

# **Deep Learning based Semantic Segmentation for Unstructured Outdoor Environments**



**Nkosinathi Ndlovu**  
**417596**

**Supervised by: Turgay Celik, PhD: Wits University**  
**Quentin Williams, PhD: Council for Scientific and Industrial Research**

University of the Witwatersrand

This dissertation is submitted to the Faculty of Science in fulfilment of the  
requirements for the degree

*Master of Science (Computer Science)*

June 2020



## Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text.

Signature:  Date: 30/06/2020

Nkosinathi Ndlovu  
417596  
June 2020





## **Acknowledgements**

This has been by far the most challenging experience of my life, both academically and all other personal aspects of my life. Giving up seemed to be the best option at some stage. This experience, however, turned out to be the greatest learning experience of my life, both academically and all other aspects of my professional life. I am really grateful for all the struggles I went through in this journey.

I would like to thank the CSIR for the opportunity to conduct research on a real-world project. I would also like to thank my friends and family for the support and encouragement they afforded me when things got tough. Especially my sisters Lindi, Thuli, Nothi, Noma and Linda. Hopefully this will help them understand why I missed some of the important family gatherings.

Above all, I would like to extend my utmost gratitude to my supervisor Prof. Turgay Celik for his support and guidance throughout this journey. None of this would have been possible without his patience and understanding. I dedicate this work to him, and myself.



## **Abstract**

The past decade has seen increased interest in, and demand for autonomous vehicles. Complete and successful autonomy of mobile systems such as unmanned ground vehicles (UGVs) depends on perception. Perception is the ability of an agent to semantically interpret its operational environment through vision. Deep learning approaches have recently been continuously more successful over traditional/classical methods in perception and vision tasks. This is primarily due to their non-reliance on selected hand-crafted features, they adopt a more robust and generalised learned-feature approach through representation learning. Convolutional Neural Networks (CNNs) have been widely used towards the goal of scene parsing and perception. In this research we focus on using CNN architectures for semantic segmentation in unstructured outdoor environments for autonomous navigation. Our first contribution is to provide a novel dataset for unstructured outdoor domains: the CSIR dataset. We seek to establish whether it is possible to semantically segment an unstructured scene into pre-defined classes such as grass, road, sky, trees etc. This is achieved through an exhaustive comparative study on state-of-the-art CNN architectures on this dataset, and a similar additional dataset: the Freiburg Forest dataset. Furthermore, we seek to establish whether there are any benefits in using transfer learning and pre-trained weights in training CNN architectures for semantic segmentation with limited datasets. Lastly, we identify the important architectural factors necessary for successful semantic segmentation in unstructured outdoor scenes.



# Contents

|                                                                     |             |
|---------------------------------------------------------------------|-------------|
| <b>List of Figures</b>                                              | <b>xiii</b> |
| <b>List of Tables</b>                                               | <b>xvii</b> |
| <b>List of Abbreviations</b>                                        | <b>xx</b>   |
| <b>1 Introduction</b>                                               | <b>1</b>    |
| 1.1 Background . . . . .                                            | 1           |
| 1.2 Motivation . . . . .                                            | 3           |
| 1.3 Aim and scope of this research . . . . .                        | 5           |
| 1.3.1 Delimitations and assumptions of the research . . . . .       | 5           |
| 1.4 Dissertation layout . . . . .                                   | 6           |
| <b>2 Background and related work</b>                                | <b>7</b>    |
| 2.1 Introduction . . . . .                                          | 7           |
| 2.2 Traditional semantic segmentation approaches . . . . .          | 7           |
| 2.3 Bottom-up approaches . . . . .                                  | 9           |
| 2.3.1 Discrete approaches: . . . . .                                | 9           |
| 2.3.2 Continuous approaches . . . . .                               | 11          |
| 2.3.3 Superpixels . . . . .                                         | 12          |
| 2.3.4 Interactive methods . . . . .                                 | 12          |
| 2.4 From traditional methods to deep learning approaches . . . . .  | 13          |
| 2.5 Introduction to artificial neural networks . . . . .            | 14          |
| 2.5.1 Biological neuron model . . . . .                             | 14          |
| 2.5.2 Universal approximation theorem (Cybenko's theorem) . . . . . | 16          |
| 2.5.3 Artificial neural networks . . . . .                          | 17          |
| 2.5.4 Feature-based neural networks . . . . .                       | 23          |
| 2.5.5 Convolutional neural networks: Building blocks . . . . .      | 27          |
| 2.5.6 Loss function . . . . .                                       | 33          |

|          |                                                                                  |           |
|----------|----------------------------------------------------------------------------------|-----------|
| 2.5.7    | Back-propagation . . . . .                                                       | 36        |
| 2.5.8    | Optimisation . . . . .                                                           | 37        |
| 2.5.9    | Learning rate scheduling and gradient descent optimisation . . . . .             | 39        |
| 2.5.10   | Hyperparameters . . . . .                                                        | 42        |
| 2.5.11   | Network initialisation and transfer learning . . . . .                           | 44        |
| 2.6      | Summary . . . . .                                                                | 45        |
| <b>3</b> | <b>Literature Review</b>                                                         | <b>47</b> |
| 3.1      | Introduction . . . . .                                                           | 47        |
| 3.2      | CNN architectures . . . . .                                                      | 47        |
| 3.2.1    | State-Of-The-Art architectures for semantic segmentation . . . . .               | 52        |
| 3.3      | Datasets . . . . .                                                               | 61        |
| 3.4      | Summary . . . . .                                                                | 63        |
| <b>4</b> | <b>Research methodology</b>                                                      | <b>65</b> |
| 4.1      | Introduction . . . . .                                                           | 65        |
| 4.2      | Research objectives . . . . .                                                    | 65        |
| 4.3      | Research approach . . . . .                                                      | 66        |
| 4.3.1    | Data acquisition and pre-processing . . . . .                                    | 66        |
| 4.3.2    | Research pipeline . . . . .                                                      | 69        |
| 4.4      | Hardware and Software . . . . .                                                  | 71        |
| 4.5      | System details . . . . .                                                         | 71        |
| 4.5.1    | Hardware . . . . .                                                               | 71        |
| 4.5.2    | Software . . . . .                                                               | 72        |
| 4.6      | Summary . . . . .                                                                | 72        |
| <b>5</b> | <b>Results and discussion</b>                                                    | <b>75</b> |
| 5.1      | Introduction . . . . .                                                           | 75        |
| 5.2      | Qualitative results and analysis . . . . .                                       | 76        |
| 5.3      | Qualitative Analysis and Discussion - extended results . . . . .                 | 81        |
| 5.4      | Quantitative results - All models on Freiburg Forest and CSIR datasets . . . . . | 86        |
| 5.4.1    | Convergence . . . . .                                                            | 89        |
| 5.4.2    | Convergence - extended results . . . . .                                         | 93        |
| 5.5      | Summary . . . . .                                                                | 97        |
| <b>6</b> | <b>Conclusion</b>                                                                | <b>99</b> |
| 6.1      | Conclusion . . . . .                                                             | 99        |

|                               |     |
|-------------------------------|-----|
| 6.2 Future research . . . . . | 101 |
|-------------------------------|-----|

|                     |            |
|---------------------|------------|
| <b>Bibliography</b> | <b>103</b> |
|---------------------|------------|





# List of Figures

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |    |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1 | Images of Unstructured Outdoor Scenes (CSIR). Non-urban environments are typically characterised by homogeneous features and low contrast. . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 3  |
| 1.2 | An example on the input-groundtruth pairs. Given an input image (a), semantic segmentation assigns every pixel in the image to pre-defined classes such as grass, trees, road and sky. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 4  |
| 2.1 | Shows the progression of computer vision approaches from classical methods to modern learning approaches. From top to bottom, the <i>first row</i> shows an era when problem solving using computers began, everything was designed by the human, hand-designed programs mapped input to output ( <b>rule-based approaches</b> [99]); the <i>second row</i> is where entire section above falls under, elements of learning where introduced but the features where still crafted by the human ( <b>classical machine learning</b> ), the <i>third</i> and <i>fourth rows</i> show the era of <b>representation learning</b> [51, 101], most state-of-the-art approaches fall under this category. . . . . | 14 |
| 2.2 | The <b>biological neuron model</b> is the main inspiration behind ANNs. In the biological model the <b>dendrites</b> receive signals as <b>input</b> , that signal is then processed as <b>neural impulses</b> into the <b>cell body</b> and flows through the <b>myelin sheath</b> to produce <b>output</b> signals at the <b>axon terminals</b> . [1, 62] . .                                                                                                                                                                                                                                                                                                                                            | 15 |
| 2.3 | Examples of activation functions, Sigmoid (red), Rectified Linear Units (green) and the hyperbolic tangent (blue). . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 16 |
| 2.4 | An example of a 4-Layer Artificial Neural Network (multi-layer perceptron). This network takes in $x_1$ , $x_2$ and $x_3$ and feeds it forward through the hidden layers, $h_{(i,j)}$ , it also takes in a <i>bias</i> (yellow) signal at every processing layer. The output is three different classes $y_1$ , $y_2$ , and $y_3$ . . . . .                                                                                                                                                                                                                                                                                                                                                                | 18 |
| 2.5 | Categories of classification models [120]. Starting from basic logistic regression models and the various ways to address their major limitations gives rise to various categories of regression approaches. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 24 |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |    |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.6 | An example of the convolution operation with an input image ( <b>I</b> ) of dimension $[h,w,d] = [7,7,1]$ and a kernel ( <b>K</b> ) of dimension $[h,w,d] = [3,3,1]$ . The depth of the input image and the kernel in this example is 1 (one colour channel), and the convolution produces output <b>G</b> . The weights in the kernel are the parameters that has to be learned or trained. Furthermore, in this example $stride = 1$ , and $padding = 0$ . We will explain padding and stride in the text below. . . . . | 28 |
| 2.7 | Max Pooling. A pooling operation has a kernel and stride of similar dimensions. This operation replaces the output of the area covered by the pooling kernel with a summary statistic (the maximum) of that area [66]. The resultant feature map's resolution is reduced by a factor of $K_h \times K_w \times K_d$ pixels, where $K_{(h,w,d)}$ are the pooling kernel dimensions. . . . .                                                                                                                                 | 31 |
| 2.8 | Dropout, a regularisation technique that helps alleviate the problems of overfitting by randomly dropping neurons and their connections. Fig. (a) is a standard neural network with all neurons fully connected, and in (b) dropout randomly removes some of the neurons and connections forcing the network to learn more parameters thus eliminating or reducing the chances of pattern memorisation. Source [131]. . . . .                                                                                              | 32 |
| 3.1 | VGG Architecture variants. VGG has various models and configurations. These various models are usually named with a number suffix which denotes the number of weight layers in the model. VGG-16 (block D), submitted at the 2013 ILSVRC achieved a TOP-5 test accuracy of 92.7% on the ImageNet dataset. LRN is the local region normalisation, maxpool is short for max-pooling and FC stands for the Fully-connected layer. Source [130] . . . . .                                                                      | 49 |
| 3.2 | Residual Learning Building Block . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 50 |
| 3.3 | Fully Convolutional Networks by Long et al. [92]. FCNs can be trained to make dense pixel-level classifications. This is achieved by transforming fully connected layers into convolution layers to produce heat maps from the feature maps. Upsampling and transposed convolution (fractionally strided convolution) gives the desired segmentation output. . . . .                                                                                                                                                       | 54 |
| 3.4 | SegNet Architecture. This built on the VGG backbone, except that it uses pooling indices for upsampling, unlike FCNs. SegNet is also fully convolutional just like the FCNs. Source [5] . . . . .                                                                                                                                                                                                                                                                                                                          | 55 |
| 3.5 | AdapNet Architecture. Source [142] . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 56 |

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |    |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.6  | Full-Resolution Residual Networks (FRRNs). This approach uses two processing streams, the pooling stream performs the convolution and pooling operations while the residual stream carries features at full resolution. The two streams are combined using full-resolution residual units (FRRUs) providing better localisation and boundary adherence [118]. . . . .                                                                                                                 | 56 |
| 3.7  | Graphical visualisation of Atrous convolution (b) compared to standard convolution (a). Atrous convolution allows control of the field of view, for all $r > 1$ . This is important for enlarging the field of view to incorporate larger context as well finding the most optimal trade-off between accurate localisation and context assimilation [22]. . . . .                                                                                                                     | 57 |
| 3.8  | Atrous Spatial Pyramid Pooling is used to encode features at multiple scales. From atrous convolution, ASPP is used to adjust the Field-of-Views (shown in different colours) when performing pooling operations [22]. . . . .                                                                                                                                                                                                                                                        | 58 |
| 3.9  | DeepLabv3, Going Deeper with Atrous Convolutions. In part (a) we can see that when using standard convolution, output stride increases as the depth of the network increases, resulting in much smaller feature maps which negatively impacts segmentation results. In part (b) however, atrous convolution helps keep the stride constant while the field of view is increased, without increasing the number of parameters. This leads to better segmentation results [22]. . . . . | 59 |
| 3.10 | BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation [151]. . . . .                                                                                                                                                                                                                                                                                                                                                                                            | 60 |
| 3.11 | DenseASPP, (a) illustrate DenseASPP in detail, the output of each dilated convolutional layer is concatenated with input feature map, and then feed into the next dilated layer. Each path of DenseASPP compose a feature representation of correspond scale. (b) illustrate this structure in a more concrete version [148]. . . . .                                                                                                                                                 | 61 |
| 3.12 | Examples from the CSIR dataset. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                               | 63 |
| 4.1  | Data collection vehicle and camera mounting. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                  | 67 |
| 4.2  | An example of the class annotations from the CSIR dataset. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                    | 67 |
| 4.3  | Comparisons between the Freiburg Forest (top row) and CSIR (bottom row) datasets. As illustrated in this figure, the two datasets are different but they compliment each other. Thus, the Freiburg Forest dataset was a fitting choice for this research. . . . .                                                                                                                                                                                                                     | 68 |

|     |                                                                                                                                                                                                                                                                              |    |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4.4 | An overview of the design steps in our research pipeline. This research is a comparative study of state-of-the-art CNN architectures for semantic segmentation. All the architectures were trained and evaluated on two similar datasets (CSIR and Freiburg Forest). . . . . | 69 |
| 5.2 | Progression of the learning process through the segmentation networks. This is from the BiSeNet architecture. . . . .                                                                                                                                                        | 91 |
| 5.3 | Progression of the learning process through the segmentation networks. This is from the BiSeNet architecture. . . . .                                                                                                                                                        | 92 |
| 5.4 | Performance plots for the top 4 models on Freiburg Forest dataset . . . . .                                                                                                                                                                                                  | 93 |
| 5.5 | Performance plots for the top 4 models on CSIR dataset . . . . .                                                                                                                                                                                                             | 94 |
| 5.6 | Performance plots, models on Freiburg Forest dataset, extended results . . . .                                                                                                                                                                                               | 95 |
| 5.7 | Performance plots, models on CSIR dataset, extended results . . . . .                                                                                                                                                                                                        | 96 |

# List of Tables

|     |                                                                                                                                                                                             |    |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.1 | State-Of-The-Art CNN Architectures for semantic segmentation in 2016. The stars represent the rating of IoUs on a 5-point scale. Source [44] . . . . .                                      | 53 |
| 4.1 | Data collection device general specifications. . . . .                                                                                                                                      | 66 |
| 4.2 | Hardware Specifications . . . . .                                                                                                                                                           | 72 |
| 5.1 | Predictions on Freiburg Forest dataset - Top 4 models . . . . .                                                                                                                             | 77 |
| 5.2 | Predictions on CSIR dataset - Top 4 models . . . . .                                                                                                                                        | 78 |
| 5.3 | Predictions - All models - Freiburg Forest extended results . . . . .                                                                                                                       | 82 |
| 5.4 | Predictions - All models - Freiburg Forest extended results . . . . .                                                                                                                       | 83 |
| 5.5 | Predictions - All models - CSIR extended results . . . . .                                                                                                                                  | 84 |
| 5.6 | Predictions - All models - CSIR - extended results . . . . .                                                                                                                                | 85 |
| 5.7 | Freiburg results - Model size, computation time, and mean IoU. The best performing model in each category is highlighted in bold text. The top four models are highlighted in blue. . . . . | 86 |
| 5.8 | CSIR results - Model size, computation time, and mean IoU. The best performing model in each category is highlighted in bold text. The top four models are highlighted in blue. . . . .     | 86 |



# List of Abbreviations

## Acronyms / Abbreviations

*ANN* : Artificial Neural Networks

*ARM* : Attention Refinement Module

*ASPP* : Atrous Spatial Pyramid Pooling

*BiSeNet* : Bilateral Segmentation Network

*CIFAR* : Canadian Institute For Advanced Research

*CMoDE* : Convolutud Mixture of Deep Experts

*CNN* : Convolutional Neural Network

*CRF* : Conditional Random Fields

*CSIR* : Council for Scientific and Industrial Research

*DBEL* : Detonics, Ballistics and Explosives Laboratory

*DDSC* : Dense Decoder Shortcut Connections

*DSI* : Department of Science and Innovation (South Africa)

*FCN* : Fully Convolutional Networks

*FFM* : Feature Fusion Module

*FPS* : Frames Per Second

*FRRN* : Full-Resolution Residual Networks

*FRRU* : Full-Resolution Residual Units

*GCN* : Global Convolutional Network

*GMM* : Gaussian Mixture Model

*ICNet* : Image Cascade Network

*IID* : Independently and Identically Distributed

*ILSVRC* : ImageNet Large Scale Visual Recognition Competition

*IOU* : Intersection Over Union

*LRN* : Local Response Normalisation

*MRF* : Markov Random Fields

*MSE* : Mean Squared Error

*ORB* : Oriented FAST and rotated BRIEF

*PSPNet* : Pyramid Scene Parsing Network

*ReLU* : Rectified Linear Units

*RGB* : Red-Green-Blue

*SGD* : Stochastic Gradient Descent

*SIFT* : Scale Invariant Feature Transform

*SLAM* : Simultaneous Localisation and Mapping

*SURF* : Speeded Up Robust Features

*SVM* : Support Vector Machine

*UGV* : Unmanned Ground Vehicles

*VGG* : Visual Geometry Group



# Chapter 1

## Introduction

Artificial intelligence has been growing in leaps and bounds leading to the automation of many tasks that were historically only performed by human operators. The idea of mobile vehicles driving themselves was only a fantasy a few decades ago. However, research efforts towards self-driving vehicles has intensified and progressed, reaching maturity and popularity to such an extent that this fantasy has become a reality. Society in general is still grappling with the advent of autonomous mobile vehicles and how to successfully and safely integrate them into people's daily lives. There are ongoing debates regarding infrastructure and legislation for autonomous vehicles. On the scientific front, most research has been focused on self-driving cars in structured urban environments. The use cases for autonomous mobile vehicles also extends to rural and unstructured domains and not much attention has been directed towards this area of research. The focus of this work is on perception for autonomous vehicles in these adverse and unstructured terrains.

### 1.1 Background

In the past decade the world has seen an increasing interest in, and demand for, autonomous vehicles and autonomous mobile systems in general. Complete and successful autonomy of mobile systems such as unmanned ground vehicles (UGVs) is unequivocally dependent on perception. Perception is the ability of an agent to semantically interpret its operational environment through vision [41, 129, 137]. Autonomous systems can only operate in a goal directed manner if and only if they are equipped with the ability to successfully perceive and understand their operational environments [108].

The dual problems of mapping and localisation are at the core of research in autonomous mobile systems. It has been established that the problems of mapping and localisation have

to be performed concurrently. This is generally referred to as simultaneous localisation and mapping (SLAM) [7]. Extensive research has been conducted and notable milestones reached on the SLAM problem [7, 38, 76, 104, 105, 109, 114, 138]. Some of these milestones include mapping, loop closure, and localisation in indoor and structured outdoor environments [7, 104].

SLAM solutions seek to solve one joint problem. However, their approaches and applications vary greatly. Applications include SLAM for indoor or outdoor environments, ground or aerial vehicles. The type of data used for the SLAM solutions also determines the approaches used. SLAM approaches have manifested in varying combinations of algorithms [7, 76].

Most SLAM algorithms rely on unique stable features, key-points and anchor points [105] this enables autonomous agents to execute navigation decisions while performing various tasks. A general lack of stable hand-crafted features for classical vision algorithms was the main challenge to the initial objective of this research. This was scene comprehension and localisation through hand-crafted features. The focus of the research then shifted to pixel-level semantic segmentation using deep learning approaches. This is fundamentally the classification and localisation of objects in a scene at a pixel level. In this regard, the term object is used loosely to refer to any distinctly classifiable collection of pixels.

Deep learning approaches have recently been continuously reigning superior over traditional/classical methods in perception and vision tasks [56, 75, 78, 92]. This is primarily due to their non-reliance on selected hand-crafted features, they adopt a more robust and generalised learned-feature approach through representation learning [51].

### **Definition: Unstructured outdoor environments**

For the purposes of this research we define unstructured outdoor environments to be natural and non-urban scenes. These are environments largely characterised by sparse and homogeneous features. Furthermore, there is usually low contrast among different object classes. Thus, the very closely related features and fine boundaries of these different object classes can be extremely difficult to ascertain, even for the human eye.

Examples of the unstructured non-urban environments from our custom dataset (CSIR) are provided in Fig. (1.1). Rural environments are typically characterised by homogeneous features. This is shown by the similarities of the features among the different classes, for

instance, the road and grass classes. A further challenge is the general lack of distinctive colours for some of the classes.



**Figure 1.1** Images of Unstructured Outdoor Scenes (CSIR). Non-urban environments are typically characterised by homogeneous features and low contrast.

The input image and sample output in Fig. (1.2) show what this research aims to achieve. The original image is provided on the left, and the image on the right shows all the pixels of the original image assigned to all pre-defined semantic classes: grass, trees, dirt\_road, sky and void. We are interested in assigning all the pixels in each sample to unique semantic classes. These are the classes that are practical and directly useful for the purposes of autonomous navigation in these adverse environments.

## 1.2 Motivation

This project is part of a joint initiative by the Department of Science and Innovation (DSI), and Council for Scientific and Industrial Research (CSIR) to develop an autonomous vehicle for unstructured environments. As the Perception Team, part of our objective was solving the object localisation problem in outdoor unstructured environments for UGVs. Our first attempt made use of classical hand-crafted feature detectors such as Scale Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), and Oriented FAST and rotated BRIEF (ORB) [102, 104, 105]. We soon encountered challenges due to the nature of the operational environment, rendering most, if not all, classical solutions sub-optimal for our problem.

Given the shortcomings of classical methods in these environments: homogeneous features, low contrast and a general lack of distinctive key points and objects, pixel-level scene understanding was the next best alternative. This shift of research objectives to semantic



(a) Sample input image

(b) Sample target image

**Figure 1.2** An example on the input-groundtruth pairs. Given an input image (a), semantic segmentation assigns every pixel in the image to pre-defined classes such as grass, trees, road and sky.

segmentation was based on the hypothesis that scene understanding through semantic segmentation can be used as a basis for perception in solving the localisation problem. Our current research thus seeks to perform successful pixel-level classification and localisation in unstructured adverse environments.

### Semantic segmentation, instance segmentation and object detection

There is usually a lack of clarity with regards to semantic segmentation, instance segmentation, and object detection in terms of their associated usage in computer vision. Here we provide brief clarification to some of these most fundamental computer vision topics. **Semantic segmentation** refers to the parsing of an image and associating each pixel to a predefined class label. On the other hand, **object detection** is the identification of objects of predefined categories and localising them using bounding boxes [34, 83, 118].

While semantic segmentation is tasked with assigning pixels to known classes in an image, it has no notion of the different instances of an object. Object detection gives coarse localisation of different object instances. Unlike the usage of bounding boxes in object detection, **instance segmentation** localises different objects at pixel level [4, 34, 140].

Most research has been focused on segmentation for vehicles operating in structured urban environments [6, 27, 34, 113, 130, 158]. Consequently the majority of datasets and benchmarks publicly available are for urban environments. For the purposes of this project, we have created our own novel dataset for unstructured scenes: the CSIR dataset. We followed the labelling and annotation rules as stipulated in the Cityscapes dataset [27]. This will be made publicly available for use by other researchers. More details on this will be provided in the methodology and experiments chapters.

## 1.3 Aim and scope of this research

As already alluded to in Section. (1.2) above, the primary objective for this research is pixel-level scene segmentation. The outcome of this work can be used towards equipping UGVs with the ability to perceive and accurately interpret unstructured outdoor scenes. Convolutional Neural Networks (CNNs) have shown great success in most computer vision tasks, including semantic segmentation [92]. However, this success has mostly been in structured city environments.

To this end we conduct a thorough survey on state-of-the-art deep learning architectures for semantic segmentation. This research seeks to provide answers to the following questions:

- Using Convolutional Neural Networks (CNNs) and colour RGB images, is it possible to segment an unstructured scene according to its multiple semantic classes: **dirt\_road**, **grass**, **trees**, **sky**, and **void**?
- Given limited resources and time constraints for data collection and annotation, are there benefits in leveraging transfer learning and weights from models trained on images from other datasets for related tasks in order to improve semantic segmentation results in unstructured adverse environments?
- What are the key factors necessary for optimal results for the task of pixel-level segmentation in unstructured outdoor domains?
- What is the best, or most efficient, CNN architecture for real-time semantic segmentation in unstructured outdoor environments?

### 1.3.1 Delimitations and assumptions of the research

The datasets used in this research only include clear sky and daylight illumination conditions. These two datasets used were collected during dry winter (CSIR) and dry summer (Freiburg

Forest) conditions. This was due to limited time and resources for collecting data that encompasses all the other possible conditions in unstructured environments. The data collection for the CSIR dataset was determined by the time of the year when the research commenced as well as availability of resources. The Freiburg Forest is an open source dataset is an open source dataset and the reasons for their choices regarding conditions for data collection are unknown to us.

We note that the conditions covered in both datasets are limited. However, these two datasets are sufficiently rich to satisfy the requirements of pixel-level semantic segmentation. As stated above, the primary goal is pixel-level segmentation of unstructured outdoor scenes. To this end, it suffices to only use monocular vision. Additionally, all pixel are classified according to colour and texture.

The two datasets used in this research do not include cloudy and rainy conditions, cover rocky, snowy, and muddy terrains. Furthermore, datasets used do not include pedestrians, cars and other mobile agents that could appear in the scenes. This research did not use stereo (multiple viewpoints) and depth information. This is an initial attempt to determine the feasibility of pixel-level segmentation in outdoor environments using the minimum applicable conditions. The remainder of the conditions will be considered in future research.

## 1.4 Dissertation layout

The rest of this dissertation is laid out as follows, in Chapter (2) and (3) we provide exhaustive background research and literature review, respectively. Chapter (4) outlines the primary purpose of this work, stating and defining the research problem and the proposed research objectives. Chapter (4) also provides details of the research problem and the solution space and procedure we will be exploring. Furthermore, in Chapter (4) we discuss all our implementation details, hardware and libraries used. In Chapter (6) all the results and findings are analysed and discussed. Chapter (6) a summary of the entire dissertation.

# Chapter 2

## Background and related work

### 2.1 Introduction

In this section we present an extensive account on the progression of image processing, machine learning and computer vision with respect to semantic segmentation. We begin with traditional image processing and computer vision approaches. Furthermore we introduce convolutional neural networks (CNNs) and their applications in semantic segmentation. We focus our attention mostly on state-of-the art convolutional neural network approaches. Lastly, we conclude the section with a brief summary of everything presented in this section.

### 2.2 Traditional semantic segmentation approaches

Gestalt psychology [146] has long been fixated on understanding the perception mechanisms of the human mind. The fundamental question behind Gestalt psychology is how the human mind is able to acquire and maintain semantic perceptions from seemingly chaotic observations, even without any prior knowledge of the objects or the environment as a whole. Perceptual grouping is the ability of the biological mind to extract high level features from low level image features with no specific understanding of the image content. Psychologists in this area empirically concluded that the human perception system visualises a global whole with rules governed by psychological groupings [146].

Psychologists proposed that the biological brain makes use of hierarchical grouping of low-level features to get context of high-level structures. Hierarchical grouping constitutes of proximity, similarity, continuation, closure, as well as symmetry. Tasks such as processing, indexing, and retrieving can be aided by perceptual grouping through highly compact image



representations [147, 157]. Pixel-level or semantic segmentation of images is founded on this theory.

As mentioned in the previous chapter, image segmentation is a procedure that refers to the task of partitioning a natural image into a finite number of non-overlapping semantic classes. Segmentation has proved its usefulness in a multitude of computer vision challenges. Examples include annotation [18] and superpixel segmentation [2, 18, 26], scene parsing [94] and model estimation [71]. Mori et al. [17, 53, 103] use segmented classes in object recognition for better localisation. Approaches used for image segmentation: mean shift [2] and normalised Cuts [35] are also applicable to related challenges such as clustering and density estimation.

Traditional bottom-up segmentation methods made segmentation an ill-posed problem because of the absence of an apparent definition of object. This gives rise to a carefully devised metric for the task of image segmentation. In addition to Gestalt psychology, other research in perception has produced robust protocols for advancing segmentation algorithms. Cognition studies by [97] concluded that the human mind perceives object class boundaries in regions of drastic contrast changes and the object class salience depends mostly on boundary strength, orientation, and relative size. Furthermore, Gestaltism empirically argues that human perception groups entities by similarities in colour or shape among other properties. Biological perception systems conveniently link contours whenever the properties of the observed entity have noticeable gradients and direction. Furthermore, segmentation is particularly challenging because it is a representation problem. It begs the question, "What is the best way to represent the object?" [53, 97].

Fully supervised methods perform well in learning high-level and global features. However, they are heavily dependent on labelled data thus limiting them to a finite, known number of classes. While semi-supervised and unsupervised approaches have very little to none labelled-data dependencies, they are much more challenging to structure and implement [15, 110]. In the past decade most approaches have leaned towards well defined objective functions and optimisation of specific global criterion [2, 15, 35, 144]. Other approaches consist of iterative processes with a human-in-the-loop. These are commonly referred to as Interactive Methods [13].



The following section presents a review of mainstream traditional bottom-up segmentation methods namely K-means (very closely related to Lloyd's Algorithm or Voronoi Relaxation) [91], mean-shift [26], super-pixels [2] and normalised cuts [24].

## 2.3 Bottom-up approaches

This category of methods is based on an initial focus on low-level features such as pixel groupings according to some local feature similarity such as colour, curvature or texture. These various low-level local similarities or local groupings can then be clustered together using techniques such as graph cuts, mixture models and mode shifting among many others. These method generally fail to incorporate the notion of an explicit object [2, 15, 35, 39, 144].

The overall bottom-up category can be further grouped into two secondary categories, **continuous** and **discrete**. Discrete approaches represent an image as a discrete grid whereas continuous approaches represent images as continuous functions.

### 2.3.1 Discrete approaches:

**k-means** [96] - This algorithm tries to assign each data point to one specific cluster based on some defined criteria. The centre of these clusters are called centroids. The first step is to choose a random number of centroids then assign each data point to one of these centres. Updates to re-assign each point to more accurate centres are then iteratively performed based on the feature space, or a distance measure. An example would be to compute the Euclidean distance of each data point from the given k-centroids and re-assign the data points to new centres accordingly until a termination criteria is satisfied. This criteria could be when there are no more new assignments of data points to different centres in the iterative process. This entire algorithm essentially assumes that all the data points in any given distribution can be successfully and accurately assigned to a computable number of centroids. Consequently, k-means has a generalisation problem with more complex and asymmetric distributions.

**Mixture of Gaussians** [96] - this is a slight improvement of k-means. Unlike with k-means, the cluster centroid is replaced by a covariance matrix within a mixture of Gaussians. Assuming that a set of  $n$  distributions with feature vectors  $\{x_1, x_2, \dots, x_n\}$  are drawn from a Gaussian Mixture, we have:

$$p(x|\pi_k, \mu_k, \sum_k) = \sum_k \pi_k \mathcal{N}(x|\mu_k, \sum_k) \quad (2.1)$$

where  $\pi_k$ ,  $\mu_k$ ,  $\Sigma_k$  represent mixing weights, mean vector and covariance matrix of the  $k$ th component, respectively. The Gaussian Mixture Model (GMM) [137] is given by,

$$\mathcal{N}(x|\mu_k, \Sigma_k) = \frac{\exp\{-\frac{1}{2}(x - \mu_k^T)\Sigma_k^{-1}(x - \mu_k)\}}{(2\pi)^{n/2}|\Sigma_k|^{1/2}} \quad (2.2)$$

The parameters of the GMM are estimated using the Expectation Maximisation (EM) algorithm [30, 137] in computing estimate quantities of sample points in each cluster.

