

DISSERTATION

Object Recognition Beyond RGB

Author:

Bryce ENGELBRECHT (818232)

Supervisor:

Prof. Terence VAN ZYL



UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

submitted to
the Faculty of Science, in fulfilment of the requirements for the degree of
Masters of Science

in the

Wits Institute of Data Science (WIDS)
School of Computer Science and Applied Mathematics

September 29, 2020

Declaration of Authorship

I, Bryce ENGELBRECHT (818232), declare that this dissertation titled, “Object Recognition Beyond RGB” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this dissertation has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely my own work.
- I have acknowledged all main sources of help.
- Where the dissertation is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Bryce Joseph Engelbrecht

Date: 2020/09/29

UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

Abstract

Faculty of Science
School of Computer Science and Applied Mathematics

Masters of Science

Object Recognition Beyond RGB

by Bryce ENGELBRECHT (818232)

Object recognition and the subproblem of land cover classification has been a key focus of computer vision research. An increasing number of devices have begun supporting the capture of images with additional bands beyond the standard RGB bands, including depth and other spectra such as near infrared. There is an opportunity to study the use of RGB images with depth and multispectral images to improve the accuracy of the object recognition and land cover classification. We do this by taking existing state-of-the-art object recognition models and modifying them to work with RGB images with depth. For land cover classification we present a novel model, LandNet, which allows varying the number of backbone feature extractors and the image bands in each. We also study the impact of adding the additional depth information, bands and the use of multiple feature extractors on the training and inference times of the models. We find that adding depth data did not show any benefits for object recognition but has little effect on the training and inference times. Utilizing multispectral images allows for improvements for the accuracy of land cover classification. Adding the additional bands in single feature extractor has no effect on the training and inference times, however using multiple feature extractors does increase the training and inference times. The results leads us to conclude that depth data has the potential to improve object recognition accuracy but a larger dataset than SUN RGB-D is required to demonstrate improved performance when using RGB and depth images. We can conclude that using multispectral images for land cover classification has tangible benefits.

Acknowledgements

I would like to thank my supervisor, Prof. Terence van Zyl, for all the guidance and assistance that he provided while working on this research. I would also like to thank Dr Pravesh Ranchod for pointing me to the Facebook Research Detectron GitHub repository. Finally, I would like to thank my parents for their patience and support.

This work is based on the research supported wholly / in part by the National Research Foundation of South Africa (Grant Numbers: 117066).

Parts have been submitted to GRSL for publication under the title "Comparing CNN Structures for Land Classification on Multispectral Images" and is currently in revision.

Parts have been submitted to Fusion2020 for publication under the title "Object Recognition with RGB-D Images" and is currently under review.

Contents

Declaration of Authorship	i
Abstract	ii
Acknowledgements	iii
1 Introduction	1
1.1 Background	1
1.2 Problem Statement	1
1.3 Significance and Motivation	2
1.4 Research Aims and Objectives	3
1.5 Research Questions	3
1.6 Delineations, Limitations and Assumptions	4
1.7 Definitions	4
1.8 Outline	4
2 Background and Related Work	5
2.1 Useful Machine Learning Techniques	5
2.1.1 Deep Learning	5
2.1.2 Recursive Neural Networks	6
2.1.3 Convolutional Neural Networks	6
2.2 Object Recognition with RGB Images	6
2.2.1 Classical Object Recognition Models	7
2.2.2 State-of-the-art Object Recognition Models	7
R-CNN	7
RetinaNet	8
2.3 Object Recognition using 3D Images	8
2.3.1 Surface Matching	8
2.3.2 Volumetric Models	9
2.3.3 Convolutional Recursive Neural Network	9
2.4 Land Cover Classification	10
2.4.1 Land Cover Classification	10
2.4.2 EuroSAT	10
2.4.3 BigEarthNet	10
2.5 Conclusion	11
3 Research Methodology	12
3.1 Introduction	12
3.2 Wits RGB-D	12
3.3 Object Recognition	16
3.3.1 Data	16
3.3.2 Models	17
Mask R-CNN + nC	17

	RetinaNet + nC	19
3.4	Land Cover Classification	20
3.4.1	Data	20
3.4.2	LandNet	21
3.5	Implementation	24
3.6	Metrics and Analysis	24
3.6.1	Precision, Recall and F-score	24
3.6.2	Mean Average Precision	24
	IoU	24
	AP	25
	mAP	25
3.6.3	Analysis	25
3.7	Limitations	26
3.8	Conclusion	26
4	Results	27
4.1	Introduction	27
4.2	Object Recognition	27
4.2.1	Classification Results	27
4.2.2	Training and Inference Times	29
4.2.3	Efficiency	29
4.3	Analysis and Discussion	30
4.3.1	Mask R-CNN and RetinaNet	30
	Results on SUN RGB-D	32
	Results on Wits RGB-D	34
4.3.2	Wits RGB-D	35
4.4	Land Cover Classification	36
4.4.1	Results	36
	Training and Inference Times	38
	Efficiency	38
4.4.2	Analysis and Discussion	38
	Results on EuroSAT	40
	Results on BigEarthNet	44
4.5	Limitations	48
4.6	Conclusion	48
5	Conclusion	50
5.1	Summary of Research	50
5.2	Summary of Results	51
5.3	Future Work	51
5.4	Contributions	52
	Bibliography	53

List of Figures

3.1	Wits RGB-D RGB and depth image.	14
3.2	Wits RGB-D blurry image.	15
3.3	Comparison between two similar images.	15
3.4	Example annotated image.	15
3.5	Structure of Mask R-CNN + nC_1 with one RGB-D network.	17
3.6	Structure of Mask R-CNN + nC with separate RGB and additional channel networks.	18
3.7	Structure of RetinaNet + nC_1 with one RGB-D network.	19
3.8	Structure of RetinaNet + nC_1 with separate RGB and depth networks.	20
3.9	Structure of LandNet with 2 ResNet feature extractors. Different bands can be used as input to the two ResNets which extract features from the image. The combined feature vector is then passed through a final fully connected (FC) layer. The final feature vector is then fed into a softmax to determine the class.	22
3.10	Structure of LandNet with 3 ResNet feature extractors. Different bands are used as input to three separate ResNets. The outputted features are combined and sent through a FC layer before being used as input into a softmax function which determines the class.	23
4.1	Bar graph showing mAP scores of on SUN RGB-D	30
4.2	Bar graph showing accuracy of Mask R-CNN models trained for 8 epochs on SUN RGB-D	31
4.3	Bar graph showing accuracy of RetinaNet models on SUN RGB-D	31
4.4	Bar graph showing accuracy of various LandNet configurations on EuroSAT	40
4.5	Confusion matrix for LandNet 2 on EuroSAT	41
4.6	Confusion matrix for LandNet 6 on EuroSAT	41
4.7	Confusion matrix for LandNet 9 on EuroSAT	42
4.8	Bar graph showing mAP scores of on SUN RGB-D	44
4.9	Bar graph showing accuracy of Mask R-CNN models trained for 8 epochs on SUN RGB-D	45
4.10	Bar graph showing accuracy of RetinaNet models on SUN RGB-D	45

List of Tables

3.1	Table showing the number of objects belonging to each class in Wits RGB-D.	13
3.2	Table showing the classes in SUN RGB-D.	16
3.3	Sentinel-2 bands.	21
3.4	Band and feature extractor combinations.	23
4.1	SUN RGB-D classification results.	27
4.2	SUN RGB-D per class classification results.	28
4.3	Wits RGB-D classification results.	28
4.4	SUN RGB-D training and inference time results.	29
4.5	SUN RGB-D efficiency results.	30
4.6	EuroSAT classification results.	36
4.7	BigEarthNet classification results.	36
4.8	Classification accuracy for each class of BigEarthNet using LandNet configurations 6 and 9 as well as the number of samples in each class. .	37
4.9	EuroSAT training and inference time results.	38
4.10	BigEarthNet training and inference time results.	38
4.11	EuroSAT efficiency results.	39
4.12	BigEarthNet efficiency results.	39

List of Abbreviations

RGB	Red Green Blue
RGB-D	Red Green Blue - Depth

Chapter 1

Introduction

In this chapter we introduce the problem and present some background surrounding the problem. We also state the impact of the research. This is followed by a brief outline of the aims and objectives of this research. We then present our research questions, assumptions and limitations of the research and finally, the chapter is ended with a brief outline of the organisation of the paper.

1.1 Background

Object recognition and the subproblem of land cover classification are not new problems and models aiming to tackle the problems exist as far back as 1999 [1], [2]. In this section we present a brief summary of related work, with a more detailed review of related provided in Chapter 2. First, we discuss some general machine learning techniques that may be applied to object recognition and land cover classification. The first technique is deep learning. Deep learning techniques use many layers of hidden nodes which allows them to learn complex functions [3]. Convolutional Neural Networks (CNN) and Recursive Neural Networks (RecNN) are examples of deep learning networks [3], [4]. Both methods can be used as part of a model to perform object recognition [5]. Prior approaches to object recognition included the use of SIFT, Histogram of Oriented Gradients (HOG), textons, shape contexts, a dynamic link architecture, pictorial structures and spin images for point cloud data [1], [6]–[11]. These methods have largely been superseded by CNNs as various state-of-the-art models use CNNs [5], [12]–[14]. These state-of-the-art models include Mask Region-based CNN (Mask R-CNN) and RetinaNet for RGB images and Convolutional-Recursive Neural Network (CNN-RecNN) and the volumetric models from Qi *et al.* [12] for RGB-D images and point cloud data [5], [13], [14]. Many of the techniques applied to object recognition have also been applied to land cover classification [15]. More recently, deep CNNs have also been applied to the problem of land cover classification with multispectral images [2], [16]. We aim to produce models that propose regions of interest in the image and perform classification using RGB-D images with the belief that adding depth will improve the classification accuracy. We also aim to produce models that are able to perform land cover classification using multispectral images. In the next section we detail the problem we are proposing a solution for.

1.2 Problem Statement

General object recognition and land cover classification are still open problems in computer vision [5], [16]. Most approaches to object recognition and land cover classification use RGB images to attempt to build a model to solve this problem [13],

[14]. However, robots, autonomous vehicles, computing devices and satellites are increasingly being equipped with sensors that can measure additional data, such as depth or additional spectra, in a scene [2], [17], [18]. Depth information has the potential to enable models to be built that better solve the problem of object recognition [5]. Typical state-of-the-art systems such as Mask R-CNN and RetinaNet are able to process images of a whole scene which requires that they first detect regions where objects may be and then classify these proposed regions as either a specific object or background. Unfortunately, Mask R-CNN and RetinaNet do not use RGB-D data [13], [14]. The CNN-RNN model proposed by Socher *et al.* (2012) [5] does make use of RGB-D but it requires images that have been broken down into regions of interest containing a single object. Modifying these systems to use RGB-D, for Mask R-CNN and RetinaNet, or to process images that have not been broken down into regions, for CNN-RNN, may prove to be an effective strategy to solve the problem [5], [13], [14]. Speed of the system is another important metric as we would like the systems to be able to perform object recognition at a rate of around 5 fps [13], [14], [19]. Multispectral images also have great potential to improve land cover classification. The models tested in Helber *et al.* [2] and Sumbul *et al.* [16] were simple models designed to display the effectiveness of the the EuroSAT and BigEarthNet datasets they created respectively. There is space for further studies to be performed on the use of multispectral images for land cover classification on these two datasets [2], [16]. Our contributions to the field will be discussed in the next section.

1.3 Significance and Motivation

We produce trained models that can perform object recognition on RGB-D images. We use state-of-the-art object recognition models as our starting point, namely Mask R-CNN and RetinaNet [13], [14]. The addition of the depth channel provides the models with extra information. RGB-D images can be thought of as a 2.5D image. The depth channel provides further information about object edges, shape, size and possibly texture, if the sensors are capable enough. A well designed and properly trained model will be able to extract these details from the RGB-D image helping to improve the accuracy. We measure the impact of adding depth as an additional channel and using the depth information in a separate feature extractor on the training and inference times of the models. We test our models on the SUN RGB-D dataset which contains images captured from four different RGB-D cameras [18]. We also construct a small labelled dataset of around five hundred images captured using a ZED RGB-D camera. We also train and test our models on this dataset. We build and train a novel land cover classification model, LandNet. LandNet is designed to work with multispectral images with single or multiple labels. LandNet uses up to three backbone feature extractor networks and allows for varying which bands are used in each feature extractor which allow us to thoroughly test combinations of bands and feature extractors and demonstrate the benefits of multispectral images for land cover classification. We test LandNet on both EuroSAT and BigEarthNet and demonstrated state-of-the-art performance on BigEarthNet [2], [16]. We also test the impact of using additional bands and multiple feature extractors on the training and inference times of LandNet. The next section outlines our research aims and objectives.

1.4 Research Aims and Objectives

This research project aims to:

- develop a model that first segments images into regions of interest and then performs object recognition incorporating depth and colour information to improve the accuracy; and
- build a dataset consisting of five hundred RGB-D images of various objects with labelled regions; and
- develop a model that performs land cover classification on multispectral images to improve the classification accuracy.

The above aims of this research project will be achieved through the following objectives (goals):

- replicate an object recognition model that uses depth information to be used as a baseline with which to compare our models;
- build a model that performs image segmentation and then object recognition, using depth information in both steps;
- build a model that performs land cover classification using multispectral images
- compare our object recognition model for image segmentation and object recognition with the baseline model on our dataset and one or two other datasets; and
- compare our object recognition and land cover classification models with the results of other competing models from various papers.

1.5 Research Questions

Here we present our research questions that will guide us through the research process. The questions we are aiming to answer are:

- Does adding depth as the fourth channel to existing state-of-the-art models improve accuracy of the models?
- Does processing the depth data separately and then combining the RGB and depth features in a softmax improve the accuracy of existing state-of-the-art models?
- How does adding depth as a fourth channel compare to processing the RGB and depth data separately and combining the RGB and depth features in a softmax?
- Do these models generalize to images captured with a different camera?
- Does adding additional spectra help improve the accuracy of land cover classification models?
- What trade-off between speed and accuracy do we observe as a result of the extra data (depth and additional spectra)?

1.6 Delineations, Limitations and Assumptions

Our goal is to build working models that use RGB-D and multispectral images to more accurately perform object recognition or land cover classification. We are not aiming to improve upon the speed of current models. Optimizing the implementation of the models for speed may be done if time permits.

Our object recognition models will also be restricted to using RGB-D data and not point clouds. Our models will also be restricted to the task of object recognition. They will not perform free space analysis, room layout estimation or scene categorization. Our models will only be able to classify objects that belong to a class that they were trained on. Our models make use of a softmax classifier with a set number of classes and thus they can only classify objects into the classes they have been trained on. We will assume that any object not in one these classes will fall either into the background class or to what the model believes to be the closest class.

The dataset that we will create using our camera will only contain five hundred images which will not be enough data to train any models without some pretraining. The dataset may contain different objects that what are present in the dataset used for pretraining. The next section provides some definitions to aid in the understanding of this proposal.

1.7 Definitions

In this section we briefly give the definitions of lesser known terms used in this research.

- **RGB-D:** Standard three channel RGB images with the addition of a relative depth image or channel.

1.8 Outline

The dissertation is organized as follows: the following chapter details related work to our topic and provides the reader with some relevant background information with the aim of familiarizing the reader with prior work on the topic. The third chapter discusses the methodology that was used to answer our research questions as well as the limitations of our research. In the fourth chapter we detail our results as well as discussing and analysing them. The final chapter will conclude the paper with a brief summary of the research, our results, opportunities for future work and the contributions of our research.

Chapter 2

Background and Related Work

In this chapter we briefly present some background information into useful machine learning techniques for the problem of object recognition followed by a review of previous work on object recognition. We also discuss some prior work done on land cover classification using the EuroSAT and BigEarthNet datasets [2], [16]. The chapter is organised as follows: in the first section we overview some useful machine learning techniques, specifically deep learning techniques, Recursive Neural Networks (RecNN) and Convolutional Neural Networks (CNN). The following two sections discuss previous work on object recognition. The first overviews image segmentation discusses work on object recognition without any depth information, followed by an overview of object recognition with images including depth. The final section gives a brief overview of land cover classification.

2.1 Useful Machine Learning Techniques

In this section we briefly overview some machine learning techniques that are used in other object recognition systems [5], [6], [12], [17], [19]–[21]. We consider three machine learning techniques, the first is Deep Learning, the next is Recursive Neural Networks and the final technique is Convolutional Neural Networks. The aim of this section is not to extensively detail the techniques but rather to give a brief explanation of them. RecNNs and CNNs can be considered a form of Deep Learning as is discussed in the next subsection [22].

2.1.1 Deep Learning

Representation learning is defined as "a set of methods that allows a machine to be fed with raw data and to automatically discover the representation needed for detection or classification". Deep learning systems are representation learning systems with many layers of representation. These many layers of representation are created by composing non-linear subsystems together that in each layer take in data and alter it into a more abstract representation that is then fed into the next layer of representation. With enough layers, it is even possible to learn very complex functions which aids these systems to be able to perform tasks such as classification. Deep learning has already proven itself as a powerful tool to solve many different problems in many different fields [22]. Deep Boltzmann Machines, which are discussed later in this chapter, are a form of deep learning that can be used to perform object recognition [21]. RecNNs and CNNs, which are deep learning techniques, can also be trained to perform object recognition [5], [20], [22].

2.1.2 Recursive Neural Networks

Recursive Neural Networks are similar to Artificial Neural Networks but a key difference lies in the network structure. In RecNNs, the nodes are arranged in a tree structure where each layer in the tree is the same neural network applied again. The aim of this approach is to learn hierarchical feature representations of the input data [5]. RecNNs have been applied to both natural language processing tasks as well as image segmentation and object recognition tasks [4], [5]. These methods are discussed later in the chapter. Like RecNNs, Convolutional Neural Networks are also a modification of the standard Artificial Neural Network architecture [22].

2.1.3 Convolutional Neural Networks

The aim around developing CNNs was to be able to effectively process data in the form of multiple arrays such as signals, images and videos. There are four key ideas utilised in CNNs that allow them to be so effective at these tasks, they are: local connections, shared weights, pooling and the utilisation of a multiple layers [22].

CNNs are typically structured in stages with the initial stages consisting of convolutional layers and pooling layers [22]. In the convolutional layer, the input is convolved with a filter [5]. Since a filter is just a set of weights, the process of convolving the input with a filter can be thought as sharing the same weights across different patches in the input. This has the effect of making the architecture invariant to the location of features in the input. This is useful in images as a feature could appear in different parts of the image. A bank of filters is convolved with the input in each convolutional layer, which makes it possible to look for many different features in the input. The output from the convolutions is then passed through a non-linear activation function such as the sigmoid function or the rectified linear unit (ReLU) before being passed on to the next layer [22].

