Tree segmentation from AHN4

Using a non-end-to-end neural network and random forest

by

G. Blokker

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Wednesday August 24, 2022 at 13:00.

Student number:	4468589	
Project duration:	September 1, 2021 – August 1, 2022	
Thesis committee:	Dr. R. C. Lindenbergh,	TU Delft, supervisor
	Dr. S. L. M. Lhermitte,	TU Delft
	Dr. A. A. Nurunnabi,	University of Luxembourg
	Ir. F. Dahle,	Tu Delft

This thesis is confidential and cannot be made public until August 31, 2022.

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Abstract

In this thesis effective ways to segment trees from AHN4 will be studied. With the modern day focus on biodiversity, maintaining it or increasing it, a good knowledge of vegetation is especially vital. AHN4 is the fourth edition of a national point cloud covering the entire Netherlands. It is distributed with basic classification, but trees are part of a larger class 'other' which also contains other non-terrain objects like street furniture and cars. The methods considered here are non-end-to-end which means that first features need to be generated which will be consecutively fed to a classification algorithm. Features are created by applying Principal Component Analysis (PCA) with a suitable point neighborhood. A neighbourhood of 20 nearest neighbours turned out to work most effectively. A Neural Network and Random Forest will be tested as classification algorithms. The training dataset consisted of a combination of a tree database from the municipality of Delft and AHN4. This training data is imperfect since it contained both false positives and false negatives due to how it was created. For both algorithms a training dataset of 400 000 points was used, to prevent overfitting, with a 50/50 split between tree and non-tree points. Both algorithms were validated against two datasets, Validation Dataset one is a small generic dataset and Validation Dataset two is a dataset consisting of problem scenarios such as cars and construction sites. The Neural Network scored an accuracy of 92.8% and the Random Forest 94.4% against Validation Dataset one. Against Validation Dataset two the Neural Network scored an accuracy of 66.9% and Random Forest 72.4%. Both algorithms had trouble with correctly classifying cars and construction sites as non trees. In the end, the Random Forest algorithm had a higher accuracy than the Neural Network, however this might be the result of the imperfect training data.

Preface

This thesis was written as a completion of the master course Geoscience and Remote Sensing at the Civil Engineering faculty of Technical University of Delft. The focus of this thesis is upon using machine learning to classify objects out of 3D multi-object datasets. I started the thesis in September of 2021 and the final presentation takes place on the 24th of August in 2022.

This subject was chosen in cooperation with Roderik Lindenbergh because of my interest into getting a better grasp on Machine Learning. Especially Deep Learning which is a field of study which I believe will become even more relevant in the future.

Special thanks goes to my supervisor Roderik for the advice and supervision that helped me finish my thesis. Emma Blanken was also a huge support with proofreading and for helping with QGIS problemshooting. And finally thanks goes out to Deannie Yap for correcting and checking the grammar and spelling in this thesis.

Contents

1 Introduction 1

2 Background 3
2.1 AHN . . . 3
2.2 Algorithm Theory . . . 4
2.2.1 Deep Learning . . . 5
2.2.2 Principal Component Analysis . . . 6
2.2.3 Random Forest . . . 7
2.2.4 DBSCAN . . . 7
2.3 Previous work on tree segmentation . . . 8
2.4 Deep Learning for Point Clouds . . . 8
2.5 Training Strategy . . . 9

3 Methodology 11
3.1 Preprocessing . . . 11
3.1.1 Validation Dataset 1 . . . 11
3.1.2 Validation dataset 2 . . . 12
3.1.3 Training Data . . . 13
3.2 Non-end-to-end Method . . . 14
3.2.1 Feature Engineering . . . 15
3.3 Neural Network . . . 17
3.3.1 Training strategy . . . 18
3.3.2 Neural Network Parameters . . . 19
3.4 Random Forest . . . 20
3.5 DBSCAN . . . 21

4 Results 25
4.1 Results Neighbourhood Selection . . . 25
4.2 Algorithm Results . . . 27
4.2.1 Quantitative Results . . . 28
4.2.2 Scenarios: Absolute Classification . . . 29
4.2.3 Scenarios: taking a look at the probabilities. . . . 33
4.3 Other results . . . 37

5 Discussion 39
5.1 Training Data . . . 39
5.2 Feature Engineering . . . 39
5.3 Neural Network . . . 40
5.4 Random Forest . . . 41
5.5 Omissions and Commissions . . . 41
5.6 State of the Art . . . 41
5.7 Tree inventories . . . 42

6 Conclusion and Recommendations 43
6.1 Research Question and Subquestions . . . 43
6.2 Recommendations . . . 44

1

Introduction

The goal of this research will be to classify trees out of the 'other' class within AHN4. AHN4 already has a basic classification present in its data. However it still has a significant 'other' class that lacks proper classification. AHN4 is a height database of the Netherlands in the form of a point-cloud, which is data represented in the form of points where each point has its own specific properties. The area of interest will be Delft, Delft is a city of around 100 000 citizens in the province of South-Holland in the Netherlands. Especially with the modern day focus on biodiversity, be it maintaining it or increasing it, a good knowledge of vegetation is vital (K. Wang, T. Wang, and Liu 2018). An object will be called a tree if it has an identifiable stem and crown and a height of at least 2 metres. A first step for getting a clear picture of the biodiversity in a city is to be able to get a full database out of AHN4 of all the trees within Delft. From this database some main properties of the trees can be established such as height and volume. This research will solely focus on setting the first step in obtaining these trees out of the AHN data, for further research tree species can for example possibly be determined.

In this research a non-end-to-end method will be used which means that there will be a processing step before inputting the data into the classification algorithm instead of just straight inputting the points into the classification algorithm which is an end-to-end method (Nurunnabi et al. 2021). First there will be features made for each point to help the algorithm with classification before they get put into the classification algorithm. To classify these points, two classification algorithms are used: a deep learning network, feed forward neural network in this case, and a random forest. Both of these classify points based upon the properties provided with a point. Deep learning makes use of a structure containing a lot of nodes which each have a simple mathematical function and which in the end comes together to give a classification. Random forest contains multiple trees where each of these trees make a classification and in the end the majority is the final classification (Breiman 2001). Each of these algorithms learns how to classify points by looking at an already classified dataset. How the algorithm learns from this dataset is called a training strategy.

One of the main difficulties this thesis will face is making a correct classification using imperfect training data. Perfect training data is not available, as that would mean that a correct segmentation of trees from AHN4 already exists, which would render this thesis unnecessary. Another problem this thesis will face stems from the same root as the imperfect training data, namely how to quantify the accuracy of the method. There is no perfect validation dataset present for the same reason as there is no perfect training dataset present. This will make it challenging to find a way to quantify the quality of the method.

The main research question is: **How to segment trees from AHN4 data effectively?**. To segment means that trees are correctly separated from the rest of the data. With effectiveness what is meant is that the method will both segment the trees correctly as well as efficiently. In this manner the method may be run on a desktop instead of a supercomputer. The following 5 sub-questions will also be asked:

- How can the performance of the methods be measured?
- What effect do the features have on the algorithm?
- What is the effect of the training strategy on the outcome of the different algorithms?

- With what kind of cases do the algorithms struggle?
- What is the difference between using a feed forward Neural Network and a Random Forest algorithm in a non-end-to-end method?

2

Background

In this chapter relevant literature to the research and theory which was used in this thesis will be discussed and explained. In section 2.1 basic information of the dataset used in this thesis, AHN4, will be given. In section 2.2 the theory behind the algorithms used in this thesis will be explained. Sections 2.3, 2.4 and 2.5 will then provide some background based on previous literature relevant for this thesis.

2.1. AHN

AHN stands for 'Actueel Hoogtebestand Nederland' which is a point cloud which can be used for classification of the Dutch landscape and to obtain the height of the Netherlands. AHN is obtained through airborne LiDAR. LiDAR itself stands for Light Detection And Ranging, and is an active sensor which works from the principle that a laser gets emitted and reflected back to the sensor. Airborne LIDAR is such as the name already suggest a LiDAR sensor installed upon a plane or drone from which the data is obtained. From this LiDAR sensor one can obtain properties such as location, intensity and possibly more properties dependent upon the instrument.

With the recent arrival of AHN4 , it is important to classify this data, as this classification is vital for further analysis. AHN4 is a point-cloud obtained from aerial LiDAR, during a three year period between 2020 to 2022 (*AHN* 2021). So far, AHN4 classifies ground, buildings and water, which corresponds to the classification in AHN3. However it does have a significant class 'Other' which consists of multiple objects that do not fall in the previous classification categories such as street furniture, cars, construction sites, trees and other vegetation, see figure 2.1.

Figure 2.1: Some examples of objects present in the other classification of AHN4. On the top right a construction site is shown, on bottom right street furniture, on the top left cars and bottom left trees.

One of the big differences between AHN4 and AHN3 is that AHN4 includes the following extra fields per point:

- Amplitude, is the amplitude of the echo of the signal reaching the laser scanner. (GmbH 2017)
- Reflectance, "The relative reflectance provided is a ratio of the actual amplitude of that target to the amplitude of a white flat target at the same range, orientated orthonormal to the beam axis, and with a size in excess of the laser footprint." (GmbH 2017)
- Deviation, "The pulse shape deviation can be interpreted as the comparison of the area below the shape curve and is one of the additional attributes to each point of the point cloud." (GmbH 2017)

These fields obviously give extra input for classification which can possibly make the AHN4 data easier to classify than the AHN3 data. Another big difference between the AHN3 and AHN4 data is the point density, AHN4 has roughly 2.5 times the amount of points that AHN3 has in the study area (Delft). The point density of AHN4 lies around 10 to 14 points per square metre (*AHN* 2021). The other fields which are present in AHN4 are as follows:

- Intensity
- Number of Returns
- Scan Angle

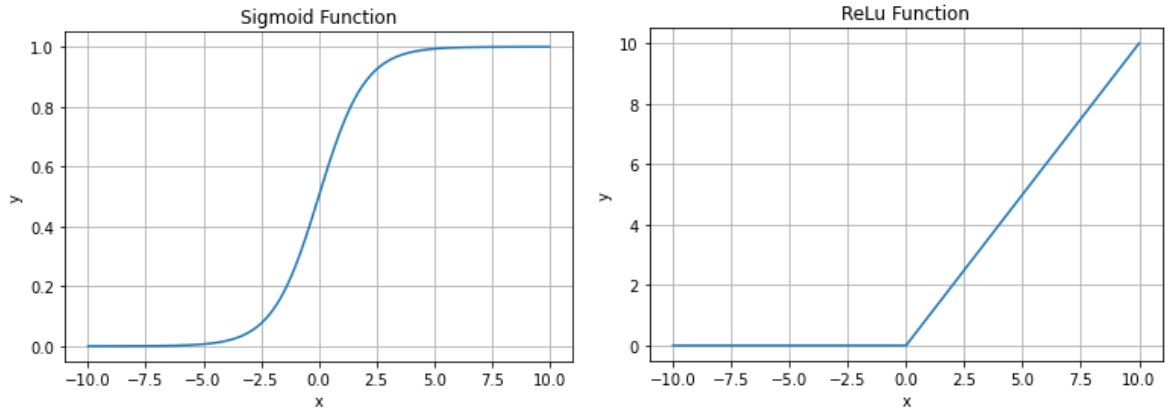
For this thesis the AHN4 data will be extracted from *https://geotiles.nl/* (Natijne van 2021). This site has subdivided AHN4 into 1km by 1.25km squares which makes it easier to process and obtain. All of these squares have colour added to them from the closest aerial image in time since AHN4 does not scan colour. Keep in mind then that the colours in figures from AHN4 do not always exactly match with the points. The colour is also not used in this research for the classification of the points for this reason.

2.2. Algorithm Theory

In this section the basic theory of the algorithms used in this thesis are discussed and explained.

2.2.1. Deep Learning

Neural Networks (NN) have become a popular tool for analysis. So first off what is a neural network? A neural network's purpose is to classify data based on training data which is used to train its network. A Neural Network consists of an input layer, some hidden layers and an output layer. The input layer consists of the raw data which needs to be classified, this then gets passed to the first of the hidden layers. Each of these hidden layers consist of a number of neurons, in these neurons a non-linear operation is performed with the presented value this can be as simple as a sigmoid function, see figure 2.2a, or a Rectified Linear Unit (ReLU) function, see figure 2.2b.



(a) A sigmoid function $y = 1/(1+e^{-x})$

(b) The ReLU function.

Figure 2.2: Two examples of possible activation functions for a Neural Network.

The data then gets passed to the next neurons which consist of the same or different non-linear operation until it eventually reaches the output layer. Between each of these passes a weight is assigned to the data, when training the network this weight is also the variable that gets adjusted. This is also called a feed forward network, see figure 2.3. The way a Neural Network learns is by the use of training data. Training data is already classified data unto which a Neural Network adjusts its weights based on the predicted classification and the actual classification present in the training data, the difference between the two is the loss. The loss is an error, which depends upon the optimizer, which represents the difference between the expected outcome of the Neural Network and the actual outcome. The method with which the Neural Network calculates this loss and adjusts the parameters is called an optimizer. The optimizer works by trying to minimize the loss in the Neural Network.

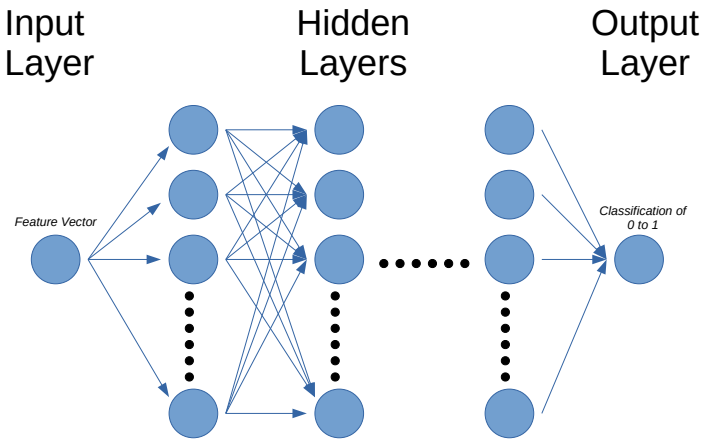


Figure 2.3: An example of a Neural Network representative for the one used in this thesis, the blue dots represent Neurons.

A form of Neural Network that is often used for classification is a Convolutional Neural Network (CNN). For a convolutional neural network a convolution operation is performed. First a kernel is passed over the data after which a max or average pooling is applied. The max value feature is extracted or the average of all the features in the kernel. The pooling reduces the dimensionality while the convolution extracts the dominant features such as edges. The output of this then gets passed to a feed forward network which eventually gives the output. In this thesis it is opted to not use a CNN but rather a simple feed forward network, since this corresponds better to the eventual method used in this thesis.

2.2.2. Principal Component Analysis

Principal Component Analysis (PCA) is a technique which reduces the amount of dimensions (dimension reduction).For example if there are 10 features for a certain point, PCA can reduce it to 3 features. PCA does this by looking at a linear combination of the 10 features and finding the combination with the highest variability. A second component can then be found by removing the first component and repeating the process resulting in a max of 10 components for this example, where each component has a reduced variability compared to the previous component (Bro and Smilde 2014). The features with the highest variability are then selected to represent the point, in this manner the number of features are then reduced.

PCA makes these features by looking at the covariance matrix. The covariance matrix consists of the variance for each feature, present on the diagonal of the matrix, and the covariance between the features, present in the other fields of the matrix. The covariance is the joint variance of two random features with each other. Mathematically the covariance (denoted by cov) is defined as the mean (denoted by E) of the product of the deviations of the two random features, so for example if there are the following two random features X and Y the mathematically formulated covariance would be as in equation 2.1. The variance is defined as the covariance of a feature with itself. To get the eigenvalues (represented by λ), which represent the variance of the features, the covariance matrix gets diagonalized. The values left on the diagonal are then the eigenvalues, this is represented by matrix D. To get D, equation 2.3, called the characteristic equation, needs to be solved in this equation I is an identity matrix, A is the covariance matrix and det is the determinant. Matrix P which consists of the corresponding eigenvectors, this is the vector that gives the direction of the eigenvalue also called the principal components in this case, can then be solved using 2.2. This decomposition of the covariance matrix is called Principal Component Analysis (D. C. Lay, S. R. Lay, and McDonald 2016).

$$cov(X,Y) = E[(X - E[X])(Y - E[Y])] \tag{2.1}$$

$$cov(X,Y) = PDP^T, P^T cov(X,Y)P = D \tag{2.2}$$

$$det(A - \lambda I) = 0 \tag{2.3}$$

Another way PCA can be used is for feature engineering. Then the three eigenvalues of the xyz components of the points within a certain neighbourhood get extracted contrary to the input of all features of a point which is the case for dimension reduction. The input in this case consists of only the xyz coordinates of the points within a certain neighbourhood while the input for reducing the variance consists of the features of a point. From the three eigenvalues extracted from the xyz coordinates of the neighbourhood of points features can be engineered. The essential part though is that this has to be done for every point for a certain neighbourhood. This way local features can be engineered specific to the point, as such is the neighbourhood vital for an accurate representation of the spatial attributes of the point (Nurunnabi et al. 2021). The features represented in table 2.1 (Weinmann et al. 2015) can be created using this approach, λ_1, λ_2 and λ_3 represent the three eigenvalues created using this approach.

PCA created features	Formula
Linearity	$\frac{\lambda_1-\lambda_2}{\lambda_1}$
Planarity	$\frac{\lambda_2-\lambda_3}{\lambda_1}$
Scattering	$\frac{\lambda_3}{\lambda_1}$
Omnivariance	$(\lambda_1 * \lambda_2 * \lambda_3)^{\frac{1}{3}}$
Anistropy	$\frac{\lambda_1-\lambda_3}{\lambda_1}$
Eigentropy	$\lambda_1 ln\lambda_1 + \lambda_2 ln\lambda_2 + \lambda_3 ln\lambda_3$
Sum of Eigenvalues	$\lambda_1 + \lambda_2 + \lambda_3$
Change of curvature	$\frac{\lambda_3}{\lambda_1+\lambda_2+\lambda_3}$
Curvature	λ_3

Table 2.1: The different features created using PCA

The method of PCA that is used in this paper is made by Halko, Martinsson, and Trop 2009. The reason behind this is that it is more efficient than normal PCA and better at dealing with large datasets. Quoting Halko, Martinsson, and Trop 2009: '*This method presents a modular framework for constructing randomized algorithms that compute partial matrix decompositions. These methods use random sampling to identify a subspace that captures most of the action of a matrix. The input matrix is then compressed—either explicitly or implicitly—to this subspace, and the reduced matrix is manipulated deterministically to obtain the desired low-rank factorization. In many cases, this approach beats its classical competitors in terms of accuracy, speed, and robustness. These claims are supported by extensive numerical experiments and a detailed error analysis.*' This was implemented in python by the 'sklearn' toolkit (*sklearn* n.d.).