Parameter estimation is followed by the segmentation step which is achieved through assigning pixels to clusters with the highest probability [137]. Following [47, 132, 137], an extension of Gaussian Mixture Models encodes texture and boundaries by use of minimum description length theory to achieve improved results.

**Mean-shift** - First presented in 1975 by Fukunaga and Hostetler [43], Mean-shift is a non-parametric approach that considers the feature space as a probability density function [43]. The underlying intuition of this algorithm is that if the input is a set of data-points sampled from a density distribution, with a given mode, clustering can be achieved [26]. Consider the dense regions in the feature space to be the equivalent of local minima and maxima in a continuous function. Performing gradient descent on every data point until convergence. The modes of the feature space are then the equivalent of stationary points shown by gradient descent. Iterative convolution and mean shifting guarantees convergence at the centre of these dense regions. This dynamically determines the number of clusters and centroids using the modes in the feature space. For the mathematical details on Mean-shift we refer the reader to [26, 43]. Mean-shift is a great non-parametric approach but it has some limitations. The lack of a dynamic way to determine the optimal kernel size or step size during gradient descent is critical for convergence. Furthermore, this algorithm does not fare-well with high dimensional data. [43].

**Watershed** [144] - Considering an image as morphological surface, this algorithm applies flooding to segment the image into catchments and basins. This is achieved by constructing ridges where at image parts where different components meet. One main issue with this algorithm is over-segmentation. This is due to the association of each region with a local minima.

**Region Merging Methods** [3, 15, 39, 63] - Following Felzenszwalb and Huttenlocher [39], relative dissimilarity can be employed for segmentation purposes. This is achieved by optimising global group metrics. An image can be mapped to a graph with a 4- or 8-neighbour

structure where the pixels represent nodes and the weights of their edges are a reflection of their colour dissimilarity. At first, each node is its own component, then define a dissimilarity coefficient  $diss(C)$  as the largest value in the minimum weight spanning tree of a component  $C$  [3]. Sort in ascending order. Then two regions  $C_1$  and  $C_2$  are merged provided the weight of the edge between them is less than a certain threshold which is dynamically computed.

A closely related approach to the one above is **Edge Based Region Merging**. Starting with pixels, or super-pixels, this approach merges regions based on a similarity or dissimilarity measure. As mentioned in the previous method, this can be based on colour difference, but it also includes edge-strength and boundary length. **gPb-OWT-UCM** [3] propose the transformation of sets of contours produced by normalised cuts (Ncuts) [24]. In this approach, a probability edge map (**gPb**) is first generated to delineate salient contours. This is followed by applying watershed over the topological (**OWT**) surface defined by **gPb** for finer segmentation. The final step involves sorting and merging the edges between regions in ascending order [3]. This forms an ultrametric or super-metric contour map (**UCM**) which can be further threshold by a factor  $\lambda$  for final segmentation [3].

**Normalised cuts** [24] - Local image features such as lighting and perspective are crucial factors to image segmentation. Citing the limitations of approaches that use only local image statistics for segmentation, Shi et al. [24] argue in favour of segmentation by splitting an affinity graph which encodes global image statistics. The underpinning intuition here is that large clusters have large volumes, thus minimising  $Ncut$  encourages  $vol(S_i)$  to be proportionate in size therefore this approach achieves balanced clustering [24].

### 2.3.2 Continuous approaches

Variational methods [3, 14, 19] are also used for image segmentation. These approaches partition images into regions and they regard images as continuous surfaces as opposed to discrete grids and produces more visually coherent results. Below is a review of some of the mainstream traditional continuous bottom-up image segmentation methods:

**Mumford-Shah (MS) model** - this approach focus and emphasises the homogeneity of image features. It also boosts strong piece-wise edges or boundaries [111]. The **Active contour model** object detection is achieved by deforming contours towards image edges [14].

There is another basis for categorising bottom-up methods into two main groups. One group focuses on the likelihood of a region belonging to an object [35, 63, 110] and the other

group tends to over-segment [2, 39]. Bottom-up methods have difficulty in obtaining object regions as they only use low level features. Proposed alternatives include a combination of hand-crafted low-level features to represent global image features [68].

### 2.3.3 Superpixels

Super-pixel methods aim to partition an image into homogeneous regions smaller than the object or their parts [121]. Ren and Malik [121] argue that super-pixels are a more intuitive and efficient representation compared to the ordinary pixel techniques because local cues extracted at pixel-level are ambiguous and sensitive to noise. Arguments for superpixels:

- **Semantically more meaningful than ordinary pixel.** The super-pixels are generally perceptually and semantically consistent with regards to colour, and texture. Most structures can be well preserved. Furthermore, superpixels provide insights into shapes which are difficult to capture at pixel level.
- **Reduces model complexity.** Pixel-based methods have to deal with tens of thousands of individual pixels and this has a strong memory footprint and compromises the stability of the solvers. This impacts negatively on model segmentation accuracy and model efficiency. The use of super-pixels can reduce the number of model parameters thereby alleviating computation burden.

#### Superpixel paradigms

Many bottom-up approaches are amenable to over-segmentation by parameter tuning. For instance, increasing number of clusters for normalised Cuts, controlling region size for Region Merging methods and tuning the kernel size for Mean-Shift.

Another paradigm focuses on producing much faster super-pixel segmentation by re-scoping optimisation from a global image to local non-overlapping initial regions. Region boundaries are then adjusted to account for salient object contours. TurboPixel [82] uses deformation of the original spatial grid into more compact regular regions by incorporating geometric flow as indicated by local gradients.

### 2.3.4 Interactive methods

Some image segmentation approaches rely on a human-in-the-loop for accurate results. Unlike bottom-up methods that rely strictly on low-level features, the human user has better

intuition in terms of the object regions. This can be used to guide segmentation algorithms. Vision problems like image editing, medical image analysis and other tasks where precision is of paramount importance generally find interactive methods really helpful.

Most research [54, 98, 149] on interactive segmentation techniques generally agree on the following procedures:

1. initial input is provided by the user,
2. segmentation network produces segmentation outputs,
3. agent provides further constraints, then repeat step 2,
4. iterate till a certain criteria for accuracy is satisfied.

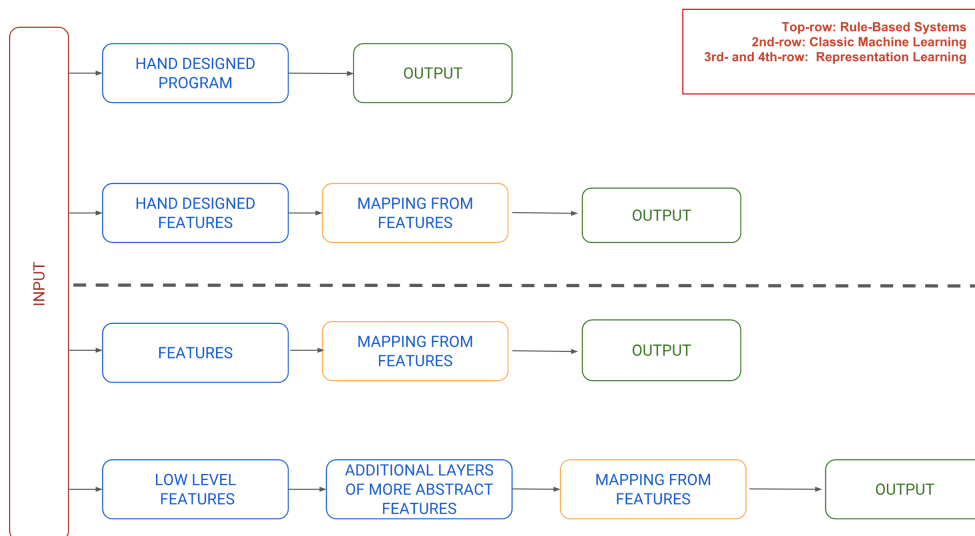
Furthermore, interactive segmentation methods should satisfy the following:

1. an interface for user-in-the loop to express various constraints,
2. produce fast and accurate results with minimal user assistance,
3. allow user to make additional adjustments for further refinements of results.

## 2.4 From traditional methods to deep learning approaches

The main challenge for artificial intelligence in general is solving tasks that humans perform naturally and intuitively, with ease. Yet, in the same vein humans struggle to describe formally, for example, looking at an scene while driving and being able to tell different objects apart within split seconds. Giving a formal description of how humans are able to perform this task with such ease, even in highly dynamic scenes, is a challenge that is yet to be fully solved. That is why methods that depend on hand crafted features have failed to generalise in these adverse dynamic environments.

Contrary to classical approaches that depend on specified features and human prescribed feature extractors, deep learning approaches attempt to learn the hierarchy of these features and concepts from low level features such as edges and corners to create a new feature stack that can be further learned to provide a better understanding of the input. This is called *Representation Learning*, where the solution to the perception problem is learning to discover the mapping from given input to the desired (according to human perception) output, as well as learning the representation itself. The progression of solutions to the perception problem



**Figure 2.1** Shows the progression of computer vision approaches from classical methods to modern learning approaches. From top to bottom, the *first row* shows an era when problem solving using computers began, everything was designed by the human, hand-designed programs mapped input to output (**rule-based approaches** [99]); the *second row* is where entire section above falls under, elements of learning were introduced but the features were still crafted by the human (**classical machine learning**), the *third and fourth rows* show the era of **representation learning** [51, 101], most state-of-the-art approaches fall under this category.

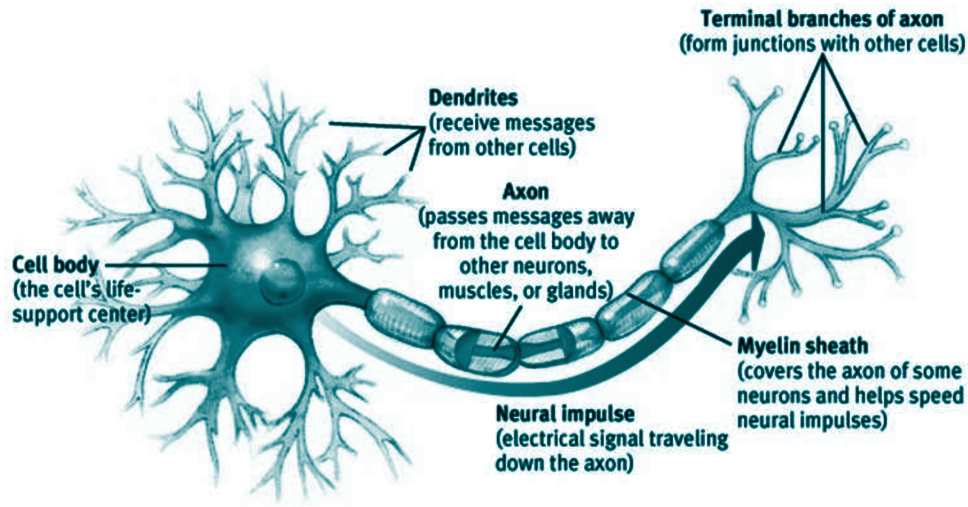
is condensed in Fig. 2.1 above.

In the sections below, we provide a brief walk-through of the learning process used by deep learning architectures.

## 2.5 Introduction to artificial neural networks

### 2.5.1 Biological neuron model

The mathematical formalisation of an Artificial Neural Network (ANN) is modeled on the biological neural systems. In the biological system, dendrites receive neurons as input signals that are processed as neural impulses or synapses inside the cell body to produce output signals via the axon terminals [62]. Neurons are the fundamental computational units in the biological model. Synapses are transitions of impulses between axons of neurons and the dendrites of other axons. The levels of interactions in these systems are determined by the strength of the synapses.



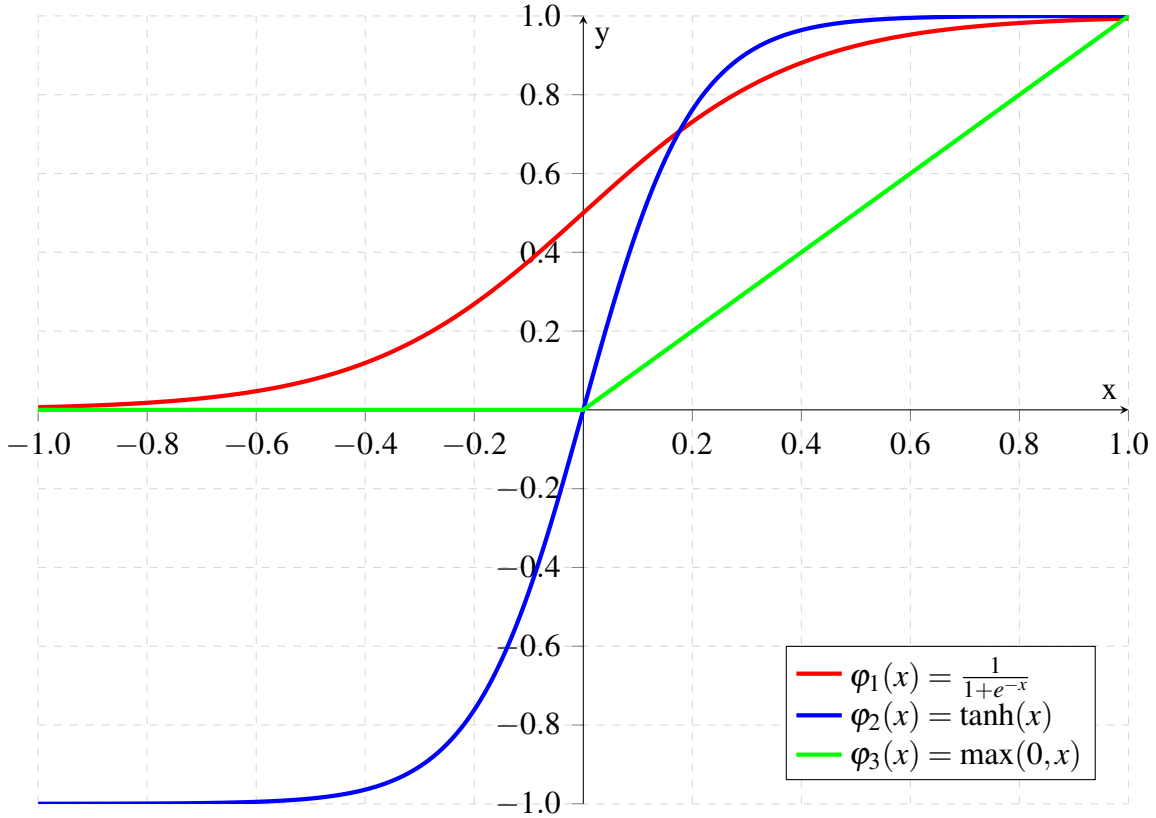
**Figure 2.2** The **biological neuron model** is the main inspiration behind ANNs. In the biological model the **dendrites** receive signals as **input**, that signal is then processed as **neural impulses** into the **cell body** and flows through the **myelin sheath** to produce **output** signals at the **axon terminals**. [1, 62]

Mathematically, these synaptic interactions are learnable and can be modeled with *weights*  $w_i$ . Input signals  $x_i$  flow through dendrites into the cell body where they are summed. When certain thresholds are reached, the neurons then produce an output via the axons. This output production via the axons can be modeled with a non-linear activation function  $f(x)$  as follows:

$$A_i = f\left[\sum_i (w_i x_i + b)\right] \quad (2.3)$$

where,  $A_i$  is the activation of a neuron,  $f(x)$  is a non-linear activation function,  $w_i$  is the weight that stores the synaptic strength,  $x_i$  is the input signal, and  $b$  represents the neuron bias [62].

An activation is a function that defines the output of a given neural node, this output can also be used as input to other neural nodes [63]. This involves mapping or squashing a set of output values into a prescribed range, for example a range of  $-1$  to  $1$ . There are various activation functions. Examples include the **sigmoidal**, **identity**, **binary step**, **sinusoidal** and **hyperbolic tangent** among a few others.



**Figure 2.3** Examples of activation functions, Sigmoid (red), Rectified Linear Units (green) and the hyperbolic tangent (blue).

The **Rectified Linear Unit (ReLU)** has proven more computationally efficient and is the most commonly used [51, 62]. There are also many other variants of activation functions, but those are beyond the scope of this work. In this text we focus on the ReLU activation function, this will be covered at length in Section 2.5.5.

### 2.5.2 Universal approximation theorem (Cybenko's theorem)

Artificial Neural Networks (ANNs) are formed by compositions and superpositions of a single simple nonlinear activation function. Consequently, the output of an ANN is the value of the resultant function from a particular composition and superposition of the nonlinearities [29]. The challenge for ANNs is therefore to identify the realisable classes of functions consistent with these compositions and superpositions.

Kolmogorov representations [72] argued that all continuous functions of  $n$ -variables have exact representations with regards to finite superpositions and compositions of a small



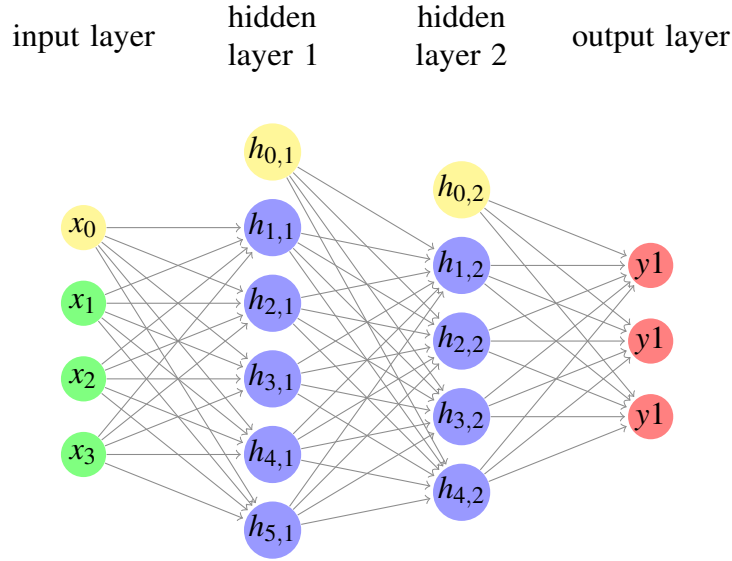
number of functions over one variable. The issue with these representations, however, was that they involved different nonlinear functions. The power of ANNs was realised through representations of finite linear combinations involving the same univariate function courtesy of the works by George Cybenko [29]. In particular, Cybenko's Theorem illustrates an ANN made up of a single hidden layer with a finite number of neurons has the capability to approximate any continuous function to arbitrary degrees of precision.

### 2.5.3 Artificial neural networks

Artificial Neural Networks (ANNs) are modeled as a system of neurons connected via multiple layers. These layers consist of the input layer, an arbitrary number of intermediate layers called hidden layers, then finally the output layer. Just like in the biological model above, the input is propagated from one layer and processed through the hidden layers to produce output in the output layer [51, 62]. ANNs are thus made out of dense networks of simple units, each unit taking a number of real-valued inputs and produces a real-valued output. The input values to some of the units can be the output values of other units. These networks therefore provide robust and practical ways of approximating vector-valued, real-valued, or discrete-valued target functions from target labels [101].

As illustrated in (Fig. 2.1) above, an ANN starts learning basic rules and structures regarding representation of the raw input in the first layer. These learned basic rules and representations are then combined to form more complex rules and representations in a hierarchical manner until the model is able to infer abstract concepts or conclusions regarding the raw input and the specified task. The process of passing and processing the input through hidden layers is called the **feed-forward process**. Accordingly, because of the operations at each layer, networks that only follow this procedure are called **feed-forward networks** [51, 101]. Feed-forward networks are weighted directed graphs that map inputs to outputs, and they have no loops. There are different variants of feed-forward networks. The most common is the **multi-layer perceptron** where neurons are organised into layers with unidirectional connections between them and different connectivities result in different network behaviour [62].

It is worth noting that the terms **artificial neural networks** and **deep learning** are sometimes used interchangeably. They are fundamentally synonymous. More precisely, the depth of a network is the number of hidden layers in an artificial neural network. In recent years, networks with more hidden layers have seen more success in vision problems as well as other computer science problems. Hence the name *deep learning* has become popular



**Figure 2.4** An example of a 4-Layer Artificial Neural Network (multi-layer perceptron). This network takes in  $x_1$ ,  $x_2$  and  $x_3$  and feeds it forward through the hidden layers,  $h_{(i,j)}$ , it also takes in a *bias* (yellow) signal at every processing layer. The output is three different classes  $y_1$ ,  $y_2$ , and  $y_3$ .

[31, 51, 62, 101].

There are various definitions for **deep learning** due to its diversity in application. A definition most befitting of this research is a branch of machine learning concerned with learning multiple levels of representations with regard to a hierarchy of features where high-level features are defined from low-level features. In computer vision, an observation, such as an image, can be represented in many ways but some representations are optimal for learning specific tasks from given examples. For example, understanding different regions of an image, semantic segmentation. The fundamental concern for deep learning is therefore defining what makes better representations and how to learn them [31, 51].

Deep learning has been around since the 1940's and it has gone through multiple name changes echoing or reflecting the various philosophical viewpoints and sentiments of various researchers over time. Some the name changes it has gone through include *cybernetics* and *connectionism* [80] among several others [51, 62]. It has also gone through various cycles of popularity and successes. The recent and ongoing popularity and success of deep learning approaches is courtesy of the ever increasing availability of training data. Additionally, computer hardware and software infrastructure has been continually improving thus making it possible larger deep learning models. As a result deep learning continues to successfully

solve even more complicated problems [31, 51].

### The learning process

In computer science, **machine learning** is a category of **AI** primarily concerned with the design of algorithms that are able to learn and improve from experience. In this research we adopt a loose definition of **artificial intelligence** as an algorithm design paradigm inspired by biological learning processes. A broad definition of machine learning is provided by Tom Mitchell [101]. Learning in this regard is taken to mean the ability of a computer program or algorithm to improve performance (**P**) from a experience (**E**) when solving a class of tasks (**T**). An example of a task is identifying specific objects in an image.

### Learning task (T)

A learning task refers to the problem to be solved. Problems such as classification (specifying which category some given input belongs to), regression (the prediction of numerical values given some input), speech recognition, transcription, machine translation, anomaly detection, density estimation and image-to-image translation are some examples of machine learning tasks [51, 101]. As mentioned in Section 1.2, the task for this research is semantic segmentation. This will be covered at length in the methodology chapter (4) ahead.

### Learning Experience: Training (E)

The learning experience is the nature of the learning process. The two polarities for the learning process are **supervised** and **unsupervised**. However, some learning paradigms do not directly fit into these two. Those are called **semi-supervised** learning approaches. In supervised learning the model experiences a set of examples and associated targets, which are respectively input vectors and vectors of desired features. In other words, the learning algorithm observes a set of random input vectors  $\mathbf{x}_i, \mathbf{x} \in \mathbb{R}^n, i \in [1, N]$  ( $N$  = number of examples) and their associated target vectors  $\mathbf{y}_i, \mathbf{y} \in \mathbb{R}^m, i \in [1, M]$  and learns to predict  $\mathbf{y}$  from  $\mathbf{x}$  by estimating  $\rho(\mathbf{y}|\mathbf{x})$ , the posterior probability of output  $\mathbf{y}$  given observed input  $\mathbf{x}$ . The learning algorithm has to learn make sense of the input based solely on the representation of the raw input [51, 62, 101]. This observation process is generally referred to as the the **training** process.

Factors that influence the design of the training process include data, training time and how the learning abilities will be measured. The learning process therefore demands careful and strategic choices how the model experiences the examples. Common practice is to split the datasets into three sets namely **train**, **validation**, and **test**.

The training set is the set that a learning model is allowed to experience and learn all the useful representations and rules as specified by the labels or ground-truths. This is usually takes up the largest portion of the original dataset, normally **60%**. The neural network adjusts the weights, learning parameters in accordance with this set. The generated models are then tested for performance comparison using the validation set, usually **20%** of the original dataset. This aims to help the neural network choose the best solution space. Even though the training and validation have to be kept mutually exclusive of each other, simultaneous training and validation helps with the selection of more robust solution spaces.

The test set, also mutually exclusive with the validation and training set, is held out during the training phase. It is used to depict real-world scenarios where the network will be required to solve similar tasks on unseen data. Performance on the testing set indicates the model's ability to generalise. Further details on data and datasets will be provided in the datasets section (3.3) ahead when we cover the datasets used in this research. Next, we delve into the performance measure and performance metrics.

### **The performance measure (P)**

The performance measure evaluates the learning ability of a machine learning algorithm. Performance evaluations can be both quantitative and/or qualitative. The choice of a performance measure is neither obvious nor always intuitive. Rather, it is dependent on its correspondence with the task being learned [51, 62, 101]. Below, we document some of the key concepts and metrics required for evaluating neural network performance.

### **Confusion matrix**

The confusion matrix  $C$  is an  $N \times N$  matrix used for comparing the actual values to the predictions, where  $N$  is the number of classes. Most metrics for classification are computed using the confusion matrix  $C$ . The confusion matrix has identical rows and columns corresponding to the classes being predicted. It is the basis for several other performance metrics. A

pixel-level confusion matrix is used for semantic segmentation to aggregate predictions for the whole dataset  $D$ .

$$\mathbf{C}_{i,j} = \sum_{I \in D} |x \in I \text{ such that } S_t^I(x) = i \text{ and } S_p^I = j|, \quad (2.4)$$

where  $S_t^I(x)$  is the groundtruth, target label of pixel  $x$  in image  $I$ ,  $S_p^I$  is the predicted label, and  $|\cdot|$  is the cardinality of the set  $\cdot$ . Equivalently,  $\mathbf{C}_{i,j}$  is the number of pixels with label  $i$ , whose corresponding correct classification is  $j$ . Let,

$$\begin{aligned} G_i &= \sum_{j=1}^N [\mathbf{C}_{i,j}] \text{ be the sum of pixels with groundtruth label } i, \\ P_j &= \sum_{i=1}^N [\mathbf{C}_{i,j}] \text{ be the sum of pixels whose correct prediction is } j, \end{aligned} \quad (2.5)$$

where  $N$  is the number of classes.

This research is solving a multi-class classification problem. In this context, the true-positives (tp) are all the correctly predicted pixels, i.e. all the diagonal elements in the confusion matrix. False-positives (fp), false-negatives (fn) and true negatives (tn) are the sums of all non-diagonal elements in the confusion matrix, and these represent the total number of prediction errors. Next, we define the metrics necessary for pixel-level semantic segmentation.

### Accuracy

Accuracy is the quantification of correct predictions. That is, the proportion of correctly classified pixels. The accuracy paradox highlights why accuracy is not a good metric. It can be crude, in the case where there are more false negatives than true negatives,  $tn < fn$ , accuracy will always increase. Similarly, accuracy will always increase when there are more false positives than true positives,  $tp < fp$ . This is usually the case when there are class imbalances. For pixel-level semantic segmentation we use the average accuracy defined by,

$$accuracy = \frac{\sum_{i=1}^N \mathbf{C}_{i,i}}{\sum_{i=1}^N G_i} \quad (2.6)$$

### Intersection over union (Jaccard index)

Named after Paul Jaccard [61], the Jaccard coefficient measures similarity or diversity of finite sets. This metric is of particular importance in semantic segmentation. Semantic

segmentation is pixel-level classification thus input images and their associated groundtruths can be considered as a sets of pixels. The *intersection over union* is therefore a measure of the overlap between the predictions and the groundtruths. This is calculated as the ratio of the *pixels covered by intersection* of the sets to *the pixels covered by the union* for those sets. Averaging the IoU over all classes gives the mean intersection over union.

$$Mean IoU = \frac{1}{N} \sum_{i=1}^N \frac{C_{i,i}}{G_i + P_i - C_{i,i}} \quad (2.7)$$

### Precision

This metric shows precisely the fraction of correct( true positives) predictions out of all the positive predictions. Its an indicator of all correct positive predictions, formalised as the ratio of the true positive predictions to all positive predictions.

$$precision = \frac{\sum tp}{\sum tp + \sum fp} \quad (2.8)$$

### Recall

In addition to precision, we may want to know the overall number of true events that were detected. That is, the amount of predictions out of all positive samples. It is sometimes referred to as the sensitivity or true positive rate.

$$recall = \frac{\sum tp}{\sum tp + \sum fn} \quad (2.9)$$

### F-Score

The F-score (F1-score) is used to summarise the classifier's performance with one number. It is the weighted average of precision and recall. Thus it takes into account both the true and false predictions. It is given by harmonic-mean of precision and recall scaled to 1.

$$f - score = 2 * \frac{precision * recall}{precision + recall} \quad (2.10)$$

There are various other performance metrics and divergence measures. Those are beyond the scope of this research.

At this stage the inspiration from, and the similarities with, the biological learning systems are now apparent. In order for an Artificial Neural Network to learn the general rules and the underlying representations of the given examples, the training data the model has to adjust the *weights* from Eqn. (2.3). This learning approach leverages algorithms such

as **backpropagation** [51] and optimisation approaches like **stochastic gradient descent** [48, 51] to understand and tune parameters of the network to accurately map sets of input-output pairs [101]. This will be covered exhaustively in Convolutional Neural Networks section (2.5.5) to follow.

### Frame rate

In computer vision **frame rate** is the rate at which an image acquisition collects image data. This is commonly quantified using the metric **frames per second (FPS)**. For instance, the image acquisition device used in our data collection captures information at 30. This is also used to measure how the speed at which an algorithm processes sequential or video image data. In the context of autonomous navigation, the FPS is a good metric for measuring how well the embedded navigation suite acquires and processes video data. The lower bound for human perception is 20, this is when human perception begins to register sequential frames as a video [42, 87]. This rate, however, is still very low and infeasible for motion as it contains a lot of flickering. For the human vision system to process a sequence of images as a continuous video without flickering the frame rate needs to be at least 46. However, biological vision systems can process image data at much higher frame rates [33, 137]. For autonomous navigation the frame rate needs to be higher than 10 for the vehicle to be able to react and adjust according to a dynamic operational environment. This is the lower bound and is far below human-level perception rates. The ideal situation is to get autonomous navigation perception systems to have a frame similar to, or better than, human perception rates [42, 87].

## 2.5.4 Feature-based neural networks

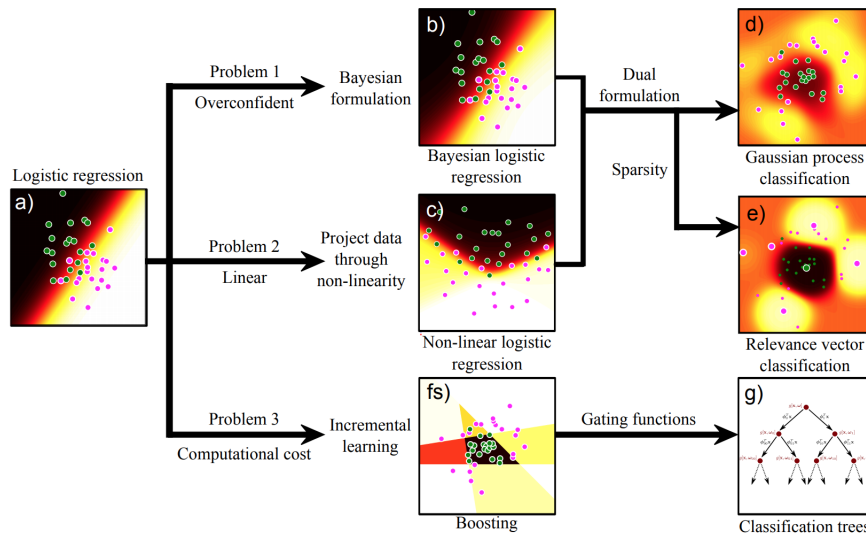
### Feature detectors and descriptors

Feature detectors and descriptors are used for establishing interesting points and regions in an image. These are used for extracting high-level information from raw pixels and can be used to capture distinctions among the various categories of an image. Examples of feature descriptors and detectors are Laplacian of Gaussians, Gabor filters, Haar-like filters, local binary patterns (LBP), texton maps, histogram of oriented gradients (HOG features), Scale Invariant Feature Transforms (SIFT), Speeded Up Robust Features (SURF), Oriented Robust Brisk (ORB) features, Bag of Visual Words and shape context descriptors among many others [9, 84, 93, 104, 120, 137].