Pooling takes the output from a convolution layer and combines neighbouring outputs, such as in a rectangular window, using a summary statistic [3]. Different summary statistics may be used. Socher *et al.* (2012) [5] makes use of average pooling where the average value in the window is used, but one can also use max pooling which uses the maximum value in the window. Other pooling techniques include using the L^2 norm of the region or a weighted average. Pooling over spatial regions, such as windows in the output, has the effect of making the representation approximately invariant to small translations. Pooling may also be done over different convolutions which has the effect of making the network learn which transformations of the input to be invariant to [3].

After having used a number of convolution and pooling stages, the final stage usually consists of a fully connected network that forms a classifier that forms the final output of the CNN [22]. The classifier layer need not consist of a fully connected network, other architectures are possible [3]. CNNs, RecNNs and other deep learning techniques often form the basis of object recognition systems but other methods may be used as well, which is shown in the following two sections where we discuss some prior approaches to object recognition [5], [7], [20], [21].

2.2 Object Recognition with RGB Images

In this section we discuss prior work on object recognition with RGB images. In the first subsection we briefly consider classical object recognition models. In the

following subsection we look at some state-of-the-art methods that primarily use some from of CNNs.

2.2.1 Classical Object Recognition Models

Various approaches to the problem of object recognition are considered in classical models. Some of the approaches taken include using shape contexts, using pictorial structures, using a dynamic link architecture to perform invariant object recognition, using textons and local scale invariant features [1], [8]–[11]. An interesting point to note about SIFT and HOG based models is that they bear strong resemblance to the early stages of the object recognition process in primates. Models using local scale invariant features, usually SIFT features, or a Histogram of Oriented Gradients (HOG) have proven popular until recently where CNN based approaches have proven more effective and have superseded SIFT and HOG based models[1], [6]. The next section discusses some of the state-of-the-art techniques which make use of CNNs.

2.2.2 State-of-the-art Object Recognition Models

This subsection is broken into two parts, the first discusses the Regions with CNN features (R-CNN) model and several improvements to it, followed by a brief overview of the RetinaNet model [6], [13].

R-CNN

The R-CNN model performs object recognition in three stages. First, category independent region proposals are extracted from the input image [6]. The method used to do so is the selective search method, but other methods can be used [6], [23]. This generates approximately 2000 region proposals [6]. The proposed regions are then warped to a fixed size of 227 by 227 as the CNN layer requires images of this fixed size. The CNN is used to extract a 4096-dimensional feature vector from each image [6]. The CNN is a Caffe implementation that consists of five convolutional layers and two fully connected layers [6], [24]. The feature vector extracted by the CNN is then fed into a SVM, which performs the final classification. There is an optimized SVM for each image class [6]. Although, R-CNN produces good results, it has a long runtime, a problem which is solved by Fast R-CNN [6], [25].

Fast R-CNN builds upon R-CNN with the primary aim of reducing the time and space complexity of R-CNN. This improvement is achieved by reducing the multi-stage training process to a single stage that jointly learns to classify the objects and improve their spatial locations. Fast R-CNN also boasts better classification performance than R-CNN. Fast R-CNN replaces the SVM with a softmax classifier that outputs the probability of each region belonging to each class. Fast R-CNN also outputs bounding-box regression offsets for each of the object classes. The classification and bounding-box regression are jointly trained using a multi-task loss function. SVD may also be used to further reduce detection time with a small (around 0.3%) loss in mean average precision (mAP) [25]. Faster R-CNN aims to improve R-CNN even further [19].

With Fast R-CNN, the computational bottleneck shifts from the network to the region proposals. To alleviate this bottleneck, Region Proposal Networks (RPN) are used instead of the selection search method to produce region proposals. A RPN is designed to take in an image of any size as input and output a set of rectangular

region proposals. This process is modelled by a fully connected CNN which allows for sharing of convolutional layers between the RPN and the Fast R-CNN. The sharing of the convolutional layers allows the region proposal step to be nearly cost free, greatly enhancing the speed of the model. The RPN also produces better region proposals which improves the accuracy of Faster R-CNN [19]. Faster R-CNN achieved an AP score of 36.2 with a ResNet-101 Feature Pyramid Network (FPN) backbone. Mask R-CNN takes the accuracy of Faster R-CNN ever further [14].

Mask R-CNN leaves the RPN stage of the model unchanged but in the second stage Mask R-CNN outputs a binary mask for each proposal in parallel with the class and box offsets [14]. The addition of the binary mask output improves the accuracy of the model, allowing it to outperform baseline variants of other state-of-the-art models but the additional computation required by adding in the binary mask output does slightly reduce the speed of the system. Mask R-CNN was able to achieve a bounding box AP score of 39.8 when using a ResNeXt-101 FPN backbone [14], [26]. R-CNN is an example of a two-stage model where first region proposals are extracted from each image and then the classification is performed on each proposed region. It is also possible to have one-stage models where detectors are applied over dense sampling of possible object locations, one such model is RetinaNet [13].

RetinaNet

One-stage object detectors have traditionally trailed the performance of two-stage detectors such as R-CNN. In Tsung *et al.* (2017) [13] they identify the large imbalance between the background and foreground classes as the primary cause. To rectify this imbalance, they propose a new Focal Loss function which effectively deals with the class imbalance by significantly reducing the contribution of easy examples to the loss function during training. The Focal Loss function is used in RetinaNet. RetinaNet uses a simple architecture consisting of FPN backbone that is built on top of a ResNet architecture [13], [27]. The resulting convolutional features are then fed into two subnets, one for classifying each box, and another for regressing from the estimated boxes to the ground truth boxes [13]. Using this simple structure and the new Focal Loss function, RetinaNet achieved a bounding box AP score of 40.8 with a ResNeXt-101 FPN backbone and is able to outperform the state-of-the-art object detectors including Mask R-CNN in the COCO benchmark [13], [28]. RetinaNet also runs noticeably faster than two-stage models [13]. All the systems discussed thus far make use of RGB images but RGB-D images or point clouds can also be used [5], [17]. The next section discusses models that make use of 3D data such as RGB-D images and point clouds.

2.3 Object Recognition using 3D Images

In this section we give an overview of various systems that make use of depth for object recognition. The first subsection considers models that use surface matching [7]. The next section discusses volumetric approaches and the final subsection details the Convolutional Recursive Neural Network model (CNN-RecNN) [5], [12], [17].

2.3.1 Surface Matching

Surface matching is a technique where an observed 3D object surface is matched to a known object surface stored in a computer's memory. Shape representations are

used to efficiently compare surfaces. In Johnson *et al.* [7] they use spin images as a data level descriptor to represent and compare surfaces. To further improve the speed of the system, principal component analysis is used as a compression technique to reduce the dimensionality of the spin images. To perform object recognition, first multiple spin images of the scene are created around different points. Then each scene spin image is correlated against each model spin image, with best match indicating both the best matching model and point. In contrast to the models discussed next, the method presented in Johnson *et al.* [7] does not make use of CNNs [5], [12], [17].

2.3.2 Volumetric Models

Volumetric convolutions are spatially 3D convolutions. An example of a model that makes use of volumetric CNNs is VoxNet. VoxNet operates on point cloud data which is used to create an occupancy grid, a voxel representation of the scene. The occupancy grid allows them to estimate free, occupied and unknown space. The occupancy grid is used as input to a 3D CNN which performs the classification [17]. Although VoxNet boasts impressive accuracy, Qi *et al.* [12] notes that volumetric models trail behind multi-view CNNs [17].

In Qi *et al.* [12] they propose three improvements to volumetric models. They propose two different network architectures as the first improvement. The first network aims to reduce overfitting by introducing subtasks where the network is tasked with predicting classes from partial volumes. The second network is crafted to mimic multi-view CNNs using an anisotropic kernel to reduce 3D shapes into 2D. The next improvement involves augmenting the data by changing the elevation and azimuth rotations. This helps as the two networks' structures are sensitive to rotations of the scene. The final improvement is multi-orientation pooling which again helps as the networks are sensitive to changes in the orientation of the scene. These improvements allowed the models presented in Qi *et al.* to produce more accurate results than VoxNet [17]. Unlike VoxNet and the models in Qi *et al.* [12], the CNN-RecNN model does not use 3D CNNs [5], [12].

2.3.3 Convolutional Recursive Neural Network

The CNN-RecNN model uses a combination of a CNN and a recursive neural network to solve the problem of object recognition. The model makes use of separate networks for the RGB images and the depth images [5]. The filters for the CNN layer are also learned separately using the K-means method described in Socher *et al.* (2011) [4]. The filters are used in a single CNN layer that takes the full image as input. The resulting output for each filter from the CNN layer is average pooled before being used as input to a number of random fixed-tree RecNNs. The aim of using the RecNNs is to learn hierarchical feature representations of the input. Instead of training a RecNN, they found that using a number of random RecNNs produced good quality feature vectors. The purpose of the networks is to extract features from the images. The features from the RGB and depth images are combined into a single feature vector which is used as input into a softmax classifier that then outputs the inferred class of the object in the image. It is important to note that the model does not use an image of a complex scene as input, but rather just an image of an individual object [5]. The dataset the CNN-RecNN model was trained on does not contain a background class [29]. It may be possible to use various techniques, such as selection search or RPNs, to first generate region proposals for the CNN-RecNN

model [19], [23]. The next section gives a short overview of land cover classification techniques presented with the EuroSAT and BigEarthNet datasets. While models for land cover classification share some similarities with models for object detection, such as the use of CNNs and other deep learning techniques, there are also many differences since they usually do not attempt to find regions belonging to each class within a single image but instead only label the image with one or more classes [2], [16].

2.4 Land Cover Classification

In this section we will first provide a brief overview of land cover classification. Then we will discuss the approaches taken by Helber *et al.* [2] and Sumbul *et al.* [16] to solve the land classification problem on the EuroSAT and BigEarthNet datasets.

2.4.1 Land Cover Classification

Many land cover classification datasets consist of either aerial or satellite RGB images with a single label or multiple labels for each image [2], [16]. Much of the computer vision, machine learning, feature extraction and classification techniques that exist can be applied to single or multi-label image datasets. Many of these methods have been applied to land cover classification and studied extensively [2], [15], [30]. Some common approaches include using deep belief networks, CNNs, and stacked denoising autoencoders [30]. Castelluccio *et al.* [15] performed an extensive evaluation of various machine learning techniques; including the use of scale-invariant feature transform (SIFT), CNNs, and spatial pyramids; for land cover classification. As we used EuroSAT and BigEarthNet for our tests, we present the methods employed to these datasets in greater detail in the next two subsections [2], [16].

2.4.2 EuroSAT

In presenting their dataset, EuroSAT, Helber *et al.* [2] demonstrated a few different machine learning methods that could be used on EuroSAT for the purpose of land cover classification. These methods included a Bag of Visual Words (BoVW) with a Support Vector Machine, a shallow CNN, a pretrained GoogLeNet and a pretrained ResNet-50 [2], [27], [31]. Helber *et al.* [2] used various combinations of the thirteen different bands of the Sentinel-2 mission in their experiments. They achieved good results on the EuroSAT dataset, with their best classifier achieving 98% accuracy. The best performing classifier was the ResNet-50 model, using only RGB images without the rest of the bands. This result is surprising as one would expect that adding extra information, in this case in the form of the additional bands, would lead to a higher accuracy. Helber *et al.* [2] theorizes that they saw the highest accuracy from using the RGB bands as the ResNet was pretrained on ImageNet, which is an RGB dataset, even when they used other band combinations [2]. This result differs from what Sumbul *et al.* [16] were able to achieve BigEarthNet which we will discuss in the next subsection.

2.4.3 BigEarthNet

Sumbul *et al.* [16] compared the performance of two models on their BigEarthNet multi-label dataset, one of which they tried training on just the RGB bands as well as all the bands. The two models they tested were a shallow CNN and a pretrained

Inception-v2 network. The Inception-v2 network they used was pretrained on ImageNet and they only performed fine tuning on the BigEarthNet dataset, using just the RGB bands. With this, they were able to achieve a precision value of 48.23% and a recall value of 56.79%. They improved upon these results with a shallow CNN consisting of three convolutional layers, a fully connected (FC) layer and a classification layer. They did not use pretrained weights for this network, rather the weights were randomly initialised and full training was performed on the BigEarthNet dataset. Tests were performed on both the RGB bands and all the bands for this network. With just RGB bands, they saw a precision value of 65.06% and a recall value of 75.57%. Using all bands resulted in a precision value of 69.93% and a recall value of 77.10% [16]. This improvement is in contrast to what Helber *et al.* [2] saw and does show that the additional information from the extra bands can help to improve the results of models with only small modifications. The following section concludes this chapter.

2.5 Conclusion

In this chapter we have presented various building blocks, specifically deep learning, CNNs and recursive neural networks, that can be used to solve the problem of object recognition. We outlined some classical models for object recognition followed by a discussion of the current state-of-the-art systems for object recognition on RGB images. Next, we looked at some models that make use of depth data for object recognition. Finally, we concluded the chapter by looking at land cover classification on EuroSAT and BigEarthNet [2], [16]. The next chapter details the techniques and models we will use to solve this problem based on what we have gleaned from the literature.

Chapter 3

Research Methodology

3.1 Introduction

In this chapter we detail the methodology that we used to answer the research questions posed in the first chapter. We also outline the datasets that we used to train and test the various models that we built and we detail our RGB-D dataset that we constructed. The chapter is organised as follows; first we describe Wits RGB-D. Next we detail the methodology for our object recognition experiments and following this we present the methodology used for our land cover classification models. After that we provide implementation details and discuss the metrics and analysis performed. Finally, we discuss limitations of our research before concluding the chapter.

3.2 Wits RGB-D

Wits RGB-D is an annotated RGB-D image dataset that we built using images captured by a Zed Stereo Camera. We used the Zed Stereo Camera to take videos at a resolution of 1920×1080 and then later scrubbed through the recorded videos, selected good and varied frames and saved the left RGB image and the depth image of the same frame. For the depth image we used the ultra quality setting with the confidence set to 97% in the Zed toolkit with filling enabled and a maximum depth range of 2 meters. We did not process the depth images any further. Figure 3.1 shows the left RGB image and the corresponding depth image. It is easy to see that even with fill enabled, not all the missing values are able to be filled in in the depth image. We have noticed that the camera struggles to determine the depth for parts of the scene that are dark or very bright. This is something that the models will need to be able to overcome. Both the RGB and depth images have a resolution of 1920×1080 pixels. Unfortunately some of the images do appear a bit blurry likely due to either the motion of the camera or delays in the focus. An example is shown in Figure 3.2. We decided to leave these few blurry images in the dataset as some blurriness is something that often occurs in images in reality. Some of the images are slight translations or rotations but generally these contain either another object or part of one or removes part of an object. An example of this is shown in Figure 3.3a and 3.3b. Although these images look very similar, there are some differences. In Picture 1 there is a mug that is mostly visible, but in Picture 2 this mug is almost cut out completely. Another difference between these is around the top right of the images, where in Picture 1, there are more boxes that do not appear in Picture 2. Other objects in the image are also seen from a slightly different angle.

Wits RGB-D contains 500 annotated images most of which were taken around The University of the Witwatersrand (Wits). Wits RGB-D contains images mostly of computer labs and open plan offices, but it also includes images of shared rest areas and

TABLE 3.1: Table showing the number of objects belonging to each class in Wits RGB-D.

Class	Class Name	Number of Objects
1	Monitor	1413
2	Chair	846
3	Keyboard	580
4	Mouse	488
5	Cellphone	236
6	Table	24
7	Sofa	187
8	Whiteboard	92
9	Fridge	17
10	Microwave	20
11	Mug	113
12	Box	428
13	Bottle	101
14	Travel mug	4
15	Rubbish bin	9
16	Coffee machine	19
17	Wallet	56
18	Bag	116
19	Desktop PC	157
20	Laptop	244
21	Blackboard duster	6
22	Headphones	8
23	Lamp	76
24	Speaker	145
25	Router	114

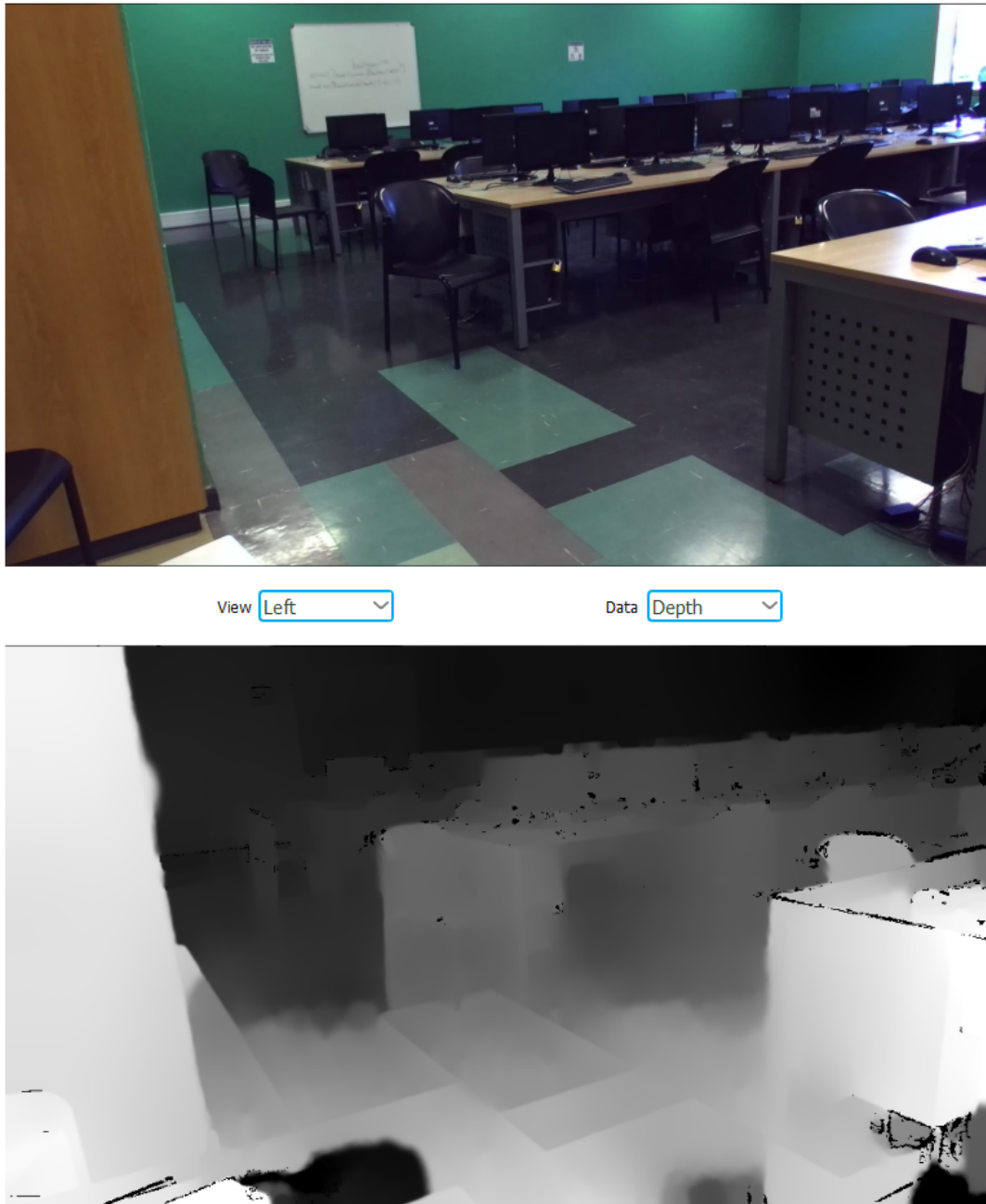


FIGURE 3.1: The left RGB image and the depth image as displayed in the Zed Depth Viewer.