2.2.3. Random Forest

Another machine learning algorithm that is used is Random Forest. Random Forest also needs training data to learn from, just as the Neural Network. Training data is classified data from which the random forest can learn and accordingly adjust its parameters. With the newly acquired parameters, it can classify new unclassified data. Random forest is an algorithm that consists of multiple tree predictors. A tree consists of multiple nodes (leaves) which each monitor error scores for a certain feature to make a decision (Breiman 2001). Each of these nodes makes a split to another layer of nodes, the recommended amount of splits for classification is the square root of the number of features (Biau and Scornet 2016), to prevent using all the features all the time so that favouritism for a certain feature does not take place. Using a multiple of trees decreases the error caused by the randomness of the decision making done by the tree to decide which feature to use for its leaf. The two most important variables which can be changed within the random forest are the depth of the individual trees (so the amount of splits) and the amount of trees used within the random forest, see figure 2.4 for an example of n amount of trees and a depth of 3. The amount of trees does not matter as long as it is large (Biau and Scornet 2016) what is large is however unclear, so in this paper empirically there will be looked at what that amounts to in practice.

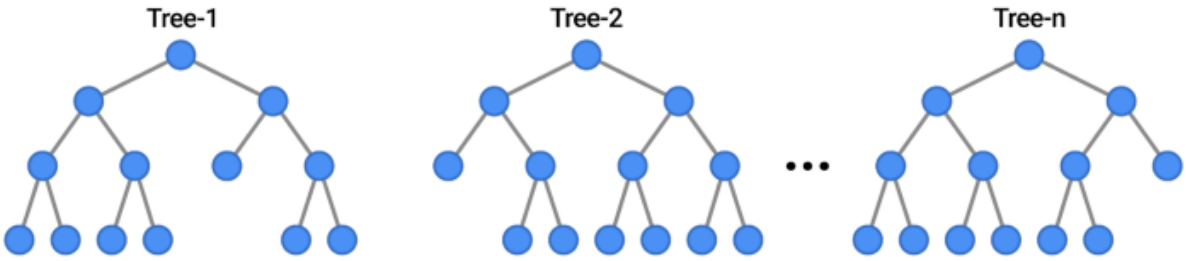


Figure 2.4: An example of a Random Forest with n number of trees with a maximum depth of 3 and 2 splits per node (Loukas n.d.).

2.2.4. DBSCAN

DBSCAN stands for Density Based Spatial Clustering of Applications with Noise. DBSCAN clusters points together and classifies points that fall outside of the parameters as noise. The two parameters that DBSCAN

uses to cluster points are: the minimum amount of points necessary for it to be called a cluster (minimum sample size or MinPts in figure 2.5) and the maximum distance the points are allowed to have from another point within the cluster (epsilon ϵ). When a point either does not have enough neighbours or is too far away from another nearest point it is classified as noise (Daszykowski and Walczak 2009). Figure 2.5 showcases the principle of DBSCAN.

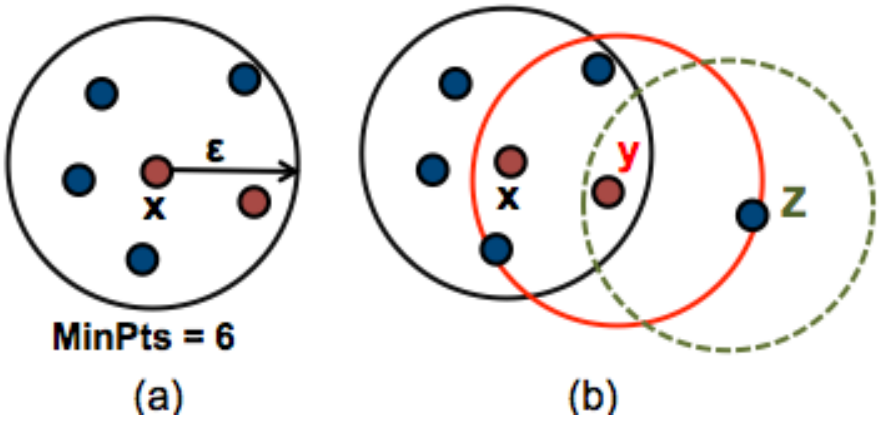


Figure 2.5: An example showcasing how DBSCAN works with an epsilon (ϵ) and a minimum sample size of 6 (*DBSCAN* n.d.). (a) showcases all the points within the cluster, (b) shows point z falling outside the cluster since within a maximum distance of ϵ of y there are not enough points, 6, to make a cluster.

2.3. Previous work on tree segmentation

A lot of research has already been conducted on tree segmentation using point clouds, some using deep learning and some using other machine or feature learning. Tree segmentation is the separation of trees from the data surrounding it. Previous works that have been conducted on segmenting trees that use machine learning are: S. Li et al. 2017 which makes use of a voxel based method. Yan et al. 2020 makes use of a kernel based approach which determines the tree crown based on a local maximum (treetop). Xia et al. 2021 is based upon an inversion method of the tree. All these methods segment trees from datasets only containing trees, while the goal of this thesis is to segment tree points from non-tree points. Zhang, Zhou, and Qiu 2015 makes use of a combination of lidar with NDVI to segment the trees. Man et al. 2020 makes use of a combination of hyperspectral and lidar data to segment trees and grasses in an urban area. The downside of these two methods is that it requires an extra dataset to extract the trees. Itakura and Hosoi 2018 uses an uphill clustering method where it first recognizes the tree trunks and then moves upward to segment the canopies. J. Li et al. 2021 also makes use of an uphill clustering method but first makes supervoxels out of the dataset and then selects the trees based upon the created structure. Herrero-Huerta, R. Lindenbergh, and Rodríguez-Gonzálvez 2018 makes use of multiple machine learning techniques to determine different properties of the tree. The three before mentioned methods all make use of a ground-based LiDAR while the LiDAR from AHN4 is air-based. Some previous work that has been conducted using deep learning include Seidel et al. 2021 which transforms the 3D data to 2D and Chen, Gao, and Devereux 2017 which makes use of PointNet (Qi et al. 2017). Contrary to earlier performed research, this thesis will test the suitability of an non-end-to-end method on airborne LiDAR to seperate trees, which has not been assessed before

2.4. Deep Learning for Point Clouds

In this section end-to-end deep learning methods which have been used for point cloud classification will be discussed. End-to-end means that there is no feature generation done before the point is input into the algorithm. The issue that arises with the use of CNN's for point clouds is that point clouds are not a fixed grid, something which is necessary for conventional CNN's since the convolution operation requires a fixed grid. One of the solutions that is often used is by transforming the point cloud to voxels (Maturana and Scherer 2015). Another method is by projecting the point cloud into 2D space from different angles. Rizaldy et al. 2018 does this by converting the whole point cloud into one image to save on computational power while Hang et al. 2015 and C. Wang, Pelillo, and Siddiqi 2019 uses multiple images from different angles. Seidel et al. 2021 also makes use of a transformation of 3D to 2D along multiple axii but also augments the data re-

sulting in a better accuracy. The issue that arises with both these methods is that they require pre-processing of the data which increases the computation time and has a higher chance of losing data in translation.

The first method to suggest using the point-clouds as a direct input is PointNet (Qi et al. 2017). PointNet is based upon creating a global feature vector which it pools with the features per point to eventually create a classification. The main advantage of PointNet is that the data does not have to be transformed, which reduces computation time and the loss of information due to translation. However the main issue of PointNet is that it extracts features per point and as such can lose the local geometric properties of the point-cloud (Thomas et al. 2019) (Peyghambarzadeh et al. 2020). A newer method is suggested which does not require any transformation of the data to either 2D or voxels and does take geometric properties into account is by the use of kernels. Yiru Shen et al. 2017 and Thomas et al. 2019 both make use of points as kernels while Peyghambarzadeh et al. 2020 uses a plane as kernel.

The thesis will draw inspiration from Nurunnabi et al. 2021. In Nurunnabi et al. 2021 ground surface extraction from aerial laser scanning point clouds is the goal. This is somewhat similar to extracting trees from AHN4 data since AHN4 is a point cloud obtained by aerial laser scanning. However tree points have entirely different features than ground points, so whether this method will correctly translate to trees is uncertain. The way the ground points are extracted in Nurunnabi et al. 2021 is by first computing extra features making use of PCA, which gives local spatial attributes of the points. The points are then put through a simple Neural Network with these extra features to finally make the classification. This is referred to in the paper as a non-end-to-end method while a network such as Pointnet (Qi et al. 2017) is referred to as an end-to-end method. Because of this approach a simple Neural Network will suffice to get a decent classification which saves in computational power and makes it easier to adjust the inbetween steps contrary to an end-to-end method. Combined with an accuracy of more than 97% makes this a very compelling reason to use this research as the inspiration of extracting trees from AHN4 data.

The main advantages of using an end-to-end method like PointNet (Qi et al. 2017) over a non-end-to-end method suggested by Nurunnabi et al. 2021 is that deep learning networks like PointNet learn the features itself. However an end-to-end method is computationally more intensive than non-end-to-end method. Another big disadvantage is that it is harder to almost impossible to exactly see how the algorithm learns and what goes wrong or right. As such for a method that uses a shallower Neural Network such as suggested by Nurunnabi et al. 2021 it is easier to control all the in-between steps contrary to a method such as PointNet. This makes it easier to improve the network and learn the where the mistakes happen, hence the reason why this paper will focus on the method suggested by Nurunnabi et al. 2021.

2.5. Training Strategy

For machine learning and especially deep learning the training strategy is the most important part to make an algorithm work properly. As such, the quality of the training data is very important. There is not a perfectly labelled dataset available by Aerial LiDAR for the Netherlands, so a training dataset will have to be created which will most definitely have some faults such as false positive and false negative labels. Creating training data can for example be done by hand or by the combination of existing datasets.

One possibility of dealing with corrupt samples is by removing samples with the highest loss difference as proposed by (Yanyao Shen and Sanghavi 2019). Loss is a metric by which a neural network trains itself. It can for example be something as the mean squared error. When calculating the loss difference for each sample the samples with the highest difference change the network the most and are most likely to be corrupt according to Yanyao Shen and Sanghavi 2019.

3

Methodology

In this chapter an overview of the methods used in this thesis are given. In sections 3.1.3, 3.1.1 and 3.1.2 the relevant datasets for the thesis are explained and made. In section 3.2 the basis for the algorithm used is explained and the different methods tried for feature engineering are shown. In sections 3.3, 3.4 and 3.5 the algorithm and different setups used for the algorithm are shown and explained.

3.1. Preprocessing

In this section the datasets and how they are acquired and processed are discussed. For machine learning both training and validation datasets are needed.

3.1.1. Validation Dataset 1

The accuracy obtained in classifying the validation data is the number of points correctly classified in the validation dataset divided by the number of points in the validation dataset. The accuracy will either be displayed as a percentage or a number between 0 and 1 where 1 is fully accurate and 0 fully inaccurate. This can also be done for just the tree points or non-tree points, in that case the accuracy of the tree classification or the non-tree classification gets obtained. The accuracy of the tree points is also called the true positive and the accuracy of the non-tree points the true negative. False positives are non-tree points falsely classified as tree points, false negatives are tree points falsely classified as non-tree points. Adding the false positives and true positives should add to 1, the same goes for false negatives and true negatives. A confusion matrix is a 2x2 matrix that has the true positive in the top left, false positive top right, false negative bottom left and true negative bottom right.

The validation datasets are hand selected datasets of tree and non-tree points. The reason it is hand selected is to make sure all the points are correctly classified. Two validation datasets are created, one validation dataset is created to help find the optimal setup and one validation dataset is used to quantify the performance after the optimal setup has been found. Reasoning behind this is that it is easier to create a validation dataset that contains problem scenarios after there has been a prelimenary result showcasing those problem scenarios. For the first validation dataset, Validation Dataset 1, the following three scenarios have been classified by hand to get a good idea of how well the classification worked. Figure 3.1 shows Validation Dataset 1.

- Big stand alone tree
- Small tree close to building
- Collection of non-tree points

Figure 3.1: Validation Dataset 1, green are trees and blue is other. The lower and more to the left located tree is the big stand alone tree and the higher and more to the right located one is the small tree close to a building.

The reasoning behind choosing these scenarios is to best represent the problems the algorithm will face. Namely extracting tree points only surrounded by other tree points, tree points surrounded by non-tree points and non-tree points such as cars, lanterns, fences, etc. In the end that the accuracy that will be given by this validation dataset will give a good indication of how well the algorithm is working. Validation dataset 1 has the following properties:

- Number of points: 7163
- Number of tree points: 4185
- Number of non-tree points: 2978

3.1.2. Validation dataset 2

A second validation dataset, validation dataset 2, has also been created for the purpose of comparing Random Forest and Neural Network in the final results. This validation dataset was made after the first results were created. The reason behind this is that since there is a preliminary result already present, problem scenarios can be more easily identified by eye. From the preliminary result this validation dataset was hand selected from problem scenarios as described in section 4.2. The goal for this validation dataset is to give a clear insight into which algorithm works better, since this validation dataset consists of more problem cases then the previously described validation dataset the accuracy will as a result probably also be lower. Keep in mind that the accuracies made using validation dataset 2 will then also not be comparable to accuracies made using validation dataset 1. Validation dataset 2 is not visualized since it consists of a lot of small areas selected of problem scenarios spread out over a large area, so visualizing the dataset will not give any clarity. The second validation dataset has the following properties:

- Number of points: 82780
- Number of tree points: 41390

- Number of non-tree points: 41390

3.1.3. Training Data

For machine learning algorithms a training dataset is required upon which to train the algorithm. The training data should represent the goal of extracting trees from AHN data. To create a training dataset the 'other' class of AHN4 is combined with the Delft tree register (Delft 2021) which contains the trees that the Delft municipality maintains see figure 3.2. This tree register contains the location of the tree, the Latin name and other specialties. The only field which the tree register contains that is relevant for this thesis is the location of the trees. It does contain fields for height and diameter which might also be relevant, however maybe 1 in 10 trees has these fields filled in and if they are filled in than it is generally given in a range. For example if the height is given for a tree it is given as between 6-12 metres (Delft 2021). Because of these reasons these fields are not used to determine a more accurate size of the tree. What should be noted from this dataset is that it only contains trees on the side of the road. In other words, this dataset does not contain citizen owned trees within the gardens. This can, when used to create training data, result in false negative labels since these points belong to trees which will falsely be labeled as non-tree point (Delft 2021). To give an indication of the number of false positives and false negatives present in the training data is the training data run against Validation Dataset 1 (section 3.1.1). Keeping in mind that Validation Dataset 1 does not represent a whole dataset, it follows that it will not give a full overview of the quality of the training data but rather, serves to only give an indication. The following results were obtained:

- Accuracy: 63.7 %
- True positives: 62%
- False positives: 38 %
- True negatives: 100%
- False negatives: 0%

It was opted to go for the tree register of Delft because the author of this thesis is located in this city, which makes it easier to inspect trees in person if it proves necessary. There are other tree registers maintained by other municipality's in the Netherlands (such as Eindhoven) however a tree register with more useful information, such as an accurate height or diameter, than the one from Delft was not found. The datasets used in this thesis are small subsets of the AHN data located in Delft. The reason for using a smaller subset is purely computational. (Natijne van 2021) was used to get a subset of 1km by 1.25km see figure 3.2.

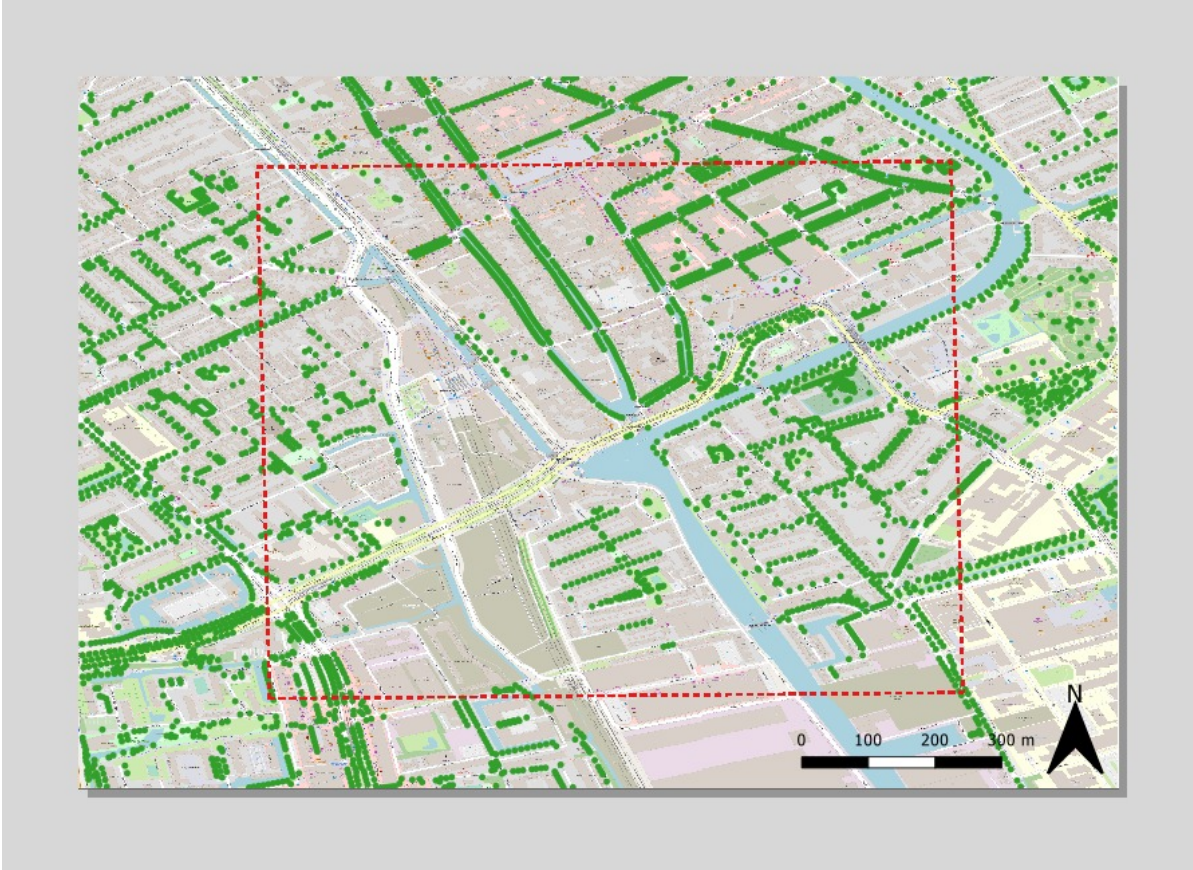


Figure 3.2: One of the AHN4 subsets obtained from geotiles (Natijne van 2021) represented by the red square, the green dots represent trees maintained by the municipality of Delft.