After applying the feature detectors and descriptors, they can then be fed into classification models for segmentation. In the next section we discuss some of the main classification approaches.

## Classification

Classification is a discriminative approach for modelling the posterior probability  $\rho(k|\mathbf{x})$  over a discrete set of possibilities  $k \in \{1, \dots, K\}$  given a continuous vector of observations  $\mathbf{x}$  [8]. Given some feature vectors or data points from a training example  $\mathbf{x}$ , classification aims to assign ascertain the probability or likelihood of a new unseen data point belonging to some prescribed class. A classification module is trained using the original dataset and the associated labels. Probabilistic approaches are based on either the *Bernoulli probability* for binary classification or the *log categorical probability* for multi-class purposes [8, 120]. There are various probabilistic classification approaches, but they can be grouped as shown in Fig. 2.5 below.



**Figure 2.5** Categories of classification models [120]. Starting from basic logistic regression models and the various ways to address their major limitations gives rise to various categories of regression approaches.

Logistic regression models perform well on simple datasets. However, they are limited on complex datasets as they can only describe linear decision boundaries. Logistic regression models are trained using maximum likelihood. This leads to overconfidence in their performance thus making them prone to overfitting [8, 120]. Non-linear regression, Boosting and Bayesian Logistic regression are all approaches that attempt to address the shortcomings



of basic logistic regression approaches [8, 120]. Furthermore, the combination of Bayesian logistic regression and non-linear logistic regression leads to more complex and more robust approaches such as Gaussian process classification and relevance vector classification. Gating functions applied to boosting lead the family of various classification trees such as *decision trees* and *random forests* [51, 120, 137].

Very often in computer vision non-probabilistic classification approaches such as *Adaptive Boosting* (AdaBoost) [145] and *Support Vector Machines* (SVMs) [11, 127] are used. AdaBoost can often be used with the output from other weak learners/classifiers. A weighted sum from weak learners is used to represent the boosted classifier with a better performance. The adaptiveness is in its ability to give more weight to weak learners with higher classification results, this leads to improved overall classification. AdaBoost, however, is too sensitive outliers and noise [120]. SVMs are kernelised classifiers that represent the training examples as points in space mapped in such a manner that there is a clear boundary between points belonging to different classes. SVM clustering [11] is a modification to standard SVMs for unsupervised clustering of data points into various possible classes. Though non-probabilistic, SVMs can be used with *Platt-scaling* [117] for transforming the outputs of a classification model into probability distributions over multiple classes. Other feature-based classification methods include *nearest neighbour search* [10] and *k-nearest neighbour classification* [28].

Feature extraction can not be tweaked according to classes. The lack of generalisation when using prescribed features negatively performance in classification problems. Additionally, the use of numerous feature heuristics and manual tweaking of parameters to adjust for domain changes is not feasible for efficient scene parsing in autonomous navigation.

Most feature-based neural networks tend to suffer from the *curse of dimensionality* [51]. This is a phenomena that arises when, due to an increase in dimensions, the spatial volume increases rapidly to a point where the data becomes sparse. Learning with high-level prescribed features can lead to rapid increases in dimensionality compromising the statistics of the original features thus negatively impacting classification results. CNNs and deep learning in general have been shown to suffer minimal to no impact from dimensionality due to feature pooling [51].

Feature based classification on its own might be inadequate, however the associated classification methods can be used to classify features learned using CNNs. Different other solutions and models have also been used towards semantic segmentation. These borrow principles

from Markov Random Fields (MRF) and Conditional Random Fields as well as the energy function. We discuss these in the next section.

### Semantic image parsing

The objective of semantic parsing is to divide an image into non-overlapping parts that correspond to predefined semantic regions. Semantic segmentation saw a rise in popularity in the early 2000's due to success in vision problems namely object detection, tracking, pose estimation, reconstruction, and fast conditional random field (CRF) solvers. The most important objective of semantic segmentation is to provide a holistic understanding of the visual.

It depends on provided information/labels however learned high-level representations distinguishes it from interactive methods. Furthermore, semantic segmentation focuses on parsing an entire image into the "thing" (specific) and "stuff" (general) classes whereas object region proposal methods focus on producing possible "thing" candidates.

Most semantic parsing solutions focus on finding the most likely labelling on a **Markov Random Field (MRF)** or **Conditional Random Field (CRF)**. CRF is a probabilistic approach for modelling complex associations between observations and the associated outputs. This is made possible by the ability to factorise probability distributions over different labels of random variables, dense representations and efficient inference [57, 100, 128, 128, 137].

**Top-down and bottom-up combination:** This approach leverages the bottom-up approach's ability to capture "stuff", that is, general classes. On the contrary object detectors capture the "thing" class (specific objects). Therefore, their combination helps create a holistic understanding scenes.

**Inference:** Optimisation of the energy function can be achieved using various techniques such as belief propagation, GraphCut, Primal-dual methods etc. [67]. Traditionally CRF and MRF are limited to 4- or 8-neighbours. Modern graphical models that connect all pixels have become popular because of faster image filtering. The only requirement is that the pairwise term must be a Mixture of Gaussians. Krahenbuhl et al. [73] proposed higher-order terms for fully connected CRF.

**Data Driven:** This pipeline requires a pre-trained classifier and this is restricting when new classes are included. Other researchers [67, 89, 90] suggested non-parametric and data-driven solutions.

Traditional machine learning approaches like feature based neural networks are only feasible with relatively small training sets, limited by the curse of dimensionality, and consequently less computational resources. On the contrast, deep learning approaches such as CNNs are far more demanding with respect to both the required amount of training examples and compute resources. [31, 51, 137].

### 2.5.5 Convolutional neural networks: Building blocks

CNNs popularised by Yan le Cun [77] on the MNIST handwritten digit recognition problem, are specialised networks that process data that has grid-like topology, for instance image data. In this section we first separately provide brief theoretical underpinnings and building blocks for convolutional neural networks. We eventually bring together all the building blocks to describe how they are used in constructing complete convolutional neural networks.

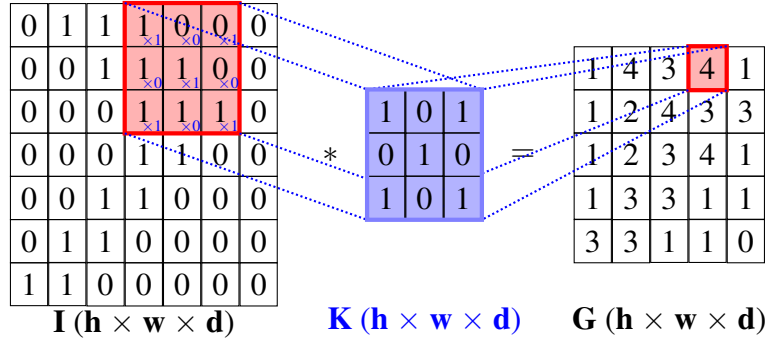
#### Convolution

Artificial Neural Networks are capable of solving different kinds of problems. Some networks specialise in specific families of tasks, usually depending on the nature of the input data. Instead of **general matrix multiplication**, CNNs use a mathematical operation called **convolution**.

In computer vision and machine learning applications, the majority of the training data is usually represented in the form of multi-dimensional arrays called tensors. The convolution operation is one of the most commonly used neighbourhood operators. It is a linear operator used for producing higher-level feature maps from lower-level feature maps. Consider an object detection task, convolution will start by detecting low-level features such as edges and corners. As the convolution progresses further into the network higher complex features that are descriptive of the object will be detected. As an example, suppose we have an image RGB (red, green and blue) ( $\mathbf{I}$ ) of size  $h$  (*height*),  $w$  (*width*), and  $d$  (*depth*). Depth is number of colour channels in an image, not to be confused with the depth of a neural network. To perform a convolution operation on this image we consider a kernel (the weight matrix) ( $\mathbf{K}$ ) that also has dimensions  $[h, w, d]$ . The convolution will produce output  $\mathbf{G}$ . A graphical illustration is provided in Fig. 2.6 below.

The mathematical formalisation of a 2-dimensional convolution is given as follows:

$$G(i, j) = (\mathbf{I} * \mathbf{K}) = \sum_m \sum_n \mathbf{I}(i - m, n - j) * \mathbf{K}(m, n) \quad (2.11)$$



**Figure 2.6** An example of the convolution operation with an input image ( $\mathbf{I}$ ) of dimension  $[h, w, d] = [7, 7, 1]$  and a kernel ( $\mathbf{K}$ ) of dimension  $[h, w, d] = [3, 3, 1]$ . The depth of the input image and the kernel in this example is 1 (one colour channel), and the convolution produces output  $\mathbf{G}$ . The weights in the kernel are the parameters that has to be learned or trained. Furthermore, in this example  $stride = 1$ , and  $padding = 0$ . We will explain padding and stride in the text below.

The area covered by the kernel of filter, the dimensions of  $\mathbf{K}$  is called the receptive field. Convolution is the sum of the sum of the element-wise multiplication of the original image  $\mathbf{I}$  and the kernel  $\mathbf{K}$ . This operation produces the feature map. The depth dimensions have to be the same for both the  $\mathbf{I}$  and  $\mathbf{K}$ , and consequently the output  $G$  also has the same depth value.

Since the convolution is a local linear operator the receptive filed , kernel size, is typically chosen to be smaller than the original image. This is important for getting meaningful and localised feature maps from the image. The operation iteratively slides the kernel across the entire image until all pixels are covered as shown in Eqn. (2.11) above. **Stride** ( $s$ ) is the number of pixels by which the kernel shifts at every sliding iteration. There are different ways of handling boundary pixels when performing convolution. Depending on various considerations, one can choose to ignore a certain number of boundary pixels or specify exactly how many boundary pixels to ignore. This is called **padding**,  $p$ . Kernel size, stride and padding all affect the size of the output feature maps. The size of the feature maps ( $G$ ) from any convolution operation with stride and padding included has the following form,

$$\begin{aligned}
 G_h &= \frac{\mathbf{I}_h + (\lambda p) - \mathbf{K}_h}{s} + 1, \\
 G_w &= \frac{\mathbf{I}_w + (\lambda p) - \mathbf{K}_w}{s} + 1
 \end{aligned} \tag{2.12}$$

where  $\lambda \in \mathbb{R}^+$ ,  $\mathbf{I}, \mathbf{K}, p, s$  are image, kernel, padding and stride respectively.

The equations 2.12 above show that convolution reduces the resolution of the output feature maps. Thus, careful consideration is needed when choosing kernel size, stride and padding. In most CNN architectures convolution is performed at each convolution layer throughout

the network. Depending on the number of kernels in the convolution process, the depth of the feature maps increases as the input is passed through all the convolution layers. From equation 2.12, taking depth and number of kernels  $n$  into consideration means the resultant activations or feature maps will have dimensions  $[G_h \times G_w \times n]$ .

Unlike ordinary matrix multiplication convolution has proven to be more efficient for several reasons. Convolution leverages *sparse connectivity*, *parameter sharing* and *equivariant representations* [51]. Training images are usually large with sparse useful features in localised regions, smaller sized kernels and sparse interactions or sparse weights help control over the number of parameters stored as well as the computational complexity of the model. Parameter sharing allows the model to use the same parameters for more than one function in that model. This means that instead of learning a separate set of parameters for every location the model only needs to learn one set. Consequently, parameter sharing facilitates equivariance. Equivariance means that as the input changes, the output changes the same way. This is useful for transformations and sequential tracking of features as the model goes from detecting low-level features to high level features.

### Rectified linear units

In section 2.5.1 above we mentioned several activation functions for non-linearisation. The Universal Approximation Theorem in [29] illustrated the importance of non-linearising activation functions. Multi-layer neural networks are proven to be universal function approximators when the activation functions are non-linear. Of particular relevance to this work is the Rectified Linear Unit (ReLU).

ReLUs are very similar to linear except that ReLUs output zeros across half its domain [51]. As a result ReLUs are easy to optimise. Derivatives and gradients through a ReLU are large and consistent for all active units. The gradient direction provided by ReLU for learning has been shown to be much more efficient than gradient directions provided by other activation functions that use second order derivatives [51]. Precisely, the ReLU is an element-wise operation that ensures non-linearity for all pixels in the feature maps by computing,

$$g(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ x, & \text{if } x > 0. \end{cases} \quad (2.13)$$

In short, equation 2.13 above shows that the ReLU eliminates negative activations. A graphical illustration of the ReLU and other activation functions was presented in Fig. (2.3).

One major limitation of the ReLU is that it can not be optimised via gradient-based techniques on examples for which their activation is zero [51]. Consequently, several ReLU variants have been proposed in an attempt to address this shortcoming. These variants are **absolute value rectification** [64],

$$g(x) = |x|, \quad (2.14)$$

**leaky ReLU** [95],

$$g(x) = \begin{cases} 0.01x, & \text{if } x < 0 \\ x, & \text{if } x \geq 0, \end{cases} \quad (2.15)$$

**parametric ReLU** [55],

$$g(x) = \begin{cases} \alpha x, & \text{if } x < 0 \\ x, & \text{if } x \geq 0, \end{cases} \quad (2.16)$$

and **maxout units** [52],

$$g(x) = \max(w_1^T + b_1, w_2^T + b_2). \quad (2.17)$$

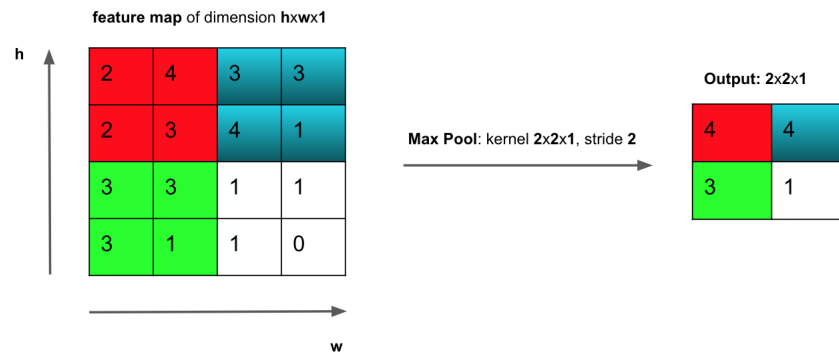
## Pooling

Pooling is a downsampling operation that replaces pixel values within a certain neighbourhood (pooling window) by one pixel. There various approaches to applying the pooling operation. The most commonly used is max-pooling [66] illustrated in Fig. (2.7). Other methods average pooling,  $L^2$ -norm, and weighted sum pooling [40].

Further from efficiently reducing feature map resolution and number of learnable parameters, pooling also enforces a certain degree of *translational invariance*. This is particularly important in problems where the presence of a feature is more important than where the feature is [51].

## Overfitting

One of the most important things in artificial neural networks is *model capacity*. Model capacity is the ability of an artificial neural network to fit a robust number of functions. The right combination of model capacity and *hypothesis space* guarantees better learning capabilities for neural networks. The hypothesis space is the set of solution functions that a



**Figure 2.7** Max Pooling. A pooling operation has a kernel and stride of similar dimensions. This operation replaces the output of the area covered by the pooling kernel with a summary statistic (the maximum) of that area [66]. The resultant feature map's resolution is reduced by a factor of  $K_h \times K_w \times K_d$  pixels, where  $K_{(h,w,d)}$  are the pooling kernel dimensions.

learning algorithm can choose from.

Regardless of how well designed a supervised learning model is, its ultimate learning abilities are bounded above by what the network observes during training. *Generalisation Error* or the testing error measures the performance of the network on new, previously unseen data. Ideally, it is desirable to have both the training and test errors at minimal levels. However most neural networks tend to have lower training errors but much higher test errors.

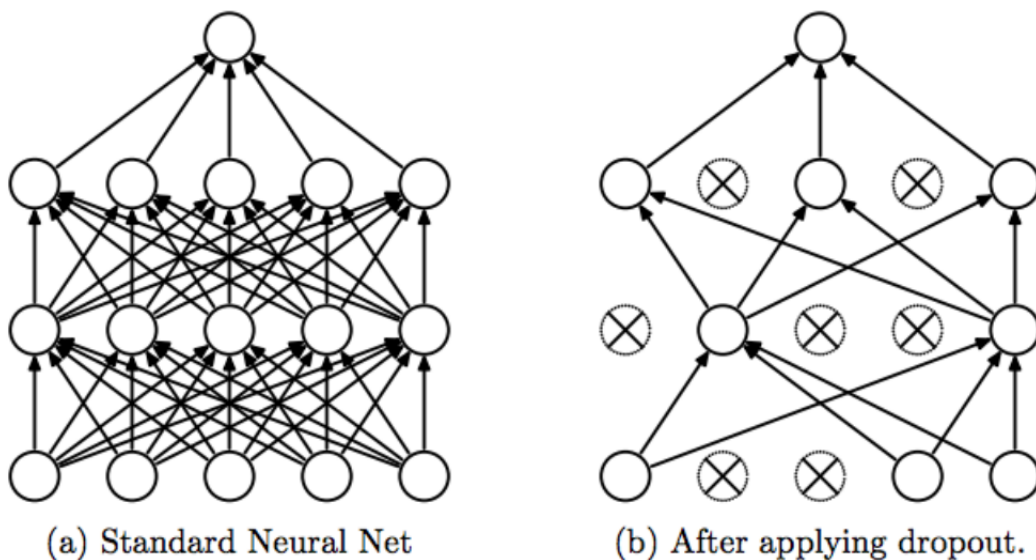
Overfitting is a problem that occurs when models have a higher performance on the training data, but perform poorly on the test set. This means that the model is over committing to the training data. It has also learned all the noise associated with training data but it has failed to learn the general underlying information. This means the model is of no use for predictions outside of the training data, it has no generalisation capabilities.

Overfitting can be caused by various things, but ultimately its an indication of low model capacity. This can be due the lack of sufficiently robust and diverse training data. Data that is representative of, or captures the an accurate distribution of the real world conditions will help avoid overfitting and increase model capacity. There are various approaches to addressing overfitting. Some focus on improving model capacity, for example *Occam's Razor* [49], *Vapnik-Chervonenkis dimension* and several others.

There is a collective family of techniques meant for dealing training error and the generalisation error. These will be covered at length in sections ahead. In the meantime, we focus on one of the most commonly adopted solutions for overfitting: *dropout*.

### Dropout

Neural networks are powerful because they have very large number of learnable parameters. However, this makes them prone to overfitting, for example, by memorisation of parameters. Dropout [131] helps alleviate the problems of overfitting by randomly dropping neurons and their connections from the neural network during the training phase.



**Figure 2.8** Dropout, a regularisation technique that helps alleviate the problems of overfitting by randomly dropping neurons and their connections. Fig. (a) is a standard neural network with all neurons fully connected, and in (b) dropout randomly removes some of the neurons and connections forcing the network to learn more parameters thus eliminating or reducing the chances of pattern memorisation. Source [131].

### Normalisation

In addition to the operations above, it is often common practice to add some variant of a normalisation operation. This is a local operator that changes the values of a specified region depending on the normalisation approach being used. *Lateral Inhibition* is a neuro-biological term referring to the potential of excited neurons to reduce the activity of its neighbouring



neurons. Normalisation ensures uninhibited flow of action potentials with local regions. An example of these techniques is the Local Response Normalisation (LRN).

$$G(x)^* = (1 + \frac{\alpha}{n} \sum_i x_i^2)^\beta \quad (2.18)$$

where,  $G(x)^*$  is the new region,  $n$  is the area of the local region,  $\alpha$  is a scaling weight, and  $\beta$  is used for centralising the sum.

### Fully Connected Layer

The entire neural network can essentially be split into two phases, feature extraction and classification. Regular feature based classifiers like those explained in section 2.5.4 above can be used after the feature extraction phase. However, there is a cheaper and more efficient way of extracting high-level meaning of the extracted features.

The *fully connected layer* is a non-convolutional layer that uses affine transformations and addition of the bias to facilitate classification. This layer is sometimes called the inner product layer. Neurons in this layer are connected to all the feature maps from the preceding layer, hence the name fully connected. It takes the feature maps from the extraction phase and flattens them into a one-dimensional array with the number of classes as the number of columns. The fully connected layer also enables end-to-end trainability.

After the fully connected layer computes the final classification scores, the next step is to verify the gap between the desired results and the predicted results. This is done via loss functions.

### 2.5.6 Loss function

The loss function essentially computes the difference between the learned feature maps from the input images ( $\mathbf{x}_i$ ) after going through all the layers and the feature maps of the desired or corresponding annotated groundtruths ( $\hat{\mathbf{x}}_i$ ). The loss function is usually taken as an average sum over individual per-example losses. This function is also called the cost or error function. There exists different variants of the loss function. Below is a brief account of some of the most commonly used loss functions.

### Mean squared error (MSE)

The mean squared error is one of the most commonly used, mostly so with feature-based neural networks (section 2.5.4) as well as classical segmentation methods (section 2.2). This is also sometimes referred to as the *quadratic cost function*. Consider a linear regression example where the task is to predict a target  $\hat{\mathbf{x}}_i$  given an input  $\mathbf{x}_i$  and  $N$  training images, then the loss is given by,

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (\mathbf{w}^T \mathbf{x}_i - \hat{\mathbf{x}}_i)^2, \quad (2.19)$$

where  $\mathbf{w}^T$  represents the learned parameters.

### Categorical cross-entropy loss

The Softmax function takes an  $N$ -dimensional vector of output scores  $\mathbf{s}$  and squashes them into a vector of real numbers in range  $[0,1]$ , a probability distribution. The SoftMax function is computed as,

$$f(\mathbf{s})_i = \frac{e^{s_i}}{\sum_{j=1}^C e^{s_j}}, \quad (2.20)$$

where  $\mathbf{s}_j$  are the network predictions for each class,  $\mathbf{s}_i$ .

The Cross-entropy loss is defined as,

$$CE = - \sum_{i=1}^C \mathbf{t}_i \log(\mathbf{s}_i),$$

where  $\mathbf{t}_i$  and  $\mathbf{s}_i$  are the groundtruth and CNN score for each class  $i \in C$ , respectively.

(2.21)

The Softmax function is applied as an activation function to the scores before the Cross-entropy loss computation. We use  $f(\mathbf{s}_i)$  to refer to the activations. For multiclass semantic segmentation we use categorical cross-entropy loss. This is a combination of the Softmax function and the Cross-entropy loss. In multi-class classification the labels are one-hot encoded, thus only the positive class  $C_p$  keeps its term in the loss computation. Furthermore, there is only one element of the target class  $\mathbf{t}$  which is not zero. Therefore  $\mathbf{t}_i = \mathbf{t}_p$ . Taking

into account the zero target labels, the Categorical cross-entropy can now be computed as,

$$CE = - \sum_{i=1}^C \mathbf{t}_i \log(f(\mathbf{s}_i)),$$

$$CE = - \sum_{i=1}^C \log\left(\frac{e^{\mathbf{s}_p}}{\sum_{j=1}^C e^{\mathbf{s}_j}}\right), \quad (2.22)$$

where  $\mathbf{s}_p$  is the CNN score for the positive class.

This can be extended to the multi-label scenario where the loss contains a value for each positive class. Consider  $N_p$  positive classes of a single sample, then the Cross-entropy loss with Softmax activations can be computed as,

$$CE = - \frac{1}{N_p} \sum_{p=1}^{N_p} \log\left(\frac{e^{\mathbf{s}_p}}{\sum_{j=1}^C e^{\mathbf{s}_j}}\right), \quad (2.23)$$

where  $\frac{1}{N_p}$  is a scaling used to make the loss invariant to positive classes.

For optimisation purposes cross-entropy with softmax can be formalised as,

$$J(\theta) = - \left[ \sum_{i=1}^C \sum_{n=1}^N 1\{\mathbf{y}_{(i)} = n\} \log \rho(y_i = n | \mathbf{x}_i; \theta) \right],$$

where  $N$  is the number of classes,  $1\{\cdot\}$  is an indicator function such that  $1\{true\} = 1$  and  $1\{false\} = 0$ ,  $\rho(\cdot)$  is the posterior probability of class  $\mathbf{y}_i$  given inputs  $\mathbf{x}_i$  and the network parameters  $\theta$ . (2.24)

In other words, the Softmax layer is constructed from unnormalised class log probabilities that are normalised and combined with multinomial loss to give the cross-entropy loss. This approach works well for classes that are distinct and independent from each other. Loss functions play a crucial role in how the neural network optimally adjust model parameters for learning any given task. Further details on the importance of loss in the optimisation of neural networks will be provided in section 2.5.8.

There are different variants of the cross-entropy functions, examples include the *bootstrapped cross-entropy loss* and the *binary cross-entropy* [51].

### 2.5.7 Back-propagation

After all the processes described above, the neural network now needs to adjust or update its trainable parameters in order to reduce the loss. At this stage, the learning problem becomes an optimisation problem with the objective of minimising the loss. Optimisation will be discussed at length in section 2.5.8 below. Back-propagation is a technique from the connectionism era that is used to compute the gradients of the network with respect to the weights or parameters to facilitate gradient-based learning or optimisation [81, 123].

The back-propagation process consists of four stages, the forward pass, the the loss function, the backward pass and the parameter updates.

#### Forward pass

This process involves taking an input image and passing it through all the feature extraction layers in the network. Passing forward all the activations or feature maps until the classification layer is reached the losses can be computed using any of the techniques described in section 2.5.6 above.

#### Backward pass

As mentioned earlier, this process computes the gradient of the loss function at each iteration with respect to the parameters. This requires the chosen loss function to be continuously differentiable at all iteration steps. Every neuron in each layer in a neural network depends on all the activations of all the connected neurons from its preceding layer. Thus the derivatives are computed in a partial-wise manner. The gradient computation is thus a recursive application of the *chain rule of calculus*. The derivatives are calculated for every layer with respect to the weights. This enables the neural network to ascertain the actual contribution the weights had in the learning process. More specifically, the contribution of the weights towards the obtained loss.

Once the gradients are computed, the next step is the optimal update or adjustment of the parameters.

#### Parameter updates

Finally, a gradient-based optimisation technique can be used to update the parameters based on the gradients computed in the backward pass. *Stochastic Gradient Descent* (SGD) [45] is

one of the most commonly used approaches for neural network optimisation. In the section below (2.5.8), we provide an in-depth discussion of the optimisation process.

### 2.5.8 Optimisation

There are different heuristics for choosing the initial weights of a convolutional neural network. Some of the widely used approaches are *random*, *constant*, *uniform*, Xavier [50], and He initialisation [56]. As a result, in the initial stages of training there is hardly any relationship between the input and the groundtruths. The predictions are thus somewhat meaningless and the losses large. The training process is therefore the process of adaptively adjusting the network parameters in such a manner that the loss is minimised. In pure optimisation the ultimate goal is strictly minimising the cost function  $J(\theta)$ . However, neural network optimisation is slightly different. The goal for neural networks is to achieve the best performance measure, that is measured as the generalisation ability of the learning framework. This is measured on the test set but the optimisation is done on the training set with hopes of acquiring generalised performance [51].

Parameter updates in Gradient based learning are done according to an update rule that minimises the cost function. The two common variants of gradient based learning are *Gradient Descent* (GD) and *Stochastic Gradient Descent* (SGD). Gradient Descent is a deterministic and iterative computation of the exact gradients. This involves going through all the training samples to perform a single update for one iteration. The optimisation process is one of the most expensive processes both in terms of memory and computational time. In light of these requirements, Gradient Descent is extremely inefficient.

A more efficient and widely used variant is the Stochastic Gradient Descent. We discuss this in the next section (2.5.8).

#### Stochastic gradient descent

Stochastic Gradient Descent is a non-deterministic approach that computes an approximation of the true gradient. Unlike the Gradient Descent, this method randomises the order of the training data and the computation of the associated gradients. This is achieved by using a subset of the training samples for parameter updates. The mini-batch is drawn under the *independently and identically distributed* (iid) assumption [25]. When the size of the subset is greater than one, the variant is called *mini-batch* Stochastic Gradient Descent. The stochasticity introduced by the randomisation ensures the avoidance of convergence at local

minima. Convergence at local minima is not ideal as it limits the generalisation capability of the learning network. What is really needed is convergence at a global minima.

The update rule is given by,

$$\theta = \theta - \alpha \nabla_{\theta} J(\theta), \quad (2.25)$$

where  $\alpha$  is the learning rate, the step-size required to get to the desired minima. Mini-batch SGD computes a parameter update for every mini-batch of  $m$  samples. The overall SGD process is shown below (Alg. 1).

---

**Algorithm 1** Stochastic Gradient Descent, at the  $k$ -th iteration of training [51]

---

**Require:** learning rate  $\varepsilon_{(k)}$

**Require:** initial parameters  $\theta$

- 1: **while** stopping criterion not met **do**
  - 2: sample mini-batch  $m$  samples from training set  $\{x_{(0)}, \dots, x_{(m)}\}$  and corresponding targets  $\mathbf{y}_{(i)}$
  - 3: compute gradient estimate  $\tilde{g} \leftarrow +\frac{1}{m} \nabla_{\theta} \sum_i J(f(\mathbf{x}_{(i)}; \theta), \mathbf{y}_{(i)})$
  - 4: update parameters:  $\theta \leftarrow \theta - \varepsilon \tilde{g}$
  - 5: **end while**
- 

Furthermore, different learning rate values contribute differently to the optimisation process. Smaller learning rate values tend to make the learning process much slower whereas larger learning rates risk overshooting and missing the desired global minima. It is common practice to have a fixed learning rate for the entire training. However performance gains have been reported from using different learning rate scheduling heuristics. Further details on this are provided in section 2.5.9 ahead.

The partitioning of the training set into mini-batches accommodates using larger learning rates and speeding up of the optimisation process. Computation time per update does not grow with the number of training samples when using SGD. Additionally, the partitioning of the training set has many benefits but could potentially compromise the learning process if not handled carefully. One of the most efficient ways of handling mini-batches is *batch normalisation* [60, 126]. This is covered in the next section.

### Batch-Normalisation

Batch-normalisation [60] was inspired by the need to improve performance while boosting stability. During the optimisation process in deep networks, the gradient advises how each parameter ought to be updated. This based on the assumption that other layers do not change

during this update. Practically, the updates occur simultaneously. Neural networks comprise of compositions of several layers and functions. This could cause instabilities when functions that were assumed to remain constant are composed with functions that are being updated [51].

Furthermore, these updates during the optimisation process imply that the distribution of each layer's inputs changes as the parameters of the previous layers change. This requires careful rescheduling of learning rates and careful initialisation of the parameters. In light of this, the training of networks with saturating nonlinearities becomes extremely difficult. This challenge is referred to as the *internal covariance shift* [60].

To mitigate internal covariance shift Ioffe et al. [60] proposed layer-wise normalisation of inputs. This also requires the normalisation of each mini-batch of training samples. Batch-normalisation has been reported to act as a regulariser and in some cases it serves as a replacement for Dropout. It allows for the usage of larger learning rates without having to be extremely careful about parameter initialisation. This process has been widely adopted by many deep learning practitioners and it has shown great performance improvements with regards to both computation times and accuracy.

### 2.5.9 Learning rate scheduling and gradient descent optimisation

The standard SGD estimation process is susceptible to noise introduced by the random sampling of training examples. If unchecked this noise may not vanish even after the desired minima is reached. This can compromise the optimisation process and subsequently the performance of the learning network. One remedy for this is the gradual reduction (decaying) of the learning rate as the training progresses. The condition for the convergence of SGD is that,

$$\begin{aligned} \sum_{k=1}^{\infty} \varepsilon_k &= \infty, \text{ and} \\ \sum_{k=1}^{\infty} \varepsilon_k^2 &< \infty. \end{aligned} \tag{2.26}$$

A common approach for learning rate linear decay of the learning rate until iteration  $\tau$  is,

$$\varepsilon_k = (1 - \alpha)\varepsilon_0 + \alpha\varepsilon_\tau, \tag{2.27}$$

where  $\alpha = \frac{k}{\tau}$ ,  $\varepsilon$  is usually left constant after iteration  $\tau$ . In this linear schedule, the critical parameters are  $\varepsilon_0$ ,  $\varepsilon_\tau$  and  $\tau$ .  $\tau$  can be set to equal any reasonable (a few hundreds for large

datasets) number of epochs, and  $\varepsilon_\tau$  is usually chosen at 0.001 of the  $\varepsilon_\tau$ . The challenge is choosing the initial learning rate  $\varepsilon_0$ . Too large a value will risk overshooting, and too small values result in slow convergence [51].

