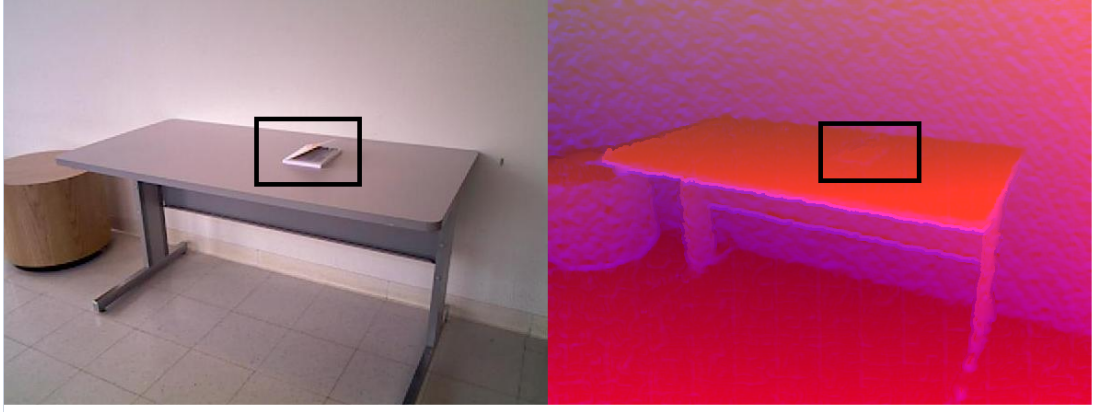
images enjoy both full resolution and reasonably accurate cues from 3D geometry when input with HHA images.

In order to extract features from the RGBD images for each task, a dual backbone approach will be employed, using the same backbone architecture for each modality, such as two ResNet-50 models. The first backbone will be input with an RGB image, while the second backbone will be input with an HHA image. Certainly, the features extracted from depth images or HHA images with convolutional neural networks differ from those extracted from regular RGB images. Broad and sometimes crude structural relations in the scene define interactions between objects and surfaces within depth imagery. Since it is known that the early layers of CNNs tend to focus on extracting low level features such as edges, corners, curves, lines, contrast changes, etc. it may be reasonable to posit that early depth/HHA features extracted by a CNN are along the lines of:

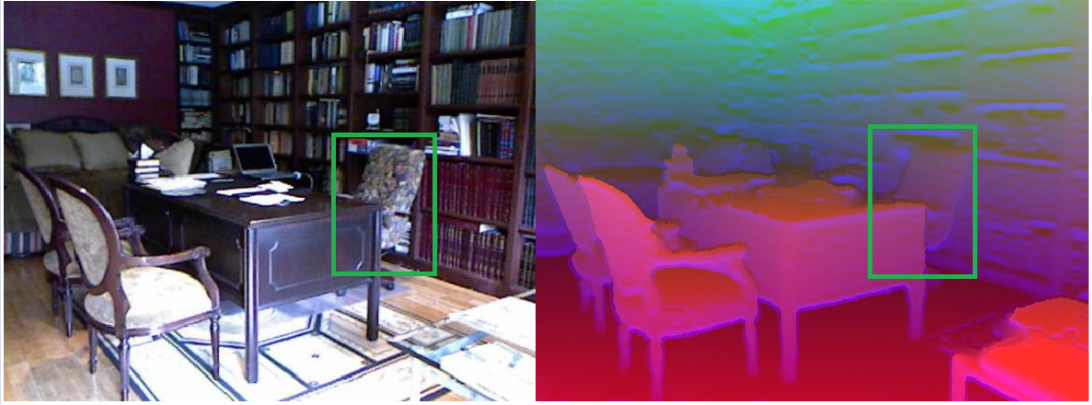
- Depth edges, a sharp change in how close entities are to the camera, sometimes due to a change in objectness.
- Surface normal edges, in which the outward orientation of an object changes sharply.
- Gradual height changes or disparity changes as objects and surfaces increase in height or get further away from the camera.

The later layers of the model may extract discriminative task-specific features such as surface normal information related to an object class, wall and ceiling segments, or distinct entities separated by closed depth contours.

For detection and segmentation, a cursory visual inspection of RGB/HHA image pairs reveals that some objects/surfaces may be easier to distinguish in one modality versus the other. For example, observe Figure 4.2a. The paper on the table makes a very slight outline in the HHA image, while its color and textural information in the RGB image is quite noticeable. Hence, one might expect a pure RGB model to have an easier time detecting/segmenting such a piece of paper. However, if one examines Figure 4.2b, there is a high-frequency texture on a chair outlined with a green bounding box that may confuse an RGB model, whereas the chair in the HHA image has clear contour and surface normal information consistent with a sloped back rest, which is clearly separate from the image background.



RGB/HHA Image Pair a)



RGB/HHA Image Pair b)

Figure 4.2: RGB and HHA image pairs with different qualities for object detection and image segmentation. The RGB/HHA image pairs are taken from the SUN RGBD dataset [8].

In light of the advantages and disadvantages expected for object detection and image segmentation from each modality, improved performance from the use of both modalities is possible in a deep learning context if a network can learn to incorporate the best information from both image types in one unified representation. Within the context of detection this should mean more accurate localization using the rough shape properties in the depth/HHA image to disambiguate object contours, and more accurate classification by ignoring extraneous textural and color information. Suppose that we have two identical Faster R-CNN networks, one trained with HHA Images, and one trained with RGB images, and consider a category C_i . Each model when tested will have a certain AP_{50} performance output on category C_i . An ideal dual-modality model should then perform as

$$AP_{50}(C_i)_{Dual} \geq \max(AP_{50}(C_i)_{RGB}, AP_{50}(C_i)_{Depth}) \quad (4.1)$$

When the depth model outperforms the RGB trained model or vice versa, the

dual approach should *at least match the better performing of the two modalities*, and ideally outperform both. Hence, researchers employing dual backbone multi-modal architectures need to intelligently fuse features from both modalities so as to get the best of both worlds. However, due to the fact that the RGB images contain much more information than their depth/HHA counterparts, it will be strongly conjectured that RGB models should outperform the depth/HHA models for all tasks considered, and so in some sense the *the RGB features should take priority*. To further bolster this point, one can observe that some depth images exhibit extreme noise when compared to their color image counterpart. In this scenario, an ideal depth-aware model should learn to massively favor the RGB features, despite the potential to learn priors about objects and surfaces prone to noise discussed earlier. Consider Figure 4.3, a very noisy depth image from the SUN RGBD dataset; the signal to noise ratio in such an image should be accounted for in the fusion process of an ideal depth-aware network.



Figure 4.3: Left: An RGB image, Right: A very noisy depth image. Images taken from the SUN RGBD dataset [8].

An adjacent concern, one which requires extensive experimentation, is to structure network training such that the learned RGB features emanating from RGB-trained backbones are not corrupted by losses from fused features. This corruption is defined by decreased model performance at test time. While works such as RGBD R-CNN [7] discuss this problem, the combinations of transfer learning and weight freezing are not fully explored. In this thesis' experiments, in order to mitigate the possibility of a corrupting influence occurring, weight freezing of the RGB backbones for each task is employed as needed, determined by its effect on overall performance. Depth backbones will require fine-tuning more frequently; transfer learning used to initialize the weights of the backbones is always based on RGB

images, and so it is reasonable to assume that training depth backbones should improve performance. Based on early experimentation, the most successful transfer learning and freezing procedures for each task are outlined in Section 5.2.

Upon examining the literature for RGBD-based computer vision using this dual backbone approach, a taxonomy of fusion approaches can be observed.

- *Early Fusion*, in which a single backbone is used, where its input consists of a concatenation of RGB and depth channels using a 4-channel image or a 6-channel RGB/HHA pair. Gupta et al. [7] explore early fusion, noting a degradation in performance compared to the pure RGB counterpart.
- *Intermediate Fusion*, where the feature maps output from intermediate (and possibly final) layers of the backbones interact in some manner. An example of this approach would be the FuseNet architecture [58] discussed in Section 4.2.
- *Late Fusion*, Late Fusion, where the task-specific network head/s are input with a feature map fused from each backbone’s outputs, or the predictions of two networks can be combined as in the FCN approach [36]
- *Hybrid Fusion*, using some combination of the above.

An alternative to feature map fusion is the model stacking approach, in which the predictions of two base networks for each modality are input to another module to obtain an overall prediction. This approach disregards the possibility of exploiting the interrelations between the feature maps of both modalities, possibly at multiple layers of the CNN architectures used, and so will not be explored.

Fusion approaches may incorporate deep learning modules, or consist of non-learned tensor operations such as element-wise multiplication, element-wise addition, or concatenation along the channel axis. The fusion models employed in this thesis are exclusively learned approaches, the underlying assumption being that raw interaction of features is sub-optimal. By comparing a diversity of learned fusion types, the goal is to determine if more complex modules are worth the extra parameters, in terms of performance gains. Another important idea behind the development of these modules is the *plug-and-play approach*, namely that given two CNN backbones, the fused feature maps can be input to the same exact task-specific modules that a pure RGB algorithm would employ by ensuring the fusion process maintains the appropriate output shape. This is done to allow for transfer

learning on pre-trained components such as the CNN backbone and the RPN and FPN weights of the models used for object detection and instance segmentation to be introduced in Section 4.2.1.

The first fusion module, a simple baseline meant for comparison, will be referred to as a *learned concatenation* operation, denoted f_{LC} , and has been employed in works such as the model of Ophoff et al. [47], albeit without batch normalization. The module consists of a concatenation of RGB feature tensors and HHA feature tensors along the channel axis, assuming that both have the same number of channels, say C . This concatenated tensor is input to a 1×1 convolution with no bias to reduce the channels to C , followed by batch normalization, as shown in Figure 4.4 (ignoring the batch index).

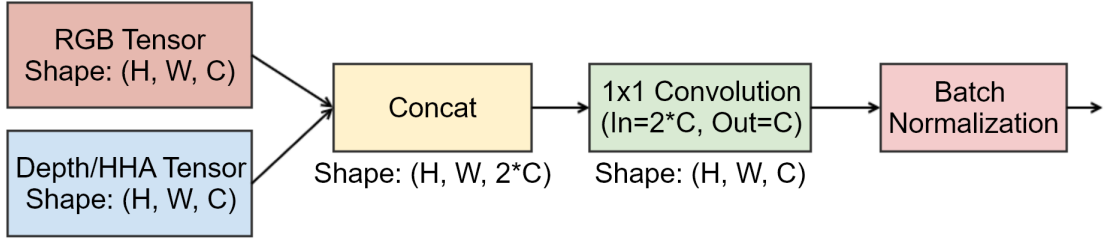


Figure 4.4: The learned concatenation approach, f_{LC} .

The f_{LC} operation learns linear combinations of features from both tensors, where the output channels of the convolution is reduced so that the resultant tensor can be input to network structures from the pure RGB counterpart for which the depth-aware models are based upon. For example, by applying f_{LC} to the output of two ResNet-50 backbones, the resulting tensor can be input to the RPN and RoI heads of Faster R-CNN. The ReLU activation function applied to the output of f_{LC} was explored in early experiments, however its addition was found to be detrimental. The addition of batch normalization was found to speed up network convergence.

The second fusion approach to be used involves a novel residual learning approach, inspired by the bottleneck layer used in ResNets [14]. This operation deliberately differs from the Multi-Modal Feature (MMF) module given in RDFNet [61] by giving explicit preference to the RGB features in the residual connection, rather than treating both modalities as equally valuable. In particular, the *learned residual connection* proposed, denoted f_{Res} , operates as a multi-modal bottleneck layer with dimension reduction.

In f_{Res} , the input RGB and depth/HHA feature tensors are concatenated, with a reference to the RGB tensor maintained. The resultant concatenated tensor is input to the first three convolutions of the bottleneck layer with the intermediate dimension equal to half of the input channel dimension, and an expansion factor of 2. The reduced intermediate dimension is used to reduce the parameters of the 3×3 convolution, hence the term bottleneck. The output of these convolutions is added element-wise to the input RGB tensor, forming the residual connection, followed by a 1×1 convolution to reduce the output channels in half. Early experiments with ReLU activation at the output of the f_{Res} operation showed a degradation in performance, and so it was omitted. See Figure 4.5 for a comparison of the ResNet bottleneck layer archetype, the f_{Res} operation, and the MMF module given in RDFNet [61], described in Section 3.2.

The intuition behind f_{Res} stems from a similar idea presented by He et al. [14]. If the fused RGB and depth features do not provide valuable information, then f_{Res} should be capable of learning to approximate the identity function, letting mostly RGB features influence the output. On the other hand, if there is valuable complementary information, the output of the convolutions added to the RGB tensor should yield a more discriminative output for further processing.

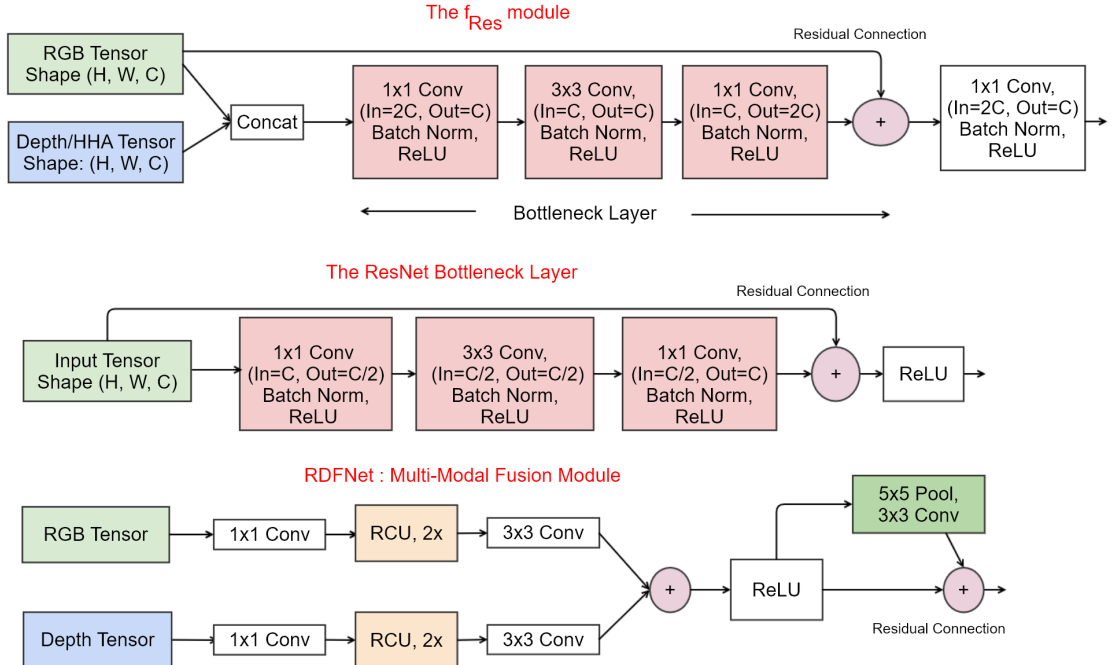


Figure 4.5: The f_{Res} operation, the ResNet bottleneck layer [14], and the MMF module from RDFNet [61].

The third fusion module proposed uses a novel gating approach, denoted as f_{Gated} .

Concurrently to the development of ResNets, an approach known as Highway Networks [69] was developed, incorporating gated connections in order to increase the depth of feed-forward deep neural networks. This gated approach was then adapted by Oyedotun et al. [16] in the context of RGBD image classification by making use of a gate $G = \sigma(WM_D + b)$ where W is a matrix of weights, M_D is the depth tensor output by a CNN backbone, b is a learnable bias, and σ is the log-sigmoid activation function. G is a fully connected layer with an activation and a bias. Let M_{RGB} be a RGB tensor output by another backbone. Oyedotun et al. use G as in the operation:

$$T(M_{RGB}, M_D) = \text{Concat}(((1_T - G(M_D)) \circ M_{RGB}), (G(M_D) \circ M_D)) \quad (4.2)$$

where 1_T is a tensor with all 1 entries of the same shape as M_D , and $\circ$ denotes element-wise multiplication. Then $T(M_{RGB}, M_D)$ is input to fully connected layers leading to a classification vector.

The f_{Gated} approach will replace the function G with G' : a 7×7 convolution operation with same padding and no bias, with input channels equal to output channels. Now

$$T'(M_{RGB}, M_D) = \text{Concat}(((1_{T'} - G'(M_D)) \circ M_{RGB}), (G'(M_D) \circ M_D)) \quad (4.3)$$

The result of $T'(M_{RGB}, M_D)$ is fed through a batch normalization layer, followed by a 1×1 convolution to reduce the dimension of the output to the same number of channels as the RGB/depth input, followed by another batch normalization layer.

To understand the gating effect, note that if the operation $G'(M_D)$ results in a tensor with all 1 entries, the gate will only allow the depth tensor's entries to pass, concatenated with a tensor of the same shape but all 0 entries. Similarly, if the operation $G'(M_D)$ results in a tensor with all 0 entries, only the entries from the RGB tensor will pass through, again concatenated with a tensor of the same shape with all 0 entries. Hence the goal of the gate is to learn how to filter the depth tensor to function as two complementary masks acting on the RGB and depth tensors. By replacing the fully connected layer in the gating approach of Oyedotun et al. [16] with a convolutional layer, the shape of the image tensors can be preserved, allowing for both intermediate and late fusion. By contrast, a fully-connected layer requires the backbone outputs to be flattened, losing all notion of image shape. Moreover, the filtering operation of f_{Gated} allows each pixel

and channel $G'(M_D)$ to influence the corresponding pixel and channel of M_{RGB} , merging depth and RGB features in a spatially-aware manner.

In order to reduce the number of parameters of the f_{Gated} module to decrease training/inference time as well as the number of parameters for semantic segmentation, a variation of the gated approach will be used. In this variation, both the RGB and depth/HHA tensors will first be reduced channel-wise in half, followed by two separate parallel batch normalization operations. The tensor G' will be computed on the channel-reduced M_D , followed by the application of equation 4.3 and batch normalization. See Figure 4.6 for a visualization of both gating approaches. Comparing the number of trainable parameters for each fusion module input with RGB/depth tensors with C input channels, seen in Table 4.1, for $C = 256$ the gated approach uses the most parameters by far, while f_{LC} uses a modest number of parameters. Indeed there may be a trade-off between performance gains and the memory footprint of the fusion modules, not to mention the risk of overfitting induced by introducing a large increase in parameters. Gated variations will be denoted v1 and v2.

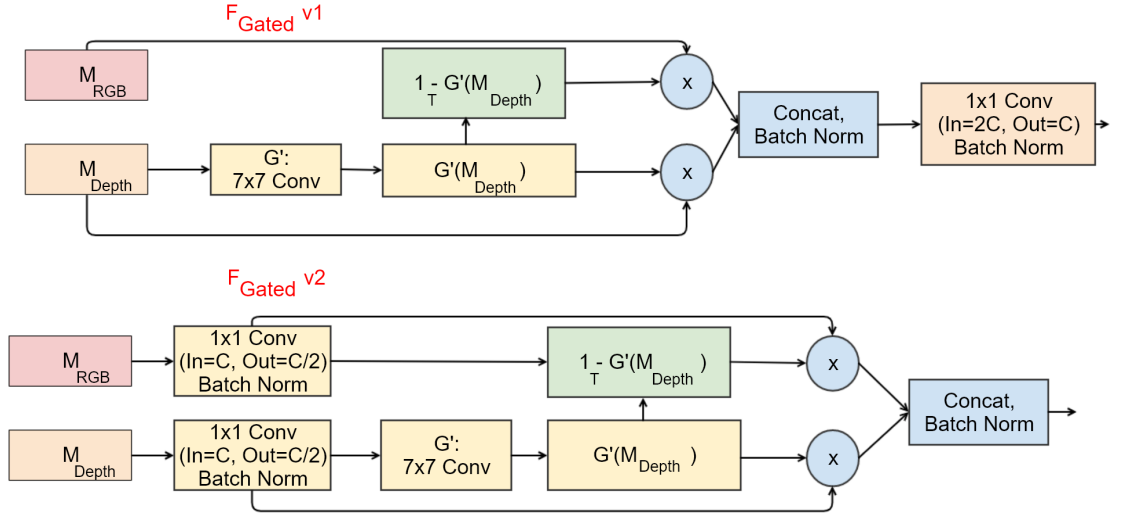


Figure 4.6: The f_{Gated} fusion module, versions 1 and 2.

For each of the tasks considered, except in the case of panoptic segmentation, each fusion module will be tested, employing a single fusion type per model. This allows for both task-specific comparisons, and a more holistic evaluation of which approach is beneficial in general. The panoptic segmentation model will make use of a mixture of fusion approaches.

Fusion Module	Params: C channels	Params: 256 channels
f_{LC}	$2C^2 + 2C$	131584
f_{Res}	$4C^2 + 6C$	263680
f_{Gated} v1	$51C^2 + 6C$	3343872
f_{Gated} v2	$53C^2/4 + 6C$	869888

Table 4.1: The number of parameters per fusion module for C input channels, and for $C = 256$ input channels.

4.2 Proposed Architectures

4.2.1 Object Detection

For the objective of object detection, a Faster R-CNN FPN will be modified to use two backbones, inputting fused features to the FPN layers, creating a depth-aware FPN using the fusion modules described in Section 4.1. The philosophy and novelty behind this choice can be expressed through a series of key points:

- In approaches such as RGBD R-CNN [7], and the follow-up work of Gupta et al. [43], the region proposals, while using both RGB and depth features, are pre-computed for each image, where the backbone does not make use of the regression and objectness losses backpropagated from an RPN. With the exception of pre-training of the CNN backbones, the depth-aware model presented in this section is *trained fully end-to-end*.
- In the model of Xu et al. [44], three different sets of region proposal networks and RoI heads are used in a modified Faster R-CNN algorithm. The predicted region proposals and detected boxes are obtained by simply averaging the objectness/classification and regression values for each head. This approach triples the number of RPN and RoI head parameters. Two separate Fast R-CNN models are trained for each modality with pre-computed region proposals, where the classification and regression outputs of both are again averaged. The model presented in this section is designed to use *a single RPN and network head*. Hence the idea of 2.5D region proposals from RGBD R-CNN is revitalized in the approach proposed in this thesis; whereas now features are extracted from dual backbone CNNs, and region proposals are computed using region proposal networks.
- The creation of a depth-aware Faster R-CNN architecture that uses learned

intermediate fusion is novel to this thesis. A dual backbone FPN should combine the benefits of multi-scale object detection advertised by FPNs with the advantages of feature fusion at multiple levels of a CNN backbone, with each layer’s output containing different semantic information.