To be able to use the Delft tree register it will have to be in the same coordinate system as that of AHN. Using QGIS the tree register dataset is first reprojected to EPSG:28992 - Amersfoort / RD New. Next the points in AHN are cut in a 3 metre radius around the locations of the trees found in the tree register. The reason behind using this method for obtaining a dataset is that there is a lack of classified point-clouds with the same properties as AHN4 and with a by hand classified training dataset the same amount of training data will be very work intensive to obtain. 3 metres is chosen as an arbitrary number to get the points around the tree. After testing this number also ended up being the best working radius. It was opted to not use a dynamic diameter for trees since the tree register only contains the diameter for a very small number of the trees present and the diameter if given generally consists of a large range. These values were then not trustworthy to use to create a dynamic diameter per tree. In the end every point within the 3 metre radius of the location of the trees from the municipality of Delft is selected as a tree point and everything outside as a non-tree point.

The downside of this training strategy is it creates training data containing false positives and false negatives. This can cause overfitting. Overfitting is when an algorithm fits to closely to the training and involuntarily fits to much on the falsely classified data. This causes a decrease in accuracy and is something which will occur with the training data used if not taken into account.

3.2. Non-end-to-end Method

The method in this thesis is inspired by the method used by Nurunnabi et al. 2021. Namely the spatial features are created, such that the classification algorithm does not have to learn them itself, making use of PCA instead of letting the algorithm learn this. The advantage of such a method is that it is computationally less intensive then an algorithm which has to learn the spatial attributes itself. It is also easier to control the in-between steps since more steps are implemented manually than a Pointnet (Qi et al. 2017) for example. The workflow described in figure 3.3 will be used for both the Random Forest and Neural Network. In this

workflow both the algorithms are already trained so there are no steps included for training the algorithms.

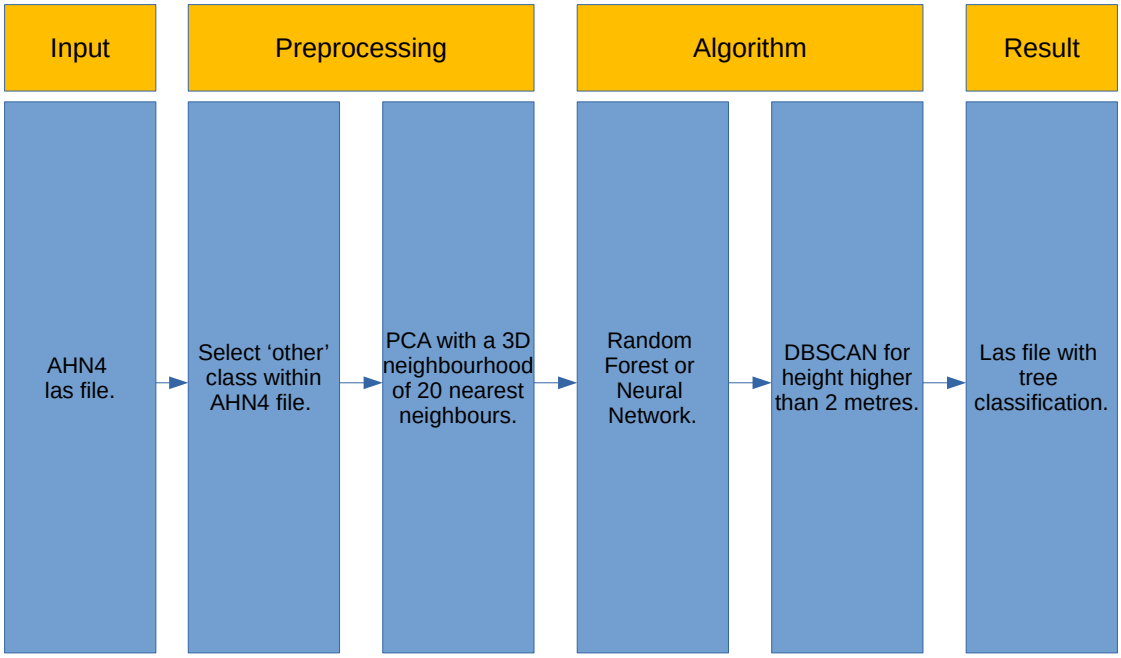


Figure 3.3: Workflow used to obtain the results, in this workflow the algorithm is already trained.

3.2.1. Feature Engineering

To be able to get spatial features per point Principal Component Analysis (PCA) is applied. This is done within a given neighbourhood of the point so that the features created are representative of the local characteristics. PCA gives three eigenvalues from which different characteristics of the point can be calculated (see section 2.2.2). Another thing that was added to the dataset was a delta z component to be able to get an accurate height of a point from the ground up, delta Z is computed as follows: $\frac{z-minz}{maxz+minz}$. There min z is the minimum height in the entire dataset used and max the maximum z in the entire dataset used.

To validate which feature is of importance for tree recognition a histogram is made for each feature in table 2.1, delta Z and features already present in AHN4 (section 2.1) for tree points and non tree points (see figure 3.4). The training dataset was used to determine what was a tree and a non-tree point (see section 3.1.3). The tree points are all points from the 'other' class in AHN4 within a 3 metre radius surrounding a tree location from within the tree registry. The rest of the points are then the non-tree points. Take into account that this method has false positives and false negatives within the dataset since a 3 metre radius is not the perfect circumference for all the trees. Also trees which are not maintained by the municipality of Delft, such as trees within private gardens, are not included in the dataset. The features that show a difference in distribution between tree and non-tree points are then selected to be used for classification. In the end the 12 features presented in table 3.1 were used for classification, these include the features which were already present in AHN4 (see section 2.1).

Feature	Source
delta Z	AHN4
Number of Returns	AHN4
Amplitude	AHN4
Reflectance	AHN4
Intensity	AHN4
Curvature	PCA
Planarity	PCA
Linearity	PCA
Omnivariance	PCA
Anistropy	PCA
Change of Curvature	PCA
Scattering	PCA

Table 3.1: All the features used for classification.

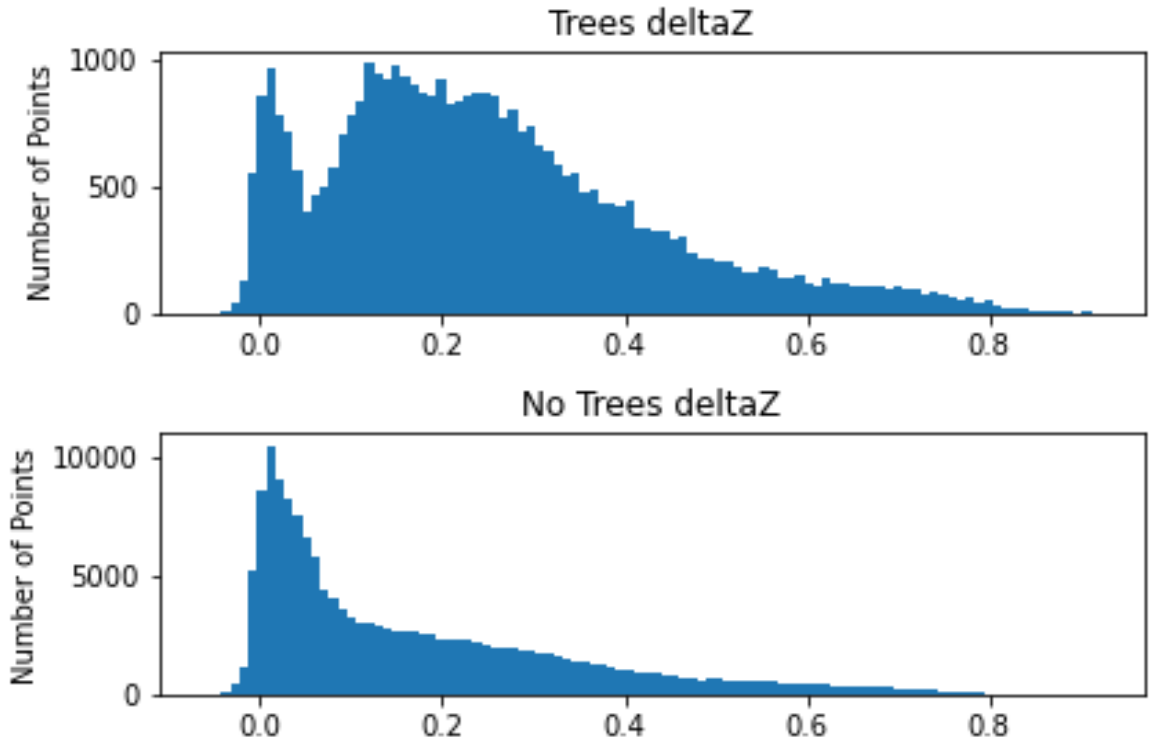


Figure 3.4: An example of a histogram of the distribution of tree and non-tree points used to determine which features to use for classification. This example shows the distribution for delta Z scaled from 0 to 1.

Another attribute of the features is that they are dependent on the neighbourhood, since PCA calculates a points characteristics based on a neighbourhood surrounding the points. What kind of neighbourhood is used impacts the features values. The following neighbourhoods were used:

- PCA with 5 nearest neighbours in 3D
- PCA with 10 nearest neighbours in 3D
- PCA with 20 nearest neighbours in 3D
- PCA with 2D 20 nearest neighbours
- PCA with circular neighbourhood of 1m radius

The neighbourhoods with the nearest neighbours were chosen because a points feature is generally dependent upon its closest neighbours. These are the nearest neighbours in a three dimensional space. The reasoning behind chosing 5, 10 and 20 is that any less neighbours then 5 probably doesn't represent the neighbourhood very well and is more sensitive to anomalies in the data. No more then 20 neighbours is chosen because computationally having more then 20 neighbours is very intensive. This was also done in a 2D neighbourhood so in the xy dimension to see if the z dimension is important for computing the features. Finally a circular neighbourhood was also computed since trees have a by approximation circular shape so the hope was that this would better catch the characteristics of the tree.

3.3. Neural Network

The neural network used is a shallow feed-forward network made using the Tensorflow module in python. It has the following construction: 6 dense layers of 70 Neurons each with a rectified linear unit (ReLU) (see figure 2.2b) activation function and a final layer of 1 Neuron with a sigmoid (see figure 2.2a) activation see figure 3.5. A ReLU function is a function where, for a negative x, the y is 0 and, for a positive x, it is linearly positive. Using a linear activation function works well for simple classification problems such as in this thesis. Using ReLU has an advantage over using a linear activation function though since it is not activated for negative input, so it is computationally less intensive. Leaky ReLU is also an option, leaky ReLU is when instead of giving an output of 0 for negative values, such as ReLU, it gives a very small negative output. The advantage of leaky ReLU is that it prevents data from being lost since the neuron is always activated, in some cases this is necessary. However in this paper it is unnecessary since there are no unactivated neurons, so no data gets lost, and as such will not be used. As a last layer a sigmoid activation function gets used since a classification between 0 and 1 is wanted for any sort of input, since a point is either a tree or not a tree, which a sigmoid function provides, see figure 2.2a. As an optimizer (see section 2.2.1) ADAM is used, ADAM is quoting the author: "*The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters.*" (Weinmann et al. 2015). Since it is computationally efficient and the dataset used in this paper can consist of up to millions of points, Adam was chosen as optimizer. The other ways the network can be tuned is subdivided into two parts: by adjusting the training data and by tweaking the network parameters.

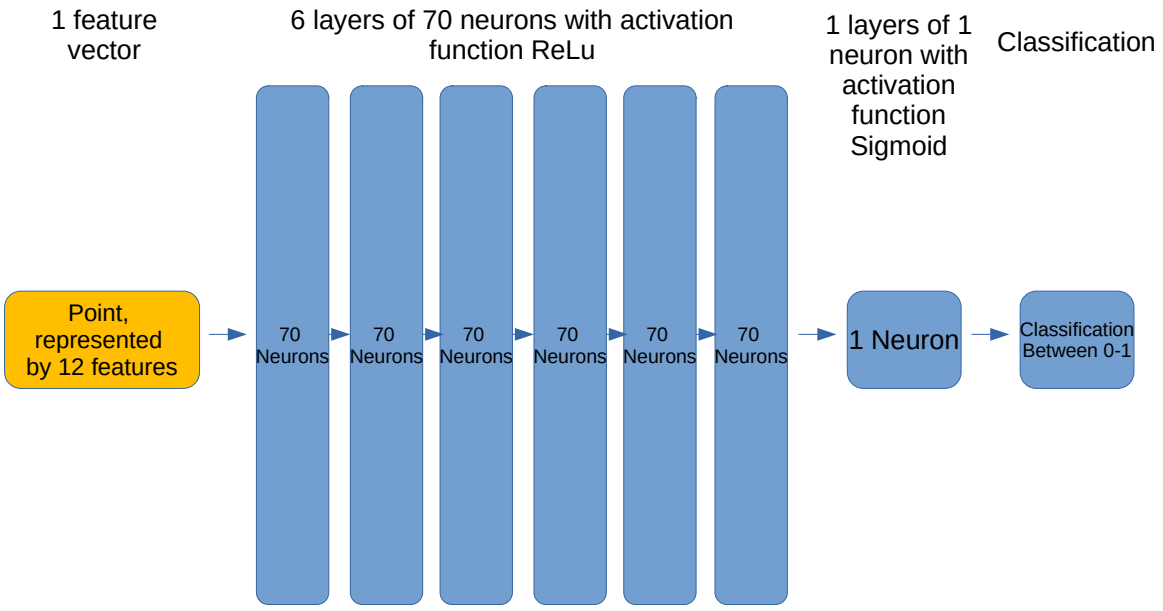


Figure 3.5: The structure of the Neural Network

The output of the Neural Network, figure 3.5, is a value between 0 and 1. Here 0 represents that the algorithm

is certain that the point is a not a tree while 1 represents that the algorithm is certain that the point is a tree. To get a final classification the number gets round to the nearest integer. This means that a feature vector is classified as a non-tree point if the output is < 0.5 and classifies as a tree point if the output is => 0.5, where in the final classification 0 represents a non-tree point and 1 a tree point.

3.3.1. Training strategy

For a Neural Network, or any type of supervised machine learning, training data is vital. Different training strategies can be applied to the training dataset described in section 3.1.3.

- Different distributions of tree and non-tree points.
- Using all classes (see section 2.1) instead of just the 'other' class in AHN4.
- Only using samples with a low loss.
- Applying PCA over the features.

Different distributions were used to see whether the neural network would perform better with more or less tree points. The different distributions used were 40% non-tree points and 60% tree points, 60% non-tree points and 40% tree points and a 50/50 split.

Another training strategy that can be applied is to use all the classes instead of just the 'other' class in AHN4. The reasoning behind this is that, when using all the classes, there will be new neighbouring points within the these classes not used before. Which might give the points more definite features that can be more easily recognized by the neural network.

A way to try removing corrupt samples, to improve the training data, is by looking at the loss. The loss is an error (such as the root mean squared error) which compares the actual label of the points to the predicted labels by the neural network. This can also be done per point. From the loss per point the points with the highest loss are then removed. These are the points that the neural network has the most problems to predict. Then the new training data without these high loss points gets used for the final product (see section 2.5).

To tackle overfitting PCA (see section 2.2.2) can also be applied to the features made, to condense them into less features. To see how many features there have to be made while representing the same amount of features as previously the variance can be used, see figure 3.6. In this figure it can be seen that the cumulative variance basically stays around 1 from five features and onwards. This means that for 5 features basically all of the 12 originally used features are represented.

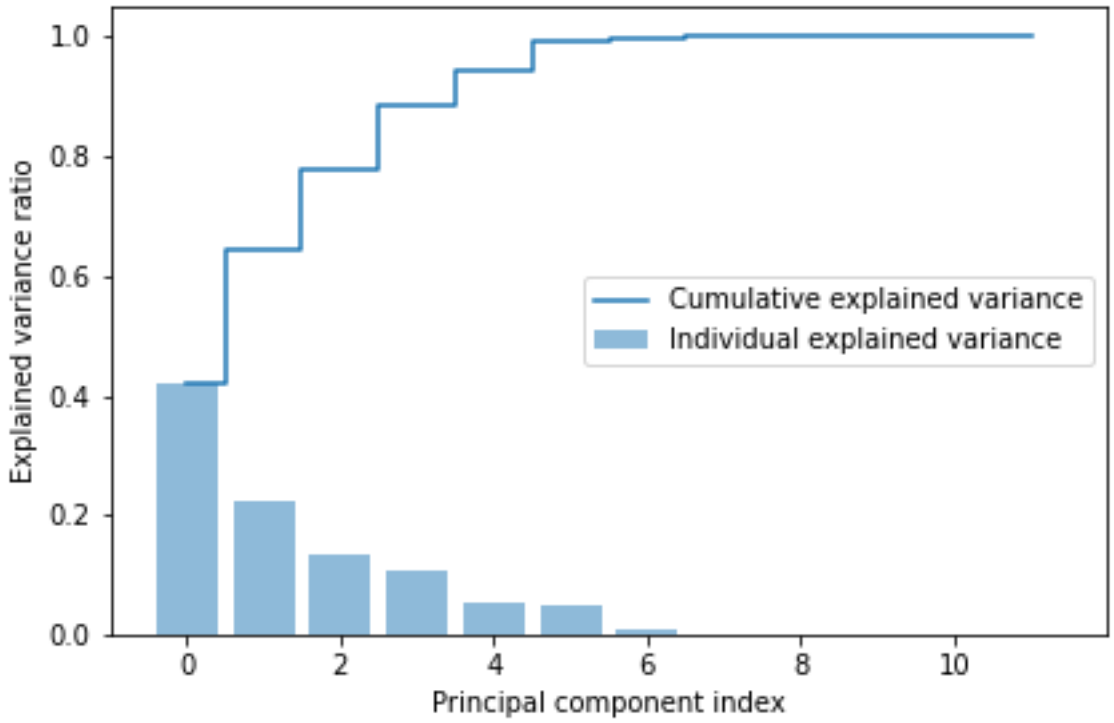
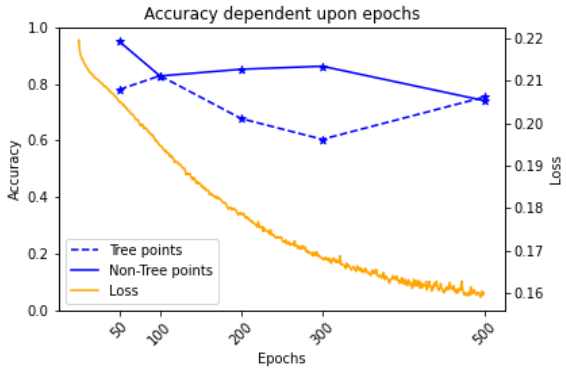


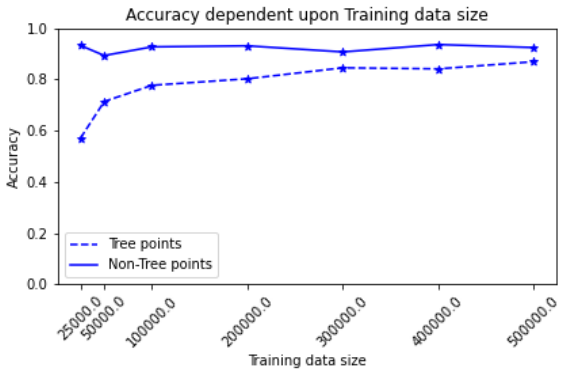
Figure 3.6: The variances of the features after applying PCA to it. The x axis represent the amount of principal components used and the y axis the variance of the components. So the higher the variance the more it represent the entire dataset.