The area of learning rate scheduling and convergence for deep neural networks is currently understudied. However, it is common practice to measure the *excess error*  $J(\theta) - \min_{\theta} J(\theta)$ , that is, how much the current error exceeds the possible minimum error.

In the next section we discuss some of the currently widely used SGD learning rate scheduling heuristics.

### Momentum

Inspired by Newton's Laws of Motion, here the negative gradient is regarded as a particle flowing through a parameter space. *Momentum* [119] is mass times velocity. For optimisation purposes the unit mass is used. The momentum approach accumulates an exponentially decreasing running average of previous gradients, continuously moving in a general direction [51]. This is achieved by introducing a variable  $\gamma$  to inculcate the notion of velocity. This is done as follows,

---

**Algorithm 2** Stochastic Gradient Descent with Momentum, at the  $k$ -th iteration of training [51]

---

**Require:** learning rate  $\varepsilon_{(k)}$ , momentum parameter  $\alpha$

**Require:** initial parameters  $\theta$ , initial velocity  $\gamma$

**while** stopping criterion not met **do**

        2: sample mini-batch  $m$  samples from training set  $\{x_{(0)}, \dots, x_{(m)}\}$  and corresponding targets  $\mathbf{y}_{(i)}$

        compute gradient estimate  $\tilde{g} \leftarrow +\frac{1}{m} \nabla_{\theta} \sum_i J(f(\mathbf{x}_{(i)}; \theta), \mathbf{y}_{(i)})$

        4: compute velocity update:  $\gamma \leftarrow \alpha\gamma - \varepsilon\tilde{g}$

        apply update:  $\theta \leftarrow \theta + \gamma$

        6: **end while**

---

### Nesterov accelerated gradient descent

Nesterov Accelerated Gradient Descent [106, 133] is a variant of the momentum technique. This disregards the gradient direction, instead it computes the gradient after the velocity has been applied. This attempts to approximate the next position of the parameters. For convex SGD problems, Nesterov momentum changes the excess error rate from  $\mathbf{O}(\frac{1}{k})$  to  $\mathbf{O}(\frac{1}{k^2})$ . This



method does not improve the rate of convergence [51].

There is a family of methods that attempt to adapt the learning rates of model parameters. These methods try to combat the need for introducing more hyperparameters to the neural network, like in the momentum case. We provide a brief discussion on these methods in the sections below.

### AdaGrad

AdaGrad [32] technique does an individual revision of the learning rates of all model parameters. This is achieved by inversely scaling the learning rates proportional to the square root of the summation of all their previous squared gradients. Precisely, parameters with larger gradients of the loss get a corresponding large decrease in their learning rates, and parameters with small gradients are updated with a corresponding small decrease in their learning rate [51]. For convex settings, this helps with a steady descent in a general direction towards the desired minima.

---

**Algorithm 3** Stochastic Gradient Descent with AdaGrad, at the  $k$ -th iteration of training [51]

---

**Require:** global learning rate  $\varepsilon$

**Require:** initial parameter  $\theta$

**Require:** small constant  $\delta$  for numerical stability, (say  $1e-7$ )

    initialise gradient accumulation variable  $r = 0$

**while** stopping criterion not met **do**

        3: sample mini-batch  $m$  samples from training set  $\{x_{(0)}, \dots, x_{(m)}\}$  and corresponding targets  $\mathbf{y}_{(i)}$

        compute gradient  $g \leftarrow +\frac{1}{m} \nabla_{\theta} \sum_i J(f(\mathbf{x}_{(i)}; \theta), \mathbf{y}_{(i)})$

        accumulate squared gradient:  $r \leftarrow r + g \cdot g$

        6: compute update:  $\Delta\theta \leftarrow -\frac{\varepsilon}{\delta + \sqrt{r}} \cdot g$  (element-wise division and square root)

        apply update:  $\theta \leftarrow \theta + \Delta\theta$

**end while**

---

The accrued squared gradients may prematurely and excessively decrease the effective learning rate leading to sub-optimal performance.

### RMSProp

AdaGrad is meant to accelerate convergence when applied to convex functions. The RMSProp algorithm [139] seeks to improve AdaGrad in the non-convex case by using an exponentially weighted moving average instead of the accumulated gradients. This helps

mitigate instabilities caused by extremes in the accumulated gradients . RMSProp introduces a new hyperparameter  $\rho$  to moderate the moving average.

---

**Algorithm 4** Stochastic Gradient Descent with RMSProp, at the  $k$ -th iteration of training [51]

---

**Require:** global learning rate  $\epsilon$ , decay rate  $\rho$

**Require:** initial parameter  $\theta$

**Require:** small constant  $\delta$  for numerical stability , (say  $1e-7$ )

    initialise gradient accumulation variable  $r = 0$

**while** stopping criterion not met **do**

        sample mini-batch  $m$  samples from training set  $\{x_{(0)}, \dots, x_{(m)}\}$  and corresponding targets  $\mathbf{y}_{(i)}$

4: compute gradient  $g \leftarrow +\frac{1}{m} \nabla_{\theta} \sum_i J(f(\mathbf{x}_{(i)}; \theta), \mathbf{y}_{(i)})$

        accumulate squared gradient:  $r \leftarrow \rho r + (1 - \rho) g \cdot g$

        compute update:  $\Delta\theta \leftarrow -\frac{\epsilon}{\sqrt{\delta+r}} \cdot g$  (element-wise division and square root)

        apply update:  $\theta \leftarrow \theta + \Delta\theta$

8: **end while**

---

RMSProp has shown great success and is widely used by most deep learning practitioners. AdaDelta [152] is also another modification of AdaGrad. It attempts to solve the problems caused by the accumulated gradients by enforcing a specific fixed range for these accumulated gradients.

## Adam

This is yet another adaptive learning rate scheduling technique, dubbed "Adam", inspired by *adaptive moments* [69]. It has two main contributions. The first one is the direct incorporation and combination of momentum as an estimate of the first order moment of the gradient into the notion of exponential weighting. Second is the inclusion of *bias corrections* in the momentum term and second order moments as a proxy for the original initialisation [51, 69]. The method is shown in detail in Algorithm 5 below.

Adam has been widely adopted and is currently the state-of-the-art optimiser. The recommended defaults may need to be changed depending on task at hand and initialisation scheme.

### 2.5.10 Hyperparameters

Additional to optimising the learning process using gradient based methods, there are other aspects of optimisation. Hyperparameters are network settings that need to be defined before

**Algorithm 5** Stochastic Gradient Descent with Adam, at the  $k$ -th iteration of training [51]

---

**Require:** learning rate  $\varepsilon$  (recommended default: 0.001)  
**Require:** exponential decay rates for moment estimates  $\rho_1$  and  $\rho_2 \in [0, 1]$  (recommended defaults: 0.9 and 0.999 respectively)  
**Require:** small constant for  $\delta$  for numerical stabilisation (recommended default:  $1e-8$ )  
**Require:** initial parameters  $\theta$ , initial velocity  $\gamma$   
 initialise 1st and 2nd moment variables  $s = 0, r = 0$   
 initialise time step  $t = 0$   
**while** stopping criterion not met **do**  
 sample mini-batch  $m$  samples from training set  $\{x_{(0)}, \dots, x_{(m)}\}$  and corresponding targets  $\mathbf{y}_{(i)}$   
 5: compute gradient estimate  $\tilde{g} \leftarrow +\frac{1}{m} \nabla_{\theta} \sum_i J(f(\mathbf{x}_{(i)}; \theta), \mathbf{y}_{(i)})$   
 $t \leftarrow t + 1$   
 update biased first moment estimate:  $s \leftarrow \rho_1 s + (1 - \rho_1) \tilde{g}$   
 update biased second moment estimate:  $r \leftarrow \rho_2 r + (1 - \rho_2) \tilde{g} \cdot \tilde{g}$   
 correct bias in first moment:  $\hat{s} \leftarrow \frac{s}{1 - \rho_1^t}$   
 10: correct bias in second moment:  $\hat{r} \leftarrow \frac{r}{1 - \rho_2^t}$   
 compute update  $\Delta\theta = -\varepsilon \frac{\hat{s}}{\sqrt{\hat{r} + \delta}}$  (element-wise operations)  
 apply update:  $\theta \leftarrow \theta + \Delta\theta$   
**end while**

---

the training process, unlike normal learnable parameters whose values have to be learned during training. These comprise of weight initialisation choice, learning rate, the optimisation technique, regularisation heuristics, mini-batch sizes, number of epochs, the kernel size for convolution, function choice for non-linearisation, number of hidden layers, the loss function, and many other possibilities depending on the task at hand. These are factors that influence the overall behaviour of the learning framework. They affect the quality of the learning process as well as the generalisation ability of the network. Yet there is no deterministic way of setting them. There is no one-size fits all method for hyperparameter selection. There only exists general guidelines.

The choice of hyperparameters is influenced by different factors. However, the main factors are the nature of the learning task and the training data, the time and space complexity implications on the overall learning process and the deep learning practitioner's experience. There are two basic paradigms for hyperparameter tuning, either manual or automatic. Automatic hyperparameter tuning is beyond the scope of this work.

The manual hyperparameter tuning paradigm generally involves closely monitoring the training process for known performance indicators. The objective of manual hyperparameter tuning is to adjust the *effective capacity* of the model. Effective capacity is constrained by 1.) the representational capacity of the model, 2.) ability to successfully minimise the cost function, and 3.) the extent to which the model is regularised [51]. The choice of hyperparameters has immediate impact on whether the model overfits or underfits. This is mainly done by observing the loss function. The loss function is tracked over batches and epochs. An epoch is a complete pass (both forward and backwards) of the entire dataset through the network. The effects of certain hyperparameters like learning rate and batch size can be easily noticed and adjusted manually. The learning rate is the most important factor in manual hyperparameter selection.

Overall, manual hyperparameter selection is not a trivial task, it requires profound theoretical knowledge and technical experience. It is indeed a great challenge to choose the right configuration with such large numbers of different and distinct yet mutually connected settings.

### 2.5.11 Network initialisation and transfer learning

As explained in the theory section (2.5) above, the learning process is fundamentally a search for the best parameters that describe the best representation of the given inputs. There are different heuristics for initialising the weights of CNN. One commonly adopted technique is random initialisation where the weights are sampled from a random distribution. Weight initialisation is of paramount importance since it determines how the gradients propagate through the entire network. Weights need not be too small nor too large lest the sought global minima will never be achieved. This jeopardises the learning process and negatively impacts results.

Several other techniques have been proposed as better alternatives to random initialisation. Xavier [50] initialisation is based on weights that are drawn from a truncated normal distribution centred on zero with a specified standard deviation and specific variance. He et al. [55] proposed a modified version of the Xavier initialisation. The main difference between Xavier and He initialisation schemes is that the latter addresses rectifier non-linearities. Furthermore, Xavier initialisation is mostly compatible with the hyperbolic tangent nonlinearity rectifier whereas He initialisation is most suitable when using the ReLUs [50, 55]. Other methods include LeCun initialisation [70], zeros, ones and many others.

Moreover, Bengio and Yosinski [150] found that in vision tasks CNNs exhibit similar behaviour in the first layer. They all learn basic edges and colour blobs. This is independent of the initialisation scheme used. Thus, in the first layer the learning process is specific to neither any task nor any particular dataset. Feature maps only transition from general to specific towards the last convolutional layer of the network. This meant there existed a transferability of features from one task to other related but distant tasks. Bengio and Yosinski [150] went on to prove that the transfer of features from networks trained on slightly unrelated tasks was often better than random initialisation. Additionally, effective feature transferability was found to be based on two main factors. First, the layers from which the transferred features are selected. Higher layer neurons tend to specialise more towards the original layer negatively impacting performance on the target task. Second, cross-task adaptation of neurons caused optimisation challenges. The process of transferring features from one task to related tasks is called **transfer learning**.

Formally, transfer learning is when the first  $n$  layers are transferred from a base network (and a base task) and used to initialise a target network (for a related target task). Cross-task feature transfer has been widely adopted as an alternative to training from scratch. Transfer learning also, to an extent, serves as a solution to the problem of limited data. Fine-tuning [150] is a technique that was designed for optimal transfer learning. Widely adopted fine-tuning techniques include learning scheduling and saving the weights from the base network as a pre-trained network for use in related and distant tasks. The commonly used architectures and datasets for transfer learning are given in the sections below.

## 2.6 Summary

In this chapter we started with traditional semantic segmentation approaches. We then gave a detailed account of the progression from traditional methods to ANNs. We provided details on the inspiration behind ANNs as well as the mathematical formalisation of ANNs. Furthermore, we discussed various approaches of learning with ANNs such as feature based ANNs and CNNs. We also discussed the theoretical building blocks of CNNs and semantic image parsing. In the next chapter we provided an exhaustive literature review on state-of-the-art CNN architectures for semantic segmentation as well as the associated datasets.



# Chapter 3

## Literature Review

### 3.1 Introduction

In this section we provide brief descriptions of various architectures. First, we then discuss some of the main works that paved way for most current state-of-the-art segmentation architectures. Additionally, we provide exhaustive reviews of the current state-of-the-art with regards to their contributions and their shortcomings. Lastly, we provide a brief overview of the widely used datasets for segmentation.

### 3.2 CNN architectures

Yann Le Cun et. al was the first successful CNN with the **LeNet-5** architecture [79]. This was inspired by the need to address the dilemma between small networks and large networks. Small networks were unable to learn and large networks suffered over-parametrisation. LeNet-5 was the first paper to design specialised networks for recognising two-dimensional shapes such as images [79]. This produced striking results on the MNIST handwritten zip codes problem.

**ImageNet Large Scale Visual Recognition Competition (ILSVRC)** [124] is a platform for evaluating object detection/localisation and classification algorithms on images and videos. It is one of the most popular platforms for measuring global research progress in computer vision. ImageNet also provides a very comprehensive dataset (ImageNet dataset) which contains over 1000 categories and millions of images.

After almost over a decade of minimal to no success from CNNs and domination by classical vision approaches in the ILSVRC challenge Krizhevsky et al. [75] emerged with another successful implementation of CNNs. This was named the **AlexNet** architecture. AlexNet is very similar to LeNet-5 except it was much deeper and bigger, it also used the ReLU non-linearity instead of the hyperbolic tangent. AlexNet won the competition with a test accuracy of 84.6% while the second best only achieved a test accuracy of 73.8%. This architecture consisted of 5 convolutional layers, 5 max-pooling layers, 5 ReLUs and 3 fully connected layers.

Following the astounding success of AlexNet, vision research efforts started being directed towards the components of CNNs. Mostly receptive fields (kernel sizes), stride, multiple scales, and dense training. The University of Oxford vision lab, Visual Geometry Group (VGG) [130] was the first to focus strictly on the depth of a network. Investigations were conducted by adding more layers while fixing all other parameters. This architecture was named after the lab it came from, **VGG** [130]. It beat state-of-the-art during the 2015 version of the ILSVRC challenge on the ImageNet dataset.

The VGG architecture comprises of, first, random initialisation of weights, stacks of convolutional layers with  $(3 \times 3)$  kernels/filters,  $(1 \times 1)$  filters for linear transformation of input channels followed by non-linearisation, one pixel stride, one pixel padding. This also included 5 max-pooling layers with  $2 \times 2$  pooling windows and stride of 2 pixels. The convolution layers were followed by 3 fully connected layers, the first two having 4096 channels each and the third performed 1000-way classification, thus 1000 channels. The final layer was the SoftMax layer. It is important to note that unlike AlexNet, VGG did not use Local Response Normalisation (LRN) as prior research had shown it to not improve performance on ImageNet but only increased memory and computation footprint [130].

VGG experiments involved different variations, configurations had the generic design described above, they only varied in depth,

After the success of increased network depth by VGG, Microsoft research [56] questioned if learning better networks was as easy as stacking more layers? They argued that the convergence of deeper networks gives rise to the degradation problem. A challenge inherent in depth increments, as the depth increases accuracy saturates then degrades rapidly. Furthermore, they noted that the degradation was not necessarily a result of overfitting but

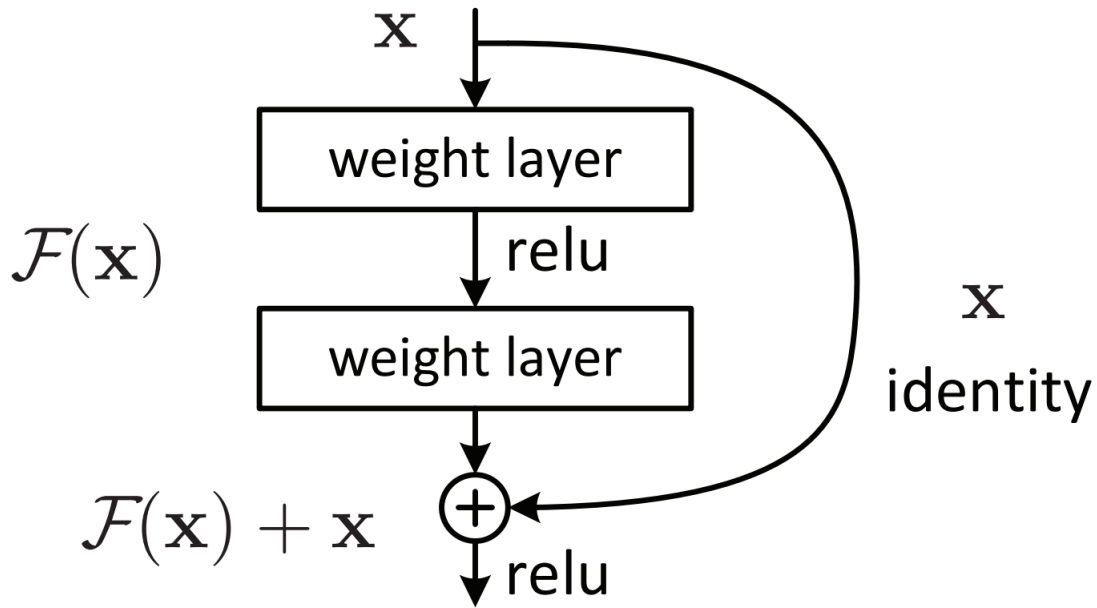


| ConvNet Configuration               |                        |                               |                                            |                                            |                                                         |
|-------------------------------------|------------------------|-------------------------------|--------------------------------------------|--------------------------------------------|---------------------------------------------------------|
| A                                   | A-LRN                  | B                             | C                                          | D                                          | E                                                       |
| 11 weight layers                    | 11 weight layers       | 13 weight layers              | 16 weight layers                           | 16 weight layers                           | 19 weight layers                                        |
| input ( $224 \times 224$ RGB image) |                        |                               |                                            |                                            |                                                         |
| conv3-64                            | conv3-64<br><b>LRN</b> | conv3-64<br><b>conv3-64</b>   | conv3-64<br>conv3-64                       | conv3-64<br>conv3-64                       | conv3-64<br>conv3-64                                    |
| maxpool                             |                        |                               |                                            |                                            |                                                         |
| conv3-128                           | conv3-128              | conv3-128<br><b>conv3-128</b> | conv3-128<br>conv3-128                     | conv3-128<br>conv3-128                     | conv3-128<br>conv3-128                                  |
| maxpool                             |                        |                               |                                            |                                            |                                                         |
| conv3-256<br>conv3-256              | conv3-256<br>conv3-256 | conv3-256<br>conv3-256        | conv3-256<br>conv3-256<br><b>conv1-256</b> | conv3-256<br>conv3-256<br><b>conv3-256</b> | conv3-256<br>conv3-256<br>conv3-256<br><b>conv3-256</b> |
| maxpool                             |                        |                               |                                            |                                            |                                                         |
| conv3-512<br>conv3-512              | conv3-512<br>conv3-512 | conv3-512<br>conv3-512        | conv3-512<br>conv3-512<br><b>conv1-512</b> | conv3-512<br>conv3-512<br><b>conv3-512</b> | conv3-512<br>conv3-512<br>conv3-512<br><b>conv3-512</b> |
| maxpool                             |                        |                               |                                            |                                            |                                                         |
| conv3-512<br>conv3-512              | conv3-512<br>conv3-512 | conv3-512<br>conv3-512        | conv3-512<br>conv3-512<br><b>conv1-512</b> | conv3-512<br>conv3-512<br><b>conv3-512</b> | conv3-512<br>conv3-512<br>conv3-512<br><b>conv3-512</b> |
| maxpool                             |                        |                               |                                            |                                            |                                                         |
| FC-4096                             |                        |                               |                                            |                                            |                                                         |
| FC-4096                             |                        |                               |                                            |                                            |                                                         |
| FC-1000                             |                        |                               |                                            |                                            |                                                         |
| soft-max                            |                        |                               |                                            |                                            |                                                         |

**Figure 3.1** VGG Architecture variants. VGG has various models and configurations. These various models are usually named with a number suffix which denotes the number of weight layers in the model. VGG-16 (block D), submitted at the 2013 ILSVRC achieved a TOP-5 test accuracy of 92.7% on the ImageNet dataset. LRN is the local region normalisation, maxpool is short for max-pooling and FC stands for the Fully-connected layer. Source [130]

rather an indication of more complex optimisation needed for deeper networks. Without this optimisation, naively stacking more layers would lead to higher training error. As a solution they proposed what is now widely known as the **ResNet** architecture [56].

The intuition behind ResNet is as follows: given a shallower network and its deeper counterpart (one that adds more layers), there exists a solutions by construction of the deeper model, which means additional layers are identity mappings, then rest of the layers are copied



**Figure 3.2** Residual Learning Building Block

from the shallower model, then the deeper networks are supposed to perform better than its shallower network in terms of the training error [56].

ResNet's contribution was thus to address the problem degradation problem using this intuition for deep residual learning [56]. The identity mappings in ResNet are also referred to as *skip connections*. The ResNet architecture managed to increase network depth even further to as many as 152 while simultaneously reducing the number of parameters. With these advancements ResNet won first place in the 2015 ImageNet and CIFAR [74] datasets challenge reaching top-1 and top-5 error rates [56].

The ongoing success of deep neural networks became widely accepted and applications that needed these solutions also increased. Most of them were mobile and embedded systems that need real-time solutions. Some research efforts began to focus on the development of more computationally efficient models in terms the training time, memory footprint and accuracy gains. The **InceptionV1** architecture by Google researchers Szegedy et. al [134] focused on the efficient utilisation of computational resources by increasing network depth and with while keeping the computational budget constant. They borrowed ideas from Hebb's postulate [58] which when adapted for ANNs conjectures that if the probability distribution of the dataset is representable by a large, very sparse deep neural network, then the optimal

network topology can be constructed layer by layer via analysis of the correlation statistics of the activations of the last layer and clustering neurons with highly correlated outputs. This can be translated in simple terms, neurons that fire together, wire together [58].

InceptionV1 noted the inherent sparsity problem in feature distributions against current hardware architectures being mostly feasible for dense operations. Furthermore, this name was inspired by Network-In-Network (NINs) [86]. To leverage sparsity for efficiency InceptionV4 proposed feature extraction with multiple different sized kernels ( $(3 \times 3)$  and  $(5 \times 5)$ ) then a  $(1 \times 1)$  convolution for dimensionality reduction. The dimension reduction idea was adapted from NINs [86]. With these modifications, InceptionV1 was able to beat state-of-the-art in most classification and detection tasks.

In a booming era of CNNs, transfer learning also gained traction. Performance gains on one task tends to transfer significantly towards performance gains to a wide variety of application domains. Though InceptionV1 showed great success, it was difficult to adapt it and modify it to new use-cases. As a result, **InceptionV2** and **InceptionV3** [135] was developed with factorised convolutions and aggressive regularisation. It also included new general principles and optimisation ideas for efficiently scaling up CNNs.

ResNets had similar performance with InceptionV3, yet they were using different approaches. **InceptionV4** [136] was a further progression that leveraged the skip-connection approach from ResNets. This provided empirical evidence that there were major benefits from combining the two approaches. InceptionV4 accelerated training time significantly and also improved model performance reaching top-5, 3.08% error rate on ImageNet.

As CNN solutions became ubiquitous in computer vision, general research efforts leaned towards deeper and complicated networks in pursuit of improved performance. However, these networks still lacked efficiency in terms of model size and speed. Model size is measured by the number of parameters and memory requirements for a given model.

Size and speed are crucial factors for applicability in real world problems such as robotics, self-driving cars, augmented reality and so forth. **MobileNets** [59] were developed for mobile and embedded vision systems. MobileNets focused on streamlining architectures using *depth-wise separable* convolutions to build light-weight networks. This approach introduced two new global hyper-parameters for efficiently trading off between latency and accuracy. These hyperparameters are the width multiplier and resolution multi-

plier, they both allow the model builder to choose the most efficient size of the model for applications based on the constraints of the problem [59]. The objective being smaller and faster networks. MobileNets demonstrated superiority over state-of-the-art with regards to size, speed and accuracy characteristics in a multitude of tasks [59]. MobileNetV2 [125] is an improvement on MobileNets. It is inspired by the inverted residual structure from DeepLabV3 [22]. Shortcut connections are between thin bottleneck layers, intermediate expansion layers use lightweight convolutions as a source of non-linearity. The MobileNetV2 framework showed improved results from the removal of non-linearities from narrow layers. It also provided a convenient framework for further analysis and adaptation.

In the section we reviewed methods that have paved way for most of the leading CNN architectures at the moment. The models were mostly used for object detection and classification. However, their feature extraction processes and the acquired are transferable to other task domains. One example is pixel-wise classification. Part of the objectives for this research is to compare is to use and compare weights obtained from these architectures on other tasks as pre-trained weights for our task of semantic segmentation on our dataset. Further details will be discussed in the methodology and results section.

In the next section we review some of the leading architectures that are directly involved in semantic segmentation.

### 3.2.1 State-Of-The-Art architectures for semantic segmentation

Computer vision research efforts have increased and become popular in light of the relentless success of CNNs in object detection and classification problems. There is a plethora of new state-of-the-art publications being released frequently. The table below (Table (3.1)) is a condensed account of what was state-of-the-art deep learning methods for semantic segmentation at the end of 2016. It is worthwhile noting that most these works where for pixel-level classification of objects, indoor and city structured environments, and not necessarily for unstructured domains. Most of the core segmentation principles are however applicable and adaptable across domains. We provide reviews of some of the forerunners and recent leading works in the next paragraphs.

**Fully Convolutional Networks (FCN)** by Long et al. [92] were the pioneers at noting that powerful CNNs designed for classification could be re-purposed for semantic segmentation. Their milestone was leveraging already successful classification architectures such an AlexNet [75], VGG [130], InceptionV1 [134], and ResNets [56] in learning feature hierar-

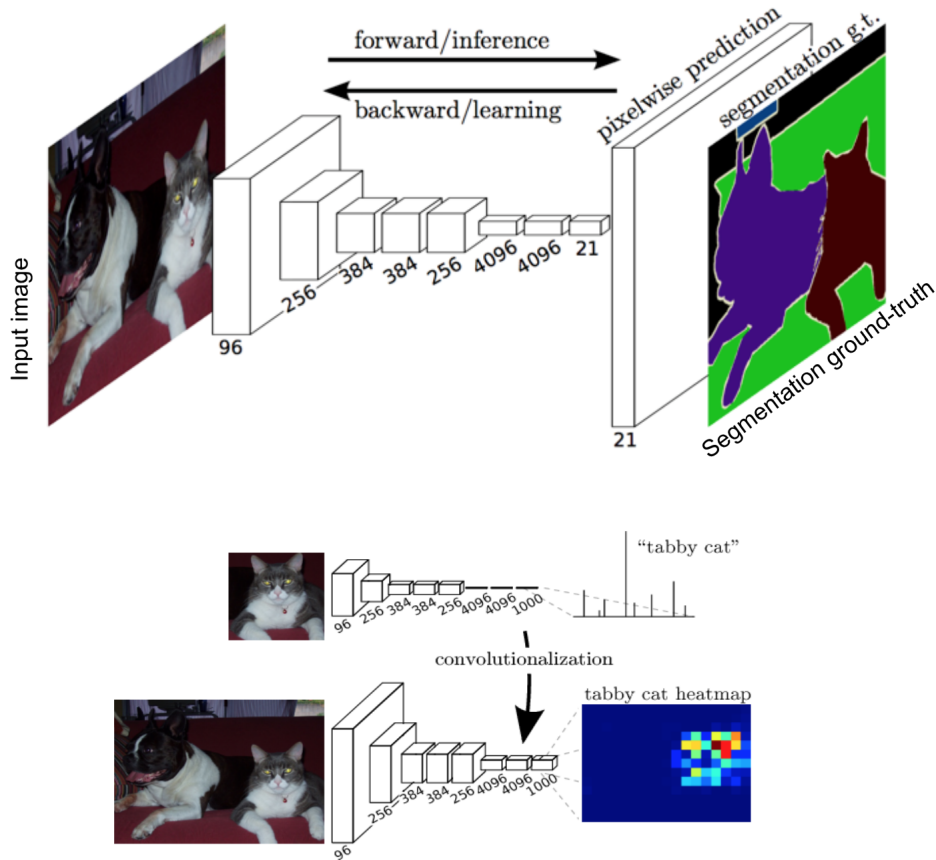
|    | Name and Reference               | Architecture          | Contribution(s)                                        | Accuracy (IoU) |
|----|----------------------------------|-----------------------|--------------------------------------------------------|----------------|
| 1  | Fully Convolutional Network [92] | VGG-16 (FCN)          | Forerunner                                             | *              |
| 2  | SegNet [6]                       | VGG-16 + Decoder      | Encoder-decoder                                        | ***            |
| 3  | Bayesian SegNet                  | SegNet                | Uncertainty Modelling                                  | ***            |
| 4  | DeepLab                          | VGG-16/ResNet-101     | Standalone CRF, atrous convolutions                    | ***            |
| 5  | MINC-CNN                         | GoogLeNet (FCN)       | Patchwise CNN, Standalone CRF                          | *              |
| 6  | CRFasRNN                         | FCN-8s                | CRF reformulated as RNN                                | *              |
| 7  | Dilation                         | VGG-16                | Dilated Convolutions                                   | ***            |
| 8  | ENet                             | ENet Bottleneck       | Bottleneck module for efficiency                       | **             |
| 9  | Multi-Scale CNN-Raj              | VGG-16 (FCN)          | Multi-scale architecture                               | ***            |
| 10 | Multi-Scale-CNN-Eigen            | Custom                | Multi-scale Sequential Refinement                      | ***            |
| 11 | Muiti-Scale-CNN-Roy              | Multi-Scale-CNN-Eigen | Multi-scale Coarse-to-fine refinement                  | ***            |
| 12 | Multi-Scale-CNN-Bian             | FCN                   | Independently trained multi-scale FCNs                 | **             |
| 13 | ParseNet                         | VGG-16                | Global Context Feature Fusion                          | ***            |
| 14 | ReSeg                            | VGG-16 + ReNet        | Extension of ReNet to semantic segmentation            | **             |
| 15 | LSTM-CF                          | Fast R-CNN + DeepMask | Fusion of contextual information from multiple sources | ***            |
| 16 | 2D-LSTM                          | MDRNN                 | Image Context Modelling                                | **             |
| 17 | rCNN                             | MDRNN                 | Different Input Sizes, image context                   | **             |
| 18 | DAG-RNN                          | Elman Network         | Graph image structure for context modelling            | ***            |
| 19 | SDS                              | R-CNN + Box CNN       | Simultaneous detection and segmentation                | ***            |
| 20 | DeepMask                         | VGG-A                 | Proposal Generation for Segmentation                   | ***            |
| 21 | SharpMask                        | DeepMask              | Top-down refinement module                             | ***            |
| 22 | MultiPathNet                     | Fast R-CNN + DeepMask | Multi-path information flow through network            | ***            |
| 23 | Huang-3DCNN                      | Own 3DCNN             | 3DCNN for voxelized point clouds                       | *              |
| 24 | PointNet                         | Own MLP based         | Segmentation of un-ordered point sets                  | **             |
| 25 | ClockWork Covnet                 | FCN                   | Clockwork scheduling for sequences                     | **             |
| 26 | 3DCNN-Zhang                      | Own 3DCNN             | 3D convolutions and Graph cut for sequences            | **             |
| 27 | End2End Vox2Vox                  | C3D                   | 3D convolutions/deconvolutions for sequences           | **             |
| 28 | FRRN                             | ResNet + Own          | two-stream network, full resolution segmentation       | **             |

**Table 3.1** State-Of-The-Art CNN Architectures for semantic segmentation in 2016. The stars represent the rating of IoUs on a 5-point scale. Source [44]

chies. The fully connected layers were then replaced by convolutional layers to produce spatial heat-maps instead of classification scores. Further upsampling and transposed convolution produced dense pixel-level segmentation [92]. This work was the first to show that CNNs could be trained end-to-end for semantic segmentation tasks with arbitrary input sizes. This is shown in Fig. 3.3.