- The use of residual or gated learning fusion between the features extracted from RGB and depth/HHA trained backbones could result in a more sophisticated learned integration of the two modalities R-CNN based works in the literature. For instance the correlation operation in the Correlated Detection Network [44] is a fixed mathematical operation with no learnable parameters.

Using two ResNet-50 CNN backbones, without the image classification layers, denote the ConvX layer outputs for each modality as $R := \{R_2, R_3, R_4, R_5\}$ and $D := \{D_2, D_3, D_4, D_5\}$ for RGB and depth/HHA feature maps respectively (recall that FPNs do not use Conv1 due to its large memory footprint). A set of fusion modules parameterized by the number of input channels of the tensors is used at each stage of the image pyramid i , say f_i , to obtain a set of multi-modal feature maps $M := \{M_2, M_3, M_4, M_5\}$ computed as $M_i = f_i(R_i, D_i)$ for $2 \leq i \leq 5$. The fusion module used will be the same type, for example the f_{Res} module, for each i . The standard operations of an FPN are then computed on the M_i . First each M_i is reduced channel-wise to 256 channels with a 1×1 convolution, obtaining C'_i . Set $P_5 := C'_5$. For $2 \leq i \leq 4$, P_i consists of the element-wise addition of C'_i with an upsampled map (by $2x$) C'_{i+1} , followed by a 3×3 convolution of stride 1 and same padding for anti-aliasing. The *pool* tensor consists of a $2x$ subsampling of the P_5 tensor using max pooling with a kernel of size 2×2 applied with stride 2. In other words, the output of the fusion modules are then input to the standard layers of an FPN to yield $\{P_2, P_3, P_4, P_5, Pool\}$, which will then be used for RPN and Faster R-CNN network head training. The depth-aware FPN is pictured in Figure 4.7.

In order to project the region proposals to the correct level of the feature pyramid network output, a Multi-Scale RoI Align operation is used, where initially the scale-equation input with an RoI of size (w, h) of the form

$$k = \lfloor 4 + \log_2(\sqrt{wh}/224) \rfloor \quad (4.4)$$

is used to determine the FPN output level at which to project the region proposal. Next, RoI Align is performed [17] to extract a fixed size tensor of spatial resolution

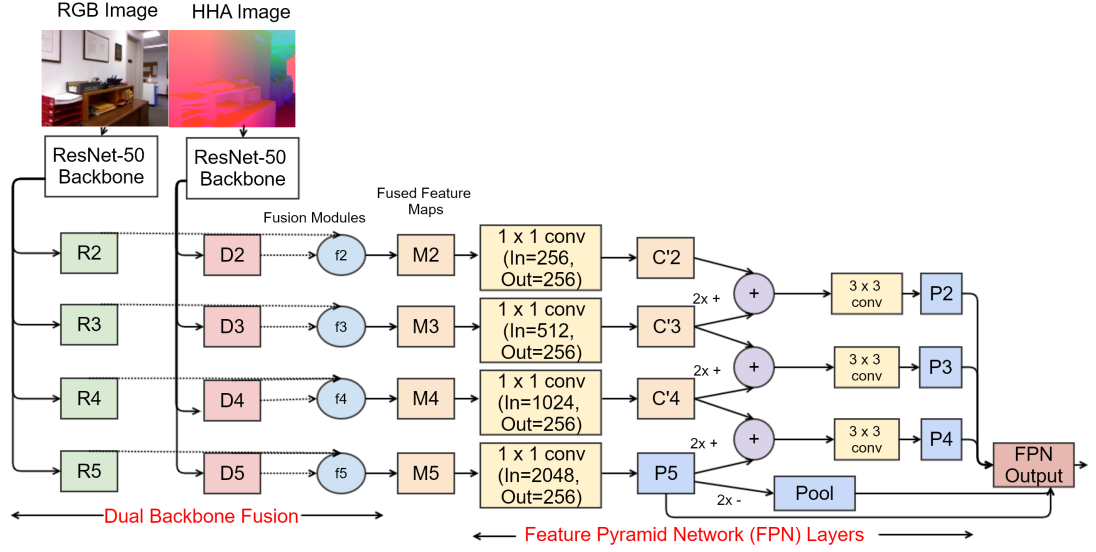


Figure 4.7: A dual backbone multi-modal feature pyramid network. Example RGB/HHA image pair taken from the SUN RGBD dataset [8].

7×7 , replacing RoI pooling used in the original Faster R-CNN implementation. Notably, the RGBD R-CNN based object detection models in the literature review section all make use of RoI pooling, rather than the more modern RoI Align.

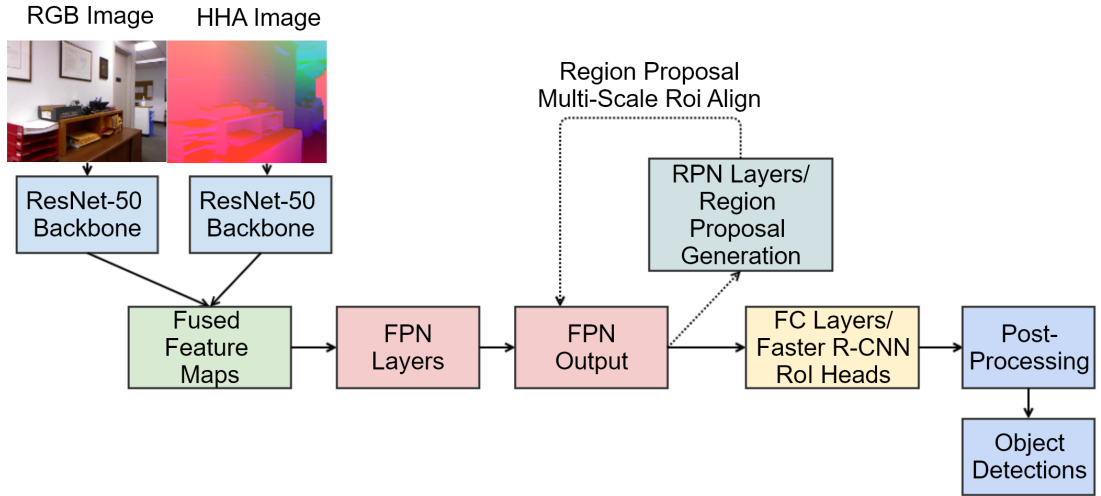


Figure 4.8: The proposed depth-aware Faster R-CNN FPN using the dual backbone FPN presented in Figure 4.7. FC = Fully Connected. Example RGB/HHA image pair taken from the SUN RGBD dataset [8].

An overview of the depth-aware Faster R-CNN FPN network is shown in Figure 4.8, using the approximate joint training scheme where the RPN is trained alongside the RoI heads. During the training of a standard RGB Faster R-CNN FPN, the contribution of the backbone weights is computed with respect to the output of the FPN layers. However, in the dual backbone fusion approach the loss of the

backbone weights is now computed with respect to the fusion output. This leaves open the possibility of fine-tuning pre-trained RGB and depth backbones explicitly for the fusion task, and even training both networks simultaneously, which differs from the separate training approach described by Gupta et al. [43].

Testing on RGBD object detection benchmark datasets, by comparing the results to a Faster R-CNN FPN input with RGB images, the goal of the proposed RGBD network trained is to achieve:

- A higher mean average recall for the region proposals generated from the RGBD network than the RGB counterpart.
- A higher mAP_{50} , mAP , and average mAR for the objects detected by the depth-aware model.
- A consistent class-wise improvement of the AP_{50} metric.

Detailed experiments comparing the single input-type Faster R-CNN FPN models with the depth-aware adaptations are given in Section 5.2.1.

4.2.2 Instance Segmentation

For the task of instance segmentation, a novel dual backbone Mask R-CNN FPN will be used, extending the method described in Section 4.2.1 by adding the mask-branch described by He et al. [17], which runs in parallel to the RoI heads of the Faster R-CNN framework. Due to its strong performance, Mask R-CNN is a good choice for a modern architecture to adapt for this objective. The most prominent RGBD instance segmentation approaches in the literature, RGBD R-CNN [7] and the follow-up work by Gupta et al. [43], use a sequential algorithm in which objects are first detected using an R-CNN based approach using region proposals computed from superpixels developed with RGBD features. A random forest regressor input with hand-crafted RGB and depth features is computed on features within the superpixels, whereafter instance mask predictions from the superpixels are aggregated. The superpixels that have non-empty intersection with some object within a detected bounding box, and that achieve a certain prediction threshold, have their inner pixel values set to 1, giving an instance mask for each object. This method suffers from the weakness of the sequential, non end-to-end nature of the pipeline. For instance, if the region proposals do not have high recall with the ground truth boxes, the objects detected may suffer low recall. Further if the objects detected do not have high recall, then there is little hope that the

instance masks will have high recall.

The Mask R-CNN FPN architecture however has the benefit of using multi-task learning to achieve segmentation, and does not use any hand-crafted features. In particular the CNN backbone benefits from the loss during training corresponding to the RPN, the RoI heads, and the mask branch. Moreover, it is shown by He et al. [17] that objects detected using the bounding box and class labels output by Mask R-CNN were more accurate after training with a mask-branch than a corresponding Faster R-CNN model, and so it is reasonable to argue that the loss from each sub-task (region proposals, object detection, segmentation masks), weighted properly, aid in the instance segmentation objective. One small modification of the Multi-Scale RoI Align algorithm used is to increase the resolution of the pooled tensor from 7×7 to 14×14 , since the fully convolutional layers of the mask branch require a higher resolution for accurate pixel-level predictions. The proposed depth-aware Mask R-CNN FPN is shown in Figure 4.9.

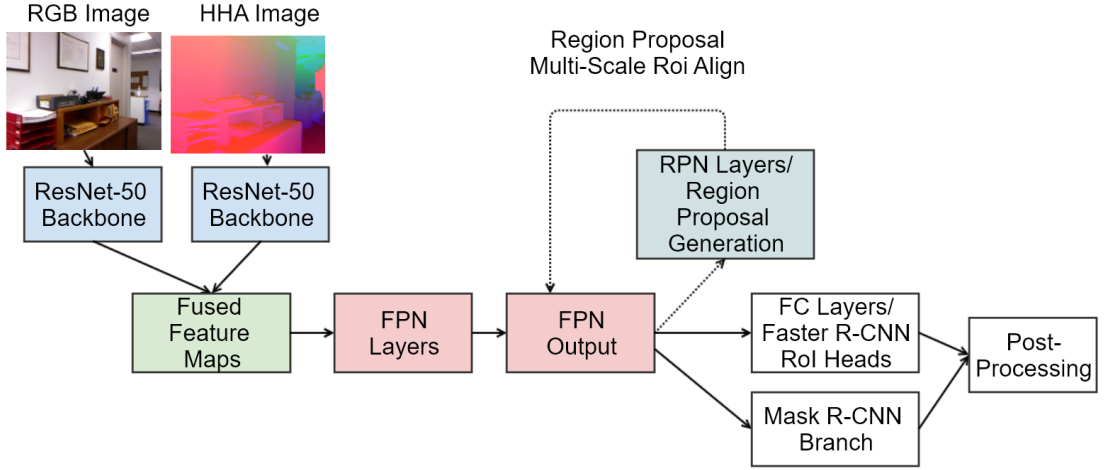


Figure 4.9: The proposed depth-aware Mask R-CNN FPN using the dual backbone FPN presented in Figure 4.7. Example RGB/HHA image pair taken from the SUN RGBD dataset [8].

By adding features extracted from a FPN with input fused from an RGB backbone and a depth backbone, the goal is to improve the localization accuracy of the detected boxes, to obtain more precise segmentation where one modality fails to give adequate information, and to use depth features to disambiguate certain classes from one another. By using intermediate fusion and taking advantage of the image pyramid used in the FPN/Multi-Scale RoI Align approach, the semantic and pixel level information at the appropriate scale are used to inform mask predictions. In Section 5.2.2, experiments are conducted to measure the performance

boost of depth features fused with RGB features in this context.

4.2.3 Semantic Segmentation

The semantic segmentation framework to be modified for RGBD inputs is the DeepLabv3 architecture [18], chosen for its strong performance and simplicity. Using a late fusion approach, a novel RGBD semantic segmentation model is achieved where the originality lies in the adaptation of DeepLabv3 to RGBD inputs, the type of fusion used, and the diversity of fusion approaches tested.

The backbones used will be ResNet-101 models, which use dilated convolutions and reduced downsampling in order to preserve the spatial resolution of the feature maps. The backbone’s output is input to the DeepLabv3 network head, which makes use of the Atrous Spatial Pyramid Pooling (ASPP) operation, consisting of highly dilated convolutions computed in parallel. These convolutions allow for multi-scale context at each pixel location. By fusing features from RGB and depth/HHA feature tensors, this context is augmented. Features such as depth edges, surface normal edges, and height changes within multiple radii from a pixel can now influence the prediction results, alongside the multi-scale context provided by RGB features.

The ASPP layer is a fully convolutional approach mimicking the post-processing technique of conditional random fields used in models like DeepLabv1 [39], used to propagate a wider context to each predicted pixel, resulting in less spurious patches and more consistent object contours in the output segmentation mask. Approaches such as the DeepLabv1 modified with the depth-aware convolution described by Wang and Neumann [11], or FCN [36], which simply sum the results of two networks, discard any potential to learn interactions between modalities, and further allow the corrupting influence of depth image noise to potentially detract from an accurate prediction based on RGB feature tensors. In 3D-GNN [21], RGB and HHA tensors are concatenated before the message propagation algorithm with no learned fusion between modalities. The most similar technique to the one presented in this section is RDFNet [61], where each fusion module treats the RGB and depth features equally; the novel fusion modules detailed in Section 4.1 differentiate between modalities. RDFNet also employs intermediate fusion rather than late fusion, made easier by using the RefineNet architecture [62]. Intermediate fusion in the context of RGBD semantic segmentation may be less useful than late fusion since depth tensors may require significant smoothing

due to sensor noise, possibly corrupting the overall network prediction.

The depth-aware adaptation of DeepLabv3 presented in this section will be tested with each fusion module from Section 4.1 to learn to combine features from two ResNet-101 backbones *before being input to the network head*. The entire proposed network structure, including a detailed visualization of the head is shown in Figure 4.10.

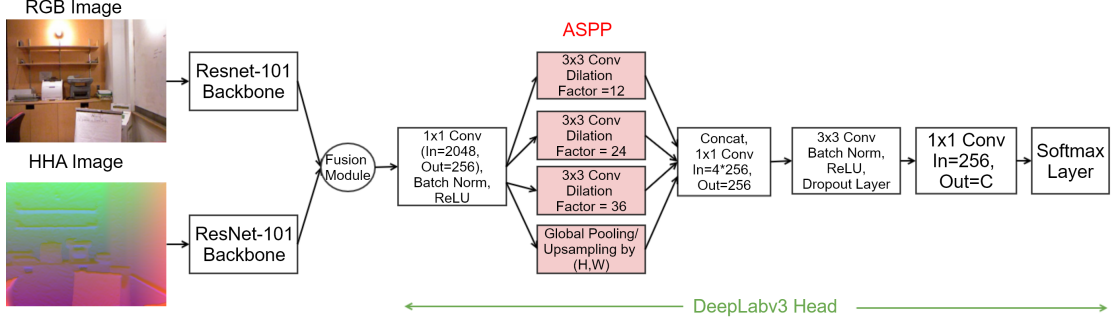


Figure 4.10: The proposed depth-aware DeepLabv3 architecture. Example RGB/HHA image pair taken from the SUN RGBD dataset [8].

The expectation of the benefit of fusing depth/HHA features with RGB features in the context of semantic segmentation should be similar to that of instance segmentation for individual objects. For stuff classes, in particular indoor classes such as walls, ceilings, floor, the use of height and surface normal information should play a prominent role. For example, in order to distinguish walls from a ceiling or floor with similar textural information, the height above the ground is a good indicator for which class a pixel may belong. This disambiguation should result in a higher mean IoU and better pixel level accuracy. Experimental validation with the proposed depth-aware version of DeepLabv3 will be presented in Section 5.2.3.

4.2.4 Panoptic Segmentation

The majority of models within the RGB literature for panoptic segmentation operate by using separate network heads for instance segmentation and semantic segmentation which share a single backbone, such as EfficientPS [68]. The output of the heads is combined either through heuristics and fixed post-processing techniques, or a learned approach, yielding the predicted panoptic masks. One of the earliest approaches in the literature which was meant as a very simple baseline, described by Kirillov et al. [31], simply consisted of post-processing techniques applied to the outputs of completely separate Mask R-CNN and PSPNet

[70] models, where the latter is a high performing semantic segmentation architecture. However, by using a single backbone for instance segmentation and semantic segmentation feature extraction, multi-task learning incorporates weighted losses from bounding box regression/classification and segmentation heads, as well as region proposal losses in the case of region-based methods. Each of these losses are weighted and summed, to be backpropagated to a set of shared weights.

The RGB model to be modified in this thesis to incorporate RGBD images will be a hybrid of Mask R-CNN FPN and DeepLabv3 models, with a novel modification of the FPN structure. The backbone used will be a ResNet-101 model, with dilated convolutions and reduced downsampling as in the DeepLabv3 backbone discussed in Section 2.7. The depth-aware variant will then use two backbones, one for RGB images and one for the HHA images to perform intermediate fusion for input to the FPN, and late fusion for input to the DeepLabv3 head. For this objective, the fusion module chosen will no longer necessarily be of the same type for the entire network. The best performing fusion module, taking into account performance and efficiency for the instance segmentation and semantic segmentation models described in Sections 4.2.2 and 4.2.3, will be used to incorporate *task-specific fusion*, a concept introduced in this thesis. The idea behind task-specific fusion is that semantic segmentation and instance segmentation from RGBD input may require different flavors of feature extraction, and so there may need to be different learning approaches for each sub-task within a single panoptic network.

Consider Table 4.2, which compares the output sizes per layer of the DeepLabv3 ResNet-101 backbone vs. the standard ResNet-101 model, both input with an 800×800 image. The outputs of the DeepLabv3 backbone have redundant spatial resolutions, destroying the in-network image pyramid. To achieve both a feature pyramid network, and a larger spatial resolution for the DeepLabv3 head, max pooling is used to reduce the resolution of the output of layers Conv4 and Conv5, using 2×2 kernels of stride 2 and 4 respectively. The DeepLabv3 head will continue to take as input the output of Conv5, while the FPN will make use of the entire backbone output starting from Conv2, layer per layer, using max pooling as needed. The depth-aware network will perform fusion before the max pooling operations, and before the input to the DeepLabv3 head. The modified RGB FPN is shown in Figure 4.11, with C5 and the FPN outputs fed to the network heads of Mask R-CNN and DeepLabv3 respectively. The dual backbone adaptation is

shown in Figure 4.12.

Layer	DeepLabv3 ResNet-101	Standard ResNet-101
Input	800×800	800×800
Conv1	400×400	400×400
Conv2	200×200	200×200
Conv3	100×100	100×100
Conv4	100×100	50×50
Conv5	100×100	25×25

Table 4.2: Resolution sizes per output layer of the DeepLabv3 and standard ResNet-101 backbones.

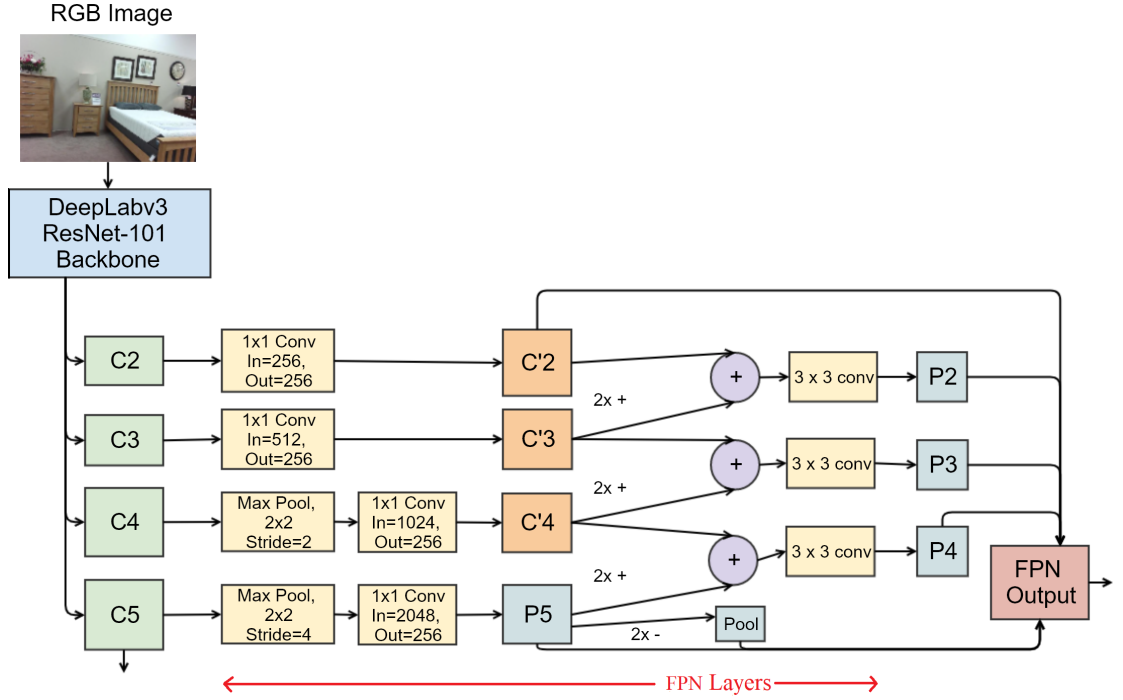


Figure 4.11: The modified RGB FPN to be used in the proposed panoptic segmentation architecture. Example RGB image taken from the SUN RGBD dataset [8].

