THE UNIVERSITY OF SYDNEY

The University of Sydney

School of Aerospace, Mechanical and Mechatronic Engineering

Forest for the Trees: Semantic Segmentation of LiDAR Point Clouds in Complex Forest Environments

Author:
Cassandra Sargeant

Supervisor:
Dr. Mitch Bryson

A thesis submitted in fulfilment of the requirements for the degree of Bachelor of Engineering

November 2023

Abstract

Forests are a key part of the Australian ecosystem as well as being a major industry. They contain a large amount of biodiversity and remove massive amounts of carbon dioxide from the atmosphere each year. In order to effectively manage forests it is important to be able to measure key properties such as the abundance and distribution of trees. This was once achieved with hand measurements but now the use of LiDAR to gather point clouds for analysis is common. LiDAR is used to scan forests and determine the number of trees, their heights, the amount of wood, and other key attributes. The point clouds are gathered and used with machine learning networks to perform semantic segmentation to get this understanding of the forest. A major challenge in this task is that the networks are designed to take point clouds that have thousands of points, while the point clouds for most forests exceed many millions of points. This can be handled be splitting up the point cloud into a number of smaller sections or windows. The question of how to generate these windows in a way that maximises the efficacy of the network is important in improving the management of our forests.

This thesis aims to improve understanding of how the applying data pre-processing methods on highly unstructured forest point clouds can improve results for semantic segmentation. One existing technique used in literature is the use of a sliding window that moves along the xy-plane of the plot and takes samples of the contained area. The size of the window used has been arbitrary. This thesis explores the possibility of a relationship between the size of the window and overall results for a PointNet++ based network performing four label semantic segmentation with the labels 'Foliage', 'Stem', 'Ground', and 'Undergrowth'. We develop a methodology for sampling from the point cloud and generating training data with augmentations and testing data. We then use these to train various models and test them on a separate dataset to determine how changing the window width used for generating the datasets affects the overall results.

Through these experiments, we are able to determine that on the datasets used the best window size is approximately 1.0m x 1.0m. The fall in results before and after the 1.0m window indicates that there is a link between the window size used and the results of the network, with a possible link between ideal window size and the structure of the forest. This suggests that it is preferable for the network to have a number of smaller more dense point clouds rather than larger more sparse point clouds, at least until a lower limit. The best IoUs achieved by the model were: Ground 0.070, Foliage 0.904, Stem 0.664, and Undergrowth 0.837, with an overall accuracy of 92.1%.

Statement of Contribution

- I carried out the literature review in order to understand existing approaches to semantic segmentation of forest point clouds and the various deep learning networks that have been used to process point clouds, with guidance from my supervisor.
- I designed and implemented a methodology for splitting up the point cloud into more easily processed parts based off of the work of Windrim and Bryson [1] in using a sliding window in the xy-plane of the point cloud. I ran this a number of times while varying the window widths in order to gather training, validation, and testing datasets.
- I modified David Griffiths implementation [2] of PointNet++ [3] from a much older Tensorflow and CUDA version so it was suitable to work with the datasets generated, would use the metrics needed for evaluation, and would save weights and metrics generated.
- I configured my own hardware so it would be capable of training these models, installing Ubuntu, CUDA, cuDNN, and Tensorflow.
- I trained a number of models based on this architecture, each one being trained on the same training dataset that was split using a different window width.
- I carried out testing of the trained models on the variety of test datasets generated in order to establish a relationship between window size and model performance.
- I generated all plots, visualisations, and various data transformations needed for the presentation of results.
- I carried out the discussion and conclusions of the work based on the results, literature, and discussion with my supervisor.

Cassandra Sargeant Date

Acknowledgements

Firstly I would like to thank my academic supervisor Dr. Mitch Bryson for being such a supportive, helpful, and fun supervisor to work with. Your continual guidance and helpful advice made this thesis possible and stopped me from losing the plot more than once. Thank you for all your enthusiasm and consistency, I'm grateful.

To my family, thank you for listening to me talk about trees and LiDAR and other nonsense for a whole year, and all the other mechatronics challenges I've loudly struggled through over my degree. You've kept me fed, sane, and loved. Thank you for supporting me always.

To my fellow thesis students Jack and Lewis, its been great having you all to work alongside. Thank you for your help in throwing ideas back and forth, in practising seminar presentations, and convincing each other we'll be done on time. Having your company in this journey made it all the easier.

To my friends Jonah, Ben, Alex, and Conan, who have been there since we were little Year 7s. We've all changed a lot since those days, but your friendship has stayed the same. Thank you for the past decade of support.

Finally, to Bella, thank you for being by my side. Your love, kindness, and support through this thesis and through all of life's challenges has kept me going. I couldn't have done it without you, and if I could, I wouldn't want to. I love you.

List of Figures

2.1 Classification of various LiDAR systems, adapted from [30]. 5

2.2 Examples of tools used for the inventory of trees in forests. 6

2.3 A point cloud from a recreational forest in Rotorua, New Zealand. . . . 7

2.4 Visual comparison of two types of forests, (a) a managed plantation and (b) a native forest 7

2.5 Comparison of digital surface models (DSM), digital terrain models (DTM) showing that: $DSM - DTM = CHM$. From [48]. 8

2.6 Example of how models are trained and then used in supervised machine learning. 10

2.7 Applications of PointNet [22] 11

2.8 Architecture of PointNet [22]. 12

2.9 Architecture of PointNet++, shown as applied in a 2D Euclidian space [3] 13

2.10 Density adaptive layers used by PointNet++ [3] 14

2.11 Application of RandLA-Net downsampling point clouds 16

2.12 3D visualisation of individually segmented trees within a forest using the methodology described by Qin et al. using AMS3D [19] 17

2.13 A graphical comparison between a voxel based 3D-FCN approach to segmentation and raw-point-based PointNet based approach [1] 18

2.14 Geometric feature balancing as performed by Shen et al. [15]. 19

2.15 Handcrafted and PointNet++ feature extraction used by Wang and Bryson [13] 21

3.1 Methodology for training PointNet++ 23

3.2 The dataset 'DogPark' used for training the models and the dataset 'HQP' used for testing the models. 24

3.3 2D depiction of the sliding window used 25

3.4 Visualisation of the Intersection-over-Union parameter 27

4.1 A summary of the experimental process. 29

4.2 Mean IoU while training models over 50 epochs. 30

4.3 Accuracy while using different window widths to train different models over 50 epochs. 31
4.4 Loss while using different window widths to train different models over 50 epochs. The loss metric used was cross-entropy. 31
4.5 Final per class IoU values for all window widths for both training and validation. 32
4.6 Final values of mean IoU for all window widths for both training and validation. 33
4.7 Final values of accuracy across all window widths for both training and validation. 33
4.8 Final values of loss across all window widths for both training and validation. 34
4.9 IoU results across various models trained on different window widths when using the testing dataset. 35
4.10 Accuracy results across various models trained on different window widths when using the testing dataset. 36
4.11 Loss results across various models trained on different window widths when using the testing dataset. The loss metric used was cross-entropy loss. 36
4.12 Confusion matrix on the testing dataset for a window width of 1.0m, with each box indicating the number of points that have been predicted with each label when compared to the true label. 37
4.13 Normalised confusion matrix on the testing dataset for a window width of 1.0m, with each box indicating the proportion of points that have been predicted with each label compared to their true labels. 38
4.14 Normalised IoU results across various models trained on different window widths when using the testing dataset. 39
4.15 Visualisation of predicted labels compared to the ground truth labels on a 0.5m window width example. 40
4.16 Visualisation of predicted labels compared to the ground truth labels on a 1.0m window width example. 41
4.17 Visualisation of predicted labels compared to the ground truth labels on a 4.5m window width example. 42
4.18 How often each label appears proportionally in the training data. 43
4.19 How often each label appears proportionally in the testing data. 43

1 Confusion matrix on the testing dataset for a window width of 0.5m . . 63
2 Confusion matrix on the testing dataset for a window width of 1.0m . . 64
3 Confusion matrix on the testing dataset for a window width of 1.5m . . 65
4 Confusion matrix on the testing dataset for a window width of 2.0m . . 66

5 Confusion matrix on the testing dataset for a window width of 2.5m . . 67
6 Confusion matrix on the testing dataset for a window width of 3.0m . . 68
7 Confusion matrix on the testing dataset for a window width of 3.5m . . 69
8 Confusion matrix on the testing dataset for a window width of 4.0m . . 70
9 Confusion matrix on the testing dataset for a window width of 4.5m . . 71

List of Tables

4.1 Window widths used to train various PointNet++ semantic segmentation models for testing. 30

4.2 Window width compared to size of data size after using the sliding window in the xy-plane when using a test dataset originally 604.2MB and training dataset originally 345.4MB 44

5.1 Best IoU values for different labels used in semantic segmentation of forest points across different papers. 47

Acronyms

Acronym	Meaning
ALS	Airborne Laser Scanning
AMS3D	Adaptive Mean Shift 3D
CHM	Canopy Height Model
CNN	Convolutional Neural Net
CUDA	Compute Unified Device Architecture
DBH	Diameter at Breast Height
DSM	Digital Surface Model
DTM	Digital Terrain Model
FCN	Fully Convolutional Network
FPS	Furthest Point Sampling
GNSS	Global Navigation Satellite System
IoU	Intersection-over-Union
kNN	k-Nearest Neighbours
LiDAR	Light Detection and Ranging
MLP	Multi-Layer Perceptron
MLS	Mobile Laser Scanning
MRG	Multi-Resolution Grouping
MSG	Multi-Scale Grouping
NLLS	Non-Linear Least Squares
RANSAC	Random Sample Consensus
SIFT	Scale-invariant Feature Transform
TLS	Terrestrial Laser Scanning

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Problem Statement 2

1.3 Aims and Objectives 2

1.4 Outline 2

2 Literature Review **4**

2.1 LiDAR for Sensing 4

2.1.1 Introduction to LiDAR Technology 4

2.1.2 How LiDAR Works: A Technical Overview 4

2.1.3 Advantages of LiDAR for Forest Landscapes 5

2.1.4 Early Techniques for Forest Inventory using Point Clouds . . . 8

2.2 Deep Learning for Point Clouds 9

2.2.1 An Overview of Supervised Machine Learning 9

2.2.2 Existing Methods for Deep Learning on Point Clouds 10

2.3 Applications of Deep Learning to Forest Point Clouds 16

2.4 Summary 21

3 Methodology **22**

3.1 Datasets 22

3.2 Sliding Window 24

3.3 Normalisation 25

3.4 Augmentation 25

3.5 Model Compilation and Training 26

3.5.1 Software and Hardware Configuration 26

3.5.2 Metrics 27

3.6 Limitations 27

3.7 Preliminary Tests 28

4 Results **29**

4.1 Training and Validation 29

4.2 Testing 34

4.3 Label Occurrence in Data 42

4.4 Computational Results 44

5 Discussion **45**

5.1 Summary of Results 45

5.2 Justification of Results 45

5.3 Model Strengths and Weaknesses 46

5.4 Key Findings 46

5.5 Relation to Literature 46

5.6 Limitations 47

6 Conclusions and Future Work **49**

6.1 Future Work 50

6.1.1 Alternative Point-Based Networks 50

6.1.2 Forest Variety 50

6.1.3 Window Geometry 50

6.1.4 Voting System 50

6.1.5 Bi-Density Windows 50

6.1.6 Alternative Environments 51

Appendices **61**

A Confusion Matrices 62

CHAPTER 1

Introduction

"Forests are the lungs of our land, purifying the air and giving fresh strength to our people." - Franklin D. Roosevelt

1.1 Motivation

Forests are a key ecosystem on our planet, containing most of Earth's terrestrial biodiversity. They contain a huge variety of species, are found all over the Earth, and cover 31% of the worlds landmass [4]. In the challenges that climate change poses, forests play a large role as a carbon sink, with the world's forests removing approximately 7.6 billion tonnes of CO_2 from the atmosphere per year [5]. In Australia, forests play a crucial role for biodiversity and the ecosystem [6], as well as being part of the nation's large forestry industry, contributing $9.2 billion in 2018 [7].

Having a technical, quantitative understanding of our forests is critical to the forestry industry in order to accurately understand inventory and to model future growth and timber return within an estate [8]. Similarly, forest inventory is invaluable for the conservation and management of native forests. Aspects measured for forest inventory typically include: number of trees, species, height, stem volume, sweep, and more structural traits that are of interest to forest managers. The use of different types of LiDAR (Light Detection and Ranging) scanning methods within recent years has provided point clouds (a type of 3D data structure) of forests that can be used to determine these measurements, which saves significant amounts of time over direct measurements [1, 9–21]. LiDAR scans that are commonly used for forests include airborne scanning that may be performed by aircraft or by drones, or terrestrial scanning that involves a system fixed to a tripod, or to some kind of vehicle.

The use of supervised machine learning methods has been shown to be effective in performing semantic segmentation on these point clouds, isolating the stems from other aspects of the forest such as foliage and undergrowth [1, 14, 15]. Previous works have successfully implemented deep learning methods onto point clouds, managing to take point sets and perform classification, part segmentation, and semantic segmentation. They adapt traditional deep learning techniques to work with these unordered sets of vectors, and try to account for interaction between points and any transformations the data may undergo [3, 22–29]. These neural networks have been used by a number of

researchers to work with forest point clouds, resulting in labelled point clouds from which key geometric parameters of the trees can be analysed [13–18].

1.2 Problem Statement

This thesis aims to investigate how to best prepare point cloud data in order to improve the results of a neural networks when applied to forest datasets. The literature surrounding this topic has a focus on the training of the models themselves and the ways that the neural nets can be adapted and improved to more effectively segment the point cloud. There is very little focus or discussion on how the point clouds are processed before being used in network. The point clouds cannot be directly used as they often contain many millions of points and the networks are only equipped to take points in the range of thousands. This challenge can be managed by using sections of the point cloud at a time rather than the entire point cloud at once, which makes the pre-processing phase necessary in many instances. The choices on how the point cloud is separated into smaller parts may have a significant impact on results, and this relationship has not yet been sufficiently explored in literature.

1.3 Aims and Objectives

The aim of this paper is to investigate how applying data pre-processing methods on highly unstructured forest point clouds can improve results for semantic segmentation. This will be achieved through comparing methods for using varying resolutions and sampling rates to separate the data into sections that can be processed by a network suitable for point cloud semantic segmentation. Understanding how changing pre-processing methods affects overall results allows future implementations to improve their efficacy.

The full methodology in order to achieve this aim is discussed in Chapter 3. In brief, we intend to experiment on separating the forest point clouds into sections and changing their size and resolution to determine whether smaller high resolution or larger low resolution areas grant the best semantic segmentation results using PointNet++ [3]. This will be tested across multiple datasets to compare the highly structured homogeneous commercial forests to the unstructured heterogeneous native forests. Semantic segmentation will be done using the labels stem, foliage, undergrowth, and ground.

1.4 Outline

Chapter 2 is the literature review, and aims to give a thorough understanding of the existing research surrounding LiDAR, point clouds, deep learning, and the various methods with which forests have been analysed in the past and present through traditional and now data driven methods.

Chapter 3 details the methodology that will be performed in pursuit of identifying the best methods for processing the point cloud data before using it with the neural net.

This includes key decisions on what is most important to test, how the experiments will be performed validly, and the limitations present.

Chapter 4 presents the results gained from the experiments numerically through tables and figures. It highlights key quantitative findings and outlines facts and trends that are evident from the information presented.

Chapter 5 is a discussion on the results shown in Chapter 4, hypothesising reasons behind trends and relating the significance of the numerical findings to the aims and objectives discussed in Chapter 1, and to existing literature.

Chapter 6 concludes this thesis by summarising the reasoning behind the study, its processes, and overall findings. Suggestions are made for improvements and avenues of future work.

CHAPTER 2

Literature Review

"The more that you read, the more things you will know. The more that you learn, the more places you'll go." - Dr. Seuss

2.1 LiDAR for Sensing

2.1.1 Introduction to LiDAR Technology

LiDAR uses the return time of a reflected laser to determine how far an object is from the scanner. This becomes highly effective in situations where you can quickly scan many points using the laser to generate a high resolution map of an environment, called a point cloud. Moving this LiDAR scanner in conjunction with a GPS receiver permits the creation of a map over a much larger area, which can then be used by researchers to examine the environment of interest in great detail [30].

The first experiments of a kind of 'laser radar' were seen in the early 1960s [31], soon seeing applications in meteorology and quickly changing the field [32, 33]. LiDAR has a tendency to rapidly and effectively reshape the field it is introduced to, as has been seen in agriculture [34], transportation [35], archaeology [36], and forestry [11].

2.1.2 How LiDAR Works: A Technical Overview

LiDAR bases its operational principles on the universal constant of the speed of light. By shining the concentrated laser to the target and measuring the time of return the distance to the target can be calculated using Equation 2.1 where,
$D =$ The distance from the LiDAR system to the target
$c =$ The speed of light
$\Delta T =$ Time difference from the light leaving and returning

$$D = c(\Delta T/2) \tag{2.1}$$

There are four main types of LiDAR systems that are used regularly, and they can be separated into two categories: airborne and terrestrial. These types are shown hierarchically in Fig. 2.1. Airborne LiDAR is designed to point the laser downwards

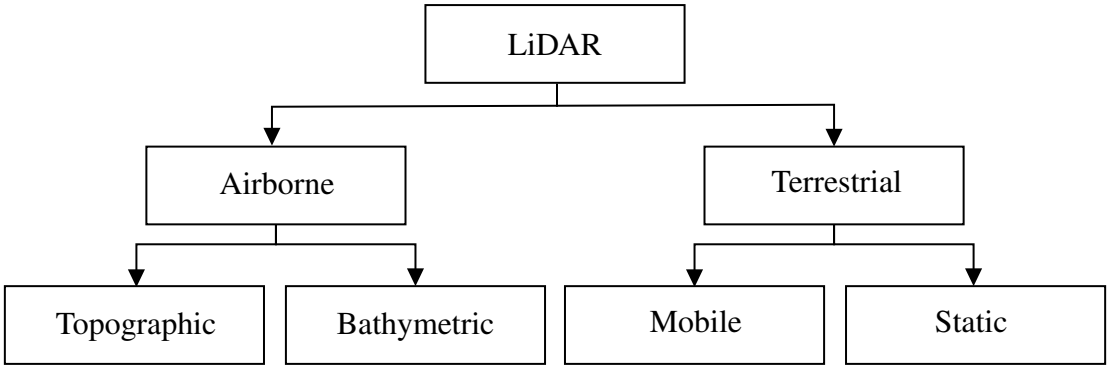


Figure 2.1: Classification of various LiDAR systems as either airborne (attached to some kind of aerial moving platform) or terrestrial (attached to a stationary or moving platform on the ground), adapted from [30].

and scan a large area as the aeroplane, helicopter, or drone flies overhead. Topographic LiDAR is used to find features of the land below (either natural or artificial) such as mountains, valleys, cities, or roads. This differs from bathymetric LiDAR which is designed for water penetration and determining features around or below the surface of a body of water. Terrestrial LiDAR is used on the ground, with mobile LiDAR being attached to some kind of vehicle along with a suite of other sensors, taking scans of the environment around it while the vehicle is driver around. Static LiDAR is typically a single LiDAR system mounted to a tripod mount, often being used for indoor areas or surveying. [30]

Bathymetric LiDAR is irrelevant for the purposes of forestry. Topographic LiDAR is referred to as ALS (Airborne Laser Scanning), mobile LiDAR as MLS (Mobile Laser Scanning), and static LiDAR as TLS (Terrestrial Laser Scanning). With ALS and MLS the scanners are moving over time, so a system is needed to align the scans made over a period of time and ensure the same object scanned at two different times are registered in the same location in the resulting point cloud. This can typically be achieved using GNSS (Global Navigation Satellite System) however there are often inaccuracies. A global registration system is used with an algorithm such as ICP (Iterative Closest Point) to align these point clouds, ensuring that the same tree across multiple scans appear in the same location in the combined scan [37].

2.1.3 Advantages of LiDAR for Forest Landscapes

LiDAR has a number of distinct advantages for 3D sensing that lends itself to being particularly effective for mapping forest landscapes. They are a non-destructive and highly accurate measuring technique, and it is often able to get information that is nearly impossible to get using other methods [38]. One early application of LiDAR in forests used MLS to scan trees and determine the key geometric parameters such as stem location, tree height, stem diameter at breast height (DBH), stem density, and timber volume. This was all done through manual segmentation of the point clouds and digital measurements, proving that the data can be extracted with reasonable accuracy from LiDAR measurements alone [39]. LiDAR is a very convenient method for gaining this data, especially when compared to previous traditional methods. In the past, measurements

of tree parameters were done by hand, where to measure the stem diameter of a tree an operator would need to visit the site directly and use a diameter tape, or using calipers depending on the size of the tree. Similarly the height of a tree would be measured by using a telescoping pole or with a handheld laser rangefinder [8], shown in Figure 2.2.

(a) Nikon Forestry Pro II, a compact laser rangefinder and height meter

(b) Fibreglass telescopic measuring poles for measuring the height of trees up to 8m

Figure 2.2: Examples of some tools used for the inventory of trees in forests. To learn the heights of every tree in an area accurately an operator would need to use tools like these rather than using a LiDAR scan. Images from [40].

If an observer wanted to extend these measurements from single trees into measuring the entire forest, it would be an extremely laborious and time consuming process. Alternate methods extend on this by using clever sampling techniques to approximate the properties of a local stand (a relatively uniform area of trees in species composition, structure, age or condition), or of an entire forest. These practices have been effective in the past, however it is time consuming, imprecise, and the formulas can vary dramatically depending on the species of tree [8]. LiDAR has been consistently proven over the years as a strong method for forest inventory, being proven over twenty years ago that it has potential to identify attributes of individual trees [9]. Since then its usages have only grown, being used with deep learning for identifying individual trees within a forest [10, 17–19], and for performing overall forest inventory estimations [11, 12, 21], all using a variety of TLS, MLS, and ALS data. LiDAR has seen such extensive usage as it is highly accurate, gives consistent results regardless of weather, and requires a smaller workforce compared to making measurements by hand. LiDAR acquisition of data is faster than other methods and can be scaled from a local stand into working with the entire forest, even when including time to process and sort the data. There is minimum operator interaction when collecting LiDAR data which dramatically reduces the opportunities to introduce human error into the measurements, however mistakes made while labelling point clouds is still a significant source of error [13,41]. Additional data can be gathered from LiDAR such as intensity of the laser return, which gives information about the surface that the laser reflected from. LiDAR can penetrate the canopy of trees, which allows for the collection of data from the sky while still being able to detect the topography underneath the canopies [42]. This is a significant advantage over other modern techniques such as photogrammetry which uses aerial photography, as even the most basic information about a forest such as the number of trees may be

trivial to determine in some managed plantations from images alone but can be near impossible to determine from images of a native forest due to their more complex irregular structure. A semantically segmented point cloud is shown in Fig. 2.3, and a visual comparison of managed forests to native forests in Fig. 2.4.

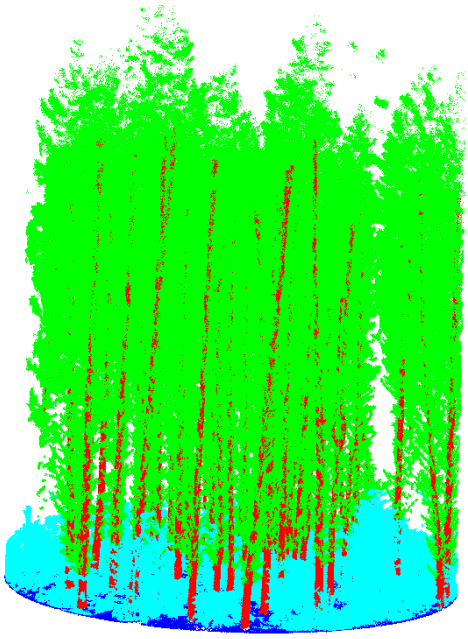


Figure 2.3: A point cloud from a recreational forest in Rotorua, New Zealand composed of species/single age class radiata pine, consisting of over 10 million points captured using MLS. It has been semantically segmented into stem, foliage, undergrowth and ground.