FIGURE 3.2: There is some blurriness in the image, but objects are still distinct.



(A) Picture 1

(B) Picture 2

FIGURE 3.3: Comparison between two similar images.



FIGURE 3.4: Example annotated image from Wits RGB-D.

TABLE 3.2: Table showing the classes in SUN RGB-D.

Class	Class Name
1	Bathtub
2	Bed
3	Bookshelf
4	Box
5	Chair
6	Counter
7	Desk
8	Door
9	Dresser
10	Garbage bin
11	Lamp
12	Monitor
13	Night stand
14	Pillow
15	Sink
16	Sofa
17	Table
18	TV
19	Toilet

an office kitchen. The objects contained in the dataset vary from large objects such as chairs and sofas to small objects such as mice and cellphones. Wits RGB-D contains 25 different object classes with a total of 5499 objects for an average of 11 objects per image. Each image contains at least one object, with some images containing over 40 objects. Due to the chosen scenes for the dataset, Wits RGB-D contains many instances of some classes, such as chair and monitor, but few of other classes, such as fridge. This class imbalance is something that can be fixed in future revisions of Wits RGB-D. The full distribution of objects in each class is shown in Table 3.1. We used the LabelImg tool to label the images in Wits RGB-D [32]. The labels are created in the Pascal VOC format, we then run Python script to convert the labels to the COCO format [28], [33]. An annotated image from Wits RGB-D is shown in Figure 3.4. The next section details the methodology used for our experiments with object recognition.

3.3 Object Recognition

In this section we describe our methodology for testing object recognition. First, we detail the datasets we use for testing. In the following subsection we detail the model design of Mask R-CNN + nC and RetinaNet + nC.

3.3.1 Data

Both Mask R-CNN and RetinaNet were trained on the COCO2017 dataset with pre-training performed on the ImageNet1k dataset [13], [14], [28]. Unfortunately, neither dataset contains RGB-D images. We still used pretrained models for Mask R-CNN + nC, RetinaNet + nC and LandNet. We trained and tested Mask R-CNN + nC, RetinaNet + nC and RPN CNN-RNN on SUN RGB-D [18], [34]–[36]. SUN RGB-D is a

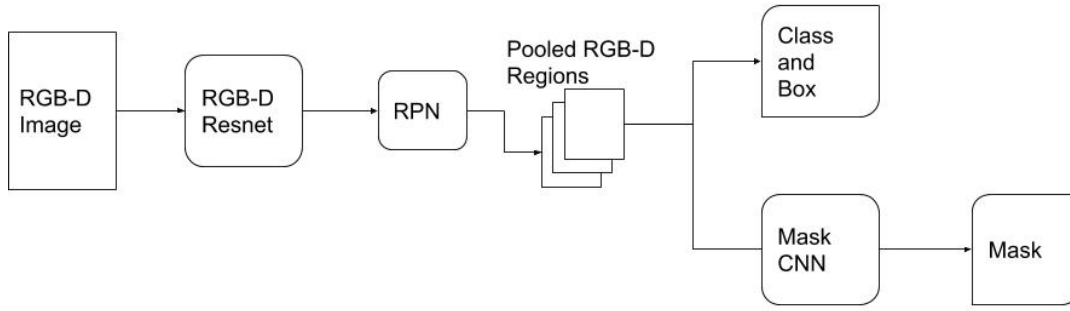


FIGURE 3.5: Structure of Mask R-CNN + nC_1 with one RGB-D network. The RGB-D image is fed to the ResNet as an image with four channels. The extracted features are then passed through the RPN which generates region proposals. These are then pooled together and fed through a subnet which performs classification and bounding box regression and another subnet that generates an object mask.

RGB-D image dataset with annotations for object recognition and other tasks. For object recognition, SUN RGB-D contains 19 classes which are shown in Table 3.2. SUN RGB-D uses images from multiple different RGB-D cameras, specifically the Kinect v1, Kinect v2, Intel RealSense and Asus Xtion cameras. In total, SUN RGB-D contains 10335 images split into 5285 training images and 5050 testing images. In addition to testing on the SUN RGB-D, we also tested Mask R-CNN + nC , RetinaNet + nC and RPN CNN-RNN on our own RGB-D dataset that we built. We also created our own RGB-D dataset that we detail in a later section in this chapter. SUN RGB-D is not a land cover classification dataset and so other datasets were needed to train and test LandNet [18].

In this section we begin by detailing the various models that we used. Each is discussed in their own subsection. The following subsection details the implementation of our models. Next, we describe the datasets that we used to train and test our various models and how the training was performed. Next, we discuss the metrics that we used for determining the performance of our models and finally how we analysed the results. Detailed analysis of our results is presented in the following chapter.

3.3.2 Models

In this subsection, the various models we used are detailed, including implementation details. First we present Mask Region-based CNN plus n Channels (Mask R-CNN + nC), followed by RetinaNet plus n Channels (RetinaNet + nC) both of which are based on the respective models discussed in Chapter 2 [13], [14]. Finally, we present a novel model, LandNet, to perform land cover classification.

Mask R-CNN + nC

The first model we present is based on the Mask R-CNN model from [14]. We augmented the original in two ways. First, we added an additional channel, depth, as the fourth channel in the image and tested if this improves the classification accuracy. While we have tested with adding depth as an additional channel, it is possible to add other image-like (such as additional bands) information as additional channels. This model is denominated as Mask R-CNN + nC_1 . Figure 3.5 visualises this

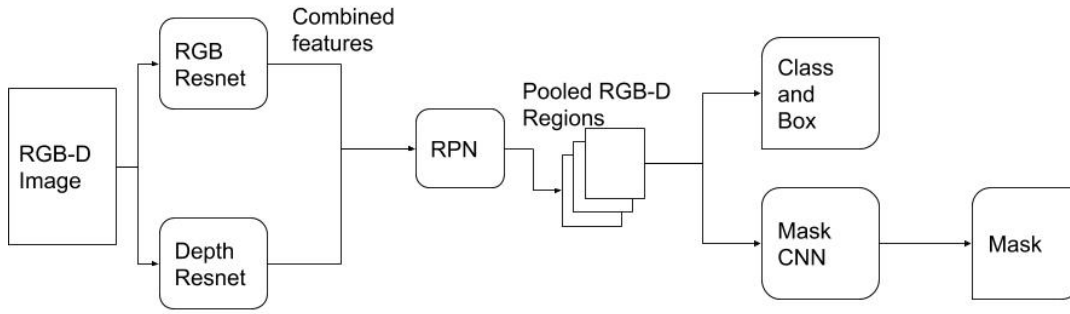


FIGURE 3.6: Structure of Mask R-CNN + nC with separate RGB and additional channel networks. Features are extracted from the RGB channels and depth channel separately. The extracted features are then appended together before being processed and pooled in the RPN layer. The pooled features for each proposed object detection are then fed through a subnet to perform classification and bounding box regression. The features are simultaneously fed through another subnet to infer the object mask.

network design.

The second modification we took a similar approach as is done in CNN-RecNN, that is, to have two separate networks, one for RGB and another for additional channels, such as depth, and then combine the output from the ResNet backbone network into a single output that is used as input to the later stages of the model [5], [14], [25]. This model is denominated as Mask R-CNN + nC₂. The 1024 channel output of the fourth stage of the RGB and additional channel ResNets is concatenated as a 2048 channel input to the RPN. This change also required us to increase the number of input channels to the first convolutional layer of the RPN to 2048. The number of output channels of this convolutional layer was not modified. Concatenating the features before the first convolutional layer should allow the RPN to extract some correlation between the high level RGB and additional channel features. The concatenated features are then passed through the ROI-Align layer of Mask R-CNN, which then outputs the pooled regions. Since this process does not affect the number of channels, the RGB and additional channel features are still present in the output. The RGB and additional channel features are then inputted separately into the fifth stage of the RGB and additional channel ResNets. The outputs from both networks are then average pooled separately before being concatenated together. This is then used as input to the fully connected (FC) layers responsible for performing bounding box regression and producing class scores. The concatenated features are also used as input to the mask layer. This network design is shown in Figure 3.6. Unfortunately, there were difficulties with getting the required masks for the mask subnet from the SUN RGB-D dataset so the mask subnet was disabled for all of our experiments. Disabling the mask branch will have reduced the overall mAP score, but it does not affect the comparison between using RGB and RGB-D images [14]. The implementation of the various parts of the model was kept the same as in [14] except for any of the changes we noted here.

As is standard practice, we used ResNets pretrained on the ImageNet1k dataset. We replaced only the input layer of the additional channel ResNet and the RGB-D ResNet due to the difference in the number of input channels compared to the original ResNet [6]. The final training of the model was performed on the SUN RGB-D

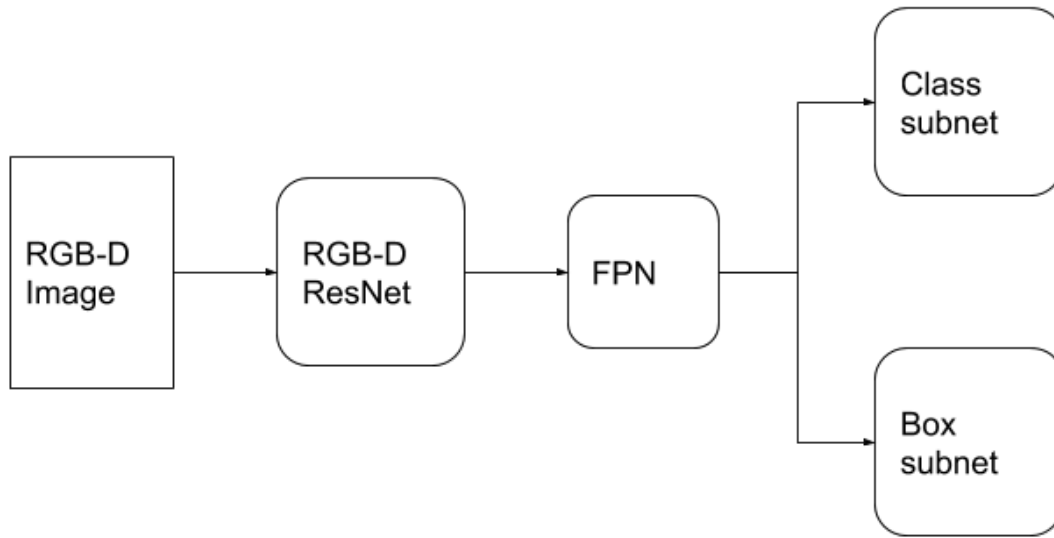


FIGURE 3.7: Structure of RetinaNet + nC_1 with one RGB-D network. The RGB and depth channels are used as single input image to initial feature extractor. The extracted features are then used two separate subnets; the first classifies the object in the bounding box, the other determines the bounding box of the object.

dataset [18]. We did not modify the loss functions used in Kaiming *et al.* [14]. We also utilize the same initial learning rate as in Kaiming *et al.* [14]. We reduced the learning rate by a factor of 10 every five epochs and we performed a total of 10 epochs over the SUN RGB-D training split [18]. We augmented the data by horizontally flipping each image and training on these flipped together with the unmodified images. We did not retrain the models for testing on Wits RGB-D, instead we performed fine-tuning. Fine-tuning was achieved by freezing all but the final layers of model, that is, the bounding box regression and class score FC layers, and then fine-tuning the weights of the those two layers at a low learning rate of 0.001 for ten epochs over the training data. Similar modifications were made to RetinaNet.

RetinaNet + nC

As with Mask R-CNN plus n Channels, RetinaNet plus n Channels is based on respective RetinaNet model with two augmentations to allow it to work with depth or other additional channels[13]. The first modification was to add depth as a fourth channel to the original RetinaNet [13]. We denominate this model as RetinaNet+ nC_1 and this is visualized in Figure 3.7.

The next modification was to create a separate network for RGB channels and additional channels (such as depth) in the same way as CNN-RecNN [5]. The extracted features were combined as a single vector as input to the classification and box-regression subnets [5], [13]. This is shown in Figure 3.8.

We initialised the models in the same way as in Tsung *et al.* [13] where possible. Any new layers to the ResNet or FPN where initialised as in He *et al.* (2015) [37]. We used the Stochastic Gradient Descent (SGD) algorithm with the learning rate initially set to 1e-3 for training our model. The learning rate is reduced by a factor of 10 every ten epochs over the training dataset. The learning we used differs from that used in Tsung *et al.* [13] as we were having problems with the model diverging with the higher learning rate used in Tsung *et al.* [13]. We used pretrained ResNets for the

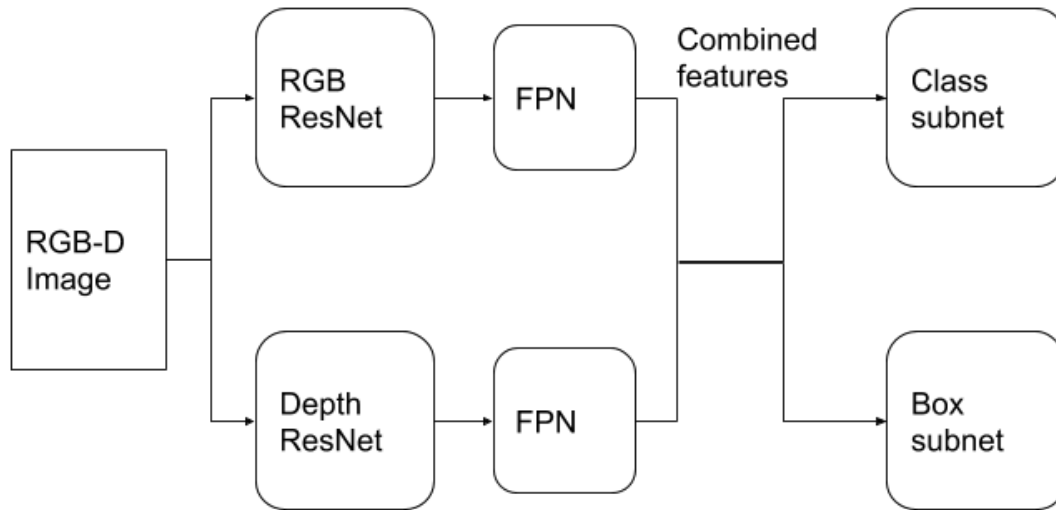


FIGURE 3.8: Structure of RetinaNet + nC_1 with separate RGB and depth networks. The RGB and depth channels are fed as input into two separate ResNets and FPNs to extract features from each. The features are then appended together to form a single set of features. These features are then used in a classification subnet to identify which class the object in the bounding box belongs to and a bounding box regression subnet to determine the bounding box of the object.

model and performed the final training on SUN RGB-D [18]. As with Mask R-CNN + nC , we performed fine-tuning on Wits RGB-D. We did this by freezing all layers in the models except the final convolutional layers in the bounding box regression and class score subnets. We then fine-tuned those two convolutional layers using a low learning rate of 0.001 for twenty epochs over the training dataset.

In the next section we discuss our methodology for our experiments with land cover classification.

3.4 Land Cover Classification

This section focusses on the problem of land cover classification. First we discuss the datasets we used to train and test LandNet. This is succeeded by a detailed presentation of LandNet.

3.4.1 Data

Land cover classification datasets usually consist of either images taken from aircraft or images taken from satellites. We decided to use datasets containing multispectral images taken from satellites. EuroSAT and BigEarthNet are datasets consisting of multispectral images taken from the Sentinel-2 mission [2], [16]. The Sentinel-2 mission has a short repeat cycle of just five days which allows the predicted land cover classification to be updated frequently. The images taken by the Sentinel-2 mission contain 13 different bands [2]. The various bands and a description of them are given in Table 3.3. Bands B01, B09 and B10 are usually used for correcting atmospheric conditions and have been omitted from the table as we did not use them in any of our tests. Different bands also vary in spatial resolution from as low as 10m per pixel up to 60m per pixel. Usually the lower resolution bands are upsampled to be

TABLE 3.3: Bands of images from the Sentinel-2 mission. The bands used for correcting for atmospheric conditions have been omitted. Table adapted from [2].

Band Number	Sentinel-2 Band	Description	Wavelength (nm)
1	B02	Blue	490
2	B03	Green	560
3	B04	Red	665
4	B05	Red edge 1	705
5	B06	Red edge 2	740
6	B07	Red edge 3	783
7	B08	Near infrared (NIR)	842
8	B08A	Red edge 4	865
11	B11	Short wave infrared (SWIR) 1	1610
12	B12	SWIR 2	2190

the same resolution as the highest resolution bands.

EuroSAT contains 27000 64×64 image patches taken from various Sentinel-2 images over European cities. EuroSAT contains 10 classes. Each image is labelled as belonging to a single class [2]. BigEarthNet is a more recently published dataset compared to EuroSAT, having been published in 2019 [2], [16]. BigEarthNet is also far larger dataset as it contains 590236 image patches. Each image patch is 120×120 pixels. BigEarthNet is a multi-label dataset with 43 unique classes. 70987 of the images in BigEarthNet are fully covered by cloud, seasonal snow and cloud shadow and we removed these images as suggested by Sumbul *et al.* [16]. We tested LandNet on both EuroSAT and BigEarthNet. As with Mask R-CNN + nC and RetinaNet + nC, the ResNets were pretrained on ImageNet1k. Next we present the LandNet models which we train and test on these datasets.