The semantic segmentation head can predict segments from both instance and stuff categories, while the predicted instance mask categories are reserved for things. To obtain the final panoptic mask, the predicted instance masks must be integrated with the predicted semantic segmentation mask. In order to combine the instances with the semantic segmentation mask, the method of Kirillov et al. [31] is used, a kind of specialized non-maximal suppression. Proceeding in decreasing order

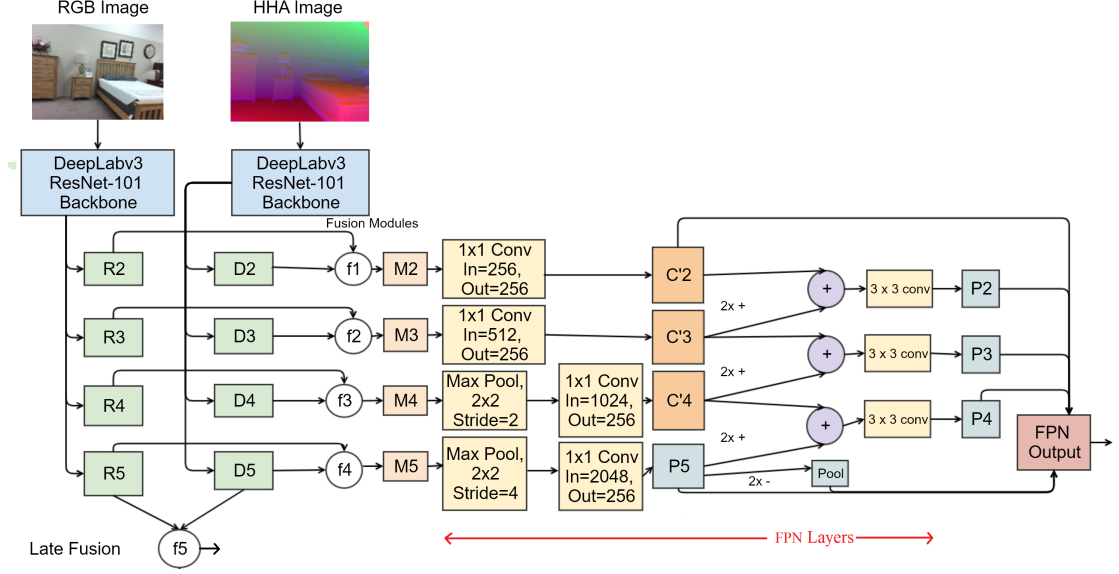


Figure 4.12: The modified depth-aware FPN. The fusion modules are denoted by f_i where f_5 may differ from the type of fusion used in f_j for $1 \leq j \leq 4$. Example RGB/HHA image pair taken from the SUN RGBD dataset [8].

of confidence scores, instance masks are pasted on top of the segmentation mask. For each instance mask, if more than 75% of the pixels have not already been predicted by other masks with higher confidence values, the pixel locations not already predicted will be pasted on top of the segmentation mask, otherwise the instance mask is discarded. Only those instances with a confidence value of 0.6 or larger are considered, in order to further address pasting issues. A naive approach of pasting instance masks without suppression can result in overlapping issues, resulting in poorer performance, and a very aesthetically unpleasant result. A full overview of the proposed depth-aware panoptic segmentation architecture is visualized in Figure 4.13.

Indeed, the RGBD literature has demonstrated depth-aware models with improved performance on tasks such as image classification, object detection, image segmentation, and scene labeling compared to an RGB counterpart. Therefore it stands to reason that panoptic segmentation will benefit from depth features as well, with instance boundaries outlined more clearly, clearer distinction between classes with similar textural information, and structural classes benefiting from height and surface normal information. Using the instance masks and semantic segmentation masks provided by the NYUDv2 and SUN RGBD datasets, a make-shift panoptic annotation is developed for training and testing, detailed in Section 5.2.4.1. For these benchmarks, there is a class imbalance between stuff and things

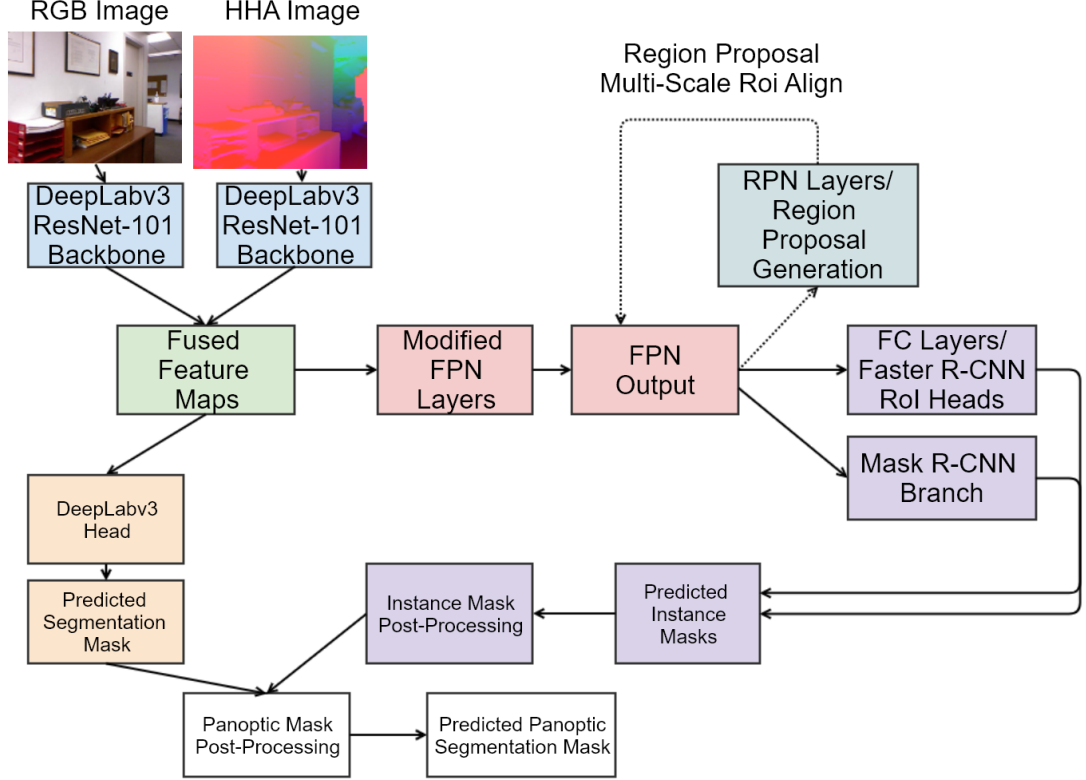


Figure 4.13: An overview of the proposed depth-aware panoptic segmentation architecture. Example RGB/HHA image pair taken from the SUN RGBD dataset [8].

categories in favor of stuff. As a result, weighting the semantic segmentation loss in relation to the instance segmentation loss will be an important consideration during training, given that instance segmentation is targeting things only. Transfer learning will favor backbones trained on semantic segmentation tasks, rather than instance segmentation.

Training such a complex model comes with many considerations. A second ResNet-101 backbone as well as fusion modules added on to an already parameter-heavy RGB baseline model results in a huge number of model parameters. The approach employed in Section 5.2.4 will consist of fine-tuning each backbone separately and freezing parameters during the final overall network training in order to maximize performance and ensure network convergence.

Chapter 5: Experimental Validation

5.1 Datasets

A relatively recent trend in computer vision research is the use of publicly available datasets, allowing researchers to compare different models and approaches by training and testing on the same sets of images. In the early days, researchers trying to demonstrate a novel method for a given task, such as facial recognition or image classification, would publish their results tested on *their own data*. The algorithms described would cater to the type of image instances from the authors' respective image repositories, and so the robustness of the approach for real-world application was not measured. As a consequence, large-scale public datasets began to emerge for a more realistic tool to compare model performance. An algorithm's performance could be objectively measured by considerations such as:

- A performance metric such as *mean average precision* for object detection, or *pixel accuracy* for semantic segmentation.
- The training/inference time of an algorithm with a certain performance expectation.
- Performance on a subset of the data, such as objects in a specific size range.

Large-scale datasets can be used for pre-training a model for task A , and then using some components of the pre-trained model for task B on a different dataset, making use of so-called *transfer learning*. In this thesis, CNN backbones used for object detection and image segmentation make use of pre-training on the MS-COCO dataset specific to each respective task, with the exception of panoptic segmentation, where a backbone pre-trained on semantic segmentation will be used. As argued by He et al. [71], task-specific pre-training is beneficial for faster network convergence as well as increased performance, particularly when training and testing on smaller datasets. Contrast this with the common approach of using networks pre-trained on the ImageNet dataset, which is an image classification dataset.

The majority of large-scale datasets within the literature focus exclusively on color images without a depth channel, although there are RGBD datasets for model comparison. They are typically smaller than the larger more prominent

RGB datasets. This creates an added challenge: not only do algorithms input with RGBD images have to make effective use of the depth channel, but they *have to do so with a relatively small number of training examples*. Evaluation of the models described in Section 4.2 will be conducted on the NYUDv2 and SUN RGBD datasets, which were chosen based on their popular use throughout the literature concerning RGBD computer vision models for indoor settings.

5.1.1 NYUDv2

Just as the ImageNet dataset occupies the title of the canonical RGB image classification dataset, one could argue that the New York University Depth Version 2 RGBD-dataset (NYUDv2) [6] is the de facto standard dataset for RGBD-based semantic segmentation. The dataset is comprised of video sequences from 464 different indoor scenes of varied types: furniture stores, bathrooms, office spaces, kitchens, offices, bedrooms, etc. The data is obtained with Microsoft’s Kinect v1 depth camera. From the overall video sequence, there are 1449 aligned RGB and depth image pairs with dense per-pixel annotation, organized at the instance level where each region is given its own binary segmentation mask. For regions corresponding to entities deemed objects such as chairs or bookshelves, each distinct instance is given a unique id, i.e chair 1 or chair 2, alongside a class label. Regions that are not deemed individual objects such as floors and ceilings are given a single id with a class label, reminiscent to the *stuff vs. things* distinction discussed by Kirillov et al. [31].

In order to account for noise inherent to the Kinect sensor’s output of a depth map for a given view, a so-called *in-painting* [72] algorithm is used to estimate the depth values of the missing depth pixels. As noted by Silberman [73], missing values in the depth image are generally caused by:

- Shadows caused by the disparity between the near-infrared emitter and the camera.
- Random missing or spurious values caused by specular (reflective) or low albedo surfaces, the latter being surfaces that reflect very little incoming radiation.

Two important benchmarks in the literature have been developed from this dataset:

- The 40-class semantic segmentation benchmark developed by Gupta et al.

[22]. These classes have the labels: wall, floor, cabinet, bed, chair, sofa, table, door, window, bookshelf, picture, counter, blinds, desk, shelves, curtain, dresser, pillow, mirror, floor-mat, clothes, ceiling, books, fridge, tv, paper, towel, shower-curtain, box, whiteboard, person, night-stand, toilet, sink, lamp, bathtub, bag, other-struct, other-furniture, other-prop.

- The 19-class object detection benchmark proposed in RGBD R-CNN [7], consisting of classes: bathtub, bed, bookshelf, box, chair, counter, desk, door, dresser, garbage-bin, lamp, monitor, night-stand, pillow, sink, sofa, table, television, toilet.

One of the difficulties inherent to this dataset is the small number of training examples (given by a 795/654 train/test split), which can cause overfitting to occur. The environments are cluttered and dense in content, with many objects partially occluded and viewed from many different orientations, offering researchers a serious challenge. Additionally, the NYUDv2 dataset suffers from annotation quality issues, namely:

- There are missing ground truth instances for classes from the 19-class object detection benchmark, resulting in false positives at test time and less than ideal training data.
- Erroneous class labels are given to instances and semantic segmentation regions.
- The issue of inconsistent segmentation style of objects. For example in some images, the outer frame of the bookshelf class is segmented as an instance mask, while in others the entire structure is given one large segment without holes.
- Ambiguous class labels are chosen for the 19-class object detection dataset, for example many desks look like tables and vice versa. The box class spanned objects such as cereal boxes to mailroom packages. There is also ambiguity with respect to class labels for the 40-class semantic segmentation benchmark, most notably the classes other-prop, other-struct and other-furniture which do not represent consistent object or surface types.
- The boundary and inner pixel values as given by the ground truth instance and semantic segmentations masks are at times inaccurate for certain object types, such as the chair class.

- There exists large class imbalances both for the object detection benchmark and for the semantic segmentation benchmark.

5.1.2 SUN RGBD

The proliferation of RGBD datasets continues to grow as consumer-grade depth sensors become more mainstream. Motivated by the success of data-hungry computer vision algorithms, Song et al. developed the SUN RGBD dataset [8] to provide the community with a dataset at a similar scale to that of the Pascal VOC dataset [30]. The NYUDv2 dataset is an order of magnitude smaller than Pascal VOC; models trained on smaller datasets can suffer from degraded performance as a result of overfitting when compared to those models trained on larger datasets. Furthermore, algorithms trained on a depth dataset taken with a single sensor may exhibit *dataset bias* and *sensor bias*. It is known that many popular RGB datasets (Pascal VOC, Caltech 101, etc.) can themselves be identified by a classifier due to biases inherent to the data collection process such as the camera used, the types of scenes, and other factors [74]. For this reason, Song et al. use aligned RGB and depth images from 4 different sensors:

- The Intel RealSense 3D Camera designed for tablets (capturing 1,159 images). This sensor projects an infrared pattern to the environment and uses stereo matching to obtain the depth map.
- The Asus Xtion Live Pro for laptops (capturing 3,389 images), which uses a near-infrared light pattern with stereo matching.
- Microsoft’s Kinect v1 depth sensor (capturing 2,003 images), which also uses a near-infrared light pattern with stereo matching.
- Microsoft’s Kinect v2 (capturing 3,784 images) depth sensor which uses time-of-flight technology.

The SUN RGBD dataset is a mix of newly acquired data, and previous depth datasets. The NYUDv2 depth dataset is included in the dataset, along with 554 scenes from the B3DO dataset [75], and 3,389 frames from the SUN3D RGBD videos [76]. Newly acquired data is obtained from the Kinect v2 (3,784 images) and Intel RealSense (1,159) sensors. The overall focus of the dataset is indoor scenes within universities, office spaces, living/dining rooms, bedrooms, and furniture stores in North America and Asia. The training/test split samples from each

sensor evenly for 5285 training images, and 5050 testing images.

Each image’s ground truth annotation includes 2D bounding boxes, instance segmentation masks, and semantic segmentation masks. There are also 3D annotations: 3D bounding boxes, object orientation/pose, and room layout estimation, which will not be made use of in this thesis. The labels of the bounding boxes/instance masks are extensive (800 categories), however a base set of classes is used in the author’s benchmark tool using the same 19 object detection categories as the NYUDv2 benchmark. Similarly for semantic segmentation, two types of masks for each image are given: an *all* mask of extensive categories labeled densely, and a smaller 37-category mask, composed of the first 37 categories of the NYUDv2’s 40-class segmentation benchmark, removing the other-struct, other-furniture, other-prop classes. The 37-category semantic segmentation benchmark is the most common use of the SUN RGBD dataset for training and testing [61, 21, 58, 11].

The raw depth images from each sensor type contain measurement noise and missing pixels, and so Song et al. employ a depth map improvement algorithm [8]. An example of a denoised depth image is shown in Figure 5.1. Similarly to the NYUDv2 dataset, the annotation quality of the SUN RGBD dataset suffered from the same issues outlined in Section 5.1.1, with the added issue that many pixels in the ground truth semantic segmentation masks are given a value of 0 corresponding to the unknown class, resulting in less training pixels per image. In some cases these pixels should have been labeled as one of the 37 classes in the benchmark described. There is also a large class imbalance for both the object detection and semantic segmentation benchmark.

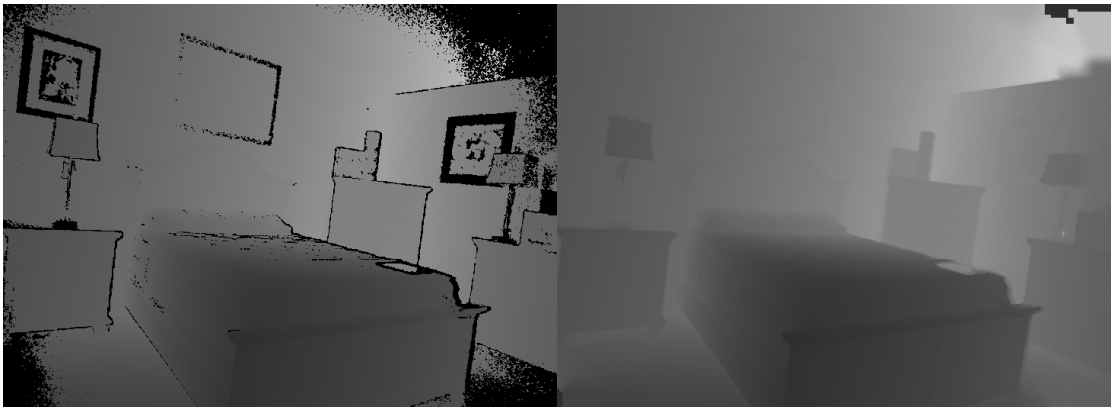


Figure 5.1: A noisy depth image and a denoised depth image from the SUN RGBD dataset [8].

5.2 Experiments

In this section, the implementation details and results of the tests conducted on the NYUDv2 and SUN RGBD datasets for each task defined in Section 2.2 are reported, relating to each of the proposed depth-aware architectures introduced in Section 4.2, making use of the fusion modules described in Section 4.1. The performance of these models is compared to the RGB architectures from which they are derived using RGB, HHA, and depth images for a single input-type model, using training and testing images of the same type. For each experiment, the PyTorch machine learning framework was used (version 1.8), with code taken from the open-source Torchvision library [77]. For training and testing, an NVIDIA RTX 3090 GPU (graphics processing unit) with 24 GB VRAM is used, with an Intel-9900k 5.0 GHZ 8-core CPU and 64 GB of RAM. This is quite a memory-heavy setup, allowing for large image resolutions and batch sizes that may be less feasible on the typical 6-12 GB of VRAM that entry level GPUs for machine learning typically have. In order to further aid memory management and speed up training and inference, PyTorch offers mixed-precision training, which combines 32-bit floating point (FP32) arithmetic (the default precision level) with a half-precision FP16 data storage format, which will be used in this thesis' experiments. Computer vision/natural language processing deep learning models trained with the mixed precision format have been shown to be as accurate as those models trained with FP32 data using the same hyperparameters, while using less memory and exhibiting faster training/inference time [78]. For reference, with mixed precision, the largest network trained, the dual backbone panoptic segmentation architecture, used approximately 22 GB VRAM with a batch size of 4 images.

For each model trained, for each task, the number of training epochs was determined by an optimal ratio of epoch loss to training set size, obtained through validation. All hyperparameters were either chosen based on default values from the literature or validation. A batch size of 4 images was used for each model trained, as in 4 RGB and HHA images each for dual backbone models, and 4 RGB/HHA/depth images for single input type models. All fusion module weights were initialized using the Kaiming Uniform weight initialization scheme [79]. Sub-experiments were conducted to determine the best configuration of transfer learning and backbone layer freezing techniques, where for brevity the most successful approach will be described, possibly with differences for each dataset and/or fusion module used. Running stats were kept for each batch normalization layer

so that during testing these statistics could be used to normalize layer inputs. Further, the batch normalization parameters in the semantic segmentation backbones are frozen in the PyTorch implementation in order to account for smaller batch sizes. No data augmentation techniques were used in order to conduct the fairest comparison between other prominent models, which either used different data augmentation schemes or none at all, where for instance the CMAC model of Li et al. [20] simply used horizontal flipping as the augmentation strategy, and in FuseNet [58] no data modification was performed at training time other than image re-scaling. The image order was shuffled at each epoch.

The backbone used was a ResNet variant for each task, using the same number of layers for RGB and depth/HHA images. While depth images contain less information than RGB images, a cursory visual evaluation of HHA images reveals that there is still a substantial amount of detail within each channel, even within relatively small pixel neighborhoods, with a significant degree of variance observed for the angle with gravity channel. Some dual backbone works in the literature explore the use of a lighter depth backbone, such as the model of Gupta et al. [43], while other models such as FuseNet [58] and RDFNet [61] use the same backbone for each modality. In particular, the best performing RDFNet model uses two ResNet-152 models, and so there is precedent for the use of two heavy backbones, although there is a clear performance to efficiency trade-off. Another advantage of using the same backbone for each modality is that fusion operations at the same layer number can make use of feature maps with identical tensor shapes.

Each backbone used, regardless of input type, is initially pre-trained on an MS-COCO dataset, relevant to the task, for example backbones for object detection are pre-trained on the MS-COCO 2015 object detection training dataset. The choice of the MS-COCO dataset for pre-training was made due to the availability of pre-trained backbone weights from the PyTorch framework, as well as the decreased training time and performance boosts obtained from task-specific transfer learning. In particular, the 2015 object detection training dataset was used, which has bounding box and instance mask annotations, as well as the MS-COCO Stuff training dataset. According to the PyTorch documentation [80], the RGB inputs for pre-trained ResNet models should be normalized in a specific manner to optimize training and performance. The images should be mapped to the range $[0, 1]$, after which *channel-normalization* should be computed using the average normalized RGB values of the ImageNet dataset and the stan-

dard deviation of these channels, which are $[\mu_1, \mu_2, \mu_3] = [0.485, 0.456, 0.406]$, and $[\sigma_1, \sigma_2, \sigma_3] = [0.229, 0.224, 0.225]$ respectively. These values are provided in the documentation. For an input image $I(c_i, x, y)$ at channel c_i , and spatial location (x, y) , the expected input for pre-trained backbones is of the form

$$I'(c_i, x, y) = \frac{\frac{I(c_i, x, y)}{255} - \mu_i}{\sigma_i} \quad (5.1)$$

During early experiments, pre-trained backbones used for depth or HHA images also benefited from this normalization strategy, and so for every experiment conducted, the input images were normalized as in Equation 5.1. Depth images are first converted to the repeated grayscale format outlined in Section 2.1.1 before this normalization.