(a) Bluegum plantation in the Green Triangle in Victoria [43].

(b) Rainforest canopy at Barron River, Queensland. Photograph by Claire Howell [44].

Figure 2.4: Visual comparison of two types of forests, (a) a managed plantation where all the trees are able to be visually distinguished from each other and (b) a native forest where it is difficult to tell individual trees apart. This highlights the benefits of LiDAR in native forests to get aerial information about the forest underneath the canopy due to the penetrative qualities of LiDAR.

2.1.4 Early Techniques for Forest Inventory using Point Clouds

An early example of analysing forests using point clouds comes from J. Hyyppä and M. Inkinen in their 1999 paper "Detecting and estimating attributes for single trees using laser scanner" [9]. In this they demonstrated "for the first time that high-pulse-rate laser scanners are capable to detect single trees in boreal forest zone." [9]. This was built on a number of previous works that had focused on building digital surface models (DSM) and digital terrain models (DTM), as by using these together a good approximation can be made for the height of the trees. DTM are extracted by identifying which points from the LiDAR scan correspond with the terrain underneath the forest and using a predictive algorithm to smooth them together to create a surface [45]. The DTM is useful for its ability to correct the scans or images of an area into being a more even, consistent plane surface at the base, such that measurements the tree canopy can be used to create a canopy height model (CHM), shown in Fig. 2.5 [9, 46, 47]. In Hyyppä and

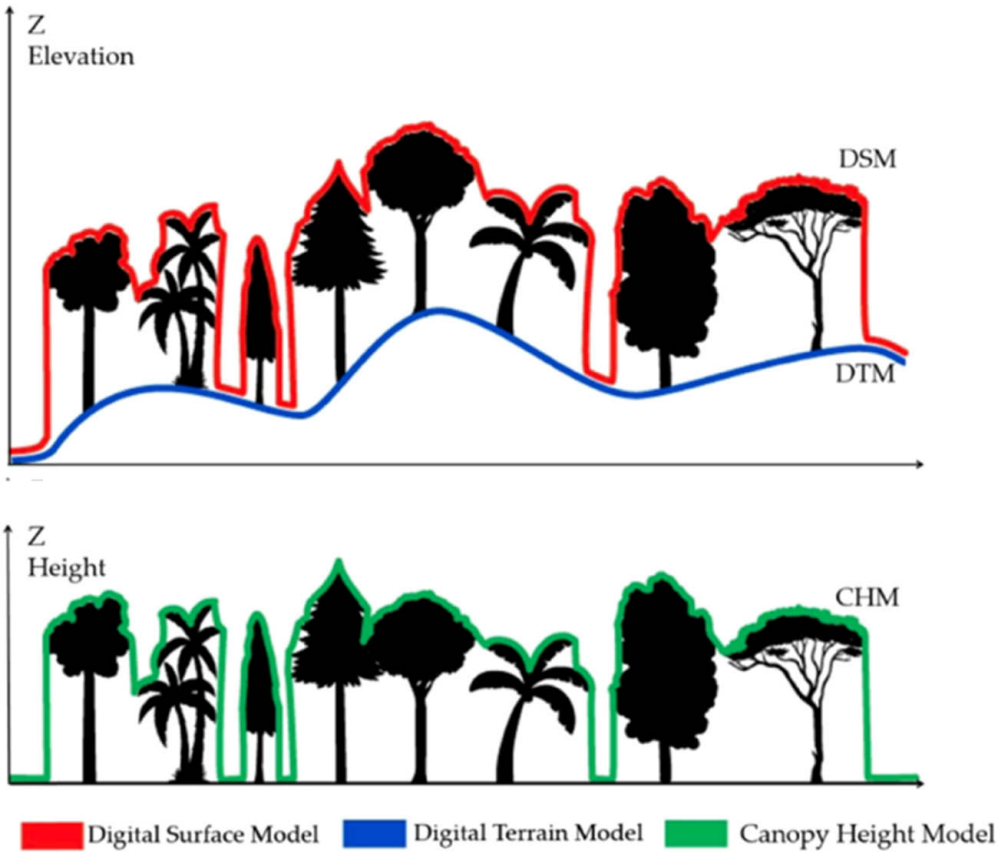


Figure 2.5: Comparison of digital surface models (DSM), digital terrain models (DTM) showing that: $DSM - DTM = CHM$. From [48].

Inkinen's paper [9], trees were found by looking for the local maxima in the CHM (also known as a digital tree height model) and this was used to find an approximate crown diameter for each tree, which was used in conjunction with ground measurements to calculate stem diameter and basal area of each tree. Height was found with a standard error of 13.6%. These types of methods were employed extensively in the early 2000s in boreal forests as improvements were made to GPS, marking a shift from manual and photogrammatic methods and a movement towards reconstructing individual trees based on the LiDAR scans [49], however the need for sample ground measurements

in conjunction with the LiDAR scans remained a major drawback [11, 50, 51]. With later improvements on canopy penetration using LiDAR and better availability of ALS data, a number of individual tree detection methods rose to the forefront of the field and showed their effectiveness, using geometric models to do more advanced segmentation approaches; including methods such as watershed segmentation [10, 12, 52–55]. These works firmly cemented the ability of LiDAR to be used in performing forest inventory with automated or semi-automated methods. The most popular method at the time for approximating the volume of each individual tree in the point cloud was to fit circles or cylinders to points using non-linear, or linearised, least square adjustment [56–64]. These mostly centred on TLS data, which has more data points for each tree compared to ALS data, and is more labour intensive to collect.

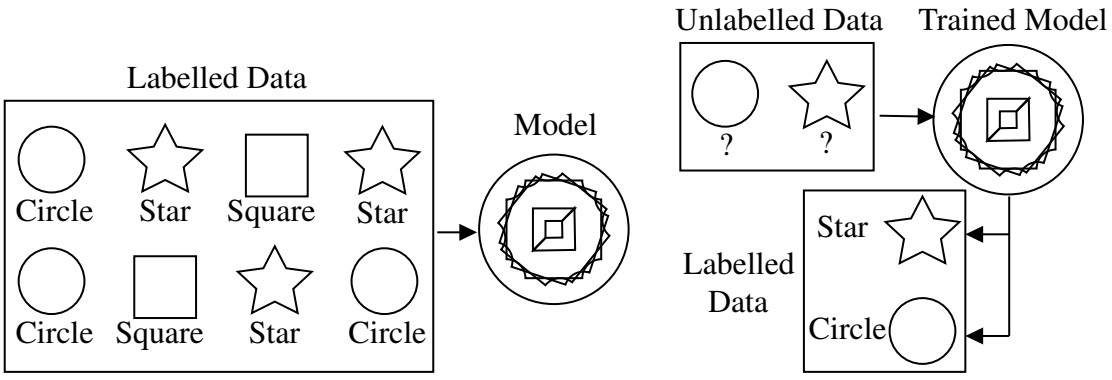
2.2 Deep Learning for Point Clouds

2.2.1 An Overview of Supervised Machine Learning

A good way to begin understanding what machine learning is, is to first understand what it isn't. Artificial intelligence is the pursuit of creating intelligent entities, where intelligence is defined as thinking or acting in a way which is either human or rational. Machine learning however is a subfield of artificial intelligence, focused on processing data in a way which allows a machine to adapt and learn from examples rather than instructions. The field is unconcerned with 'intelligence', however machine learning is sometimes used as part of artificial intelligence models [65].

The concept behind machine learning is to take some inspiration from the human behaviour of learning from example. This was proven to work early in the history of machine learning using simple shapes [66], and was explored further by others [67, 68]. Early key examples investigating pattern recognition [69] and game-playing [70] relied on the ability of their algorithms to store previous examples and perform some kind of comparison. It is difficult for a human to describe the way that another person looks enough that they can be easily recognised. Describing the curves of a face, the shape of a nose, or explaining all of their physical characteristics is difficult and cumbersome, it is an attempt to translate visual information into language for someone else to translate back into visual information again. However showing them a few photos of this person and suddenly the person can be easily recognised as their key traits are subconsciously picked out. Rather than trying to lay out to a machine all of the different elements of the data that it should make decisions on, we give the machine some examples and allow it to make its own inferences of what information is most relevant to make decisions on [71]. In our case, rather than attempting to define all the different aspects of what may be seen in a forest point cloud we provide labelled examples and train the network to learn its own way to classify future examples based on this.

Algorithms used for machine learning can be classified based on their desired outcome and the way in which the algorithm learns. The type on which this thesis is primarily concerned with is supervised learning [71–73]. This is where a set of labelled data is provided, in our case we will have point clouds where the points are labelled with one



(a) Model training (b) Model inference

Figure 2.6: Example of how models are trained and then used in supervised machine learning. To train the model (a) we provide labelled examples of data, and then to infer unlabelled examples (b) we provide the data in the same format we did while training.

of four categories: stem, foliage, undergrowth, or ground. Then the algorithm attempts to predict and classify unlabelled examples of data through approximating a function to map the input into the outputs based on the previous examples [29]. A simple visualisation of supervised machine learning can be seen in Fig. 2.6. Alternatively there is unsupervised machine learning which attempts to classify inputs into groups of outputs based on its own classification criteria without target labels [71–73]. In our context that could be helpful to find different meaningful labels, and is also helpful as it eliminates the time consuming process of labelling all the data. In between these lies the aptly named semi-supervised learning which utilises a combination of labelled and unlabelled data [71–73]. Supervised machine learning has had excellent results on an enormous range of real life problems such as recognising handwritten digits [74], filtering email spam [75], weather forecasting [76], and even detecting credit card fraud [77].

When applying traditional machine learning methods to images, the concepts begin to fail due to the density of images. A single 1024x768 colour image contains over 3 million pieces of information which relate to each other in complex ways, which poses a major computational challenge. Slight translations and rotations in the images result in a significant change in the actual pixel values while being semantically identical. Therefore a technique was needed to both reduce dimensionality and add redundancies to translation and rotation. This solution appeared in the form of Convolutional Neural Nets (CNNs), which apply a convolution kernel over the image to create a convolved feature. This solution was highly effective and has been applied with excellent results across a number of 2D domains, becoming a core aspect of the image classification field [78–80]. When moving onto point clouds, CNNs do not perform as well as they do on images, leading to the next stage of development in machine learning.

2.2.2 Existing Methods for Deep Learning on Point Clouds

In Section 2.1 we discussed the effectiveness of LiDAR as a 3D sensing tool for a variety of environments and landscapes, as they are able to capture a multitude of

information about geometry, scale, and shapes [81]. In Section 2.2.1 we focused on the benefits of machine learning when applied and understand its many benefits. To combine these two topics and perform deep learning on point clouds is more challenging than naively applying previous techniques. Point clouds have a high dimensionality and are unstructured by nature (having sections of highly varying density), which does not align with previous deep learning methods such as convolutions which require a highly regular data structure. Previous solutions to this problem involved transforming the point cloud into layers of images and performing analysis on those, or by transforming them into voxels (Essentially 3D pixels in a discretised 3D space rather than points in an irregular space as point clouds [82]). This data transformation can increase the size of the data significantly, making it more computationally expensive to work with, and also creates quantization artifacts that do not reflect the true nature of the data [22]. From these issues, a new process was needed to revolutionise point cloud processing the way that CNNs did for images [23, 28].

PointNet

The earliest approach to processing point clouds directly using deep learning approaches was from Qi et al. in their seminal work "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" in 2017 [22]. The purpose of PointNet was to be "a unified architecture that directly takes point clouds as input and outputs either class labels for the entire input or per point segment/part labels for each point of the input." [22], which avoids the problems discussed above from transforming the data to a more readily processed format. The design of the network was to allow for classification or segmentation to be performed on point clouds, where classification is assigning a semantic label to the entire point cloud and segmentation is assigning these labels to each point or region of the point cloud. This is shown in Figure 2.7.

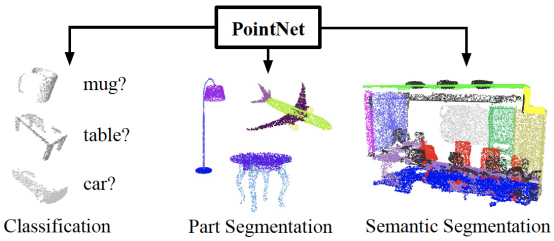


Figure 2.7: Two specific tasks for which the PointNet framework can be used for, classification and segmentation [22]

Qi et al. consider point clouds from a data structure perspective as an unordered set of vectors $\{P_i | i = 1, ..., n\}$ where P_i contains the data associated which each point; typically (x, y, z) but additionally can be extended to other data such as intensity or colour information [22]. What separates this avenue of research from previous work on deep learning on unordered sets [83] is the geometric relation between points despite their lack of order.

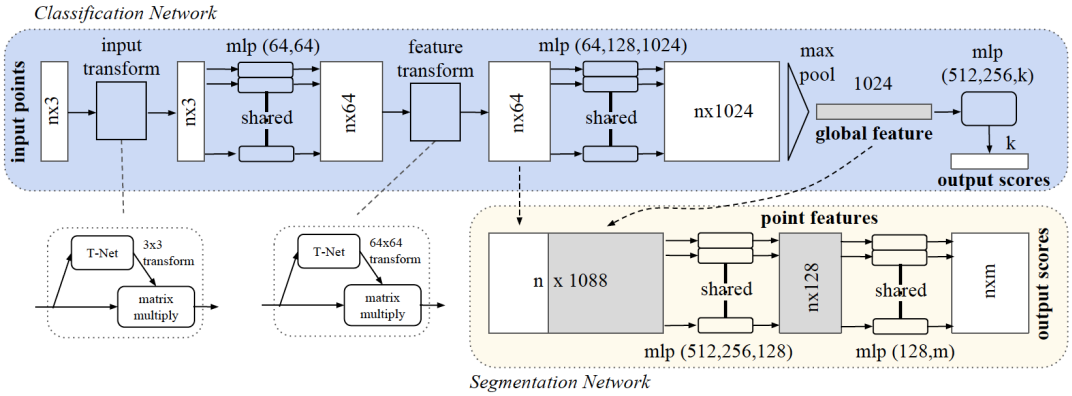


Figure 2.8: Architecture of PointNet. When performing classification, the network takes n points as inputs, applies transformations, and aggregates point features by max pooling, and outputs the scores for the k class labels. The process is extended if performing segmentation instead, using local and global features to output k scores per point instead. [22]

The three main design decisions that create the architecture shown in Fig. 2.8 were explained in depth by the authors. A **symmetric function** is used on the input point cloud to create a new vector that is now invariant to any transformations in the original [22]; essentially this ensures that if the same point cloud was fed into the network but the points are input in a different order then it would appear the same to the network after this symmetric function is applied. This addresses the issue of having an unordered input, and appears in Fig. 2.8 as the 'input transform'. **Local and global feature aggregation** is performed on the global point cloud feature vector, shown in Fig. 2.8 (*Segmentation Network*). The global feature is concatenated with the local per point features before being used in the next step, giving each point some level of relation between local geometry and global semantics [22]. This aims to address the challenge of capturing geometric interaction within an unordered set. A **joint alignment network** is used to make the network invariant to geometric translations the point cloud could undergo. If the same point cloud is input multiple times after being rotated or translated then the results should not vary drastically. This is achieved by predicting an affine transformation matrix which is applied to the input, and is applied again in a modified form to the features [22]. These are shown as the two 'T-Net's in Fig. 2.8.

The new ideas presented in PointNet were revolutionary to the field, finally presenting a way in which point clouds could be used directly by a neural network without the errors introduced through transforming it to a different data structure. The challenges which had prevented this solution from being implemented before were well overcome through suitable design decisions. The empirical results from the paper also indicate that this architecture gives results fast enough to be promising for real-time applications. Performance on existing benchmark datasets show that PointNet performs as well as or better than previous state-of-the-art methods [22]. While PointNet made major strides in the field and provided a solid foundation for future papers to continue work on, there were still some shortcomings. PointNet considers the interaction between global and local features, however it does not effectively capture local dependency between points as it does not appropriately account for the metric space in which these points exist [3, 28]. The relationship between points based on the distance between them is

not factored into the architecture, meaning that it is much more difficult for fine-grained patterns to be recognised [3].

PointNet++

Later in 2017 some of the same authors of PointNet [22] worked to amend some of its shortcomings in publishing "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space" [3]. To briefly explain what a metric space is: if there exists a set P that has some distance function Q, then the set P is a metric space. The elements of a metric space are called points, and given two points a, b the distance is the number $Q(a,b)$. The distance measure we typically care about is the m-dimensional Euclidean distance, shown in Eq. (2.2) [84].

$$Q(a,b) = \sqrt{\sum_{i=1}^{m} (a_i - b_i)^2} \tag{2.2}$$

Point clouds are by this definition metric spaces, and as we are concerned only with (x,y,z) data in this case then it is a 3-dimensional Euclidean space. Local structures are important to neural nets for assigning labels, and this can be seen in CNNs where the convolutions are inherently designed to summarise local structural information. This information is captured at multiple scales in a CNN to create a spatial hierarchy, linking features at all levels together [3, 85]. It is this effective concept from CNNs that PointNet++ aims to adapt, applying hierarchical feature learning to point clouds. This is achieved by using *set abstraction* levels. Each level is used to process a set of points and abstract these to create a smaller set. These levels are processed iteratively from the bottom up, gradually abstracting larger local regions as it moves up hierarchically. Within the levels, three tasks are performed in layers; a *sampling layer* which defines centroids of local regions, a *grouping layer* which constructs regions around these centroids, and a *PointNet layer* to find feature patterns, these can be seen in Fig. 2.9 [3].

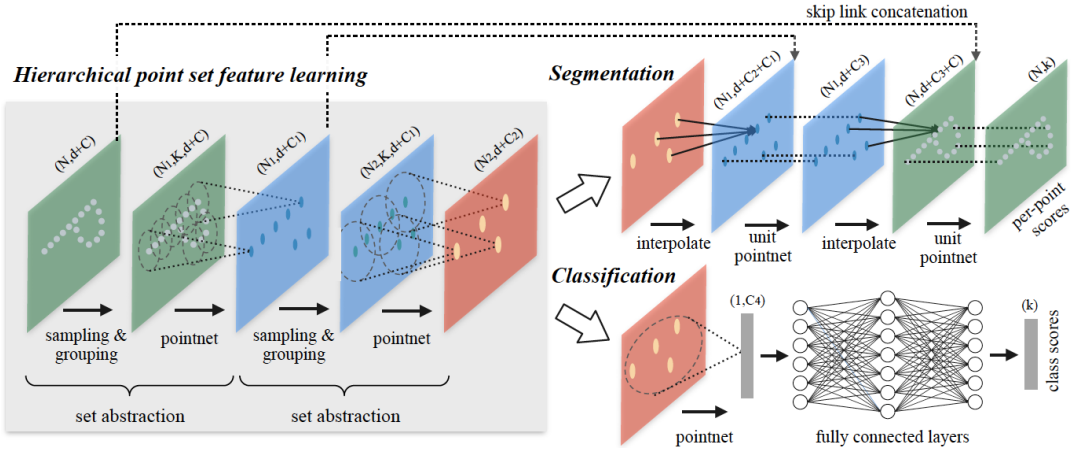


Figure 2.9: Architecture of PointNet++, shown as applied in a 2D Euclidian space [3]

The other related challenge with working in a metric space with point clouds is that the density of the points can vary dramatically over a single or multiple point clouds. If searching for features point by point it is challenging to find them in a low

density area due to the lack of nearby points, so it is important to adapt and search on a larger scale. Additionally, if a feature is effectively learned in a dense point cloud the network may struggle to recognise it again in a more sparse point cloud. The way this is addressed is by looking at the grouping layer in each abstraction level and using *density adaptive layers*. The two options for these are **multi-scale grouping** (MSG) or **multi-resolution grouping** (MRG) [3]. MSG concatenates together features at different scales to create a large feature vector, however it is computationally expensive, as it applies PointNet over all the centroid points found in the sampling layer. Alternatively, MRG finds the features of a region in a given abstraction level in two parts, one part being a summary of the features in the region of the previous (lower) abstraction level, and the second part being found by using all raw points in the local region using an instance of PointNet [3]. These are visualised in Fig. 2.10.

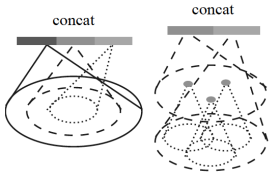


Figure 2.10: Density adaptive layers used by PointNet++. Multi-scale grouping (left) and multi-resolution grouping (right) [3]

The new improvements on the original PointNet [22] architecture in PointNet++ show large improvements when tested on the same benchmark datasets, indicating that the ability to account for the metric space which the points are in does successfully extract valuable information for segmentation and classification. It also shows good robustness against variable densities of point clouds, which is important in our context of forests, where point clouds taken of different forests using different types of laser scanning techniques can dramatically alter the density in different regions. This builds on the previous work of PointNet [22] well by continuing to utilise the point cloud data format to its fullest extent without transforming it to another data format and introducing quantisation artifacts. While the network does relate low level features to high level features hierarchically, a shortcoming of this network is that it does not account for the overall spatial distribution of the point cloud [29]. This architecture shows good improvements and would be appropriate for use in analysing forest point clouds for the purpose of this thesis. To continue understanding the state of the field, we will briefly discuss some other prominent networks.

PointCNN

An example of taking inspiration from 2D convolutional methods for use in 3D point cloud datasets is seen in "PointCNN: Convolution On $\mathcal{X}$-Transformed Points" [27]. This paper proposes that the symmetric function applied to the input point cloud that is used in PointNet [22] and other networks cause them to lose some information. The solution to this is by randomly sampling a set of representative points, using kNN to select points in their neighbourhood, applying an $\mathcal{X}$-transformation to the points, then using a type of convolution operator. The aim of this process is change the point cloud from an irregular domain into a regular domain such that convolutions can be applied effectively [27,28]. The $\mathcal{X}$-transformation at the heart of PointCNN is beyond the scope of this thesis, but essentially aims to transform the points into a canonical format while preserving the spatial distribution between points [23,28]. In further studies, PointCNN

was found to have a very strong learning ability, effectively recognising input patterns it had seen before, however the overall generalization did not compare well to similar networks [29]. This poor generalisation ability makes it a less favourable choice for the purposes of this thesis due to the significant differences between commercial forests and native forests.

PointSIFT

A popular algorithm in the field of computer vision is SIFT (scale-invariant feature transform), which is essentially a method for finding the distinctive consistent features that are the most reliable for matching between different images. This method became popular as the features are invariant to scale and rotation (which change often between images) and are robust to noise, lighting changes, distortion, and change in viewpoint [86]. Its these beneficial qualities that the authors of "PointSIFT: A SIFT-like Network Module for 3D Point Cloud Semantic Segmentation" [25] attempt to capture in their network PointSIFT. This network is built on a orientation-encoding unit, which convolves the features of nearby points into eight orientations. These units are stacked together in order to achieve multi-scale representation [23]. This excellently transfers the benefits of the original algorithm into a way that can be used with point clouds, and results successfully shows the network is aware of scale. Additionally, this module is able to be applied recursively in order to achieve hierarchical feature representation, similarly to PointNet++. Another good improvement comes from the downsampling methods used by the network; in PointNet++ the grouping layer does not capture all the points in the point cloud which results in some loss of information, whereas in PointSIFT the orientation-encoding unit uses all points, ensuring the entire point cloud is used as part of the prediction [25].

RandLA-Net

In the other methods investigated there is typically some aspect of sampling or pre-processing used in order to render a point cloud usable by the network or to improve later computational efficiency. RandLA-Net [87] focuses on the efficient semantic segmentation of large-scale 3D point clouds, trying to identify per-point labels quickly and accurately, a critical endeavour for real-time systems such as autonomous vehicles. The way in which RandLA-Net achieves this is by using random sampling instead of other more selective sampling methods such as the popular FPS (furthest point sampling). Random sampling has the drawback of discarding information that may be key for understanding the scene. This is addressed using a new local feature aggregation module, the specific implementation of which can be found in the paper. This is applied in layers, ensuring that the size of the point cloud is reduced by several orders of magnitude which keeping the features that render it identifiable. This is shown in Fig. 2.11. Results from this architecture are impressive, outperforming many other approaches when compared on the S3DIS dataset [88]. While working well on urban and indoor environments, it would be interesting to see how RandLA-Net performs with forest point clouds.