3.4.2 LandNet

We designed a network architecture, that we call LandNet, to solve the problem of land cover classification. Both Helber *et al.* [2] and Sumbul *et al.* [16] demonstrated the usefulness of using CNNs for land cover classification. Furthermore, Helber *et al.* also showed the potential of very deep CNNs, such as a ResNet-50, in solving this problem. Based on their results we decided to use a ResNet-101 as the feature extractor. Helber *et al.* [2] used a ResNet-50 pretrained on ImageNet1k with no modifications. Likewise, we also used a ResNet-101 pretrained on ImageNet1k, but we did make some changes. Taking some cues from Socher *et al.* (2012) [5], we decided to test using different ResNet-101 feature extractors for different bands in the images, up to a total of three different feature extractors. When a band combination resulted in a different number of input to channels than the three that the ResNet was pretrained with, we replaced the first convolution layer with a new randomly initialised one with the required number of input channels. We randomly initialised the weights with a Gaussian distribution with a mean 0 and a standard deviation of 0.01. We took the output of each of the separate feature extractors after the final average pool layer that occurs just before the final FC layer, and concatenated them

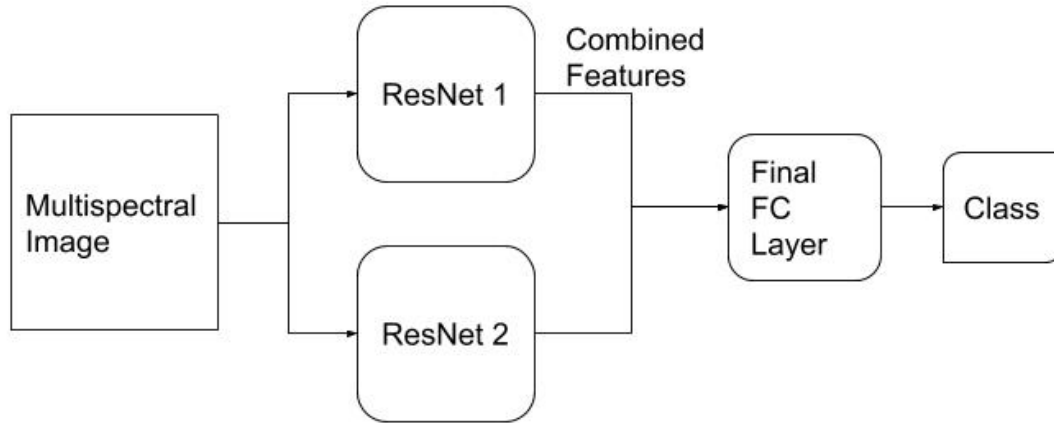


FIGURE 3.9: Structure of LandNet with 2 ResNet feature extractors. Different bands can be used as input to the two ResNets which extract features from the image. The combined feature vector is then passed through a final fully connected (FC) layer. The final feature vector is then fed into a softmax to determine the class.

into a single feature vector before passing it through a FC layer. The output of the FC layer then passes through a softmax to get the final predicted class vector. Due to the small size of the images in EuroSAT, we had to modify the average pool so that it pools over 4×4 pixels instead of 7×7 as in ResNet normally [27]. For EuroSAT we used the cross entropy loss function, but since BigEarthNet is a multi-label dataset, we could not use the same loss function for both datasets. We used the binary cross entropy loss when training on the BigEarthNet dataset [16]. We trained models with various combinations of bands and numbers of feature extractors.

For training LandNet we used the stochastic gradient descent algorithm with a low fixed learning rate of $1e-4$ on the EuroSAT dataset. For training on BigEarthNet, we started with an initial learning rate of $1e-2$, which we then decreased to $1e-3$ after 3 epochs over the dataset. We trained all the layers with the same learning rate. We did not fix the batch normalization layers for LandNet. For training on EuroSAT we used a random 80/20 training/test split with the split applied to each of the ten classes of the dataset. We chose this split as Helber *et al.* [2] showed that this split performed the best in their case. Likewise, we used a random 80/20 training/test split on BigEarthNet. Before splitting the dataset, we removed image patches with seasonal snow as well partial shadow and cloud cover from the dataset using the list provided by BigEarthNet. As mentioned above, we tried various band splits with up to three separate feature extractors. We trained each model for 9 epochs over the EuroSAT training images and 5 epochs on BigEarthNet [2], [16]. The combinations are shown in Table 3.4. In our various combinations we tried keeping similar bands together, e.g. we used the red edge bands and near infrared (NIR) bands together in one ResNet and RGB and short wave infrared bands in two other ResNets in combination 9. We also tried using just RGB, just the red edge bands as just the SWIR bands. In combination 3, we use a special combination of bands that is useful for detecting forests. We use this special combination of bands in other combinations as well. In the next section we detail the implementation of all our models.

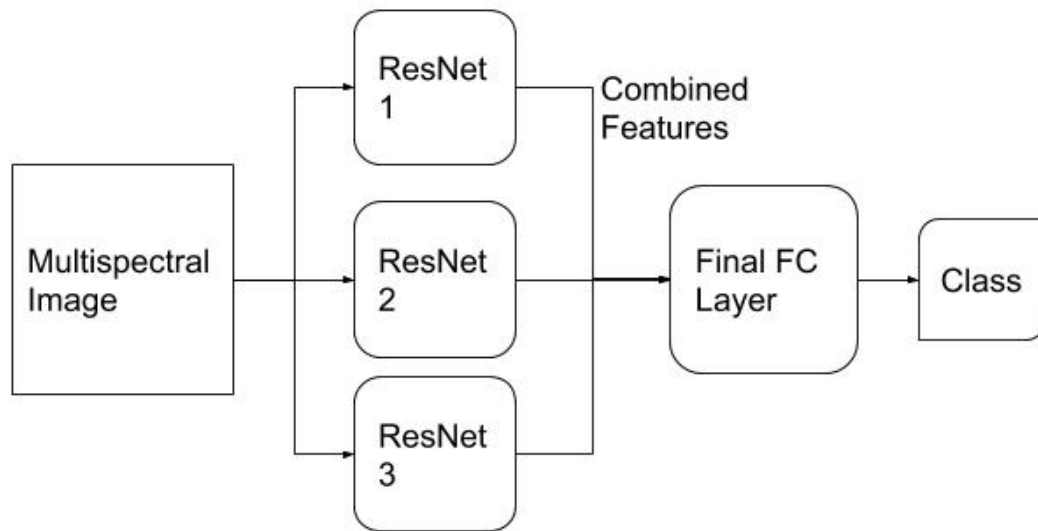


FIGURE 3.10: Structure of LandNet with 3 ResNet feature extractors. Different bands are used as input to three separate ResNets. The outputted features are combined and sent through a FC layer before being used as input into a softmax function which determines the class.

TABLE 3.4: Various band and feature extractor combinations tested for LandNet. N/A indicates that that ResNet was not used. Details of each band can be seen in Table 3.3.

Combination Number	Bands in ResNet 1	Bands in ResNet 2	Bands in ResNet 3
1	1, 2, 3	N/A	N/A
2	1, 2, 3, 4, 5, 6, 7, 8, 11, 12	N/A	N/A
3	3, 7, 11	N/A	N/A
4	4, 5, 6, 7, 8	N/A	N/A
5	11, 12	N/A	N/A
6	1, 2, 3	4, 5, 6, 7, 8, 11, 12	N/A
7	1, 2, 3	3, 7, 11	N/A
8	1, 2, 3	3, 7, 11	4, 5, 6, 8, 12
9	1, 2, 3	4, 5, 6, 7, 8	11, 12

3.5 Implementation

All our models were implemented in PyTorch 0.4.1. Our implementation of Mask R-CNN is based the implementations of Yang *et al.* [38] and Tseng [39]. We made modifications to these implementations to allow the use of RGB-D images and to support having multiple feature extractors. Our implementation of RetinaNet uses parts of the implementation by Henon [40]. For LandNet, we used the ResNet implementation from Yang *et al.* [38]. We implemented the rest of model ourselves. All of our models were run on a machine containing two 16 core Intel Xeon processors, 256GB of RAM and a Nvidia GTX 1080Ti. All the other models used the GPU for training. The next subsection discusses the dataset we used to train our models.

3.6 Metrics and Analysis

Here we discuss the metrics used to test and analyse the performance of our models and follow this by detailing the analysis we perform.

3.6.1 Precision, Recall and F-score

Precision, recall and the F-score are widely used to measure the performance of machine learning models. Precision and recall are calculated from the true positives (TP), false positives (FP), false negatives (FN) and true negatives (TN). Precision is defined as:

$$Precision = \frac{TP}{TP + FP} \quad (3.1)$$

Recall is defined as:

$$Recall = \frac{TP}{TP + FN} \quad (3.2)$$

The F-score is the "weighted, harmonic mean of precision and recall" and is defined as:

$$F_\beta = (1 + \beta^2) \frac{precision \times recall}{recall + \beta^2 precision} \quad (3.3)$$

[41]

3.6.2 Mean Average Precision

Mean Average Precision (mAP) is a metric that is commonly used to test object recognition systems [6], [13], [14], [19], [25]. As the name suggests mAP is a mean over the average precision of a model [28]. In order to define mAP, first we must define Intersection over Union (IoU) and Average Precision (AP) which are used to calculate mAP [28], [33].

IoU

IoU measures the overlap between the ground truth object boundary with the predicted object boundary [33]. Mathematically, it is defined as:

$$IoU = \frac{\text{area of intersection}}{\text{area of union}}$$

[33]. IoU is used to determine the correctness of a prediction [28], [33]. Normally, an IoU value of 0.5 or greater is taken as correct but the IoU threshold can be varied as is done in the mAP value calculated for the COCO Challenge [28], [33].

AP

Average precision is the mean precision at eleven different recall values, denoted r , ranging from 0 to 1.0 with steps of 0.1 and is defined mathematically as:

$$\begin{aligned} AP &= \frac{1}{11} \sum_{r \in \{0.0, \dots, 1.0\}} AP_r \\ &= \frac{1}{11} \sum_{r \in \{0.0, \dots, 1.0\}} p_{interp}(r) \end{aligned} \quad (3.4)$$

where $p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r})$. $p_{interp}(r)$ is the interpolated maximum precision for all recall values greater than or equal to r [33].

mAP

mAP is the mean AP over all the classes in the dataset. In Lin *et al.* [28] they also vary the IoU values from 0.5 to 0.95 in steps of 0.05.

3.6.3 Analysis

We analysed the performance of our object recognition models in two ways. The first is classification accuracy, that is, how well does it classify the objects in the image. This is usually scored using mean average precision (mAP) [6], [13], [14], [19], [25]. We compared the scores of our models with depth information to the original models on the SUN RGB-D dataset to analyse the impact on performance by adding in depth information. We also tested the classification accuracy of our land cover classification models. For testing on EuroSAT, we just used the classification accuracy, but since BigEarthNet is a multi-label dataset we used the precision, recall, F_1 and F_2 scores to test how well the models predicted the class labels. We present the micro and macro averaged results for the precision, recall and F_1 . The micro average of the metrics is calculated from the global TPs, FPs and FNs. For the macro average, the precision, recall and F-score are calculated for each class and then the averaged across the classes [2], [16]. Run time performance is another metric that we used.

We measured the time taken to train our models as well as the time taken to perform inference with the trained models. We compared these results to the original RGB trained on SUN RGB-D models [18]. We did not include the time taken to pretrain the models on the ImageNet1k dataset as we downloaded the pretrained model weights. We will also analyse the training and inference times of our land cover classification models in the same way.

We present a measure of the efficiency of the modifications to the models. This efficiency compares the percentage increase in classification accuracy to the percentage in training time. We define it as:

$$efficiency = \frac{\text{percent increase in accuracy}}{\text{percent increase in training time}} \quad (3.5)$$

An efficiency score of 1 would imply that the percent increase in accuracy was achieved with an equal percent increase in training time. An efficiency score less

than 1 means that the percent increase in training time exceeded the percent increase in accuracy, while a score greater than 1 shows the opposite, that is, that the percent increase in accuracy exceeded the percent increase in training time. If accuracy did not increase or even decreased, then we define efficiency to be zero. A higher efficiency score is better. As we expect that there may be variance in the time to perform an epoch over the data between the same models and models with the same number of backbone feature extractors, in the case that a model completes an epoch in less time than the baseline model, we treat it as if there was no change in the time taken between the baseline and the model. As Equation 3.5 does not work in this case, we define efficiency as:

$$\text{efficiency} = 1 + \text{percent increase in accuracy} \quad (3.6)$$

Ideally, we would not like the efficiency score to be significantly below 1, but, as we will show in Chapter 4, this is usually the case. We calculate efficiency using the $mAP_{0.5:0.95}$ score for the Mask R-CNN and RetinaNet based models. We use the micro F_1 score to calculate the efficiency for the LandNet models trained on BigEarthNet. The following section discusses the limitations of our work.

3.7 Limitations

There are several limitations to our research. First, we do not aim to create completely novel models, we attempt to show that adding depth information can produce better results than just using RGB images. Second, we only consider the problem of object detection and recognition. We do not attempt to produce 3D bounding boxes, we do not try to analyse free space in room, we do not aim to be able to estimate the layout of rooms and we do not attempt to perform total scene understanding [18]. Finally, we limit our models to work with RGB-D images and not point clouds or video streams. The next section concludes the chapter.

3.8 Conclusion

In this chapter we have outlined the methodology that we will use to answer our research questions. We discussed the models we will use, how we modify them, how they will be trained and tested. We also detailed the datasets that we plan to use and the metrics we will use to analyse the performance of our models. We then concluded the chapter by discussing some limitations of our research.

Chapter 4

Results

4.1 Introduction

In this chapter we present and discuss our results. In the first section we present our object recognition results, analyse and discuss them. In the following section, display analyse and discuss our land cover classification results. Next, we discuss some limitations of the results and then the final section concludes this chapter.

4.2 Object Recognition

In this section we will present our object recognition results. We divide the results into two subsections, the first subsection shows the classification accuracy while the second subsection details the training and inference times of the various models. The final subsection shows the efficiency of all the models on the SUN RGB-D and Wits RGB-D.

4.2.1 Classification Results

We begin presenting our results with Table 4.1 shows the mean AP (mAP) scores of the models on the SUN RGB-D dataset [18]. Table 4.1 contains the mAP score with Intersection over Union (IoU) values ranging from 0.5 to 0.95 in steps of 0.05, which is also the average score over results for each class shown in the next table, as well as the mAP score at IoU values of 0.5 and 0.75. It also shows the mAP for small, medium and large objects. The objects are binned into these three sizes based on the

TABLE 4.1: mAP scores achieved by our models on SUN RGB-D.

Model	$mAP_{0.5:0.95}$	$mAP_{0.5}$	$mAP_{0.75}$	mAP_S	mAP_M	mAP_L
Mask R-CNN	9.31	24.07	5.35	0	0.74	12.08
Mask R-CNN (8 epochs)	9.68	24.27	6.13	0.02	0.68	12.52
Mask R-CNN + nC ₁	7.03	19.06	3.48	0	0.38	9.03
Mask R-CNN + nC ₁ *	8.93	22.04	5.21	0	1.08	11.34
Mask R-CNN + nC ₁ * (8 epochs)	9.69	23.37	6.42	0	0.94	12.26
Mask R-CNN + nC ₂	5.93	17.97	2.14	0	0.4	7.53
Mask R-CNN + nC ₂ (8 epochs)	7.79	19.76	4.92	0	0.44	10.0
RetinaNet	7.3	16.17	5.52	0.02	0.39	9.29
RetinaNet + nC ₁	4.94	11.32	3.5	0.01	0.25	6.03
RetinaNet + nC ₁ *	5.36	12.52	3.82	0	0.19	6.59
RetinaNet + nC ₂	4.74	11.83	3	0.02	0.25	5.89

TABLE 4.2: AP scores achieved by our models for each class in SUN RGB-D. Table 3.2 contains the class names. Class is abbreviated as Cl. Mask R-CNN is abbreviated as MR. Mask R-CNN + nC₁ is abbreviated as MR₁. Mask R-CNN + nC₂ is abbreviated as MR₂. RetinaNet is abbreviated as RN. RetinaNet + nC₁ is abbreviated as RN₁. RetinaNet + nC₂ is abbreviated as RN₂. The asterisk indicates models using the improved initialisation.

Cl	MR	MR(8)	MR ₁	MR ₁ *	MR ₁ *(8)	MR ₂	MR ₂ (8)	RN	RN ₁	RN ₁ *	RN ₂
1	2.1	3.6	9.0	12.0	13.9	9.0	7.8	0.2	2	2.2	1.5
2	26.8	27.3	22.5	29.7	27.4	18.4	29.1	24.2	19.9	21.3	17.9
3	9.7	7.5	3.1	6.8	6.6	4.4	5.4	7.5	1.7	2.6	1.5
4	0.3	0.4	0.1	0.2	0.3	0.2	0.4	0	0	0	0
5	10.5	11.4	8.8	11.2	12.5	6.2	9.1	10.5	8	8.7	8.3
6	15.4	17.4	6.7	8.4	12.2	6.9	10.3	11.7	6.4	8.5	7.9
7	4.4	4.4	3.2	4.5	4.8	2.7	3.5	1.4	0.5	1.1	1.4
8	9.8	11.1	0.8	1.6	2.0	3.1	3.5	7.5	1	1.5	1.3
9	7	6.7	2.4	2.7	3.5	2.5	3.3	0.3	0	0	0.1
10	8	9.3	3.7	5.4	6.1	2.0	5.0	7.1	1.1	1.9	1.8
11	3.6	3.7	3.5	6.6	6.4	3.4	2.3	2.2	2.4	2	2.5
12	1.6	1.9	1.6	2.0	2.3	1.4	1.8	0	0.4	0	0
13	7	7.4	2.2	6.8	6.5	3.2	5.0	0.2	0	0.6	0
14	1.7	1.5	0.3	1.5	0.8	0.6	1.4	1.9	1.6	0.8	1.3
15	6.8	7.5	7.1	5.5	10.2	4.1	0.7	9.6	1.5	4.3	4.5
16	13.3	12.3	11.5	16.3	15.0	10.5	13.8	11	9.3	9.5	8.7
17	11.2	12.2	11.9	13.5	14.8	9.7	11.5	11.7	8	11	10
18	9.1	7.5	5.3	6.1	7.8	3.1	5.0	4.5	1.9	3.7	1.8
19	28.5	30.4	29.8	29.1	31.1	21.3	29.3	27.2	28.1	22.2	19.4

TABLE 4.3: mAP scores achieved by our models on Wits RGB-D.