FCNs became state-of-the-art reigning supreme over the leading traditional segmentation methods at the time. However, despite all the success FCNs still had room for improvement. It could further be adapted for various other domains, computational efficiency and accuracy could also be improved. This sparked a new wave of new methods as shown in table 3.1.

Most leading approaches in semantic segmentation are inspired by FCNs. The majority of state-of-the-art works in semantic segmentation comprise of three main phases. The first is the **encoder** part, this is a feature extraction pipeline that uses a series of convolution and pooling (downsampling) to extract coarse semantic features. The second phase is the **decoder**, this uses a series of strided convolution to recover the image at full resolution. The third phase, **refinement**, is considered an optional stage in some works. While some approaches use the feature maps from the decoder phase for the final pixel classification, some

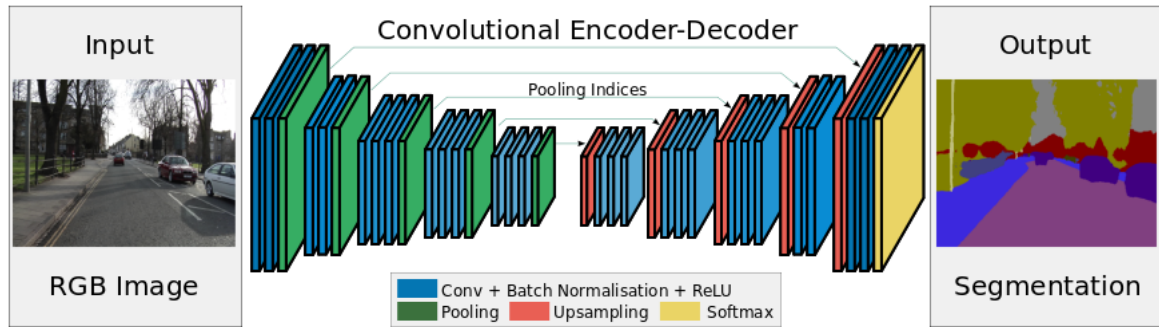


**Figure 3.3** Fully Convolutional Networks by Long et al. [92]. FCNs can be trained to make dense pixel-level classifications. This is achieved by transforming fully connected layers into convolution layers to produce heat maps from the feature maps. Upsampling and transposed convolution (fractionally strided convolution) gives the desired segmentation output.

works find it worthwhile to use post-processing to refine the segmentation predictions. There are multiple ways in which various works implement these three phases.

**SegNet** [5] is an encoder-decoder architecture motivated by the need to map low-resolution features to input for pixel-level classification. Furthermore, SegNet noted that most of the prior works that had attempted direct adoption of deep classification architectures for semantic segmentation either produced coarse segmentation results, grappled with boundary delineation and were generally not efficient in terms of memory and inference time [5, 92]. The most notable contribution of SegNet was the use of pooling indices in the decoder phase of the network. Unlike all the prior works that employed learned upsampling, SegNet computed pooling indices during the max-pooling operations in the encoder phase. These indices were then used to perform non-linear upsampling. The upsampled sparse activation

maps were finally convolved with trainable kernels to produce dense feature maps [5]. Just like FCNs, all SegNet layers are fully convolutional. SegNet was able to achieve more computational efficiency as well as class boundary localisation. For example, adding SegNet modifications to the VGG architecture reduces the number of trainable parameters from 134 million to 14.7 million [5]. The challenge, however, was trading off accuracy for efficiency.

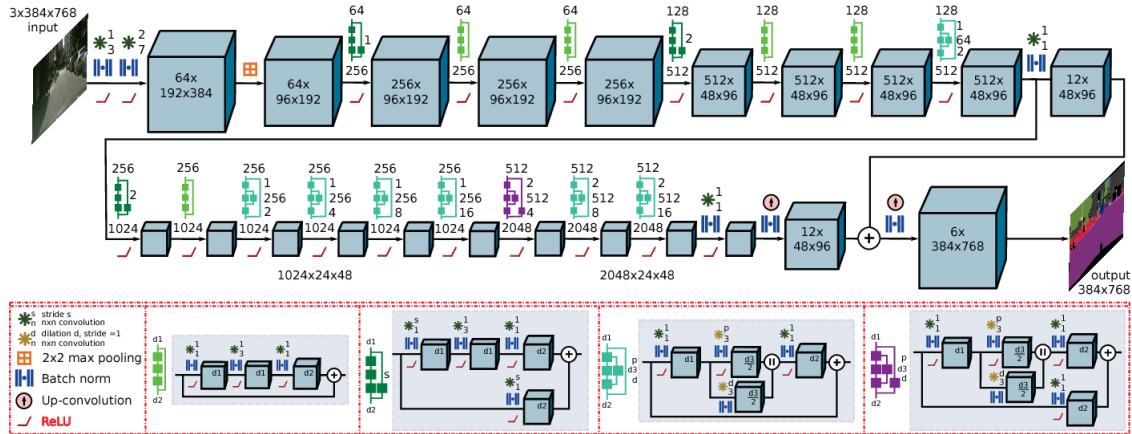


**Figure 3.4** SegNet Architecture. This built on the VGG backbone, except that it uses pooling indices for upsampling, unlike FCNs. SegNet is also fully convolutional just like the FCNs. Source [5]

With rising success of CNNs in semantic segmentation tasks, most architectures lacked robustness and were only suitable for specific environments. **AdapNet** [142] sought to increase robustness to robotic perception in outdoor environments. This was done by introducing the Convolved Mixture of Deep Experts (CMoDE), a fusion scheme for learning robust kernels from complimentary modalities and spectra [142]. Individual expert networks specialise in particular sub-spaces their associated features, CMoDE adaptively weights class-specific features of expert networks using learned probability distributions. This produced state-of-the-art robustness in pixel-wise segmentation and was tested for perception in autonomous navigation. The challenge however, this architecture was still not suitable for real-time semantic segmentation and still struggled with boundaries in adverse environments [118, 142].

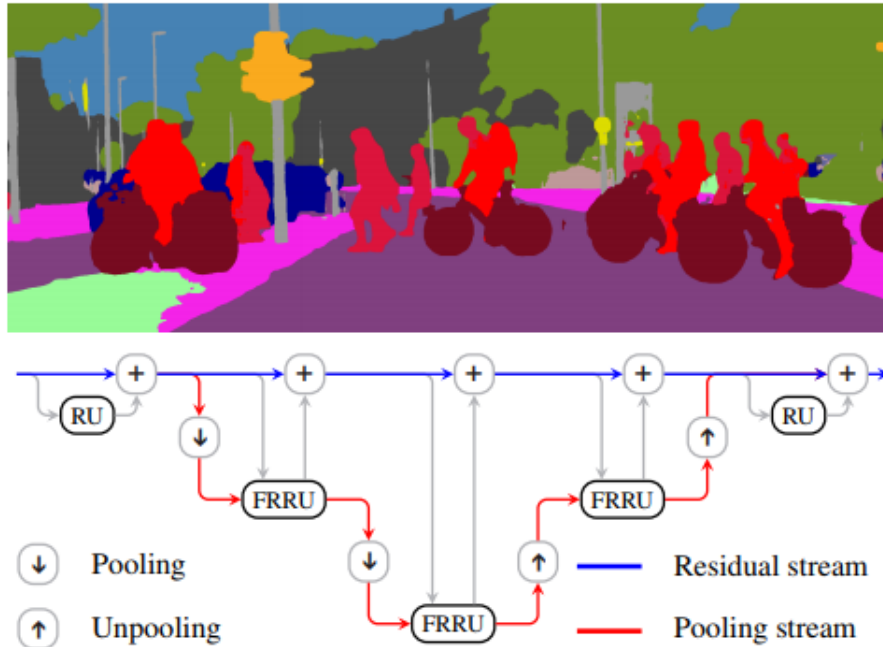
The general trend for most classification CNNs that have been re-purposed for semantic segmentation is to use pre-trained networks. This and the other shortcomings mentioned above found most these approaches lacking in terms of localisation and boundary adherence. Full-Resolution Residual Networks (**FRRN**) [118] addressed these challenges by using a modification of ResNet. This modification has two processing streams. One carries features at full resolution ensuring better boundary adherence. The other performs a series of convolution and pooling operations to obtain low-level features. The two streams are then combined using full-resolution residual units (FRRUs). Furthermore, FRRNs use their own initialisation and train from scratch as opposed to using pre-trained models. The major





**Figure 3.5** AdapNet Architecture. Source [142]

disadvantages of this approach are the requirement massive amounts of labelled examples for different domains compared to other networks, this approach also requires significantly more training time than most networks. This makes it a challenge to use in most real-world problems.



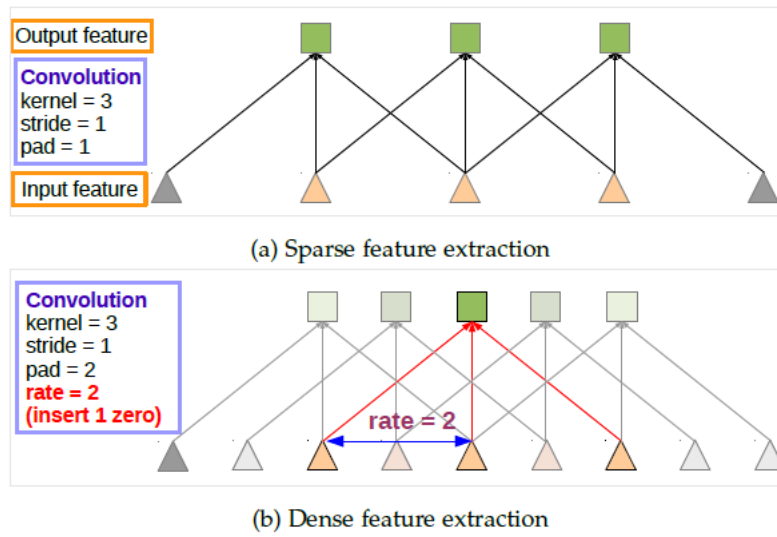
**Figure 3.6** Full-Resolution Residual Networks (FRRNs). This approach uses two processing streams, the pooling stream performs the convolution and pooling operations while the residual stream carries features at full resolution. The two streams are combined using full-resolution residual units (FRRUs) providing better localisation and boundary adherence [118].



The DeepLab [21, 22] series from Google were developed almost in with the Inception [135, 136] series and they follow closely related design principles. These papers were motivated by the need to address the challenges of speed, accuracy and model simplicity. DeepLabv1 noted the mutual technical hurdles of signal down-sampling and spatial insensitivity (invariance) as key contributors to poor localisation and poor boundary detection. They proposed **atrous** convolution as a solution for signal down-sampling. *Atrous* is a French word for *with holes*. Atrous convolution is used for controlling the field of view.

$$y[i] = \sum_k x[i + r \cdot k]w[k] \quad (3.1)$$

Atrous convolution applied over input feature maps  $x$ ,  $r$  is the atrous rate (the sampling stride),  $w$  are the filters and  $y$  is the output.

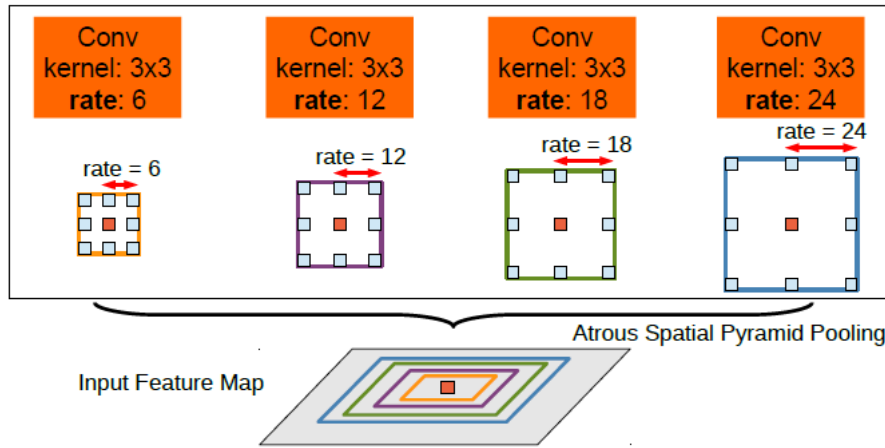


**Figure 3.7** Graphical visualisation of Atrous convolution (b) compared to standard convolution (a). Atrous convolution allows control of the field of view, for all  $r > 1$ . This is important for enlarging the field of view to incorporate larger context as well finding the most optimal trade-off between accurate localisation and context assimilation [22].

The second challenge that DeepLab sought to address is spatial invariance. Similar to FCN [92], Conditional Random Fields were used as a form of post-processing or refinement after feature extraction. This was achieved by combining activations at the final fully connected layer with the CRFs [22]. This became state-of-the-art with an average IoU of 71.6% and 8 FPS on most benchmark datasets. However, there were still challenges with global context

integration and model simplicity for adoption.

Further improvements were made to the DeepLab models [21]. In DeepLabv2 the the researchers made attempts to resolve problems caused by reduced feature resolutions and the existence of different semantic classes at multiple scales. At the time, most works would present the network with re-scaled version of the same image in order to capture feature at multiple scales, however, this method resulted in extra computational burden. Alternatively, as a continuation of their work with *atrous* convolution, Chen et al [22] proposed **atrous spatial pyramid pooling** (ASPP) to encode features at multiple scales. Additionally, this facilitates global context encoding. The intuition of ASPP is shown in Fig. 3.8.

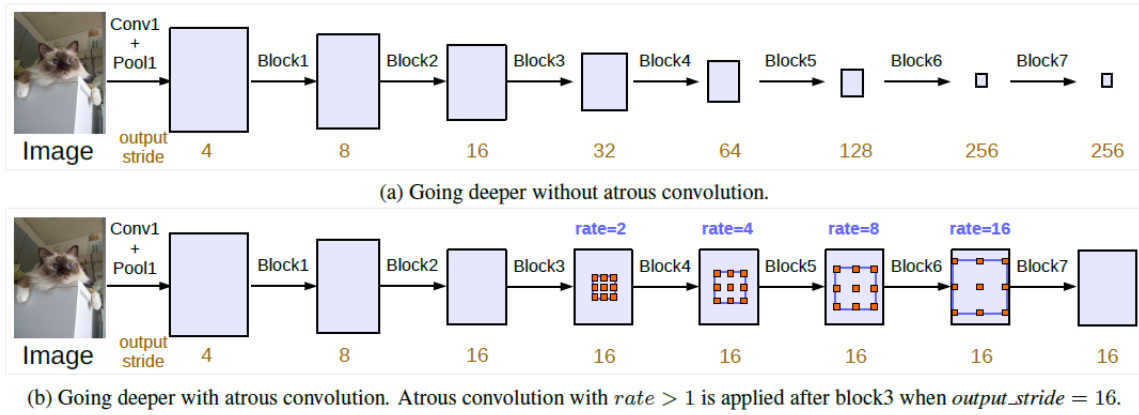


**Figure 3.8** Atrous Spatial Pyramid Pooling is used to encode features at multiple scales. From atrous convolution, ASPP is used to adjust the Field-of-Views (shown in different colours) when performing pooling operations [22].

Further efforts from DeepLab involved increasing the depth of the network as well as incorporating batch normalisation similar to the InceptionV2 model.

The Pyramid Scene Parsing Network (PSPNet) [154] also explores the idea of local-region context aggregation in order to get a global comprehension of the feature maps using a pyramid parsing module. This is performed after the final convolution layer followed by upsampling and concatenation layers that encapsulate both sub-regional and global context information. Leveraging global context information played a major role in the semantic segmentation of complex scenes and it achieved state-of-the-art results in 2006 [154].

The Tiramisu Network (DenseNets) [65] is an extension of ResNet. It uses dense blocks and pooling operations. The DenseNets are interactive concatenations of previous feature maps.



**Figure 3.9** DeepLabv3, Going Deeper with Atrous Convolutions. In part (a) we can see that when using standard convolution, output stride increases as the depth of the network increases, resulting in much smaller feature maps which negatively impacts segmentation results. In part (b) however, atrous convolution helps keep the stride constant while the field of view is increased, without increasing the number of parameters. This leads to better segmentation results [22].

The main contributions from DenseNets are 1.) the iterative summation and re-use of feature maps, 2.) parameter efficiency and implicit deep supervision [65]. DenseNets improved state-of-the-art results in 2017. However the size of the network renders impractical for real-time applications.

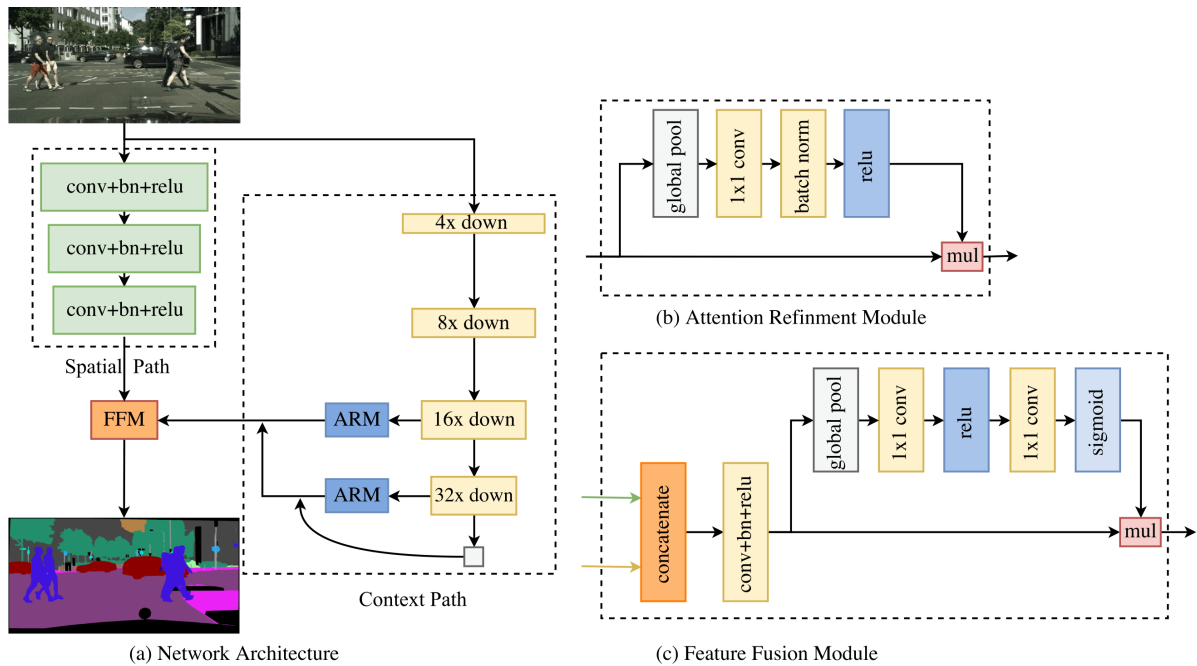
Citing the limitations of an ongoing trend of stacking small filters, Global Convolutional Network (GCN) [115] argued that larger kernels are much more useful for capturing global context information. The GCN architecture was aimed at directly addressing the simultaneous yet conflicting challenges of localisation and classification. GCN network also used a residual-based boundary refinement technique and achieved state-of-the-art results [115]. Though useful for global context, this approach has challenges with salient classes such as tree trunks or street lights.

Dense Decoder Shortcut Connections (DDSC) [12] for single-pass semantic connections noted a prominent problem with the usage of image pyramids for multi-scale inference and evaluation. The increased computational burden and the consequent in-feasibility with real-time applications. In an attempt to remedy this challenge GCN proposed a technique based on cascaded networks that incorporates the ResNeXt [153] residual building blocks and further uses repeated building blocks that aggregates a sets of transformations with the same topology. The dense decoder shortcut connections allows for the usage of semantic feature maps from all preceding decoder layers thus enhancing effective information propagation

from one decoder block to another. Furthermore, this helps with multi-level feature fusion.

Multi-path refinement networks (RefineNet) by Lin et. al [85] sought to further address the challenges with CNNs with regard to reduction in input resolution. They presented a generic multi-path refinement network that explicitly exploits all down-sampling information to facilitate high-resolution segmentation using long-range residual connections.

An ongoing challenge in semantic segmentation is obtaining real-time performance while maintaining state-of-the-art accuracy levels. Most approaches often compromise one for the other. One of the most recent works that attempted to address both challenges simultaneously is the Bilateral Segmentation Network (BiSeNet) [151]. BiSeNet uses two main parts, one for processing spatial information (Spatial Path) and another for processing context information (Context Path). This is done through two respective modules in these respective paths, the Feature Fusion Module (FFM) and the Attention Refinement Module (ARM). The graphical illustration of BiSeNet is shown in Fig. 3.10

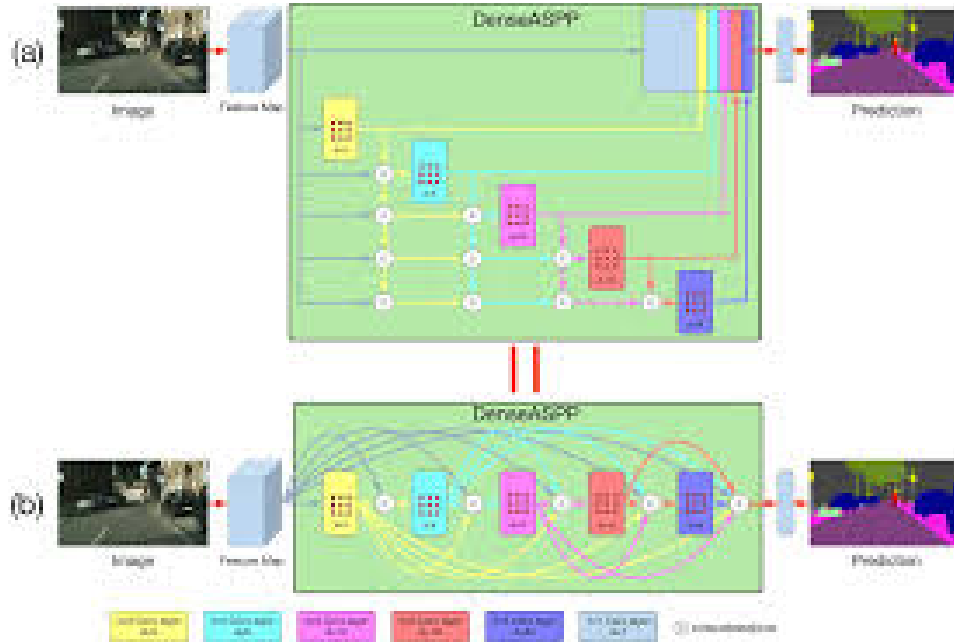


**Figure 3.10** BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation [151].

The Image Cascade Network (ICNet) by Zhao et al [155] is another recent publication that also focuses on real-time semantic segmentation. ICNet uses multi-resolution branches and label guidance as part of their solution. Precisely, ICNet is a novel state-of-the-art approach

that utilises semantic information from low-resolution feature maps in conjunction with high-resolution feature maps for refined segmentation results. ICNet obtains state-of-the-art results at 5x speed-up and reduces the memory footprint by 5 times, and 30fps on high-resolution images. This is achieved by the usage of the cascade feature fusion unit along with a label guidance strategy for fast recovery and refinement of segmentation predictions Furthermore [155].

Dense Atrous Spatial Pyramid Pooling (DenseASPP) [148] combines the ideas of *atrous convolution* [22], *atrous spatial pyramid pooling* [21] and *DenseNets* [12]. This architecture generates multi-scale features that cover larger and dense scale ranges without significant increases in model sizes. The DenseASPP architecture is shown in Fig. 3.11.



**Figure 3.11** DenseASPP, (a) illustrate DenseASPP in detail, the output of each dilated convolutional layer is concatenated with input feature map, and then feed into the next dilated layer. Each path of DenseASPP compose a feature representation of correspond scale. (b) illustrate this structure in a more concrete version [148].

### 3.3 Datasets

The widespread success of deep learning solutions in vision can be attributed to the availability of large-scale datasets such as ImageNet [124], PASCAL VOC and PASCAL-Context

[37], and Microsoft COCO [88] among many others. While most public datasets cater for object detection and classification, not many were meant for scene classification.

Datasets such as CamVid [16], ADE20K [156], and a few others made great progress towards scene understanding. However, they were still insufficient for autonomous driving purposes. The challenge with datasets for scene understanding in autonomous navigation is diversity and variability of the environments. The KITTI Vision Benchmark Suite [46] is extensive but its meant for feature based neural networks and lacks pixel-level annotations necessary for semantic segmentation. Synthia [122] was an effort to remedy this challenge via the use of synthetic data. Mapillary’s VISTAS [107] also made quite a significant contribution towards publicly available data for semantic segmentation.

The largest and most robust public dataset for scene understanding in autonomous navigation to date is the CityScapes [27] dataset. It has been widely adopted as the benchmark dataset. The CityScapes dataset focuses on autonomous navigation in structured urban scenes. This dataset has a total of 19 classes commonly found in city road scenes. Despite these significant advancements, the challenge still remains for outdoor unstructured, non-urban environments. Furthermore, the Cityscapes dataset also provides thorough metrics for bench-marking purposes.

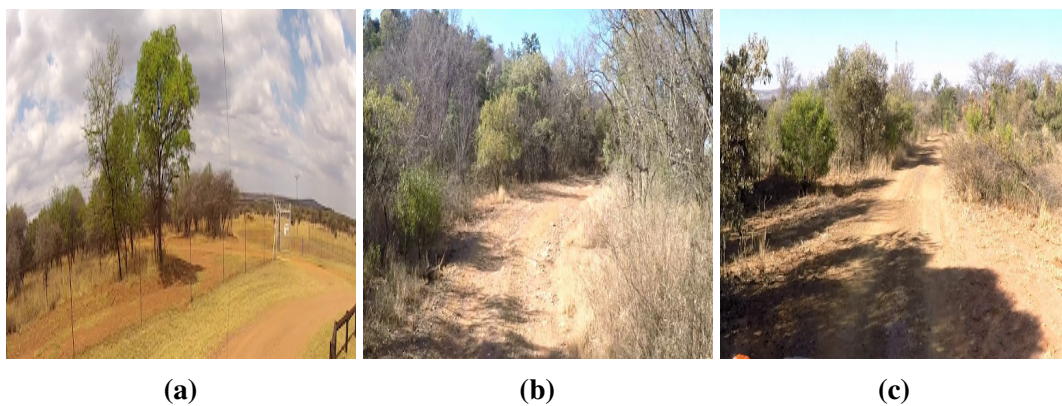
The Freiburg Forest [141] is a multi-spectral and multi-modal dataset for semantic segmentation in outdoor unstructured scenes. This dataset contains 15,000 images with 366 groundtruth fine-annotations. It was collected over 3 different days in summer on a 4.7km circuit. The dataset consists of 6 main classes namely *obstacle*, *trail*, *sky*, *vegetation* and *void*.

The domain of the Freiburg Forest dataset is very similar to ours (CSIR). Moreover, it is not sufficient to capture all the different possible conditions necessary for successful autonomous navigation in unstructured terrains. To this end, our dataset is a contribution towards capturing some of the variability and complexity in unstructured domains.

The data collection and preparation process is usually expensive and extremely time consuming. For instance, fine annotations using the LabelMe toolkit takes about 1.5 hours by trained expert annotators. On the other hand, successful deep learning solutions depend on copious amounts of data. A widely adopted solution for inflating small datasets to amounts feasible with deep learning requirements is **data augmentation** [116]. This process involves

applying several transforms to the existing small dataset in attempt to make it more diverse and robust. This increases the size of the dataset by a factor equal to the number of transforms applied. For example, AdapNet [142] augments each data point in their training 300 times [141].

The CSIR dataset was created using the CityScapes and Freiburg Forest data collection and annotation principles. Due to limited resources and limited data this research also leverages various techniques of data augmentation. This will be covered at length in the methodology section ahead. Examples from the CSIR dataset are provided in Fig. (3.12).



**Figure 3.12** Examples from the CSIR dataset.

## 3.4 Summary

In this chapter we started with traditional semantic segmentation approaches. We then gave a detailed account of the progression from traditional methods to ANNs. We provided details on the inspiration behind ANNs as well as the mathematical formalisation of ANNs. Furthermore, we discussed various approaches of learning with ANNs such as feature based ANNs and CNNs. We also discussed the theoretical building blocks of CNNs and semantic image parsing. Lastly, we provided an exhaustive literature review on state-of-the-art CNN architectures for semantic segmentation as well as the associated datasets. The next chapter provides details of our research methodology.





# Chapter 4

## Research methodology

### 4.1 Introduction

In the introduction section of the previous chapter we mentioned the objectives of this research, to perform semantic segmentation in outdoor unstructured scenes. In this section we provide details on the full methodology that will be used in pursuit of this goal. We provide details from the data collection and preparation process to the actual implementation techniques used. Lastly, we provide details on the hardware and software used in this research.

### 4.2 Research objectives

The fundamental purpose of this work is to identify the key factors for CNN architecture design for effective and efficient real-time semantic segmentation in unstructured outdoor environments. To accomplish this, we break down our objectives as follows:

1. To create our own novel dataset for semantic segmentation in unstructured outdoor environments (CSIR),
2. To find the best architecture for semantic segmentation in unstructured environments,
3. To establish whether transfer learning can be used for multi-class semantic segmentation to achieve state-of-the-art results on limited datasets

## 4.3 Research approach

### 4.3.1 Data acquisition and pre-processing

In this research we used two datasets, the Freiburg Forest[141] dataset and our own custom (CSIR) dataset. The CSIR dataset was recorded using a **GoPro Hero 5** mounted on the front of a Jeep truck travelling at  $20\text{km/h}$ . The general specifications of the data collection device are given in Table (4.1) below.

**Table 4.1** Data collection device general specifications.

| Device property            | Specification        |
|----------------------------|----------------------|
| Image recording format     | JPEG, RAW            |
| Maximum video resolution   | 3840 x 2160 (4K)     |
| Effective photo resolution | 12.0 Megapixel       |
| Continuous shooting speed  | 30 frames per second |
| Image stabiliser           | Electronic           |

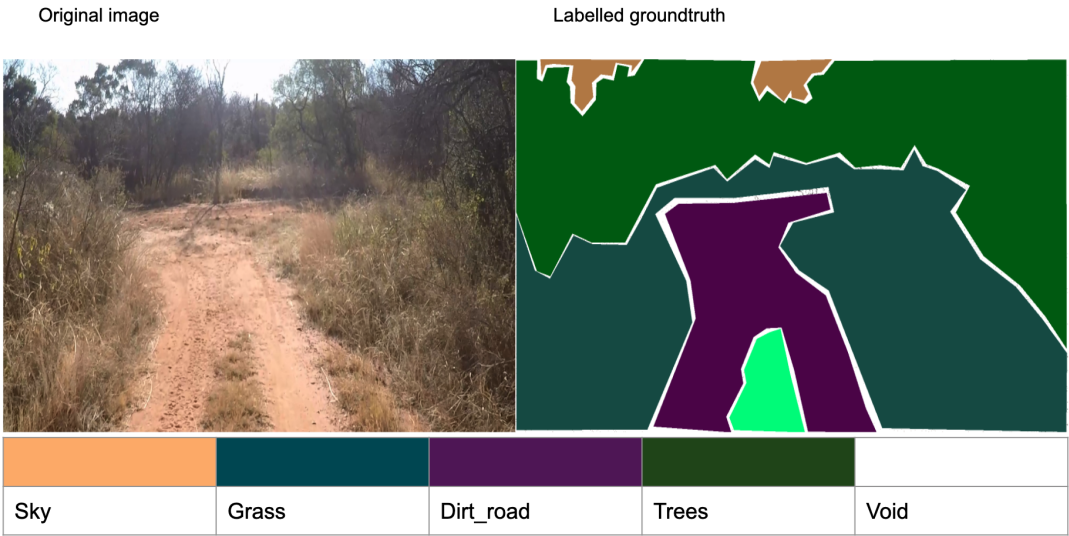
The cameras were forward-facing tilted downward at an angle of 15 degrees. These images were collected for 3 hours during winter time on a clear to partially overcast day. These conditions are far from capturing all the possible variations in the CSIR environment. Ideally, diverse data would have required collection over several days across different seasons, different weather conditions and different lighting conditions. This was a challenge due to limited resources. However, the conditions captured still suffice for the purpose of this research. Furthermore, our dataset presents a unique set of conditions compared to what is currently available in the public domain [27, 141]. These conditions include cloudy skies, dry season conditions with poor local contrast among different classes, and the overall unstructured-ness of the scenes. Consequently, the condition considered in this research are restricted to those encompassed in the CSIR and Freiburg Forest datasets.