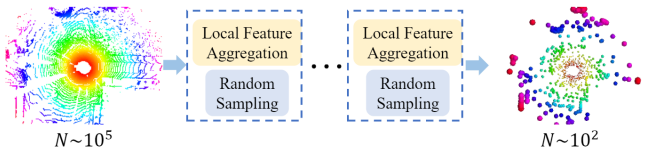


Figure 2.11: Application of RandLA-Net demonstrating how it downsamples point clouds to a significant degree while retaining key features due to its local feature aggregation module [87].

2.3 Applications of Deep Learning to Forest Point Clouds

At this point in the literature review, we have covered the ways that LiDAR is useful for scanning forests and acquiring point clouds, and the ways that deep learning is used to analyse point clouds, and now can discuss how combining this knowledge can be used to directly analyse how deep learning can be used on LiDAR point clouds in order to better understand forests and woods.

A key idea present in previous works that utilised ALS data was its use in conjunction with additional sample measurements from the the field in order to build their models [9, 11, 50, 51], however with denser ALS measurements being available over time, the ability for direct measurement of trees and their parameters without sample measurements became a more achievable goal [1]. In 2014, Qin et al. published "Individual tree segmentation over large areas using airborne LiDAR point cloud and very high resolution optical imagery" which aimed to detect individual trees within a stand using the AMS3D (Adaptive Mean Shift 3D) clustering algorithm, an example of unsupervised learning [19]. The simplified procedure for this paper was to start by finding the CHM, then using the very high resolution optical imagery to separate the forest into stands. The assumption made in separating the forest into stands centres on the idea that the patches of forest that appear homogeneous from aerial imagery in fact have the same overall characteristics. Next, the CHM is used for each stand to identify individual trees based on watershed segmentation using the LiDAR data, essentially using the slopes of each tree crown to tell them apart. Finally, AMS3D is used to cluster the points that have been normalised using the CHM so that all points not associated with the group have been assigned to a tree [19]. The results of this paper showed that they appeared to be overall effective in identifying individual trees, however there were a number of shortcomings that make this method unsuitable for larger scale application. There was no comparison to a ground truth dataset in order to determine if the results were accurate, nor any suitable quantitative analysis on the results. The generation of stand level results comes from the assumption that the trees within each stand (identified from aerial optical imagery) have similar growth conditions, however this assumption does not hold up in all forests, particularly in Australian native forests which we are focused on. Unsupervised learning techniques have been effective in identifying tree species [89, 90], and the overall characteristics of forests [91, 92].

Moving to supervised machine learning methods, there have been a number of examples where deep learning has been used on the 3D LiDAR point clouds before the use of direct point cloud methods that were discussed in Section 2.2.2. Many methods have

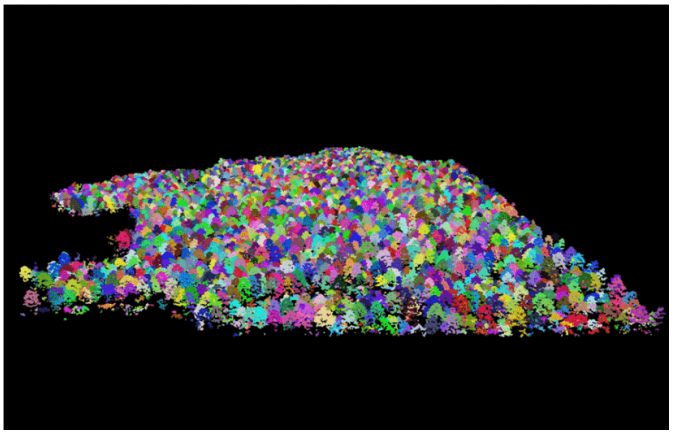


Figure 2.12: 3D visualisation of individually segmented trees within a forest using the methodology described by Qin et al. using AMS3D [19]

involved in some way voxelising or rasterising the point cloud [18, 20, 21]. Voxelising a point cloud involves taking the point cloud as unordered points that exist in a metric space and transforming it into a discretised 3D grid. In this more regular format it can be used with 3D-CNNs for deep learning, and this is applied in a forestry context by Ayrey and Hayes using ALS data to discern key metrics of the forest [21]. In their paper, they needed to use a collection of older field data to train the model, some of which was from five years before the LiDAR data was collected. This is challenging given the variability of tree growth over time, but was accounted for well by using simulations to project the data forward in time. The CNNs were used for an area-based forest inventory, meaning that individual trees were not measured and results on biomass, tree count, and percent needleleaf were all approximations of large areas. These make the interpretability of the results low, as it is difficult to identify where errors lie over an entire area compared to an individual tree. This compounds with the issues of voxelisation increasing the size of the data structure and removing fine grained features. Similarly, Zou et al. [20] project their point clouds onto 2D images from a variety of different positions, similarly to taking photos from different angles. The density of the points in each pixel are used to determine its grayscale intensity. This is performed with individual trees that are first extracted based on the high density of points on each stem. The multiple images of each extracted tree are used to train a deep belief network that can be used to interpret future unlabelled trees. Benefits to this method include the high amount of data augmentation that is achieved from the multiple viewpoints of the trees and the good classification rate that is achieved. However, this method would perform poorly with ALS data instead of the TLS data used, as the density of the point cloud would not be high enough to extract individual trees consistently based on point density around the trunks. Also, this method does not take full advantage of the data structure, losing details through rasterisation. An example of this 2D rasterisation is also seen in the work of Hamraz et al. [93] where multiple views of the DSM are used with a CNN in order to assess if individual trees are deciduous or coniferous. Additionally, recent work by Persson et al. [18] uses four projections of detected trees at different angles to generate images to be used by a CNN to detect tree species as part of a larger forest inventory process using data from ALS, TLS, and aerial photos.

As was discussed in Section 2.2.2 when the data is transformed it creates quantization

artifacts and typically increases the size of the data structure. By using point-based deep learning methods such as PointNet [22] this can be overcome, which is what is utilised by Windrim and Bryson in their paper "Detection, Segmentation, and Model Fitting of Individual Tree Stems from Airborne Laser Scanning of Forests Using Deep Learning" [1]. In this work the authors construct a complete pipeline from a forest point cloud all the way to individual tree measurements of DBH, stem volume, taper, and sweep. As a simplified explanation of their process, they first generate a DTM to remove ground points from the point cloud, then individual trees are detected with a view from the *xy*-plane that is used in a Faster R-CNN object detector to generate 2D bounding boxes for each tree that are later projected into 3D cuboids, associating all the points that fall within them to a given tree. Next, a comparison is done between between a voxel-based 3D-FCN (fully convolutional network) deep learning approach and a PointNet method to segment the stem out from the remainder of the tree, and finally the stem is reconstructed using RANSAC (RANdom SAmple Consensus) and NLLS (non-linear least squares). Their pipeline from point cloud to labelled points for both methods is shown in Fig. 2.13. Having a reconstructed stem allows for the discernment for individual tree measurements mentioned previously by fitting them to geometric models. In this paper, the authors found the performance of the voxel-based method was better than that of PointNet, however it used significantly more memory, and this was likely due to the shortcomings of PointNet in not being able to account for local structures induced by the metric space. The results also suggest that the voxel-based approach had mislabelling between the stems and foliage where they are close together. This likely came from the coarseness of the voxels compared to individual points, however this had a low impact on the final stem reconstruction. Additionally, the RANSAC reconstruction was too rigid and did not perform well for trees with high amounts of sweep [1].

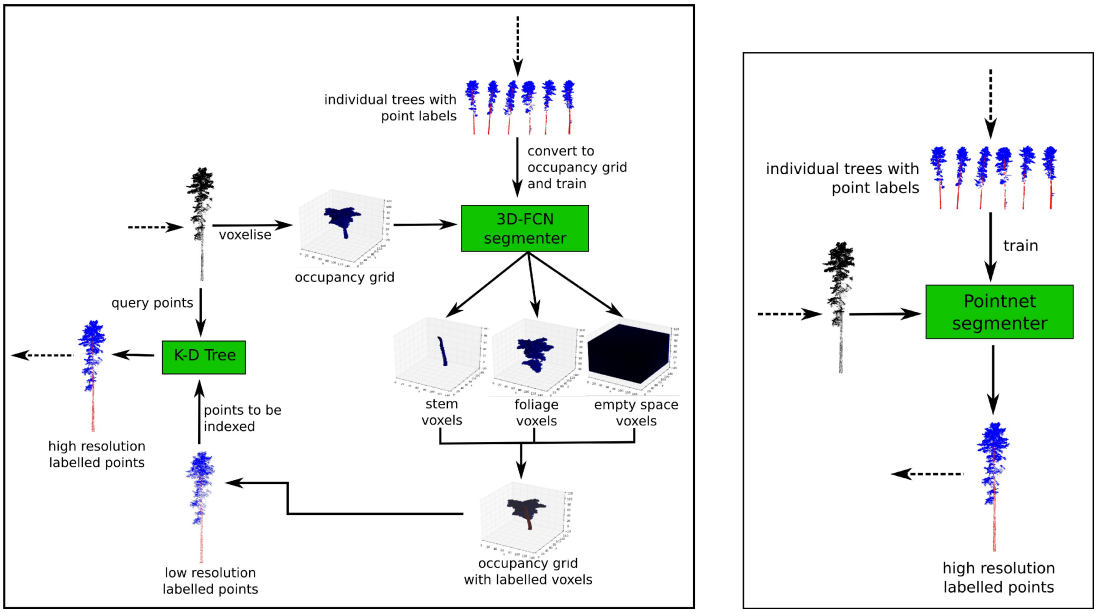


Figure 2.13: A graphical comparison between a voxel based 3D-FCN approach to segmentation (left) and raw-point-based PointNet based approach (right) [1]

In a work by Shen et al. [15], the authors propose that some of the challenges in

semantic segmentation of forest point clouds come from the mismatch of point cloud geometric features within a scene. Essentially stating that while the tree crowns are characterised more by scattering, the ground is characterised more by planarity, and that the networks struggle to learn about all of the geometric features within the scene and achieve the same results that convolutional methods do in indoor scenes. The suggested solution to this is energy partitioning, an unsupervised approach to cluster parts of the point cloud into geometric partitions of similar geometry, followed by geometric feature balancing that model that focuses on increasing geometric features that do not appear as often as others in order to improve the network's ability to generalize and recognise features that appear less frequently than others (e.g. more examples of branches assist the network in distinguishing them from stems) [15]. A visualisation of the geometric feature balancing model is seen in Fig. 2.14. Following this, the balanced data is used

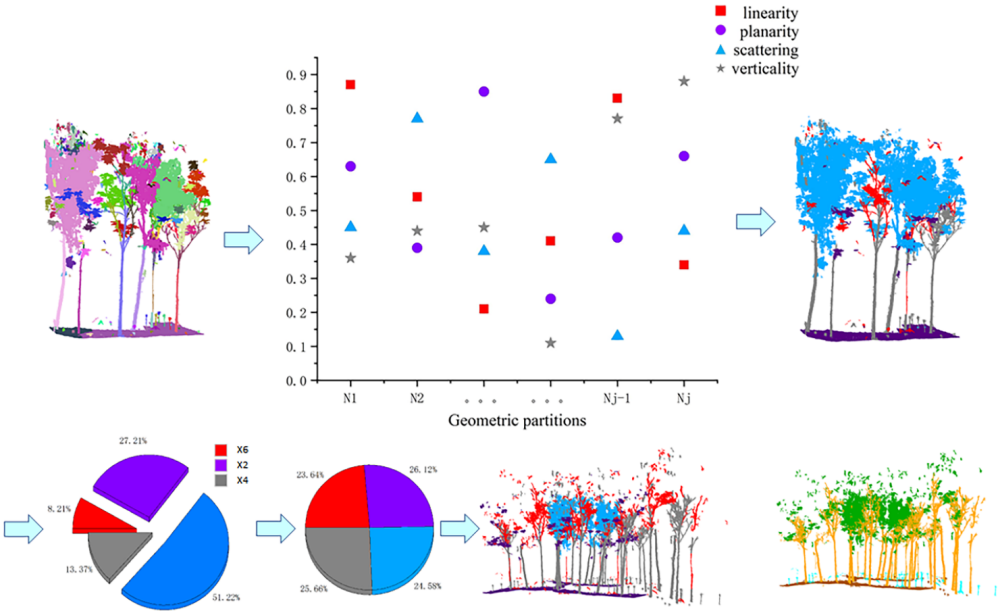


Figure 2.14: Geometric feature balancing as performed by Shen et al. in order to assist the network in semantic segmentation by providing more training examples of features which appear less frequently than others [15]

by a PointCNN based network to semantically segment the point cloud into terrain, foliage, stem, and other points. This approach was overall found by the authors to be effective for their purposes, however the precision and recall for the stem label was the lowest out of all four [15]. This is unfortunate given that the stem is arguably the most important label for forests as the most useful measurements of the trees are made from the stem. This may be due to the geometric balancing reducing the comparative number of examples for the stem to increase focus on other labels such as foliage. Additionally, the overall accuracy dropped by 4% between different forest datasets due to the different species of trees present, showing a limitation in the generalisation ability between different datasets. The methods described are applied only on TLS data, which has a higher point density for stems compared to other scan types, so further testing would be required on ALS data.

In the work of Krisanski et al. "Sensor Agnostic Semantic Segmentation of Structurally Diverse and Complex Forest Point Clouds Using Deep Learning", the authors aim

to utilise PointNet++ in order to perform four label semantic segmentation (terrain, vegetation, stems, and coarse woody debris) in a methodology that is transferable to TLS, MLS, or ALS datasets [14]. A large amount of data was manually labelled in order to make this possible, segmenting 177 trees across all three scan types, in seven different plots across four different forests of varying tree density. This approach aimed to use PointNet++, however that network is not designed to take millions or billions of points all at once. In order to remedy this the authors split the point cloud into cube-shaped regions of the same size, and set a limit on the maximum and minimum number of points each region must contain. The data was kept at full resolution (except for when the maximum number of points were reached, in which case points were randomly removed) and the size of the boxes was chosen arbitrarily as 6m by 6m in order to provide enough context to identify most elements within it. Additionally the cubes were designed to overlap with each other in order to prevent elements on the borders of the cubes from being consistently misidentified. This was performed prior to data augmentation and model training. In their results, the authors found that they had better overall precision than Windrim and Bryson [1] did using their 3D-FCN or PointNet approaches, indicating that the strength of PointNet++ to infer details about point clouds relating to the metric space improves results for forests compared to the vanilla PointNet approach. A limitation with this paper, along with that of most approaches attempting semantic segmentation, is the subjectivity of humans labelling the reference datasets, and the many mistakes that can be made during this process. This challenge was approached in a interesting way by Bryson et al. in "Using Synthetic Tree Data in Deep Learning-Based Tree Segmentation Using LiDAR Point Clouds" [41]. As we have seen in previous papers on supervised learning, a large amount of accurately labelled data is required in order to train the network to recognise unlabelled regions correctly. This is a challenge due to the large amount of time it takes operators to label point clouds and the natural rate of human error that this entails. By using artificially generated data models can have a large amount of training examples while avoiding the human labour of collecting and labelling this data. The exact methods of this synthetic data generation are beyond the scope of this literature review. The study found that using synthetic data with PointNet++ for semantic segmentation was highly effective, and the authors suggest future research looks towards improving the generation of this artificial dataset in order to better represent highly multi-apical trees and fine details, and also to investigate domain adaptation in combining both real and artificial datasets.

As for other examples of PointNet++ being used for forestry point clouds, Liu et al. worked with a backpack laser scanning system in order to identify tree species using PointNet++ [16]. The procedure followed for this paper involves normalising the height of the trees across the point cloud by using a DTM to remove ground points and then segmenting individual trees using an algorithm called 'comparative shortest path' that is inspired by ecological basis and classical metabolic ecological theory [94]. Following segmentation of individual trees, they are each normalised to fit within a unit sphere, which assists in tree size not becoming a dominant factor for classification. The experiments that this paper focuses on is in the downsampling methods that are used on each tree before it is used to train the model, as well as testing if trees are better classified with the leaves removed from the point cloud first or remaining part of it. The authors concluded that PointNet++ was a good network for this task, however the high classification accuracy they achieved was partially attributed to the high resolution backpack laser

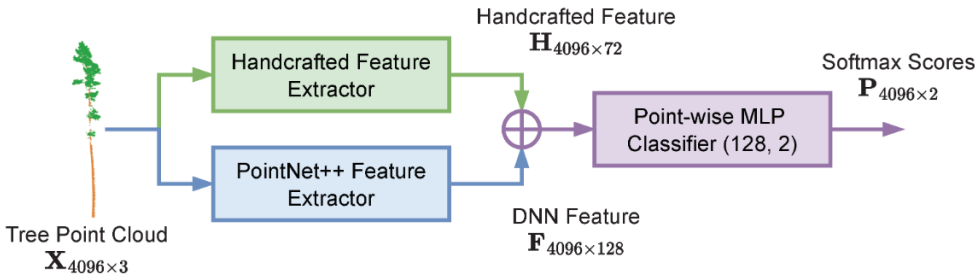


Figure 2.15: Overview of the backbone model used by Wang and Bryson with an example point cloud with 4096 points. Both feature vectors are extracted and concatenated before being used with a point-wise MLP to predict labels for each point [13].

scanning used when compared to other studies using more sparse ALS data [16]. Additionally, using the unit sphere normalisation was a good choice as when tree height features were accounted for the classification accuracy dropped. Leaf-wood separation was found to be unnecessary for species classification unless data is collected from the same trees in different seasons. FPS was identified as the best downsampling method for this task as it led to the highest classification accuracy, with nonuniform grid sampling found to also be suitable. These preprocessing techniques investigated are useful and should be applied similarly to semantic segmentation tasks, as it can reduce memory usage, computational complexity, time, and improve overall results. The trend towards raw-point-based algorithms in recent literature is noted, as well as its effectiveness across a variety of different tasks in forest point clouds. This is also shown in the work of Wang and Bryson [13] in investigating the use of handcrafted features for semantic segmentation of trees. In this work the authors focused on the used of handcrafted local features to use in combination with a PointNet++ feature extractor in order to increase performance. An overview of this process is shown in Fig. 2.15. Overall this methodology outperformed similar popular supervised methods, and the final parameters estimated for each tree was comparable to established methods. The strong trend in recent literature towards these point-based methods and their reported success indicates it is an area which warrants further investigation.

2.4 Summary

In conducting this literature review, we have identified the strengths of LiDAR for scanning forests and the ways in which deep learning techniques can be used for analysing the resulting point clouds. The lack of investigation into pre-processing techniques for point clouds stands out as a gap in the literature, shown particularly by the arbitrary region size used by Krisanski et al. [14], and the unknown size of the sliding *xy*-window used by Windrim and Bryson [1]. Similarly, the normalisation methods discussed by Liu et al. [16] and left unsaid by other authors indicate that the area is not well discussed. The lack of discussion on native forests similarly stands out, with the vast majority of literature focusing on homogeneous managed forests which are highly structured, leaving unstructured heterogeneous native forests largely unresearched. In Chapter 3 we hope to investigate these areas and begin to fill the gap in literature.

CHAPTER 3

Methodology

"Great things are done by a series of small things brought together." - Vincent van Gogh

This section develops a pipeline for progressing from a large point cloud of millions of points into a number of smaller point clouds that can be processed by PointNet++ to train it for semantic segmentation. This consists of data selection and separation, localisation, normalisation, and augmentation. Once we have a number of networks trained based on this process we can evaluate how they all perform on four label semantic segmentation, aligning with the aim of the thesis in investigating how these methods impact the overall results. The complete pipeline for training PointNet++ can be seen in Figure 3.1 and is explained over the course of this chapter.

3.1 Datasets

The experiments for this project were performed using two datasets. Both are point clouds that were taken from forests using ground-based mobile scanning. They were captured using mobile back-pack mounted Emesent Hovermap scanners which capture data at approximately 8000 points per m^2. The dataset used for training and validation was called 'DogPark', which was captured in a recreational forest in Rotorua, New Zealand, and consists of various species of trees. A quadrant of the dataset was split off and chosen as the validation set, with the remaining three-quarters used for training. The dataset used for testing was called 'HQP' and consisted of Caribbean Pine trees from a commercial plantation in Queensland, Australia. When following the methodology outlined ahead and in Figure 3.1, the entire process was followed for the training data, while the validation and testing data only had a single subsample taken from each window, and had no augmentation applied. The two datasets are shown alongside each other for reference in Figure 3.2.

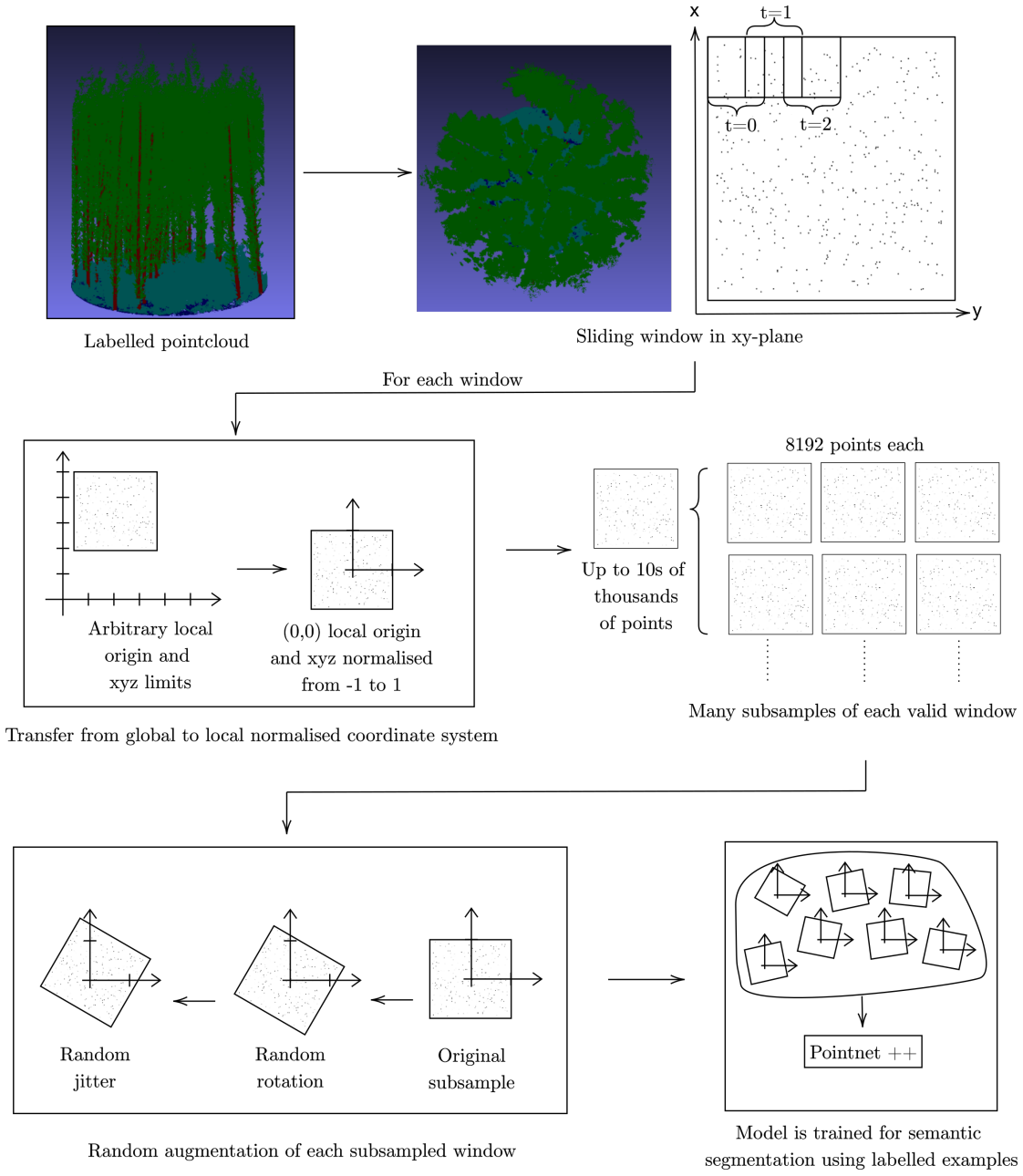


Figure 3.1: Graphical overview of the pipeline for training PointNet++ to perform semantic segmentation on a region of the point cloud. Beginning with the labelled point cloud a sliding window passes over all points and localises and subsamples them. Each subsampled window is then augmented and used to train the network to label future unlabelled windows. This entire process is repeated with different window widths of the sliding window in order to evaluate which will have the best overall results.