Model	$mAP_{0.5:0.95}$	$mAP_{0.5}$	$mAP_{0.75}$	mAP_S	mAP_M	mAP_L
Mask R-CNN	0.02	0.08	0	0	0	0.02
Mask R-CNN (8 epochs)	0.02	0.14	0	0	0	0.03
Mask R-CNN + nC ₁	0.04	0.14	0.02	0	0	0.04
Mask R-CNN + nC ₁ *	0.14	0.35	0.14	0	0	0.14
Mask R-CNN + nC ₁ * (8 epochs)	0.18	0.55	0.2	0	0	0.18
Mask R-CNN + nC ₂	0.06	0.32	0	0	0	0.07
Mask R-CNN + nC ₂ (8 epochs)	0.06	0.3	0	0	0	0.06
RetinaNet	0.06	0.1	0.07	0	0	0.06
RetinaNet + nC ₁	0.03	0.13	0.02	0	0	0.03
RetinaNet + nC ₁ *	0.03	0.13	0.01	0	0	0.03
RetinaNet + nC ₂	0.01	0.05	0	0	0	0.01

TABLE 4.4: Training and inference times for various models on SUN RGB-D.

Model	Training Time (minutes)	Inference Time (minutes)
Mask R-CNN	32.48	11.60
Mask R-CNN + nC ₁	31.25	12.43
Mask R-CNN + nC ₂	61.25	22.16
RetinaNet	19.78	6.010
RetinaNet + nC ₁	20.82	7.420
RetinaNet + nC ₂	33.47	9.570

area of the objects in pixels in each image. These mAP scores are calculated with the IoU ranging from 0.5 to 0.95. In Table 4.2 we dig deeper into the results and provide show the average precision (AP) scores of the models for each class in the SUN RGB-D dataset. We use several abbreviations in Table 4.2 so that the table fits within the width of the page. We abbreviate Class as Cl, Mask R-CNN as MR, Mask R-CNN + nC₁ as MR₁, Mask R-CNN + nC₂ as MR₂, RetinaNet as RN, RetinaNet + nC₁ as RN₁, RetinaNet + nC₂ as RN₂ [13], [14]. We use an asterisk to indicate that the model is trained using the improved initialisation method. When checking our results after each epoch we noticed that the Mask R-CNN + nC models trained for 8 epochs performed better than they did after 10 epochs and thus we include these results as well. We provide Table 4.3 which provides the same metrics as in Table 4.1 metrics but for our models trained on Wits RGB-D. We omit a table displaying the per class results on Wits RGB-D as the models perform poorly with the majority of classes seeing near-zero scores and thus limiting the benefit of a more close examination of these results.

In the next subsection, we continue our results by presenting the training and inference times of our models.

4.2.2 Training and Inference Times

The training time is the time taken in minutes to complete one epoch over the training data. The inference time is the time taken in minutes to perform inference over the entire testing data. We begin by showing the training and inference times of our models on SUN RGB-D in Table 4.4. We do not present the training and testing times on Wits RGB-D as the results should be similar to those attained on SUN RGB-D and since we do not train the feature extractors on Wits RGB-D.

4.2.3 Efficiency

In this subsection we present the efficiency score of models that we defined in Equation 3.5 and Equation 3.6. First, we present the efficiency of the Mask R-CNN and RetinaNet models on SUN RGB-D in Table 4.5. We do not present the efficiency results on Wits RGB-D as we do not present the training time results on Wits RGB-D. In the next section we analyse and discuss these results.

TABLE 4.5: Efficiency scores for the models on SUN RGB-D.

Model	Increase in Accuracy (%)	Change in time (%)	Efficiency
Mask R-CNN	Baseline	Baseline	1
Mask R-CNN + nC_1	Decrease	Decrease	0
Mask R-CNN + nC_1^*	0.1	Decrease	1.1
Mask R-CNN + nC_2	Decrease	88.57	0
RetinaNet	Baseline	Baseline	1
RetinaNet + nC_1	Decrease	5.26	0
RetinaNet + nC_1^*	Decrease	5.26	0
RetinaNet + nC_2	Decrease	69.21	0

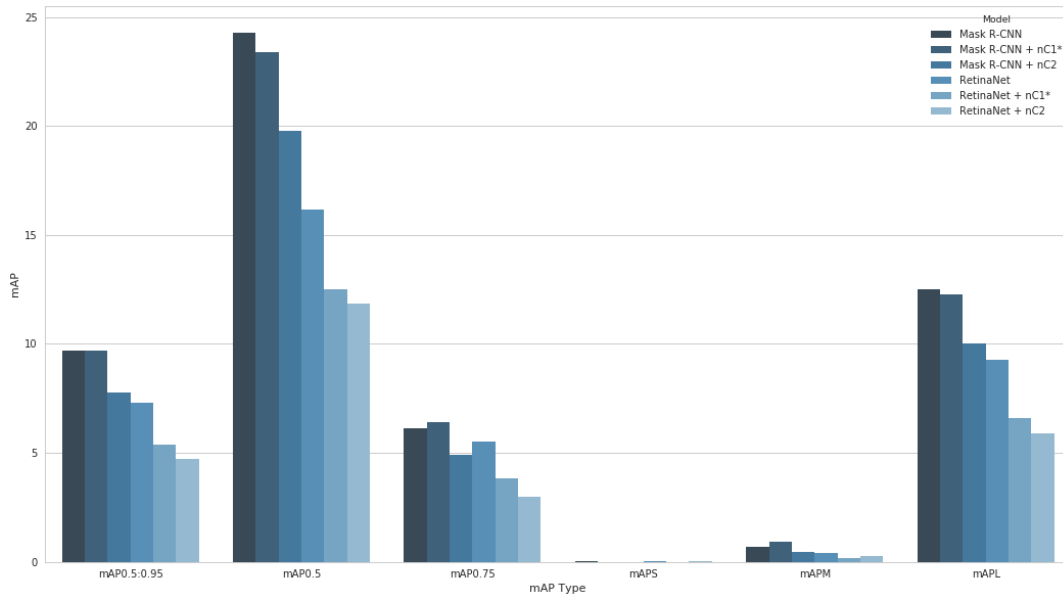


FIGURE 4.1: Bar graph showing mAP scores of Mask R-CNN and RetinaNet on SUN RGB-D.

4.3 Analysis and Discussion

We split this section into two subsections. In the first subsection we focus on analysing and discussing the results from the Mask R-CNN and RetinaNet based models on both SUN RGB-D and Wits RGB-D. The following section discusses the dataset, Wits RGB-D, its strengths and areas for improvement.

4.3.1 Mask R-CNN and RetinaNet

This section concentrates on analysing and discussing the results achieved by the Mask R-CNN and RetinaNet models. We further split this subsection into two subsections. The first subsection focuses on the results attained on SUN RGB-D. Both accuracy and the training and inference times are discussed in this subsection. This is followed by a subsection that examines the accuracy results on Wits RGB-D.

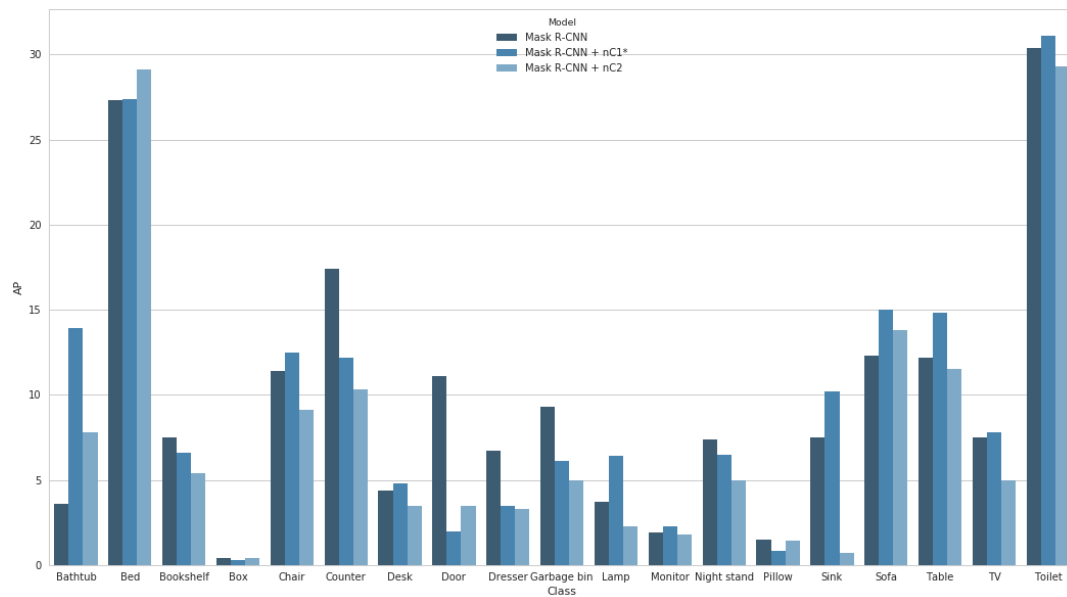


FIGURE 4.2: Bar graph showing accuracy of Mask R-CNN models trained for 8 epochs on SUN RGB-D.

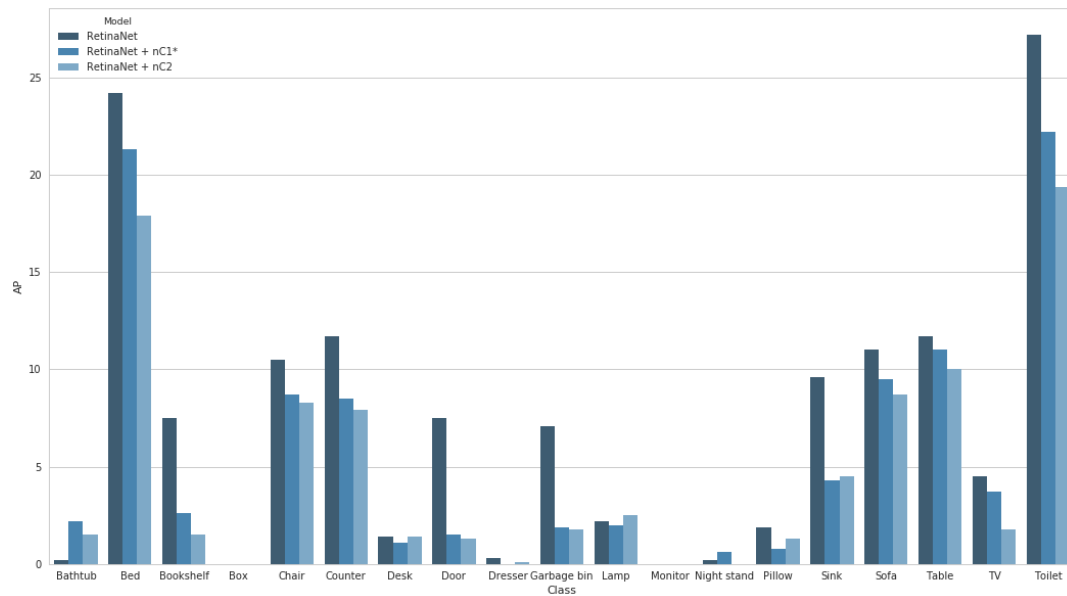


FIGURE 4.3: Bar graph showing accuracy of RetinaNet models on SUN RGB-D.

Results on SUN RGB-D

We start our analysis with the results on SUN RGB-D. We provide Figure 4.1 which presents the mAP results for Mask R-CNN and RetinaNet from Table 4.1 in a bar graph to allow comparing the results. Next we provide Figure 4.2 and Figure 4.3 which present the results for the Mask R-CNN and RetinaNet models contained in Table 4.2 in such a way that we can easily compare the results between the various models and classes. Unexpectedly, we do not see an improvement, but actually a reduction to mAP by adding the additional depth channel. A possible explanation for this is simply an inadequate amount of training data for the models with additional channels. This is likely due to the fact that the first convolutional layer in the pre-trained ResNet backbones need to be replaced so that the models are able to process the images with additional channels.

We see that the improved initialisation where we keep the initial weights for the RGB channels in the first CNN layer when adding depth as the fourth channel of the models helps to improve the accuracy. Additionally, we see hints of overtraining the Mask R-CNN models as the mAP decreases from 8 to 10 epochs. When we compare the results of Mask R-CNN and Mask R-CNN + nC_1 at 8 epochs, we see that the models perform similarly. Mask R-CNN achieves better results than Mask R-CNN + nC_1 with Mask R-CNN+ nC_1 close behind in some of the mAP measurements and vice versa. The same is not true when we consider the RetinaNet models.

Using the improved initialisation with RetinaNet + nC_1 does improve the results however the results are still not as good as RetinaNet. RetinaNet + nC_1 still trails RetinaNet by a significant amount across all the various mAP measurements. Unlike Mask R-CNN, RetinaNet did not suffer from any overtraining. This could be due to the fact that we use a lower learning rate for RetinaNet than the learning rate we use Mask R-CNN and suggests that training the RetinaNet models further may achieve improved results.

Comparing the results between Mask R-CNN and RetinaNet, we see that the various Mask R-CNN models always perform better than their RetinaNet counterparts. One would expect that RetinaNet would outperform Mask R-CNN based on the results presented in He *et al.* [14] and Lin *et al.* [13] where RetinaNet is shown to perform noticeably better than Mask R-CNN. The cause of the underwhelming performance of the RetinaNet models may be due to the lower learning rate that was used in order to avoid instability in the focal loss function. The lower learning rate may have forced a need for more training epochs for the model to converge and reach peak performance. As we test on a different dataset, namely SUN RGB-D, to He *et al.* [14] and Lin *et al.* [13] which tested on the COCO2017 dataset, the dataset may also be a factor in the differences in placement of the models. RetinaNet and the focal loss function were designed to improve performance in situations where there is a large class imbalance. SUN RGB-D may have a less class imbalance with more of the background being labelled classes when compared to COCO2017, thus reducing the benefits of the focal loss function. SUN RGB-D is also a significantly smaller dataset than COCO2017. It is possible that RetinaNet may require a larger dataset for optimal training.

Diving into the results for individual classes, we can see from Figure 4.2 that while Mask R-CNN trained with only Red Green Blue (RGB) images had an overall higher mAP score than Mask R-CNN + nC_1 and Mask R-CNN + nC_2 , it does not have a higher AP score for every class than both the other models. In fact, the Mask R-CNN models with the additional depth channel did manage to achieve a three times higher AP for the bathtub class. There are other classes where the addition of depth

has improved the results such as lamp, sofa, table and TV. We believe that the extra information about shape and relative location that can be extracted from the depth channel assisted Mask R-CNN + nC_1 and Mask R-CNN + nC_2 to better distinguish bathtubs from the background and other classes. While Mask R-CNN + nC_2 generally performs worse than Mask R-CNN and Mask R-CNN + nC_1 , it does manage to return a higher accuracy score on the bed class and comes close to matching the other models in other classes such as desk and monitor. This is not always the case as it does poorly on the sink class, achieving an accuracy of 0.7 compared to the 10.2 achieved by Mask R-CNN + nC_1 . There are also some classes where the addition of depth has had a large negative impact on accuracy. A prime example of this is the door class where the accuracy of Mask R-CNN + nC_1 and Mask R-CNN + nC_2 is less than half of the accuracy of the Mask R-CNN. The reason for this may be due the fact that doors may be at an angle to the camera, causing the depth of the door to vary over the surface of the door and confusing the boundaries of the door when using the depth image. There some similarities and differences to the results on the Mask R-CNN models when looking at the per class results of RetinaNet, RetinaNet + nC_1 and RetinaNet + nC_2 .

Examining the results per class of RetinaNet, RetinaNet + nC_1 and RetinaNet + nC_2 reveals some key differences in performance compared to the Mask R-CNN models. The first notable difference is that the depth models pull ahead of the RGB models for only 4 classes, namely, bathtub, lamp and night stand. The differences in the accuracy is not convincingly in the favour of the depth models, with the most significant improvement being an increase of 2 for the bathtub class when comparing RetinaNet and RetinaNet + nC_1 . However, there are a few classes where the RetinaNet convincingly outperforms both RetinaNet + nC_1 and RetinaNet + nC_2 . As with the Mask R-CNN models, the door class is a notable example of where RetinaNet proved superior to the models using depth. Other notable examples of this include the garbage bin, sink and bookshelf classes. It is interesting to note that when considering the sink class, RetinaNet performs better than RetinaNet + nC_1 , RetinaNet + nC_2 and Mask R-CNN, however, the addition of depth allows Mask nC_1 to outperform RetinaNet. This is notable as adding depth works against RetinaNet + nC_1 , RetinaNet + nC_2 and Mask R-CNN + nC_2 , but is favourable for Mask R-CNN + nC_1 . This is also one of the two classes where a RetinaNet model shows better results than any of the Mask R-CNN models, the other class being pillow. In general, the RetinaNet models perform worse than the Mask R-CNN models. There are also a few cases where they the difference is staggeringly in the favour of Mask R-CNN, such as the monitor class, where the RetinaNet models achieve an AP of 0 but the Mask R-CNN models at least achieve an AP around 2. Another interesting example of this is the night stand and dresser classes. The RetinaNet models struggle to correctly identify these classes achieving a near zero AP while the Mask R-CNN classes manage an AP of around 5.

Unfortunately, not only did the mAP scores not improve, but the time taken to train Mask R-CNN + nC_1 , Mask R-CNN + nC_2 , RetinaNet + nC_1 and RetinaNet + nC_2 and perform inference with the trained models also increased compared to Mask R-CNN and RetinaNet. The percent increase in training time for the models, as well as the efficiency scores is shown in Table 4.5. We see that adding an additional channel, as is done in Mask R-CNN + nC_1 and RetinaNet + nC_1 , only results in a minor increase in the training and inference times. This is not an unexpected finding as the majority of calculations in deep neural networks occur in the hidden layers of the network. The calculations in the initial layer only make up a fraction of the total number of calculations in a deep neural network.

Adding an additional backbone feature extractor network as is done in Mask R-CNN + nC_2 , nearly doubles the training time and inference times when comparing Mask R-CNN and Mask R-CNN + nC_2 . However, the same doubling is not seen when comparing the training and inference times of RetinaNet and RetinaNet + nC_2 . In this case, the training time increased by 69.21% and the inference time increased by 59.23%. While the backbone feature extractors are doubled up in Mask R-CNN + nC_2 and RetinaNet + nC_2 , the convolutional subnets used for classification and bounding box regression are not doubled up. This fact accounts for the less than two times increase in training and inference times. Additionally, the smaller increase in training and inference times of RetinaNet + nC_2 compared to Mask R-CNN + nC_2 shows that the convolutional subnets in RetinaNet accounts for a larger percentage of calculations of the model than the convolutional subnets in Mask R-CNN. There is also the case with Mask R-CNN + nC_1 where the training time decreased when compared to Mask R-CNN. As the decrease was not significant, the cause of this phenomenon can be attributed to stochastic differences in the Linux scheduler and the computer state. The times were measured for a single epoch and were not averaged over a number of epochs, causing the measurements to be susceptible to variations in scheduling and the state of the machine.