After the recording process we grabbed every single frame from the video clips. We then further selected the frames that were most representative of the entire dataset for labelling. The CSIR sequence contains a total of 10,000 images with 310 coarse annotations. The annotations were done by a single individual. As a result, labelling accuracy was traded for labelling speed. The annotation process followed the CityScapes and Freiburg Forest labelling policies. The labelling task requires assigning every pixel in the image to a specific label. No boundaries were allowed to overlap. Pixels were assigned to the void class/label in cases where the annotator was uncertain which label those pixels belonged to. Class selection was based on frequency across the entire set, importance with regards to the goals of this research, and practicality in terms of the annotation efforts. In many cases it was a challenge



**Figure 4.1** Data collection vehicle and camera mounting.

even for the human annotator to distinguish between classes. Labelling was done by manually selecting the 30<sup>th</sup> frame of a 30-frame video snippet. It was adequate to label one in every 30 frames due to the vehicle speed during the data collection process, there are minimal to no changes in the 30-frame snippets. Exceptions were made in cases where there were either notably significant or notably non-significant changes to the scenery within 30 frames. This resulted in a total of 6 classes namely **grass\_short**, **grass\_tall**, **sky**, **dirt\_road/trail**, **trees** and **void**. An example of the annotations is shown in Fig. (4.2) below,



**Figure 4.2** An example of the class annotations from the CSIR dataset.

The distinction between the two grass types were first necessitated by traversability. Short grass is traversable compared to tall grass, thus semantically they ought to be different classes. This condition was however relaxed during experiments. This will be explained in the results section ahead. No other object classes were considered for the purposes of this research.

The CSIR dataset is still too small for deep learning purposes. Data augmentation techniques were utilised to increase the datasets. A total of 6 transformations were applied to each image. These transformations are affine/warp, vertical and horizontal flipping, rotation, brightness adjustment, and random cropping. Given the unstructured nature of the scenes and the limitations of the dataset size, these transformations were necessary for adding robustness. Additionally, during the annotation process only one frame in 30-frame snippet was annotated, these transformations catered for additional possible changes that could occur in such snippets. This resulted in a total of 1,860 images with corresponding annotations for the CSIR dataset and 2,136 for the Freiburg Forest dataset. The datasets are split into training, validation and testing using 60%, 20% and 20% ratio, respectively.

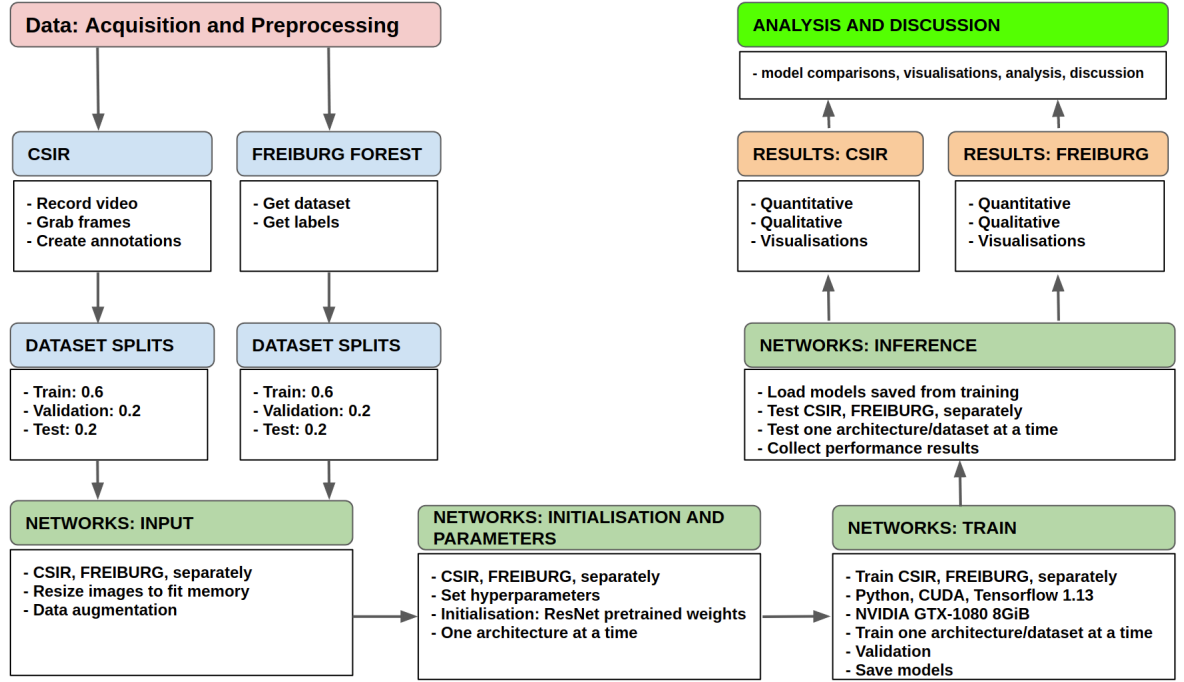
Figure (4.3) below illustrates some of the similarities and differences between the CSIR and Freiburg Forest datasets:



**Figure 4.3** Comparisons between the Freiburg Forest (top row) and CSIR (bottom row) datasets. As illustrated in this figure, the two datasets are different but they compliment each other. Thus, the Freiburg Forest dataset was a fitting choice for this research.

### 4.3.2 Research pipeline

The overall framework for this research approach from data collection to the final results is shown in Fig. (4.4) below:



**Figure 4.4** An overview of the design steps in our research pipeline. This research is a comparative study of state-of-the-art CNN architectures for semantic segmentation. All the architectures were trained and evaluated on two similar datasets (CSIR and Freiburg Forest).

Figure 4.4 above illustrates the general overview of our research methodology. In the section below we provide details on every step of this research.

**Data acquisition and pre-processing.** This is the first step, as described in Section (4.3.1). Additionally, the resulting images after annotations were of resolution ( $1920 \times 960$ ) for the CSIR dataset. The Freiburg Forest dataset had various dimensions ranging from ( $300 \times 300$ ) to ( $900 \times 900$ ). Both datasets contained 3 channels of red, blue and green (RGB). Both these datasets were further manually split into respective **train**, **validation** and **test** sets using the 60%, 20%, 20% ratio, respectively. These train and validation sets were then fed into various CNN architectures as input during the training phase. The test set was reserved for inference/testing after training.



**CNN models used.** The next steps involved passing in the input to the CNN models, setting hyperparameters and network initialisation. In this research we used the following CNN architectures: AdapNet [142], BiSeNet [151], FRRN A and B [118], Encoder-Decoder (SegNet) [5], Encoder-Decoder with skip connections [5], PSPNet [154], MobileUNet [59], MobileUNet with skip connections [59] and our own custom architecture. For consistency and GPU memory considerations images in both datasets were resized to  $(512 \times 512)$  using adaptive interpolation just before they were fed into the networks for training and validation. Due to the limited size of the datasets it was necessary to perform data augmentation. We used online data augmentation, that is when the augmentation process is only done just before training without needing to explicitly store the augmented data outside of the training process. A total of 6 transformations were used during the augmentation process: affine, horizontal flip, vertical flip, rotation, brightness adjustment, and random crop transformations. For a CNN model to train, it first goes through the **forward pass**. This is when the dataset and all parameters are loaded into memory. If the required memory for the forward pass surpasses available memory then the model will be unable to train. Furthermore, these memory limitations also forced us to reduce the resolutions of our training images to  $(512 \times 512)$ , notwithstanding that this could negatively impact the final segmentation results.

**Initialisation and training.** For network initialisation we leveraged the advantages of transfer learning as alluded to in the theory Section (2.5.11) above. Due to our limited data problem we used pre-trained ResNet weights [56]. The ResNet networks are trained on the ImageNet dataset which contains tens of thousands of images for various vision tasks.

**Losses and optimisation.** All architectures were trained separately for each dataset with hyperparameters as specified in their original papers. These hyperparameters are batch size, learning rates, number of epochs, kernel sizes. These were discussed in detail in Section (2.5.10) as well as the state-of-the-art literature review section. The only exception was on the **custom model** where we experimented with a combination of varying hyperparameter settings until we achieved satisfactory results on the CSIR dataset and the Freiburg Forest dataset. We used two gradient based optimisation methods. Adam and RMSProp optimisation. These were also explained at length in Section (2.5.8).

**Performance evaluation.** During the training phase, feature maps are extracted from the inputs using a series of convolutions and pooling. At the end, feature maps from the inputs are then compared with feature maps from the corresponding groundtruths. This is during the Softmax layers. The losses are then calculated and the parameters adjusted according to

the optimisation scheme. Validation was done using the validation set and was performed concurrently with the training process. The losses were computed using multi-label **categorical cross-entropy loss**. As explained in Chapter (2.5.3) accuracy alone is not a good performance measure. We thus use the much stricter Jaccard Index commonly known as the mean intersection over union (IoU). IoU is the commonly adopted metric for pixel-level segmentation.

**Testing and inference.** Performance logs, prediction masks and the models are saved during the training phase. The saved models are then used on the separately held testing data for testing and inference. Similar to the training process, all architectures were tested separately for each dataset. Further performance logs, and prediction masks from testing are saved for later use in the analysis and discussion of results.

**Analysis and discussion.** After collecting all the results from the training, validation and testing phases we provide detailed quantitative and qualitative analysis as well as a discussion.

## 4.4 Hardware and Software

This section provides descriptions of all the hardware, software, and libraries that were used throughout this research.

## 4.5 System details

### 4.5.1 Hardware

This research was conducted on a single workstation with the following specifications:

**Compute unified device architecture** (CUDA) is a parallel computing platform and programming model from the NVIDIA corporation. The use of CUDA and **Graphic Processing Units** (GPUs) has revolutionised accelerated computing. Traditionally, parallel computing was CPU based. Graphics cards that would normally be used for gaming and visual aesthetics can be re-purposed for accelerated and parallel computing. This is achieved via the use of the CUDA programming model. In this research we used CUDA 9.5.

**Table 4.2** Hardware Specifications

| Hardware property          | Specification                                |
|----------------------------|----------------------------------------------|
| Operating System           | Linux Ubuntu 18.04 LTS                       |
| Architecture               | x86_64                                       |
| CPU op-mode(s)             | 64-bit                                       |
| CPU Cores                  | Intel® Xeon(R) CPU E5-2690 v4 @ 2.60GHz × 28 |
| RAM                        | 128 GiB                                      |
| Thread(s) per core         | 2                                            |
| Core(s) per socket         | 14                                           |
| CPU max MHz                | 3500 MHz                                     |
| GPU                        | NVIDIA GeForce GTX 1080/PCIe/SSE2 (8GiB)     |
| GPU CUDA Cores             | 2560                                         |
| GPU Memory Clock           | 10 Gbps (Processor Clock (MHz) 1733)         |
| GPU Standard Memory Config | 8GiB GDDR5X                                  |

### 4.5.2 Software

**LabelMe**, as explained in Section (4), this is an image annotation toolkit that uses Javascript for drawing polygons of arbitrary sizes. The coordinates of these arbitrary shapes are saved in JSON format, which is then used for creating target labels or groundtruths.

Furthermore, the coordinates returned by LabelMe were then used in conjunction with the Open Computer Vision library (OpenCV) for assigning different colours to the different target classes. **OpenCV** was also used for various other pre-processing and post-processing requirements. **TensorFlow** and Keras are optimised multi-purpose framework for training deep CNNs. We used these frameworks for setting up and training all the architectures. All the implementations were scripted using **Python3.6**. We also used the **Scientific Learning Toolkit** (SciKitLearn) for some processing tasks as well as logging and visualising experiment results.

## 4.6 Summary

This chapter provided all the details on our research methodology from the data acquisition and pre-processing all the way to the discussion and analysis of results. The top results from all architectures are compared and visualised. We also provided an overview of the hardware and software used in this research. The next chapter provides an extensive quantitative and qualitative discussion and analysis of the results as well as convergence of the CNN



---

architectures. Experiment details, detailed results and discussion will be covered in the next chapter (and [Chapter 6](#)).



# Chapter 5

## Results and discussion

### 5.1 Introduction

The objectives of this research were presented in Chapter (4). We now present a detailed study of the various CNN architectures with regards to their performance on the two datasets used. We were particularly interested in the CNN architectures responses to different classes. The primary goal was correctly segmenting the specified classes and minimise misclassifications. The classifications were based on standard RGB images thus, the study will focus the CNN responses to colour and texture of different classes.

It is important to elaborate that there are two broad categories of factors contributing to the overall results. These are CNN **architectural factors** and **dataset factors**. We therefore discuss and analyse each of the main architectural factors, for each architecture, in relation to their effect on the dataset specific factors and the contribution to the ultimate segmentation results. Architectural factors include **network topology**, **number of convolutional layers**, **pooling strategies**, **number of layers**, and **classification strategies**. The dataset factors include **label granularity**, **dataset size**, **class distribution** as well as **domain** and **contrast**. Furthermore, we discuss the effects of data augmentation and other factors.

This was a multi-class segmentation problem. Naturally, class distribution is an important contributor to the overall performance of the models. We provide both qualitative and quantitative results for all the architectures that were used in this research. In the next section we begin by providing a discussion and analysis of the results from a qualitative perspective. This will be followed by quantitative analysis and discussion in the section after.

In order to get an unbiased understanding of the different CNN architectures and their performance on our datasets we trained and tested them as specified in the original papers. All architectures were initialised using ResNet pretrained weights. It is important to note that some state-of-the-art architectures were not considered in this study due to memory and compute limitations. For brevity, we first show only the top four results on each dataset. The results of the remaining architectures are provided in the extended results section.

## 5.2 Qualitative results and analysis

In this section we show and compare the actual predictions from the various state-of-the-art architectures. We focus on how each architecture performed on both datasets as well as how each architecture compares with the other architectures.

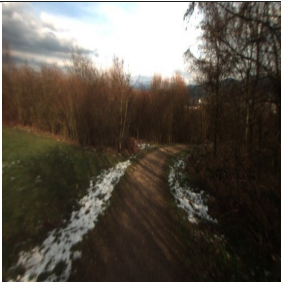
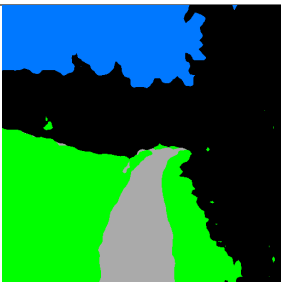
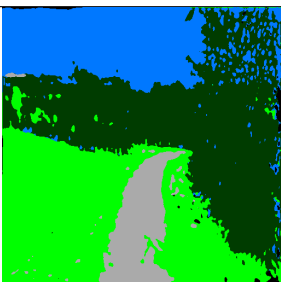
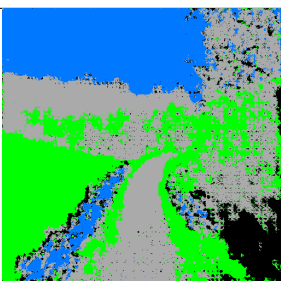
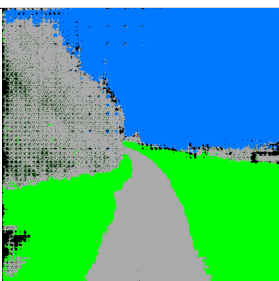
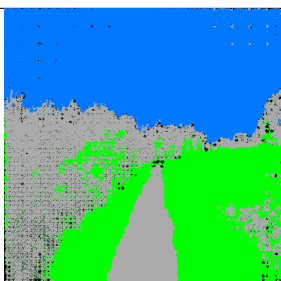
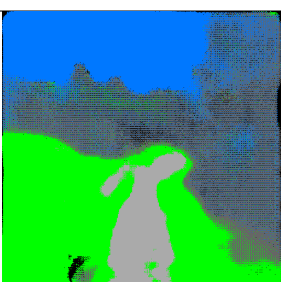
Tables (5.1) and (5.2) show only the prediction results from the top-four architectures. We show the original test image, the target groundtruths and the predictions from the models. The remainder of the results from the other architectures is provided in the extended results section. These predictions were collected during the testing/inference phase using model checkpoints that were saved after training each architecture. Similar to the training procedure where the input was a full image consisting of various possible classes, the testing procedure also takes in a full image. The testing phase thus attempts to infer the semantic classes in an unseen sample using the parameters learned using the training set.

The main challenges in multi-class semantic segmentation are **pixel localisation** and **class boundary delineation**. Pixel localisation helps with understanding the relative location of any given pixel in the whole image. This constitutes of the relative location with respect to a segmented class or relative location with respect to the entire image. This is crucial for successful autonomous navigation and scene comprehension.

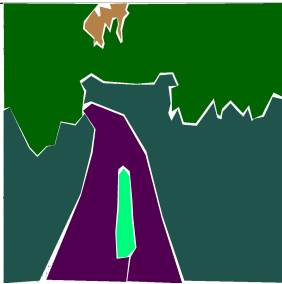
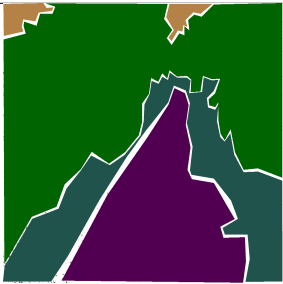
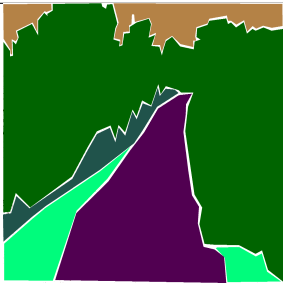
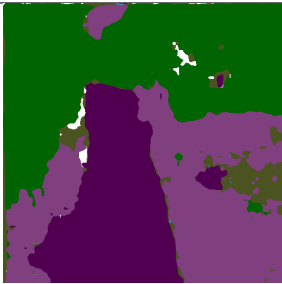
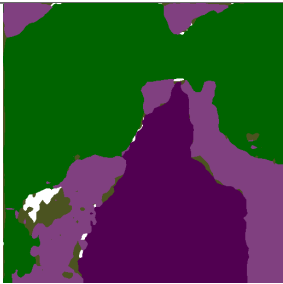
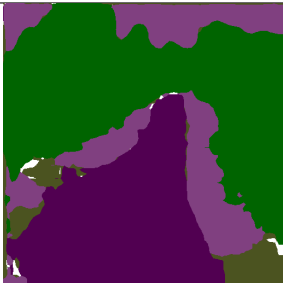
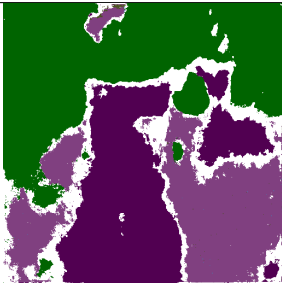
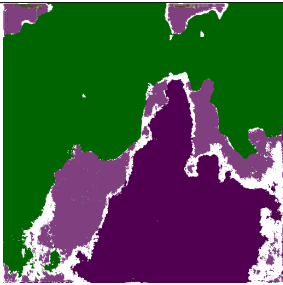
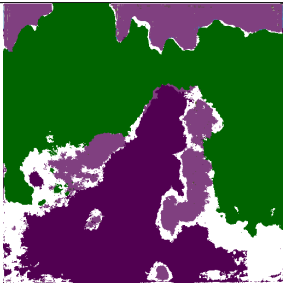
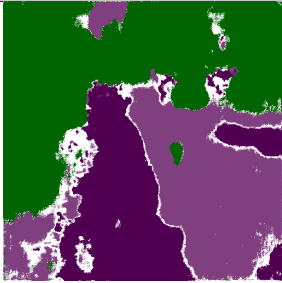
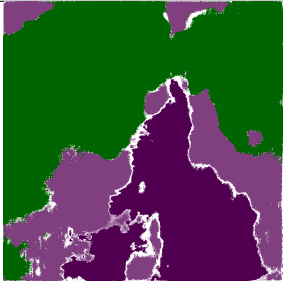
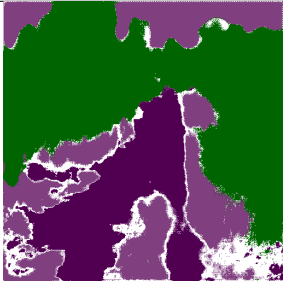
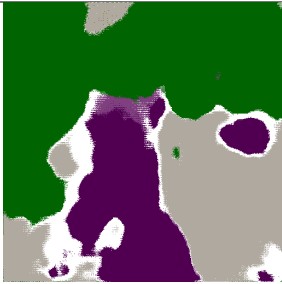
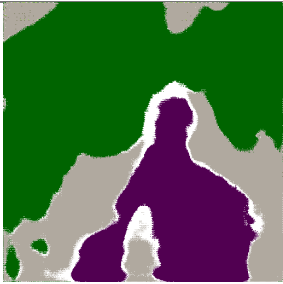
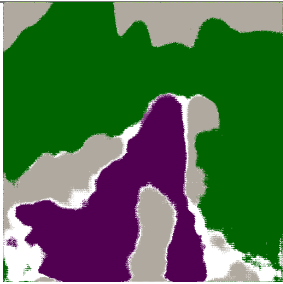
Though secondary to pixel localisation, boundary delineation is also equally important. This involves the ability of a segmentation model to tell different classes apart. Precisely, where one class ends and where the other starts.

Qualitative analysis enables the visualisation of the model performances with respect to class-wise segmentation. Specifically, pixel localisation and class boundary delineation.

**Table 5.1** Predictions on Freiburg Forest dataset - Top 4 models

|                          |                                                                                     |                                                                                      |                                                                                       |
|--------------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Test samples             |    |    |    |
| Groundtruth/Target       |    |    |    |
| BiSeNet                  |   |   |   |
| FRRN-B                   |  |  |  |
| Custom                   |  |  |  |
| Encoder-Decoder (SegNet) |  |  |  |

**Table 5.2** Predictions on CSiR dataset - Top 4 models

|                          |                                                                                     |                                                                                      |                                                                                       |
|--------------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Test Samples             |    |    |    |
| Groundtruth/Target       |    |    |    |
| BiSeNet                  |   |   |   |
| FRRN-B                   |  |  |  |
| Custom                   |  |  |  |
| Encoder-Decoder (SegNet) |  |  |  |

As explained in the literature review, Section (3.10), the **BiSeNet** architecture uses two processing streams, the Spatial path or the Feature Fusion Module (FFM) and the Context or the Attention Refinement Module (ARM). This explains its ability to successfully classify and localise pixels while maintaining boundary delineation. The classes of interest were **dirt\_road**, **grass\_tall**, **grass\_short**, **trees**, **sky** and **void**. Dirt\_road is shown in grey on the Freiburg dataset and in dark purple on the CSIR dataset. Grass is shown in lighter green on Freiburg and light purple on CSIR. Trees are shown in dark green on both datasets. The void class is shown in white and the sky class is shown in blue on the Freiburg dataset whereas on the CSIR dataset its shown in brown.

As shown in Tables (5.1) and (5.2), the **BiSeNet** architecture outperforms the other models on the Freiburg Forest dataset. It successfully segments all the target classes, **dirt\_road**, **trees**, **sky** and **grass**. The Freiburg Forest dataset has fine or granular groundtruths. As a result, the **void** class is almost absent in this dataset. Furthermore, there is a well balanced class distribution on the Freiburg Forest dataset, thus the BiSeNet model satisfactorily classifies all the pixels in the test images according to their respective classes. Unlike the Freiburg Forest dataset, the CSIR dataset is more challenging since it has coarse and less granular groundtruths, and there is an unbalanced class distribution. Additionally, there is higher contrast among different classes in the Freiburg Forest dataset compared to the CSIR dataset. As a result, BiSeNet was able to correctly classify the pixels in the dominant and frequent classes such as dirt\_road, grass and trees. In some cases the model confused the void and grass classes. This is not surprising given the lack of granularity in the annotations as well as the low contrast among the different classes. Model accuracy and convergence will be discussed in the quantitative analysis section ahead.

The **FRRN** architectures are also similar to the BiSeNet architecture. They also use two processing streams. The residual stream uses **residual units (RUs)** to process the images at full resolution whereas the pooling stream uses **full resolution residual units (FRRUs)** undergoes a series of convolution and pooling procedures processing features at more finer levels. Just like with the BiSeNet model, this explains the FRRN model's ability to classify and localise pixels while also maintaining class boundary delineation. FRRN-B goes through more pooling and unpooling steps compared to FRRN-A. Furthermore, FRRN-B maintains the full resolution of the input image compared to FRRN-A that reduces the resolution before processing. As a result, FRRN-B outperformed FRRN-A on both datasets.

Both BiSeNet and FRRN-B perform qualitatively well on the Freiburg Forest dataset, but BiSeNet outperforms FRRN-B on the CSIR dataset. This can be attributed to their architectural factors and specific factors of the CSIR dataset. BiSeNet has more convolutional blocks compared to FRRN-B. Naturally, more convolution steps on a dataset with very low contrast among classes would lead to excessive averaging out of the information, negatively impacting performance. BiSeNet circumvents this problem by using global pooling, the ARM and the FFM. Furthermore, the number of filters in the FRRN-B model is capped at 48 whereas on the BiSeNet model it is continually doubled until the FFM is evoked. BiSeNet model uses the **sigmoid** function for classification in both the ARM and FFM whereas FRRN only uses the **softmax** classifier after concatenation of the residual and the pooling streams. This results in BiSeNet's superior ability to successfully classify and localise pixels.

We also noticed that the BiSeNet model was the only model that attempted to perform finer segmentation between classes such as grass\_tall (light purple) and grass\_short (dark green). All the other models were unable to differentiate between the two grass classes. We also note that the FRRN-B model was able to correctly segment the void class (white) whereas BiSeNet was biased towards the larger classes near the void regions. Both architectures were unable to distinguish the sky class from the grass\_short class. This can be attributed to the class infrequency and size of the sky class. Bearing in mind the motivation behind this research, results from these models show progress towards autonomous navigation scene comprehension requirements. Another critical factor to consider for practical purposes would be speed and frame-rate. These will be discussed in the quantitative section below.

Unlike FRRN and BiSeNet architectures, the **Encoder-Decoder** architecture is strictly convolutional. It uses one processing stream with a series of convolution, batch normalisation, ReLUs and max pooling. This consists of 13 blocks for encoding and another 13 blocks for decoding. This is then followed by a multi-class Softmax layer for the classification. As can be seen in Tables (5.1) and (5.2), the Encoder-Decoder model performs relatively badly on the Freiburg Forest dataset compared to the other models. The performance on the CSIR dataset is slightly better compared to the performance on the Freiburg dataset. However, it is evident that the strictly convolutional approach has difficulties in pixel localisation and boundary delineation. This is mainly caused by the loss of global context as the images go through the encoding and decoding procedures as explained fully in the literature review. Furthermore, loss of spatial context and localisation precision due to excessive convolution is evidently not suitable for datasets with closely related features and very fine boundaries



across different classes.

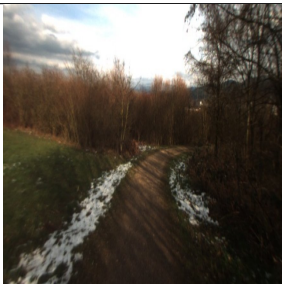
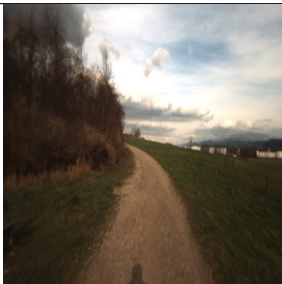
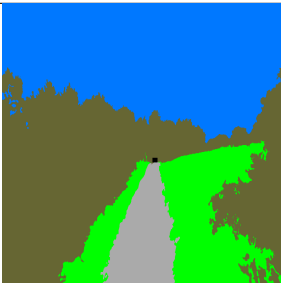
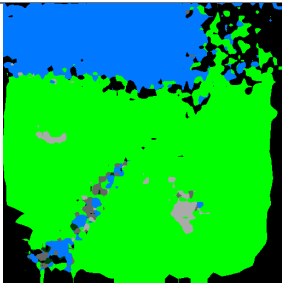
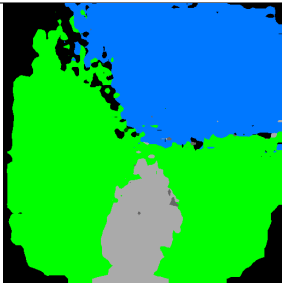
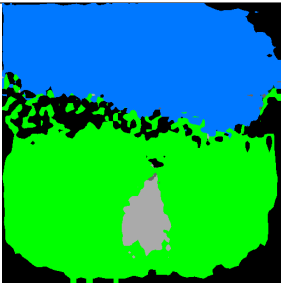
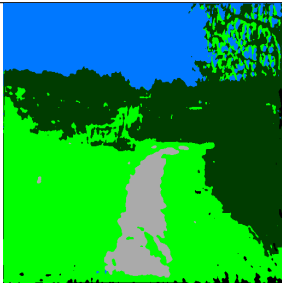
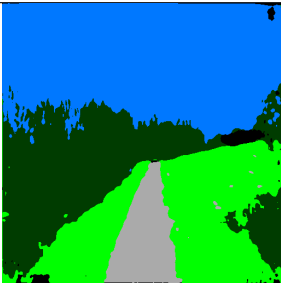
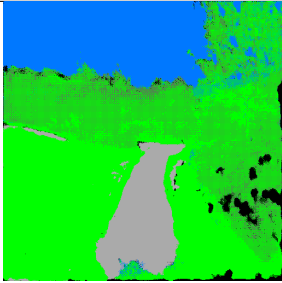
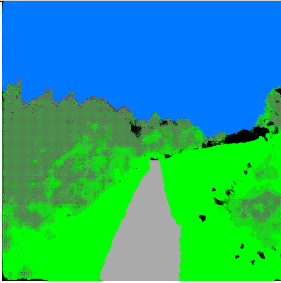
The **custom model** is based on the Encoder-Decoder architecture. We modified the hyperparameters in pursuit of a better performing model with respect to both time and accuracy. Qualitatively this model performs satisfactorily on the Freiburg dataset, but it struggles with boundary delineation on the CSIR dataset. The main modifications were to use less convolutional layers, maintaining the same number of filters and kernel size while doubling the stride after every convolution layer. Additionally, we reduced the learning rate significantly. This resulted in better results on the Freiburg Forest dataset due to reduced averaging out of features compared to the Encoder-Decoder model on the same dataset. The custom model's results on the CSIR dataset were somewhat satisfactory as the model was able to segment the main classes though the boundaries were not as finely defined as in the other top models. The custom model outperformed the remainder (extended results) of the models when looking at the results for both datasets. The main difference was that the custom model needed much less computation time and parameters. These quantitative factors are discussed in the next section.

### 5.3 Qualitative Analysis and Discussion - extended results

In Tables (5.3), (5.4), (5.5) and (5.6) we show extended results from the remainder of the models that were also used in this research. The models are AdapNet, FRRN-A, Encoder-Decoder with skip connections, PSPNet, MobileUNet and MobileUNet with skip connections. These models were also trained and tested on both the CSIR and Freiburg Forest datasets in the same manner as the top performing models in the section above. Generally, similar behaviour is observed as with the above models where most models perform much better on the Freiburg Forest dataset compared to the CSIR dataset.