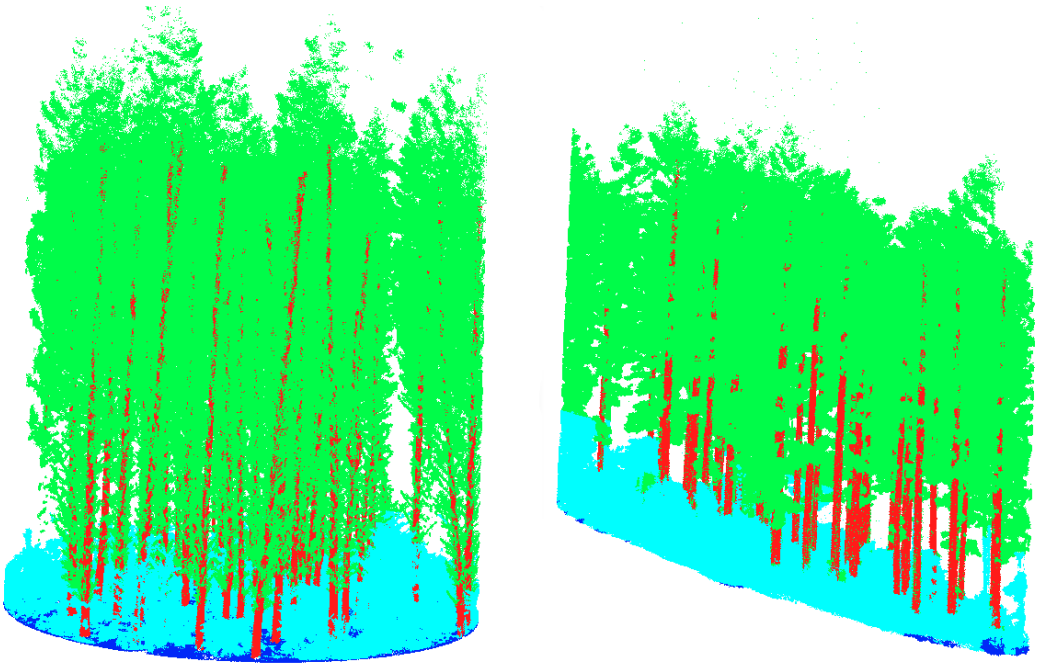


Figure 3.2: The dataset 'DogPark' used for training the models (left) and the dataset 'HQP' used for testing the models (right), it is clear that the testing set has a significant slope compared to the training set, and this will impact our results of ground and undergrowth detection.

3.2 Sliding Window

As was discussed in the work of Windrim and Bryson [1], a window can be used to slide in the xy-plane of the point cloud and a selection of points can be chosen within this plane for use. The purpose of this is that most point-based semantic segmentation networks are limited in the number of points they can process at a time, and by sectioning off our large pointcloud into a collection of smaller examples the entire set can be more effectively used. By using only a smaller section of the entire point cloud, it also prevents the network from learning forest-wide features that may not apply to other datasets, and brings more focus onto individual trees. There is a balance in choosing the size of these sections or windows, as having a window too small will limit the amount of contextual information available to the network which can often be helpful in classification. **The main test being run in this thesis will involve changing these window sizes to train different networks in order to evaluate which window size is best.**

In using this window it is important to have one variable being changed, and in this instance we will be varying the side length of the square window and keeping the overlap between these squares at a constant 30%, and the number of points taken for an example as 8192. 8192 points was chosen as a good middle ground for the number of points as the original implementation of PointNet [22] uses 2500 points, PointNet++ [3] limits itself to between 2048 and 10000 points, and the implementation used by David Griffiths [2] uses 8192 points in testing with the ScanNet and ModelNet indoor datasets. A 2D visualisation of the sliding window can be seen in Fig. 3.3. When sampling from

the window the points will all be in their global coordinates, so we will instead convert them to a local coordinate frame by subtracting the window's centre *xy* coordinate from all points. This will better allow the network to learn general features within each window and not related to the overall forest structure. A 2D depiction of the sliding window can be seen in Figures 3.1 and 3.3. As was briefly discussed in Section 1.3, the width of this sliding window is the key variable that is being changed over the course of experiments with the hope of identifying whether a larger window with a lower point density (but a broader overall snapshot of the region) or a smaller window with a higher point density will lead to better results in semantic segmentation.

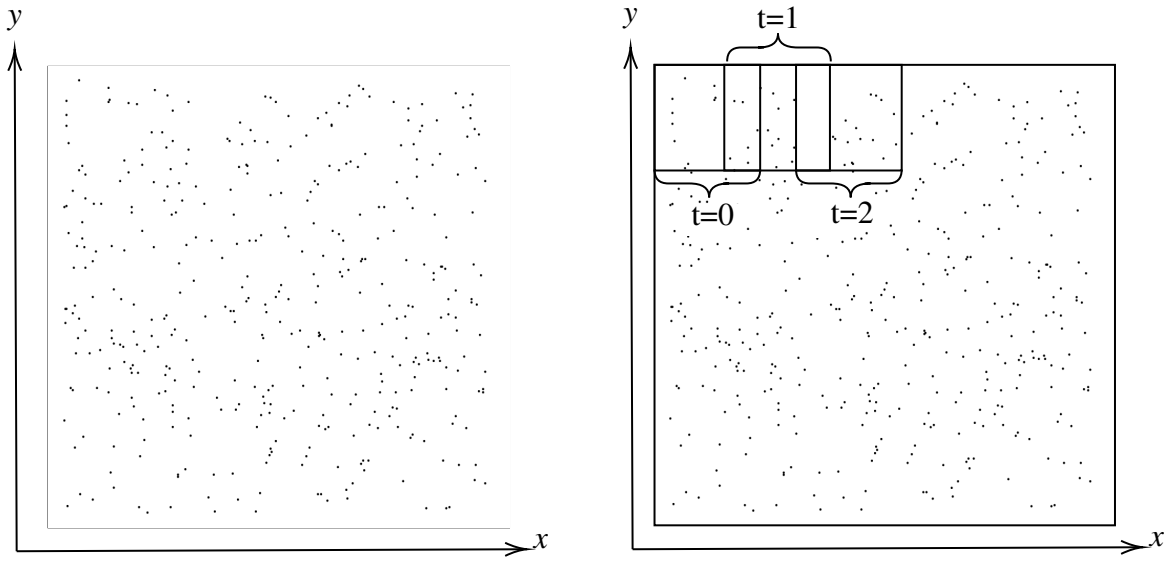


Figure 3.3: 2D depiction of the sliding window used. The left window shows the entire plot that may consist of many millions of points, while the right shows how the window moves over three timesteps, each overlapping 30% with the previous. This pattern is followed on both axes until the end of the plot is reached.

3.3 Normalisation

The functioning of PointNet++ [3] was discussed earlier in Section 2.2.2, and in we must be careful to shape our data in such a way that the network can more effectively be trained on it. One of the aspects to keep in mind is the implementation of the *set abstraction layers*. They are formed by iteratively searching for nearby points for each point using farthest point sampling (FPS). The radii used for this search have been implemented between 0 and 1, so it is important that our data is normalised between -1 and +1 in *x*,*y*, and *z* so that the local relationships can be accurately grouped.

3.4 Augmentation

As the window slides over the xy-plane of the plot we will subsample a set number of points from the window if it contains at least the set number of points (8192) or more. In the higher density centre region of the point cloud there are a larger number

of points. In order to maximise the training data we can perform repeated subsamples within the window, resulting in different collections of points. This helps the network better generalise the overall relationship between points in a region as it will have slightly different local and global relationships between points in an identical region. Each example can be rotated about its centre by a random angle between 0 and 360°, which is aimed at preventing the network from learning features more associated with how the scan was taken and the directional alignment of the forest. This is particularly important for the managed plantations where the trees are often planted in a grid and the network may otherwise inadvertently begin to associate trees together based on this direction which will not be present in other datasets. The final augmentation is in adding a small amount of random jitter to each set of subsampled points. The jitter is generated across all points as a number between +-0.01 with a normal distribution. This addition helps simulate additional noise in the data and gives the network more varied examples to learn with. As each window is subsampled, localised, normalised, and augmented they are saved into a TFRecord file to be used for training.

3.5 Model Compilation and Training

3.5.1 Software and Hardware Configuration

Once we have a set of these points they can be used with the network in order to train it to identify future unlabelled sections of forests. The implementation of PointNet++ used for these experiments was created by David Griffiths [2] with reference to the original paper and code [3]. The code required a number of changes to work with current versions of Nvidia's CUDA platform and current Tensorflow implementations. Initial experiments were performed with Torch Points 3D, an alternative framework for developing deep learning models on 3D data such as point clouds, however the implementation was found to be too closed to sufficiently run the tests as described previously when compared to Tensorflow. The versions of all software used needed to be carefully coordinated to work with the code written by David Griffiths, as well as being up to date with all required metrics, and being compatible with the hardware used. The graphics card used for all training was a Nvidia GeForce RTX 2070, CUDA 11.2, cuDNN 8.1, Tensorflow 2.11, on Ubuntu 18.04.6. It was important to ensure that all versions were compatible with each other and supplied sufficient functionality, as metrics such as per label IoU were not introduced until Tensorflow 2.8, which is not compatible with CUDA 10.1 which is the required version for running David Griffiths implementation of PointNet++ due to elements of the C++ code required for custom Tensorflow operations. These were all able to be upgraded by updating the C++ code to be compatible with newer versions. The model was kept constant across all experiments, with a batch size of 16, a learning rate of 0.001, batch normalisation set to true, and number of points in an example set at 8192. Before the data is used for training all examples are all shuffled in order to prevent the collection order from affecting results.

3.5.2 Metrics

A number of different metrics were used in order to evaluate the overall performance of the model and ensure training leads to results that are favourable for future predictions of unlabelled data. The way the model operates behind the scenes is to try and minimise the loss function that is chosen for the problem, and as this is a multi-label classification problem then using a cross-entropy loss function is most appropriate. The logic of how cross-entropy loss works is outside the scope of this report, however it essentially relates to the amount of surprise/uncertainty between outcomes. We want to minimise it. The other key metrics used for evaluating the effectiveness of the model were accuracy and Intersection-over-Union (IoU). The categorical accuracy is a simple generic measurement and compare the correct predictions to overall predictions as shown in Equation 3.5.2. This effective for understanding the overall performance, but does not take specific label information into account as IoU does.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

IoU is a particularly relevant metric and is used frequently in image segmentation, it is defined as the intersection of the prediction and the truth data divided by the union of the two. This is shown in Figure 3.4 with a bounding box example that can be more easily visualised, or alternatively as an equation using true positives, false positives, and false negatives in Equation 3.1. IoU is so helpful for describing the performance of the model as there are so many points that are being labelled that it is unrealistic to expect them all to be correct, however IoU manages to effectively describe the proportionality of correct predictions to overall predictions and is useful to look at both overall for all predictions and per label so we can identify which labels are being identified more or less effectively. A higher IoU is desirable. All of the metrics used were saved after an experiment was completed so that they could be compared and graphed alongside each other.

$$IoU = \frac{TP}{TP + FP + FN} \tag{3.1}$$

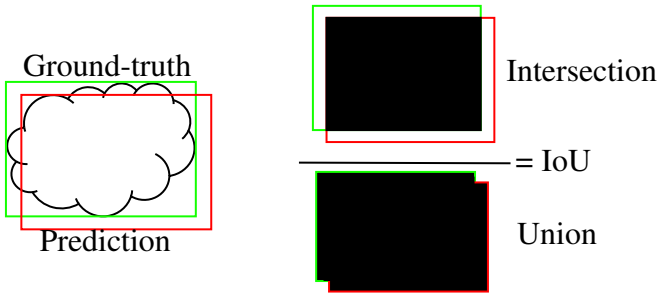


Figure 3.4: Intersection-over-Union is the area of overlap between two predictions divided by the area of union between predictions. In our point cloud examples it will be the discrete points predicted rather than an area.

3.6 Limitations

There are a number of limitations to this methodology and these would ideally be addressed in future works or any longer term continuations of this project.

- The model was trained and validated on a single forest in which the trees had a largely homogeneous structure. This was good for testing how the window size changes affects results, however these results may not generalise to other forests with more varied appearances such as native forests.
- The shape of the window used was a square, however PointNet++ uses a circular iterative search with increasing radii to try and associate different points with each other structurally. This means that in this implementation the points in the corners of the bounding boxes are underutilised by the network. Repeating with different window shapes would yield interesting results.
- The overlap between window was set at 30%. This was decided to be sufficient to ensure a partially obscured or cut-off tree in one window would later appear as complete in another window. This does not apply effectively for the smaller window sizes as the overlap also shrinks proportionally. Experiments with lower and higher overlap percentages would be helpful.
- Due to the overlap discussed previously as well as repeated subsampling there are many predictions made for most of the points in the point cloud, however all of these predictions are used separately. Implementing a voting system in which all the predictions for a point are used to determine a final label would increase accuracy and be a good avenue for future research.
- As each window needs to have the minimum amount of points to be considered valid, sectioning off a quadrant of the circular point cloud for validation data resulted in a low amount of data, as the majority of the valid windows were in the central high density region of the point cloud. A higher amount of validation data would allow better hyperparameter tuning.

3.7 Preliminary Tests

The validation of this experimental design was done with a number of sanity tests of each section to ensure that it was operating as intended. As subsamples of each window were taken they were visualised in MeshLab to ensure that the sections taken contained the correct number of points over the correct area. After the models were trained visualisations of their predictions were generated and compared alongside the true labels to check where mistakes were made, ensuring that the models were predicting intelligently and not randomly. Predictions were made with the model both before training and after loading the weights to validate that the predictions were a result of training and not a characteristic of the model itself. Augmentations were visualised to check they were occurring as intended. Some geometric calculations were made in order to validate the correct number of windows were being scanned by the sliding window over a given area.

CHAPTER 4

Results

"There is just news. There is no good or bad." - Master Oogway, Kung Fu Panda

4.1 Training and Validation

The model was configured using the specifications outlined in Section 3.5.1 and kept consistent across all tests. The datasets of various window sizes were all generated according to the process shown in Figure 3.1 and used to train different PointNet++ models, and the test data was generated in the same way but without augmentations. The testing set was then used to evaluate the final performance across models. A summary of the experimental process is shown in Figure 4.1, using window widths selected from Table 4.1.

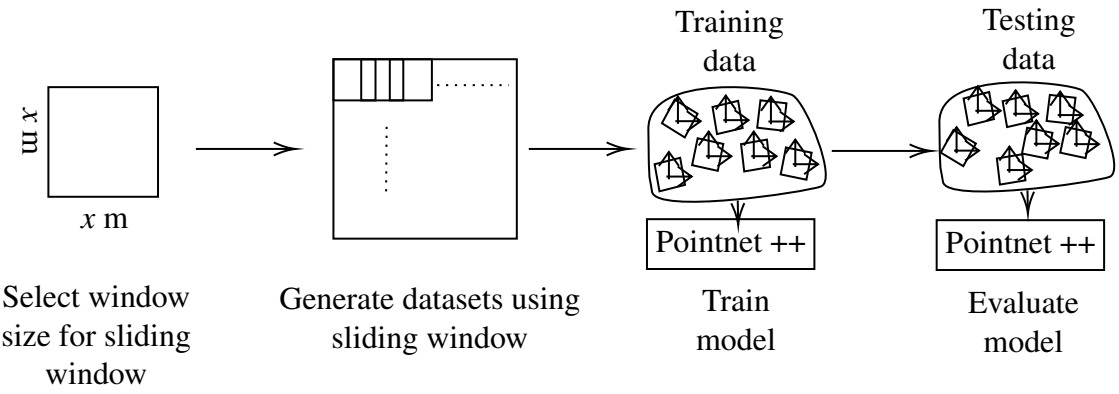


Figure 4.1: A summary of the experimental process showing the experimental process. It begins with the window size being selected from Table 4.1, datasets being generated according to Figure 3.1, then using these datasets to train and then test the model.

Window Width (m)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

Table 4.1: Window widths used to train various PointNet++ semantic segmentation models for testing.

While training the models it was found that training results improved based solely on having a smaller window size, as shown in Figures 4.2, 4.3, and 4.4. This relationship breaks down for IoU results at the point of reaching a window width of 0.5m, shown in Figure 4.2.

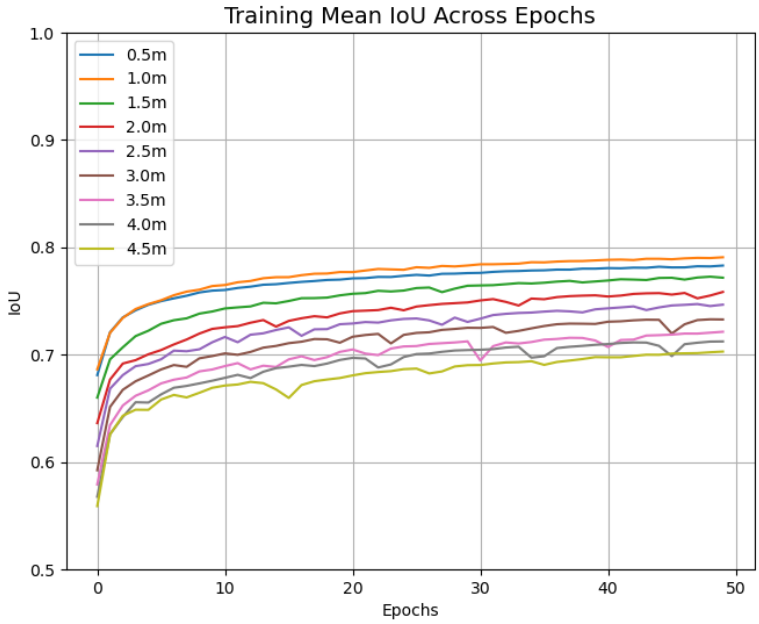


Figure 4.2: Mean IoU while using different window widths to train different models over 50 epochs. IoU was calculated per class for the labels 'Ground', 'Stem', 'Undergrowth' and 'Foliage' and averaged for mean IoU. While IoU rose generally as the window size shrunk the 0.5m window performed worse than the 1m window.

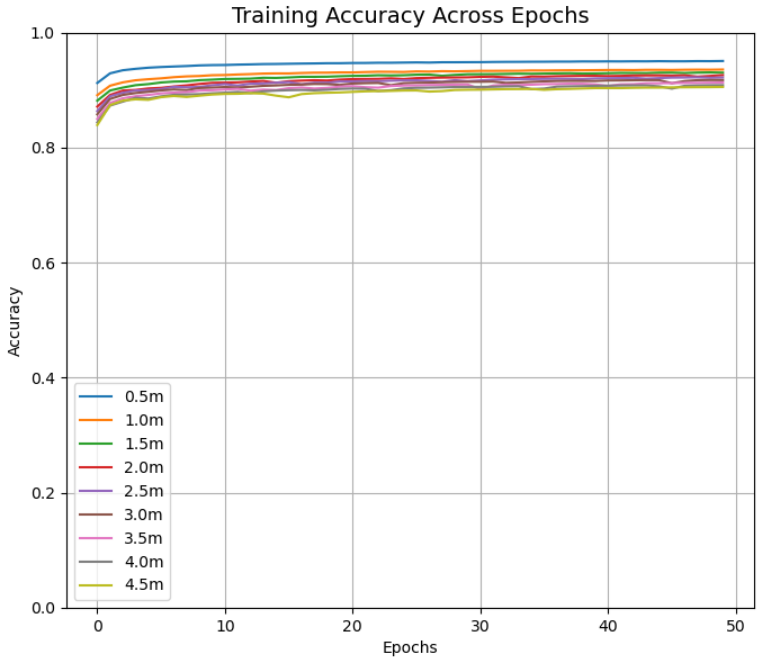


Figure 4.3: Accuracy while using different window widths to train different models over 50 epochs. Accuracy consistently rose as window size shrunk.

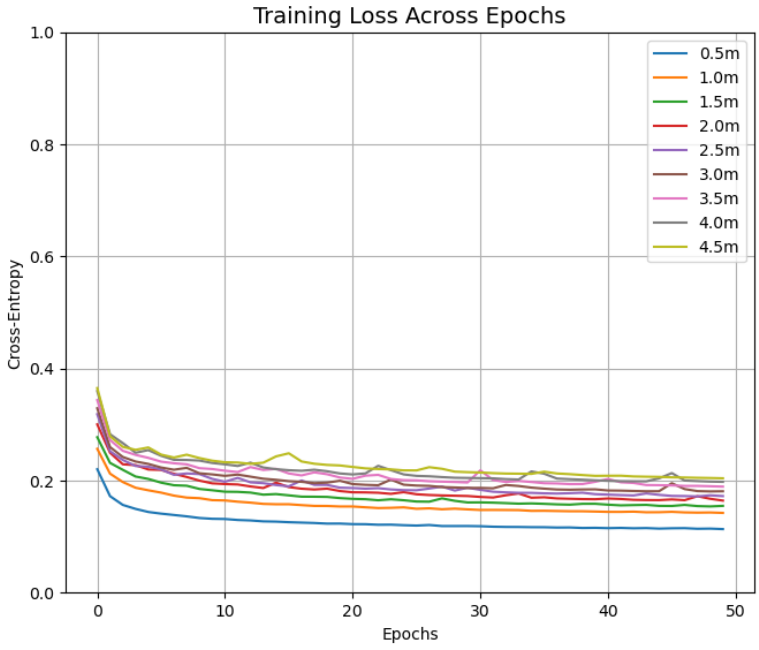


Figure 4.4: Loss while using different window widths to train different models over 50 epochs. The loss metric used was cross-entropy. Loss consistently dropped as the window size shrunk, indicating that smaller windows were identified more confidently by the model.

Consistently over testing, it appears results are more favourable when using a smaller window size barring the IoU dropping at a window width of 0.5m. Importantly, the validation data also generally rose along with the training data in following this trend, indicating that the results did not improve as a result of overfitting but rather improved model capabilities. These results can be seen more clearly when comparing the final training values for mean IoU, accuracy, and cross-entropy alongside their validation results as shown in Figures 4.5, 4.6, 4.7, 4.8. Figures 4.5 and 4.6 indicate that the models get better at identifying individual classes on the training and validation data up until there is a drop for some of the labels at 0.5m window width. This does not apply for the Foliage label, and as there is a high number of foliage points in the dataset it results in the accuracy following the trend of foliage closely, having the same rise and falls in both training and validation data between Figures 4.5 and 4.7. The loss values shown in Figure 4.8 indicates that the labels are still being assigned with more confidence by the network at a window size of 0.5m despite performing worse.