Having a closer look at the efficiency scores shown in Table 4.5, we see that since almost all the models saw a decrease in accuracy compared to the baseline models, the modal score is zero. There is only a single model that managed to outperform the baseline, that is Mask R-CNN + nC_1 . Since the training time, which is used to calculate the efficiency, decreased slightly, Mask R-CNN + nC_1 managed to achieve an efficiency score of 1.1. The results on Wits RGB-D and differences between the various models do not follow the patterns set by SUN RGB-D.

Results on Wits RGB-D

Right away, it is evident from Table 4.3 that the accuracy results on Wits RGB-D are notably low. The highest $mAP_{0.5:0.95}$ we recorded was only 0.18 and the highest $mAP_{0.5}$ was just 0.55. These scores were achieved by the same mode, namely Mask R-CNN + nC_1 using the improved initialisation strategy and trained for 8 epochs on SUN RGB-D before being fine-tuned on Wits RGB-D. The low $mAP_{0.5}$ score indicates that, at best, the model only correctly detects and classifies objects less than 1% of the time. The worst $mAP_{0.5:0.95}$ goes as low as 0.01, which is simply atrocious. These poor results prompt us to take a closer look at potential issues with Wits RGB-D, which we do in the following section.

Examining the differences between the various models reveals some interesting trends that differ to what we see on SUN RGB-D. First, the addition of depth always leads to an improvement in accuracy for Mask R-CNN + nC_1 and Mask R-CNN + nC_2 when compared to Mask R-CNN. This goes against the results on SUN RGB-D where the addition of depth led to reduced accuracy compared to the RGB models. Only Mask R-CNN + nC_1 with the improved initialisation and trained for 8 epochs was able to match Mask R-CNN when trained on SUN RGB-D, however, with Wits RGB-D we see all the various Mask R-CNN models using depth outperform the RGB model. The additional depth information may help in this case due to small size of the dataset. The additional depth information results in more data that the models are trained on, which could reduce the ill-effects of the small number of training images. As we discuss in the next section, the dataset contains many difficult to detect objects. The depth information may help detect these objects. The results with the RetinaNet models follow the patterns set on SUN RGB-D.

With the RetinaNet models, we see that RetinaNet outperforms RetinaNet + nC_1 and RetinaNet + nC_2 when looking at the $mAP_{0.5:0.95}$ score. However, when examining the $mAP_{0.5}$, it is evident that RetinaNet + nC_1 is able to achieve slightly better results than RetinaNet. RetinaNet + nC_2 is consistently the worst performer across both the RetinaNet models and the Mask R-CNN models. RetinaNet is able to provide a better $mAP_{0.5:0.95}$ score than Mask R-CNN, which is different to what we see on SUN RGB-D where Mask R-CNN achieved better scores. As already stated, adding depth only reduces the accuracy for RetinaNet + nC_1 and RetinaNet + nC_2 and as such, Mask R-CNN + nC_1 and Mask R-CNN + nC_2 perform better than the RetinaNet models.

A possible cause of the poor performance of all our models on Wits RGB-D could be due the models use of Non-Maximum Suppression (NMS) and the large amount of overlap between objects in the dataset. Both models make use of NMS when performing inference. NMS is used to reduce the number of predicted boxes by suppressing the number of low confidence boxes and overlapping boxes. As many of the objects overlap, it is likely that the NMS is discarding some of these boxes, resulting in a reduced accuracy. Usually, the NMS aides accuracy as it reduces the number of incorrectly classified boxes, but in this case, it may be working against the models.

Another interesting point to note is that Wits RGB-D contains a large number of instances of the monitor class. From the results on SUN RGB-D, we see that the models do not perform well on the monitor class present in SUN RGB-D. The large number of instances of the monitor class in Wits RGB-D may be negatively affecting the results. This is further exasperated by the fact that many instances of the monitor class are partially obscured in Wits RGB-D, making the class even more challenging than it appears to be in SUN RGB-D. We discuss more about the Wits RGB-D dataset in the next section.

4.3.2 Wits RGB-D

In this section we analyse the dataset we created as part of this research, namely Wits RGB-D. We look at both the strengths and weaknesses of this dataset. We start by discussing the types of images in the dataset. Wits RGB-D mostly contains images taken in various areas around the campus of the University of the Witwatersrand. The areas are open to students and were not set up or modified for the purpose of taking images for Wits RGB-D. Thus, the images are natural and the types of images that a model would expect to find in the real world. The images were taken in various positions in these areas to provide different views of the same objects so that models can be trained to recognise objects from a variety of points of view. As the environments were not modified before taking the images, there are many images where there are many overlapping objects and partially obscured objects. This increases the difficulty of the dataset, the partially obscured objects represent quite the challenge in particular.

Wits RGB-D contains only 500 images, split into 400 training images and 100 testing images. The low number of images in the dataset creates a challenge for deep learning models which require large amounts of training so that the models are able to converge and perform well. A larger number of images would help with the partially obscured objects as well, as it would provide more samples of these objects for the models to learn on. The small number of images restricts the usefulness of Wits RGB-D and makes it difficult to use for training a model from scratch. Wits RGB-D is best used for fine tuning models pretrained on other larger datasets.

TABLE 4.6: Classification accuracy for various LandNet configurations on EuroSAT.

Combination Number	Bands	Classification Accuracy (%)
1	(1-3)	92.35
2	(1-8, 11, 12)	93.52
3	(3, 7, 11)	96.63
4	(4-8)	90.11
5	(11, 12)	88.02
6	(1-3), (4-8, 11, 12)	96.87
7	(1-3), (3, 7, 11)	93.76
8	(1-3), (3, 7, 11), (4-6, 8, 12)	96.31
9	(1-3), (4-8), (11,12)	95.37

TABLE 4.7: Classification accuracy for various LandNet configurations on BigEarthNet.

Combo. Num.	1	2	3	4	5	6	7	8	9
Micro Precision	0.82	0.82	0.82	0.81	0.74	0.82	0.82	0.81	0.82
Macro Precision	0.73	0.70	0.73	0.66	0.62	0.75	0.74	0.75	0.72
Micro Recall	0.72	0.69	0.72	0.67	0.51	0.74	0.73	0.75	0.74
Macro Recall	0.48	0.46	0.47	0.40	0.24	0.49	0.50	0.50	0.52
Micro F_1	0.77	0.75	0.77	0.73	0.60	0.78	0.77	0.78	0.78
Macro F_1	0.55	0.53	0.54	0.46	0.29	0.57	0.57	0.57	0.58
Micro F_2	0.74	0.71	0.74	0.69	0.54	0.75	0.75	0.76	0.75
Macro F_2	0.52	0.49	0.51	0.43	0.27	0.53	0.53	0.54	0.55

Wits RGB-D contains a large number of classes, 25, for the size of the dataset. Wits RGB-D also faces the challenge of class imbalance. We see that there are some classes where there are over 1000 instances of the class, such as the monitor class, while many others have fewer than 100 instances of the class. Worse still, is that there are some classes, namely travel mug, rubbish bin, blackboard duster and headphones, that appear fewer than 10 times in the dataset. Due to the low numbers of occurrences of these classes and the random training/test split, it is likely that instances of these classes do not appear in the testing dataset at all. The solution for this issue would be to hand pick at least some images for the testing dataset that contain these instances of these classes. In the next section we turn our attention to the results from the LandNet models.

4.4 Land Cover Classification

4.4.1 Results

We present the classification accuracy on EuroSAT in Table 4.6 [2]. The accuracy represents the percentage of correct classifications. Table 4.7 completes our land cover classification results by presenting the micro and macro precision (P), recall (R), F_1 , F_2 scores of various LandNet models on BigEarthNet [16]. We also present the accuracy scores of configurations 6 and 9 of LandNet and the number of samples in the testing set for each class of BigEarthNet in Table 4.8. Next we provide the

TABLE 4.8: Classification accuracy for each class of BigEarthNet using LandNet configurations 6 and 9 as well as the number of samples in each class.

Class	Samples	Con. 6 P	Con. 9 P	Con. 6 R	Con. 9 R	Con. 6 F_1	Con. 9 F_1
1	35310	0.84	0.83	0.81	0.82	0.83	0.83
2	32717	0.89	0.89	0.85	0.86	0.87	0.87
3	36749	0.88	0.88	0.84	0.83	0.86	0.86
4	29677	0.71	0.71	0.65	0.64	0.68	0.68
5	28520	0.83	0.81	0.73	0.75	0.78	0.78
6	26133	0.72	0.71	0.66	0.64	0.69	0.67
7	20754	0.71	0.76	0.72	0.63	0.72	0.69
8	19706	0.81	0.79	0.70	0.73	0.75	0.76
9	11528	0.87	0.91	0.85	0.81	0.86	0.85
10	14675	0.99	0.99	0.98	0.98	0.99	0.99
11	13151	0.85	0.84	0.71	0.72	0.78	0.78
12	6011	0.77	0.88	0.86	0.72	0.81	0.79
13	3298	0.77	0.75	0.69	0.68	0.73	0.71
14	2697	0.84	0.83	0.52	0.56	0.65	0.67
15	2292	0.71	0.73	0.49	0.46	0.58	0.56
16	2179	0.76	0.69	0.41	0.46	0.53	0.55
17	2551	0.79	0.76	0.35	0.42	0.49	0.54
18	2224	0.74	0.74	0.75	0.75	0.75	0.75
19	2126	0.90	0.76	0.61	0.82	0.72	0.79
20	2013	0.85	0.85	0.73	0.70	0.78	0.77
21	1889	0.81	0.69	0.34	0.44	0.48	0.54
22	1406	0.78	0.70	0.26	0.34	0.39	0.46
23	1135	0.71	0.70	0.21	0.21	0.32	0.33
24	1106	0.61	0.61	0.43	0.39	0.50	0.47
25	997	0.82	0.71	0.18	0.22	0.29	0.34
26	954	0.85	0.68	0.21	0.27	0.33	0.38
27	803	0.68	0.69	0.36	0.31	0.47	0.43
28	759	0.73	0.71	0.53	0.57	0.62	0.63
29	654	0.60	0.56	0.20	0.24	0.30	0.33
30	578	0.77	0.71	0.58	0.62	0.66	0.67
31	340	0.55	0.51	0.13	0.25	0.21	0.34
32	300	0.81	0.80	0.59	0.60	0.68	0.69
33	232	0.64	0.69	0.28	0.30	0.39	0.42
34	264	0.80	0.61	0.37	0.47	0.51	0.53
35	312	0.94	0.94	0.83	0.79	0.88	0.85
36	229	0.60	0.63	0.07	0.05	0.12	0.10
37	200	0.79	0.62	0.42	0.66	0.55	0.63
38	194	0.76	0.68	0.47	0.49	0.58	0.57
39	165	1.00	0.81	0.02	0.10	0.05	0.18
40	165	0.43	0.47	0.09	0.08	0.15	0.14
41	89	0.73	0.70	0.34	0.35	0.46	0.47
42	95	0.73	0.75	0.42	0.59	0.53	0.66
43	63	0.00	0.00	0.00	0.00	0.00	0.00

TABLE 4.9: Training and inference times for various LandNet configurations on EuroSAT.

Combination Number	Training Time (seconds)	Inference Time (seconds)
1	109.94	12.59
2	103.34	12.39
3	101.94	12.36
4	102.46	12.46
5	110.28	12.95
6	195.39	23.62
7	185.62	23.24
8	269.36	33.85
9	271.99	34.85

TABLE 4.10: Training and inference times for various LandNet configurations on BigEarthNet.

Combination Number	Training Time (seconds)	Inference Time (seconds)
1	1525.28	160.27
2	1552.01	162.26
3	1538.06	161.04
4	1532.00	158.24
5	1535.44	157.73
6	3015.93	271.87
7	3006.04	268.99
8	4591.12	387.53
9	4612.19	389.44

training and inference times of the various LandNet configurations on EuroSAT and BigEarthNet.

Training and Inference Times

We start by presenting the training and inference times for various LandNet configurations on EuroSAT in Table 4.9. We present the training and inference times for our LandNet models on BigEarthNet in Table 4.10. In the following section we present our efficiency results for LandNet.

Efficiency

First we display the efficiency of the various LandNet models on EuroSAT in Table 4.11. We then complete the presentation of our results by showing the efficiency results of LandNet on BigEarthNet in Table 4.12.

4.4.2 Analysis and Discussion

In this section we look at the results from the various LandNet models. We split this section into two subsections. In the first subsection we investigate and discuss the

TABLE 4.11: Efficiency scores for various LandNet configurations on EuroSAT.

Combination Number	Increase in Accuracy (%)	Change in time (%)	Efficiency
1	Baseline	Baseline	1
2	1.27	Decrease	2.27
3	4.63	Decrease	5.63
4	Decrease	Decrease	0
5	Decrease	0.31	0
6	4.89	77.42	0.06
7	Decrease	68.83	0
8	4.29	145	0.04
9	3.27	147.40	0.02

TABLE 4.12: Efficiency scores for various LandNet configurations on BigEarthNet.

Combination Number	Increase in F_1 score (%)	Increase in time (%)	Efficiency
1	Baseline	Baseline	1
2	Decrease	1.75	0
3	No change	0.84	0
4	Decrease	0.44	0
5	Decrease	0.67	0
6	1.3	97.73	0.01
7	No change	97.08	0
8	1.3	201	0.006
9	1.3	202.38	0.006

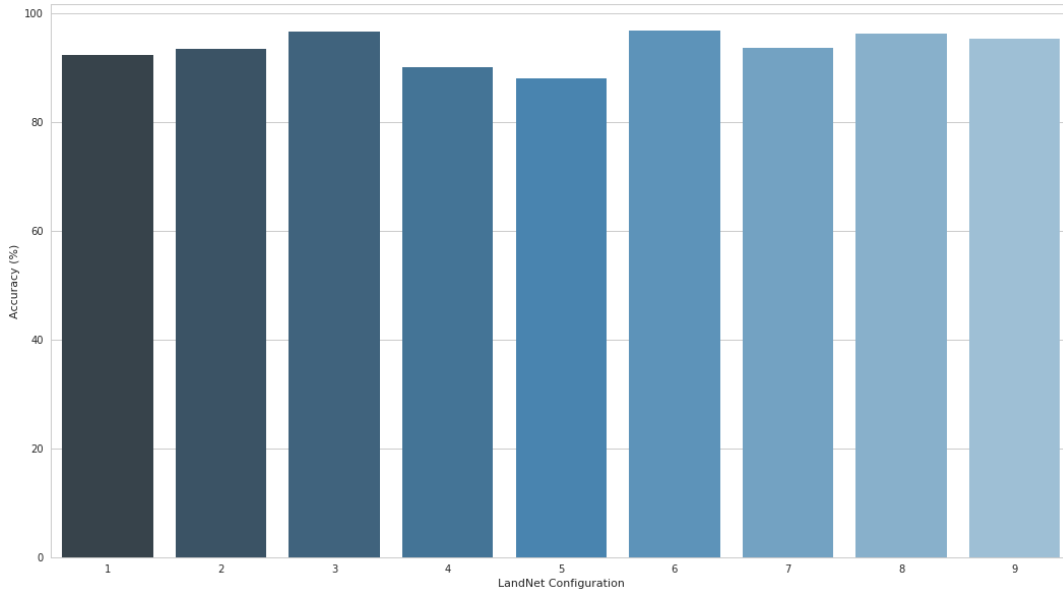


FIGURE 4.4: Bar graph showing accuracy of various LandNet configurations on EuroSAT.

results on EuroSAT. The following subsection focuses on discussing and analysing the results on BigEarthNet.

Results on EuroSAT

Figure 4.4 shows the information presented in Table 4.6 so that the results are easy to compare to each other. We can see that adding additional bands beyond just RGB always resulted in an improvement in the classification accuracy of the model. Using all the bands with one feature extractor improved the accuracy by only 1.2%. However, we see a larger increase of over 4% by using the special selection of bands aimed at detecting forests. Splitting the special bands and RGB bands into separate feature extractors gives slightly higher accuracy than using the bands in a single feature extractor. The highest accuracy we achieved was 96.87% using two feature extractors, one for the RGB bands and another for the remaining bands. Using three feature extractors; with one for RGB, one for the special bands and one for the remaining bands; gave an accuracy of 96.31%, which is the best result achieved with three. The best result is just under 2% lower than the outcome of 98.57% achieved by Helber *et al.* [2] in the debut paper for EuroSAT. Helber *et al.* [2] used a different training strategy than what we use, which may account for the difference in results. Another factor to note is that we trained the LandNets for only 9 epochs on EuroSAT. We did not perform any hyperparameter tuning so performing more epochs of training or adjusting the learning rate may still lead to better results.

To help further examine the performance of LandNet on EuroSAT we present Figure 4.5, Figure 4.6 and Figure 4.7 which are confusion graphs for configuration 1, 6 and 9 respectively. We provide the confusion graphs for these three configurations as all three configurations make use of all the bands which allows us to compare the accuracy differences for each class when using a different number of backbone feature extractors. Starting with Figure 4.5, straight away we see that LandNet 2 often misclassified annual crop as for other classes containing vegetation namely herbaceous vegetation, pasture and permanent crop. Interestingly, it also misclassified

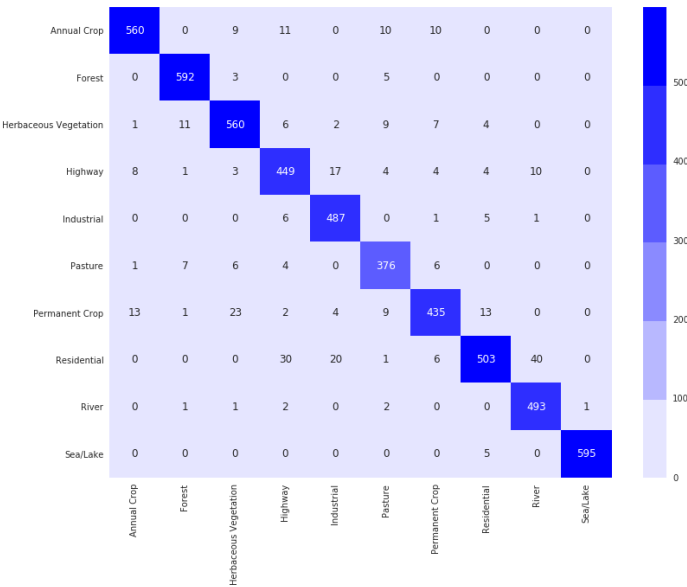


FIGURE 4.5: Confusion matrix for LandNet 2 on EuroSAT.