3.3.2. Neural Network Parameters

The two main parameters that can be adjusted in the Neural Network are the number of epochs and the size of the training data. A neural network when it trains itself runs the training data through the network, one such run is called an epoch. By adjusting the amount of epochs the amount of times that the training data gets run through the neural network gets adjusted. The optimum is enough epochs that the neural network gets properly trained but not so much it gets overfitted. Looking at figure 3.7a 100 epochs seems to work best for classifying tree points. The loss is plotted against the accuracy for the epochs to demonstrate that the loss keeps going down even though the accuracy is not. This is probably due to overfitting, since the Neural Network assumes that the training data is perfect and as such assumes that it keeps improving the network. Looking at figure 3.7a it can be seen that 100 epochs is ideal and more than 100 results in a drop in accuracy due to overfitting. To create the final results a setting of 100 epochs is used.



(a) Epochs versus accuracy for tree and non tree points. Run for a training size of 100 000 and 20 nearest neighbours in PCA. The orange line represents the loss over all epochs.



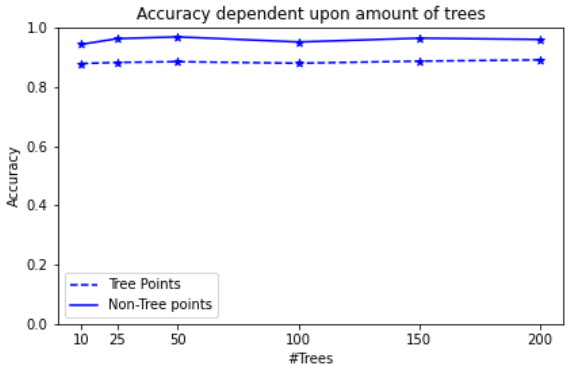
(b) Training size versus accuracy for tree and non-tree points. Run for 100 epochs and 20 nearest neighbours in PCA.

Figure 3.7: Different parameters of the Neural Network plotted for different settings against the accuracy obtained from validation dataset 1 (section 3.1.1).

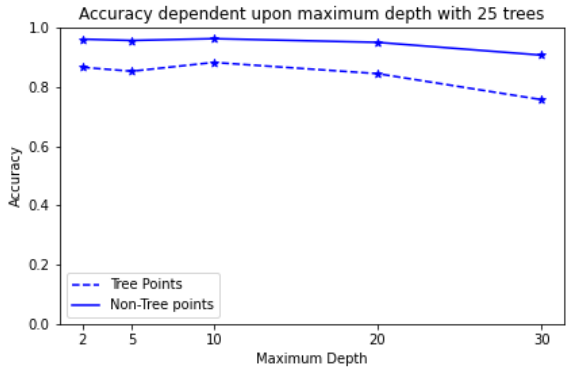
The size of the training data is the number of points passed through the network to train upon. Generally speaking more training data is better. However this is when the training data has no faults. And the training data used contains both false negatives and false positives, so using more training data can result in overfitting. Looking at figure 3.7b a training size of 400 000 points is ideal since from 500 000 points onwards the accuracy is decreasing so it starts to overfit. To create the final results a training data size of 400 000 points will be used.

3.4. Random Forest

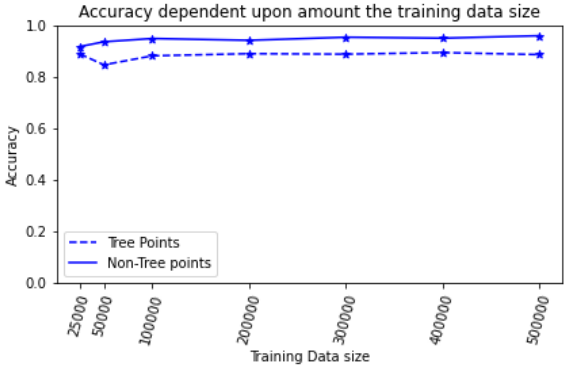
The second algorithm that is used is random forest. Random forest is an algorithm that consists of multiple tree predictors. For a Tree itself the two most important parameters are the number of splits, which is set at the square root of the number of features (see section 2.2.3), and the maximum depth. The maximum depth prevents the tree from overfitting. The parameter that can be changed over the entire forest is the number of trees (predictors). Changing these two parameters gives the results as seen in figure 3.8a and figure 3.8b. Based on these results 150 trees with a maximum depth of 10 each will work best.



(a) Amount of trees versus accuracy. Run with a maximum depth of 10 for each tree and a training size of 400 000 points.



(b) The maximum depth versus accuracy. Run with 100 trees and a training size of 400 000 points.



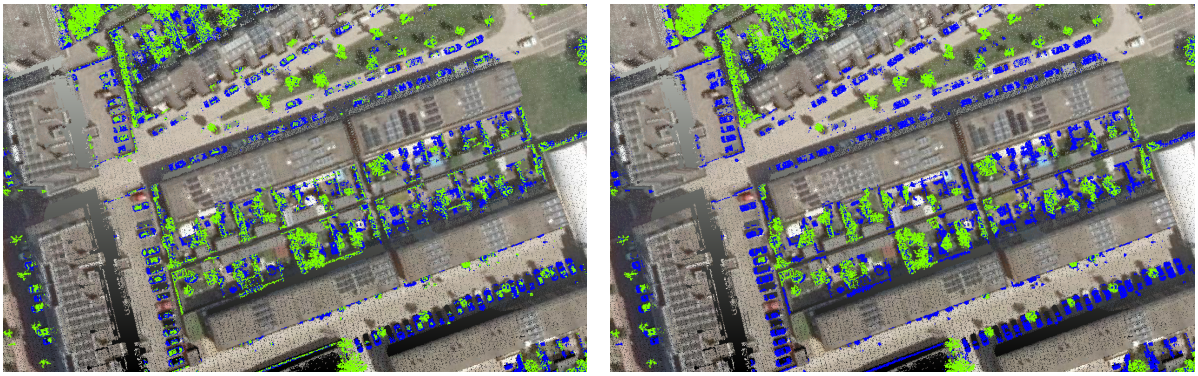
(c) Different training data sizes versus accuracy. Run for a setup of 150 trees with a maximum depth of 10.

Figure 3.8: The different parameters that can be adjust for Random Forest and what kind of effect they have on the accuracy obtained from validation dataset 1 (section 3.1.1).

Another parameter that can be adjusted is the training data size. Figure 3.8c shows the accuracy versus the training data size, here it can be seen that the accuracy starts stabilizing from training data size of 100 000 points and onwards.

3.5. DBSCAN

To reduce the amount of false positives and false negatives spread throughout the result from the Neural Network, density-based spatial clustering of applications with noise (DBSCAN) is applied (see figure 3.3). DBSCAN finds clusters of tree points, includes the points that are false negatives and excludes the points that are false positive. The reasoning behind using DBSCAN as the last step in the workflow is to reduce false negatives and false positives and segment the trees, so the first step is generating features which get processed by the Neural Network or Random Forest and the output of the Neural Network or Random Forest is then the input of DBSCAN. The problem with the output straight out of the Neural Network or Random Forest is that it contains a lot of spreadout false positives (see figure 3.9a) which makes it hard to segment trees. DBSCAN improves upon this outcome by providing clusters of trees while at the same time removing the false positives. Figure 3.9 gives an example of the effect of DBSCAN.



(a) The result of the Neural Network, on an area with a lot of trees present, before DBSCAN is applied. Blue is a non-tree point and green is a tree point.

(b) The result of the Neural Network, on an area with a lot of trees present, after DBSCAN is applied. Blue is a non-tree point and green is a tree point.

Figure 3.9: An example showcasing the effect of DBSCAN on the result of the Neural Network.

It works best when the DBSCAN is applied to the dataset above 2 metres, since the trees are defined to be at least 2m high. This already reduces noise and makes the clustering easier. At the end all the points below the clusters, so below 2 metres, are given the same classification as the points within the clusters, so above 2 metres. After this has succeeded, all the points within the cluster are found to include the false negatives, and all the points not included in the cluster are than labeled as negatives to exclude the false positives. Figure 3.10 visualizes the steps used to obtain the result.

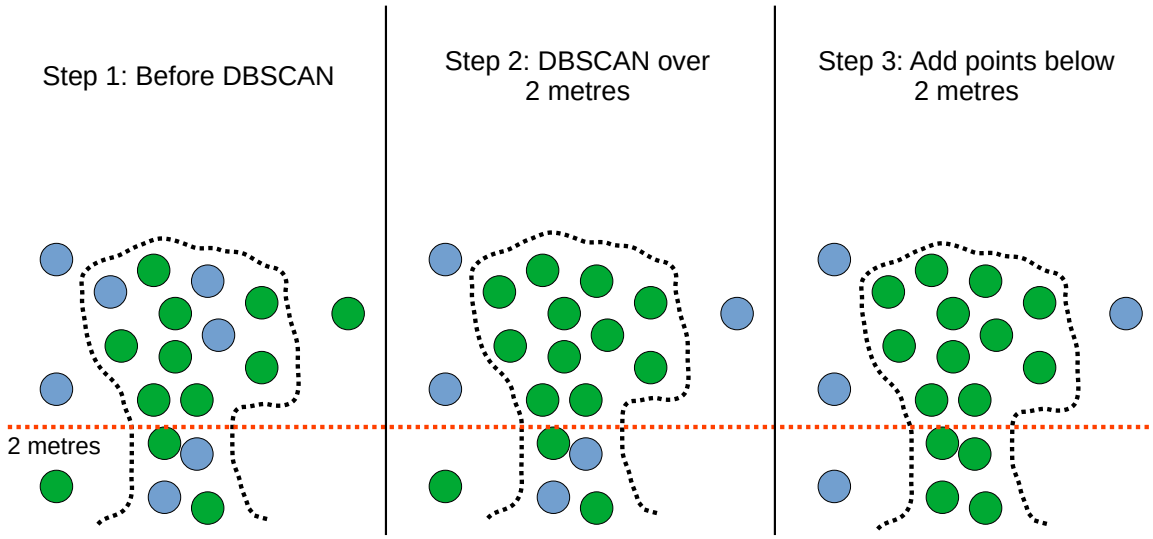
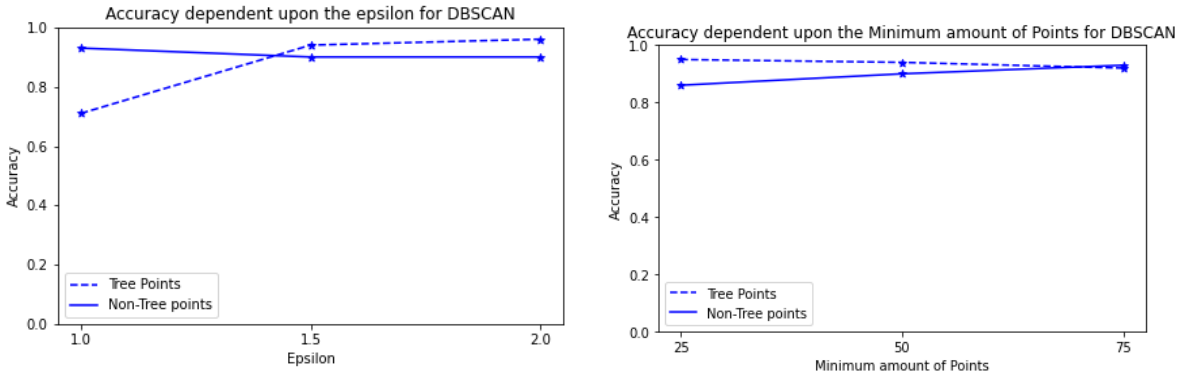


Figure 3.10: The steps used in DBSCAN algorithm visualized. Green points are tree points and blue points are non-tree points.

The two parameters that can be changed in DBSCAN are epsilon, which is the minimum distance a point can be from another point within the cluster, and the minimum amount of points that have to be present to call something a cluster. Figure 3.11 highlights the effect of the different parameters on the performance of DBSCAN.

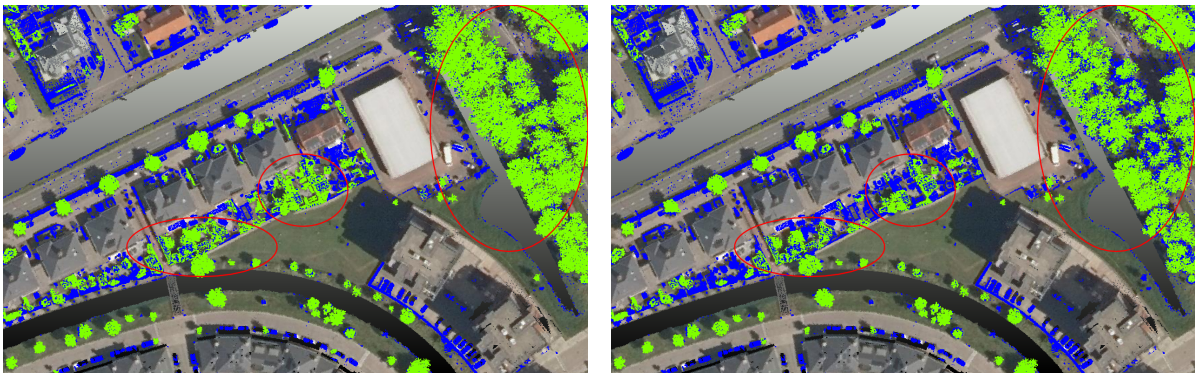


(a) The effect of different epsilons on DBSCAN.

(b) The effect of different minimum amount of samples on DBSCAN

Figure 3.11: The effect of the different parameters of DBSCAN on the accuracy obtained from validation dataset 1 (section 3.1.1). Green are tree points and blue are non-tree points. The red circles indicate areas where the algorithms differ from eachother. The scenarios in this figure were run for the Neural Network with 100 epochs and a training data size of 400 000 points.

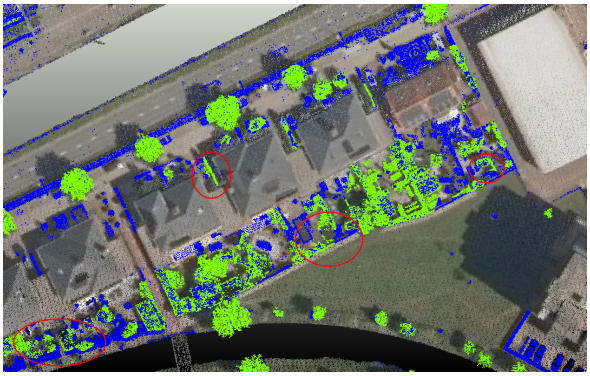
Figure 3.11 indicates that the algorithm would work best with the setting of epsilon is 2 metres and a minimum sample size of 75. However it was found to work best, by directly looking at the output, with an epsilon of 1.5 metres and a minimum amount of samples of 50 (section 2.2.4). Figures 3.12 and 3.13 showcase this by looking at direct results of DBSCAN. Figures 3.12a and 3.12b showcase the scenarios for a minimum sample size of 50 and 75. The scenario with a minimum sample size of 75 loses a part of the tree points within the gardens, in the middle of the figure highlighted by the two red circles, compared to the minimum sample size of 50. In the top right of the figure, highlighted by the red circle, the scenario with the minimum sample size of 75 also loses a part of the tree points compared to the scenario with the minimum sample size of 50. Comparing figures 3.13a and 3.13b the scenario with an epsilon of 2 metres shows that it includes more non-tree points in the clustering compared to the scenario with an epsilon of 1.5 metres. This is evident in the gardens within the picture, especially the hedges at the lower side of the gardens in the middle of the figure which is highlighted by the red circles. The settings of an epsilon of 1.5 metres and a minimum sample size of 50 worked the best in the end and will be used to generate the results.



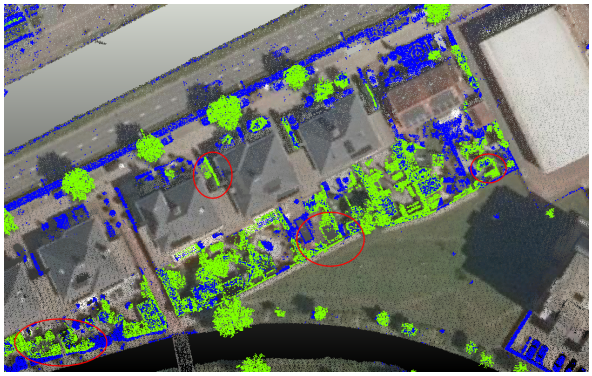
(a) DBSCAN with an epsilon of 1.5 metres and minimum amount of samples of 50.

(b) DBSCAN with an epsilon of 1.5 metres and minimum amount of samples of 75.

Figure 3.12: The effect of minimum sample size of 50 and 2 metres on DBSCAN. Green are tree points and blue or non-tree points. The red circles indicate areas where the algorithms differ from eachother. The scenarios in this figure were run for the Neural Network with 100 epochs and a training data size of 400 000 points.



(a) DBSCAN with an epsilon of 1.5 metres and minimum amount of samples of 50.



(b) DBSCAN with an epsilon of 2 metres and minimum amount of samples of 50.

Figure 3.13: The effect of epsilon of 1.5 and 2 metres on DBSCAN. Green are tree points and blue or non-tree points. The scenarios in this figure were run for the Neural Network with 100 epochs and a training data size of 400 000 points.

4

Results

In this chapter the results from this thesis are shown. First the results from the different neighbourhoods used in feature engineering are presented. Second the final results for both the Random Forest and Neural Network are shown and problem cases are highlighted. Finally some other results are shown which were obtained while trying to improve the algorithm.