5.2.1 Object Detection

5.2.1.1 Implementation Details

For the objective of object detection, a comparison in performance was made between the proposed depth-aware Faster R-CNN FPN model described in Section 4.2.1 to the standard Faster R-CNN FPN model introduced by Lin et al. [13]. The RGBD approach inputs HHA and RGB image pairs to two separate backbones, combining the feature map outputs from each backbone using the fusion modules described in Section 4.1. These fused feature maps are then input to a FPN, after which the standard RPN and RoI heads from Faster R-CNN are used for object detection on the output of the FPN. The two types of models tested were:

- Baseline single input-type Faster R-CNN FPN models. Three such models are input with only RGB, depth and HHA images respectively.
- The proposed dual backbone Faster R-CNN FPN models, input with RGB and HHA image pairs. For each fusion module type, f_{LC} , f_{Res} , or f_{Gated} , a different model is trained using a single fusion operation. These models will be referred to as the Proposed Faster R-CNN FPN X architectures for $X \in \{f_{LC}, f_{Res}, f_{Gated}\}$.

In total 6 models were trained, 3 with a single image type input, and 3 with RGB and HHA pairs. A ResNet-50 network was used for each backbone. The PyTorch implementations of the ResNet model, the FPN layers, the RPN network and the Faster R-CNN RoI head layers were used. All linear layer weights from the

RoI heads were initialized using random sampling from a uniform distribution, where for an output tensor T with Out feature vectors, samples are taken from the distribution

$$U(-Out, Out) \quad (5.2)$$

In particular for the NYUDv2 dataset, the following models were trained

- A Faster R-CNN FPN trained with RGB image input, the RGB baseline. The backbone was pre-trained on the MS-COCO 2015 [23] object detection training dataset, with all layers frozen. The FPN and RPN layers were also pre-trained on MS-COCO.
- A Faster R-CNN FPN trained with only HHA image input. The backbone was pre-trained on MS-COCO, layers unfrozen, with FPN and RPN layers pre-trained on MS-COCO. This setup was repeated for depth images for comparison, yielding the HHA and depth image baselines.
- A dual backbone Faster R-CNN FPN using each of the fusion modules. Both backbones were pre-trained on the MS-COCO dataset, but only the HHA backbone was unfrozen. The FPN and RPN layers were pre-trained on the MS-COCO dataset.

The simplest potential explanation for why freezing the RGB backbones resulted in improved performance for the NYUDv2 dataset is that the dataset is so small that training the backbone resulted in overfitting. This is not the case for HHA and depth backbones since their distribution/appearances are quite different than standard color images, and so a compromise of potential overfitting with the need to fine-tune to the new distribution was optimal.

For the SUN RGBD dataset, which is larger than the NYUDv2 dataset, a slightly different training approach was used. The baseline RGB Faster R-CNN FPN was trained with an unfrozen backbone. Additionally, the Faster R-CNN FPN f_{Res} , f_{Gated} models utilized backbones pre-trained on the SUN RGBD dataset, which were initially pre-trained on the MS-COCO object detection dataset, for both the RGB and HHA inputs; these configurations were found to be optimal in early experiments.

At training time and test time, RGB images were resized using bilinear interpolation to a resolution of 600×800 , following a similar protocol described by

He et al. [17], preserving the 4:3 aspect ratio present in the majority of images in both datasets. Ground truth bounding boxes were rescaled to this larger resolution both at training and testing time. At test time, the predicted bounding boxes were resized to the original image resolution for IoU computation with the ground truth bounding boxes. Only those detections with a confidence score larger than 0.5 were considered. Depth images and HHA images were resized to the resolution 600×800 using the nearest neighbors algorithm rather than bilinear interpolation, so as not to infer spurious interpolated values.

The hyperparameters employed by each Faster R-CNN FPN, single or dual backbone, were mostly left as the default parameters described in the original works [12, 13]. Differing from the original Faster R-CNN implementation, anchor box areas from the set $\{32^2, 64^2, 128^2, 256^2, 512^2\}$ corresponding to the levels $\{P_2, P_3, P_4, P_5, Pool\}$ output by the FPN were used with 3 different aspect ratios $\{1, 1.5, 0.5\}$. During testing and training, the top 2000 region proposals, ordered by confidence score, were considered before non-maximal suppression, after which the top 1000 proposals were regressed and classified. The number of anchors sampled per image was set to 256 (as opposed to 512 described by Lin et al. [13]), and the anchors with an IoU of 0.2 or less with any ground truth box were considered as negative samples (different from 0.3 in Faster R-CNN [12]). The Multi-Scale RoI Align grid used was of size 7×7 .

The loss function used for training weighted the RPN and RoI head losses equally, and is given by Equation 2.28 in Section 2.4. The optimizer algorithm chosen was stochastic gradient descent, using the same parameters as detailed in Faster R-CNN [12], with a momentum of 0.9 and weight decay of 0.0005. The learning rate was initialized to 0.003, using learning rate schedulers:

- NYUDv2: Decreasing the learning rate by a multiplicative factor of 0.6 every 6 epochs.
- SUN RGBD: Decreasing the learning rate by a multiplicative factor of 0.7 every 6 epochs. This factor was slightly larger to decrease training time.

The bounding boxes corresponding to the 19-class benchmark described in Section 5.1 were used for testing and training. Ground truth bounding boxes were obtained for the NYUDv2 dataset from the relevant code of Dr. Saurabh Gupta’s GitHub page [81]; this was necessary since the original NYUDv2 has 894 class categories,

and so to have a direct comparison to other works in the literature that made use of the 19-class benchmark, the precise class mapping was needed. The SUN RGBD ground truth bounding boxes were obtained using the v2 corrected bounding boxes [82], however some modifications were made to the class labels. In particular, in order to increase the number of training/testing objects and reduce the number of false positives, a mapping of class names of objects very similar to the 19 classes was used. For example, the class name *kitchen sink* is mapped to the name *sink*, where the latter is one of the 19 classes. This class mapping is given in Appendix B. For both datasets, very small bounding boxes were removed from both training and testing datasets due to poor annotation quality (less than 25*25 pixels in the original image).

5.2.1.2 Results and Analysis

The per-class AP_{50} of the baseline and proposed depth-aware Faster R-CNN FPN models tested on the NYUDv2 dataset is displayed in Table 5.1, where the AP_{50} metric is introduced in Section 2.2.1.

Class	Baseline RGB Faster R-CNN FPN	Baseline Depth Faster R-CNN FPN	Baseline HHA Faster R-CNN FPN	Proposed Faster R-CNN FPN f_{LC}	Proposed Faster R-CNN FPN f_{Res}	Proposed Faster R-CNN FPN f_{Gated}
bathtub	18.0	14.9	23.6	28.5	16.3	10.9
bed	69.2	71.9	76.2	70.3	70.1	66.8
bookshelf	40.0	27.5	33.7	46.0	44.3	49.3
box	8.6	1.4	2.3	8.3	7.6	6.6
chair	59.2	50.2	50.4	57.7	55.6	53.5
counter	46.8	38.4	38.7	48.7	45.6	42.5
desk	7.1	4.4	6.3	13.1	11.4	6.5
door	22.1	8.0	7.2	26.2	21.9	21.6
dresser	32.3	29.7	38.4	42.4	40.6	32.4
garbage-bin	51.5	30.4	31.4	52.3	45.1	46.0
lamp	51.9	42.6	41.4	53.4	44.3	50.3
monitor	61.2	21.5	19.0	58.3	53.0	52.5
night-stand	53.0	32.8	40.0	58.2	47.2	58.0
pillow	47.5	38.0	45.0	53.4	48.1	47.7
sink	69.2	37.2	40.0	70.9	66.1	63.5
sofa	55.7	50.3	56.8	55.8	57.4	54.3
table	31.6	22.5	34.1	31.6	27.9	26.5
tv	45.2	13.3	15.3	48.0	48.6	46.3
toilet	78.7	60.3	53.4	70.0	78.4	75.5
mAP_{50}	44.7	31.3	33.8	47.0	43.7	42.7
mAP	22.9	16.0	18.4	23.5	23.6	20.4

Table 5.1: The per-class AP_{50} results of the NYUDv2 19-class object detection benchmark for the baseline and depth-aware Faster R-CNN FPN models. The overall mAP_{50} and mAP are displayed in the last two rows for each model.

Overall the dual backbone approach using the learned concatenation operation yielded the highest mAP_{50} . While not quite achieving the *ideal model*, one which matches or outperforms the RGB and HHA baselines on all classes, the majority of categories benefited from depth features. Notably, there are large performance gains in the door, dresser, night-stand, pillow, and bathtub classes, indicating that the structural features provided by depth, without texture information, leads to disambiguation of class predictions and more accurate localization for these classes. For the pillow class, since pillows exhibit a large variation of textures due to different covers, the improved performance on this class for all the depth-aware models tested is encouraging.

The Faster R-CNN FPNs using fusion module approaches f_{Res} and f_{Gated} trail the RGB baseline in terms of mAP_{50} , indicating a degradation in performance. Both the f_{Res} and f_{Gated} modules use a larger number of parameters than the learned concatenation approach, and so this may have contributed to overfitting on the NYUDv2 dataset. For the f_{Res} module, a single bottleneck operation input with the RGB features and fused RGB/HHA features may not have been sufficient to learn the identity function. Future research could explore the effect of reducing the intermediate dimension of the multi-modal bottleneck layer by a larger factor, and chaining more bottlenecks in sequence. The f_{Gated} module may have benefited from channel-wise reduction, as in the $v2$ module described in Section 4.1, to reduce the parameter count. However the f_{Res} approach does outperform all other models using the more robust mAP metric, although outperforming the f_{LC} by a statistically insignificant margin, indicating more precise overall localization for the predicted bounding boxes at the cost of less detections. Further, data augmentation techniques could potentially have reduced the overfitting issues seen on the NYUDv2 dataset.

Interestingly, the HHA trained Faster R-CNN FPN outperforms the equivalent model trained on depth images in almost all categories with large performance increases seen in many classes, indicating that surface normal and height information is indeed valuable for object detection. The HHA model outperforms all models on the bed class, demonstrating that the large amount of textural information for beds, such as the color of the blankets, is successfully abstracted with depth features for more accurate localization.

Performance on the desk class is low, where desks are often mistakenly labeled as tables due to their visual similarity. The box class suffered from annotation quality and ambiguous types of boxes considered. The monitor and tv class are very close in appearance, resulting in false positives. Strong performing classes were those with consistent annotation quality and visual appearance such as the toilet, sink, night-stand and chair classes. See Figure 5.2 for sample detections on these classes. The per-class AP_{50} of each Faster R-CNN FPN model tested on the SUN RGBD dataset is displayed in Table 5.2.



Figure 5.2: Strong detections from the Faster R-CNN FPN f_{LC} model on the NYUDv2 dataset.

Class	Baseline RGB Faster R-CNN FPN	Baseline Depth Faster R-CNN FPN	Baseline HHA Faster R-CNN FPN	Proposed Faster R-CNN FPN f_{LC}	Proposed Faster R-CNN FPN f_{Res}	Proposed Faster R-CNN FPN f_{Gated}
bathtub	39.3	39.6	43.9	49.9	42.1	45.1
bed	72.7	75.3	78.1	77.2	79.7	79.6
bookshelf	50.0	25.3	27.5	46.9	46.1	47.0
box	11.9	6.7	6.2	16.8	11.9	11.8
chair	67.1	60.2	60.8	68.3	64.9	64.4
counter	40.6	34.3	40.3	43.7	43.8	43.2
desk	13.1	4.9	9.9	17.2	18.3	16.9
door	41.6	14.3	14.9	42.8	39.3	37.3
dresser	26.5	17.6	15.2	31.6	25.7	30.8
garbage-bin	66.0	49.9	55.0	66.0	66.1	66.3
lamp	60.2	50.8	48.4	62.9	56.6	54.3
monitor	36.0	16.9	14.2	42.3	26.9	29.5
nightsand	52.9	53.5	51.9	58.0	55.6	60.6
pillow	60.5	61.2	61.2	64.4	61.2	65.0
sink	68.2	60.8	61.3	70.9	67.7	69.3
table	47.6	43.1	44.2	49.7	52.3	51.0
tv	34.3	19.1	25.6	46.7	34.8	38.0
toilet	84.5	78.4	78.7	89.2	85.4	83.8
mAP_{50}	49.6	40.6	42.1	53.5	49.6	50.8
mAP	28.9	23.0	23.7	30.4	28.7	28.4

Table 5.2: The per-class AP_{50} results on the 19-class SUN RGBD object detection benchmark for the baseline and proposed depth-aware Faster R-CNN FPN models.

Once again, the dual backbone Faster R-CNN FPN f_{LC} model outperforms the RGB baseline by a significant margin, this time outperforming each class-level AP_{50} on all classes except for the bookshelf class. The other fusion models f_{Res} and f_{Gated} matched or outperformed the RGB model on the mAP_{50} metric, indicating that more training data did indeed benefit the more parameter heavy fusion modules. In general, the more intricate learned fusion modules did not outperform the simple learned concatenation operation.

Mirroring the NYUDv2 dataset results, the HHA model outperformed the depth counterpart on the SUN RGBD dataset, and outperformed the RGB baseline on the bed, bathtub, and pillow classes. The strongest detections for the Faster R-CNN FPN f_{LC} class were output on the toilet, sink, and chair classes, with examples shown in Figure 5.3. Failure cases frequently involved false positives due to erroneous class labels. This can be explained by the wide array of object types not in the 19 classes as well as a large variation in indoor setting appearance within the dataset; samples of false positives are shown in Figure 5.4, where in the left photo a window is mistaken for a door, and on the right the top of a treadmill is mistaken for a chair.



Figure 5.3: Strong detections from the Faster R-CNN FPN f_{LC} model on the SUN RGBD dataset.

The mean average recall for the predicted bounding boxes and region proposals on the NYUDv2 and SUN RGBD datasets are shown in Table 5.3 and Table 5.4. For detection, the average recall mAR^{10} , mAR^{100} values are the mean average recall for a maximum of 10, 100 detections respectively, regardless of confidence value. The metrics $R_{0.5}$ and AR_{RP} denote the region proposal recall for the IoU value 0.5, and mean average recall respectively. See Section 2.2.1 for a formal

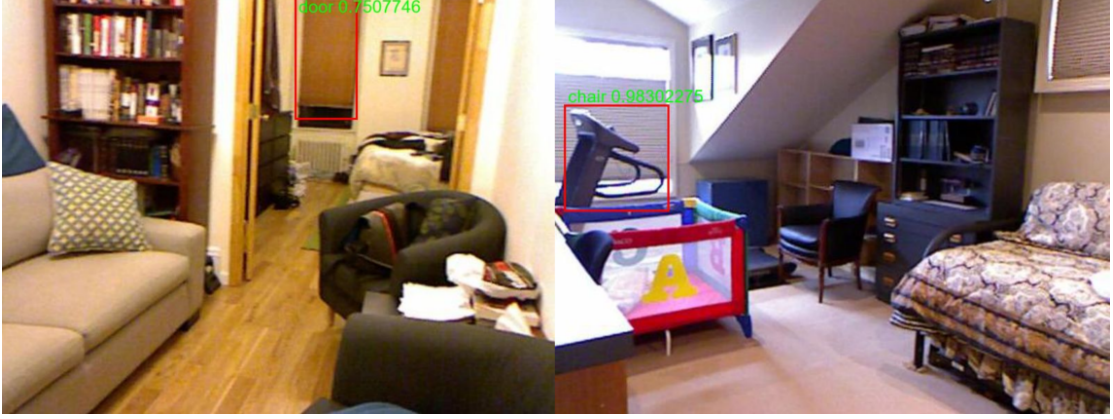


Figure 5.4: False positives from the Faster R-CNN FPN f_{LC} model on the SUN RGBD dataset.

definition of these metrics.

Model	mAR^{10}	mAR^{100}	$R_{0.5}$	AR_{RP}
Baseline RGB Faster R-CNN FPN	34.5	43.2	94.0	59.6
Baseline Depth Faster R-CNN FPN	30.5	37.6	90.2	55.5
Baseline HHA Faster R-CNN FPN	31.9	36.2	88.7	54.5
Proposed Faster R-CNN FPN f_{LC}	36.8	46.1	95.8	57.7
Proposed Faster R-CNN FPN f_{Res}	36.6	42.3	94.2	58.0
Proposed Faster R-CNN FPN f_{Gated}	33.9	44.5	96.0	56.8

Table 5.3: Average recall results for object detection and region proposal generation for the baseline and depth-aware Faster R-CNN models on the NYUDv2 test dataset.

Model	mAR^{10}	mAR^{100}	$R_{0.5}$	AR_{RP}
Baseline RGB Faster R-CNN FPN	43.1	52.1	95.5	62.5
Baseline Depth Faster R-CNN FPN	37.5	47.3	91.2	58.0
Baseline HHA Faster R-CNN FPN	37.3	46.0	91.9	57.9
Proposed Faster R-CNN FPN f_{LC}	44.2	53.4	96.3	60.7
Proposed Faster R-CNN FPN f_{Res}	43.4	48.6	93.6	61.0
Proposed Faster R-CNN FPN f_{Gated}	42.8	47.1	93.5	60.0

Table 5.4: Average recall results for object detection and region proposal generation for the baseline and depth-aware Faster R-CNN models on the SUN RGBD test dataset.

Overall, on both datasets, the Faster R-CNN FPN with f_{Res} and f_{LC} fusion modules models achieved higher average recall, and interestingly a higher recall for region proposals at the IoU value of 0.5. However the RGB model achieved a better mean average recall for region proposals on both datasets. This indicates that in general the depth features fused with RGB features input to the RPNs of the depth-aware networks resulted in more candidate regions that did indeed contain objects, at the cost of a precise localization around those objects, compared to the RPN trained only with RGB features.

The performance on the NYUDv2 and SUN RGBD datasets of the methods presented in the RGBD object detection literature review are compared to the best performing model in this thesis in Table 5.5 and Table 5.6 (for models that have been tested on these benchmarks).

Model	mAP_{50}
RGBD R-CNN [7]	32.5
Proposed Faster R-CNN FPN f_{LC}	47.0
RGBD Fast R-CNN (Supervision Transfer) [43]	49.1
Correlated Detection Net [44]	49.5
CMAC [20]	52.3

Table 5.5: Comparison of the best performing model in this work, the proposed Faster R-CNN FPN f_{LC} architecture, with state-of-the-art models on the NYUDv2 19-class object detection benchmark.

Examining model comparison on the NYUDv2 dataset in Table 5.5, the Faster R-CNN FPN f_{LC} model trails the three best performing models in the literature, however both the model of Gupta et al. [43] and the Correlated Detection Net [44] make use of supervision transfer on the depth backbone, discussed in Section 3.1. On the larger SUN RGBD dataset however, Faster R-CNN FPN f_{LC} slightly outperforms the previous state-of-the-art CMAC, indicating that overfitting may have been the performance bottleneck on the smaller NYUDv2 dataset.

In Table 5.7, the framerate measured in frames per second (fps) of each model using a batch size of 10 images, computing output for 1000 images, is shown. The Faster R-CNN FPN f_{LC} model is able to attain the 30 fps threshold needed for real-time use, while the other fusion approaches were not. The f_{Gated} model is

Model	mAP_{50}
RGBD Fast R-CNN (Supervision Transfer) [43]	43.8
Correlated Detection Net [44]	49.5
CMAC [20]	52.9
Proposed Faster R-CNN FPN f_{LC}	53.5

Table 5.6: Comparison with the best performing model in this work, Faster R-CNN FPN f_{LC} , with the state-of-the-art models tested on the 19-class SUN RGBD object detection benchmark. Note that the results reported for RGBD R-CNN (with supervision transfer) [43] are obtained from Li et al. [20] since the original work did not test on the SUN RGBD dataset.

much slower than the RGB baseline.

Model	fps
Baseline RGB Faster R-CNN FPN	47.6
Proposed Faster R-CNN FPN f_{LC}	30.2
Proposed Faster R-CNN FPN f_{Gated}	12.4
Proposed Faster R-CNN FPN f_{Res}	26.9

Table 5.7: Comparing the inference speed of baseline and the proposed depth-aware Faster R-CNN FPN models trained.

Overall, the Faster R-CNN FPN f_{LC} model using a novel intermediate fusion approach is an effective algorithm for combining RGB and HHA features, resulting in improved results compared to an RGB baseline, and demonstrating competitive results when compared to other models from the literature.

5.2.2 Instance Segmentation

5.2.2.1 Implementation Details

For the task of instance segmentation, a comparison was made between the proposed depth-aware Mask R-CNN FPN described in Section 4.2.2 and the standard Mask R-CNN model introduced by He et al. [17]. The RGBD approach inputs HHA and RGB image pairs to two separate backbones, combining the feature map outputs from each backbone using the fusion modules described in Section 4.1. These fused feature maps are then input to a FPN, after which the standard

RPN and RoI heads from Faster R-CNN are used for bounding box detection, which run in parallel to the mask-branch from the Mask R-CNN architecture to predict a segmentation mask for each non-background box detected. The models trained were:

- Baseline single input-type Mask R-CNN FPN models. Three such models will be tested with RGB, depth and HHA images as inputs respectively.
- The proposed dual backbone Mask R-CNN FPN model, input with RGB and HHA image pairs. For each fusion module type, f_{LC} , f_{Res} , or f_{Gated} , a different model is trained using a single fusion operation. These will be referred to as the Proposed Mask R-CNN FPN X architectures for $X \in \{f_{LC}, f_{Res}, f_{Gated}\}$.