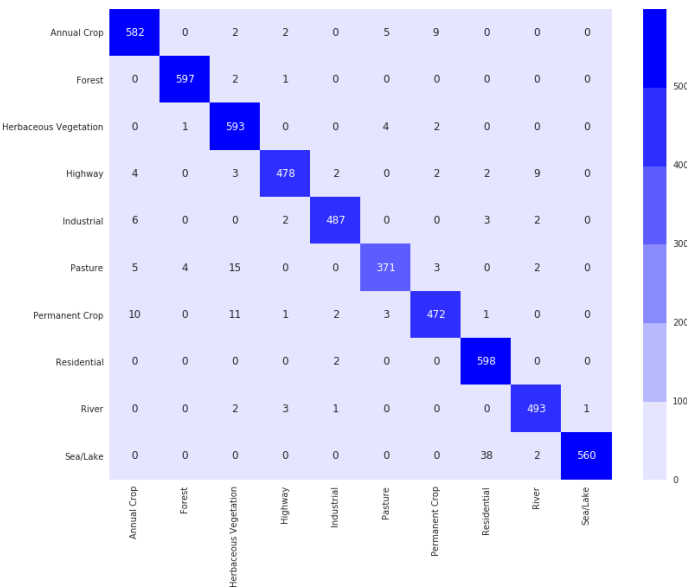


FIGURE 4.6: Confusion matrix for LandNet 6 on EuroSAT.

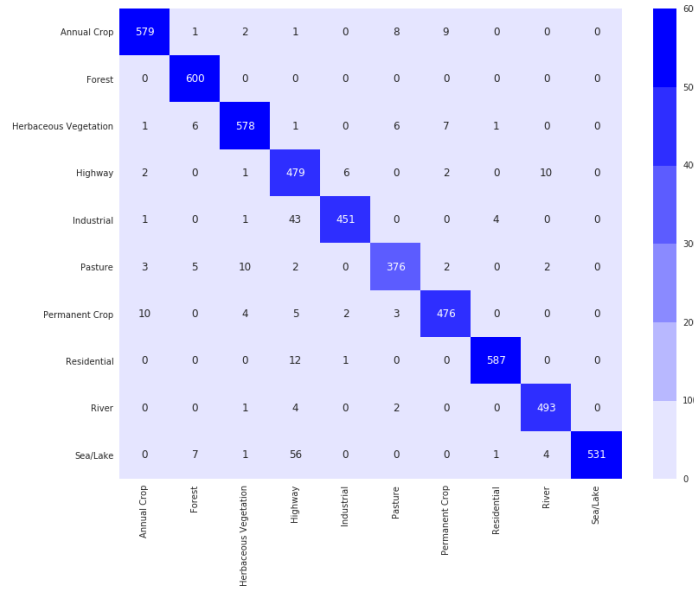


FIGURE 4.7: Confusion matrix for LandNet 9 on EuroSAT.

annual crop as highway 11 times. This is likely due to images of the highway class containing some of the area surrounding the highway including forests and farms. Comparing this to Figure 4.6 and Figure 4.7, we see that using multiple, separate feature extractors reduced the number of misclassifications of the annual crop class by around 50%. Looking at the other classes of vegetation, we see that these classes are often confused by LandNet 2. Of these classes, permanent crop fairs particularly poorly with 65 instances of the class being misclassified. Similarly with, annual crop, Figure 4.6 and Figure 4.7 show that configurations 6 and 9 of LandNet have less confusion amongst the vegetation classes.

Continuing our examination of Figure 4.5, we notice that residential is also often confused with highway and industrial. This seems reasonable as there is much similarity between these classes. More surprising is that residential was misclassified 40 times as river. It is possible that these misclassified images of residential areas contain some parts of a river in the images, thus leading LandNet 2 to misclassify them. We see from Figure 4.6 and Figure 4.7 that neither LandNet 6 nor LandNet 9 confuse residential for river at all, no doubt benefiting from the additional backbone feature extractors. It is not all negative for LandNet 2 though as we see from Figure 4.5 that it performed well on the forest, river and sea/lake classes, only misclassifying 8, 7 and 5 instances of those classes respectively. When peering into the sea/lake class specifically, we see that LandNet 1 performs far better than LandNet 6 and LandNet 9 with LandNet 6 and LandNet 9 misclassifying 40 and 69 instances of this class respectively. LandNet 6 misclassifies 38 instances of the sea/lake class as the residential class while LandNet 9 misclassifies 56 instances of the class as belonging to the highway class. The sea/lake is a notable case where the additional bands and feature extractors negatively affect the accuracy of a specific class. Shifting our focus to Figure 4.6 and Figure 4.7, we see that LandNet 6 and LandNet 9 display less confusion amongst the various classes.

Looking at the forest class in Figure 4.6 and Figure 4.7, we see that both configurations perform well on this class with LandNet 6 only misclassifying 3 instances of the class and LandNet 9 classifying all instances of the class correctly. An interesting difference between LandNet 6 and LandNet 9 when looking at the forest class is how

many instances of other classes are misclassified as belonging to the forest class. We see that LandNet 6 misclassified 5 instances of other classes as forest while LandNet 9 misclassifies 19 instances of other classes as forest. We see a significant difference in performance between LandNet 6 and LandNet 9 on the industrial class. On the industrial class, LandNet 6 correctly classifies 487 instances of the class where as LandNet 9 only manages to correctly identify 451 instances of the class. This is over a 7% difference in accuracy on this class. Another notable difference between the results on LandNet 6 and LandNet 9 is the tendency for LandNet 9 to misclassify many images as belonging to the highway class. In total, LandNet 9 misclassifies 124 instances of other classes as belonging to the highway class. Even more interesting is that LandNet 1 displays similar behaviour to LandNet 9 on the highway class, but not to the same extent. This behaviour is not exhibited by LandNet 6 on the highway class, but it does occur on the herbaceous vegetation class to a much lesser extent. Next, we shift our attention to the training and inference times of LandNet on EuroSAT.

Looking at Table 4.9, we can see that the training times for the configurations with a single feature extractor vary from 101.94 seconds to 110.28 seconds and inference times are within 1 second of each other. From this, there is no evidence that increasing or reducing the number of input channels in the image has any significant affect on the training or inference times of the models. This is to be expected as the vast majority of numerical operations are performed in the hidden layers of deep neural networks. The variation between the configurations with the same number of feature extractors can be attributed to scheduling and load differences of the machine at the times the models were trained. However, as we increase the number of feature extractors we see that both the training and inference times of the models increase. From Table 4.9, we see that adding a second feature extractor increased the training times by around 75 to 85 seconds. This represents a 68% to 77% increase in the training times. When looking at the inference times, we see that they increase from around 12 to 13 seconds to 23 seconds. Adding a third feature extractor further increases the training and inference times to between 269 and 271 seconds and 34 to 35 seconds respectively. Using three feature extractors increases the training times by 145% to 147%. We would expect that adding another feature extractor or two to LandNet which essentially doubles or triples the whole Convolutional Neural Network (CNN) would roughly double or triple both the training and inference times, but we see that this is not the case on EuroSAT. A possible explanation for this could be that since the images in EuroSAT are small, the GPU may not be fully utilised with only a single extractor CNN resulting in some additional headroom that is tapped into when training the larger configurations with multiple feature extractors. The below linear increase in training times benefits the efficiency scores of the models which we discuss next.

Delving into Table 4.11 which contains the efficiency scores of LandNet on EuroSAT, we notice immediately that the models with only a single feature extractor have the highest efficiency scores. LandNet 2 achieved an efficiency of 2.27 while LandNet 3 which saw a 4.63% increase in accuracy over the baseline of LandNet 1 managed to achieve an efficiency of 5.63. This is highest efficiency score we see from any LandNet model on EuroSAT and BigEarthNet. Neither LandNet 4 nor LandNet 5 produce a higher accuracy than the baseline of LandNet 1 so both see an efficiency of 0. Moving on to the configurations of LandNet that use two feature extractors, we notice that LandNet 7 manages an efficiency score of 0 like LandNet 4 and LandNet 5 while LandNet 6 produces an efficiency score of just 0.06. LandNet 6 achieves this low efficiency score despite the fact that it produced the highest accuracy of the

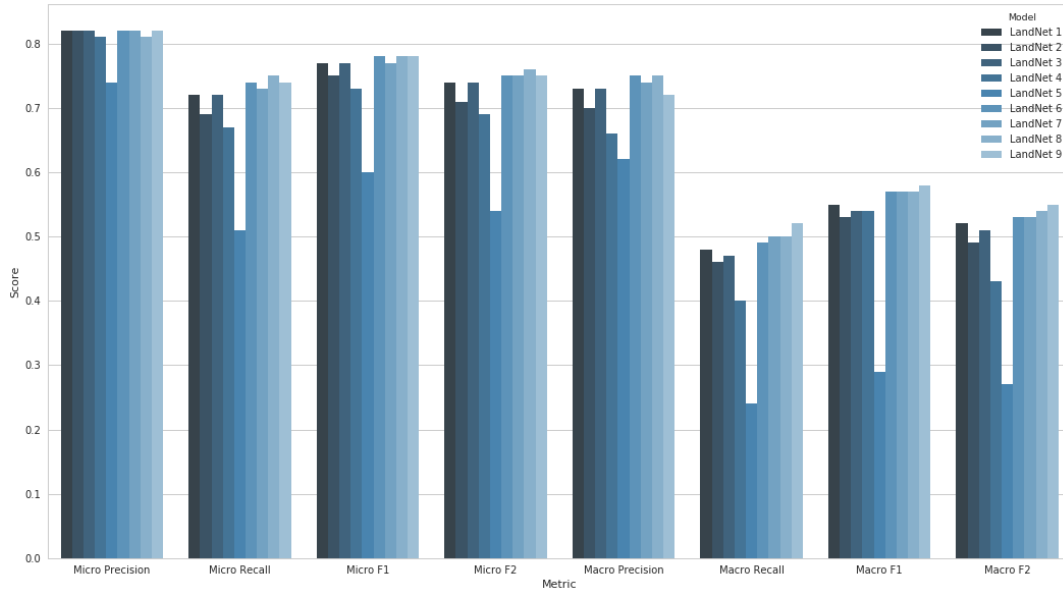


FIGURE 4.8: Bar graph showing micro and macro precision, recall, F_1 and F_2 scores of LandNet on BigEarthNet.

configurations and a 4.89% increase in accuracy over LandNet 1. Finally, looking at LandNet 8 and LandNet 9 which use three feature extractors, we see that they produce the lowest non-zero efficiency scores of all the configurations achieving scores of 0.04 and 0.02 respectively. From the efficiency scores we can conclude that if the training and inference times of the models are a consideration as well as the accuracy, then the best trade-off between speed and accuracy would be to make use of one of the LandNet configurations that only use a single feature extractor, namely LandNet 3 in this case. We cannot make the same conclusion when considering LandNet on BigEarthNet. We discuss the results LandNet on BigEarthNet in the next subsection.

Results on BigEarthNet

In this subsection we discuss and analyse the results of the various LandNet configurations on BigEarthNet. We begin by examining the micro and macro precision, recall, F_1 and F_2 scores of the various configurations. To allow for easier comparisons between the configurations, we provide Figure 4.8 which visually represents these metrics in a bar graph. Looking at the micro precision we see that all the various configurations perform similarly with six of the configurations scoring 0.82, the highest result we achieved, another two scoring 0.81 and one configuration, LandNet 5, performing notably worse and only scoring 0.74. We see more distinction between the various configurations of Landnet when considering the micro recall.

We see that there is more variance in the micro recall results with the best performing configuration, LandNet 8, achieving a micro recall score of 0.75. LandNet 6 and LandNet 9 are just behind LandNet 8 managing a micro recall score of 0.74. The highest result achieved by a model with only a single feature extractor was 0.72, achieved by both LandNet 1 and LandNet 3. LandNet 5 saw the the lowest micro recall score, managing a core of just 0.51. Even though many of the configurations with differing numbers of feature extractors achieved the same or similar micro precision scores, we see that the configurations with additional feature extractors achieve higher micro recall than those with only a single feature extractor. We now turn our

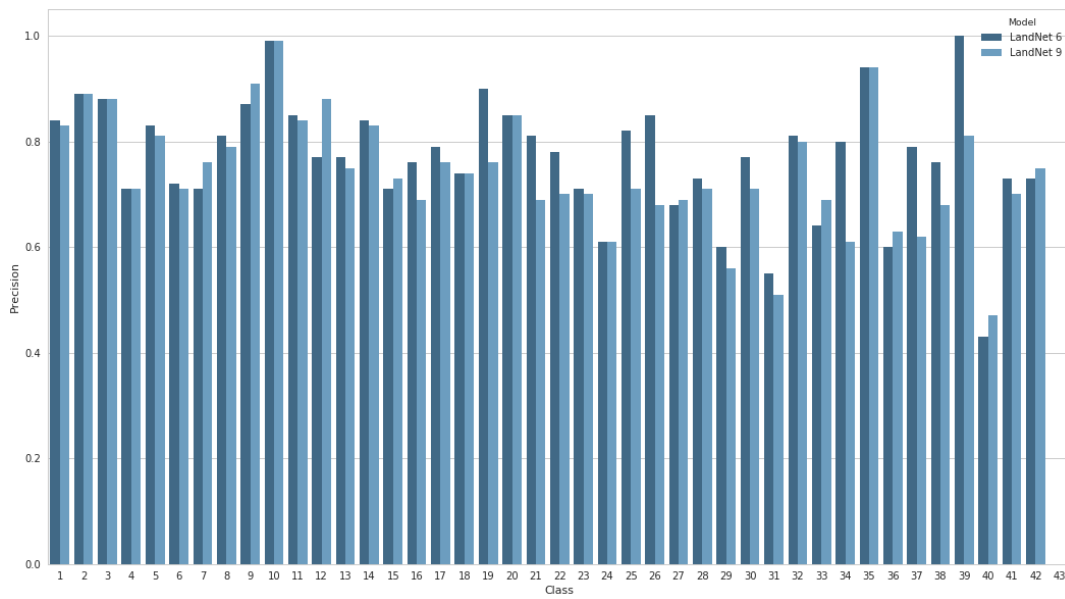


FIGURE 4.9: Bar graph showing precision per class of LandNet 6 and LandNet 9 on BigEarthNet.

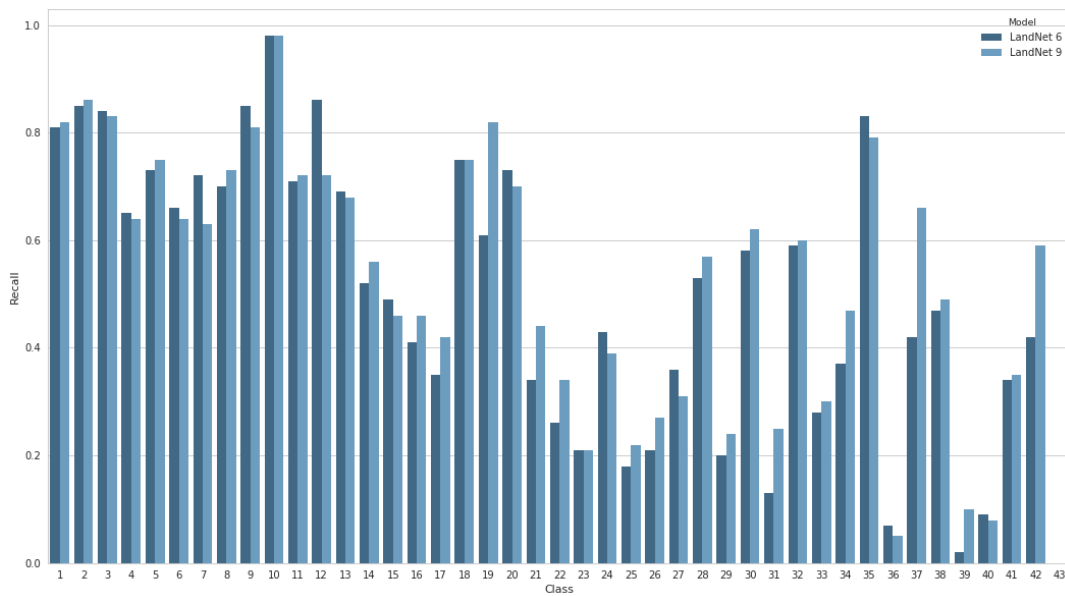


FIGURE 4.10: Bar graph showing recall per class of LandNet 6 and LandNet 9 on BigEarthNet.

attention to the micro F_1 and F_2 scores.

Like with the micro precision, the differences in the micro F_1 and F_2 scores between the various configurations of LandNet are small, with the exception of LandNet 5 which performs considerably worse than the other configurations, achieving a micro F_1 score of 0.60 and a micro F_2 score of 0.54. The highest micro F_1 score we see is 0.78 which is achieved by LandNet 6, LandNet 8 and LandNet 9. A number of configurations, namely LandNet 1, LandNet 3 and LandNet 7 produce a micro F_1 score of 0.77. Like with recall, we see that the highest micro F_1 score of 0.78 is achieved by configurations with more than one feature extractor. However, the observed difference between the best micro F_1 score from configurations with a single feature extractor and the best micro F_1 from those with multiple feature extractors is smaller than difference in the micro recall scores. Looking more at the micro F_2 , we see that LandNet 8 produces the best result of 0.76, which is not matched by any other configuration of LandNet. The next highest micro F_2 score of 0.75 is achieved by LandNet 6, LandNet 7 and LandNet 9. The highest micro F_2 score achieved by a configuration with just a single feature extractor is 0.74, achieved by LandNet 1 and LandNet 3. As previously noted, the configurations with multiple feature extractors performed better than models with a single feature extractor when considering micro recall. The F_2 score assigns more weight to recall than precision which explains why we see a greater difference between configurations with a single feature extractor and those with multiple feature extractors compared to the F_1 score.

We demonstrate higher micro precision, micro F_1 and micro F_2 scores on BigEarthNet than the results presented in Sumbul *et al.* [16] or any other papers to the authors' knowledge. The highest scores presented by Sumbul *et al.* [16] are achieved by a model trained on all the bands in BigEarthNet. Sumbul *et al.* [16] achieved a micro precision of 0.6993, a micro recall score of 0.7710, a micro F_1 score 0.7098 and a micro F_2 score of 0.7384. A number of LandNet configurations achieved a significantly higher micro precision score of 0.82. LandNet 6, LandNet 8 and LandNet 9 show a noticeably higher micro F_1 score of 0.78 and LandNet 8 achieves a better micro F_2 of 0.76. However, Sumbul *et al.* [16] are able to achieve a higher recall score of 0.7710 compared to the 0.75 achieved by LandNet 8. An interesting difference to note between the training method employed on LandNet compared to the training method use in Sumbul *et al.* [16] is the number of training epochs performed. We perform five epochs over the dataset when training LandNet on BigEarthNet while Sumbul *et al.* [16] trained their model for 100 epochs on BigEarthNet. The fact that so many epochs of training was performed by Sumbul *et al.* [16] indicates that performing more epochs of training for LandNet on BigEarthNet may lead to improved results. Turning our attention to the macro precision, recall, F_1 and F_2 scores, we immediately see that the macro scores are all notably lower than their micro counterparts. We also see more differences in the macro score between the various configurations of LandNet compared to the micro scores. When looking at the macro precision, we see that LandNet 6 and LandNet 8 produce the best results attaining a macro precision of 0.75. No configuration with a single feature extractor is able to match this, only achieving a macro precision of 0.72.