4.1. Results Neighbourhood Selection

The Neural Network method has been done for the following neighbourhoods:

- PCA with 5 nearest neighbours in 3D
- PCA with 10 nearest neighbours in 3D
- PCA with 20 nearest neighbours in 3D
- PCA with 20 nearest neighbours in 2D
- PCA with circular neighbourhood of 1m radius

That is, starting from a query point, neighbour points are selected using one of these neighbourhood scenarios. On all points of said neighbourhood, PCA features are computed as explained in section 2.2.2 and stored in the feature vector of the query point considered. The Neural Network, section 3.3, was used as the classification method. Between the Neural Network and the Random Forest there was no noticeable difference in the results, hence that the results are only shown for the Neural Network. The settings used for the training of the Neural Network are as follows: 100 epochs were used and a training data size of 100 000 points. All scenarios were run through that and compared to Validation Dataset 1 (section 3.1.1), see figure 4.1.

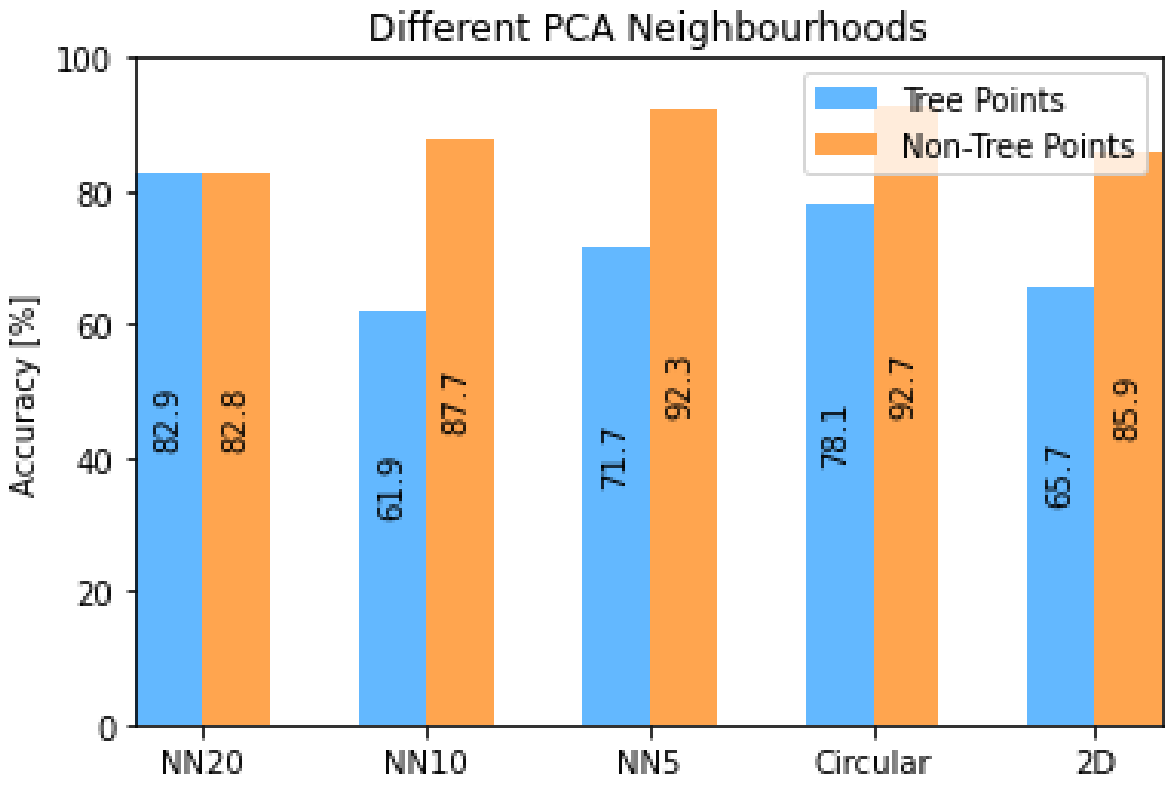


Figure 4.1: PCA for all different neighbourhood scenarios, 20NN/10NN/5NN represent the different nearest neighbours used in a 3D neighbourhood, 2D stands for the 2D scenario and circular for the circular neighbourhood.

When looking at the different number of neighbours used for the PCA the main thing that jumps out is that the less than 20 nearest neighbours neighbourhoods are less good at the classification of tree points. Looking at the tree classification (tree points in figure 4.1) the result drops with 20% when going from 20 to 10 neighbours but goes up again with 10% when going to 5 neighbours. While looking at the non-tree accuracy (non-tree points in figure 4.1) the accuracy goes up by around 5% when going from 20 neighbours to 10 and goes up again by 5% when going from 10 neighbours to 5. Even though the 5 and 10 nearest neighbours in 3D score higher for non-tree points compared to the 20 nearest neighbours in 3D, is it opted to go for the 20 nearest neighbours in 3D since it scores higher for tree point classification.

For a 3D neighbourhood the 20 nearest neighbours are the best classification for identifying trees. Two other neighbourhoods that were tried are the 2D 20 nearest neighbours and the circular neighbourhood of 1m radius. Compared to the 2D neighbourhood, the 3D neighbourhood scores more than 15% higher in recognizing tree points, however for non-trees the accuracy is lower but only by a small margin. When comparing the 3D 20 nearest neighbours to the circular neighbourhood, it looks like the circular neighbourhood scores better in classifying non-tree points with a slightly lower accuracy in recognizing trees compared to the 3D 20 nearest neighbours. However the circular neighbourhood is computationally the most intensive neighbourhood since in areas with high point density, a point can have up to 50 neighbours for which it has to perform PCA. When looking at the actual point clouds it can be seen that the circular neighbourhood has more spread out false positives than the 3D 20 nearest neighbours see figures 4.2 and 4.3. It can also be seen from these figures that the 3D 20 nearest neighbours neighbourhood has a better classification of trees compared to the circular neighbourhood. This can be seen from the more densely located clusters of green points (tree points) in figure 4.2. Instead of the more spreadout tree points seen for the circular neighbourhood which is not how trees should look like, tree points should be clustered together since trees are an object and not a random distribution of points.

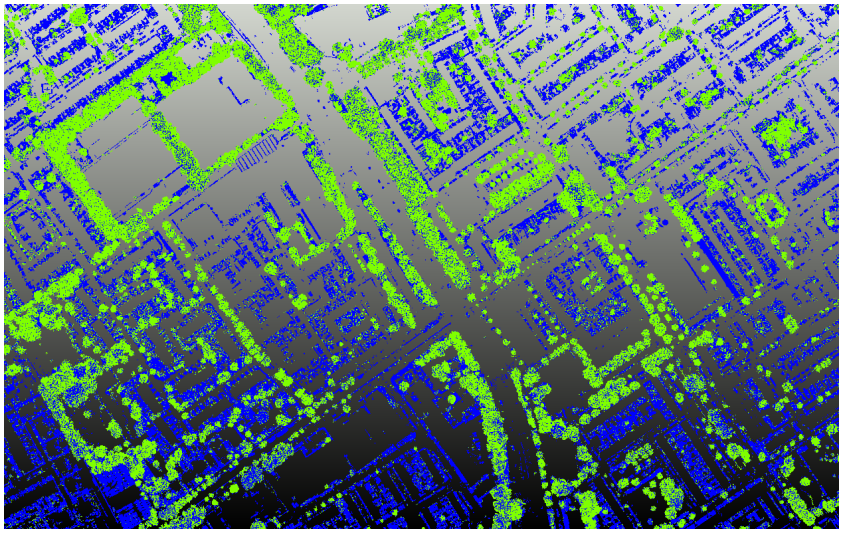


Figure 4.2: A top view of the classification done for a 3D neighbourhood of 20 nearest neighbours. Green are tree points and blue are non tree points.

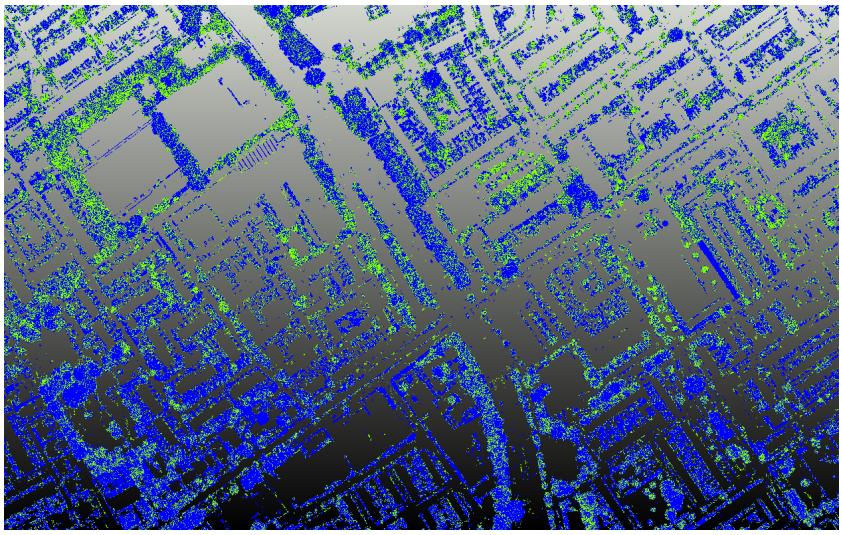


Figure 4.3: An top view of the classification done for a circular neighbourhood with a radius of 1 metre. Green are tree points and blue are non tree points.

4.2. Algorithm Results

To obtain the results the AHN4 data will go through the workflow as described in figure 3.3 for both algorithms, Neural Network and Random Forest. For each algorithm first the quantitative result of the tree point classification is represented. This is done by comparing the outcome of the algorithm to Validation Dataset 2 as described in section 3.1.2. The accuracies shown in this section will give a good representation of which algorithm, Neural Network or Random Forest, works better.

The results of the tree point classification are than investigated for three different scenarios which best reflect the different situations and difficulties the algorithms can encounter in a city. Presented results in section 4.2.2 are obtained after it has gone through DBSCAN (see figure 3.3) and the results in section 4.2.3 are the probabilities of the points before they have gone through DBSCAN. The following three scenarios are used:

- Cars
- Construction site

- Trees

4.2.1. Quantitative Results

To obtain the quantitative results AHN4 data is run through the workflow as described in figure 3.3 and than compared to Validation Dataset 2 (section 3.1.2). The setup that was used to train the Neural Network was as follows (see section 3.3):

- 50/50 split of tree and non-tree points in the training data for 400 000 points (see figure 3.7b).
- The Neural Network parameter for epochs was set at 100 epoch (see figure 3.7a).

The setup that was used to train the Random Forest was as follows (see section 3.4):

- 50/50 split of tree and non-tree points in the training data for 400 000 points (see figure 3.8c).
- Amount of trees: 150
- Amount of leaves, number of nodes per tree: 10

The following accuracies are obtained from comparing both Neural Network and Random Forest to Validation Dataset 2 using the above described setup, the confusion matrices can be seen in figures 4.5a and 4.5b.

- Random Forest: 72.4 %
- Neural Network: 66.9 %

(a) Confusion matrix for Random Forest after comparing the results to validation dataset 2.

(b) Confusion matrix for Neural Network after comparing the results to validation dataset 2.

Figure 4.4: The confusion matrices for Random Forest and Neural Network for validation dataset 2.

The main thing that is noticeable for both algorithms is that the algorithms have more issues with identifying non-tree points than tree poins. This can be seen by looking at the false negatives (bottom left) which scores way higher than the false positives (top right). Overall looking at the scores show that the Random Forest works better than the Neural Network for the setup used in this thesis.

Since Validation Dataset 2 only compares problem scenarios and not an even distribution of 'normal' and 'problem' scenarios, the accuracy is probably a bit lower than for a random subset of AHN4. To showcase how the algorithm works on an unbiased, however still very small, validation dataset, it is also compared to Validation Dataset 1. The confusion matrices can be seen in figure 4.5 and it corresponds to the following accuracies:

- Random Forest: 94.4%
- Neural Network: 92.8%

(a) Confusion matrix for Random Forest after comparing the results to validation dataset 1.

(b) Confusion matrix for Neural Network after comparing the results to validation dataset 1.

Figure 4.5: The confusion matrices for Random Forest and Neural Network for validation dataset 1.

These accuracies are considerable higher than the accuracies from validation dataset 2. Do keep in mind that validation dataset 1 is a very small dataset and as such probably does not capture the full diversity of AHN4. The actual accuracy of the algorithm when it would be compared to a fully classified part of AHN4, such as a 1km by 1km square, will probably lie between the accuracies shown for Validation Dataset 2 and 1.

Computationally did Random Forest work slightly faster than the Neural Network, however this is also depending on how the algorithm is programmed. Both Random Forest and the Neural Network could classify around 10 million points within half an hour on a normal computer (specs are shown below). Again this is dependent on the coding of the algorithm however it does show that the algorithm can be run effectively on a standard computer. The specs of the computer used are shown below:

- Memory: 15 GiB
- Processor: AMD® Ryzen 5 5500u with radeon graphics × 12
- Graphics: AMD Renoir
- Disk capacity: 512.1 GB

4.2.2. Scenarios: Absolute Classification

In this section the three different scenarios, 'cars', 'construction' and 'trees' will be shown and discussed. The scenarios 'cars' and 'construction' are both scenarios with which the algorithm struggles scenario 'trees' is to show where the algorithm performs well. For each scenario an RGB view is presented. Note that this RGB view is not from the exact same date as when the point cloud has been obtained. The RGB image is chosen to give a clear representation of the scenario's area. For each scenario there is also a bird eye's view of the classification obtained by both Random Forest and the Neural Network. This is plotted against all the classes present in AHN4 including the RGB values of those points. Keep in mind that the RGB values are pasted on values from the closest in time clear aerial image (Natijne van 2021), so the colors are not always accurate but should give a good indication. For scenarios 'cars' and 'construction' a frog eye's view gets shown, here only the classified 'other' class is shown. This is to prevent to have to many points obstructing the view. For scenario 'trees' this point of view is opted out because it was not possible to get a clear frog eye's view from this scenario.

Cars

The scenario 'cars' can be seen in figures 4.6, 4.7 and 4.8. The scenario consists of a car park with multiple cars parked inside of it, see figure 4.6. When looking at the results, figures 4.7 and 4.8, it can be seen that both algorithms struggle with identifying car points as non-tree points. Note that for figures 4.7a and 4.8a the RGB values are not true to the points but are pasted on during post processing, so sometimes cars seem to appear on points that are not car points. When comparing the results between Random Forest and the Neural Network it can be seen that **the Neural Network more often falsely classifies a car as a tree than Random Forest**. Looking at the frog eye's view in figures 4.7b and 4.8b it can be seen that only the rooftops

of the cars get falsely classified, this is because DBSCAN (see section 3.5) only gets applied from 2 metres upwards and than classifies the points below 2 metre with the same label as above.

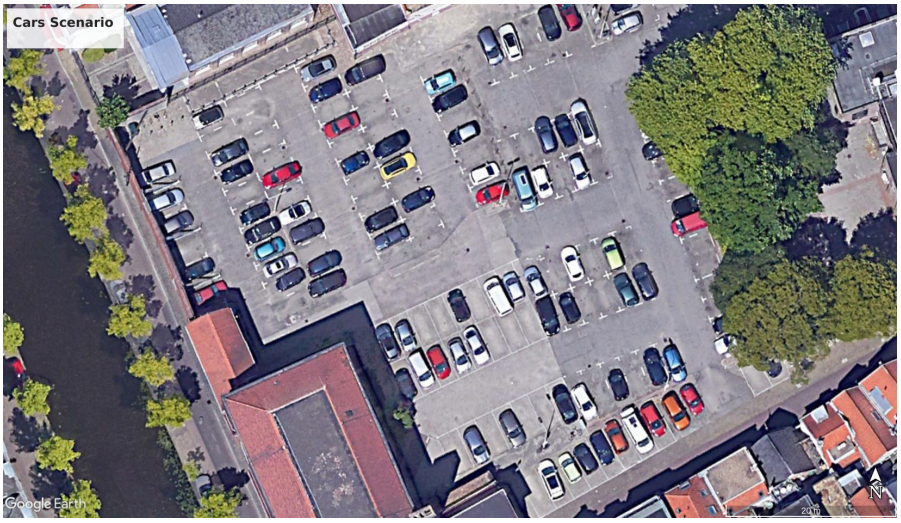
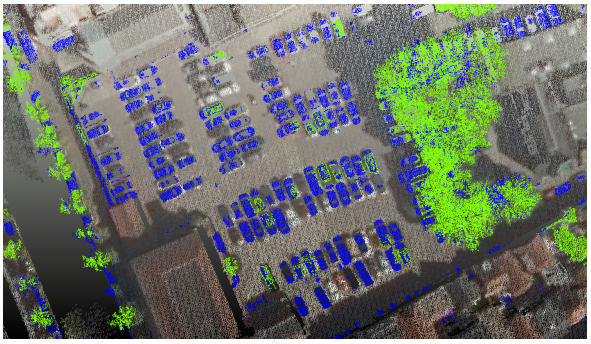
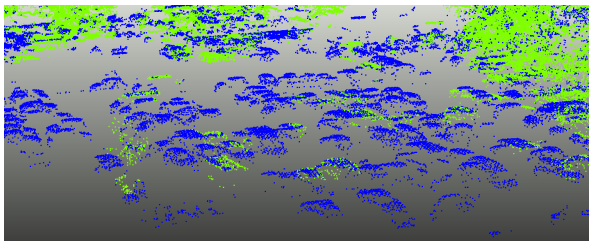


Figure 4.6: The RGB view of the scenario 'cars'

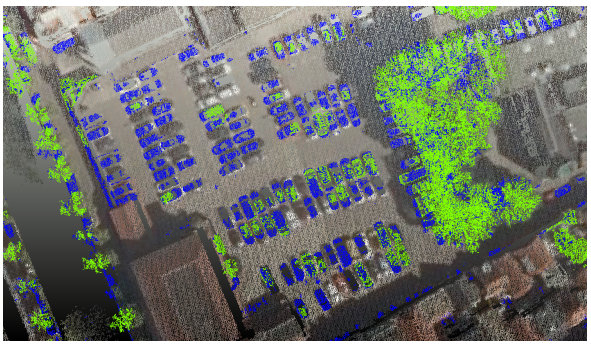


(a) The result of Random Forest for the scenario 'cars' seen from a bird eye's view. Green points classify as trees blue points classify as non-trees.