ResNet-50 models were used for each backbone. The PyTorch implementation of Mask R-CNN was used, the code of which naturally extends the *General R-CNN* code available in the Torchvision library code by adding a mask branch and introducing additional hyperparameters. Similar hyperparameters were used as in Section 5.1.1.1 for the Faster R-CNN FPN model, however the Multi-Scale RoI Align grid size was increased from 7×7 to 14×14 in order to gain more spatial information about the RoIs. The mask branch convolutional filter weights were initialized using the Kaiming uniform initialization scheme. The RPN and FPN weights were initialized from pre-training on the MS-COCO 2015 object detection training dataset, using instance masks as well as bounding boxes. For the NYUDv2/SUN RGBD datasets, the following models were trained:

- A baseline RGB Mask R-CNN FPN with the backbone inheriting pre-trained weights from a model trained on the MS-COCO 2015 object detection dataset. The backbone weights were not frozen for training on the NYUDv2 dataset, but were frozen for the SUN RGBD dataset due to performance differences exhibited on early experiments. This can be most likely attributed to the poorer annotation quality of the ground truth instance masks in the SUN RGBD dataset.
- A Mask R-CNN FPN trained with HHA input with the same initialization configuration as the RGB model, without any weights frozen. This is repeated for depth image inputs. These models are the baseline HHA and depth Mask R-CNN FPNs.
- A dual backbone Mask R-CNN FPN using the f_{LC} fusion module, with

backbones for each modality inheriting weights pre-trained on MS-COCO. The RGB backbone weights were frozen.

- Dual backbone Mask R-CNN FPNs with both backbones pre-trained on the SUN RGBD using the f_{Res} , f_{Gated} fusion modules respectively, with the RGB backbone weights frozen. For the SUN RGBD dataset, the f_{Gated} approach used the same setup as the f_{LC} approach.

In total 6 models were trained: three single input-type models, and three dual backbone approaches using each respective fusion module. Images and instance masks were resized to a resolution of 600×800 , the latter using the nearest neighbors algorithm. At test time, the predicted instance mask tensors with confidence value larger than or equal to 0.5 were interpolated to the original image size using bilinear interpolation, after which they were binarized at each pixel location with a threshold value of 0.5. The instance masks for the NYUDv2 dataset were obtained from Dr. Gupta’s GitHub repository [81]. The instance masks provided publicly are unfortunately not the same as those used in RGBD R-CNN [7], and the follow-up work [43], but rather only a subset of the 19 categories which are also part of the 40-class semantic segmentation benchmark. These categories are: bathtub, bed, bookshelf, box, chair, counter, desk, door, dresser, lamp, night-stand, pillow, sink, sofa, table, television, toilet (17 classes). For this reason a direct comparison with the aforementioned works is not possible, although many of the categories are the same.

The instance masks provided by the SUN RGBD dataset [82] suffered from quality issues when considering identical class choices from the 19-class object detection benchmark. These issues included incorrect class labeling, and ambiguous annotation style such as decomposing a dresser into each of its drawers in some annotations, and leaving the entire dresser annotated as a single mask in others. For this reason, using visual inspection, 13 of the most coherent classes were chosen: bathtub, bed, chair, table, door, garbage-bin, lamp, monitor, pillow, sofa, toilet, sink, desk. This 13-class benchmark is original to this thesis. Small instance masks with area less than 32×32 pixels in the original image resolution were removed from the testing/training datasets due to very poor annotation quality. A class mapping similar to the object detection labeling was used for very similar instance class labels, given in Appendix B.

The loss function used for training follows Equation 2.31, given in Section 2.6. The optimizer algorithm used was stochastic gradient descent, with a momentum

of 0.9 and weight decay of 0.0001 as in Mask R-CNN [17], with the learning rate initialized to 0.003. The learning rate scheduler consisted of a multiplication of the learning rate by factor of 0.6 every 4 epochs, used for both datasets.

5.2.2.2 Results and Analysis

The per-class AP_{50} of each Mask R-CNN FPN model tested on the NYUDv2 17-class benchmark is displayed in Table 5.8.

Class	Baseline RGB Mask R-CNN FPN	Baseline Depth Mask R-CNN FPN	Baseline HHA Mask R-CNN FPN	Proposed Mask R-CNN FPN f_{LC}	Proposed Mask R-CNN FPN f_{Res}	Proposed Mask R-CNN FPN f_{Gated}
bathtub	11.0	8.6	12.5	3.4	7.8	7.8
bed	60.0	65.3	61.7	58.8	65.6	62.2
bookshelf	9.6	5.2	6.7	11.7	6.6	11.0
box	4.1	2.6	2.9	8.1	4.3	5.0
chair	44.5	44.0	41.7	48.3	48.6	44.2
counter	22.9	17.8	19.2	37.1	26.5	43.5
desk	1.0	1.7	1.4	2.9	5.5	4.0
door	20.2	6.3	7.1	23.3	16.5	22.7
dresser	44.5	32.7	27.2	40.0	44.2	35.9
lamp	46.7	40.3	38.9	40.3	44.2	36.9
night-stand	40.2	30.2	41.4	54.8	38.5	44.1
pillow	38.4	36.3	45.1	49.8	50.5	43.9
sink	53.1	34.8	47.0	59.2	51.5	59.1
sofa	42.1	42.5	38.0	48.5	47.3	46.9
table	12.7	10.8	16.1	17.2	16.9	14.9
television	42.9	17.2	18.8	37.6	26.8	38.9
toilet	79.7	59.6	65.4	78.7	72.5	73.2
mAP_{50}	33.5	26.8	28.9	36.5	33.8	34.4
mAP	15.5	12.1	12.3	16.3	15.6	16.1

Table 5.8: The per-class AP_{50} results for the baseline and proposed depth-aware Mask R-CNN FPN models tested on the NYUDv2 17-class instance segmentation benchmark. The last two rows show the overall mAP_{50} and mAP results per model.

Examining Table 5.8, the Mask R-CNN FPN f_{LC} model outperforms the RGB baseline in both the mAP_{50} and mAP metrics, achieving large performance gains on the night-stand, pillow, and counter classes. The counter and pillow classes can be observed to display large textural variations, and so the abstraction of their shape provided by HHA images is the most likely reason for this performance increase. The Mask R-CNN FPN f_{LC} does not achieve the ideal model, being significantly outperformed on the television and lamp categories by the RGB baseline. The fine details of the lamp objects are difficult to capture accurately at a pixel level by a depth sensor. The television class may blend into the background in a depth image, not being distinguishable enough from the wall or the

supporting structure behind it, and so none of the fusion operations were able to successfully favor RGB features for this class.

The dual backbone models using the f_{Res} and f_{Gated} fusion modules also outperformed the RGB baseline using both the mAP_{50} and mAP metrics, lagging behind the f_{LC} fusion approach. The single input-type HHA model outperformed the RGB baseline on the bed and pillow classes, again reinforcing the notion that HHA images successfully abstract textural information from these objects for more accurate localization and recognition. Notably the single HHA input model outperformed the depth model on the majority of classes, trailing performance slightly on the bed, chair and lamp class, and dresser classes. This may be due to inaccuracies in the estimated surface normals used in the gravity channel, for which the lamp stand, chair legs, bed and dresser edges require very precise information for accurate instance mask boundary predictions.

For the f_{LC} fusion approach, strongly performing classes included the sink, night-stand, sofa and toilet class due to their consistent appearance and annotation quality. The desk class exhibited false positive issues, often being predicted as a table. The poor annotation quality/consistency of the door, box, bookshelf and bathtub classes resulted in a low AP_{50} in all models tested. For example, in some annotations the individual shelves of a bookshelf would not be considered part of the mask, while in others the entire structure would be considered one large mask block, resulting in a splotchy prediction, see Figure 5.6. Strong instance mask segmentations are shown in Figure 5.5. In general, even theses masks failed to establish pixel-perfect boundaries.



Figure 5.5: Predicted instance masks from the Proposed Mask R-CNN FPN f_{LC} model on the NYUDv2 dataset

In Table 5.9, Mask R-CNN FPN f_{LC} is compared to RGBD R-CNN [7], and

the follow-up work of Gupta et al. that makes use of supervision transfer [43], although this is not technically a fair comparison since the categories used are different (see Section 5.2.2.1). The proposed Mask R-CNN FPN f_{LC} outperforms RGBD R-CNN and trails the performance of the follow-up work.

Model	mAP_{50}
RGBD R-CNN [7]	32.1
Proposed Mask R-CNN FPN f_{LC}	36.5
RGBD R-CNN (Supervision Transfer) [43]	42.1

Table 5.9: Comparing the Proposed Mask R-CNN FPN f_{LC} with other models in the literature on the NYUDv2 testing set.



Figure 5.6: A failure case from the Mask R-CNN FPN f_{LC} model on the NYUDv2 dataset.

The performance of the single input-type and proposed depth-aware Mask R-CNN FPN models on the SUN RGBD 13-class benchmark is displayed in Table 5.10.

Class	Baseline RGB Mask R-CNN FPN	Baseline Depth Mask R-CNN FPN	Baseline HHA Mask R-CNN FPN	Proposed Mask R-CNN FPN f_{LC}	Proposed Mask R-CNN FPN f_{Res}	Proposed Mask R-CNN FPN f_{Gated}
bathtub	21.4	24.8	24.3	33.3	28.4	28.6
bed	33.4	37.3	36.1	34.3	31.1	29.2
chair	49.8	45.1	42.2	49.0	46.8	44.6
table	30.7	28.2	30.1	31.4	31.0	29.9
door	24.0	7.6	7.6	28.9	27.9	25.9
garbage-bin	7.7	4.4	8.2	21.5	11.9	5.1
lamp	40.9	36.4	35.6	40.3	37.6	36.3
monitor	50.3	30.9	35.7	51.8	48.5	45.7
pillow	32.9	29.3	34.0	36.4	33.2	30.7
sofa	34.4	29.8	37.0	41.3	38.8	32.1
toilet	70.6	56.7	59.2	70.1	68.0	59.7
sink	56.5	43.3	42.4	56.5	52.8	50.0
desk	2.6	0.3	0.2	4.3	1.5	3.8
mAP_{50}	34.9	28.8	30.3	38.4	35.2	32.2
mAP	18.1	13.7	14.5	19.6	17.3	16.7

Table 5.10: The per-class AP_{50} results for baseline and proposed depth-aware Mask R-CNN FPN models tested on the SUN RGBD 13-class instance segmentation benchmark.

Tested on the SUN RGBD dataset, the proposed Mask R-CNN FPN f_{LC} model outperforms the RGB baseline by a substantial margin with regards to the mAP_{50} and mAP metrics, being outperformed class-wise only on the chair, lamp, and toilet classes. The fine details of the chair legs and lamp stand may sometimes be too inaccurate in the HHA/depth images to meaningfully contribute to localization, although the performance disparity is small. The f_{Res} model is able to slightly outperform the RGB baseline in terms of mAP_{50} , but not overall mAP . The f_{Gated} model fails to outperform the RGB baseline; the discussion of future modifications to both these modules mentioned in Section 5.2.1.2 applies in the instance segmentation context as well.

The framerate measured in frames per second (fps) of each model type is shown in Table 5.11, using a batch size of 10 images, computing output for 1000 images. The added running time of the dual backbone models using the f_{LC} and f_{Res} fusion modules is less significant than in the case of object detection, while the gated approach is again much slower than the RGB baseline. In general, Mask R-CNN is not a real-time instance segmentation approach.

Overall, the depth-aware Mask R-CNN FPN model using intermediate fusion with the learned concatenation module is successful at fusing features from the HHA images with the RGB features for the task of instance segmentation. The residual fusion module offered insignificant performance gains at a large cost in

Model	fps
Baseline RGB Mask R-CNN FPN	18.7
Proposed Mask R-CNN FPN f_{LC}	12.6
Proposed Mask R-CNN FPN f_{Gated}	5.5
Proposed Mask R-CNN FPN f_{Res}	16.5

Table 5.11: Comparing the inference speed of the baseline and proposed depth-aware Mask R-CNN FPN models trained.

terms of additional model parameters, while the gated fusion module did not improve performance at all on the SUN RGBD dataset.

5.2.3 Semantic Segmentation

5.2.3.1 Implementation Details

The RGBD extension of the DeepLabv3 proposed in Section 4.2.3 consists of a late fusion of the output of two backbones input with RGB and HHA images. The fused output is fed into the DeepLabv3 head. For comparison, single input-type DeepLabv3 models were considered, where overall the two types of models trained were:

- The baseline DeepLabv3 model input with only RGB, HHA, and depth images.
- The dual backbone DeepLabv3 model using late fusion, making use of the fusion modules f_{LC} , f_{Gated} and f_{Res} . These will be referred to as the Proposed DeepLabv3 X architectures for $X \in \{f_{LC}, f_{Res}, f_{Gated}\}$.

The backbones used were the ResNet-101 model with dilated convolutions and reduced downsampling discussed in Section 2.7, this time opting for the heavier 101 variant due to the availability of model weights pre-trained on the MS-COCO Stuff dataset, provided by the Torchvision library. The DeepLabv3 network head implementation used was also provided by the public Torchvision library. The weights of the head were randomly initialized with the Kaiming uniform initialization scheme. The dilation factors used in the atrous spatial pooling pyramid were (12, 24, 36). The models trained for the NYUDv2/SUN RGBD dataset were:

- Single input-type RGB, HHA and depth DeepLabv3 networks trained from a backbone pre-trained on a subset of the classes from the MS-COCO Stuff

dataset. These are the baseline RGB, HHA and depth DeepLabv3 models, respectively.

- The proposed dual backbone DeepLabv3 networks, one for each fusion module. For the NYUDv2 dataset, each model inherits the weights of the RGB and HHA backbones trained on the NYUDv2 dataset with no network weights frozen. For the SUN RGBD dataset, each model’s backbones inherit weights pre-trained on the MS-COCO Stuff dataset with no weights frozen.

During training the images and segmentation masks were interpolated to the resolution 600×800 , with the segmentation mask resized using the nearest neighbors algorithm. At test time, the predicted segmentation mask is re-sized to the original image resolution, again using the nearest neighbors algorithm, for computing the mIoU and accuracy metrics. For the NYUDv2 dataset, the 40-class ground truth labels were used, provided by the code of Dr. Gupta’s GitHub repository [81]; the 37-class ground truth labels were used for the SUN RGBD dataset, provided by the v2 annotation [82].

The loss function used for training was the average cross-entropy, computed at each spatial location of the prediction tensor over the channel index. During training, loss computation for pixels with a ground truth value of 0 corresponding to the *unknown* class was ignored. The SUN RGBD dataset semantic segmentation annotation had many pixels belonging to the *unknown* class, where predicting this class led to degraded performance in early experiments. The optimizer algorithm employed was stochastic gradient descent using the same parameters/learning rate scheduler as in Section 5.2.1.1 (for Mask R-CNN), except for a 0.004 learning rate initialization for the SUN RGBD dataset for faster network convergence.

5.2.3.2 Results and Analysis

The per-class mIoU of the baseline and proposed depth-aware DeepLabv3 models on the NYUDv2 40-class benchmark is shown in Table 5.12. A more detailed performance summary of the models tested on the NYUDv2 dataset, using the metrics described in Section 2.2.3, is given in Table 5.13.

Each dual backbone DeepLabv3 model outperforms the RGB baseline for every metric used, a convincing result. In particular, the pillow, mirror, ceiling, toilet, and bathtub classes saw large performance increases, while most other classes saw modest gains. The RGB baseline was able to outperform all other

Class	Baseline RGB DeepLabv3	Baseline Depth DeepLabv3	Baseline HHA DeepLabv3	Proposed DeepLabv3 f_{LC}	Proposed DeepLabv3 f_{Gated}	Proposed DeepLabv3 f_{Res}
wall	74.0	65.8	65.8	76.8	77.2	76.1
floor	81.0	79.8	80.0	86.0	85.8	86.2
cabinet	54.2	47.7	49.6	55.9	58.2	53.4
bed	62.7	70.2	70.5	72.0	72.5	70.3
chair	62.3	57.6	55.6	64.3	64.1	64.1
sofa	61.4	59.5	58.2	65.0	64.2	63.9
table	41.0	38.0	36.6	43.3	43.1	44.6
door	29.0	14.9	16.0	27.4	27.3	28.3
window	42.2	32.4	29.7	44.4	42.8	44.6
bookshelf	38.8	29.9	31.7	41.7	42.9	40.8
picture	52.4	26.5	30.3	54.3	55.0	56.2
counter	53.1	48.2	51.6	59.6	59.7	58.3
blinds	53.3	21.7	24.8	52.8	50.8	55.9
desk	22.3	11.9	18.7	24.6	21.2	20.7
shelves	10.3	7.0	9.9	13.6	12.4	9.5
curtain	46.6	43.5	32.6	49.3	49.6	52.3
dresser	39.2	30.3	37.4	41.1	43.2	38.8
pillow	41.5	40.7	43.3	49.5	50.3	45.1
mirror	27.5	35.7	37.0	38.7	35.0	37.6
floor mat	27.1	4.0	17.0	37.7	38.9	34.8
clothes	21.2	15.4	14.5	22.7	23.6	22.4
ceiling	48.9	66.0	70.1	68.5	71.1	69.7
books	25.4	12.6	15.0	22.2	26.8	28.1
fridge	35.8	23.0	19.8	32.3	32.4	32.8
tv	56.0	34.6	34.0	57.2	52.3	59.8
paper	24.3	9.4	3.8	27.2	24.3	25.2
towel	34.7	30.7	23.9	36.9	37.9	44.1
shower curtain	30.6	28.8	34.7	28.5	26.9	40.3
box	10.7	4.9	5.4	14.7	12.7	11.3
whiteboard	51.7	3.5	5.6	57.9	61.1	63.1
person	80.3	63.5	48.3	78.3	80.3	78.9
night-stand	30.2	34.9	36.7	40.6	39.1	38.9
toilet	59.9	65.9	63.0	66.8	63.9	58.6
sink	42.1	43.9	37.2	50.8	50.1	49.9
lamp	40.6	35.5	32.4	46.9	44.9	43.6
bathtub	35.5	39.6	45.3	53.4	53.5	53.0
bag	7.9	2.7	0.5	14.9	12.8	10.5
other struct	17.1	11.8	10.2	16.0	18.7	19.6
other furniture	13.7	13.0	15.0	17.2	18.7	18.5
other prop	35.7	27.9	28.7	36.5	35.9	36.3
mIoU	40.7	33.3	33.5	44.7	44.5	44.7

Table 5.12: The per-category mIoU of baseline and proposed depth-aware DeepLabv3 models tested on the NYUDv2 40-class semantic segmentation benchmark. Overall model mIoU is shown in the bottom row.

Model	mIoU	fwIoU	Acc	mAcc
Baseline RGB DeepLabv3	40.7	55.1	69.4	53.5
Baseline Depth DeepLabv3	33.3	48.8	63.7	44.3
Baseline HHA DeepLabv3	33.5	49.2	64.0	45.5
Proposed DeepLabv3 f_{LC}	44.7	58.5	72.0	57.6
Proposed DeepLabv3 f_{Gated}	44.5	58.7	71.9	58.2
Proposed DeepLabv3 f_{Res}	44.7	58.1	71.6	58.4

Table 5.13: The performance summary of the baseline and proposed depth-aware DeepLabv3 models on the NYUDv2 40-class semantic segmentation benchmark.

models on the door and fridge classes. The door class often suffered from noise in the depth images due to windows, or range issues if the door was open. The fridge class may have been difficult to distinguish using HHA or depth features since it bears structural resemblance to other objects and surfaces, and is mostly distinguished by its color and texture.

Depth-aware models trained with the different fusion modules performed similarly, with no clear winner. Comparing the HHA and depth DeepLabv3 models, the mIoU difference is not statistically significant, indicating less of a benefit to use of one encoding type versus the other for semantic segmentation. Qualitatively, the DeepLabv3 f_{LC} , when compared to the RGB baseline, often displayed more coherent region predictions and clearer region boundaries. In Figure 5.7a the depth-aware DeepLabv3 results in a clearer table segment. In photo Figure 5.7b, the blinds, ceiling, and bed headboard (the other-struct class) segments are more clearly outlined. In photo Figure 5.7c, the dresser supporting the television exhibits much clearer regions, almost directly matching the ground truth image. The structural classes wall, floor, and ceiling contain the most average pixels per



Figure 5.7: A visual comparison of the semantic segmentation masks predicted by the depth-aware DeepLabv3 f_{LC} model compared to the RGB baseline on images from the NYUDv2 dataset. Segmentation masks are overlaid over the original image. From left to right: RGB image, HHA image, ground truth segmentation mask, RGB DeepLabv3 output, DeepLabv3 f_{LC} output.

image, and so the raw accuracy and frequency weighted IoU metrics are inflated by the success on all DeepLabv3 models' performance on these classes. The worst-performing classes were those that had ambiguous labeling or very few training instances such as other-struct, floor mat, and clothes. Since the NYUDv2 dataset

contains a large amount of objects, seen at different viewpoints, with varied scene types, the failure cases of the proposed DeepLabv3 f_{LC} often involved segmenting new object types or viewpoint changes. In these cases the model hallucinates incoherent splotches of patches within a small neighborhood, where each splotch belongs to a different class. This is demonstrated in Figure 5.8a, the refrigerator viewed from behind, is mistaken for several other classes. In Figure 5.8b, a shelf with books and various other items is incoherently segmented, resulting in very unpleasant aesthetics.