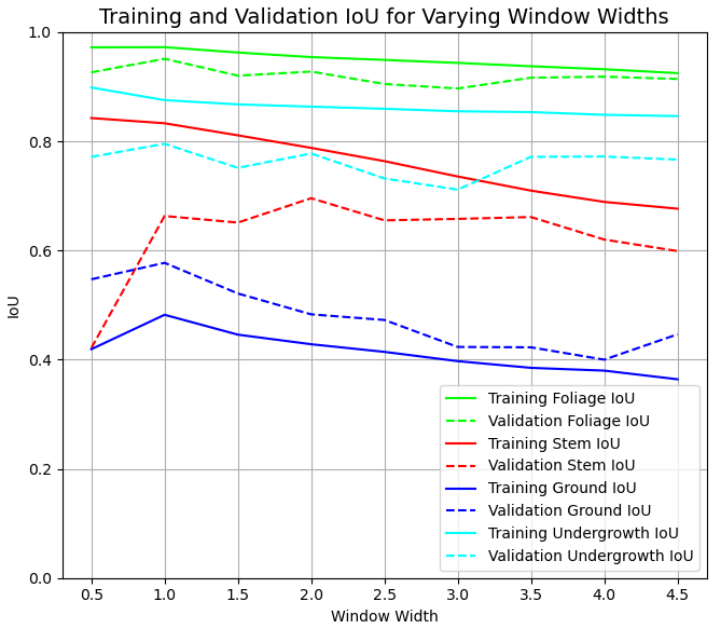


Figure 4.5: Final per class IoU values for all window widths for both training and validation. IoU was calculated per class for the labels 'Ground', 'Stem', 'Undergrowth' and 'Foliage'. There is a noticeable drop in validation Stem IoU and validation and training Ground IoU results once reaching a window width of 0.5m

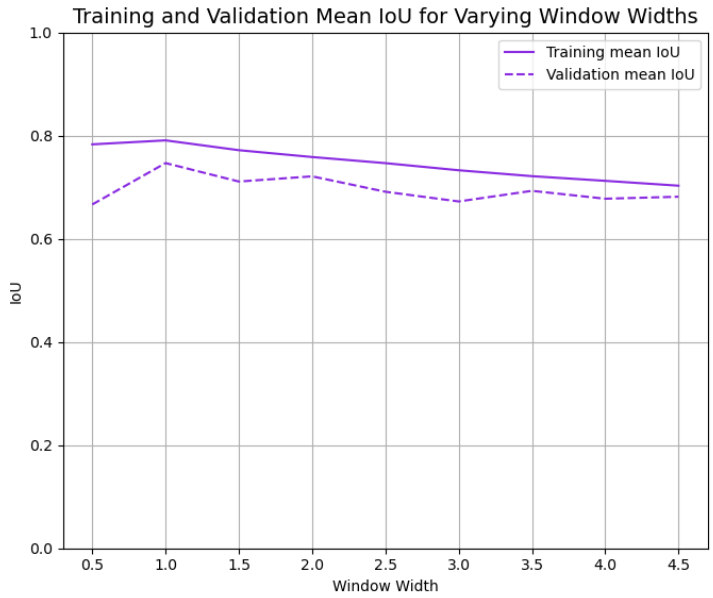


Figure 4.6: Final values of mean IoU for all window widths for both training and validation. IoU was calculated per class for the labels 'Ground', 'Stem', 'Undergrowth' and 'Foliage' and averaged to find the mean IoU. There is an overall drop in mean IoU for training and validation at 0.5m window width while it had risen consistently above this.

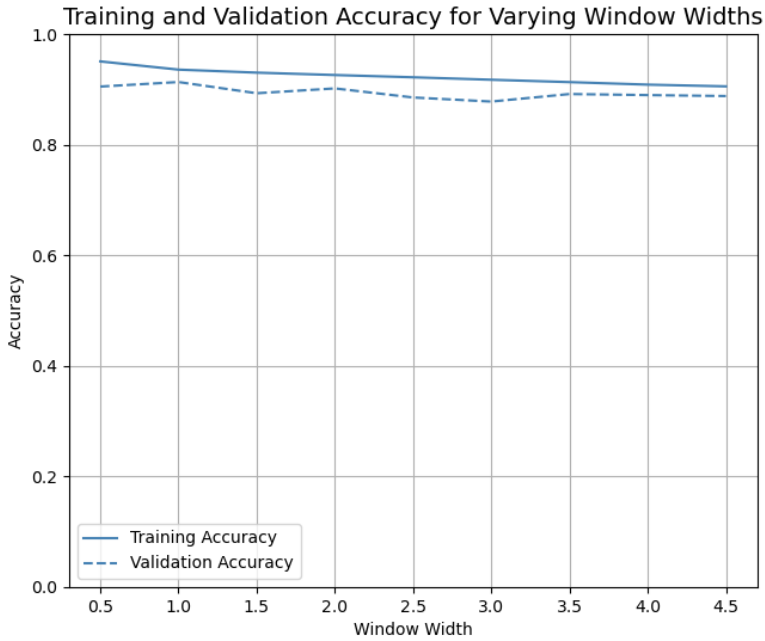


Figure 4.7: Final values of accuracy across all window widths for both training and validation. The trend for accuracy data closely follows the IoU results of the Foliage label shown in Figure 4.5 due to the high number of Foliage points present in the dataset.

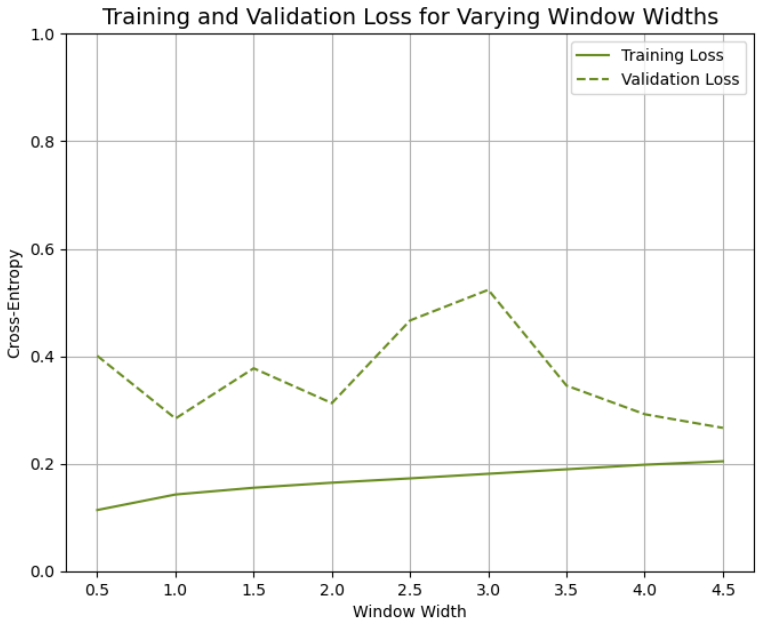


Figure 4.8: Final values of loss across all window widths for both training and validation. The loss metric used was cross-entropy. As window width gets smaller the loss drops consistently, indicating that the network is more confident in the decisions its making. The erratic validation loss suggests that this confidence is misplaced.

4.2 Testing

It is good to see how the performance changes across the training and validation data, but it is most important to analyse this on the separate testing dataset that the network has not seen, and is as large as the training and validation set combined. This will give a good indication of how well the network has learned to generalise across different plots and learn key features. A visualisation of the testing set is shown in Figure 3.2. This dataset does have a slight slope of the ground compared to the more level training and validation sets as they were gathered in different areas, so we would expect ground and undergrowth predictions to worsen. Figures 4.9 and 4.10 show the results of how the models trained on various window widths performed when tested on a separate dataset, and the results largely parallel those that were seen in the validation and training sets. The IoU results improved as the window width got smaller and peaked with the best overall results at 1.0m and dropped at 0.5m. This seems to indicate that the best window size for this dataset lies somewhere between 1.5m and 0.5m. The accuracy rose consistently and did not drop at the smallest window size, however this is likely skewed due to foliage appearing significantly more often than other points. We can try and correct for this by using normalised IoU values gathered from a confusion matrix. The confusion matrix represents which labels are mislabelled as others by the network, and the values in each row can be normalised to sum to 1 in each row. The values in the confusion matrix represent true positives, true negatives, etc, and can be used to calculate these normalised IoU values that account for how frequently a label occurs with Equation 3.1. A confusion matrix and a normalised confusion matrix are shown

in Figures 4.12 and 4.13, with the normalised IoU results in Figure 4.14. Remaining confusion matrices are included in Appendix A.

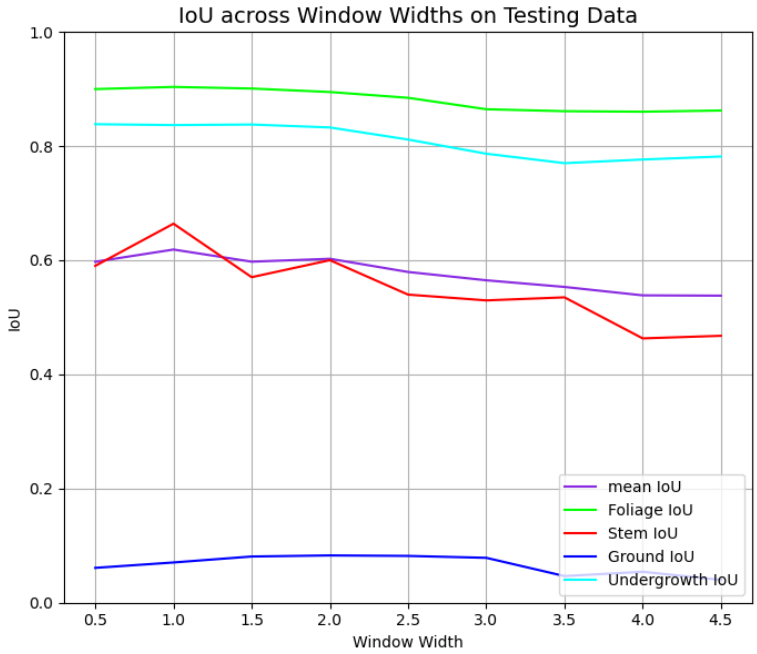


Figure 4.9: IoU results across various models trained on different window widths when using the testing dataset. There is a drop in results in Stem IoU on either side of the 1.0m window width, indicating that below this width fewer stems are being identified successfully.

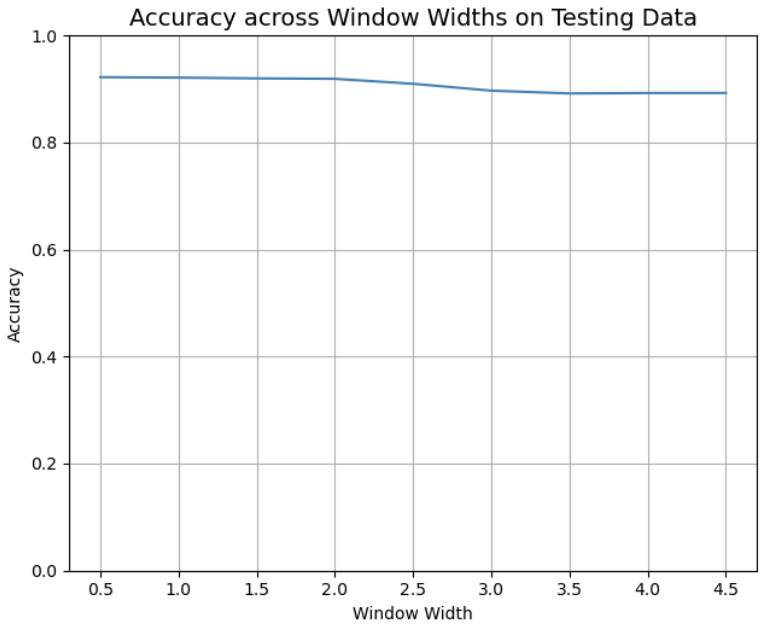


Figure 4.10: Accuracy results across various models trained on different window widths when using the testing dataset. The accuracy results closely follow the results of the Foliage IoU shown in Figure 4.9 due to the high number of Foliage points present in the dataset.

Figure 4.11: Loss results across various models trained on different window widths when using the testing dataset. The loss metric used was cross-entropy loss.

Figure 4.12: Confusion matrix on the testing dataset for a window width of 1.0m, with each box indicating the number of points that have been predicted with each label when compared to the true label.

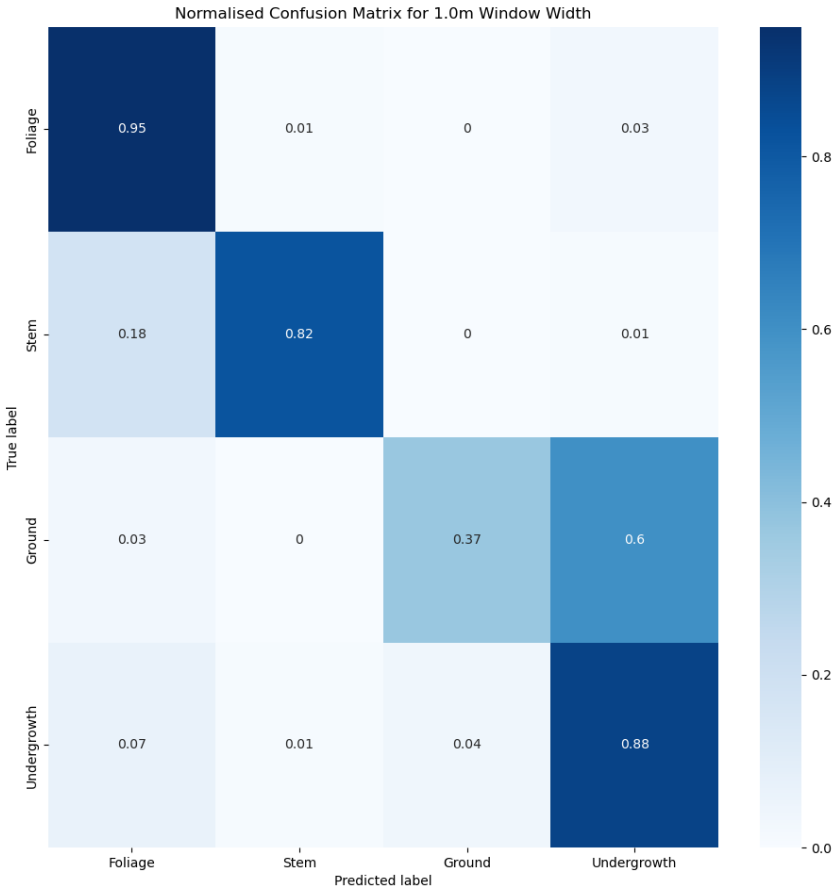


Figure 4.13: Normalised confusion matrix on the testing dataset for a window width of 1.0m, with each box indicating the proportion of points that have been predicted with each label compared to their true labels. The numbers have been normalised on the number of true labels that appear, such that each row sums to 1. This allows for a greater ease of visualising the proportion that each label is identified correctly. It is apparent that the stem points are most commonly mislabelled as foliage, and that ground points are most commonly mislabelled as undergrowth.

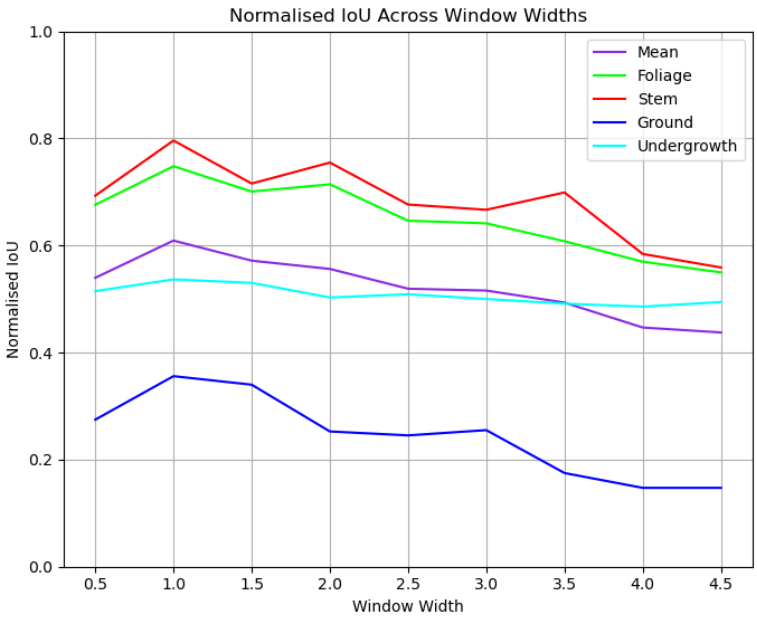


Figure 4.14: Normalised IoU results across various models trained on different window widths when using the testing dataset. These IoU values were calculated using Equation 3.1 and the normalised confusion matrix in Figure 4.13. These IoU values have effectively been 'corrected' to account for the proportion of true positives, true negatives, false positives and false negatives that appear. After this normalisation the drop in results at a window width of 0.5m after the relatively steady rise before this is apparent.

These results shown in Figures 4.12 and 4.13 allow a greater understanding of where the network struggles and where it thrives. A first point to analyse is that the results are significantly skewed towards foliage and undergrowth in Figure 4.12 as they have more examples present, which overshadows the results of stem and ground points, making it difficult to tell which labels are mistaken for each other. After normalisation shown in Figure 4.13 the results are much more interpretable, and the proportion of correct identifications to mistakes on each label is much more apparent. The vast majority of foliage labels are predicted correctly, with few mistakes being made. Some points which are in reality the stem are often mislabelled as being foliage. The weakest label is the ground, where a higher proportion of ground is mislabelled as undergrowth than what is labelled correctly. Undergrowth itself has a high number of true positives, but that is likely a result of the large proportion of ground points that are mislabelled; essentially the network is labelling the majority of low altitude points as undergrowth due to the lower number of ground points. A similar phenomenon is likely occurring with foliage and stem labels.

Beyond this, the normalised IoU results in Figure 4.14 seem to further support the IoU results seen earlier and the idea that the drop in results at a window width of 0.5m is significant. After being normalised it is more clear than in Figure 4.9 that all labels suffer at this lower window size, and that this kind of drop in results does not occur previously, with the mean IoU consistently rising up until that point. In order to better

understand the mistakes that the network makes a selection of specific windows are included in Figures 4.15, 4.16, and 4.17. In these examples some of the per class IoU values are 0 as they do not appear in the example. It is important to note that the overall IoU values calculated for Figures 4.9 and 4.14 were not dragged down incorrectly by these examples, as the IoU values were calculated from first principles using the overall true positives, true negatives, false positives, and false negatives gathered in the confusion matrices.

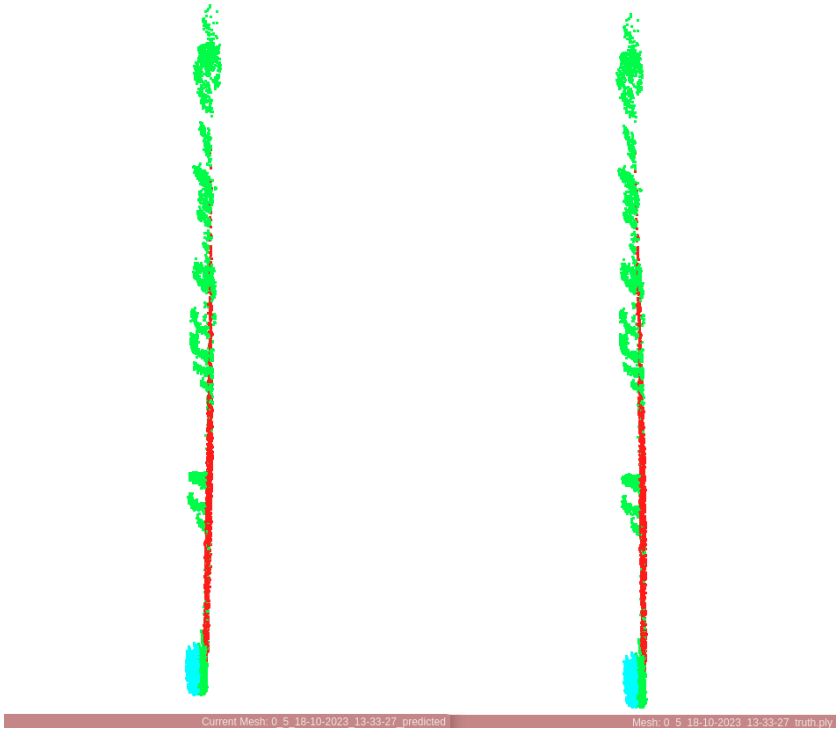


Figure 4.15: Visualisation of the labels predicted by the trained model (left) when compared to the ground truth labels (right) on a 0.5m window width example. Accuracy: 0.78, Loss: 1.81, mean IoU: 0.48, Foliage IoU: 0.61, Stem IoU: 0.73, Ground IoU: 0.00, Undergrowth IoU: 0.60

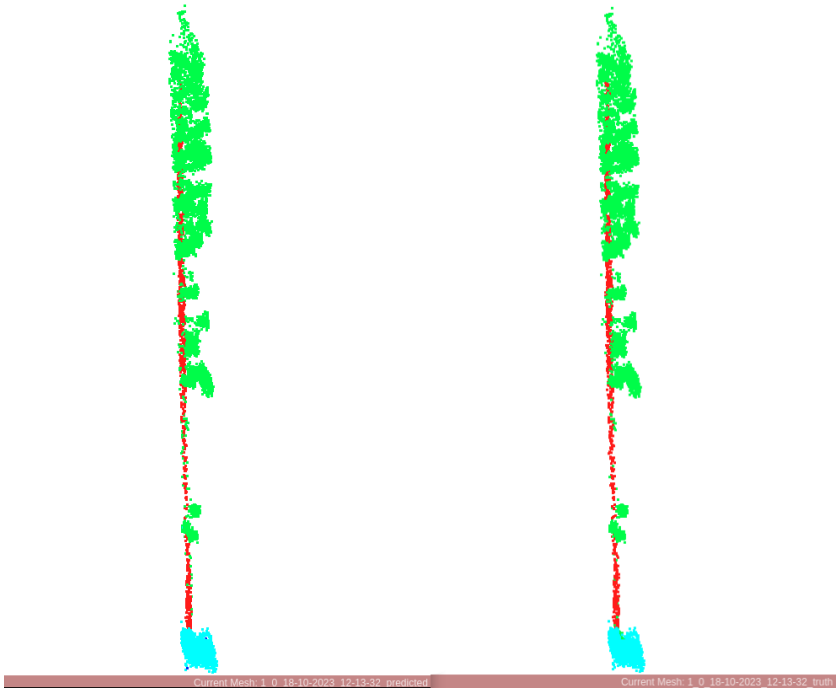


Figure 4.16: Visualisation of the labels predicted by the trained model (left) when compared to the ground truth labels (right) on a 1.0m window width example. Accuracy: 0.96, Loss: 0.17, mean IoU: 0.67, Foliage IoU: 0.95, Stem IoU: 0.80, Ground IoU: 0.00, Undergrowth IoU: 0.93

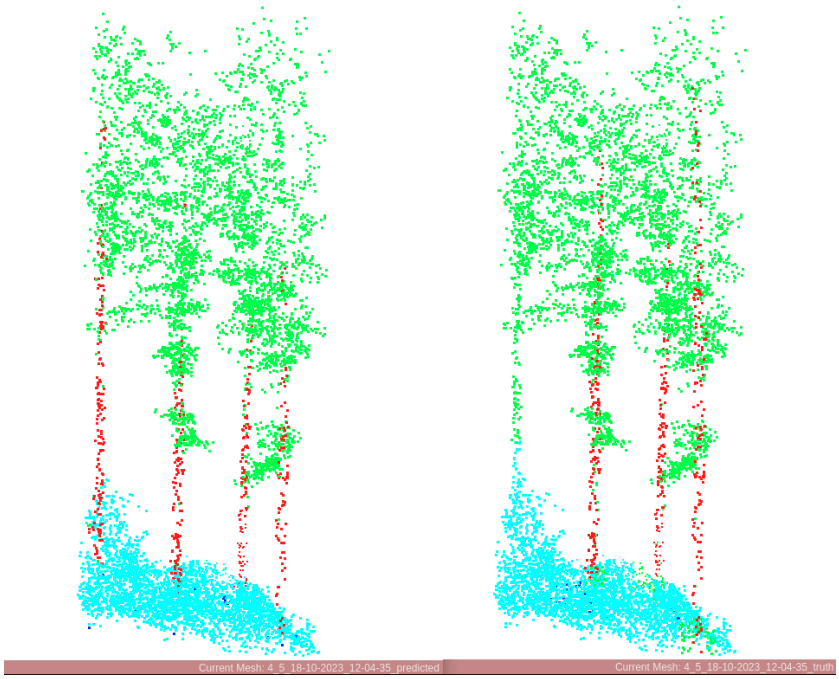


Figure 4.17: Visualisation of the labels predicted by the trained model (left) when compared to the ground truth labels (right) on a 4.5m window width example. Accuracy: 0.92, Loss: 0.36, mean IoU: 0.56, Foliage IoU: 0.90, Stem IoU: 0.44, Ground IoU: 0.00, Undergrowth IoU: 0.91

4.3 Label Occurrence in Data

After observing the dramatic changes in the results of training and testing at a window width of 0.5m it was decided that further analysis of the makeup of the datasets would be conducted. Specifically to gather data on how often each label appeared proportionally in the overall data. This information is presented in Figures 4.18 and 4.19. Figure 4.18 shows a sudden change in the proportion of label representation at 0.5m window width, and the change is against the trend of the data previously. In every instance where a labels representation had been falling it now suddenly rises, and where it was rising it now suddenly falls. This suggests a failure in the sliding window approach at this size to accurately represent the original dataset.