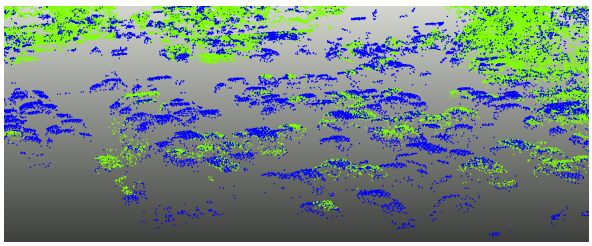


(b) The result of Random Forest for the scenario 'cars' seen from a frog eye's view. Green points classify as trees blue points classify as non-trees.

Figure 4.7: Results for Random Forest for the scenario 'cars'.



(a) The result of the Neural Network for the scenario 'cars' seen from a bird eye's view. Green points classify as trees blue points classify as non-trees.



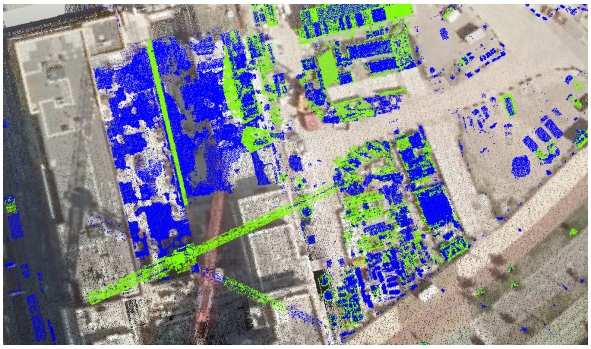
(b) The result of the Neural Network for the scenario 'cars' seen from a frog eye's view. Green points classify as trees blue points classify as non-trees.

Figure 4.8: Results for Neural Network for the scenario 'cars'.

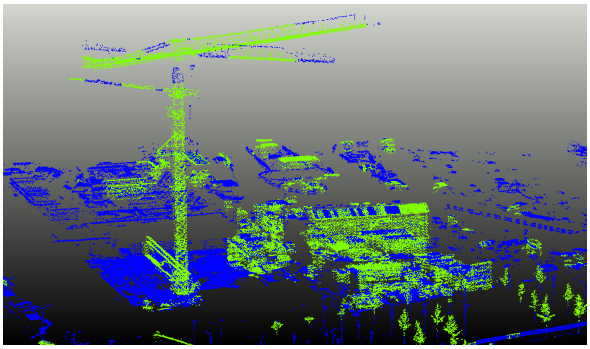
Construction

Scenario 'construction' can be seen in figures 4.9, 4.10 and 4.11. Scenario 'construction' consists of a construction site and a crane, which can be seen in figure 4.9. The scenario contains basically no trees except in the bottom left so the algorithm should give non-tree points for all other locations. Note that the crane in the RGB image is not in the same position as in the results, this is due to the RGB image not being of the exact same moment as the results. When looking at the results in figures 4.10 and 4.11, it is clear that the algorithm for both Random Forest and Neural Network has a lot of false positives since everything except the bottom left should consist of non-tree points. The bird eye's view in figures 4.10a and 4.11a shows that **the Random Forest has a little less false positives than the Neural Network**. When looking at the frog eye's view, figures 4.10b and 4.11b, both algorithms do identify correctly the trees in the bottom right of these figures.

Figure 4.9: The RGB view of the scenario 'construction'

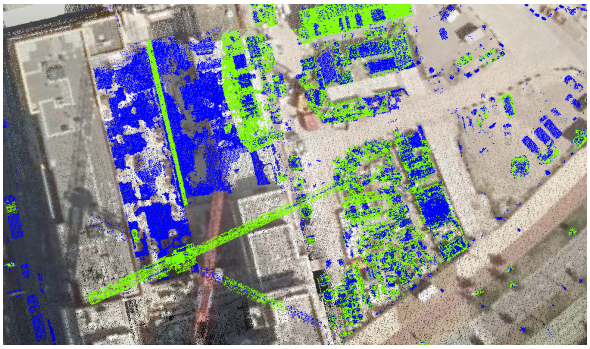


(a) The result of the Random Forest for the scenario 'construction' seen from a bird eye's view. Green points are trees blue points are non-trees.

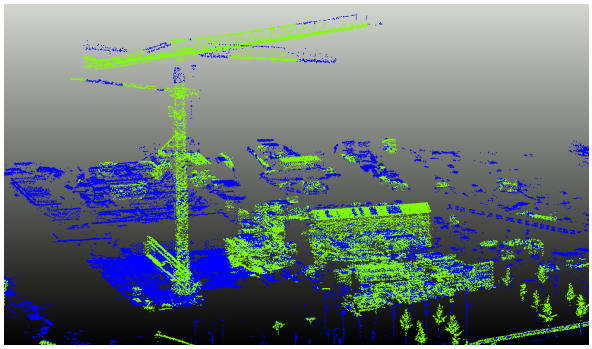


(b) The result of the Random Forest for the scenario 'construction' seen from a frog eye's view. Green points are trees blue points are non-trees.

Figure 4.10: Results for Random Forest for the scenario 'construction'.



(a) The result of the Neural Network for the scenario 'construction' seen from a bird eye's view. Green points are trees blue points are non-trees.



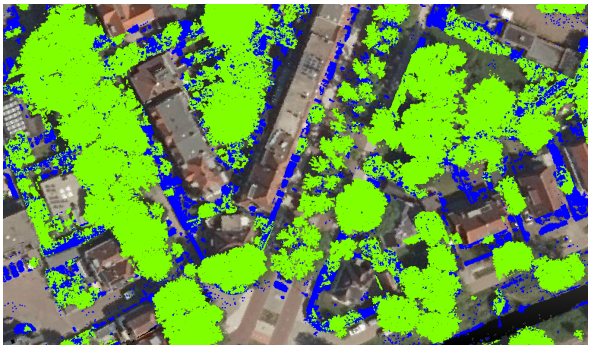
(b) The result of the Neural Network for the scenario 'construction' seen from a frog eye's view. Green points are trees blue points are non-trees.

Figure 4.11: Results for Neural Network for the scenario 'construction'.

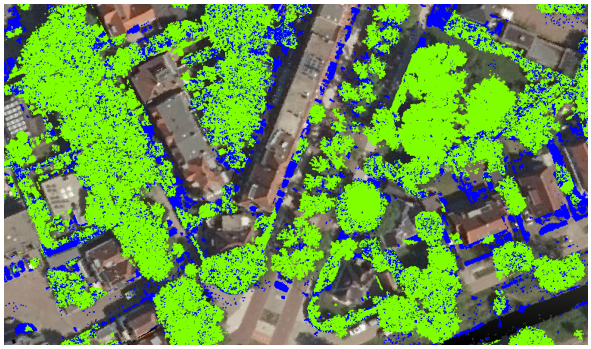
Trees

Scenario 'trees' can be seen in figures 4.12 and 4.13. In the RGB image, figure 4.12, there are red squares present which indicate trees maintained by the municipality of Delft. When looking at results presented by figure 4.13 it can be seen that the method for both algorithms is very adequate at recognizing trees. All the municipality trees have been correctly classified and even the trees not maintained by the municipality in the bottom right of the figures have been correctly classified. Looking at the difference between the Random Forest (figure 4.13b) and Neural Network (figure 4.13a) it can be seen that **the Random Forest classification is a bit more dense than the Neural Network.** What is meant by this is that for the Neural Network there are a lot of blue speckles (non-tree points) present within the green clusters (tree-points). The reason why there are still blue speckles present after DBSCAN for the Neural Network is probably because there were not enough tree points present above two metres to create full clusters of tree points. The expectation would be that within the tree there are no other non-tree points present, hence Random Forest is probably better classified.

Figure 4.12: The RGB view of the scenario 'trees', every square with a number indicates a tree maintained by municipality Delft.



(a) The result of the Random Forest for the scenario 'trees' seen from a bird eye's view. Green points are trees blue points are non-trees.



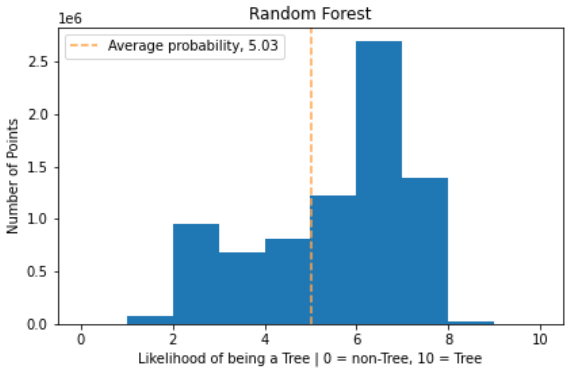
(b) The result of the Neural Network for the scenario 'trees' seen from a bird eye's view. Green points are trees blue points are non-trees.

Figure 4.13: Results for Random Forest and Neural Network for the scenario 'trees'.

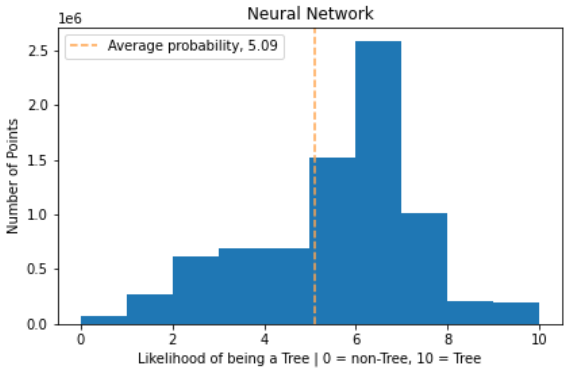
4.2.3. Scenarios: taking a look at the probabilities.

In this paragraph a closer look will be taken at places where the algorithm struggles and places which can help understand how the algorithm works. All the figures used in this paragraph give the probability for a point to be a tree point both for the Neural Network and Random Forest. The scenarios represented in section 4.2 will be used to showcase the probabilities both algorithms give to the different points.

In all scenarios shown and discussed in this section it can be seen that the Neural Network is always more definite about its classification than the Random Forest. This can be seen by that the Neural Networks scores lower and higher probabilities than the Random Forest. Figure 4.14 showcases this difference clearly where the Neural Network is more definite in its classification than the Random Forest. The reason why has to do with how the Neural Network and Random Forest are structured. The Neural Network consists of one predictor, while the Random Forest consists of 150. The Random Forest will use the average prediction of these 150 predictors which will give a more temperate classification than the one predictor from the Neural Network (Kathmann, Natijne, and R. C. Lindenbergh 2022).



(a) Histogram of the probability (0-10) for the Random Forest. 10 means the algorithm is 100% certain it is a tree point 0 means that the algorithm is 100% certain it is not a tree point.



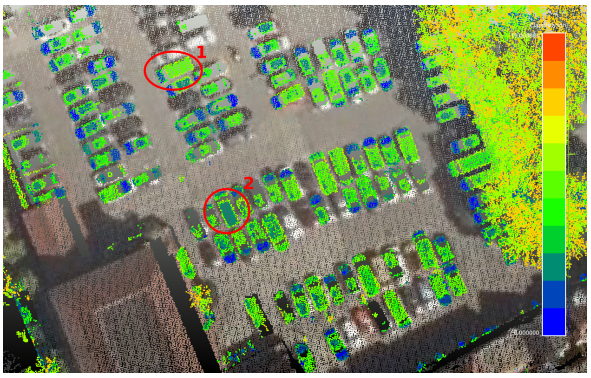
(b) Histogram of the probability (0-10) for the Neural Network. 10 means the algorithm is 100% certain it is a tree point 0 means that the algorithm is 100% certain it is not a tree point.

Figure 4.14: Histograms of the probability (0-10) for both algorithms.

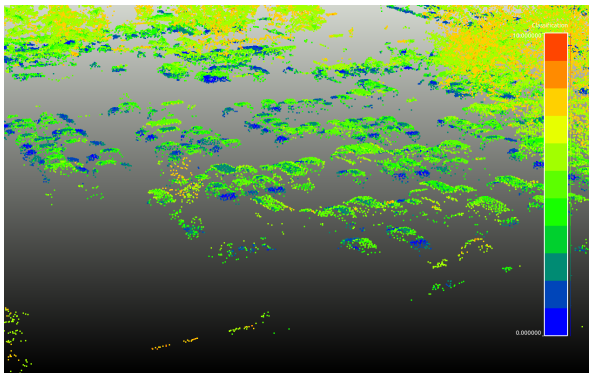
Cars

Starting at the scenario 'cars' which can be seen in figure 4.15 and figure 4.16, the RGB image can be seen in figure 4.6. The lighter the colour in these images the more likely the algorithm thinks it is a tree. It can be seen that the Neural Network is more convinced that the cars are trees then the random forest, which is evident in red circle 1 in figures 4.15a and 4.16a. However the Neural Network is also more convinced some cars are absolutely not a tree compared to the Random Forest, which is evident in red circle 2 in figures 4.15a and 4.16a. Overall the Neural Network gives a stronger prediction towards tree or non-tree than the Random Forest, which can be seen in figure 4.14. The reason why certain cars get predicted to be a tree or not a tree

might have to do with the curve of the surface. This is slightly visible in figures 4.15b and 4.16b, a further close-up is not made because due to the point density it will not give a clearer image. It is mainly based of that the hoods of the cars all get a higher chance to be a tree than the roofs, for example in red circle 2 in figure 4.16a the Neural Network is more convinced that the hood points of the car are tree points than the roof points of the car. It might be that if the surface is curved that the algorithm thinks it is a tree while if the surface is completely flat the algorithm thinks it is a not a tree. However for the Random Forest this is the other way around compared to the Neural Network. Looking at figure 4.15a in red circle 1 the Random Forest is convinced that the hood consists of non-tree points but the roof top does consist of tree points.

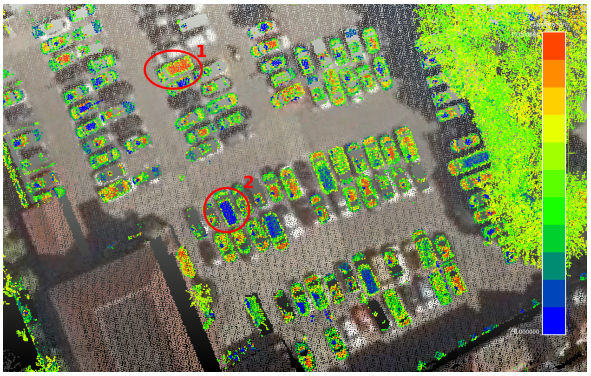


(a) The probabilities of the Random Forest for the scenario 'cars' seen from a bird eye's view. Red (10) means the algorithm is 100% certain it is a tree point blue (0) means that the algorithm is 100% certain it is not a tree point.

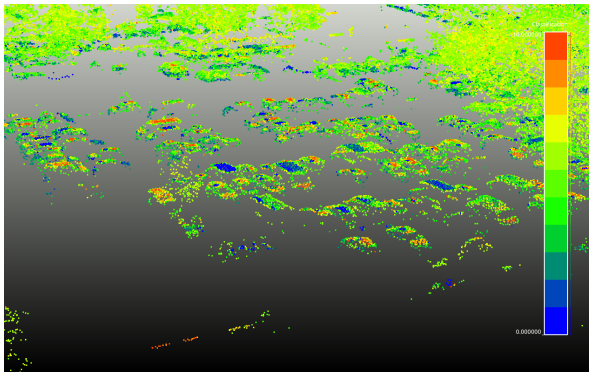


(b) The probabilities of the Random Forest for the scenario 'cars' seen from a frog eye's view. Red (10) means the algorithm is 100% certain it is a tree point blue (0) means that the algorithm is 100% certain it is not a tree point.

Figure 4.15: Probabilities for Random Forest for the scenario 'cars'.



(a) The probabilities of the Neural Network for the scenario 'cars' seen from a bird eye's view. Red (10) means the algorithm is 100% certain it is a tree point blue (0) means that the algorithm is 100% certain it is not a tree point.

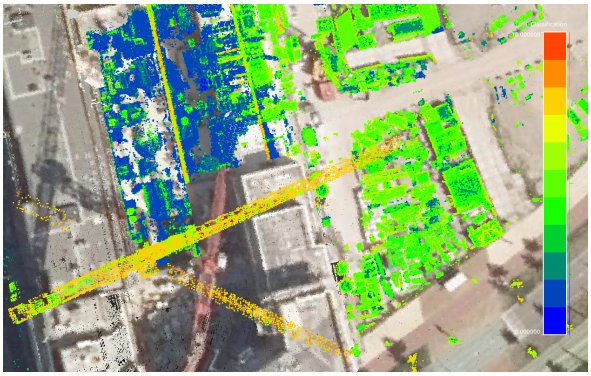


(b) The probabilities of the Neural Network for the scenario 'cars' seen from a frog eye's view. Red (10) means the algorithm is 100% certain it is a tree point blue (0) means that the algorithm is 100% certain it is not a tree point.

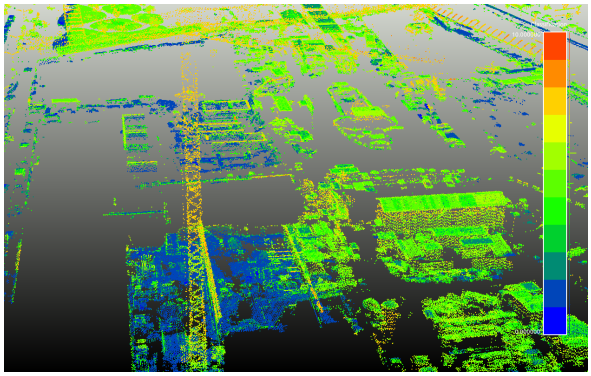
Figure 4.16: Probabilities for Neural Network for the scenario 'cars'.

Construction

The scenario 'construction' can be seen in figures 4.17 and 4.18, while the RGB image can be seen in figure 4.9. The main thing is that the Neural Network is more convinced that the construction site, especially the crane, is a tree whereas the Random Forest is more reserved about it. What is noticeable is that for the Neural Network the higher a point is the more likely the Neural Network finds it a tree point (see figure 4.18b), giving an indication that the Neural Network is biased towards the 'delta Z' feature. Since this feature is always present in the Neural Network and not in the Random Forest, because the Random Forest has the square root of the number of features as the number of splits so it will not include all features.

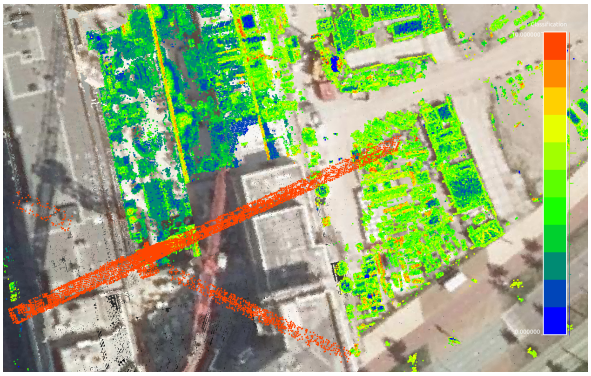


(a) The probabilities of the Random Forest for the scenario 'construction' seen from a bird eye's view. Red (10) means the algorithm is 100% certain it is a tree point blue (0) means that the algorithm is 100% certain it is not a tree point.