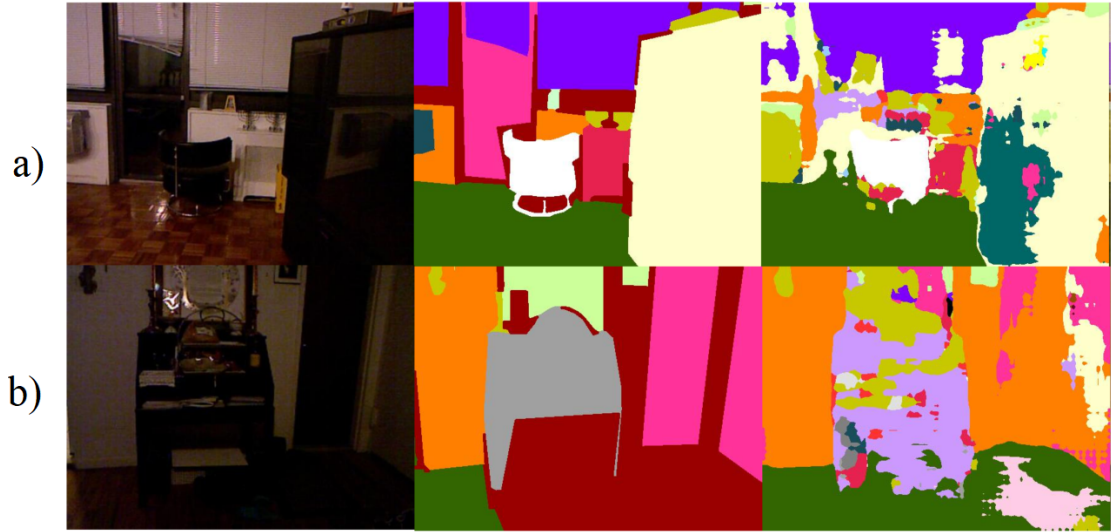


Figure 5.8: Failure cases from the proposed DeepLabv3 f_{LC} model on the NYUDv2 dataset. Left: RGB Image, Middle: Ground Truth Segmentation Mask, Right: DeepLabv3 f_{LC} output.

In Table 5.14, the result of the best performing model on the NYUDv2 dataset, DeepLabv3 f_{LC} , is compared to prominent works from the literature discussed in Section 3.2. Recalling the geometric-centered model vs. image-centered model discussion in Section 4.1, the model of Neumann and Wang [11], 3D-GNN [21], and CFN [63] fall into the category of geometric-centered models since they use the actual depth values to structure feature extraction. The proposed DeepLabv3 f_{LC} outperforms the model of Wang and Neumann and 3D-GNN, while trailing considerably behind the CFN model. This may be due to the fact that CFN uses depth-ranges more so than absolute depth values to structure extraction, which is possibly robust to outliers and small inaccuracies. Overall, the image-centered models ESANet [84] and RDFNet achieved the best performance. RDFNet, being the most similar to the proposed depth-aware DeepLabv3 f_{LC} model, clearly benefited from a stronger RGB baseline where Lee et al. [61] report a pure RGB

RefineNet achieving 46.5% mIoU, a significantly stronger baseline compared to the DeepLabv3 model’s 40.7% mIoU shown in Table 5.13. This difference in baseline RGB performance may be due to the multi-resolution fusion operation of the RefineNet architecture, where the DeepLab models do not make use of the intermediate feature maps of the backbone. The RGBD R-CNN [7] trails significantly behind all other models in the literature, most likely due to its use of hand-crafted feature and dependency on the object detection results. The model of Eigen and Fergus [57], an image-centered approach, uses less convolutional layers, and does not use pre-training for the weights of the convolutional layers used in the second and third scales, and so trails significantly behind more modern approaches. The LSTM-CF model [60], another image-centered approach, is able to leverage the additional vertical and horizontal context using LSTM modules, falling short of the state-of-the-art, but outperforming the proposed DeepLabv3 f_{LC} model.

Model	mIoU	Acc
RGBD R-CNN (Semantic Segmentation) [7]	31.3	60.3
Eigen and Fergus [57]	34.1	65.6
FCN [36]	34.0	65.4
DeepLabv1 with depth-aware convolution [11]	41.5	69.6
3D-GNN [21]	43.1	-
Proposed DeepLabv3 f_{LC}	44.7	72.0
CFN [63]	47.7	-
LSTM-CF [60]	49.4	-
RDFNet [61]	50.1	76.0
ESANet [68]	51.58	-

Table 5.14: Comparing the proposed DeepLabv3 f_{LC} model performance to state-of-the-art works on the NYUDv2 40-class semantic segmentation benchmark.

In Table 5.15, the per-class mIoU of the baseline and proposed depth-aware DeepLabv3 models tested on the SUN RGBD dataset is given, followed by a performance summary using a diversity of metrics in Table 5.16.

Class	Baseline RGB DeepLabv3	Baseline Depth DeepLabv3	Baseline HHA DeepLabv3	Proposed DeepLabv3 f_{LC}	Proposed DeepLabv3 f_{Gated}	Proposed DeepLabv3 f_{Res}
wall	79.1	72.7	72.0	80.2	80.0	80.3
floor	89.0	87.7	87.1	90.4	90.3	90.5
cabinet	45.8	39.0	41.0	48.0	47.6	47.8
bed	69.2	70.5	71.0	72.8	72.3	73.2
chair	72.1	67.4	65.6	72.6	72.8	73.5
sofa	54.1	48.9	48.3	52.2	56.0	56.9
table	53.3	50.6	49.9	53.3	54.3	55.0
door	42.9	24.6	24.9	43.8	43.4	43.7
window	51.3	37.9	37.1	52.8	53.8	52.6
bookshelf	36.8	24.0	30.5	38.0	38.8	38.1
picture	52.7	30.3	31.3	53.8	54.5	55.3
counter	33.2	29.6	27.2	34.6	33.6	34.5
blinds	29.6	16.7	15.7	34.8	35.4	33.6
desk	28.3	22.5	23.7	27.1	27.8	28.4
shelves	9.8	10.4	10.5	8.5	11.2	10.6
curtain	58.6	53.6	49.5	58.3	59.0	58.8
dresser	45.6	35.8	38.0	49.8	46.2	48.4
pillow	38.2	37.6	39.0	41.8	42.2	42.6
mirror	41.0	35.4	39.7	46.3	45.9	47.3
floor mat	0.0	0.0	0.0	0.0	0.0	0.0
clothes	25.9	20.7	23.2	28.1	26.2	27.2
ceiling	68.7	60.7	60.9	72.9	72.0	73.6
books	32.9	24.6	21.9	30.1	31.8	31.2
fridge	41.9	29.8	38.1	44.1	39.5	46.3
tv	51.4	26.3	32.0	51.3	54.5	53.7
paper	27.2	7.3	8.2	27.0	24.7	26.6
towel	27.6	18.7	22.6	30.6	30.6	34.9
shower curtain	9.6	0.6	0.1	0.5	4.7	1.8
box	29.8	19.0	17.8	28.9	28.9	30.2
whiteboard	57.1	29.5	33.9	55.9	58.4	56.8
person	48.3	28.3	29.5	56.0	52.3	52.9
night-stand	20.0	19.7	19.9	21.5	22.8	21.1
toilet	72.3	67.6	62.0	74.4	73.0	74.6
sink	54.0	52.0	49.2	53.5	53.0	54.5
lamp	42.8	38.0	37.2	44.9	43.6	45.2
bathtub	56.4	59.0	57.3	64.7	66.5	63.2
bag	19.7	14.3	15.2	22.0	21.1	21.7
mIoU	43.7	35.4	36.0	45.0	45.1	45.6

Table 5.15: The per-category IoU for the baseline and proposed depth-aware DeepLabv3 networks tested on the SUN RGBD 37-class semantic segmentation benchmark. The overall model mIoU is displayed in the last row.

Model	mIoU	fwIoU	Acc	mAcc
Baseline RGB DeepLabv3	43.7	68.6	80.0	56.1
Baseline Depth DeepLabv3	35.4	63.3	76.1	45.4
Baseline HHA DeepLabv3	36.0	62.9	75.7	47.1
Proposed DeepLabv3 f_{LC}	45.0	69.6	80.6	58.0
Proposed DeepLabv3 f_{Gated}	45.1	69.7	80.6	56.5
Proposed DeepLabv3 f_{Res}	45.6	70.1	81.1	69.4

Table 5.16: The performance summary of the baseline and depth-aware DeepLabv3 models on the SUN RGBD 37-class semantic segmentation benchmark.

Once again, the depth-aware DeepLabv3 models all outperform the RGB baseline, where this time the f_{Res} fusion module demonstrates improved performance on all 4 metrics considered, and notably the f_{Gated} slightly outperforms the f_{LC} fusion module, except for the mean accuracy metric. This performance increase, coupled with the NYUDv2 results shown in Table 5.13, shows that the residual approach developed in Section 4.1 is the most successful fusion module developed in this thesis for fusing RGB and depth features for semantic segmentation. In Figure 5.9, one can observe the fusion of RGB and depth features aids to refine segmentations for almost every class in the scene, most notably the bed, the closed door (in pink), the curtains, the blinds, and the night-stand.



Figure 5.9: Comparing the qualitative output of the RGB DeepLabv3 baseline to the Proposed DeepLabv3 f_{Res} model on an image from the SUN RGBD dataset. Left: RGB Image, Middle: RGB DeepLabv3 output, Right: Proposed DeepLabv3 f_{Res} output.

The RGB baseline slightly outperforms all depth-aware models on the books, paper and shower-curtain classes. The paper class may be difficult to distinguish in HHA images as the depth difference between paper on a wall or on a table is very slight. A similar argument could be made for thinner books. The baseline HHA model outperforms the depth image model very slightly, suggesting that depth images and HHA images offer roughly the same worth for semantic segmentation. The shower-curtain and floor-mat classes had very few training example pixels, and so results on these classes were poor for all models tested.

In Table 5.17, a comparison between the best performing model on the SUN RGBD benchmark from this thesis, the proposed DeepLabv3 f_{Res} model, to prominent works from the literature is shown. The depth-aware DeepLabv3 f_{Res} matches or outperforms all geocentric-models except for CFN [63]. Further the mean accuracy of the DeepLabv3 f_{Res} model outperforms all other models that report this metric. Notably, the best performing work, LSTM-CF [60] is also an image-centered model, where clearly the use of LSTM modules for context is a powerful addition to the standard CNN-based semantic segmentation approach. The

FuseNet model [58] which makes use of the less powerful VGG-16 backbone and does not employ learned fusion between modalities in the fusion operation performs the worst overall, validating the idea that fixed fusion without learnable parameters is sub-optimal for a simple dual backbone approach.

Model	mIoU	mAcc
FuseNet [58]	37.2	50.1
DeepLabv1 with depth-aware convolution [11]	42.0	53.5
Proposed DeepLabv3 f_{Res}	45.6	69.4
3D-GNN [21]	45.9	57.0
RDFNet [61]	47.7	60.1
CFN [63]	48.1	-
ESANet [84]	48.3	-
LSTM-CF [60]	49.4	-

Table 5.17: Comparing the proposed DeepLabv3 f_{Res} model to other models in the literature on the SUN RGBD 37-class semantic segmentation benchmark.

In Table 5.18, the framerate measured in frames per second (fps) of each DeepLabv3 using a batch size of 10 images, computing output for 1000 images, is shown. The running time of the proposed dual backbone DeepLabv3 models are all roughly double the RGB baseline, and do not meet the real-time threshold of 30 fps.

Model	fps
Baseline RGB DeepLabv3	33.6
Proposed DeepLabv3 f_{LC}	16.6
Proposed DeepLabv3 f_{Gated}	14.9
Proposed DeepLabv3 f_{Res}	15.1

Table 5.18: Comparing inference speed of the baseline and proposed depth-aware DeepLabv3 models trained.

Overall, it is demonstrated that the late interaction of the backbone outputs using the residual fusion approach described in Section 4.1 is an effective operation to fuse RGB and HHA features, exhibiting very high accuracy and strong mIoU

for semantic segmentation on both the NYUDv2 and SUN RGBD datasets. The gating approach, while generally outperforming both the RGB baseline and the learned concatenation approach, is not an effective trade-off of increased parameters vs. performance boost.

5.2.4 Panoptic Segmentation

5.2.4.1 Implementation Details

The panoptic segmentation model developed in Section 4.2.4 makes use of components from the Mask R-CNN architecture and the DeepLabv3 model, using a ResNet backbone with dilated convolutions and reduced network stride. The feature maps output from layers Conv2-5 are input to a feature pyramid network, with layers Conv4 and Conv5 downsampled by a factor of 2 and 4 respectively using max pooling. The output of layer Conv5 is input to a DeepLabv3 network head to predict a segmentation mask, while the output of the FPN layers are input to the network heads of Mask R-CNN to predict instances. The instance masks are then fused with the predicted segmentation mask using a post-processing technique described in Section 4.2.4. The depth-aware panoptic segmentation adapts this approach to RGBD inputs by making use of intermediate fusion of the feature maps output by each backbone, as well as late fusion. The ResNet-101 model was used for each backbone. The panoptic models trained and tested were:

- Baseline single input-type RGB and HHA networks with ResNet-101 backbone weights initialized from models pre-trained on the MS-COCO Stuff dataset. Due to the increased performance of the HHA images vs. depth images on all tasks previously considered, a depth image panoptic model was not considered. The RPN and FPN weights are initialized from training on the MS-COCO 2015 object training dataset from a Mask R-CNN model pre-trained for instance segmentation. The DeepLabv3 heads were initialized with Kaiming uniform initialization. The RoI heads were initialized as in Section 5.2.1.1. All weights are trainable.
- The proposed dual backbone network with RGB and HHA backbones both pre-trained initially on the MS-COCO Stuff dataset, and then individually fine-tuned on the NYUDv2 and SUN RGBD datasets. Both sets of backbone weights were frozen during joint-training. The fusion module for the depth-aware FPN was the f_{LC} module, chosen for its performance on the instance segmentation and object detection tasks. The fusion module used

for late fusion was the f_{Res} module, which outperformed both the f_{LC} and f_{Gated} modules on the SUN RGBD 37-class semantic segmentation benchmark using a dual backbone DeepLabv3 model.

Neither dataset has a strict panoptic annotation, and so make-shift panoptic ground truth segmentation masks were constructed for both datasets by pasting instances over top of the segmentation masks. By visual inspection, the instance masks coincide well enough with the segmentation masks so as not to create pasting issues such as conflicting class labels within a single object instance. The resulting segmentation mask pixels for which an instance mask from a things class has been pasted onto are assigned the class label of the instance, and assigned a unique instance number. The pixels corresponding to stuff classes from the original segmentation mask pixels, which have not been modified, are assigned an id of 0. The DeepLabv3 head still requires a semantic segmentation mask, which can be obtained from the panoptic segmentation mask by simply ignoring instance ids. This panoptic annotation for the SUN RGBD and NYUDv2 dataset is completely novel to this thesis, and will be made available to the community at the GitHub page [83]. Some of the stuff classes, which do not have instance annotations, should arguably be considered as things since there are individual instances, such as the picture class, and so this is a quasi-panoptic annotation.

For the NYUDv2 dataset, the 17 instance categories of Section 5.2.2.1 will be used as things. These are pasted over top of the 37-class semantic segmentation annotation provided by the SUN RGBD dataset (recall that the NYUDv2 dataset is a subset of the SUN RGBD dataset), where the categories floor-mat, fridge, shower curtain, were not present in any annotation, and so were removed. Additionally, the classes paper, shelves, books, and bag were removed due to low quality annotation and class imbalance, where these pixels were set to the unknown class. The resulting 30 classes are:

NYUDv2 things (17 classes): bathtub, bed, bookshelf, box, chair, counter, desk, door, dresser, lamp, night-stand, pillow, sink, sofa, table, tv, toilet.

NYUDv2 stuff (13 classes): wall, floor, cabinet, window, picture, blinds, curtain, dresser, towel, box, whiteboard, person, mirror.

For the SUN RGBD dataset, the 13 instance categories from Section 5.2.2.1 will be used as things, pasted over top of the 37-class segmentation masks. Since the

garbage-bin class from the instance segmentation benchmark is not present in the 37 classes of the semantic segmentation benchmark, the resulting panoptic benchmark will have 38 classes, which are:

SUN RGBD things (13 classes): bathtub, bed, chair, table, door, garbage-bin, lamp, monitor, pillow, sofa, toilet, sink, desk classes.

SUN RGBD stuff (25 classes): wall, floor, cabinet, bookshelf, picture, counter, blinds, shelves, curtain, dresser, mirror, floor-mat, clothes, ceiling, books, fridge, tv, paper, towel, shower-curtain, box, whiteboard, person, night-stand, bag.

The optimizer algorithm used was the same as in Section 5.2.2.1 (for Mask R-CNN), with a learning rate of 0.004, and learning rate scheduler consisting of a multiplicative factor of 0.6 every 5 epochs for both datasets. The training loss for the panoptic segmentation networks is of the form

$$\lambda_1 L_{im} + L_{seg} \quad (5.3)$$

where L_{im} is the instance mask loss given by equation 2.30, and L_{seg} is the semantic segmentation mask loss discussed in Section 5.2.3.1. The hyperparameter λ_1 is used to weight the instance segmentation loss. Training on the SUN RGBD dataset, the instance mask loss was multiplied by a weight of 0.5. This was done due to the observation that the semantic segmentation loss decreased much slower than the instance mask loss, potentially resulting in overfitting for instance predictions. Training on the NYUDv2 dataset, λ_1 was set to 1.

5.2.4.2 Results and Analysis

The per-category panoptic quality (PQ) of the models tested on the NYUDv2 dataset is shown in Table 5.19, where the PQ metric is defined in Section 2.2.4. Denote the single input-type models as baseline RGB Panoptic and HHA Panoptic, and the depth-aware network as the Proposed Depth-Aware Panoptic model.

The Depth-Aware Panoptic model outperforms both the RGB baseline and the HHA panoptic model on most classes. The RGB baseline significantly outperforms the HHA and Depth-Aware models on the person and whiteboard classes. Similar to the semantic segmentation results, depth features may not be fine-grained enough for segmenting people, where the HHA model achieves a much lower panoptic quality of 10.6% compared to 65.1% for the RGB model. The

Class	Baseline RGB Panoptic	Baseline HHA Baseline Panoptic	Proposed Depth-Aware Panoptic
wall	66.6	55.5	67.3
floor	76.7	83.0	82.5
cabinet	31.5	29.9	34.2
bed	26.5	31.0	30.2
chair	28.1	30.8	33.0
sofa	23.6	24.2	25.8
table	11.3	13.5	14.3
door	13.9	4.8	12.9
window	22.3	13.5	22.5
bookshelf	11.6	9.6	11.7
picture	44.3	16.4	42.1
counter	15.6	17.2	18.4
blinds	32.2	11.2	34.0
desk	4.0	2.1	5.6
curtain	29.6	14.3	28.4
dresser	16.4	10.8	14.4
pillow	27.5	29.9	29.9
mirror	5.6	12.6	9.2
clothes	6.6	5.4	7.6
ceiling	32.7	44.4	43.7
tv	21.5	12.5	27.0
towel	10.8	7.8	16.1
box	5.5	3.9	6.4
whiteboard	22.3	2.0	14.0
person	65.1	10.6	59.4
night-stand	22.2	23.3	22.5
toilet	27.3	23.5	30.0
sink	19.7	23.9	23.4
lamp	20.1	19.5	23.6
bathtub	10.6	15.2	11.7
Panoptic Quality	22.5	18.0	25.3

Table 5.19: The per-category panoptic quality results on the NYUDv2 30-class benchmark for the baseline and Proposed Depth-Aware panoptic networks.

whiteboard class may not benefit from depth features since it is mostly distinguished from the wall by its texture. Interestingly, the HHA model outperforms other models on the floor and ceiling classes indicating that depth features help to segment the room’s overall structure in the panoptic segmentation setting. The HHA baseline model also outperforms the RGB model on the mirror class, where mirrors exhibit considerable noise in depth images, providing credence to the notion that noise patterns in depth images can sometimes offer discriminative features.

The overall performance bottleneck comes from low performing stuff and things classes, mirroring the results from the instance and semantic segmenta-

tion analysis. One issue that results from the post-processing heuristic used is the problem of pasting a low quality instance mask over a segmentation mask. In Figure 5.10, the instance mask, shown in blue, predicted for the toilet does not properly outline the object boundary, missing pixels at the top of the lid, and the bottom of the base structure. The base semantic segmentation mask, before instance pasting, does segment these pixels properly, but since the region overlap is less than 0.5, this segment is considered a false positive. Hence badly segmented instance masks are very costly. In the case of semantic segmentation, the mIoU metric gives some recognition of the overlap of predicted segments, whereas the panoptic quality metric only assigns a true positive if a threshold of segment IoU of 0.5 has been met.



Figure 5.10: An overlap issue arising from a low quality instance mask predicted for an image from the NYUDv2 dataset.

Successful segmentations arise from images where the Depth-Aware Panoptic model is able to clearly outline the instances, with minimal pasting issues, and the background stuff classes are segmented reasonably well, typically arising from simpler scenes with minimal clutter. In Figure 5.11a, the bed instance, as well as the individual pillow instances are differentiated from one another, while the overall wall, floor and picture stuff classes are outlined well. In Figure 5.11b the individual chair and table instances are distinguished from one another, and in Figure 5.11c the sofa and pillow instances are segmented properly, and integrated relatively smoothly over top of the predicted semantic segmentation mask.

The per-class panoptic quality of the models tested on the SUN RGBD dataset



Figure 5.11: Predicted masks from the Depth-Aware Panoptic model overlaid over RGB images from the NYUDv2 dataset. Left: RGB Image, Right: Predicted panoptic segmentation mask overlaid over the original color image.

is shown in Table 5.20. The depth-aware panoptic model outperforms the RGB baseline, where in this case more training data allows for a more consistent class-wise improvement. The floor-mat and shower curtain classes had very few training examples and so had no matched segments at test time. This outcome is similar to the semantic segmentation results on these classes shown in Section 5.2.3.2, and is also observed in the performance evaluation presented in FuseNet [58] and RDFNet [61]. In the future it may be reasonable to exclude these classes from the training/testing set. On the whole, the proposed RGBD architecture is able to effectively fuse RGB and HHA image features for the task of panoptic segmentation. A set of particularly strong outputs is displayed in Figure 5.12.