The effects of the different granularity of labels in the Freiburg Forest labels and the coarseness of the labels in the CSIR dataset can be attributed to the differences in the quality of the segmentation results. Architectural factors can also be further attributed to individual model performance on the respective datasets.

**Table 5.3** Predictions - All models - Freiburg Forest extended results

|                      |                                                                                     |                                                                                      |                                                                                       |
|----------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Test Samples         |    |    |    |
| Groundtruth/Target   |    |    |    |
| AdapNet              |   |   |   |
| FRRN-A               |  |  |  |
| Encoder-Decoder-Skip |  |  |  |

**Table 5.4** Predictions - All models - Freiburg Forest extended results

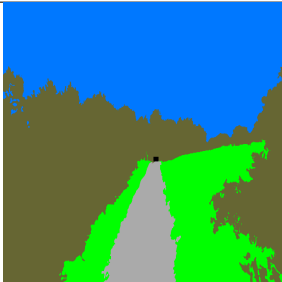
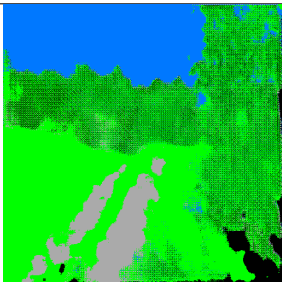
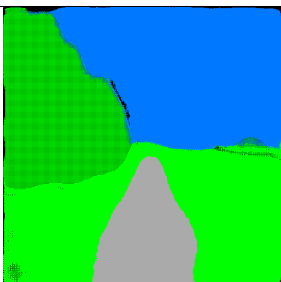
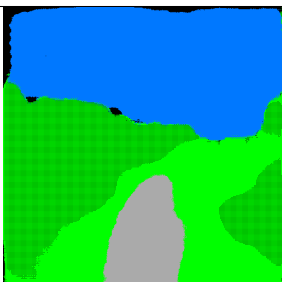
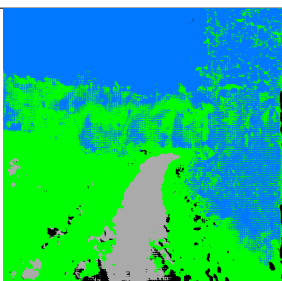
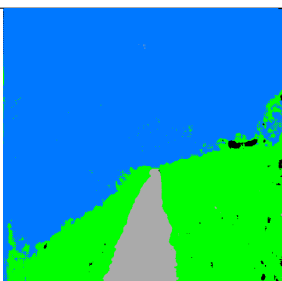
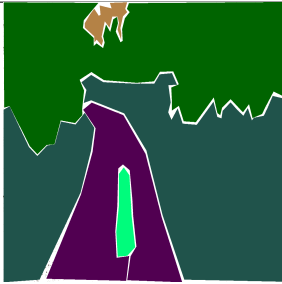
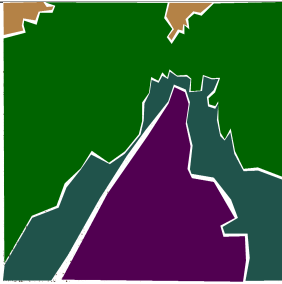
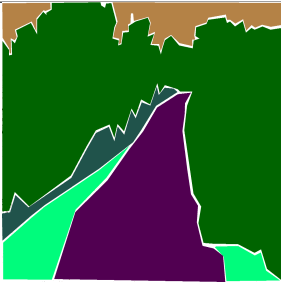
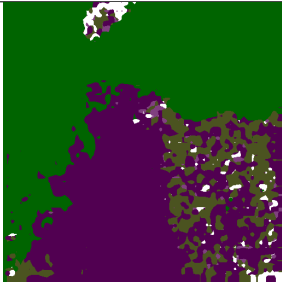
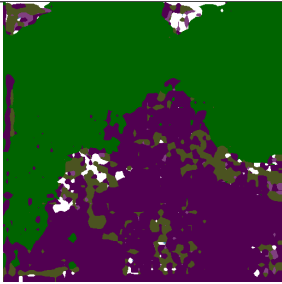
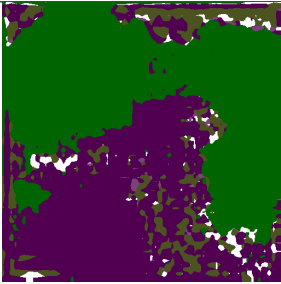
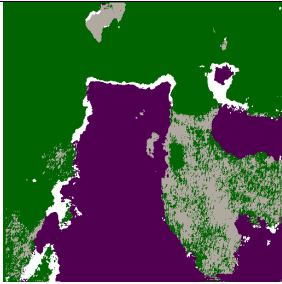
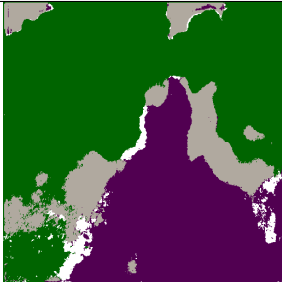
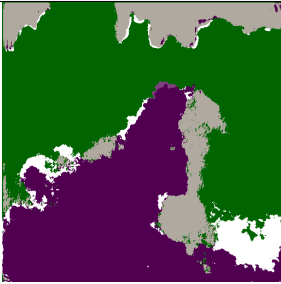
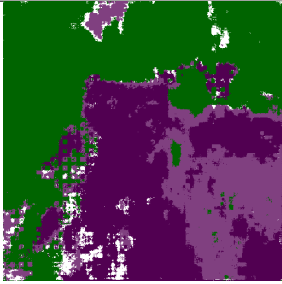
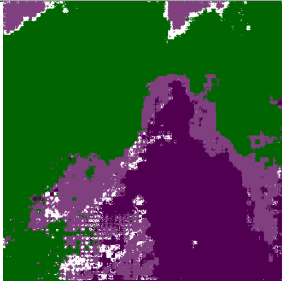
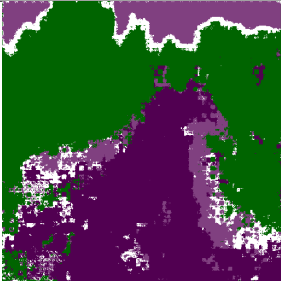
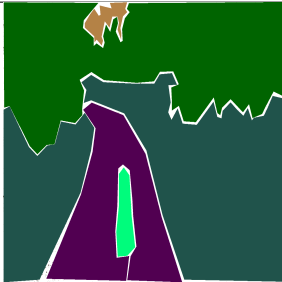
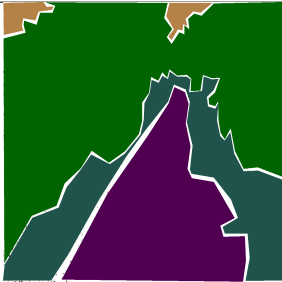
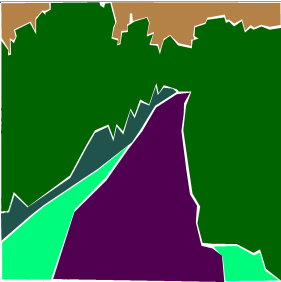
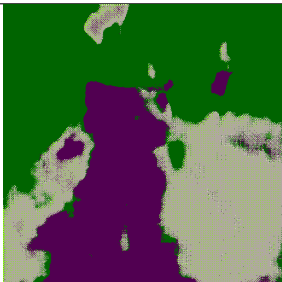
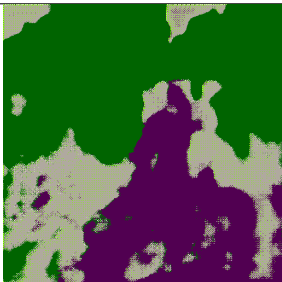
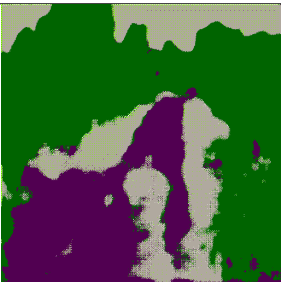
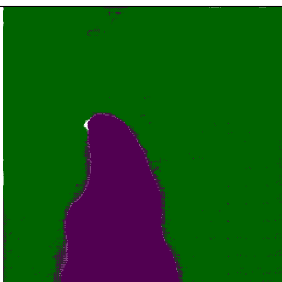
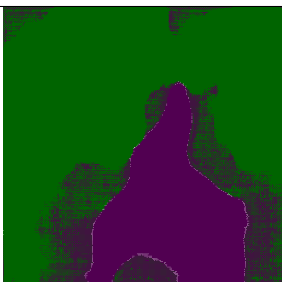
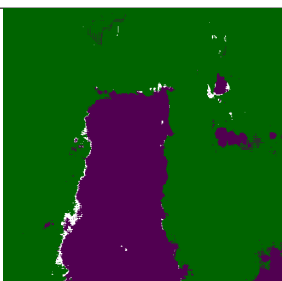
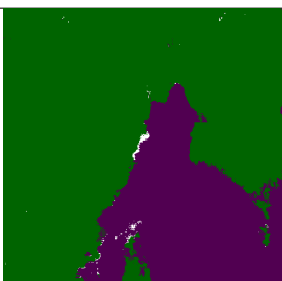
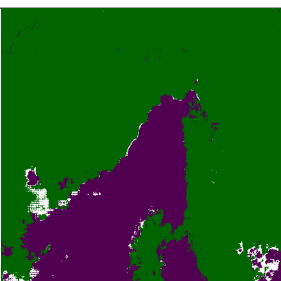
|                    |                                                                                     |                                                                                      |                                                                                       |
|--------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Test Samples       |    |    |    |
| Groundtruth/Target |    |    |    |
| PSPNet             |  |  |  |
| MobileUNet         |  |  |  |
| MobileUNet-Skip    |  |  |  |

Table 5.5 Predictions - All models - CSIR extended results

|                      |                                                                                     |                                                                                      |                                                                                       |
|----------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Input Images         |    |    |    |
| Groundtruth/Target   |   |   |   |
| AdapNet              |  |  |  |
| FRRN-A               |  |  |  |
| Encoder-Decoder-Skip |  |  |  |

**Table 5.6** Predictions - All models - CSIR - extended results

|                    |                                                                                     |                                                                                      |                                                                                       |
|--------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Input Images       |    |    |    |
| Groundtruth/Target |    |    |    |
| PSPNet             |  |  |  |
| MobileUNet         |  |  |  |
| MobileUNet-Skip    |  |  |  |



## 5.4 Quantitative results - All models on Freiburg Forest and CSIR datasets

We present Tables (5.7) and (5.8) below to provide a further overview of how the different CNN architectures fared on the two datasets figures Fig. (5.1a) and (5.1b) show the associated visualisations. Here we summarise the quantitative performance of the state-of-the-art architectures and compare them with regards to **model size** (number of parameters), **average training time** and **mean IoU**. We use the mean IoU as the accuracy measure since past works (2.5.3) have shown this to be the most suitable for pixel-level semantic segmentation.

**Table 5.7** Freiburg results - Model size, computation time, and mean IoU. The best performing model in each category is highlighted in bold text. The top four models are highlighted in blue.

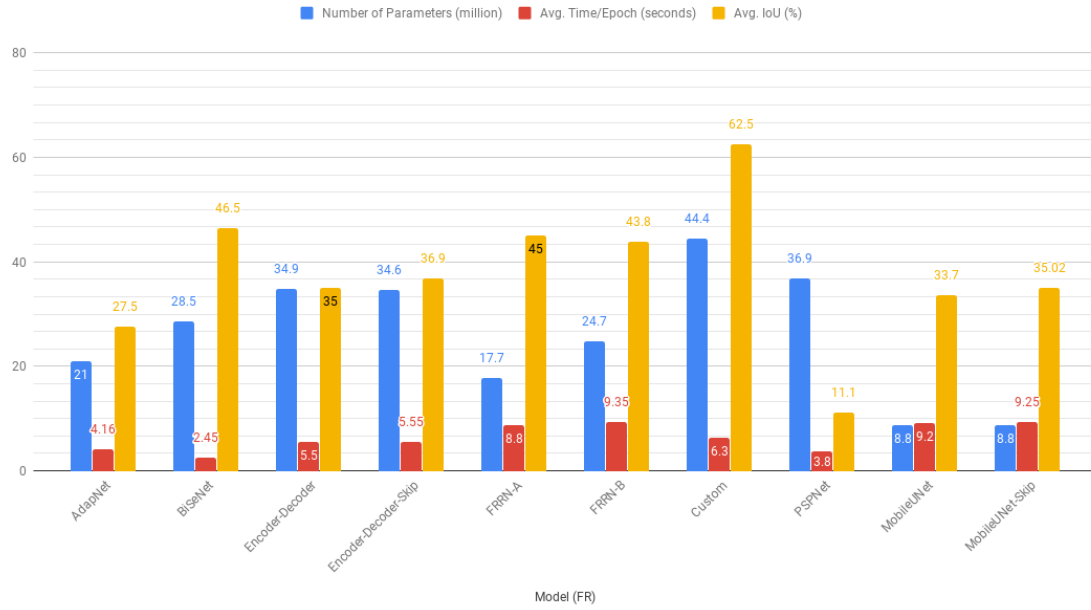
| CNN Architecture                | Parameters(million) | Avg. Training Time/Epoch(seconds) | Avg. IoU (%) |
|---------------------------------|---------------------|-----------------------------------|--------------|
| <b>BiSeNet</b>                  | 28.5M               | <b>2.45</b>                       | <b>46.5</b>  |
| <b>FRRN-B</b>                   | <b>24.7M</b>        | <b>9.35</b>                       | <b>43.8</b>  |
| <b>Custom</b>                   | 44.4M               | <b>6.30</b>                       | <b>62.5</b>  |
| <b>Encoder-Decoder (SegNet)</b> | 34.9M               | <b>5.50</b>                       | <b>35.0</b>  |
| Encoder-Decoder-Skip            | 34.6M               | 5.55                              | 36.9         |
| FRRN-A                          | 17.7M               | 8.80                              | 45.0         |
| PSPNet                          | 36.9M               | 3.80                              | 11.1         |
| MobileUNet                      | <b>8.8M</b>         | 9.20                              | 33.7         |
| MobileUNet-Skip                 | <b>8.8M</b>         | 9.25                              | 35.5         |
| AdapNet                         | 21.0M               | 4.16                              | 27.5         |

**Table 5.8** CSIR results - Model size, computation time, and mean IoU. The best performing model in each category is highlighted in bold text. The top four models are highlighted in blue.

| CNN architecture                | Parameters(million) | Avg. Training time/epoch(seconds) | Avg. IoU(%) |
|---------------------------------|---------------------|-----------------------------------|-------------|
| <b>BiSeNet</b>                  | 28.5M               | <b>4.19</b>                       | <b>36.1</b> |
| <b>FRRN-B</b>                   | <b>24.7M</b>        | <b>11.11</b>                      | <b>35.6</b> |
| <b>Custom</b>                   | 44.4M               | <b>8.13</b>                       | <b>41.2</b> |
| <b>Encoder-Decoder (SegNet)</b> | 34.9M               | <b>7.32</b>                       | <b>31.6</b> |
| Encoder-Decoder-Skip            | 34.9M               | 7.23                              | 33.8        |
| FRRN-A                          | 17.7M               | 10.60                             | 34.4        |
| PSPNet                          | 36.9M               | 5.60                              | 21.7        |
| MobileUNet                      | <b>8.8M</b>         | 10.50                             | 32.3        |
| MobileUNet-Skip                 | <b>8.8M</b>         | 11.04                             | 28.4        |
| AdapNet                         | 21.0M               | 5.04                              | 24.9        |

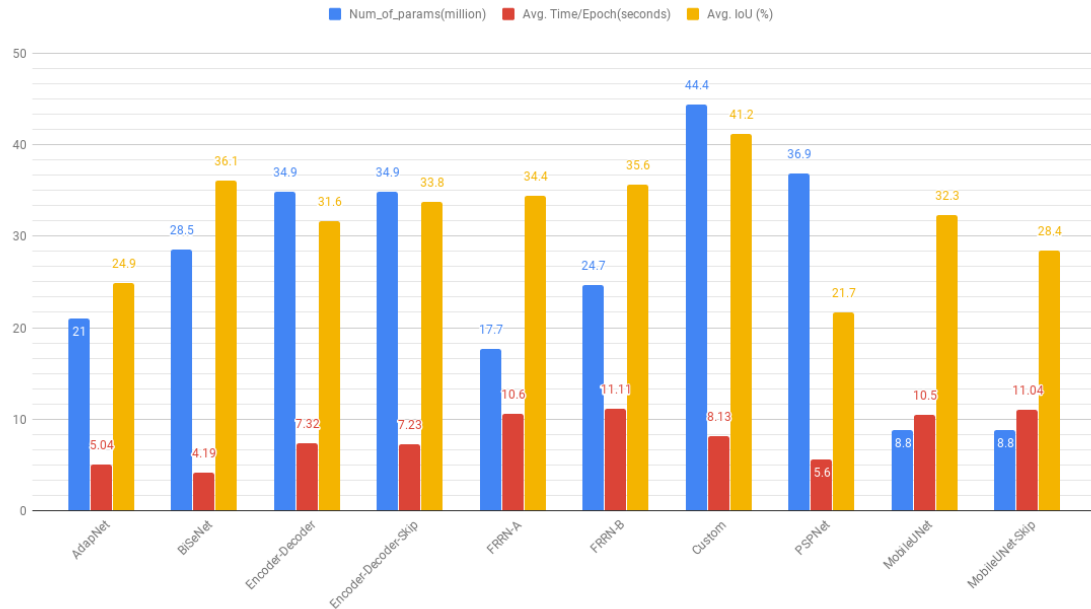
These comparisons were done bearing in mind the practical considerations of mobile autonomous systems. It is therefore important to understand whether the model is sufficiently

Model Comparisons - Freiburg Forest Dataset



(a) Performance comparisons - Freiburg Forest dataset. Architecture comparisons based on model size (number of trainable parameters, average training time, and mean IoU).

Model Comparisons - CSIR Dataset



(b) Performance comparisons - CSIR dataset. Architecture comparisons based on model size (number of trainable parameters, average training time, and mean IoU).

small to fit the limited memory of a mobile platform. This is captured in the number of parameters (model size). In addition to model size, the processing times of each model are also crucial. This is covered in the training time and inference time. Ultimately, the efficiency of a model is determined by its performance (IoU) given various memory and time configurations.

As shown in Fig. (5.1a) and Fig. (5.1b), almost all architectures exhibit consistent behaviour on both datasets. The **model-size to training time to IoU** ratios are similar for each architecture on the two datasets. For autonomous navigation purposes, the ideal combination is a minimum feasible number of parameters accompanied with faster training and inference times producing the highest possible IoUs. The number of parameters on both datasets were the same since both sets were resized to the same resolution before being fed as input for training the neural networks. The testing samples were also resized to a resolution of  $(512 \times 512)$ .

Paying close attention to the top-4 models (highlighted in blue) we see that the **BiSeNet** model had the lowest training time for both datasets. It took an average of **2.45** seconds per epoch on Freiburg Forest to train **28.5** million parameters and **4.19** seconds per epoch for the same number of trainable parameters on the CSIR dataset. The **BiSeNet** achieved mean IOUs of **46.5%** and **36.1%** on the Freiburg Forest and CSIR datasets respectively.

The **Custom** model had the highest number of trainable parameters (**44.4** million) for both datasets across all top-4 models. The Custom model also achieved the highest mean IoUs on both datasets with **62.5%** and **41.2%** on the Freiburg Forest and CSIR datasets respectively. The number of trainable parameters on the Custom model was significantly higher than the number of trainable parameters on other models. This empirically shows a direct relationship between the number of trainable parameters a model has and how well it learns. However, the Custom model needed more than twice BiSeNet's training time per epoch on both datasets.

The **FRRN-B** model had the second highest number of trainable parameters (**24.7** million) among the top-4 models. However, its training time per epoch was highest across all models on both datasets. It took **9.35** seconds on Freiburg Forest and **11.11** seconds on the CSIR dataset. FRRN-B achieved mean IoUs of **43.8%** and **35.6%** on Freiburg Forest and CSIR respectively. Note that the training time per epoch for the FRRN-B model is almost 3 times the training time per epoch for the fastest model (BiSeNet), yet their mean IoUs are very close.



For the **Encoder-Decoder** we found that the average training time per epoch was **5.50** seconds (Freiburg Forest) and **7.32** seconds (CSIR) for **34.9** million trainable parameters. Though the time and model size were very close to the average across the top-4 models, the Encoder-Decoder model had the lowest top-4 mean IoU reaching **35.0%** and **31.6%** on the Freiburg Forest and CSIR datasets respectively.

The training times per epoch on the CSIR dataset were slightly higher than the training times per epoch, even though the images had the same resolution. This is due to the different compression types used for storing the images. The Freiburg Forest dataset used the **Joint Photographic Experts Group (JPEG)** and the CSIR used the **Portable Graphics Network (PNG)**. PNG stores more information than JPEG, thus each CSIR image contains more information than each Freiburg Forest image and takes longer to process. Inference/testing time was similar for all architectures on both datasets at an average of **9s**. This translates to roughly **3 frames** per 30-second video snippet. This is only feasible for vehicles driving at really low speeds.

### 5.4.1 Convergence

As we mentioned in Section (2.5.8), the process of training a CNN is an attempt to find the best set of parameters that describe the relationship between the input images and the related labels. When using SGD approaches this is quantified by the gradient of the error with respect to the parameters. In other terms, convergence is the reduction of the error with respect to the network parameters. This reduction is highly dependent on the initialisation of the weight matrix. In this research we used the **ResNet pretrained weights** for initialisation. In the initial stages of the training process the network parameters poorly describe the relationship between the input and the associated groundtruths. As a result, the error is too large in the initial phase of the learning process. The error begins to decrease as the network learns better parameters to describe the relationship between the input and the associated labels.

Figures Fig. (5.2) and Fig. (5.3) show the progression of the learning process. In the first epochs, Fig. 5.2a and Fig. 5.3a, the scattering and mixing-up of the pixels show poor pixel localisation and classification consistent with CNN learning in the early stages. However, as the learning progresses through the intermediate epochs, figures Fig. (5.2b), and Fig. (5.3b) to Fig. (5.2e) and Fig. (5.3e) the network eventually builds an understanding of the relationship between the input and the associated groundtruths and the pixel classification

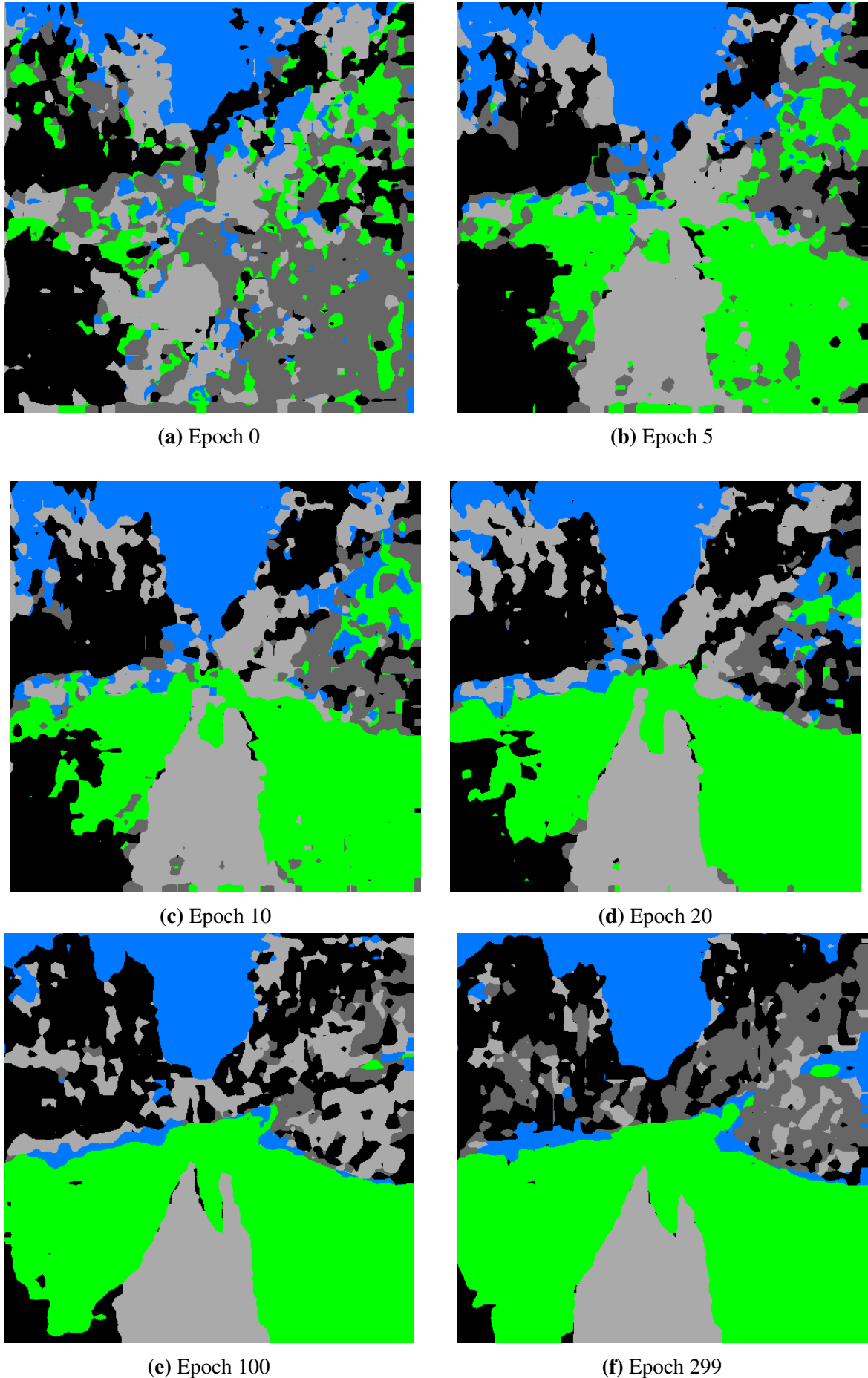
improves.

Furthermore, it can also be seen that there is not much improvement from epochs between 100 and 300, as seen in figures Fig. (5.2e), Fig. (5.3e), Fig. (5.2f) and Fig. (5.3f) on both datasets. To explain this phenomena in detail, we use the convergence curves.

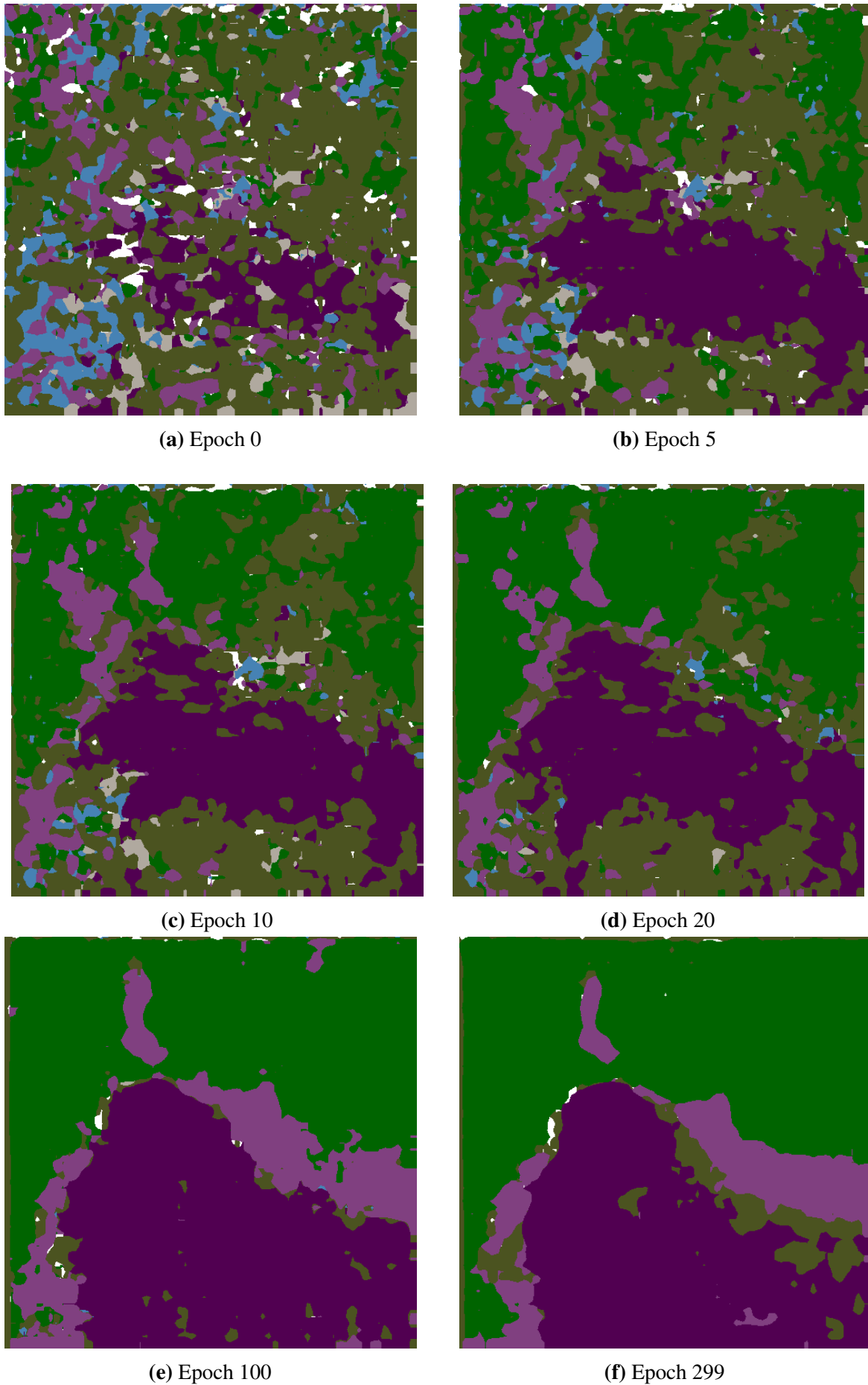
Fig. (5.4) and Fig. (5.5) show the performance plots for both datasets. We experimented with different variations of the number of epochs hyperparameter on all architectures. Recall that the learning process aims to establish the best possible parameters for describing the relationship between the inputs and targets. If we consider the parameter space, convergence is the equivalent of finding the global minima. This is the point where the gradient of the error with respect to the parameters is at its lowest. This also means that the error would have reached its lowest possible values. Ideally, this would be zero. This was not the case with our findings.

Empirically, we found that 300 was the most optimal number of epochs we needed to reach satisfactory results on all architectures. The effects of using transfer learning and ResNet pretrained weights is exhibited by the drastic decrease in the loss in the first 0-50 epochs on all architectures. This is also supported by the drastic increase in the IoU and accuracy in those epochs. This is referred to as the convergence speed. The convergence speed would be much slower and gradual when using other network initialisation techniques such as random initialisation. The learning process plateaued before 300 epochs for all architectures on both datasets. However, unlike in an ideal learning situation, the losses did not decrease all the way to regions near zero. Neither did the mean IoUs increase to the highest possible values. This phenomenon indicates convergence to a local minima as opposed to global minima.

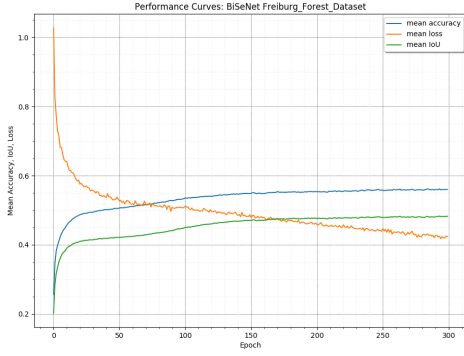
Convergence to a local minima can be a result of various factors such as the optimisation approaches used or the size of the dataset. In this case we used RMSprop and Adam optimisers. In past works these have been shown to be efficient and theoretically sound techniques that ensures convergence to global minima. This implies that, as expected, the sizes of both datasets used in this research were not sufficiently large to ensure that the CNN models learn the most optimal parameters for describing the relationship between the inputs and the associated targets, since all architectures exhibited similar convergence phenomena. Additionally, convergence to a local minima can also be indicative of low model capacity which results in model inability to learn the optimal parameters.