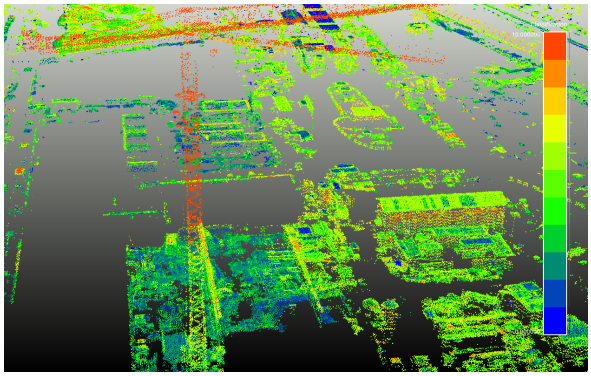


(b) The probabilities of the Random Forest for the scenario 'construction' seen from a frog eye's view. Red (10) means the algorithm is 100% certain it is a tree point blue (0) means that the algorithm is 100% certain it is not a tree point.

Figure 4.17: Probabilities for Random Forest for the scenario 'construction'.



(a) The probabilities of the Neural Network for the scenario 'construction' seen from a bird eye's view. Red (10) means the algorithm is 100% certain it is a tree point blue (0) means that the algorithm is 100% certain it is not a tree point.

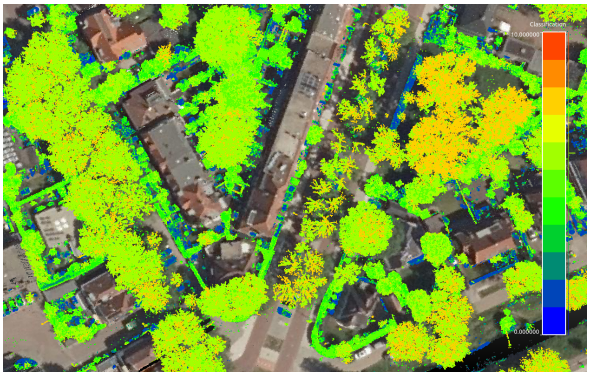


(b) The probabilities of the Neural Network for the scenario 'construction' seen from a frog eye's view. Red (10) means the algorithm is 100% certain it is a tree point blue (0) means that the algorithm is 100% certain it is not a tree point.

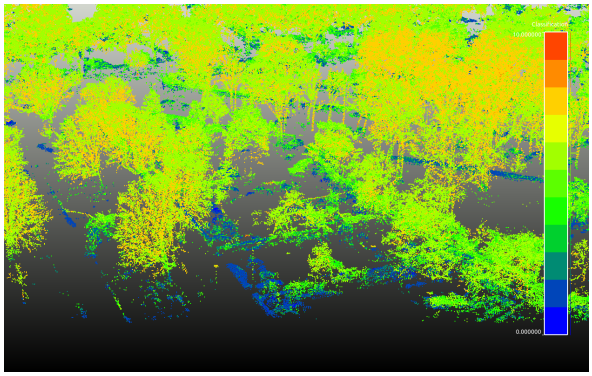
Figure 4.18: Probabilities for Neural Network for the scenario 'construction'.

Trees

The scenario 'trees' is displayed in figures 4.19 and 4.20, the RGB image can be seen in figure 4.12. This figure simply shows that both the Neural Network and Random Forest are both quite certain over what is a tree and not a tree in a case where there are no cars or construction present. It again shows in figure 4.20b that the Neural Network shows a prejudice towards height since the higher a points is the more certain the Neural Network is that it is a tree point.

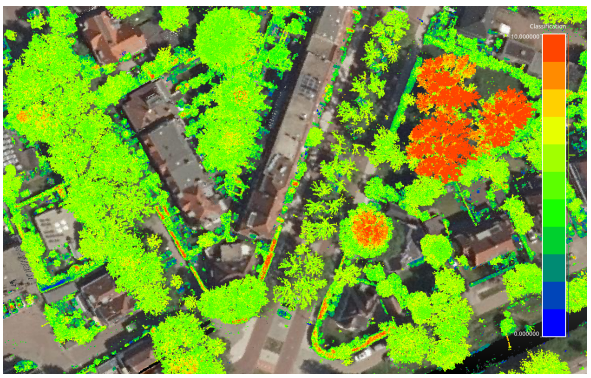


(a) The probabilities of the Random Forest for the scenario 'trees' seen from a bird eye's view. Red (10) means the algorithm is 100% certain it is a tree point blue (0) means that the algorithm is 100% certain it is not a tree point.

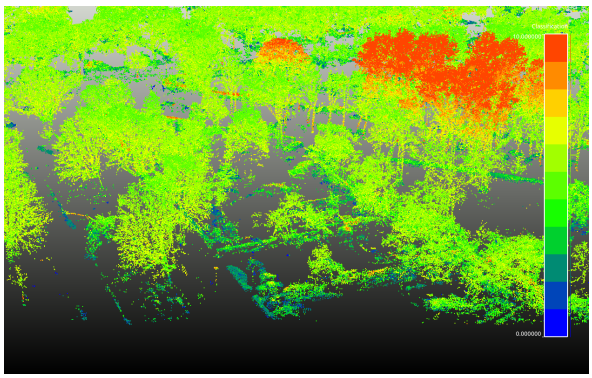


(b) The probabilities of the Random Forest for the scenario 'trees' seen from a frog eye's view. Red (10) means the algorithm is 100% certain it is a tree point blue (0) means that the algorithm is 100% certain it is not a tree point.

Figure 4.19: Probabilities for Random Forest for the scenario 'trees'.



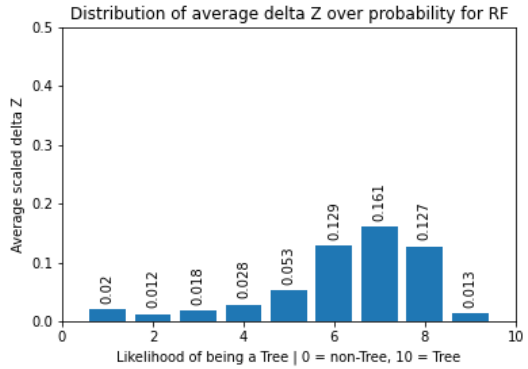
(a) The probabilities of the Neural Network for the scenario 'trees' seen from a bird eye's view. Red (10) means the algorithm is 100% certain it is a tree point blue (0) means that the algorithm is 100% certain it is not a tree point.



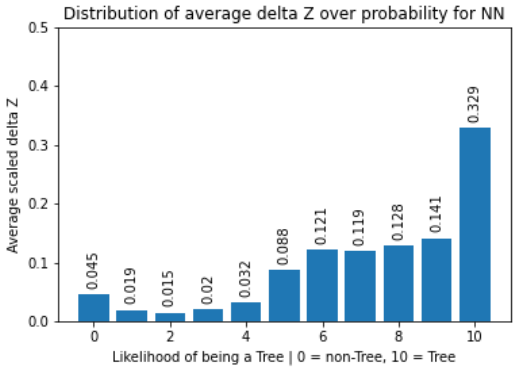
(b) The probabilities of the Neural Network for the scenario 'trees' seen from a frog eye's view. Red (10) means the algorithm is 100% certain it is a tree point blue (0) means that the algorithm is 100% certain it is not a tree point.

Figure 4.20: Probabilities for Neural Network for the scenario 'trees'.

To showcase the dependence of the Neural Network towards height compared to the Random Forest, figure 4.21 is created. This figure showcases the average scaled (from 0 to 1) delta Z compared to the probability. The probability ranges from 0 to 10 where 0 is certainly no tree point and 10 certainly is a tree point. The figure clearly shows that the average scaled delta Z is higher for a higher probability for the Neural Network (figure 4.21b) compared to the Random Forest (figure 4.21a).



(a) Histogram of the scaled average delta Z per probability (0-10) for the Random Forest. 10 means the algorithm is 100% certain it is a tree point 0 means that the algorithm is 100% certain it is not a tree point.



(b) Histogram of the average scaled delta Z per probability (0-10) for the Neural Network. 10 means the algorithm is 100% certain it is a tree point 0 means that the algorithm is 100% certain it is not a tree point.

Figure 4.21: Histograms of the scaled (from 0 to 1) delta Z against the probability for both algorithms.

4.3. Other results

The two other important scenarios that were tried to improve the algorithm was using all the classes in AHN4 instead of just the 'other' class so this includes the entire dataset of the area AHN4 has scanned instead of just the unclassified points located in the 'other' class. Removing samples with high loss was also tried as an option to improve the algorithm, the resulting accuracies for the scenarios can be seen in figure 4.22.

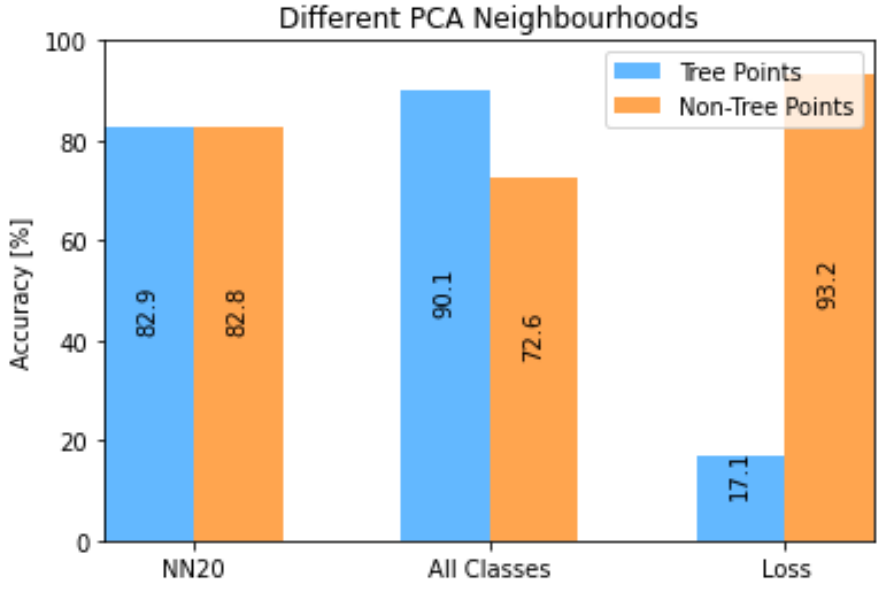


Figure 4.22: Accuracy for the following scenarios: The PCA scenario used (NN20), the scenario with all class from AHN4 (all classes) and the scenario with the highest loss removed (loss).

What is immediately noticeable is that the 'loss' scenarios perform worse for classifying trees then the scenario used, the accuracy actually dropped dramatically for tree points. Probably because the algorithm requires these high loss points to accurately define tree points. Since these points cause for the highest difference in predicted and actual classification is it probable that these points contain the information necessary for the Neural Network to properly classify trees. For identifying non-trees it works better but it does not weigh up against the loss of accuracy in the classification of trees. All classes works better at classifying trees but worse at non-trees compared to the scenario used. It was opted to keep using the original scenario since the overall accuracy is around the same and the extra computation time it takes to classify all the classes in AHN4 was not deemed worth it for a probably just as good or worse classification than the current method

used. However the advantage of this scenario is that it is a better representation of how the algorithm would work for non-AHN4 datasets, since other datasets will not have the same classes already present as AHN4.

5

Discussion

In this chapter the different methods and results of this thesis will be discussed and explained. The goal is to give an explanation of why a certain eventual method works best. The topics that will be discussed include the training data, feature engineering, Neural Network and Random Forest. Omissions and Commissions will discuss where an algorithm goes wrong. State of the art will compare the algorithm introduced in this thesis to similar algorithms already developed. And finally the future prospects of the method developed in this thesis will be discussed the section tree inventories.

5.1. Training Data

A good machine learning algorithm starts with good training data. The training data used in this study contained both false positives and false negatives (see section 3.1.3. This was due to how the training data was made, namely the location of the trees maintained by the municipality of Delft was selected in the AHN4 data and a radius around that location was drawn to select tree points. This has the disadvantage that points that are not trees are selected as trees and that points that are trees, but fall outside the radius, are selected as non trees. Another downside is that the trees which are not maintained by the municipality, such as the trees in the gardens, are not included in the training data. In the end this results in training data which contains false positives and negatives. This can be improved by hand selecting training data. However it will take a large amount of time to hand select a dataset which has the same diversity and number of points as the one created by combining the tree database from the municipality of Delft and AHN4. Yes, hand-selected data will probably give an increase in performance however to get the number of points and diversity that is necessary to have a stable algorithm will take a very long time. Another way to try to improve the training data is by combining AHN4 with another tree dataset which might be more accurate than the tree database from Delft municipality. This other database will have to contain the shapes of the trees since this is the main thing that the Delft database lacks. So far such a database does not exist. By knowing the shape of the tree the false positives and false negatives created by selecting a radius around the geo-location of the tree will be eliminated. It is possible to use another already classified aerial LiDAR database. Examples of already classified LiDAR datasets could be from Luxembourg (*Luxembourg LiDAR* n.d.) and from Slovenia (*Slovenia LiDAR* n.d.). However it is very unlikely with such a database that it has the same properties as AHN4, such as point density and the same type of point features.

5.2. Feature Engineering

A vital part of the method used in this thesis to identify trees is the feature engineering. The feature engineering consists of creating spatial features for a point based upon the point's neighbourhood. AHN4 with its current point density gives the best representation to a point with the 20 nearest neighbours in 3D out of the neighbourhoods tried in this thesis, see section 3.2.1. No more nearest neighbours were tried because 20 was already computationally intensive and more neighbours would increase the computation time to much to be able to run the algorithm on a normal computer. The other two methods that were tried, besides 3D nearest neighbours, are the 2D neighbourhood of 20 neighbours ,which did not work well probably because a 3D element is necessary to give an accurate representation of a points spatial attributes, and a circular neigh-

bourhood. The reasoning behind the circular neighbourhood was because trees generally have a circular shape so the hope would be that this would give tree points very distinct spatial attributes. However it ended up giving a very scattered classification of the data (see figure 4.3). Maybe the neighbourhood was too large and included too many points, but going for a smaller radius the issue occurred that a lot of points did not have any neighbours or only very little.

Another option that was tried was to apply PCA over the already created features to save on computation time and possibly improve the performance of the algorithm. The goal is to reduce the number of features but preserve the variability that these features represent. This reduces computation time since less features are used and can possibly increase accuracy since some algorithms perform better with less features. Indeed, computation time was reduced, but performance decreased as well. Since the performance decreased and the running of the neural network or random forest is not the computationally intensive part of the algorithm, and this is what the reduced features affect, it was chosen to not use it.

In the end the features that were used are the ones that show a clear difference in value between tree and non-tree points. The verticality factor is delta Z in the features used. Delta Z has a large effect on the results, as can be seen from the probability figures (see section 4.2.3). This indicates that the trees probably have a large difference in Delta Z compared to the other points present in the data. It is logical that the attributes that describe spatial attributes of the points, Curvature, Planarity, Linearity, Scattering, and Change of Curvature, help to distinguish tree points. Since trees have a very distinct shape compared to other types of objects present such as cars, houses, etc. Omnivariance describes the distribution of points across a 3D space, Anistropy describes how direction dependent a point is, since trees have an irregular distribution it is again logical that this helps with classification of trees. Number of Returns, Amplitude, Reflectance and Intensity are all dependent upon the laser scanners properties. Each show a clear difference for tree and non-tree points which probably has to do again with the irregular shape of trees, the same as for Anistropy and Omnivariance. In all the irregular shape of the trees give probably the most clear difference in feature values for tree and non-tree points. However the Neural Network gives a clear indication that verticality features are important for the classification seeing how much of a preference it gives to height (figure 4.21). The echo ratio could be an example of another verticality feature to add (Höfle et al. 2009), which is defined as the ratio between the number of points found in 3D for a fixed radius from a point X and the number of points found in 2D for the same radius from the same point X.

5.3. Neural Network

Starting with discussing the first classification algorithm used, the Neural Network. For training the Neural Network multiple factors play a role, the first one to discuss is the distribution of the training data. The amount of tree and non-tree points present in the training data. The 50/50 split was chosen to have the best balance in the classification of both tree and non-tree points and overall accuracy. Other splits favoured the classification of the point type with the larger split. Another way that was tried to improve the training strategy was by removing samples with high loss. Loss is the metric with which the Neural Network trains itself and is similar to an error calculation between the predicted classification and the actual classification. However removing the couple of points with high loss resulted in no significant change in the data, even caused a significant decrease in accuracy, and was ruled as an useless option to improve the training data. The reason might be that the algorithm requires these 'difficult' points with high loss to properly learn what a tree point is. What an option is which was not tried is to remove the points with low loss instead, this might improve the accuracy, since these might be the points where the algorithm does not learn from. Finally using all the classes in AHN4 instead of just the 'other' class were used for the training data, the reasoning being that the algorithm might identify tree points easier with more data. However it does increase computation time since the amount of points increases at least fourfold. It also ended up being less accurate, for tree points, than the chosen method of using just the 'other' class.

In the structure of the Neural Network the activation function ReLU (figure 2.2b) has been used for six layers and the activation function Sigmoid (figure 2.2a) for one. For the first six layers a ReLU function is probably the best, ReLU is the most used activation function for Neural Networks and works well for simple classification problems which the problem in this thesis qualifies under. For the final classification layer an activation function which gives a value between 0 and 1 is necessary, for this the Sigmoid function is used in this thesis. Again Sigmoid is probably the best activation function for this since an output between 0 and 1 is required. If more classes are added to the algorithm this should probably be changed to the Softmax activation func-

tion which is basically multiple Sigmoid functions stacked together and capable of giving probabilities for the different classes.

5.4. Random Forest

The other algorithm used instead of the Neural Network was a Random Forest. The two main parameters that could be adjusted in the Random Forest are the maximum depth and the number of trees. The number of splits per node was kept at the square root of the number of features as suggested by Biau and Scornet 2016. In the end an algorithm with the setup of a quite shallow maximum depth, 10, and a lot of trees, 150, worked best. A large amount of trees corresponds to literature (Biau and Scornet 2016). The reason behind this is probably because it prevents overfitting and prevents that the algorithm gets to dependent upon one variable. This is because for each tree the start is randomly selected so if the tree is shallow than the tree does not have enough nodes to home in on the most defining parameter of the point cloud for recognizing trees. If the tree ends up having a lot of nodes it is possible that all the trees will home in on the same defining feature which can result in a lower accuracy. What can be taken away from this is that for recognizing trees it is probably better to not depend to much on one feature but take all features equally into account.