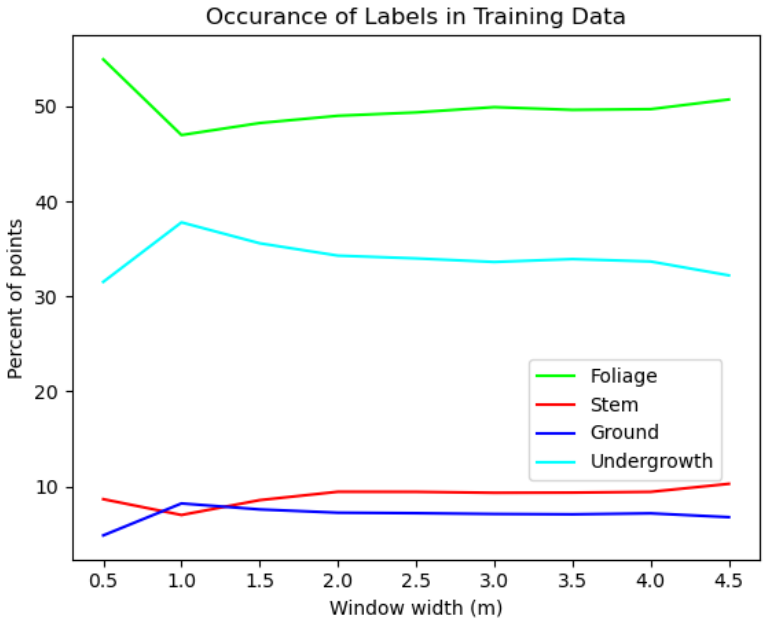


Figure 4.18: How often each label appears proportionally in the training data after applying the sliding window. There is a sharp change in the representation of all labels in the training data as the window size drops to 0.5m, contrasting greatly with the rest of the data. This would indicate that the training data no longer represents the dataset accurately.

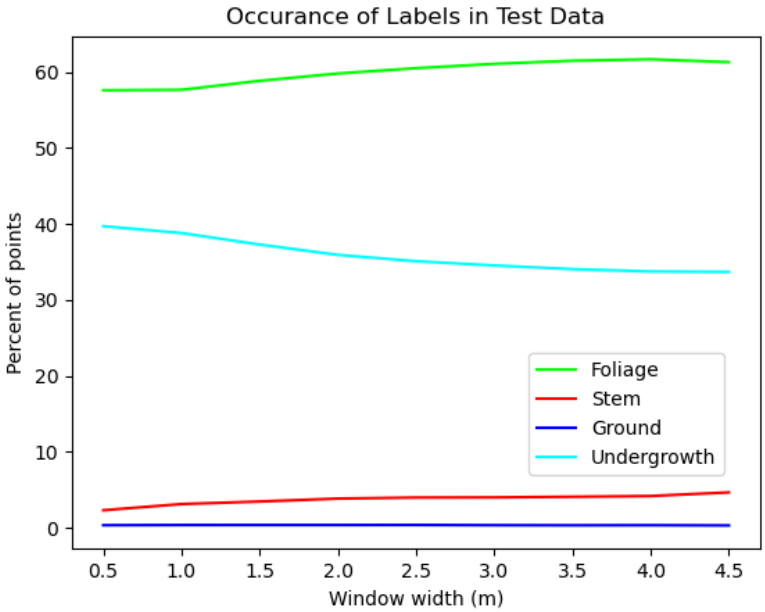


Figure 4.19: How often each label appears proportionally in the testing data after applying the sliding window. There begins to be a higher proportion of undergrowth representation at the cost of other labels.

4.4 Computational Results

Additionally to the results of evaluating how effective the models get as window size changes, it is valuable to understand the storage space of the windows at this low size, and it is shown in Table 4.2 that the growth in file size is exponential. Training times for each model were not recorded, but the longest was the 0.5m window width taking 38 hours, and the shortest was 4.5m window width taking only 2 hours.

Window Width	Testing File Size (MB)	Training File Size (MB)
4.5	21.5	354.6
4.0	27.9	469.6
3.5	35.3	582.4
3.0	47.6	745.4
2.5	63.6	996.9
2.0	97.3	1501.4
1.5	168.1	2535.6
1.0	360.8	5223.3
0.5	1330.8	8781.7

Table 4.2: Window width compared to size of data size after using the sliding window in the xy-plane when using a test dataset originally 604.2MB and training dataset originally 345.4MB

CHAPTER 5

Discussion

"It is good to have an end to journey towards; but it is the journey that matters, in the end." - Ursula K. Le Guin, The Left Hand of Darkness

5.1 Summary of Results

This thesis aimed to investigate how applying data pre-processing methods on highly unstructured forest point clouds can improve results for semantic segmentation. This was attempted by investigating the sliding window approach to a large forest point cloud and performing experiments to determine if the size of this sliding window affected the performance of a PointNet++ based network. The experiments were performed by varying the size of the window as being between 0.5m and 4.5m in increments of 0.5m while keeping the number of points sampled from this region consistent at 8192 points. These experiments determined that in this instance using the forest datasets described in Section 3.1 the ideal window width lies somewhere in the region of 0.5m to 1.5m, indicated by the drop in IoU across all labels in the testing dataset at the 0.5m window width shown in Figures 4.9 and 4.14. The results had previously only improved before reaching this smallest window width and dropping, which had indicated that having a higher number of more densely populated point clouds provides more effective training data to the network than using more sparsely populated point clouds that show a larger contextual area. However this trend could not have logically continued until reaching a window width of 0.0m, and at some point a limit had to be reached.

5.2 Justification of Results

It is my conclusion that the reasoning behind this is due to the proportion of windows that lack a sufficient amount of contextual information at these smaller sizes. This appears to be validated by Figure 4.18 which illustrates the relationship between representation of each label in the training data after applying the sliding window. In Figure 4.18 the previous trends of the data are suddenly reversed at 0.5m window width, and the proportional representation no longer realistically represents the original data. This would lead to the model being trained using information that is not representative, which leads to a model that cannot identify points as effectively. In the testing data in Figure

4.19 it is clear that the window size of 0.5m is not so low as to have this effect, and this is likely due to the structural differences in the original forests where the training data consists of evenly spaced trees approximately 4m apart, while the trees in the testing data are more chaotically spaced, many being approximately 3m apart. It is likely that this 'critical window size' at which the proportion of labels changes dramatically is likely linked to the structure of the forest, however further testing on different forests would be needed to verify this.

5.3 Model Strengths and Weaknesses

The areas where the network struggles the most were identified by the confusion matrix 4.13 and the additional matrices in Appendix A as being stem points that are mislabelled as foliage, and ground points that are mislabelled as undergrowth. In the testing set used there may have been a disadvantage to the network in identifying ground and undergrowth effectively due to the slope of the surface when compared to the more even training dataset. In looking at specific visualisations within the testing set the numbers in the confusion matrices are validated, we can see that the top of the stems are often mislabelled while the rest of them are labelled more accurately. This is seen well in Figures 4.15 and 4.17. Figure 4.17 also shows a tree that was labelled with greater effectiveness than the true labels, identifying a stem that was mislabelled as foliage. In some other examples there are small vertical growths labelled as undergrowth that are 'mislabelled' as stem points. This reflects that the network has learnt good generalisations of what constitutes a stem, while not quite agreeing with the labeller's interpretations of undergrowth vs stems in some instances.

5.4 Key Findings

Overall coming back to the original purpose of detecting the trees within the forest, the most important aspect to understand is the stem. Fortunately it appears that no stems are missed by the network, simply some that are underestimated and some that are too short to be realistically considered as trees. This problem can be remedied using other solutions such as using stem fitting models to the existing points and using the height of the tree overall to estimate where the stem ends when a tree has a sufficient number of stem points recognised, as was done in Hyyppä et als. study [12]. Additionally, more simple factors such as diameter at breast height and growth patterns of the tree can be measured directly from the points without the full height of the stem present [15].

5.5 Relation to Literature

In other relations to previous works, it would be constructive to analyse the models performance more on environments where the ground points have been previously removed using a DTM such as in [1], or at least where the ground is flattened to normalise tree height such as in [48]. This would let the network do less work in terms of compensating for the slope of the ground. This among other ideas such as fine tuning

the network more closely and using more varied training data would likely improve the overall performance of the network, and ideally make it easier to see the effects of changing window width more closely.

The results achieved in terms of successful semantic segmentation accuracy do not reach those achieved in other papers [1, 13–15, 18], as can be seen in Table 5.1 they all fall short except for foliage, and at the severe cost of IoU for the other labels. Fortunately, achieving the best results possible was not the goal of the experiment. The experiment was intended to explore the impacts of window size on semantic segmentation, inspired by the arbitrary region size used by Krisanski et al. [14], and the unknown size of the sliding *xy*-window used by Windrim and Bryson [1] to find birds-eye projections of the point clouds, and the results have indicated strongly that the size of the window does change results. It has shown that smaller windows down to a lower limit will result in better semantic segmentation (as measured by IoU values as shown in Figures 4.9 and 4.14) with the lower limit in this case being somewhere between a window width of 0.5m and 1m on the training data used.

	Ground	Foliage	Stem	Undergrowth	Other	Vegetation	CWD
Shen et al. [15] A	0.917	0.889	0.875		0.062		
Shen et al. [15] B	0.977	0.848	0.747		0.916		
Krisanski et al. [14]	0.891		0.913			0.936	0.407
Ours	0.070	0.904	0.664	0.837			

Table 5.1: Best IoU values for different labels used in semantic segmentation of forest points across different papers. The best IoU values in this thesis were from a window width of 1.0m. Shen et al. [15] A and B refer to the datasets Bajia Park and Gaotang Park respectively. The label CWD refers to 'coarse woody debris'. The label vegetation is a combination of understory vegetation and canopy vegetation.

5.6 Limitations

Along with the structural limitations discussed in Section 3.6, the results of the experiments conducted lead to some natural real limitations in terms of applications and implications that should be used to contextualise the results.

The amount of time taken for training rose significantly along with the size of the data as the size of the window decreased, shown in Section 4.4. This is due to a natural flaw in the experiment; that the smaller window sizes naturally produced more windows, which led to more training data, which generally improves results. The increasing time and space requirements of this limits small windows sizes to become impractical in some real life applications. When using the trained model on a much larger plot or a larger number of plots it is easily possible to double the size of the data as has been done for this case in the 0.5m window example. This problem is worse in the training phase with the smallest window increasing to 25 times the size of the original data.

The use of only a single forest plot for training the models and another forest for testing

leads to the results only being applicable to this limited context. Using a larger range of forests consisting of more varying structures to train a number of models would enable a hypothesis to be formed about what aspect or aspects of the point cloud determines the lower limit for window size to achieve the best semantic segmentation results.

CHAPTER 6

Conclusions and Future Work

"Your future is whatever you make it, so make it a good one" - Dr. Emmett L. Brown, Back to the Future III

This thesis approached the challenge of determining how applying data pre-processing methods on highly unstructured forest point clouds could improve results for semantic segmentation. This is a concern because semantic segmentation of forest point clouds is used to more effectively understand both plantation and native forests from a quantitative standpoint. Understanding the forests better allows them to be better understood for conservation purposes, logging, bushfire prevention, and general health management.

The approach for this was to experiment on changing the window sizes used for subsampling these large point clouds into more manageable windows that the point-based deep learning networks are able to process. The point clouds are initially too big for the networks to handle directly, but subsampling them into a large number of smaller point clouds remedies this problem.

A PointNet++ [3] based network was trained from a dataset that was split into a number of overlapping square windows with widths ranging from 4.5m to 0.5m. The different models that were trained using these window sizes were then tested on a separate dataset and results were gathered to determine how the various models performed on semantic segmentation. The results indicated that as the window sizes shrunk that the IoU values (representing effectiveness of the network) rose up until a window width of 1.0m, and then fell again at a window width of 0.5m. This indicates that smaller windows that more densely contain points help the network learn features better, up until a lower limit in size that appears to be related to the structure of the forest used (e.g. how far apart trees are spaced). The best IoUs achieved by the model were: Ground 0.070, Foliage 0.904, Stem 0.664, and Undergrowth 0.837, with an overall accuracy of 92.1%.

Determining how the window size used affects the results of semantic segmentation is used is useful in allowing future implementations of these networks to more carefully choose the window size used rather than using an arbitrary size as has been done previously. Further research into this relationship between the forest structure and the ideal window size for improving results will allow for forests to be more effectively understood and managed.

6.1 Future Work

There are a number of avenues to pursue in order to more thoroughly test ways to improve the results of point cloud segmentation networks on forest environments, some aimed at addressing the limitations outlined in Section 3.6.

6.1.1 Alternative Point-Based Networks

Examining how the results vary and change when using networks other than PointNet++ [3] would be valuable in understanding how other networks are able to handle the same challenge. Networks such as PointCNN [27], PointSIFT [25], and RandLA-Net [87] would be good starting points for this.

6.1.2 Forest Variety

Evaluating the current models on a wider variety of forests would help gain a greater understanding of how the various window widths affect the generalisation ability of the network, especially when comparing plantation forests, recreational forests, and more varied native forests containing highly apical trees. Using a wider ranges of forests and analysing statistics about what proportion of windows contain which semantic labels would allow for a more thorough understanding of what causes the peak in results as window sizes get smaller, and the drop in results beyond that.

6.1.3 Window Geometry

The window shape used was a square, and had a 30% overlap with other windows around it, however this was arbitrary and not tested, and other window shapes and amounts of overlap will likely yield different results as to indicating which window size is best for the task at hand.

6.1.4 Voting System

The implementation of a voting system in which each individual point is only assigned a final label once all the windows which contain it have been classified would allow for a more thorough pipeline in which the final forest can be reconstructed after predictions are complete. Some other 3D point cloud learning networks already utilise this [23]. This would also reduce random error in the system due to the repetition of predictions.

6.1.5 Bi-Density Windows

As the window sizes changed and the number of points in the window stayed the same, experiments were changing the density of the samples along with the size they represented. In order to provide the network with both dense local data and sparse contextual information a bi-density window experiment combining both a small dense

window and a large sparse window may lead to reaping the benefits from both approaches without the drawbacks of either.

6.1.6 Alternative Environments

The proof that sectioning the forest into windows for point cloud semantic segmentation was discussed in this thesis, however the applications extend well beyond forests. Other environments that consist of repetitive structures sharing similar geometric features also have the potential to be split up using this method. Such environments include crop fields, gardens, or underwater environments. Experimenting with this methodology on these environments has the potential to improve semantic segmentation results.

Bibliography

[1] L. Windrim and M. Bryson, "Detection, segmentation, and model fitting of individual tree stems from airborne laser scanning of forests using deep learning," *Remote Sensing*, vol. 12, no. 9, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/9/1469

[2] D. Griffiths, "Pointnet++ tensorflow 2.0 layers," https://github.com/dgriffiths3/pointnet2-tensorflow2, 2020.

[3] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[4] Food and Agriculture Organization of the United Nations, "The state of the world's forests 2020 : forests, biodiversity and people," pp. xvi – 12, 2020. [Online]. Available: http://digitallibrary.un.org/record/3978392

[5] N. L. Harris, D. A. Gibbs, A. Baccini, R. A. Birdsey, S. de Bruin, M. Farina, L. Fatoyinbo, M. C. Hansen, M. Herold, R. A. Houghton, P. V. Potapov, D. R. Suarez, R. M. Roman-Cuesta, S. S. Saatchi, C. M. Slay, S. A. Turubanova, and A. Tyukavina, "Global maps of twenty-first century forest carbon fluxes," *Nature Climate Change*, vol. 11, no. 3, pp. 234–240, 2021. [Online]. Available: https://doi.org/10.1038/s41558-020-00976-6

[6] ABARES, *Australia's State of the Forests Report 2018*. Montreal Process Implementation Group for Australia and National Forest Inventory Steering Committee, 2018. [Online]. Available: https://www.agriculture.gov.au/abares/forestsaustralia/sofr/sofr-2018

[7] L. Whittle, "Snapshot of australia's forest industry," 2018. [Online]. Available: https://www.agriculture.gov.au/abares/products/insights/snapshot-of-australias-forest-industry

[8] P. W. West, *Tree and forest measurement*, 3rd ed. Springer International Publishing, 2015.

[9] J. Hyyppä and M. Inkinen, "Detecting and estimating attributes for single trees using laser scanner. photogramm j finl," *The Photogrammetric Journal of Finland*, vol. 16, pp. 27–42, 01 1999.

[10] Y. Wang, J. Hyyppä, X. Liang, H. Kaartinen, X. Yu, E. Lindberg, J. Holmgren, Y. Qin, C. Mallet, A. Ferraz, H. Torabzadeh, F. Morsdorf, L. Zhu, J. Liu, and

P. Alho, “International benchmarking of the individual tree detection methods for modeling 3-d canopy structure for silviculture and forest ecology using airborne laser scanning,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 9, pp. 5011–5027, 2016.

[11] E. Næsset, T. Gobakken, J. Holmgren, H. Hyyppä, J. Hyyppä, M. Maltamo, M. Nilsson, H. Olsson, Åsa Persson, and U. Söderman, “Laser scanning of forest resources: the nordic experience,” *Scandinavian Journal of Forest Research*, vol. 19, no. 6, pp. 482–499, 2004. [Online]. Available: https://doi.org/10.1080/02827580410019553

[12] E. Hyyppä, A. Kukko, H. Kaartinen, X. Yu, J. Muhojoki, T. Hakala, and J. Hyyppä, “Direct and automatic measurements of stem curve and volume using a high-resolution airborne laser scanning system,” *Science of Remote Sensing*, vol. 5, p. 100050, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666017222000128

[13] F. Wang and M. Bryson, “Tree segmentation and parameter measurement from point clouds using deep and handcrafted features,” *Remote Sensing*, vol. 15, no. 4, 2023. [Online]. Available: https://www.mdpi.com/2072-4292/15/4/1086

[14] S. Krisanski, M. S. Taskhiri, S. Gonzalez Aracil, D. Herries, and P. Turner, “Sensor agnostic semantic segmentation of structurally diverse and complex forest point clouds using deep learning,” *Remote Sensing*, vol. 13, no. 8, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/8/1413

[15] X. Shen, Q. Huang, X. Wang, J. Li, and B. Xi, “A deep learning-based method for extracting standing wood feature parameters from terrestrial laser scanning point clouds of artificially planted forest,” *Remote Sensing*, vol. 14, no. 15, 2022. [Online]. Available: https://www.mdpi.com/2072-4292/14/15/3842

[16] B. Liu, S. Chen, H. Huang, and X. Tian, “Tree species classification of backpack laser scanning data using the pointnet++ point cloud deep learning method,” *Remote Sensing*, vol. 14, no. 15, 2022. [Online]. Available: https://www.mdpi.com/2072-4292/14/15/3809

[17] Z. Luo, Z. Zhang, W. Li, Y. Chen, C. Wang, A. A. M. Nurunnabi, and J. Li, “Detection of individual trees in uav lidar point clouds using a deep learning framework based on multichannel representation,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–15, 2022.

[18] H. J. Persson, K. Olofsson, and J. Holmgren, “Two-phase forest inventory using very-high-resolution laser scanning,” *Remote Sensing of Environment*, vol. 271, p. 112909, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425722000232

[19] Y. Qin, A. Ferraz, C. Mallet, and C. Iovan, “Individual tree segmentation over large areas using airborne lidar point cloud and very high resolution optical imagery,” in *2014 IEEE Geoscience and Remote Sensing Symposium*, 2014, pp. 800–803.

[20] X. Zou, M. Cheng, C. Wang, Y. Xia, and J. Li, "Tree classification in complex forest point clouds based on deep learning," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 12, pp. 2360–2364, 2017.

[21] E. Ayrey and D. J. Hayes, "The use of three-dimensional convolutional neural networks to interpret lidar for forest inventory," *Remote Sensing*, vol. 10, no. 4, 2018. [Online]. Available: https://www.mdpi.com/2072-4292/10/4/649

[22] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[23] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 12, pp. 4338–4364, 2021.

[24] W. Song, Z. Liu, Y. Guo, S. Sun, G. Zu, and M. Li, "Dgpolarnet: Dynamic graph convolution network for lidar point cloud semantic segmentation on polar bev," *Remote Sensing*, vol. 14, no. 15, 2022. [Online]. Available: https://www.mdpi.com/2072-4292/14/15/3825

[25] M. Jiang, Y. Wu, and C. Lu, "Pointsift: A sift-like network module for 3d point cloud semantic segmentation," *CoRR*, vol. abs/1807.00652, 2018. [Online]. Available: http://arxiv.org/abs/1807.00652

[26] J. Du, Z. Jiang, S. Huang, Z. Wang, J. Su, S. Su, Y. Wu, and G. Cai, "Point cloud semantic segmentation network based on multi-scale feature fusion," *Sensors*, vol. 21, no. 5, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/5/1625

[27] Y. Li, R. Bu, M. Sun, and B. Chen, "Pointcnn," *CoRR*, vol. abs/1801.07791, 2018. [Online]. Available: http://arxiv.org/abs/1801.07791

[28] S. A. Bello, S. Yu, C. Wang, J. M. Adam, and J. Li, "Review: Deep learning on 3d point clouds," *Remote Sensing*, vol. 12, no. 11, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/11/1729

[29] W. Liu, J. Sun, W. Li, T. Hu, and P. Wang, "Deep learning on point clouds and its application: A survey," *Sensors*, vol. 19, no. 19, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/19/4188

[30] N. Mehendale and S. Neoge, "Review on lidar technology," *SSRN Electronic Journal*, 01 2020.

[31] Hughes Aircraft Company, "Colidar," *Journal of the American Society for Naval Engineers*, vol. 73, no. 4, pp. 681–684, 1961. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1559-3584.1961.tb03324.x

[32] G. G. Goyer and R. Watson, "The laser and its application to meteorology," *Bulletin of the American Meteorological Society*, vol. 44, no. 9, pp. 564–570, 1963. [Online]. Available: http://www.jstor.org/stable/26247220

[33] R. T. H. Collis, "Lidar," *Appl. Opt.*, vol. 9, no. 8, pp. 1782–1788, Aug 1970. [Online]. Available: https://opg.optica.org/ao/abstract.cfm?URI=ao-9-8-1782

[34] J. P. Underwood, G. Jagbrant, J. I. Nieto, and S. Sukkarieh, "Lidar-based tree recognition and platform localization in orchards," *Journal of Field Robotics*, vol. 32, no. 8, pp. 1056–1074, 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21607

[35] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixão, F. Mutz, L. de Paula Veronese, T. Oliveira-Santos, and A. F. De Souza, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113816, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742030628X

[36] M. A. Canuto, F. Estrada-Belli, T. G. Garrison, S. D. Houston, M. J. Acuña, M. Kováč, D. Marken, P. Nondédéo, L. Auld-Thomas, C. Castanet, D. Chatelain, C. R. Chiriboga, T. Drápela, T. Lieskovský, A. Tokovinine, A. Velasquez, J. C. Fernández-Díaz, and R. Shrestha, "Ancient lowland maya complexity as revealed by airborne laser scanning of northern guatemala," *Science*, vol. 361, no. 6409, p. eaau0137, 2018. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.aau0137

[37] A. Bienert, L. Georgi, M. Kunz, H.-G. Maas, and G. Von Oheimb, "Comparison and combination of mobile and terrestrial laser scanning for natural forest inventories," *Forests*, vol. 9, no. 7, 2018. [Online]. Available: https://www.mdpi.com/1999-4907/9/7/395

[38] M. Dassot, T. Constant, and M. Fournier, "The use of terrestrial lidar technology in forest science: application fields, benefits and challenges," *Annals of Forest Science*, vol. 68, no. 5, pp. 959–974, 2011. [Online]. Available: https://doi.org/10.1007/s13595-011-0102-2

[39] C. Hopkinson, L. Chasmer, C. Young-Pow, and P. Treitz, "Assessing forest metrics with a ground-based scanning lidar." *Canadian Journal of Forest Research*, vol. 34, pp. 573–583, 03 2004.