The inference speed of the baseline and proposed Depth-Aware Panoptic models, measured in frames per second (fps), is displayed in Table 5.21, computing 1000 images using a batch size of 10.

Class	Baseline RGB Panoptic	Baseline HHA Panoptic	Proposed Depth-Aware Panoptic
wall	67.5	59.0	67.0
floor	80.2	79.5	82.2
cabinet	16.7	15.2	21.1
bed	24.4	30.0	30.0
chair	42.3	37.5	42.6
sofa	25.6	25.2	30.5
table	27.1	24.6	27.9
door	24.9	12.8	25.0
window	29.9	17.6	30.2
bookshelf	11.6	11.0	11.8
picture	33.7	15.9	36.8
counter	12.2	12.3	17.7
blinds	17.9	8.3	22.0
desk	9.2	8.2	7.6
shelves	2.1	1.3	4.3
curtain	28.3	27.4	36.9
dresser	24.5	21.0	29.4
pillow	26.0	23.2	28.2
mirror	15.2	15.3	22.4
floor-mat	0.0	0.0	0.0
clothes	10.3	6.0	11.8
ceiling	51.2	42.6	54.4
books	11.2	4.0	12.5
fridge	5.4	3.2	8.3
tv	39.7	28.1	44.6
paper	8.9	0.9	8.1
towel	12.7	6.7	16.2
shower-curtain	0.0	0.0	0.0
box	8.5	6.0	8.8
whiteboard	31.5	15.4	25.5
person	22.6	5.5	31.1
night-stand	5.8	5.3	7.3
toilet	39.6	30.6	42.3
sink	34.0	27.9	38.0
lamp	28.0	24.0	32.2
bathtub	14.4	15.0	24.5
bag	4.5	4.3	4.3
garbage-bin	8.2	13.9	18.9
Panoptic Quality	26.4	21.0	28.0

Table 5.20: Per-category panoptic quality results on the SUN RGBD 38-class benchmark for the baseline and Proposed Depth-Aware network. Overall model panoptic quality is displayed in the last row.

Model	fps
Baseline RGB Panoptic	13.4
Proposed Depth-Aware Panoptic	10.3

Table 5.21: Comparing the inference speed of the baseline single-input panoptic network with the Proposed Depth-Aware panoptic architecture.



Figure 5.12: Predicted masks from the Depth-Aware Panoptic model on the SUN RGBD dataset overlaid over the RGB image. From left to right: RGB Image, HHA Image, Ground truth panoptic mask overlay, Predicted panoptic mask overlay.

5.2.5 Overall Analysis

After testing each of the proposed models from Section 4.2 using the fusion modules outlined in Section 4.1, the important results are summarized as follows:

- For each task, the best performing depth-aware architecture outperformed the RGB baseline.
- For object detection and instance segmentation, the learned concatenation operation outperformed the more complex residual and gated approaches.
- The region proposal network from Faster R-CNN, input with multi-modal features, achieved a higher recall at the 0.5 IoU threshold than an RGB counterpart.
- For semantic segmentation, the residual fusion module f_{Res} outperformed the learned concatenation and gating operations.
- The single-input baseline models performed better when input with RGB images than with HHA or depth images. This sharply contradicts the result found in RGBD R-CNN [7] where the HHA model achieved 25.2% mAP_{50} compared to the RGB model which achieved 19.7% on the NYUDv2 19-class object detection benchmark.
- Single-input models trained with HHA images outperformed those trained with depth images, with a larger gap in performance observed for object detection and instance segmentation.
- The object detection and instance segmentation RGBD models developed were competitive with state-of-the-art works from the literature, while the semantic segmentation model developed was less competitive, warranting future research into a stronger RGB baseline to modify for RGBD inputs.
- The depth-aware panoptic segmentation architecture outperformed the RGB baseline, and achieved decent results on the NYUDv2 and SUN RGBD benchmarks introduced.

Chapter 6: Conclusion

6.1 Summary and Contributions

In this thesis, a suite of novel RGBD CNN-based architectures for object detection and image segmentation were developed by extending modern computer vision algorithms built for RGB inputs. These architectures outperform their comparable RGB counterpart by making use of the additional information provided by depth. Original fusion modules for combining RGB and depth image features from convolutional neural networks were developed. The resulting models were trained and tested on RGBD dataset benchmarks from the literature, achieving results competitive with prominent works. Altogether, the main contributions of this thesis are:

- A novel dual backbone Feature Pyramid Network framework was developed, allowing for the use of intermediate fusion for the tasks of object detection and instance segmentation using RGBD inputs, where the effectiveness of this approach was validated through extensive experimentation. This framework can be modified in future research to incorporate different fusion approaches to take advantage of varying levels of semantic information provided by the intermediate feature maps output by two CNNs trained on RGB and depth/HHA features.
- The development of two novel fusion modules, the multi-modal residual approach and a gating module, which were then compared with a simple learned concatenation approach. These modules were evaluated in multiple contexts using dual backbones for object detection, instance segmentation, and semantic segmentation, allowing researchers to evaluate the performance to efficiency trade-offs in multiple scenarios compared to a simpler baseline approach. Emphasis was placed on the use of learned fusion approaches rather than fixed mathematical operations.
- The DeepLabv3 architecture was modified to input RGBD images by combining backbone outputs using late fusion for semantic segmentation, tested with a larger diversity of learned fusion approaches than other works in the literature.
- A novel panoptic segmentation architecture was developed to use a modified feature pyramid network and incorporate both the Mask R-CNN network

heads, and the DeepLabv3 head. This panoptic model was then adapted to input RGB and HHA image pairs by using intermediate fusion as input to the feature pyramid network, and late fusion to the DeepLabv3 head, using the new concept of task-specific fusion. The result is one of the first RGBD panoptic segmentation architectures in the literature. The performance of this model can be used as a baseline for comparison by other researchers in the community.

Secondary contributions include:

- The development of a novel 13-class instance segmentation benchmark on the SUN RGBD dataset.
- The development of panoptic segmentation benchmarks on the NYUDv2 and SUN RGBD datasets, unique to this thesis. Benchmarks will be made public at the GitHub page [83].
- A detailed mathematical description of how to convert a depth image to an HHA image, using algorithms from Gupta et al. [22] and analyzing the code of RGBD R-CNN [7]. This is not fully elaborated on in the literature, and will serve as a useful tool for researchers to reference.

6.2 Future Research

A number of avenues for future research should be explored, focusing on three main directions: performance, efficiency, and novelty. Before addressing these concerns, it is worth observing that there is an absence of an official large-scale RGBD panoptic segmentation benchmark in the literature for indoor settings. The Cityscapes dataset [85] currently does have a panoptic annotation as well as disparity images paired with the RGB images provided, however this dataset is focused on outdoor settings with a limited number of classes. Researchers may wish to develop such a benchmark for the large-scale ScanNetv2 dataset [86], for which currently there exists only 2D semantic segmentation and instance segmentation benchmarks.

Much of the research concerning increasing the performance of RGBD semantic segmentation models has focused on adapting state-of-the-art models from pure RGB-based segmentation, and adapting them to a dual backbone setup for each modality. Some models make use of clever techniques to incorporate the geometry of the depth values as part of the backbone’s feature extraction from

the color image, such as CFN [63] and the depth-aware convolution operation [11]. The formula is simple:

- 1) Examine state-of-the-art techniques from the competitive world of RGB object detection and image segmentation.
- 2) Adapt the new techniques to RGBD input.

With this formula in mind, new research into improving performance for RGBD-based CNN architectures can be continuously conducted. Relevant to this thesis, one of the goals of modern panoptic segmentation research is to create a fully learned approach to fusing the predictions from the instance segmentation prediction and semantic segmentation, rather than relying on post-processing heuristic techniques. The post-processing techniques employed by the panoptic model discussed in Section 4.2.4 rely on first removing duplicate bounding box detections through non-maximal suppression, followed by another application of a modified mask-level non-maximal suppression, where overall the predicted segmentation mask does not influence the instance mask predictions. Indeed, computer vision approaches with heavy post-processing are generally discouraged in the deep learning sphere due to run-time inefficiency and the desire to create end-to-end algorithms. Hence a fully-learned depth-aware panoptic segmentation CNN model is certainly an avenue for further exploration.

In terms of run-time efficiency and memory usage, one may easily critique the dual backbone techniques employed in this work. The dual backbone models in this thesis had higher inference time, where notably the best performing depth-aware semantic segmentation algorithm was twice as slow as the comparable RGB baseline, dipping below the 30 fps needed for real-time inference. The RGBD object detection model developed using the learned concatenation operation was however able to still maintain an inference speed greater than 30 fps. A dual ResNet-101 approach uses approximately $2 \times 45\text{M} = 90$ million parameters, not to mention the additional parameters from the fusion modules. Methods like the depth aware-convolution operation [11] that use the depth values to modify the standard convolution operation dispense with the dual backbone approach in favor of a much faster and less memory-intensive result, however these approaches trail the performance of models such as RDFNet [61] in RGBD-based semantic segmentation, a dual backbone approach. Therefore the question remains for RGBD CNN architectures: *What is the most memory efficient method to adapt*

CNNs for RGBD images that does not sacrifice performance?

For faster RGBD models, researchers may wish to draw inspiration from modern real-time RGB segmentation approaches that aim to reduce memory and inference time, where the bottleneck for the panoptic segmentation task is the speed of the instance segmentation stage. For instance the Yolact (You only look at coefficients) model [87] foregoes the sequential two-stage method in Mask R-CNN of region proposal generation followed by feature extraction through RoI Pooling, in favor of a parallel approach of generating a set of prototype masks and predicting coefficients per each mask. The Yolact architecture achieves around 30 fps at a performance of 30% mAP on the MS-COCO 2015 test set, almost twice the speed of Mask R-CNN.

In the direction of novelty, drawing from deep learning approaches to natural language processing (NLP), recently computer vision researchers have begun to make use of the transformer architecture to re-frame tasks like object prediction as a set prediction problem. Transformers have achieved state-of-the-art performance in NLP tasks such as machine translation and speech recognition using an end-to-end philosophy as well as so-called self-attention mechanisms. As argued by Carion et al. [66], modern object detectors, including both two-stage region-based methods and single shot methods, address the set prediction problem indirectly, by defining surrogate regression and classification tasks on region proposals, anchor boxes, and window centers. The DETR model [66] (Detection Transformer) obtains unique predictions at the output of a transformer encoder-decoder architecture via bipartite matching, for a fully end-to-end solution. Further, this approach has already been adopted for the panoptic segmentation task. A depth-aware transformer architecture for object detection and panoptic segmentation would be a fascinating research direction to pursue.

At the same time as depth sensing technology has improved the effective range, resolution, and de-noising of the RGBD images obtained, computer vision research has entered into a boom phase of increasingly impressive and competitive works in the literature for detection and segmentation. The merging of depth sensor technology with creative algorithm development will continue to propel advances in this domain, ready to be integrated with real-world applications.

References

- [1] S. Papert, “The Summer Vision Project,” MIT AI Memos (1959-2004), Boston, Massachusetts, USA, July 1966. Accessed on May 1, 2020. [Online]. Available at <https://dspace.mit.edu/handle/1721.1/6125>.
- [2] M. Isaac, “Microsoft Releases Xbox Kinect SDK, Hackers Get to Work”, <https://www.wired.com/2011/06/microsoft-kinect-hackers/>, (Retrieved March 5, 2021).
- [3] R. A. El-laithy, J. Huang, and M. Yeh, “Study on the Use of Microsoft Kinect for Robotics Applications,” Position, Location, and Navigation Symposium, pp. 1280-1288, 2012, doi: 10.1109/PLANS.2012.6236985.
- [4] M. Gabel, R. Gilad-Bachrach, E. Renshaw, and A. Schuster, “Full Body Gait Analysis with Kinect,” Engineering in Medicine and Biology Society, pp. 1964-1967, 2012, doi: 10.1109/EMBC.2012.6346340.
- [5] X. Chen, K. Henrickson, and Y. Wang, “Kinect-Based Pedestrian Detection for Crowded Scenes,” Computer-Aided Civil and Infrastructure Engineering, vol. 31, no.3, pp. 229–240, March 2016, doi:10.1111/mice.12163.
- [6] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, “Indoor Segmentation and Support Inference from RGBD Images,” European Conference on Computer Vision, vol. 5, pp. 746-760, October 2012, doi: 10.1007/978-3-642-33715-4_54.
- [7] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik, “Learning Rich Features from RGB-D images for Object Detection and Segmentation,” European Conference on Computer Vision, pp. 345-360, 2014.
- [8] S. Song, S. Lichtenberg, and J. Xiao, “SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite,” Computer Vision and Pattern Recognition, pp. 567-576, 2015, doi: 10.1109/CVPR.2015.7298655.
- [9] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” Association for Computing Machinery, vol. 60, no. 6, pp. 84-90, 2017, doi: 10.1145/3065386.
- [10] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” Computer Vision and Pattern Recognition, pp. 248-255, 2009, doi: 10.1109/CVPR.2009.5206848.
- [11] W. Wang and U. Neumann, “Depth-Aware CNN for RGB-D Segmentation,” European Conference on Computer Vision, pp. 144-161, 2018.
- [12] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” Transactions on Pattern

- Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, June 2017, doi: 10.1109/TPAMI.2016.2577031.
- [13] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature Pyramid Networks for Object Detection,” *Computer Vision and Pattern Recognition*, pp. 936-944, 2017, doi: 10.1109/CVPR.2017.106.
 - [14] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *Computer Vision and Pattern Recognition*, pp. 770-778, 2016, doi: 10.1109/CVPR.2016.90.
 - [15] “PyTorch ResNet Implementation,” GitHub Repository File, <https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py>, Accessed March 5, 2021.
 - [16] O. K. Oyedotun, D. Aouada, and B. Ottersten, “Learning to Fuse Latent Representations for Multimodal Data,” *The International Conference on Acoustics, Speech, and Signal Processing*, pp. 3122-3126, 2019, doi:10.1109/ICASSP.2019.8682454.
 - [17] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” *International Conference on Computer Vision*, pp. 2980-2988, 2017, doi: 10.1109/ICCV.2017.322.
 - [18] LC Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking Atrous Convolution for Semantic Image Segmentation,” *arXiv:1706.05587 [cs]*, December 2017.
 - [19] FAIR Research, “PyTorch DeepLabv3 Implementation,” GitHub Repository File, <https://github.com/pytorch/vision/blob/master/torchvision/models/segmentation/deeplabv3.py>, Accessed March 6, 2021.
 - [20] G. Li, Y. Gan, H. Wu, N. Xiao, and L. Lin, “Cross-Modal Attentional Context Learning for RGB-D Object Detection,” *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1591-1601, April 2019, doi: 10.1109/TIP.2018.2878956.
 - [21] X. Qi, R. Liao, J. Jia, S. Fidler, and R. Urtasun, “3D Graph Neural Networks for RGBD Semantic Segmentation,” *International Conference on Computer Vision*, pp. 5209-5218, 2017, doi: 10.1109/ICCV.2017.556.
 - [22] S. Gupta, P. Arbeláez, and J. Malik, “Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images,” *Computer Vision and Pattern Recognition*, pp. 564-571, 2013, doi: 10.1109/CVPR.2013.79.
 - [23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, “Microsoft COCO: Common Objects in Context,” *European Conference on Computer Vision*, pp. 740-755, 2014.

- [24] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, “Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer,” *Transactions on Pattern Analysis and Machine Intelligence*, 2020, doi: 10.1109/TPAMI.2020.3019967.
- [25] “Holding up the Leaning Tower of Pisa,” <https://www.flickr.com/photos/bjcarter/2998965385/in/photolist-V6hqhh-4Eich5-48Log-4HtNqP-4EdNHF-o5NLK5-4BRJTQ-5v1qbv-5v1pZB-5v5LEs-5v1nGt-5v1pVx-5v5JEJ-5z1tVK-5v5GEU-5v5LBy-5v5Kpo-5v1mtX-5v5GD3-5v1o5X-5v1pGx-5v1ot2-5v5Lsu-5v1py6-UscGdA-5v5Hg3-5v1maT-5v1pdK-5v1mm4-5v1mW4-5v1oLn-5v1nn6-5v1ocT-7rk5pr-NDVHu-6iWFfg-eeNUbR-2HtZG-sTjJ4i-eeNTMM-gbyMr-eeUDco-n82z9-6ifMWj-8U5YDV-pNZBS>, Retrieved from Flickr Creative Commons, June 1, 2021.
- [26] R. Padilla, S. Netto, and E. da Silva, “A Survey on Performance Metrics for Object-Detection Algorithms,” *International Conference on Systems, Signals, and Image Processing*, pp. 237-242, 2020.
- [27] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” *Computer Vision and Pattern Recognition*, pp. 580-587, 2014, doi: 10.1109/CVPR.2014.81.
- [28] R. Girshick, “Fast R-CNN,” *International Conference on Computer Vision*, pp. 1440-1448, 2015, doi: 10.1109/ICCV.2015.169.
- [29] J. Hosang, R. Benenson, P. Dollár, and B. Schiele, “What Makes for Effective Detection Proposals?” *Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 4, pp. 814-830, April 2016, doi: 10.1109/TPAMI.2015.2465908.
- [30] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes (VOC) Challenge,” *International Journal of Computer Vision*, vol. 88, pp. 303-338, 2010.
- [31] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, “Panoptic Segmentation,” *Computer Vision and Pattern Recognition*, pp. 9396-9405, 2019, doi: 10.1109/CVPR.2019.00963.
- [32] “Definition of the Word Panoptic,” *Oxford Online Dictionary*, <https://en.oxforddictionaries.com/definition/panoptic>, (Retrieved March 5, 2021).
- [33] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *International Conference on Learning Representations*, 2015.
- [34] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, “Selective Search

- for Object Recognition,” *International Journal on Computer Vision*, vol. 104, pp. 154-171, 2013.
- [35] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, “Object Detection with Discriminatively Trained Part-Based Models,” *Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627-1645, Sept. 2010, doi: 10.1109/TPAMI.2009.167.
- [36] J. Long, E. Shelhamer, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” *Computer Vision and Pattern Recognition*, pp. 3431-3440, 2015, doi: 10.1109/CVPR.2015.7298965.
- [37] Z. Li, Y. Ma, Y. Chen, X. Zhang, and J. Sun, “COCO instance segmentation challenge track technical report: MegDetV2,” *IEEE International Conference on Computer Vision Workshop*, 2019.
- [38] FAIR Research, “PyTorch Mask R-CNN Implementation”, GitHub Repository File, https://github.com/pytorch/vision/blob/master/torchvision/models/detection/mask_rcnn.py, Accessed April 10, 2021.
- [39] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, “Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs,” *Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834-848, 2018, doi: 10.1109/TPAMI.2017.2699184.
- [40] P. Dollár and C. L. Zitnick, “Structured Forests for Fast Edge Detection,” *International Conference on Computer Vision*, 2013, pp. 1841-1848, doi: 10.1109/ICCV.2013.231.
- [41] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, “From Contours to Regions: An Empirical Evaluation,” *Computer Vision and Pattern Recognition*, pp. 2294-2301, 2009, doi: 10.1109/CVPR.2009.5206707.
- [42] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik, “Multiscale Combinatorial Grouping,” *Computer Vision and Pattern Recognition*, pp. 328-335, 2014, doi: 10.1109/CVPR.2014.49.
- [43] S. Gupta, J. Hoffman, and J. Malik, “Cross Modal Distillation for Supervision Transfer,” *Computer Vision and Pattern Recognition*, pp. 2827-2836, 2016, doi: 10.1109/CVPR.2016.309.
- [44] X. Xu, Y. Li, G. Wu, and J. Luo, “Multi-Modal Deep Feature Learning for RGB-D Object Detection,” *Pattern Recognition*, vol. 72, pp. 300-313, 2017, doi: 10.1016/j.patcog.2017.07.026.
- [45] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, November 1997, doi:

10.1162/neco.1997.9.8.1735.