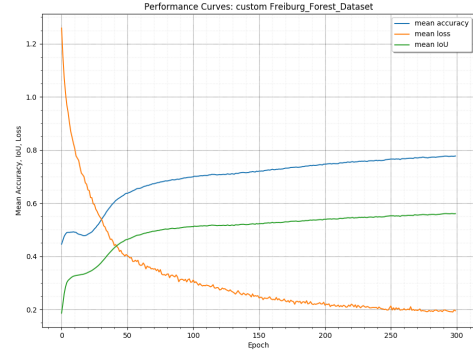
**Figure 5.2** Progression of the learning process through the segmentation networks. Examples from the BiSeNet architecture on the Freiburg Forest dataset.



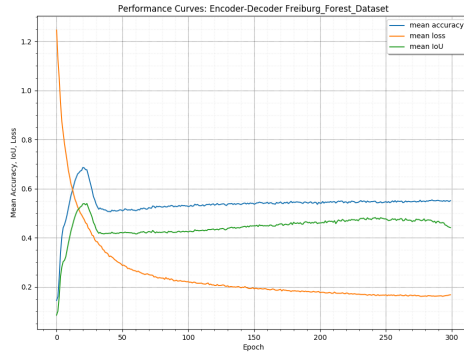
**Figure 5.3** Progression of the learning process through the segmentation networks. Examples from the BiSeNet architecture on the CSIR dataset.



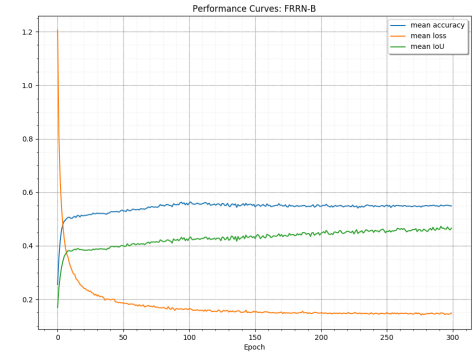
(a) BiSeNet



(b) Custom



(c) Encoder-Decoder



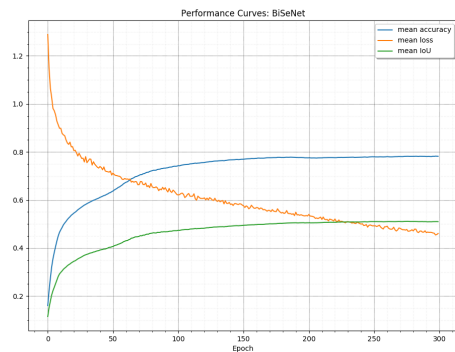
(d) FRRN-B

**Figure 5.4** Performance plots for the top 4 models on Freiburg Forest dataset. All models were initialised using ResNet pretrained weights, trained with a learning rate of 0.00001 for 300 epochs with the RMSProp optimiser. The effects of pretrained weights can be seen by the fast convergence speed towards a local minima.

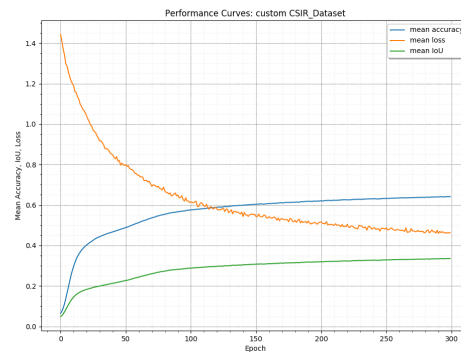
The convergence for the other architectures used in this research are presented in the extended results section. In the next section we summarise all our results with respect to the research questions we intended to provide answers to.

### 5.4.2 Convergence - extended results

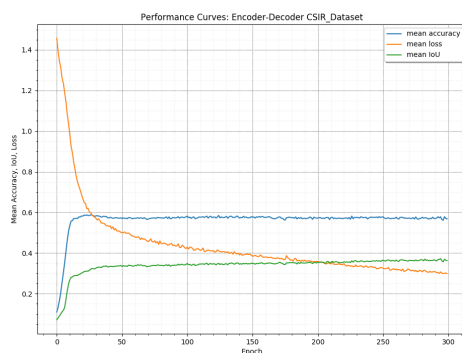
In Fig. (5.6) and (5.7) we show the convergence graphs from the remainder of the models. These models are AdapNet, FRRN-A, Encoder-Decoder with skip connections, PSPNet, MobileUNet and MobileUNet with skip connections. These models were also trained and tested on both the CSIR and Freiburg Forest datasets following the same procedure as top-four models in the above sections. The convergence of the rest of the models is consistent with the



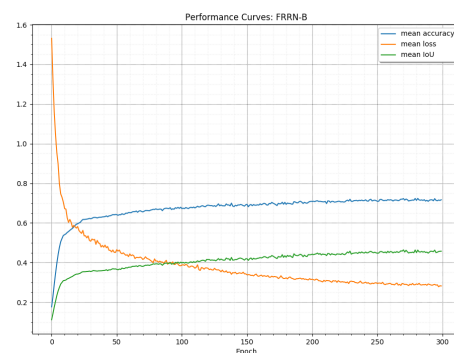
(a) BiSeNet



(b) Custom



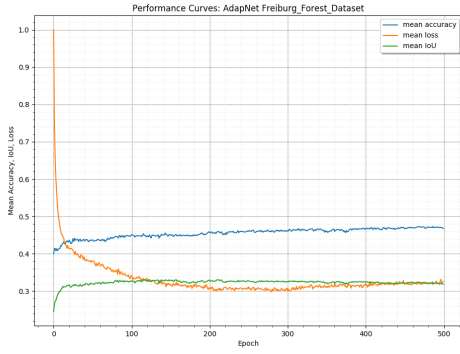
(c) Encoder-Decoder



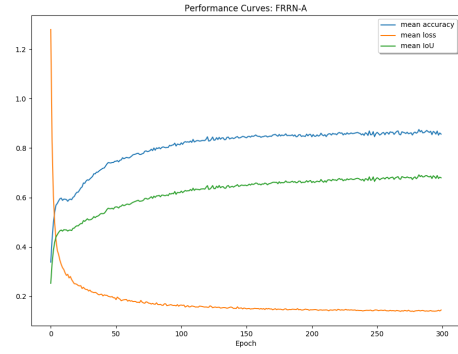
(d) FRRN-B

**Figure 5.5** Performance plots for the top 4 models on CSIR dataset. All models were initialised using ResNet pretrained weights, trained with a learning rate of 0.00001 for 300 epochs with the RMSProp optimiser. The effects of pretrained weights can be seen by the fast convergence speed towards a local minima.

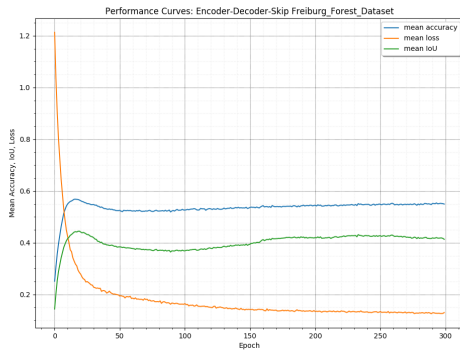
convergence behaviour of the top-four models discussed above. This can also be attributed to the use of pre-trained weights for network initialisation.



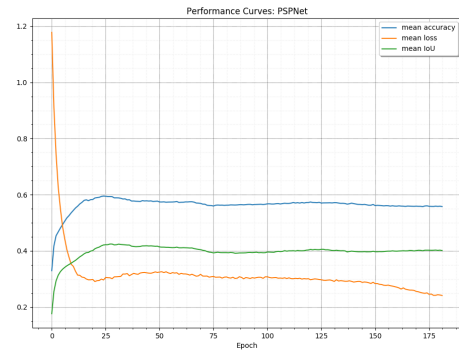
(a) AdapNet



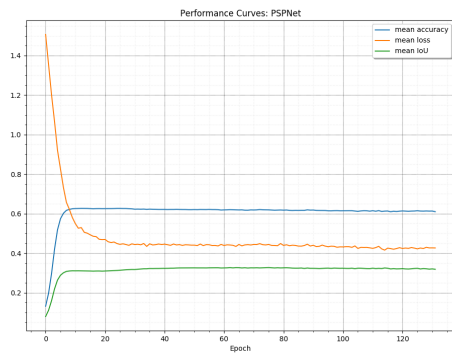
(b) FRRN-A



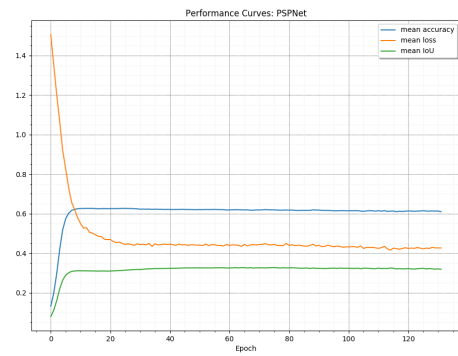
(c) Encoder-Decoder-Skip



(d) PSPNet

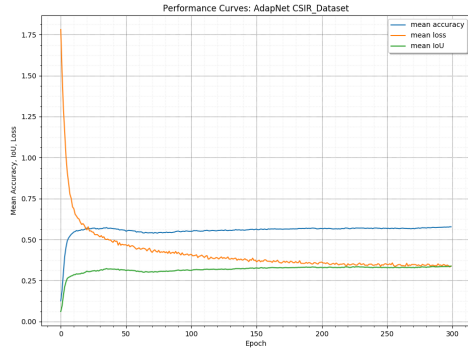


(e) MobileUNet

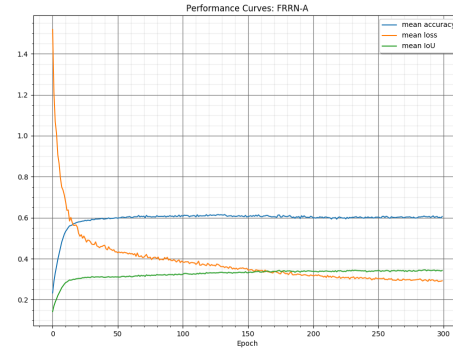


(f) MobileUNet-Skip

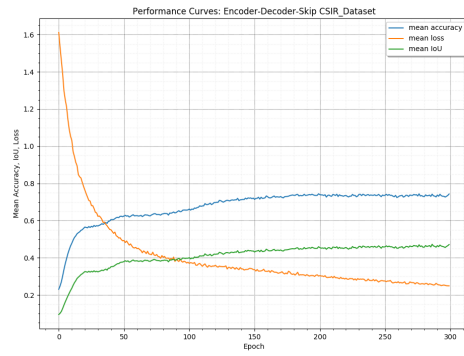
**Figure 5.6** Performance plots, models on Freiburg Forest dataset, extended results



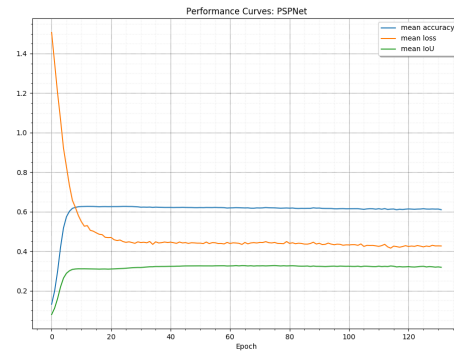
(a) AdapNet



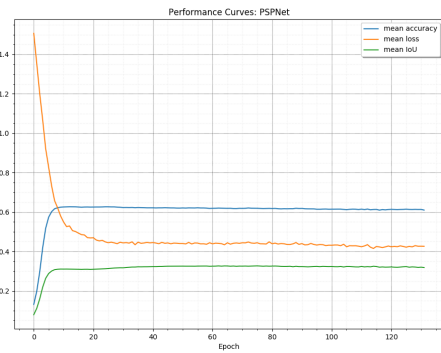
(b) FRRN-A



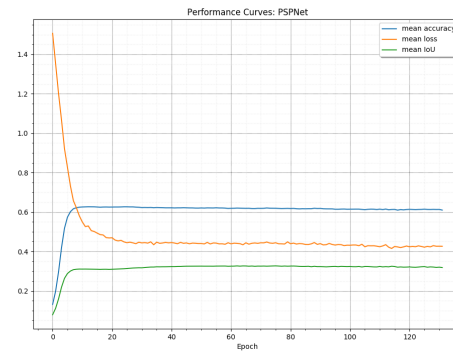
(c) Encoder-Decoder-Skip



(d) PSPNet



(e) MobileUNet



(f) MobileUNet-Skip

**Figure 5.7** Performance plots, models on CSIR dataset, extended results



## 5.5 Summary

In this chapter we provided exhaustive discussion and analysis of the top four CNN architectures on the CSIR and Freiburg Forest datasets. Recalling the objectives of this research: to perform semantic segmentation on a novel CSIR dataset, to find the best semantic segmentation CNN architecture for unstructured environments as well as to establish whether transfer learning can be used for multi-class semantic segmentation to achieve state-of-the-art results on limited datasets in unstructured environments. We included an additional dataset: Freiburg Forest to reach a more sound empirical conclusion.

We performed both qualitative and quantitative comparisons and analysis of the leading state-of-the-art architectures for semantic segmentation based on CNN architectural factors, dataset factors, and the intended application. Both architectural and dataset factors are crucial. Moreover, the quality and granularity of input image labels is also a crucial factor. After considering all factors critical for answering our research questions or satisfying our research objectives we empirically conclude that the BiSeNet architecture is the best CNN architecture for semantic segmentation in unstructured outdoor environments.

All the state-of-the-art CNN architectures used in this research were originally intended for structured city environments and they do produce better results in those domains, both qualitatively and quantitatively. Furthermore, it is important to note that these results are only a step forward towards successful perception in autonomous navigation. However, more research still needs to be conducted for more efficient and real-time application of these results. Even though the BiSeNet architecture outperformed all the other CNN architectures it is still far from being a reliable approach for real-world applications. This can be shown by the processing speeds and the frame rates.

In summary, we are able to provide answers to all of our research questions, albeit, some of them not distinctively definitive. In the next chapter we summarise the contents of this dissertation as well as comment on future research directions.



# Chapter 6

## Conclusion

### 6.1 Conclusion

The adoption of artificial intelligence solutions into the everyday human life has been growing at an increasing rate. Autonomous navigation is one of the research areas that have intensified in recent years. The main theme of this work was on perception for autonomous vehicles using RGB images. There is a plethora of literature on research efforts that were aimed at solving this problem. However, the majority of past research efforts focused on structured urban/city terrains. The available literature showed evidence of little to no research efforts being directed towards unstructured non-urban or rural terrains. The focus of this research was therefore targeted towards perception, scene comprehension, for autonomous ground vehicles in these adverse and unstructured terrains. In this regard, we provided our own novel CSIR dataset. Additionally, we used the Freiburg Forest dataset further for comparisons and bench-marking.

In Chapter (2) and Chapter (3) we provided an extensive literature survey of SOTA deep learning architectures for semantic segmentation. Furthermore, we reviewed the main datasets that these SOTA methods were trained and tested on. These SOTA methods were originally designed for urban settings. Chapter (4) presented our methodology and our overall workflow on how to adapt these architectures in unstructured non-urban environments. Moreover, our methodology also outlined how we were going to compare the performance of these methods on the CSIR and Freiburg Forest datasets based on both architectural and dataset factors.

Perception plays a crucial role in autonomous navigation problems. The initial objective of this research was to develop a perception model to assist with mapping and localisation for autonomous ground vehicles using traditional computer vision methods. These meth-

ods use prescribed or hand-crafted feature descriptors such as SURF and SIFT. The use of hand-crafted features has limitations in terms of robustness and generalisation in image classification. This is an even greater challenge in unstructured environments.

On the other hand, deep learning approaches have been proven to be more superior at this task because they use different heuristics for feature extraction as opposed to using hand-crafted features. In this research we used a deep learning approach called semantic segmentation or pixel-level classification towards scene comprehension of unstructured non-urban domains. Chapter (6) provided a detailed account of our findings through both qualitative and quantitative analysis and discussion. More specifically we were able to provide answers to our research questions. In summary, after the analysis of results in Chapter (6), we can now provide answers to the questions we asked in Chapter (1). Precisely, in the introduction chapter we posed following four research questions:

1. Using Convolutional Neural Networks (CNNs) and colour RGB images, is it possible to segment an unstructured scene according to its multiple semantic classes: **dirt\_road**, **grass**, **trees**, **sky**, and **void**?

We were able to show that it is indeed possible to perform pixel-level segmentation of an unstructured scene using RGB images and CNNs.

2. Given limited resources and time constraints for data collection and annotation, are there benefits in leveraging transfer learning and weights from models trained on images from other datasets for related tasks in order to improve semantic segmentation results in unstructured adverse environments?

We also showed that when faced with the challenge of limited resources for data collection and labelling efforts it is beneficial to leverage transfer learning and pre-trained weights. We were able to produce satisfactory results even though our CSIR data was arguably too small (Section 3.3) to fully harness the power of deep learning. Alternatives would involve initialising the weight matrix from scratch using different heuristics. No matter the heuristic, the usage of transfer learning and pre-trained weights produces far more superior segmentation results. The use of pretrained weights also helps with reducing the amount of training time needed for an end-to-end segmentation network.

3. What are the key factors necessary for optimal results for the task of pixel-level segmentation in unstructured outdoor domains?

To answer this question we split all the main factors into two broad categories: dataset factors and CNN architectural factors. We paid particular attention to architectural factors such as network topology, number of convolutional layers, pooling strategies, number of layers, and classification strategies. For the dataset factors we focused on label granularity, dataset size, class distribution as well as domain and contrast. After comparing and contrasting all the chosen CNN architectures on the CSIR and Freiburg Forest datasets, we found that dataset factors are of primary importance (Section. 5.5). They play a crucial role on how any CNN architecture can be fine tuned for better performance on any specific dataset. This was empirically confirmed by how most architectures performed a lot better on the Freiburg Forest dataset compared to the CSIR

4. What is the best, or most efficient, CNN architecture for real-time semantic segmentation in unstructured outdoor environments?

Secondary to dataset factors are the CNN architectural factors. Taking into consideration the nature of input images for unstructured outdoor domains our results showed that the BiSeNet architecture (Section. 5.5) is most suitable for semantic segmentation in unstructured environments. In general, architectures that use two processing streams: one for maintaining a global context of features and the other for extracting low-level features simultaneously perform far better than methods that only rely on a series of convolutions and feature pooling.

## 6.2 Future research

In light of the findings provided in the results section above, further research is inevitable. Future research directions will be aimed at reducing the dependency on labelled data and label granularity. This can be achieved through the use of techniques such as **unsupervised pre-training** [36, 112]. More advanced approaches for transfer learning such as **adversarial domain adaptation** [23] can also be used for more efficient transfer of weights learned on one task to similar tasks. The majority of CNN based learning approaches use passive learning where they passively go through all training samples. **Active learning** is an approach that prioritises samples that maximise entropy when learning a task [20, 143]. This

could be a better learning approach and will be considered in future research.

# Bibliography

- [1] Larry F Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain research bulletin*, 50(5-6):303–304, 1999.
- [2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, Sabine Süsstrunk, et al. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.
- [3] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.
- [4] Anurag Arnab and Philip HS Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *CVPR*, volume 1, page 5, 2017.
- [5] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, Dec 2017. ISSN 0162-8828. doi: 10.1109/TPAMI.2016.2644615.
- [6] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.
- [7] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.
- [8] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.
- [9] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.
- [10] Jeffrey S Beis and David G Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *cvpr*, volume 97, page 1000. Citeseer, 1997.
- [11] Asa Ben-Hur, David Horn, Hava T Siegelmann, and Vladimir Vapnik. Support vector clustering. *Journal of machine learning research*, 2(Dec):125–137, 2001.

- [12] Piotr Bilinski and Victor Prisacariu. Dense decoder shortcut connections for single-pass semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6596–6605, 2018.
- [13] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. Visual recognition with humans in the loop. In *European Conference on Computer Vision*, pages 438–451. Springer, 2010.
- [14] Xavier Bresson, Selim Esedoğlu, Pierre Vandergheynst, Jean-Philippe Thiran, and Stanley Osher. Fast global minimization of the active contour/snake model. *Journal of Mathematical Imaging and vision*, 28(2):151–167, 2007.
- [15] Claude R Brice and Claude L Fennema. Scene analysis using regions. *Artificial intelligence*, 1(3-4):205–226, 1970.
- [16] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, xx(x): xx–xx, 2008.
- [17] Joao Carreira and Cristian Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 7(7):1312–1328, 2011.
- [18] Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8): 1026–1038, 2002.
- [19] Tony F Chan and Luminita A Vese. Active contours without edges. *IEEE Transactions on image processing*, 10(2):266–277, 2001.
- [20] Cheng Chen, Qi Dou, Hao Chen, and Pheng-Ann Heng. Semantic-aware generative adversarial nets for unsupervised domain adaptation in chest x-ray segmentation. In *International Workshop on Machine Learning in Medical Imaging*, pages 143–151. Springer, 2018.
- [21] Liang-Chia Chen, Dinh-Cuong Hoang, Hsien-I Lin, and Thanh-Hung Nguyen. Innovative methodology for multi-view point cloud registration in robotic 3d object scanning and reconstruction. *Applied Sciences*, 6(5):132, 2016.
- [22] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.
- [23] Yi-Hsin Chen, Wei-Yu Chen, Yu-Ting Chen, Bo-Cheng Tsai, Yu-Chiang Frank Wang, and Min Sun. No more discrimination: Cross city adaptation of road scene segmenters. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1992–2001, 2017.



- [24] Mei Yeen Choong, Wei Leong Khong, Wei Yeang Kow, Lorita Angeline, and Kenneth Tze Kin Teo. Graph-based image segmentation using k-means clustering and normalised cuts. In *2012 Fourth International Conference on Computational Intelligence, Communication Systems and Networks*, pages 307–312. IEEE, 2012.
- [25] Aaron Clauset. A brief primer on probability distributions. In *Santa Fe Institute*, 2011.
- [26] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5): 603–619, 2002.
- [27] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [28] Pdraig Cunningham and Sarah Jane Delany. k-nearest neighbour classifiers. *Multiple Classifier Systems*, 34(8):1–17, 2007.
- [29] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.
- [30] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.
- [31] Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.
- [32] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul): 2121–2159, 2011.
- [33] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006.
- [34] Nikita Dvornik, Konstantin Shmelkov, Julien Mairal, and Cordelia Schmid. Blitznet: A real-time deep network for scene understanding. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4174–4182, 2017.
- [35] Ian Endres and Derek Hoiem. Category-independent object proposals with diverse ranking. *IEEE transactions on pattern analysis and machine intelligence*, 36(2): 222–234, 2014.
- [36] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.
- [37] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

- [38] Heng Fan and Haibin Ling. Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. *arXiv preprint arXiv:1708.00153*, 2017.
- [39] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.
- [40] Jiashi Feng, Bingbing Ni, Qi Tian, and Shuicheng Yan. Geometric l-p-norm feature pooling for image classification. In *CVPR 2011*, pages 2609–2704. IEEE, 2011.
- [41] Jacob Fraden. *Handbook of modern sensors: physics, designs, and applications*. Springer Science & Business Media, 2004.
- [42] Lex Fridman, Daniel E Brown, Michael Glazer, William Angell, Spencer Dodd, Benedikt Jenik, Jack Terwilliger, Aleksandr Patsek, Julia Kindelsberger, Li Ding, et al. Mit advanced vehicle technology study: Large-scale naturalistic driving study of driver behavior and interaction with automation. *IEEE Access*, 2019.
- [43] Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40, 1975.
- [44] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.
- [45] William A Gardner. Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique. *Signal processing*, 6(2):113–133, 1984.
- [46] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.
- [47] Alan E Gelfand and Adrian FM Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409, 1990.
- [48] Jean Charles Gilbert and Jorge Nocedal. Global convergence properties of conjugate gradient methods for optimization. *SIAM Journal on optimization*, 2(1):21–42, 1992.
- [49] JP Givler. Occam’s razor and mendel’s peas. *Journal of the Elisha Mitchell Scientific Society*, 49(1):108–112, 1933.
- [50] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
- [51] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.
- [52] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

- [53] Chunhui Gu, Joseph J Lim, Pablo Arbeláez, and Jitendra Malik. Recognition using regions. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1030–1037. IEEE, 2009.
- [54] Jia He, Chang-Su Kim, and C-C Jay Kuo. *Interactive Segmentation Techniques: Algorithms and Performance Evaluation*. Springer Science & Business Media, 2013.
- [55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- [56] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [57] Xuming He, Richard S Zemel, and Miguel Á Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE, 2004.
- [58] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.
- [59] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [60] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [61] Paul Jaccard. Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines. *Bull Soc Vaudoise Sci Nat*, 37:241–272, 1901.
- [62] Anil K Jain, Jianchang Mao, and KM Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 3(3):31–44, 1996.
- [63] Anil K Jain, Alexander Topchy, Martin HC Law, and Joachim M Buhmann. Landscape of clustering algorithms. In *null*, pages 260–263. IEEE, 2004.
- [64] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, pages 2146–2153. IEEE, 2009.
- [65] Simon Jégou, Michal Drozdal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 11–19, 2017.
- [66] BK Jenkins. Research in optical symbolic computing tasks. Technical report, UNIVERSITY OF SOUTHERN CALIFORNIA LOS ANGELES SIGNAL AND IMAGE PROCESSING INST, 1988.

- [67] Joerg Kappes, Bjoern Andres, Fred Hamprecht, Christoph Schnorr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Bernhard Kausler, Jan Lellmann, Nikos Komodakis, et al. A comparative study of modern inference techniques for discrete energy minimization problems. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1328–1335, 2013.
- [68] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.
- [69] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [70] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.
- [71] Pushmeet Kohli, M Pawan Kumar, and Philip HS Torr.  $P^3$  & beyond: Move making algorithms for solving higher order functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(9):1645–1656, 2009.
- [72] Andrei Nikolaevich Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Doklady Akademii Nauk*, volume 114, pages 953–956. Russian Academy of Sciences, 1957.
- [73] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.
- [74] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
- [75] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.
- [76] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On measuring the accuracy of slam algorithms. *Autonomous Robots*, 27(4):387, 2009.
- [77] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Dec 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541.
- [78] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

- [79] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [80] Yann LeCun et al. Generalization and network design strategies. In *Connectionism in perspective*, volume 19. Citeseer, 1989.
- [81] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.
- [82] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2290–2297, Dec 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2009.96.
- [83] Xiaoxiao Li, Ziwei Liu, Ping Luo, Chen Change Loy, and Xiaoou Tang. Not all pixels are equal: difficulty-aware semantic segmentation via deep layer cascade. *arXiv preprint arXiv:1704.01344*, 2017.
- [84] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *Proceedings. international conference on image processing*, volume 1, pages I–I. IEEE, 2002.
- [85] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934, 2017.
- [86] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.
- [87] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E Haque, Lingjia Tang, and Jason Mars. The architectural implications of autonomous driving: Constraints and acceleration. In *ACM SIGPLAN Notices*, volume 53, pages 751–766. ACM, 2018.
- [88] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [89] Ce Liu, Jenny Yuen, and Antonio Torralba. Nonparametric scene parsing via label transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2368–2382, 2011.
- [90] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):978–994, 2011.
- [91] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

- [92] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
- [93] David G Lowe et al. Object recognition from local scale-invariant features. In *iccv*, volume 99, pages 1150–1157, 1999.
- [94] Jiangbo Lu, Hongsheng Yang, Dongbo Min, and Minh N Do. Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1854–1861, 2013.
- [95] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 1, page 3, 2013.
- [96] Artur Maligo and Simon Lacroix. Classification of outdoor 3d lidar data based on unsupervised gaussian mixture models. In *IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2015.
- [97] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE transactions on pattern analysis and machine intelligence*, 26(5):530–549, 2004.
- [98] Kevin McGuinness and Noel E O’connor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 43(2):434–444, 2010.
- [99] Luigi Federico Menabrea and Ada Lovelace. Sketch of the analytical engine invented by charles babbage, 1842.
- [100] Branislav Mičušík and Jana Košecká. Semantic segmentation of street scenes by superpixel co-occurrence and 3d geometry. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 625–632. IEEE, 2009.
- [101] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. ISBN 0070428077, 9780070428072.
- [102] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fast-slam: A factored solution to the simultaneous localization and mapping problem. In *Aaai/iaai*, pages 593–598, 2002.
- [103] Greg Mori. Guiding model search using segmentation. In *null*, pages 1417–1423. IEEE, 2005.
- [104] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.
- [105] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5): 1147–1163, 2015.

- [106] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.
- [107] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Buló, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4990–4999, 2017.
- [108] Andreas Nüchter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926, 2008.
- [109] Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. 6d slam—3d mapping outdoor environments. *Journal of Field Robotics*, 24(8-9):699–722, 2007.
- [110] Ron Ohlander, Keith Price, and D Raj Reddy. Picture segmentation using a recursive region splitting method. *Computer Graphics and Image Processing*, 8(3):313–333, 1978.
- [111] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.
- [112] Tom Le Paine, Pooya Khorrami, Wei Han, and Thomas S Huang. An analysis of unsupervised pre-training in light of recent advances. *arXiv preprint arXiv:1412.6597*, 2014.
- [113] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.
- [114] Lina M Paz, Juan D Tardós, and José Neira. Divide and conquer: EKF slam in  $o(n)$ . *IEEE Transactions on Robotics*, 24(5):1107–1120, 2008.
- [115] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters—improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4353–4361, 2017.
- [116] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.
- [117] John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.
- [118] Tobias Pohlen, Alexander Hermans, Markus Mathias, and Bastian Leibe. Full-resolution residual networks for semantic segmentation in street scenes. *CoRR*, abs/1611.08323, 2016. URL <http://arxiv.org/abs/1611.08323>.
- [119] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

- [120] Simon JD Prince. *Computer vision: models, learning, and inference*. Cambridge University Press, 2012.
- [121] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2017.
- [122] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016.
- [123] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- [124] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.
- [125] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.
- [126] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization?(no, it is not about internal covariate shift). *arXiv preprint arXiv:1805.11604*, 2018.
- [127] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.
- [128] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.
- [129] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer Science & Business Media, 2008.
- [130] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL <http://arxiv.org/abs/1409.1556>.
- [131] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [132] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Robotics: Science and Systems*, volume 2, pages 65–72, 2005.



- [133] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. On the importance of initialization and momentum in deep learning. *ICML (3)*, 28(1139-1147):5, 2013.
- [134] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [135] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.
- [136] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [137] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.
- [138] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.
- [139] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.
- [140] Jonas Uhrig, Marius Cordts, Uwe Franke, and Thomas Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition*, pages 14–25. Springer, 2016.
- [141] Abhinav Valada, Gabriel Oliveira, Thomas Brox, and Wolfram Burgard. Deep multispectral semantic scene understanding of forested environments using multimodal fusion. In *International Symposium on Experimental Robotics (ISER)*, 2016.
- [142] Abhinav Valada, Johan Vertens, Ankit Dhall, and Wolfram Burgard. Adapnet: Adaptive semantic segmentation in adverse environmental conditions. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 4644–4651. IEEE, 2017.
- [143] Alexander Vezhnevets, Joachim M Buhmann, and Vittorio Ferrari. Active learning for semantic segmentation with expected change. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3162–3169. IEEE, 2012.
- [144] Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 6(6):583–598, 1991.

- [145] Paul Viola and Michael Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *Advances in neural information processing systems*, pages 1311–1318, 2002.
- [146] Max Wertheimer. Gestalt theory. 1938.
- [147] Max Wertheimer. Laws of organization in perceptual forms. 1938.
- [148] Maoke Yang, Kun Yu, Chi Zhang, Zhiwei Li, and Kuiyuan Yang. Denseaspp for semantic segmentation in street scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3684–3692, 2018.
- [149] Faliu Yi and Inkyu Moon. Image segmentation: A survey of graph-cut methods. In *Systems and Informatics (ICSAI), 2012 International Conference on*, pages 1936–1941. IEEE, 2012.
- [150] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.
- [151] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 325–341, 2018.
- [152] Matthew D Zeiler. Adadelata: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.
- [153] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.
- [154] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.
- [155] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–420, 2018.
- [156] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [157] Hongyuan Zhu, Fanman Meng, Jianfei Cai, and Shijian Lu. Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation. *Journal of Visual Communication and Image Representation*, 34:12–27, 2016.
- [158] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *Proc. CVPR*, number 6, page 7, 2017.