[40] Forestry Tools, "Forestry tools for forest inventory," https://www.forestrytools.com.au/collections/forest-inventory, 2023, accessed: 2023-05-06.

[41] M. Bryson, F. Wang, and J. Allworth, "Using synthetic tree data in deep learning-based tree segmentation using lidar point clouds," *Remote Sensing*, vol. 15, no. 9, 2023. [Online]. Available: https://www.mdpi.com/2072-4292/15/9/2380

[42] M. D. Behera and P. S. Roy, "Lidar remote sensing for forestry applications: The indian context," *Current Science*, vol. 83, no. 11, pp. 1320–1328, 2002. [Online]. Available: http://www.jstor.org/stable/24106956

[43] A. Jennings, "Commencement of harvesting in the green triangle," https://www.sfmes.com.au/news/2018/10/29/commencement-of-harvesting-in-the-green-triangle, 2018, accessed: 2023-05-06.

[44] Australian Government Department of Agriculture, Fisheries and Forestry ABARES, "Forest definition in australia's national forest inventory," https://www.agriculture.gov.au/abares/forestsaustralia/profiles/australias-forests-2016, 2016, accessed: 2023-05-06.

[45] N. Younan, H. Lee, D. Evans, and N. Eggleston, "Extracting digital terrain models in forestry using lidar data," in *IGARSS 2001. Scanning the Present and Resolving the Future. Proceedings. IEEE 2001 International Geoscience and Remote Sensing Symposium (Cat. No.01CH37217)*, vol. 5, 2001, pp. 2070–2072 vol.5.

[46] B. Guindon, D. Goodenough, and P. Teillet, "The role of digital terrain models in the remote sensing of forests," *Canadian Journal of Remote Sensing*, vol. 8, no. 1, pp. 4–16, 1982. [Online]. Available: https://doi.org/10.1080/07038992.1982.10855019

[47] J. Hyyppä, U. Pyysalo, H. Hyyppä, and A. Samberg, "Elevation accuracy of laser scanning-derived digital terrain and target models in forest environment," in *Proceedings of EARSeL-SIG-Workshop LIDAR*, 2000, pp. 16–17.

[48] M. Mohan, R. V. Leite, E. N. Broadbent, W. S. W. M. Jaafar, S. Srinivasan, S. Bajaj, A. P. D. Corte, C. H. do Amaral, G. Gopan, S. N. M. Saad, A. M. M. Kamarulzaman, G. A. Prata, E. Llewelyn, D. J. Johnson, W. Doaemo, S. Bohlman, A. M. A. Zambrano, and A. Cardil, "Individual tree detection using uav-lidar and uav-sfm data: A tutorial for beginners," *Open Geosciences*, vol. 13, no. 1, pp. 1028–1039, 2021. [Online]. Available: https://doi.org/10.1515/geo-2020-0290

[49] U. Pyysalo and H. Hyyppä, "Geometric shape of the tree extracted from laser scanning data," in *International Society for Photogrammetry and Remote Sensing Á/ISPRS Commission III Symposium (PCV'02), Graz, Austria, 9Á/13 September*, 2002.

[50] E. Næsset, "Predicting forest stand characteristics with airborne scanning laser using a practical two-stage procedure and field data," *Remote Sensing of Environment*, vol. 80, no. 1, pp. 88–99, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425701002905

[51] E. Næsset, "Predicting forest stand characteristics with airborne scanning laser using a practical two-stage procedure and field data," *Remote sensing of environment*, vol. 80, no. 1, pp. 88–99, 2002.

[52] Y. Qin, A. Ferraz, C. Mallet, and C. Iovan, "Individual tree segmentation over large areas using airborne LiDAR point cloud and very high resolution optical imagery," in *Multitemp 2015*. IEEE, 2015, pp. 800–803. [Online]. Available: https://www.documentation.ird.fr/hor/fdi:010067245

[53] J. Holmgren and E. Lindberg, "Tree crown segmentation based on a geometric tree crown model for prediction of forest variables," *Canadian Journal of Remote Sensing*, vol. 39, pp. S86–S98, 06 2014.

[54] E. Lindberg, L. Eysn, M. Hollaus, J. Holmgren, and N. Pfeifer, "Delineation of tree crowns and tree species classification from full-waveform airborne laser scanning data using 3-d ellipsoidal clustering," *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, vol. 7, pp. 3174–3181, 07 2014.

[55] F. Morsdorf, E. Meier, B. Kötz, K. I. Itten, M. Dobbertin, and B. Allgöwer, "Lidar-based geometric reconstruction of boreal type forest stands at single tree level for forest and wildland fire management," *Remote Sensing of Environment*, vol. 92, no. 3, pp. 353–362, 2004, forest Fire Prevention and Assessment. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425704001464

[56] K. Olofsson and J. Holmgren, "Single tree stem profile detection using terrestrial laser scanner data, flatness saliency features and curvature properties," *Forests*, vol. 7, no. 9, 2016. [Online]. Available: https://www.mdpi.com/1999-4907/7/9/207

[57] M. T. *, N. Pfeifer, D. Winterhalder, and B. G. H. Gorte, "Three-dimensional reconstruction of stems for assessment of taper, sweep and lean based on laser scanning of standing trees," *Scandinavian Journal of Forest Research*, vol. 19, no. 6, pp. 571–581, 2004. [Online]. Available: https://doi.org/10.1080/02827580410019562

[58] J. G. Henning and P. J. Radtke, "Detailed stem measurements of standing trees from ground-based scanning lidar," *Forest Science*, vol. 52, no. 1, pp. 67–80, 2006.

[59] J.-F. Lalonde, N. Vandapel, D. F. Huber, and M. Hebert, "Natural terrain classification using three-dimensional ladar data for ground robot mobility," *Journal of field robotics*, vol. 23, no. 10, pp. 839–861, 2006.

[60] P. Raumonen, M. Kaasalainen, M. Åkerblom, S. Kaasalainen, H. Kaartinen, M. Vastaranta, M. Holopainen, M. Disney, and P. Lewis, "Fast automatic precision tree models from terrestrial laser scanner data," *Remote Sensing*, vol. 5, no. 2, pp. 491–520, 2013.

[61] G. Brolly and G. Kiraly, "Algorithms for stem mapping by means of terrestrial laser scanning," *Acta Silvatica et Lignaria Hungarica*, vol. 5, pp. 119–130, 2009.

[62] P. Forsman and A. Halme, "3-d mapping of natural environments with trees by means of mobile perception," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 482–490, 2005.

[63] T. Aschoff, M. Thies, and H. Spiecker, "Describing forest stands using terrestrial laser-scanning," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 35, no. 5, pp. 237–241, 2004.

[64] N. Pfeifer and D. Winterhalder, "Modelling of tree cross sections from terrestrial laser scanning data with free-form curves," *International Archives of Photogrammetry, remote sensing and spatial information sciences*, vol. 36, no. Part 8, p. W2, 2004.

[65] S. J. Russell, P. Norvig, M.-W. Chang, J. Devlin, A. Dragan, D. Forsyth, I. Goodfellow, J. Malik, V. Mansinghka, J. Pearl, and et al., *Artificial Intelligence: A modern approach*, 4th ed. Pearson, 2020.

[66] P. H. Winston, "Learning structural descriptions from examples," Massachusetts Institute of Technology, USA, Tech. Rep., 1970.

[67] T. G. Dietterich and R. S. Michalski, *A Comparative Review of Selected Methods for Learning from Examples*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1983, pp. 41–81. [Online]. Available: https://doi.org/10.1007/978-3-662-12405-5_3

[68] Y. Kodratoff and R. Lemerle-Loisel, "Learning complex structural descriptions from examples," *Computer Vision, Graphics, and Image Processing*, vol. 27, no. 3, pp. 266–290, 1984. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0734189X8490032X

[69] O. G. Selfridge and U. Neisser, "Pattern recognition by machine," *Scientific American*, vol. 203, no. 2, pp. 60–69, 1960.

[70] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[71] M. Kubat, *An Introduction To Machine Learning*, 3rd ed. Springer, 2021.

[72] T. O. Ayodele, "Types of machine learning algorithms," *New advances in machine learning*, vol. 3, pp. 19–48, 2010.

[73] M. W. Berry, A. Mohamed, and Y. B. Wah, *Supervised and unsupervised learning for Data Science*. Springer, 2020.

[74] B. Gope, S. Pande, N. Karale, S. Dharmale, and P. Umekar, "Handwritten digits identification using mnist database via machine learning models," *IOP Conference Series: Materials Science and Engineering*, vol. 1022, no. 1, p. 012108, jan 2021. [Online]. Available: https://dx.doi.org/10.1088/1757-899X/1022/1/012108

[75] V. Christina, S. Karpagavalli, and G. Suganya, "Email spam filtering using supervised machine learning techniques," *International Journal on Computer Science and Engineering (IJCSE)*, vol. 2, no. 09, pp. 3126–3129, 2010.

[76] M. Holmstrom, D. Liu, and C. Vo, "Machine learning applied to weather forecasting," *Meteorol. Appl*, vol. 10, pp. 1–5, 2016.

[77] S. Dhankhad, E. Mohammed, and B. Far, "Supervised machine learning algorithms for credit card fraudulent transaction detection: A comparative study," in *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, 2018, pp. 122–125.

[78] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[79] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, May 2017. [Online]. Available: https://doi.org/10.1145/3065386

[80] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[81] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Rotational projection statistics for 3d local surface description and object recognition," *International Journal of Computer Vision*, vol. 105, pp. 63–86, 04 2013.

[82] Y. Xu, X. Tong, and U. Stilla, "Voxel-based representation of 3d point clouds: Methods, applications, and its potential use in the construction industry," *Automation in Construction*, vol. 126, p. 103675, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926580521001266

[83] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," 2016.

[84] E. Čech, *Point Sets*. Academic Press, 1969. [Online]. Available: https://books.google.com.au/books?id=pFUVAQAAIAAJ

[85] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, 2018. [Online]. Available: https://doi.org/10.1007/s13244-018-0639-9

[86] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. [Online]. Available: https://doi.org/10.1023/B:VISI.0000029664.99615.94

[87] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 108–11 117.

[88] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2d-3d-semantic data for indoor scene understanding," *CoRR*, vol. abs/1702.01105, 2017. [Online]. Available: http://arxiv.org/abs/1702.01105

[89] S. Kim, T. Hinckley, and D. Briggs, "Classifying individual tree genera using stepwise cluster analysis based on height and intensity metrics derived from airborne laser scanner data," *Remote Sensing of Environment*, vol. 115, no. 12, pp. 3329–3342, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425711002690

[90] H. Guan, Y. Yu, Z. Ji, J. Li, and Q. Zhang, "Deep learning-based tree classification using mobile lidar data," *Remote Sensing Letters*, vol. 6, no. 11, pp. 864–873, 2015. [Online]. Available: https://doi.org/10.1080/2150704X.2015.1088668

[91] Q. Li, W. Zhou, and C. Li, "Use of airborne lidar to estimate forest stand characteristics," *IOP Conference Series: Earth and Environmental Science*, vol. 17, no. 1, p. 012252, mar 2014. [Online]. Available: https://dx.doi.org/10.1088/1755-1315/17/1/012252

[92] J. Jung and M. M. Crawford, "Extraction of features from lidar waveform data for characterizing forest structure," *IEEE Geoscience and Remote Sensing Letters*, vol. 9, no. 3, pp. 492–496, 2012.

[93] H. Hamraz, N. B. Jacobs, M. A. Contreras, and C. H. Clark, "Deep learning for conifer/deciduous classification of airborne lidar 3d point clouds representing individual trees," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 158, pp. 219–230, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271619302485

[94] S. Tao, F. Wu, Q. Guo, Y. Wang, W. Li, B. Xue, X. Hu, P. Li, D. Tian, C. Li, and et al., "Segmenting tree crowns from terrestrial and mobile lidar data by exploring ecological theories," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 110, p. 66–76, 2015. [Online]. Available: https://dx.doi.org/10.1016/j.isprsjprs.2015.10.007