- [46] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial Transformer Networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.
- [47] T. Ophoff, K. Beeck, and T. Geodeme, "Exploring RGB+Depth Fusion for Real-Time Object Detection," *Sensors*, vol. 19, no. 4, p.866, 2019, doi:10.3390/s19040866.
- [48] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *Computer Vision and Pattern Recognition*, pp. 779-788, 2016, doi: 10.1109/CVPR.2016.91.
- [49] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed, "SSD: Single Shot Multibox Detector," *European Conference on Computer Vision*, pp. 21-37, 2016.
- [50] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *Computer Vision and Pattern Recognition*, pp. 6517-6525, 2017, doi: 10.1109/CVPR.2017.690.
- [51] T. Bagautdinov, F. Fleuret, and P. Fua, "Probability occupancy maps for occluded depth images," *Computer Vision and Pattern Recognition*, pp. 2829-2837, 2015, doi:10.1109/CVPR.2015.7298900.
- [52] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," *Computer Vision and Pattern Recognition*, pp. 3354-3361, 2012, doi: 10.1109/CVPR.2012.6248074.
- [53] P. Vanegas, J. Peeters, D. Cattrysse, P. Tecchio, F. Ardente, F. Mathieux, W. Dewulf, and J. Duflou, "Ease of Disassembly of Products to Support Circular Economy Strategies," *Resources, Conservation, and Recycling*, vol. 135, pp. 323-334, 2018.
- [54] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor Semantic Segmentation using Depth Information," *Computing Research Repository*, vol. abs/1301.3572, 2013.
- [55] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning Hierarchical Features for Scene Labeling," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915-1929, Aug. 2013, doi: 10.1109/TPAMI.2012.231.
- [56] P. Felzenszwalb and D. Huttenlocher, "Efficient Graph-Based Image Segmentation," *International Journal of Computer Vision*, vol. 59, pp. 167-181, 2004.
- [57] D. Eigen and R. Fergus, "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture,"

- International Conference on Computer Vision, pp. 2650-2658, 2015, doi: 10.1109/ICCV.2015.304.
- [58] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers, “FuseNet: Incorporating Depth into Semantic Segmentation via Fusion-Based CNN Architecture,” Asian Conference on Computer Vision, 2016.
 - [59] H. Noh, S. Hong, and B. Han, “Learning Deconvolution Network for Semantic Segmentation,” International Conference on Computer Vision, pp. 1520-1528, 2015, doi: 10.1109/ICCV.2015.178.
 - [60] Z. Li, Y. Gan, X. Liang, H. Cheng, and L. Lin, “LSTM-CF: Unifying Context Modeling and Fusion with LSTMs for RGB-D Scene Labeling,” European Conference on Computer Vision, 2016.
 - [61] S. Lee, S. Park, and K. Hong, “RDFNet: RGB-D Multi-level Residual Feature Fusion for Indoor Semantic Segmentation,” International Conference on Computer Vision, pp. 4990-4999, 2017, doi: 10.1109/ICCV.2017.533.
 - [62] G. Lin, A. Milan, C. Shen, and I. Reid, “RefineNet: Multi-path Refinement Networks for High-Resolution Semantic Segmentation,” Computer Vision and Pattern Recognition, pp. 5168-5177, 2017, doi: 10.1109/CVPR.2017.549.
 - [63] D. Lin, G. Chen, D. Cohen-Or, P. Heng, and H. Huang, “Cascaded Feature Network for Semantic Segmentation of RGB-D Images,” International Conference on Computer Vision, pp. 1311-1319, 2017.
 - [64] M. Mozer, “A Focused Backpropagation Algorithm for Temporal Pattern Recognition,” Complex Systems, vol. 3, 1989.
 - [65] “COCO 2018 Panoptic Segmentation Task,” <https://cocodataset.org/#panoptic-2018>, (Retrieved March 5, 2021).
 - [66] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-End Object Detection with Transformers,” European Conference on Computer Vision, pp. 213-229, 2020.
 - [67] X. Wang, R. Zhang, T. King, L. Li, and C. Shen, “SOLOv2: Dyanmic and Fast Instance Segmentation,” Computer Vision and Pattern Recognition, 2020.
 - [68] R. Mohan and A. Valada, “EfficientPS: Efficient Panoptic Segmentation,” International Journal of Computer Vision, vol. 129, pp. 1551-1579, 2021.
 - [69] R. Srivastava, K. Gleff, and J. Schmidhuber, “Highway Networks,” Advances in Neural Information Processing Systems, vol. 28, pp. 2377–2385, 2015.
 - [70] H. Zhao, J. Shi, X. Wang, and J. Jia, “Pyramid scene parsing network,” Computer Vision and Pattern Recognition, 2017.
 - [71] K. He, R. Girshick, and P. Dollar, “Rethinking ImageNet Pre-Training,”

- International Conference on Computer Vision, pp. 4917-4926, 2019, doi: 10.1109/ICCV.2019.00502.
- [72] A. Levin, D. Lischinski, and Y. Weiss, “Colorization Using Optimization,” Association for Computing Machinery: Transactions on Graphics, vol. 23, no. 3, pp. 689-694, 2004, doi: 10.1145/1015706.1015780.
- [73] N. Silberman, “NYU Depth Dataset v2 website,” https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html, (Retrieved February 23, 2021).
- [74] A. Torralba and A. Efros, “Unbiased Look at Dataset Bias,” Computer Vision and Pattern Recognition, pp. 1521-1528, 2011, doi: 10.1109/CVPR.2011.5995347.
- [75] A. Janoch, S. Karayev, Y. Jia, J. Barron, M. Fritz, K. Saenko, and T. Darrell, “A Category-Level 3-D Object Dataset: Putting the Kinect to Work,” International Conference on Computer Vision, pp. 1168-1174, 2011, doi: 10.1109/ICCV.2011.6130382.
- [76] J. Xiao, A. Owens, and A. Torralba, “SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels,” International Conference on Computer Vision, pp. 1625-1632, 2013, doi: 10.1109/ICCV.2013.458.
- [77] FAIR Research, “PyTorch/Torchvision,” GitHub Repository, <https://github.com/pytorch/vision/tree/master/torchvision>, Accessed February 5, 2021.
- [78] M. Huang, “Introducing Native PyTorch Automatic Mixed Precision for Faster Training on NVIDIA GPUs,” <https://pytorch.org/blog/accelerating-training-on-nvidia-gpus-with-pytorch-automatic-mixed-precision/>, (Retrieved March 5 2021).
- [79] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” International Conference on Computer Vision, pp. 1026-1034, 2015, doi: 10.1109/ICCV.2015.123.
- [80] “ResNet,” https://pytorch.org/hub/pytorch_vision_resnet/, PyTorch documentation, (Retrieved March 5, 2021).
- [81] S. Gupta, “NYU Hooks,” GitHub Repository, <https://github.com/s-gupta/nyu-hooks>, Accessed March 10, 2021.
- [82] “SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite Website,” <https://rgbd.cs.princeton.edu/>, Accessed March 10, 2021.
- [83] J. Dickens, “Masters Thesis Repository,” GitHub repository, <https://github.com/JamesMcCullochDickens>, Accessed June 20, 2021.
- [84] D. Seichter, M. Kohler, B. Lewandowski, T. Wengelfeld, and H. Gross,

- “Efficient RGB-D Semantic Segmentation for Indoor Scene Analysis,” arXiv:2011.06961 [cs], April 2021.
- [85] 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,” IEEE Conference on Computer Vision and Pattern Recognition, pp. 3213-3223, 2016, doi: 10.1109/CVPR.2016.350.
 - [86] A. Dai, A. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Niebner, “ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes,” IEEE Conference on Computer Vision and Pattern Recognition, 2017.
 - [87] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, “YOLACT: Real-Time Instance Segmentation,” International Conference on Computer Vision, pp. 9156-9165, 2019, doi:10.1109/ICCV.2019.00925.
 - [88] F. Yu and V. Koltun, “Multi-Scale Context Aggregation by Dilated Convolutions,” International Conference on Signal Processing, vol. 15, pp. 240-245, 2020, doi: 10.1109/ICSP48669.2020.9321067.
 - [89] D. Sussillo and L. Abbott, “Random Walk Initialization for Training Very Deep Feedforward Networks,” Neural and Evolutionary Computing, 2014.
 - [90] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” International Conference on Machine Learning, vol. 37, pp. 448–456, 2015.

Appendix A: Basic CNN Components

In this section, the core components of the standard CNN *backbone* are reviewed, as well as fully connected layers. The function of the backbone is to extract features from input image batches. Common backbones used include ResNets [14], and VGG models [33]. Once these features are extracted, they are then input to another part of an architecture for task-specific goals.

A.1 Convolutions

Convolution operations use filters, referred to sometimes as *kernels*, which are real-valued tensors of shape $F := (In, Out, f_w, f_h)$ where:

- In is the number of input channels¹³ of the tensor to be convolved. Note that when an image tensor is of shape $(H, W, 3)$, the tensor is said to be in channels-last format, as opposed to the shape $(3, H, W)$, which is said to be in channels-first format. In this section we will assume a channels-first format for simplicity.
- Out is the number of output channels
- f_w is the width of the filter
- f_h is the height of the filter (typically $f_h = f_w$, and f_h is a value from $\{1, 3, 5, 7\}$)

Each value or *weight* of the convolutional filter is a learnable parameter (assuming its weight has not been frozen). Moreover, convolution is a fully differentiable operation as it only involves multiplication and addition operations.

F acts on an input tensor $I := (In, H, W)$ where H, W are the height and width of the input tensor. A typical convolution with stride 1 and *valid padding* places the image filter centered at pixel locations (x, y) in the input tensor where the entire filter fits without extending past the image boundaries, to obtain a resultant tensor R of shape $(Out, H - \lfloor f_h/2 \rfloor, W - \lfloor f_w/2 \rfloor)$, where for channel t of the output, $(1 \leq t \leq Out)$,

$$R(t, x, y) := (F * I(x + \lfloor f_h/2 \rfloor, y + \lfloor f_w/2 \rfloor)) \quad (\text{A.1})$$

$$= \sum_{i=1}^{In} \sum_{k=\lfloor -f_h/2 \rfloor}^{\lfloor f_h/2 \rfloor} \sum_{l=\lfloor -f_w/2 \rfloor}^{\lfloor f_w/2 \rfloor} F(i, t, k, l) \cdot I(i, x + \lfloor f_h/2 \rfloor - k, y + \lfloor f_w/2 \rfloor - l) + b_t \quad (\text{A.2})$$

The indexing scheme of the spatial locations of the filter is such that the cen-

¹³These are sometimes referred to as input planes.

ter of the filter is at $(0, 0)$ and $(\lfloor -f_h/2 \rfloor, \lfloor -f_w/2 \rfloor)$ and $(\lfloor f_h/2 \rfloor, \lfloor f_w/2 \rfloor)$ are the respective top left and bottom right locations.

Each output channel has a learnable real-valued bias denoted b_t added element-wise to each spatial location. In practice, the implementation of convolution in frameworks such as PyTorch is actually achieved by the cross-correlation operation, given by:

$$(F \star I(x, y))_t = \sum_{i=1}^{I_n} \sum_{k=\lfloor -f_h/2 \rfloor}^{\lfloor f_h/2 \rfloor} \sum_{l=\lfloor -f_w/2 \rfloor}^{\lfloor f_w/2 \rfloor} F(i, t, k, l) \cdot I(i, x + k, y + l) + b_t \quad (\text{A.3})$$

This isn't a problem since the filter weights are learned and not fixed, and it can be easily shown that convolution by one filter is equivalent to cross-correlation with the same filter with the rows and columns swapped (and vice versa).

The *padding* of an image in the context of convolution involves adding zeros (or sometimes other values such as 1s or user input values) to the input tensor's boundaries, adding extra rows, and extending rows with these values. For zero-padding, the most common type of padding, this can be expressed as a tuple (p_w, p_h) where p_w, p_h are positive integers denoting p_w extra zeros added to the end and beginning of a row, and p_h extra rows of zeros added to the top and bottom of the image. Each added row has length equal to the sum of the old row length and $2 * p_w$. Commonly, the types of padding used are:

- *Same padding*: for a filter kernel of size (f_w, f_h) (width by height) the padding is $(\lfloor f_w/2 \rfloor, \lfloor f_h/2 \rfloor)$, where in this case the convolved output tensor has the same spatial resolution as the input tensor (hence the name *same*).
- *Valid padding*: where no padding is used.

The *stride* of a convolution refers to the placement of the image filter's center relative to the input tensor. This is expressed as a tuple (s_w, s_h) where s_w is a positive integer denoting that each filter placement, starting at 0, should consecutively be placed at s_w pixels to the right of the previous window center. The value s_h operates similarly but in the vertical direction, as in each filter placement, once a row is completed, should then be placed s_h rows below the old one. The stride of a convolution can affect the spatial resolution of the output tensor.

The *dilation* [88] of a convolution operation is a parameter used to control zeros added to spatial locations of a filter, for the purpose of increasing the filter size (and hence incorporate information from a larger patch of the image), *without increasing the number of trainable parameters*. This is also referred to as atrous

convolution. In particular, for a dilation tuple (d_1, d_2) for a convolution with filter size (f_h, f_w) , the dilated convolution operation is of the form:

$$(F \star I(x, y))_t = \sum_{i=1}^{I_n} \sum_{m=\lfloor -f_h/2 \rfloor}^{\lfloor f_h/2 \rfloor} \sum_{n=\lfloor -f_w/2 \rfloor}^{\lfloor f_w/2 \rfloor} F(i, t, k, l) \cdot I(i, x+d_1 \cdot m, y+d_2 \cdot n) + b_t \quad (\text{A.4})$$

See figure A.1.1 below for an example of a single channel 3×3 filter that is modified with dilation tuple $(2, 2)$.

2	-1	5
-2	4	7
1	3	11

0	0	0	0	0	0	0
0	2	0	-1	0	5	0
0	0	0	0	0	0	0
0	-2	0	4	0	7	0
0	0	0	0	0	0	0
0	1	0	3	0	11	0
0	0	0	0	0	0	0

Figure A.1: A dilated 3×3 filter with dilation tuple $(2, 2)$.

Overall, for a convolution of filter size (h, w) with dilation factor (d_1, d_2) , stride tuple (s_1, s_2) , and padding tuple (p_1, p_2) , the spatial resolution of the output, for an input of size H, W , will be

$$H_{out} = \lfloor \frac{H + 2 \cdot p_1 - d_1 \cdot (h - 1) - 1}{s_1} + 1 \rfloor \quad (\text{A.5})$$

$$W_{out} = \lfloor \frac{W + 2 \cdot p_2 - d_2 \cdot (w - 1) - 1}{s_2} + 1 \rfloor \quad (\text{A.6})$$

In order to compute the loss with respect to the weights of a convolutional operation, consider a filter F of shape (Out, In, h, w) acting on a same-padded tensor I of shape (In, H, W) to produce an output tensor R of shape (Out, H, W) . As a simplified example, suppose the convolution operation has dilation tuple $(1, 1)$, and uses a stride tuple of $(1, 1)$. Then for some loss value L , dependent on the output R of our convolution operation, we have by the multivariate chain rule that

$$\frac{\partial L}{\partial F} = \frac{\partial L}{\partial R} \frac{\partial R}{\partial F} \quad (\text{A.7})$$

More specifically we can infer that

$$\frac{\partial L}{\partial F(i, j, x, y)} = \frac{\partial L}{\partial R} \frac{\partial R}{\partial F(i, j, x, y)} \quad (\text{A.8})$$

Consider the tensor $\frac{\partial R}{\partial F(i,j,h,w)}$, of shape (Out, H, W) .

Then

$$\frac{\partial R(t, h, w)}{\partial F(i, j, x, y)} = \begin{cases} I(j, h + x, w + y), & \text{if } i = t \\ 0, & \text{otherwise} \end{cases} \quad (\text{A.9})$$

The full scope of the optimized linear algebra used for computing the loss of a CNN with respect to the weights of its convolutional filters is outside the scope of this work. However, one can see from the above computation that the convolution operation is differentiable with a relatively simple structure. Hence the loss of a network can be back-propagated efficiently to the weights of the filters as in the back-propagation algorithm used for gradient-based optimization methods.

A.2 Pooling Layers

Supposing we have an input tensor I of shape (C, H, W) for C input planes and spatial resolution (H, W) (height by width), then a *pooling* operation involves summarizing information from fixed sized spatial windows of the tensor applied with some stride tuple (s_1, s_2) to achieve an output tensor R of shape (C, H_{out}, W_{out}) . When the stride tuple has a component larger than 1 in either the width or height direction, the spatial resolution of the resultant tensor is reduced. A common window size would be 3×3 with stride tuple $(2, 2)$ in order to reduce the size of the input tensor's spatial resolution by a factor of 4, while leaving the number of channels intact. The two most common type of pooling operations are max-pooling and average pooling.

Assuming a same-padded input with stride tuple $(1, 1)$, in *max pooling* we apply a non-linear max-filter kernel of shape (H, W) to obtain the maximum value of the input tensor within the window, where the output at channel j will be

$$out(j, h, w) = \max_{\{m=-\lfloor H/2 \rfloor, \dots, \lfloor H/2 \rfloor\}} \max_{\{n=-\lfloor W/2 \rfloor, \dots, \lfloor W/2 \rfloor\}} I(j, h + m, w + n) \quad (\text{A.10})$$

Average pooling works similarly, except instead of taking the maximum value in the window considered, the average value is obtained. There are several reasons to apply pooling, mainly:

- To reduce the spatial resolution of the input for reduced memory consumption.
- To summarize the information provided by the spatial windows for which the pooling is applied.
- To reduce the effect of small distortions and to increase generalization ability by reducing overfitting.

Other types of pooling include minimum pooling, L_2 normalized pooling operations, spatial pyramid pooling, etc. Pooling operations, in general, are fully differentiable and so can be incorporated in the back-propagation computation, although they do not typically have trainable components.

A.3 Activation Functions

Activation functions serve to introduce non-linearities within neural networks at the output of linear layers, convolutional layers, pooling layers, and other types of layers. Some of the desired properties of activation functions are:

- Nonlinearity, which has been shown to contribute to learning intricate patterns that cannot be learned with purely linear activations.
- Continuous differentiability, in order for the activation to be seamlessly incorporated into the back-propagation algorithm.
- Approximate identity near the origin. It has been shown that when the activation function does not approximate identity near the origin, weight initialization is more involved [89].
- Monotonic functions with monotonic derivatives.

The AlexNet [9] and ResNet [14] architectures both use the ReLU (Rectified Linear Unit) activation function for the backbone layer activations, where ReLU is defined as:

$$ReLU(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{otherwise} \end{cases} \quad (\text{A.11})$$

which satisfies the above criterion. Alternative backbone activation functions include $\tanh$, and variants of ReLU such as Leaky ReLU, or Smooth ReLU, etc.

In certain cases the activation function at the end of CNN architectures is used to convert tensors to obtain prediction probabilities. For example, in the final layer of the classification architecture of AlexNet [9], a softmax layer is employed in order to convert the output tensor of the fully-connected layer to a probability tensor, which is then thresholded to obtain a class prediction at test time. The softmax operation for a tensor I of shape $(C + 1)$ for C classes (plus a background class) is given by:

$$Softmax(I)_i = \frac{e^{I_i}}{\sum_{i=1}^{C+1} e^{I_i}} \quad (\text{A.12})$$

with output in the range $(0, 1]$. $Softmax(I)$ can be interpreted as a probability vector, where the i 'th component is the probability of the model's prediction for

class C_i .

A.4 Batch Normalization

In deep neural networks, inputs to each layer have a corresponding distribution, which is affected during the training process by the parameter initialization of the network weights, the randomness inherent in the input data, and the updates to the weights of the previous layer after an optimization step. Ioffe and Szegedy [90] describe the effect of these sources of randomness and a continually changing input distribution to a layer in a deep neural network as detrimental, and refer to this phenomenon as *internal covariate shift*. The authors introduce a technique to mitigate this effect known as batch normalization. One advantage of this approach is the reduction of the dependence of gradients on the scale of the model parameters, or of their initial values, allowing for higher learning rates, and hence faster model convergence. Further, performance increases in computer vision tasks such as classification have been demonstrated from networks whose layers are augmented with batch-normalized inputs [90].

In the context of training a convolutional neural network, suppose we have an input tensor I to some layer of an architecture, where the shape of I is (N, C, H, W) for a batch size of N tensors of C channels, where each channel has spatial resolution (H, W) , (height by width). Then for channel i , the sample mean μ_i is computed as

$$\mu_i = \frac{1}{N \cdot H \cdot W} \sum_{j=1}^N \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} I(j, i, h, w) \quad (\text{A.13})$$

with the corresponding sample variance (the biased estimator)

$$\sigma^2 = \frac{1}{N \cdot H \cdot W} \sum_{j=1}^N \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} (I(j, i, h, w) - \mu_i)^2 \quad (\text{A.14})$$

For batch index t at the spatial location (h, w) in the i 'th channel, the output tensor Y , of the same shape as I , is given by

$$Y(t, i, h, w) = \frac{I(N_t, i, h, w) - \mu_i}{\sqrt{\sigma^2 + \epsilon}} * \gamma_i + \beta_i \quad (\text{A.15})$$

where ϵ is some very small constant to ensure a non-zero denominator, and γ_i, β_i are learnable parameters. Batch normalization is a fully differentiable operation and so can be seamlessly integrated into the back-propagation algorithm, where each layer taking an input of C channels has two corresponding trainable vectors γ, β of shape $(C, 1)$.

During training, some implementations of batch normalization layers keep running statistics about the mean and variance input. This is done so that at inference time, a fixed normalization of the test data batches can be computed under the assumption that the training data mean and variance, per each channel affected by the layer, will be similar to that of the testing data. Salient to this work, the ResNet backbone architectures make heavy use of batch normalization before convolutional layers and before ReLU activations, as discussed in Section 2.3.

A.5 Fully Connected Layers

Specific to computer vision and convolutional neural network training, for the purposes of region proposal objectness prediction, image classifier predictions, and bounding box regression, many architectures input features extracted from the backbone into dense or *fully connected layers* (FC layers) before transforming the tensors to the final representation of the network’s output. FC layers are a key ingredient in modern deep learning networks, and act to learn non-linear combinations¹⁴ of features output from the previous layers for many different tasks. They are frequently applied in sequence, as in multiple FC layers connected to one another (often 2 in a row), and are followed with an activation function, typically ReLU. FC layers act on an input tensor X of shape (N, I) , which is a matrix of size $N \times I$, for N batch elements and I input features as

$$XA^T + b \tag{A.16}$$

where A is a matrix of size (O, I) for O out features and b is the bias vector of shape (O) added element-wise to each out feature of the FC Layer for each batch element. The resultant tensor is of shape (N, O) .

¹⁴Assuming there is some kind of non-linear activation function applied to the output of each fully connected layer.

Appendix B: Category Mapping

Object detection class mapping for the SUN RGBD dataset:

tub → **bathtub**

bunk bed → **bed**

bookcase → **bookshelf**

stack of chairs, chairs, high chair, lounge chair, chair thumb, rocking chair, lawn chair, stacked chairs → **chair**

kitchen island → **counter**

cdoor → **door**

recycle bin → **garbage bin**

wallamp → **lamp**

kitchen sink → **sink**

sofa chair, sofa bed, ottoman → **sofa**

coffee table, endtable, dining table, sidetable, coffeetable, centertable, ping pong table, long office table, bar table → **table**

tv, flat screen tv → **television**

Instance segmentation class mapping for the SUN RGBD dataset:

bunk bed → **bed**

stacked chairs, plastic chair, sofa chair → **chair**

game table, ping pong table, coffee table → **table**

computer monitor, tv, television → **monitor**

beige sofa → **sofa**