Moving to the macro recall, we see a larger gap between the macro and micro recall than the gap between the macro and micro precision. Like the micro recall, the configurations with multiple feature extractors are unmatched by the configurations with a single feature extractor. LandNet 9 was the best performing configuration achieving a macro recall of 0.52 and LandNet 1 was the best performing model with a single feature extractor producing a macro recall of 0.48. LandNet 5 fares particularly poorly, only managing a macro recall of 0.24.

Moving on to the macro F_1 and F_2 scores, we see that there is a large drop from the micro scores to the macro scores which is to be expected considering the macro precision and recall results. LandNet 9 produces the highest F_1 and F_2 scores of 0.58 and 0.55 respectively. We again see the benefit of the additional feature extractors as the worst configuration with multiple feature extractors achieves a higher macro F_1 and F_2 score than the best with configuration with a single feature extractor. LandNet 6 and LandNet 7 achieve the same macro F_1 and F_2 scores of 0.57 and 0.53, which is the lowest result from configurations with multiple feature extractors compared to the 0.55 and 0.52 achieved by LandNet 1 which is the highest results from a model with a single feature extractor. This shows that while adding additional bands leads to better results, the structure of the neural networks and model design also play a role in the chase for ever more accurate deep learning models on multispectral images. As discussed above, the macro scores are significantly lower overall than the micro scores. This is a result of how the micro and macro scores are calculated and the nature of the BigEarthNet dataset. The micro scores are calculated across the entirety of the testing dataset while the macro scores are calculated by first calculating the scores within each class and then taking the average across all the classes. Therefore, the macro scores are affected by outliers amongst the classes. The outlying classes tend to be those that appear far less frequently in BigEarthNet. This class imbalance causes the results on some classes to be poor, as we will see when we look at the results per class, and thus cause the macro scores to be dragged down.

Figure 4.9 and Figure 4.10 present the precision and recall scores per class on LandNet 6 and LandNet 9 respectively in the form a bar graph to allow for easier comparisons between the classes and models. We chose to look more closely at these two configurations as they they presented similar micro precision and micro recall scores. As they produce similar micro scores there many classes where they perform similarly such as class 1, which is mixed forest, and class 10, sea and ocean. However, there are notable differences between the configurations when examining the results per class. We see that LandNet 6 achieves a precision score of 0.77 on class 12, agro-forestry areas, while LandNet 9 manages to a precision score of 0.88, however LandNet 6 achieves a higher recall of 0.86 compared to the recall score of 0.72 produced by LandNet 9. The higher recall score allows LandNet 6 to produce a F_1 score of 0.81 which is higher than the 0.79 achieved by LandNet 9. Class 39, airports, is an example where the F_1 scores produced by LandNet 6 and LandNet 9 differ greatly. LandNet 6 achieves a F_1 score of 0.05 while LandNet 9 achieves 0.18. We also see that LandNet 6 achieves a precision score of 1.0 on class 39, however it manages an extremely low recall score of just 0.02 which is why the F_1 score is so low as well and highlights how favouring precision can result in a low recall score.

Looking at Figure 4.9 and Figure 4.10 it is evident that there is a downward trend in the results from class 1 to 43. The classes are ordered by the number of samples in the entire dataset in descending order. The decline in recall as the number of samples in the class decreases is particularly noteworthy. This highlights an area where LandNet can be improved in the future, namely, greater effort can be made to reduce the negative effects of class imbalance within the dataset. The recall score may also be further improved by increasing the number of training epochs performed. We perform five epochs of training of LandNet on BigEarthNet. This is a far fewer training epochs compared to the 100 training epochs performed by Sumbul *et al.* [16] for their models. Training LandNet for 100 epochs would take a considerable amount of time, as well will discuss next.

We can see from Table 4.10 that the configurations of LandNet take approximately

1530 seconds to perform a single training epoch. The times increases linearly with regards to the number of backbone feature extractor networks as can be seen from the time taken to perform an epoch of training for LandNet 6 and LandNet 7 with two feature extractors and LandNet 9 and LandNet 9 with three feature extractors. LandNet 6 and LandNet 7 take approximately 3000 seconds to perform a single epoch. LandNet 8 and LandNet 9 take an additional 1600 seconds more than LandNet 6 and LandNet 7 for a total of 4600 seconds to perform a single training epoch. Training LandNet 8 and LandNet 9 for 100 epochs would take 460000 seconds or just under 128 hours. Next we will look at how the small increases in the F_1 score and the linear increase in training time affect the efficiency score.

We conclude our analysis and discussion of our results by examining the efficiency of LandNet on BigEarthNet. From Table 4.12, it is evident that no configuration with a single feature extractor achieves a non-zero efficiency score. This is due to the configurations either making no improvement over the baseline configuration of LandNet 1 or performing worse than LandNet 1. This differs to what we see with LandNet on EuroSAT where the configurations with a single feature extractor perform the best in terms of efficiency. As we do not perform any sort of hyperparameter tuning, it is possible that further training epochs may lead to greater differentiation between the various configurations. Moving on to the configurations with two feature extractors, we see that LandNet 6 achieved an efficiency score of 0.01, which is the best result we see for LandNet trained on BigEarthNet. LandNet 7 does not improve the F_1 score over the baseline so it receives an efficiency score of zero. Finally, we see that both LandNet 8 and LandNet 9 produce efficiency scores of 0.006. While they both see the same percentage increase in the F_1 as LandNet 6, they see a lower efficiency score due to the increased training time over LandNet 6. In the next section we discuss limitations of the results of our research.

4.5 Limitations

We did not perform any hyperparameter tuning for any of our testing. Thus it may possible to achieve better results than what we present by optimising the training process for each of the models, specifically the number of training epochs and the learning rate used. Another limitation specific to Mask R-CNN and the models based on it is that we do not test on any datasets that have masks for each object which prevents us from using the mask subnet potentially reducing the performance of Mask R-CNN and the models based on it. We conclude the chapter in the next section.

4.6 Conclusion

Our results on object recognition using RGB-D images did not show the improvements we had expected. Rather when using RGB-D images we were only able to match the performance of models using RGB images. However, the improvements to the initially poor performance of the RGB-D models made by improving the initialization of model may indicate that the lack of large RGB-D training dataset is the biggest hindrance to demonstrating better object recognition performance using RGB-D images.

The results from our experiments in the use of multispectral images for land cover classification are far more conclusive. We demonstrate that using the additional bands from images taken by the Sentinel-2 mission with the RGB bands improves

the land cover classification results. We also show that how the additional bands are used in the model makes impacts the results and should that model design should not be overlooked when dealing with multispectral images. We also produce state-of-the-art results on the BigEarthNet dataset achieving leading precision and F_1 scores.

Chapter 5

Conclusion

In this chapter we present a brief summary of the research followed by a summary of the results. Next, we discuss future work that can expand on the research that we have done. Finally, we conclude the chapter by discussing the contributions of our research.

5.1 Summary of Research

Object recognition has many applications from robotics to autonomous vehicles [12]. Many robots and autonomous vehicles now incorporate sensors capable of sensing depth [12]. Depth information has the potential to aid in producing more accurate object recognition models [5]. This research aimed to build an object recognition model that makes use of depth information to improve the accuracy of the model. We built our models by modifying Mask R-CNN [14] and RetinaNet [13], both of which are state-of-the-art models, to use RGB-D images. We did this by adding in depth as a fourth channel. We also tried using separate networks for RGB and depth and combine them in the classification stage as is done in Socher *et al.* (2012) [5]. We trained our models on the SUN RGB-D dataset [18]. We also tested our models on our own dataset of scene images that we captured using a Zed stereo camera. This helped us test the generalization of our models. We measured the accuracy of our models using mAP and we compared them to the models trained on RGB images. We also measured the time taken to train our models and perform inference and compared those to the models trained on RGB images. However, we did not aim to reduce the training and inference times for our models. We also measured the efficiency of the models. The efficiency is a measure of how effectively the model is able to translate the additional training time into improving the accuracy.

We also tackled the task of land cover classification. Land cover classification is a remote sensing problem where the aim is to determine the land -cover or -use from aerial or satellite imagery of a patch of land. While much research has been done on land cover classification using aerial or satellite RGB images, the increase in availability of multispectral satellite images has made it possible for novel multispectral land cover classification datasets to exist. Using these datasets presented an opportunity to study the benefits of multispectral images for land cover classification which forms another aspect of our research. To test the benefits of multispectral images we designed a novel model dubbed LandNet. We took inspiration from Socher *et al.* (2012) [5] when we designed LandNet, allowing us to not only vary which bands we used in the features extractors of LandNet, but also the number of backbone feature extractors in LandNet. We tested the various configurations of LandNet on both the EuroSAT dataset as well as the newer and larger BigEarthNet dataset. We measure the accuracy of the models on BigEarthNet using both the micro and macro precision, recall, F_1 and F_2 scores. Like with our tests with RGB-D images, we

also measured the training and inference times of LandNet as well as the efficiency of the models [2], [16]. The next section summarises the results of our research.

5.2 Summary of Results

The results from object recognition with RGB-D images were not what we had expected. Initially, the models trained on RGB-D images were not able to match the performance of the RGB models. Using the improved weight initialization method the kept the pretrained weights for the RGB channels of the initial convolutional layer improved the accuracy of the RGB-D models. However, the best performing RGB-D model was only able to match the performance of the best RGB model. This differs to the results shown in Socher *et al.* (2012) [5] where they were able to demonstrate better performance using RGB-D images. The results of the training and inference times is as we expected, with the RGB-D models with a separate feature extractor roughly doubling the training time compared to the RGB models and the RGB-D models what use the depth channel as an additional input channel in the first convolutional layer.

Our results on land cover classification using the EuroSAT and BigEarthNet datasets showed that using the additional bands of the multispectral images with the RGB bands improves the accuracy of LandNet. The results of LandNet on EuroSAT were good, but were not state-of-the-art, but LandNet was able to produce state-of-the-art results on BigEarthNet, achieving higher precision and F_1 scores than those shown by Sumbul *et al.* [16]. Our results from LandNet also show that it is pertinent to consider network structure when using multispectral images as the best results on both EuroSAT and BigEarthNet we achieved by configurations of LandNet with multiple feature extractors. The training and inference times of LandNet were inline with our expectations with a near linear relationship between the number of feature extractors and the training times while adding additional bands as additional channels in the initial convolutional layer having little to no effect on the training and inference times [2]. In the next section we discuss opportunities to build on the work we have done in future research.

5.3 Future Work

There is space to test object recognition with RGB-D images using other models as a starting point. Additionally, further work can be done in more novel model structures designed specifically for the task of object recognition on RGB-D images. There is also a gap in the availability of large object recognition datasets containing RGB-D images including labelled object masks in addition to bounding boxes. SUN RGB-D contains 5285 testing images and 5050 testing images which is significantly less than similar RGB object recognition datasets such as the COCO dataset and ImageNet [18], [28]. Deep learning models require significant amounts of data to train effectively. The number of images contained in SUN RGB-D may not be enough for effective training of the RGB-D models due to the added complexity of the relative depth channel. The construction of a large object recognition dataset containing RGB-D images and object masks can be the focus of future work.

There is opportunity for experimenting with training methodologies and loss functions that are designed to reduce the negative effects of class imbalance on the accuracy of deep learning models. Many datasets, including object recognition datasets and land cover classification datasets, suffer from the presence of class imbalance.

Taking special care to handle the class imbalance within BigEarthNet should lead to improved results. Various methods to handle class imbalance and the results of these methods can be examined in future work.

We did not perform hyperparameter tuning for any of models. Optimizing the learning rate and the number of training epochs can lead to improved results and can be explored in future work. This is especially pertinent to LandNet when trained on BigEarthNet as the models presented in Sumbul *et al.* [16] were trained for 100 epochs compared to the 5 epochs of training we performed for LandNet. We conclude the chapter in the next section by discussing the contributions of our work to the broad field of object recognition.

5.4 Contributions

Our research shows the potential of the use of RGB-D images for object recognition. It also highlights the need for a large object recognition dataset containing RGB-D images matching the number of images in datasets such as the COCO dataset or even ImageNet. We also contribute a small RGB-D dataset that can be used in future research or incorporated into a larger dataset. Our research also demonstrates that adding depth as an additional channel in the first convolutional layer of a deep learning model results in little to no increase in training and inference times of the model. This fact makes using RGB-D images more appealing as they have the potential to increase accuracy without increasing the training and inference times, thus allowing more accurate models to be used for inference in user applications without sacrificing the responsiveness or requiring increased computational capability.

Additionally, our research shows that the use of multispectral images can improve the accuracy of land cover classification models. We also demonstrate that the model structure can have an affect on the performance of models using multispectral images. We achieve state-of-the-art results on BigEarthNet using our novel model LandNet with multispectral images and multiple backbone feature extractors, demonstrating the importance of model structure and the benefits of multispectral images. Thus, we can conclude that this research has made a significant contribution to the field of object recognition.

Bibliography

- [1] D. G. Lowe, "Object recognition from local scale-invariant features", in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, Ieee, vol. 2, 1999, pp. 1150–1157.
- [2] P. Helber, B. Bischke, A. Dengel, and D. Borth, "Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification", *CoRR*, vol. abs/1709.00029, 2017. arXiv: 1709.00029. [Online]. Available: <http://arxiv.org/abs/1709.00029>.
- [3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [4] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng, "Parsing natural scenes and natural language with recursive neural networks", in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 129–136.
- [5] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3d object classification", in *Advances in Neural Information Processing Systems*, 2012, pp. 656–664.
- [6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
- [7] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes", *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 5, pp. 433–449, 1999.
- [8] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts", *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 4, pp. 509–522, 2002.
- [9] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition", *International journal of computer vision*, vol. 61, no. 1, pp. 55–79, 2005.
- [10] M. Lades, J. C. Vorbruggen, J. Buhmann, J. Lange, C. Von Der Malsburg, R. P. Wurtz, and W. Konen, "Distortion invariant object recognition in the dynamic link architecture", *IEEE Transactions on computers*, vol. 42, no. 3, pp. 300–311, 1993.
- [11] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context", *International Journal of Computer Vision*, vol. 81, no. 1, pp. 2–23, 2009.
- [12] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5648–5656.
- [13] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection", *CoRR*, vol. abs/1708.02002, 2017. arXiv: 1708.02002. [Online]. Available: <http://arxiv.org/abs/1708.02002>.

- [14] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN", *CoRR*, vol. abs/1703.06870, 2017. arXiv: 1703.06870. [Online]. Available: <http://arxiv.org/abs/1703.06870>.
- [15] M. Castelluccio, G. Poggi, C. Sansone, and L. Verdoliva, "Land use classification in remote sensing images by convolutional neural networks", *CoRR*, vol. abs/1508.00092, 2015. arXiv: 1508.00092. [Online]. Available: <http://arxiv.org/abs/1508.00092>.
- [16] G. Sumbul, M. Charfuelan, B. Demir, and V. Markl, "Bigearthnet: A large-scale benchmark archive for remote sensing image understanding", *CoRR*, vol. abs/1902.06148, 2019. arXiv: 1902.06148. [Online]. Available: <http://arxiv.org/abs/1902.06148>.
- [17] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition", in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 922–928.
- [18] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite.", in *CVPR*, vol. 5, 2015, p. 6.
- [19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", in *Advances in neural information processing systems*, 2015, pp. 91–99.
- [20] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition", in *International conference on machine learning*, 2014, pp. 647–655.
- [21] R. Salakhutdinov and H. Larochelle, "Efficient learning of deep boltzmann machines", in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 693–700.
- [22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *nature*, vol. 521, no. 7553, p. 436, 2015.
- [23] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition", *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.
- [24] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding", *arXiv preprint arXiv:1408.5093*, 2014.
- [25] R. Girshick, "Fast r-cnn", *arXiv preprint arXiv:1504.08083*, 2015.
- [26] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks", in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, IEEE, 2017, pp. 5987–5995.
- [27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [28] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context", *CoRR*, vol. abs/1405.0312, 2014. arXiv: 1405.0312. [Online]. Available: <http://arxiv.org/abs/1405.0312>.
- [29] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset", in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 1817–1824.

- [30] S. Basu, S. Ganguly, S. Mukhopadhyay, R. DiBiano, M. Karki, and R. R. Neman, "Deepsat - A learning framework for satellite imagery", *CoRR*, vol. abs/1509.03602, 2015. arXiv: 1509.03602. [Online]. Available: <http://arxiv.org/abs/1509.03602>.
- [31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", *CoRR*, vol. abs/1409.4842, 2014. arXiv: 1409.4842. [Online]. Available: <http://arxiv.org/abs/1409.4842>.
- [32] Tzutalin, *Labelimg*, <https://github.com/tzutalin/labelImg>, Accessed: June 2019, 2015.
- [33] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge", *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.
- [34] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgb-d images", in *European Conference on Computer Vision*, Springer, 2012, pp. 746–760.
- [35] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell, "A category-level 3d object dataset: Putting the kinect to work", in *Consumer Depth Cameras for Computer Vision*, Springer, 2013, pp. 141–165.
- [36] J. Xiao, A. Owens, and A. Torralba, "Sun3d: A database of big spaces reconstructed using sfm and object labels", in *Computer Vision (ICCV), 2013 IEEE International Conference on*, IEEE, 2013, pp. 1625–1632.
- [37] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification", *CoRR*, vol. abs/1502.01852, 2015. arXiv: 1502.01852. [Online]. Available: <http://arxiv.org/abs/1502.01852>.
- [38] J. Yang, J. Lu, D. Batra, and D. Parikh, *A faster pytorch implementation of faster r-cnn*, <https://github.com/jwyang/faster-rcnn.pytorch>, Accessed: June 2019, 2017.
- [39] R. Tseng, *A pytorch implementation of detectron*, <https://github.com/roytseng-tw/Detectron.pytorch>, Accessed: June 2019, 2018.
- [40] Y. Henon, *Pytorch retinanet*, <https://github.com/yhenon/pytorch-retinanet>, Accessed: June 2019, 2018.
- [41] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation", vol. 3408, Apr. 2005, pp. 345–359. DOI: 10.1007/978-3-540-31865-1_25.