Appendices

A Confusion Matrices

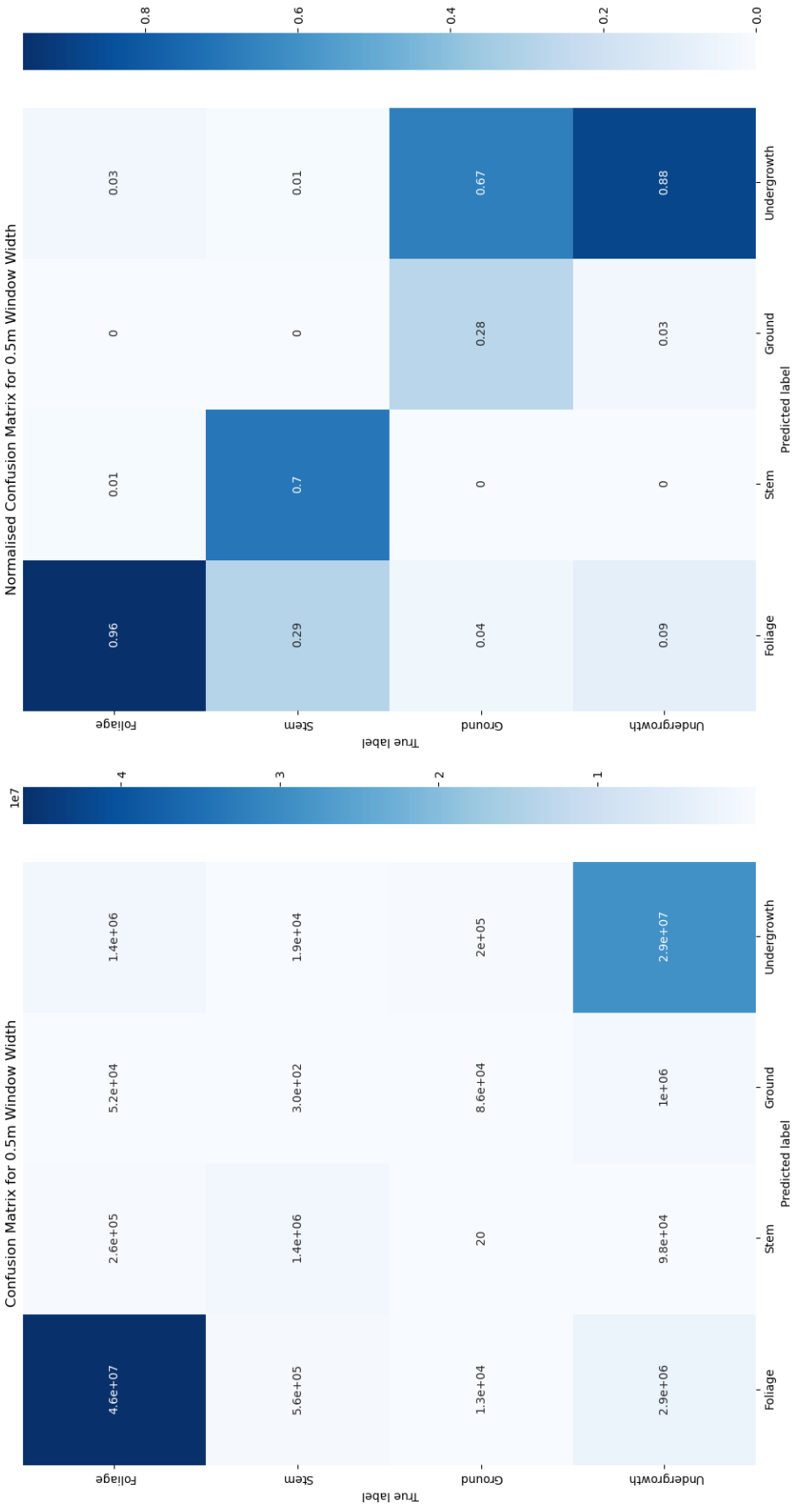


Figure 1: Confusion matrix on the testing dataset for a window width of 0.5m

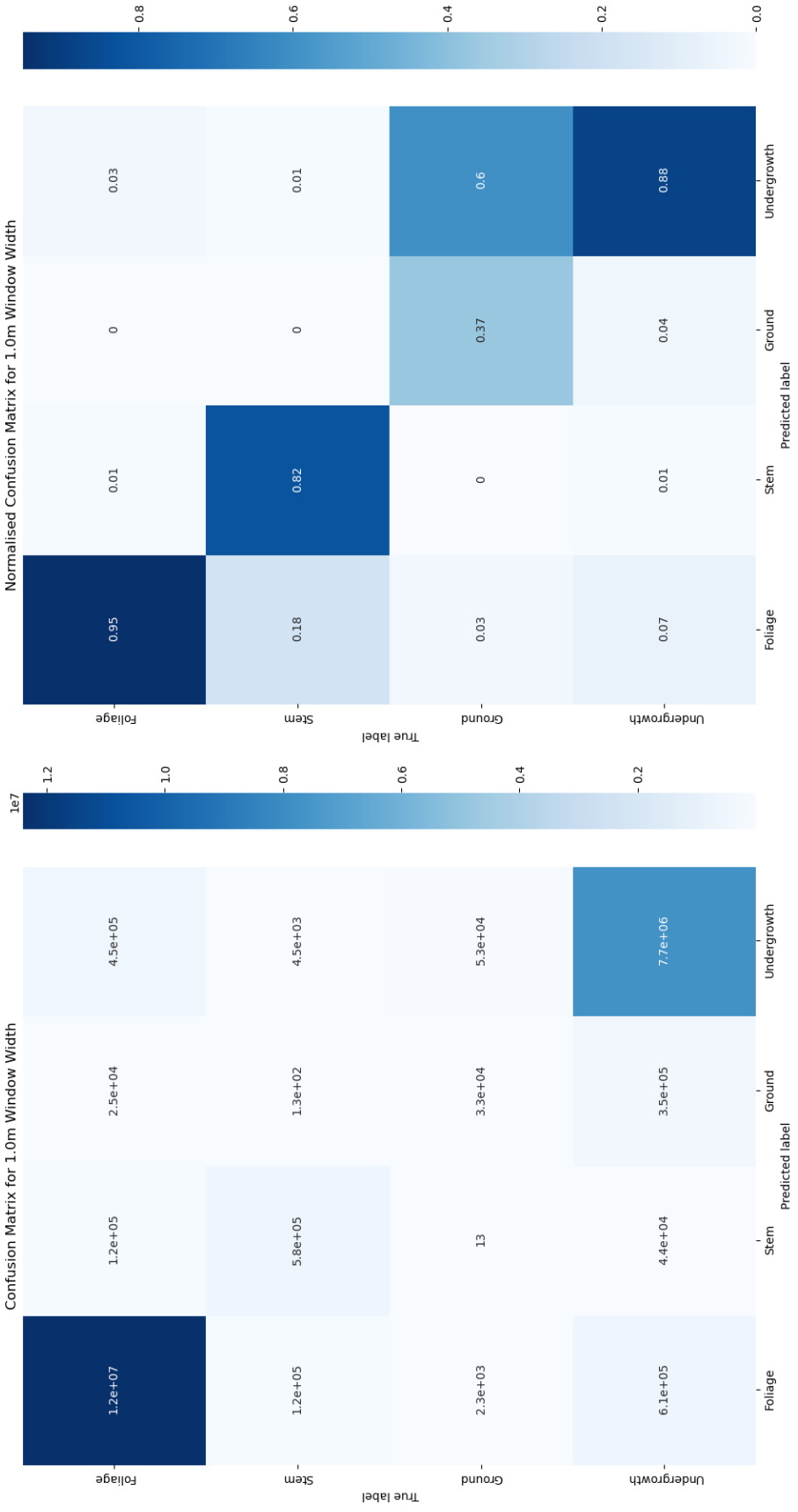


Figure 2: Confusion matrix on the testing dataset for a window width of 1.0m

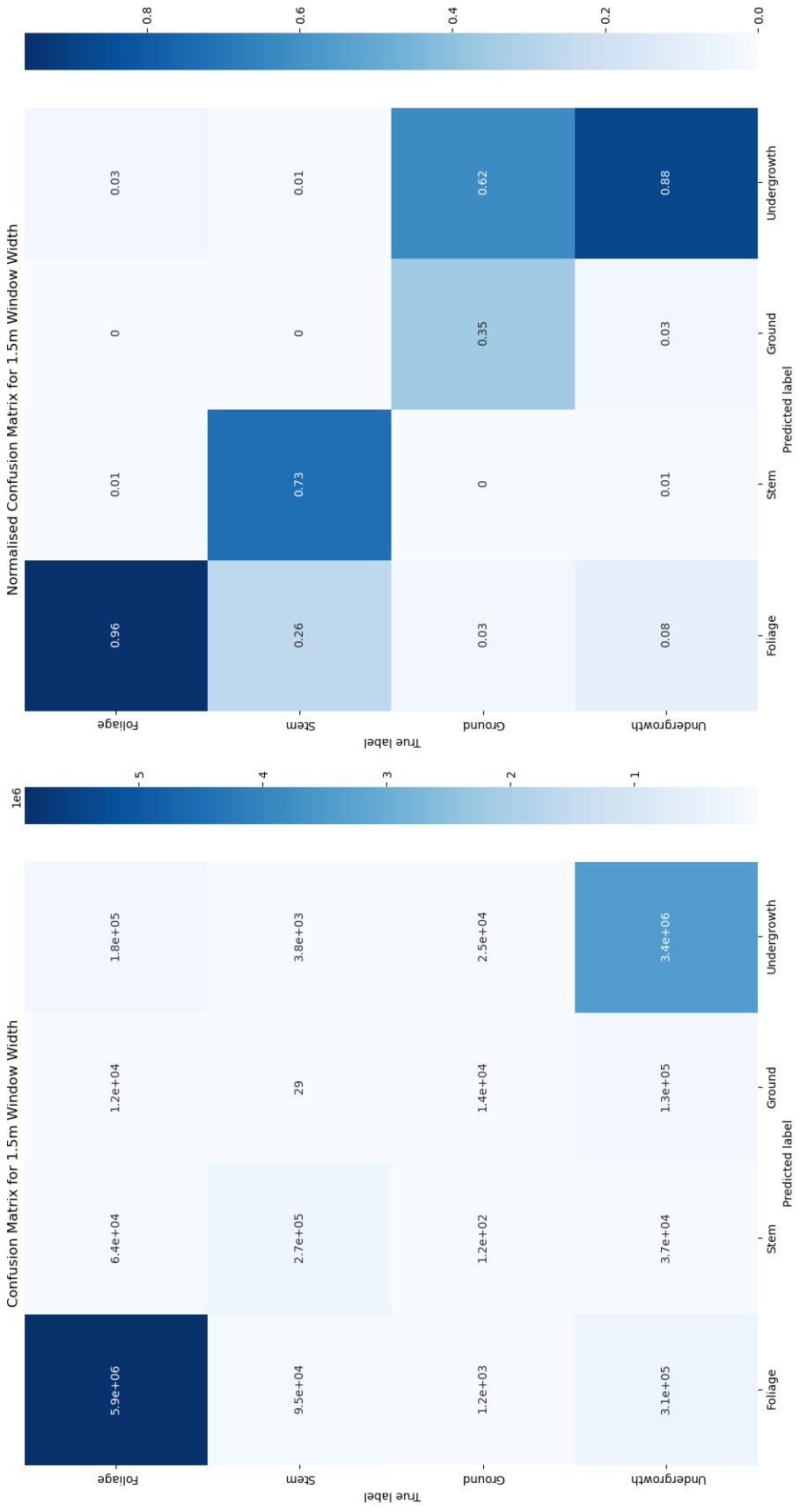


Figure 3: Confusion matrix on the testing dataset for a window width of 1.5m

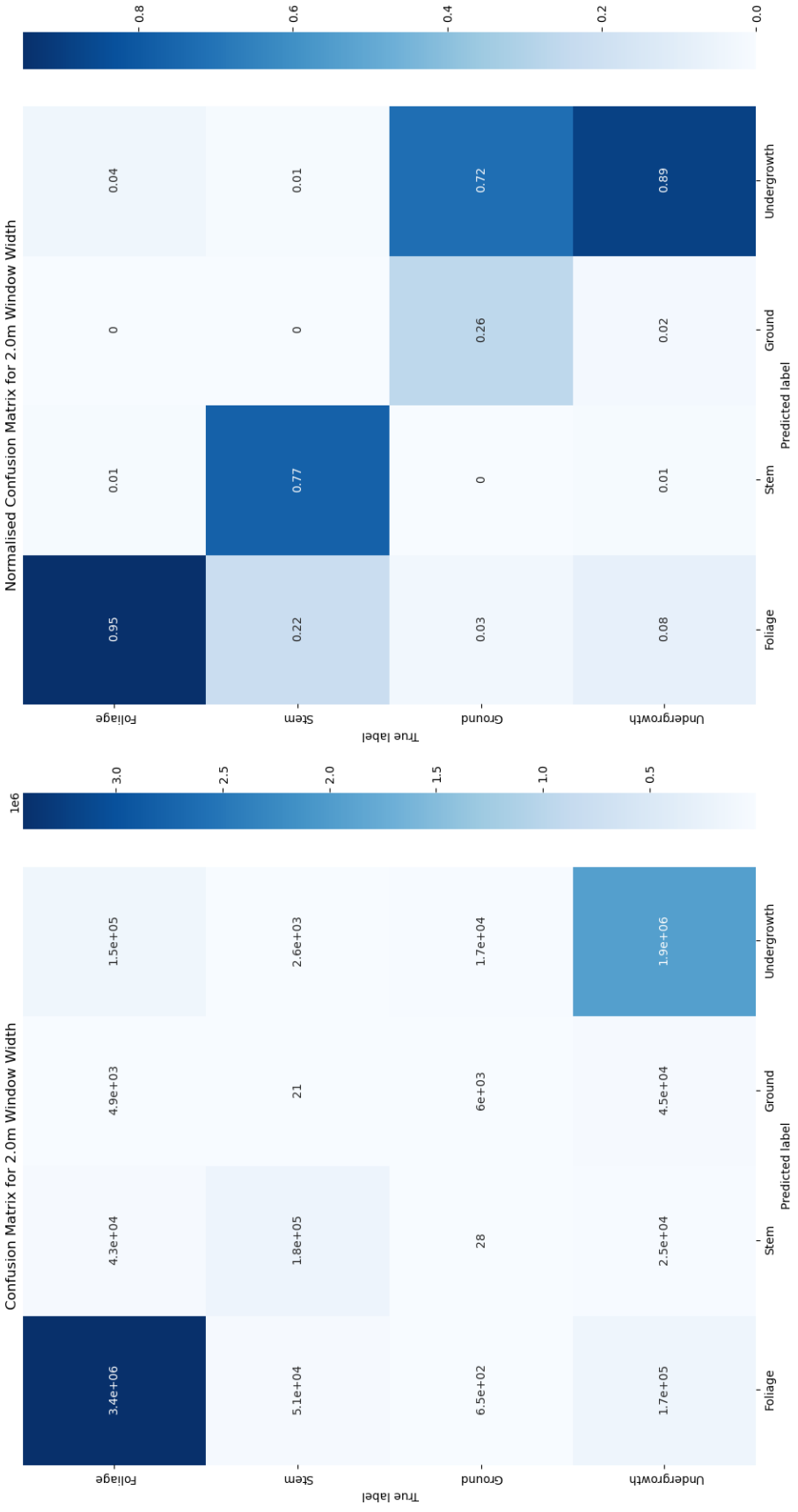


Figure 4: Confusion matrix on the testing dataset for a window width of 2.0m

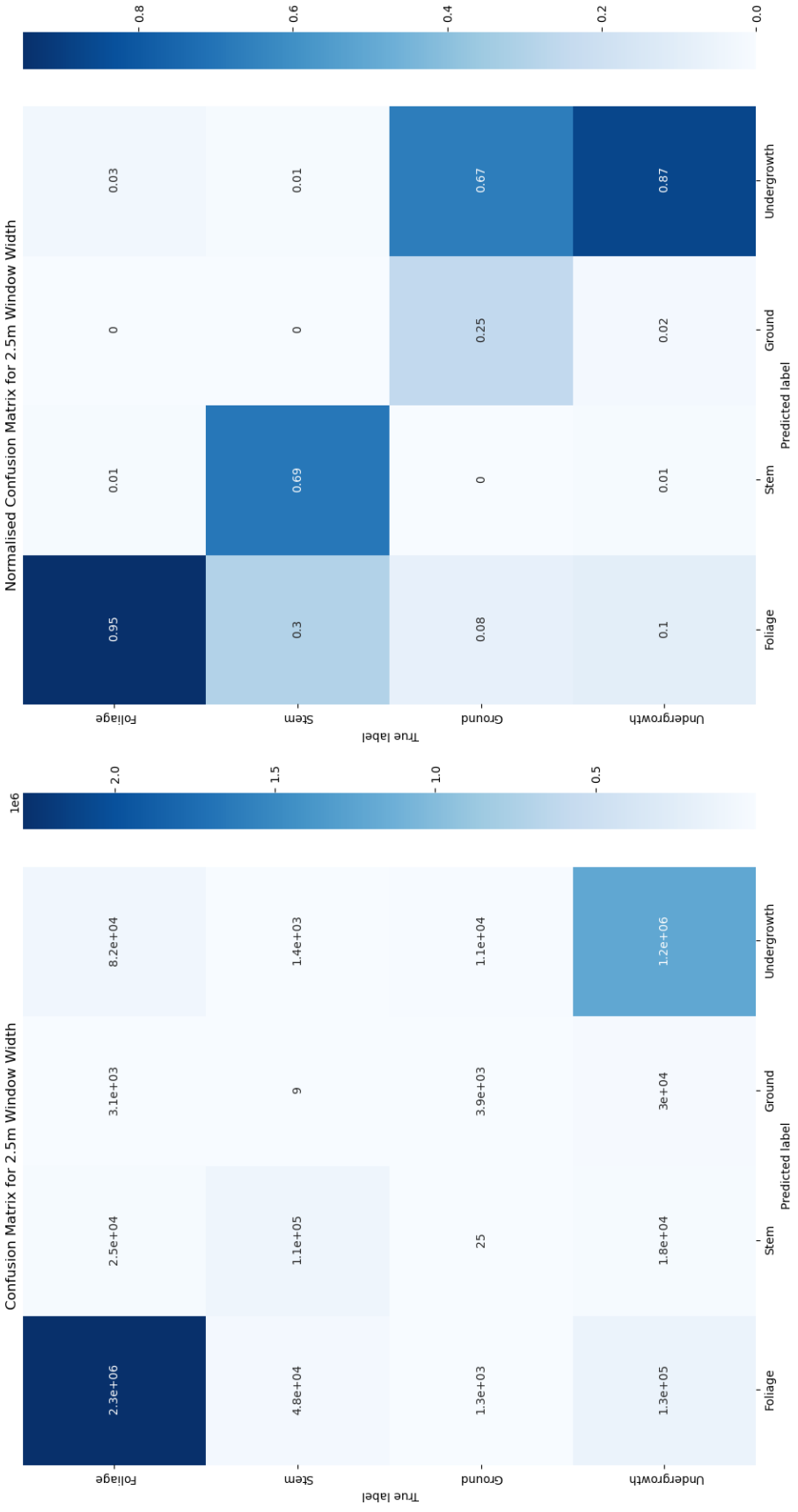


Figure 5: Confusion matrix on the testing dataset for a window width of 2.5m

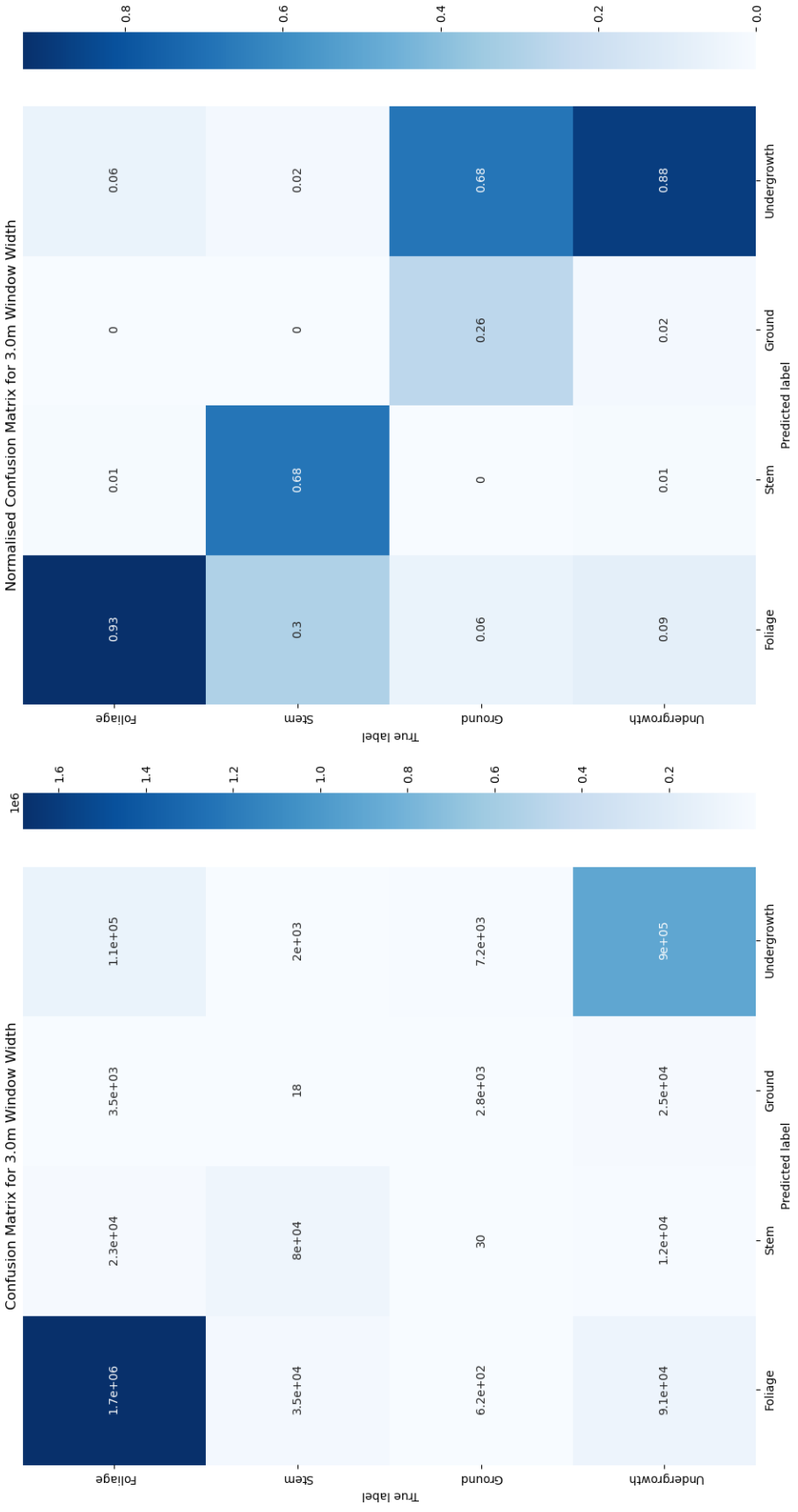


Figure 6: Confusion matrix on the testing dataset for a window width of 3.0m

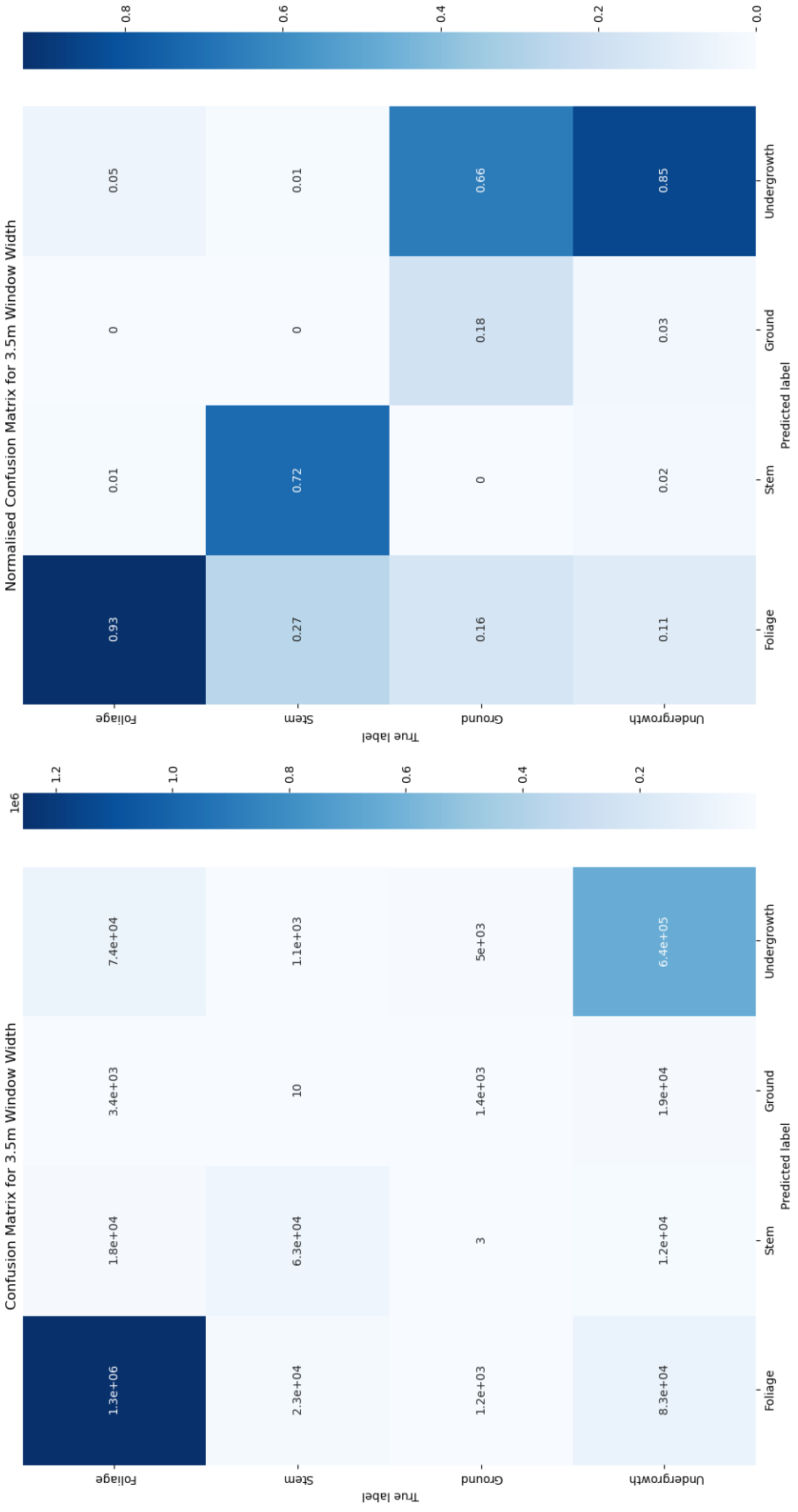


Figure 7: Confusion matrix on the testing dataset for a window width of 3.5m

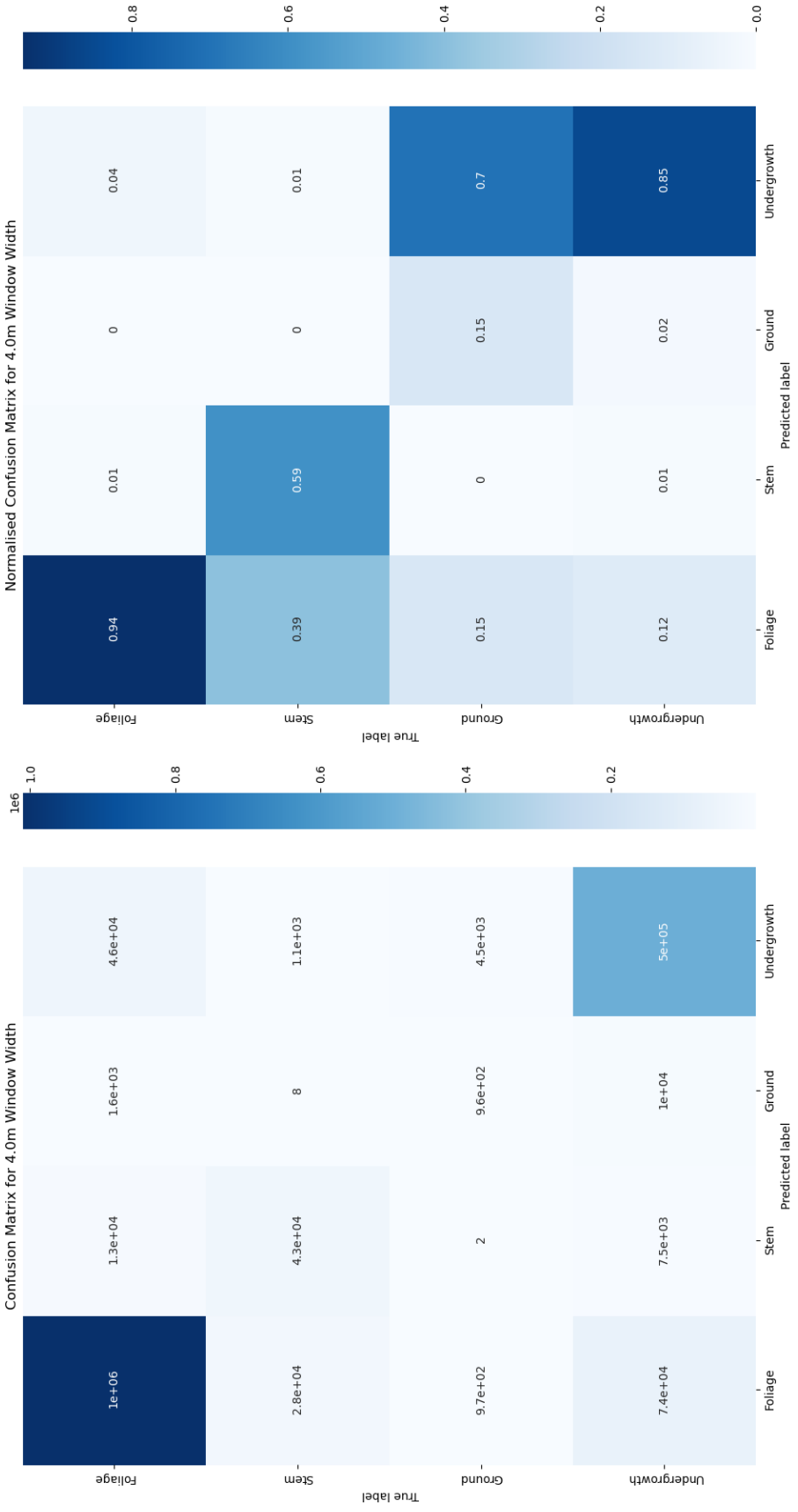


Figure 8: Confusion matrix on the testing dataset for a window width of 4.0m

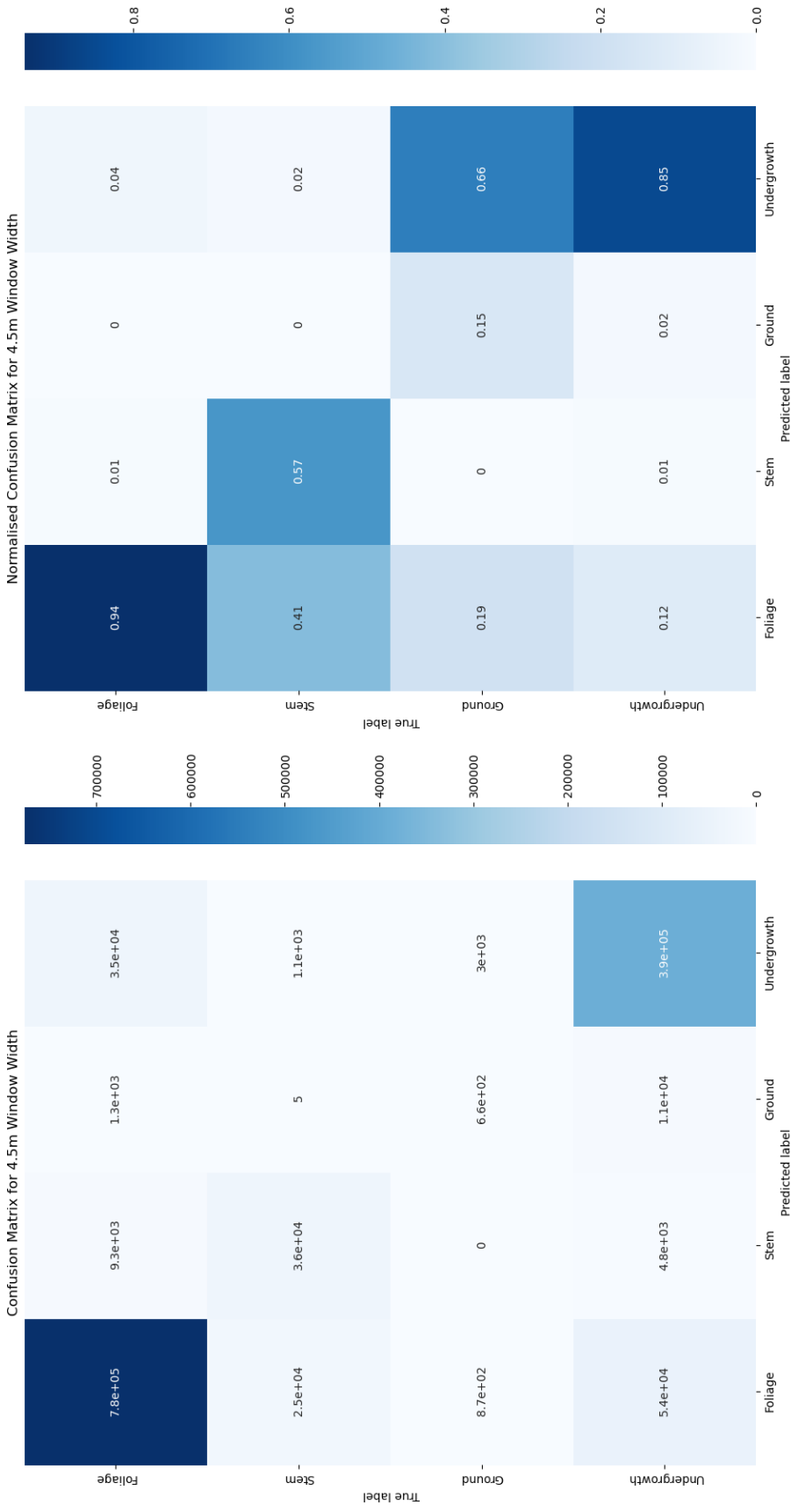


Figure 9: Confusion matrix on the testing dataset for a window width of 4.5m