5.5. Omissions and Commissions

So first off the algorithm with both Random Forest and the Neural Network work really well in identifying trees after the DBSCAN, DBSCAN removes a lot of the noise present in the data otherwise. The main thing where the algorithm goes wrong is in false positives, since false negatives rarely occur. Both the Random Forest and Neural Network run into the same problem cases, namely cars and construction. Apparently some cars show the same attributes for the algorithm as trees. This is for cars higher than 2 metres, since else it does not pass the height limit in DBSCAN. Since for some cars the algorithm is absolutely convinced that the car is not a tree while for other cars this is the other way around. The reason why there is a difference is probably due to different shapes of cars which result in different values for certain features such as curvature and planarity. One more explanation could be the point density. AHN4 is obtained through an aerial LiDAR but this does not scan every surface perpendicular to the scanner. The surfaces right below the scanner when flying over have a slightly higher point density than the surfaces just a bit to the side of the scanner. This results in a slight difference in point density and might explain why certain cars do get classified as trees and others do not. Another problem case were construction sites, what threw the algorithm off here was probably the randomness of the construction sites which is very similar to the structure of trees which is also a random conglomeration of points. A way this might be fixed is by adding these problems scenarios, cars and construction sites, as classes to the classification.

The main difference between the Neural Network and Random Forest was the Neural Network's dependence upon height. This is probably because the height turns out to be the most defining feature of the trees. The reason why only the Neural Network hones in on this is because the Random Forest works with the principle of trees (predictors) and leaves (nodes) for its algorithm, so the Random Forest is more random in its selection of features for classification than the Neural Network. Another difference is the certainty of a point being a tree or not a tree point, the Neural Network is a lot more certain of the classification of a point than the Random Forest is. The reason behind this is the same reason as to why the Random Forest is not so biased to height and that is because of the randomness of the feature selection for classification introduced in the Random Forest due to the amount of trees and lack of leaves.

5.6. State of the Art

In this paragraph the algorithm will be compared to similar methods to see how its performance stacks up. First comparing it to the method of Nurunnabi et al. 2021 by which the method in this thesis was inspired. Taking the best performing method from this thesis, which is the Random Forest, it scored an accuracy of 72.4% against validation dataset 2. Keep in mind that this dataset consists of problem scenarios for this algorithm so the accuracy of this validation dataset will probably be lower than comparing to a standard dataset which does not consist of purely problem scenarios. When comparing it to validation dataset 1 the accuracy was 94.4%. The method of Nurunnabi et al. 2021 scores an accuracy of 97% and was used for extracting ground points. This is higher than both accuracies scored by the method used in this thesis. The similarity between the method used by Nurunnabi et al. 2021 and the method used in this thesis is that both are a non-end-to-end method which first creates features for its points which are than fed into a classification algorithm

and both make use of an aerial LiDAR dataset. It seems than from the results of Nurunnabi et al. 2021 that this type of structure for a classification algorithm works better for ground points. However this could also be an implementation issue and that with some tweaking the method used in this thesis performs as well or better than the method from Nurunnabi et al. 2021. Man et al. 2020 used a combination of hyperspectral data and LiDAR data to segment trees and grasses in an urban area and scored an accuracy of upto 99%. This obviously outperforms the method used in this thesis and can be an indication that using a combination of spectral data and LiDAR data might be the best method to segment trees in an urban area as supported by K. Wang, T. Wang, and Liu 2018. However this method does require an extra dataset which introduces extra errors that can occur due to a lack of quality (such as low resolution or clouds) in the dataset. In the end the method used in this thesis does not perform as well as a combination of spectral data and LiDAR data but it also requires one dataset less.

5.7. Tree inventories

To be able to use the method developed in this thesis the problems with the classification of cars and construction sites has to be resolved first before the method can be used. Assuming these issues can be fixed the method offers prospects for the segmenting and counting of trees in AHN4, especially urban areas since so far it has only be tested on urban areas. The next step after this would be to develop tree properties such as height, volume, max diameter and Diameter at Breast Height (DBH). Whether this is actually possible is dependent on the point density of AHN4, so whether there are enough points to develop a correct 3D image of the tree to be able to get the properties of the tree. Especially since AHN4 is from aerial LiDAR it might lack the number of points on ground level to develop for example the DBH. Since the canopy might be in the way of scanning these ground points. However counting the number of trees present in an urban area, and possibly in a non urban area, and the size of the canopies of trees should be possible when the current issues with the algorithm are fixed.

6

Conclusion and Recommendations

In this section the research question and subquestions will be answered. After which recommendations will be given to possibly improve the method developed in this thesis. The subquestions will be answered as follows.

6.1. Research Question and Subquestions

- *How can the performance of the methods be measured?*

  To measure the exact performance of the method is difficult since there is no perfectly classified dataset to use. Two by hand classified datasets were used in the end, one smaller one to help optimize the algorithm and one that better reflects the difficulties that the algorithms faces to give a good distinction between the performance of the Neural Network and Random Forest. Scenarios are also a good way to get some better insight into the performance of the algorithms combined with looking at the probability of a point to be a tree point.

- *What effect do the features have on the algorithm?*

  It appears that features which give some sort of representation of the irregular shape of trees help the most with classification. The values of the engineered features are dependent upon the neighbourhood which is used in the PCA to engineer them. The neighbourhood definitely had a great effect on the performance of the algorithm and whether it was better at classifying tree or non-tree points. In the end a neighbourhood was used which had a good balance in classifying tree and non-tree points, this was the 20 nearest neighbours in 3D space.

- *What is the effect of the training strategy on the outcome of the different algorithms?*

  Training strategy highly influenced the result for both algorithms. In both cases multiple training strategies were tried to find the best working one. For both algorithms what worked best was a training strategy that would still employ as much data as possible to learn from but without overfitting. Reason is that the training data contained false positives and negatives (see section 3.1.3 which makes the algorithm prone to overfitting.

- *With what kind of cases do the algorithms struggle?*

  The algorithm seems to struggle the most with irregular shaped areas such as construction sites and vehicles. This is probably due to trees also having irregular shapes which causes similar properties as with the problem cases.

- *What is the difference between using a feed forward Neural Network and a Random Forest algorithm in a non-end-to-end method?*

  Performance wise the Random Forest works slightly better, especially in recognizing non-tree points, for the currently used method. This might be due to the fact that Random Forest is better at dealing with corrupt training data than the Neural Network. There is a chance that the Neural Network will

outperform the Random Forest if it is fed perfect training data. The Neural Network was also shown to be more height dependent than the Random Forest, which is probably due to the structure of the algorithms. Random Forest namely contains multiple shallow predictors which makes it less likely to show a preference to a certain variable than the Neural Network which consists of one big predictor.

Answering the research question of what is an effective way of segmenting trees from AHN4? The method chosen here was a non-end-to-end method that engineered features per point to pass through an algorithm which does the classification and followed by a DBSCAN step for the segmentation and noise filtering. This proved to be an effective method with the main issue being false positives such as cars and construction sites. Two classification algorithms were used for this namely a Random Forest and a Neural Network. In general the algorithms behaved almost identically except that Random Forest was slightly more accurate. The main difference could be found in the dependence that the Neural Network gave to the height, Random Forest did not have this due to how the algorithm is build. Random Forest is the better algorithm for this problem with the currently used training data since it does not give such a preference to height compared to the Neural Network (see figure 4.21) and is slightly more accurate. There is a chance that if the training data gets improved that the Neural Network outperforms the Random Forest. Keep in mind that a Neural Network is sensitive to corrupt training data and as such under-performs compared to when it would be fed perfect training data. However Random Forest will also improve with perfect training data so whether the Neural Network will definitely outperform it in this case is unsure. What is a certainty is that with the current training data Random Forest out-performs the Neural Network. A training strategy was used that consisted of a 50/50 split in tree and non-tree points and a not to large amount of points in total to prevent overfitting combined with a Random Forest setting of a lot of trees (predictors) with a small amount of leaves (nodes). In short the algorithm described in this thesis works well and rarely has omission errors but has some false positive issues with things such as cars and construction sites.

6.2. Recommendations

In this section recommendations will be given which can possibly improve the result of the algorithm.

One of the ways the algorithm could possibly be improved is by introducing a cars class, since this is the object with which the algorithm has the most issues. Giving it a separate class might fix this issue. This can then also be done for other problem cases such as constructions sites. Adding more features might also make it easier for the algorithm to recognize trees. Especially verticality features, since at the moment only delta Z is used and the Neural Network has shown that height is very important for its classification. An example of such a feature could be the echo ratio (Höfle et al. 2009), the echo ratio should give a high value for buildings and a low value for vegetation.

Another way the algorithm could be improved is by providing it better training data, since the training data used contained both false positives and false negatives. This can be done by either hand-crafting the training data or possibly finding another dataset which contains better classified data than the ones used. However a dataset which contains better classified data and displays the same properties as AHN4 (such as point density) has not been found. Training data can also be improved by using spectral data. There are multiple datasets that provide tree locations from spectral data such as Beery and Huang 2022 and the dutch *Boomregister* 2022. A combination of the locations of these trees and AHN4 could possibly give a more accurate training dataset than the one currently used. The method of Man et al. 2020, which uses a combination of spectral and LiDAR data, can also be used to create training data. To than use the training data for the method used in this thesis so the spectral data is not necessary anymore once the algorithm is trained.

The training strategy might be improved by using an adaptive neighbourhood. This is a method where the neighbourhood of the point is dependent upon local features of the area, such as point density. If this will be effective or can even be applied to this problem further research will have to be conducted. The reasoning behind why this might improve the result is because different neighbourhoods are better at classifying tree or non-tree points. Combining these different neighbourhoods for different situations might then give an overall better result. Another neighbourhood that can be tried is a spherical neighbourhood with a fixed radius, according to Biau and Scornet 2016 this is the best option for classification using the features used in this thesis.

Bibliography

AHN (2021). URL: https://www.ahn.nl/ahn-4 (visited on 10/15/2021).

Beery, S and J Huang (2022). *Mapping Urban Trees Across North America with the Auto Arborist Dataset*. URL: https://ai.googleblog.com/2022/06/mapping-urban-trees-across-north.html?m=1&s=09.

Biau, Gérard and Erwan Scornet (2016). "A random forest guided tour". In: *Test* 25.2, pp. 197–227. ISSN: 11330686. DOI: 10.1007/s11749-016-0481-7.

Boomregister (2022). URL: https://boomregister.nl/ (visited on 07/15/2022).

Breiman, L (2001). "Random Forests". In: *Machine Learning* 45, pp. 5–32.

Bro, Rasmus and Age K. Smilde (2014). "Principal component analysis". In: *Analytical Methods* 6.9, pp. 2812–2831. ISSN: 17599679. DOI: 10.1039/c3ay41907j.

Chen, Ziyue, Bingbo Gao, and Bernard Devereux (2017). "State-of-the-art: DTM generation using airborne LIDAR data". In: *Sensors (Switzerland)* 17.1. ISSN: 14248220. DOI: 10.3390/s17010150.

Daszykowski, M. and B. Walczak (2009). "Density-Based Clustering Methods". In: *Comprehensive Chemometrics* 2, pp. 635–654. DOI: 10.1016/B978-044452701-1.00067-3.

DBSCAN (n.d.). URL: http://www.sthda.com/english/wiki/wiki.php?id_contents=7941.

Delft, Gemeente (2021). URL: https://data.overheid.nl/dataset/bomen-in-beheer-door-gemeente-delft#panel-resources (visited on 10/13/2021).

GmbH, RIEGL Laser Measurement Systems (2017). "LAS Extrabytes Implementation in RIEGL Software". In:

Halko, N, P. G. Martinsson, and J. A. Trop (2009). *Finding structure with Randomness: Stochastic algorithms for constructing approximate matrix decompositions*. Tech. rep. California Institute of Technology.

Hang, Su et al. (2015). "Multi-view Convolutional Neural Networks for 3D Shape Recognition". In: *Proceedings of the IEEE International Conference on Computer Vision, 2015*, pp. 945–953.

Herrero-Huerta, Mónica, Roderik Lindenbergh, and Pablo Rodríguez-Gonzálvez (2018). "Automatic tree parameter extraction by a Mobile LiDAR System in an urban context". In: *PLoS ONE* 13.4, pp. 1–23. ISSN: 19326203. DOI: 10.1371/journal.pone.0196004.

Höfle, Bernhard et al. (2009). "Detection of building regions using airborne LiDAR – A new combination of raster and point cloud based GIS methods Study area and datasets". In: *GI_Forum 2009 - International Conference on Applied Geoinformatics*, pp. 66–75. URL: http://www.agit.at/php_files/myagit/papers/2009/7504.pdf.

Itakura, Kenta and Fumiki Hosoi (2018). "Automatic individual tree detection and canopy segmentation from three-dimensional point cloud images obtained from ground-based lidar". In: *Journal of Agricultural Meteorology* 74.3, pp. 109–113. ISSN: 18810136. DOI: 10.2480/agrmet.D-18-00012.

Kathmann, H. S., A. L. van Natijne, and R. C. Lindenbergh (2022). "Probabilistic Vegetation Transitions in Dunes By Combining Spectral and Lidar Data". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLIII-B2-2.June, pp. 1033–1040. DOI: 10.5194/isprs-archives-xliii-b2-2022-1033-2022.

Lay, D. C., S. R. Lay, and J. J. McDonald (2016). *Linear Algebra and its Applications*. Pearson. ISBN: 1-292-09223-8.

Li, Jintao et al. (2021). "An Over-Segmentation-Based Uphill Clustering Method for Individual Trees Extraction in Urban Street Areas from MLS Data". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*. ISSN: 21511535. DOI: 10.1109/JSTARS.2021.3051653.

Li, Shihua et al. (2017). "Estimating Leaf Area Density of Individual Trees Using the Point Cloud Segmentation of Terrestrial LiDAR Data and a Voxel-Based Model". In: *Remote Sensing* 9.11, p. 1202. ISSN: 2072-4292. DOI: 10.3390/rs9111202.

Loukas, S (n.d.). URL: https://towardsdatascience.com/how-do-random-forests-decision-trees-decide-simply-explained-with-an-example-in-python-6737eb183604.

Luxembourg LiDAR (n.d.). URL: https://data.public.lu/en/datasets/lidar-2019-releve-3d-du-territoire-luxembourgeois/.

Man, Qixia et al. (2020). "Automatic Extraction of grasses and individual trees in urban areas based on airborne hyperspectral and LiDAR data". In: *Remote Sensing* 12.17, pp. 1–22. ISSN: 20724292. DOI: 10.3390/RS12172725.

Maturana, Daniel and Sebastian Scherer (2015). "VoxNet: A 3D Convolutional Neural Network for real-time object recognition". In: *IEEE International Conference on Intelligent Robots and Systems* 2015-Decem, pp. 922–928. ISSN: 21530866. DOI: 10.1109/IROS.2015.7353481.

Natijne van, Adriaan (2021). URL: https://geotiles.nl (visited on 11/15/2021).

Nurunnabi, A. et al. (2021). "An efficient deep learning approach for ground point filtering in aerial laser scanning point clouds". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives* 43.B1-2021, pp. 31–38. ISSN: 16821750. DOI: 10.5194/isprs-archives-XLIII-B1-2021-31-2021.

Peyghambarzadeh, S. M.Moein et al. (2020). "Point-PlaneNet: Plane kernel based convolutional neural network for point clouds analysis". In: *Digital Signal Processing: A Review Journal* 98, p. 102633. ISSN: 10512004. DOI: 10.1016/j.dsp.2019.102633. URL: https://doi.org/10.1016/j.dsp.2019.102633.

Qi, Charles R. et al. (2017). "PointNet: Deep learning on point sets for 3D classification and segmentation". In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-Janua, pp. 77–85. DOI: 10.1109/CVPR.2017.16.

Rizaldy, A. et al. (2018). "FULLY CONVOLUTIONAL NETWORKS for GROUND CLASSIFICATION from LIDAR POINT CLOUDS". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 4.2, pp. 231–238. ISSN: 21949050. DOI: 10.5194/isprs-annals-IV-2-231-2018.

Seidel, Dominik et al. (2021). "Predicting Tree Species From 3D Laser Scanning Point Clouds Using Deep Learning". In: 12.February, pp. 1–12. DOI: 10.3389/fpls.2021.635440.

Shen, Yanyao and Sujay Sanghavi (2019). "Learning with bad training data via iterative trimmed loss minimization". In: *36th International Conference on Machine Learning, ICML 2019* 2019-June, pp. 10075–10097.

Shen, Yiru et al. (2017). "Mining point cloud local structures by kernel correlation and graph pooling". In: *arXiv*, pp. 4548–4557. ISSN: 23318422.

sklearn (n.d.). URL: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

Slovenia LiDAR (n.d.). URL: http://gis.arso.gov.si/evode/profile.aspx?id=atlas_voda_Lidar@Arso&culture=en-US.

Thomas, Hugues et al. (2019). "KPConv: Flexible and deformable convolution for point clouds". In: *Proceedings of the IEEE International Conference on Computer Vision* 2019-Octob, pp. 6410–6419. ISSN: 15505499. DOI: 10.1109/ICCV.2019.00651.

Wang, Chu, Marcello Pelillo, and Kaleem Siddiqi (2019). "Dominant set clustering and pooling for multi-view 3D object recognition". In: *arXiv*, pp. 1–12. ISSN: 23318422.

Wang, Kepu, Tiejun Wang, and Xuehua Liu (2018). "A review: Individual tree species classification using integrated airborne LiDAR and optical imagery with a focus on the urban environment". In: *Forests* 10.1, pp. 1–18. ISSN: 19994907. DOI: 10.3390/f10010001.

Weinmann, Martin et al. (2015). "Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 105, pp. 286–304. ISSN: 09242716. DOI: 10.1016/j.isprsjprs.2015.01.016. URL: http://dx.doi.org/10.1016/j.isprsjprs.2015.01.016.

Xia, Shaobo et al. (2021). "Point cloud inversion: A novel approach for the localization of trees in forests from tls data". In: *Remote Sensing* 13.3, pp. 1–10. ISSN: 20724292. DOI: 10.3390/rs13030338.

Yan, Wanqian et al. (2020). "A self-adaptive mean shift tree-segmentation method using UAV LiDAR data". In: *Remote Sensing* 12.3, pp. 1–14. ISSN: 20724292. DOI: 10.3390/rs12030515.

Zhang, Caiyun, Yuhong Zhou, and Fang Qiu (2015). "Individual tree segmentation from LiDAR point clouds for urban forest inventory". In: *Remote Sensing* 7.6, pp. 7892–7913. ISSN: 20724292. DOI: 10.3390/rs70607892.