TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

# Classification of 3D Point Clouds using Deep Neural Networks

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Geodäsie und Geoinformation**

eingereicht von

**Lukas Winiwarter**
Matrikelnummer 01325083

ausgeführt an der Forschungsgruppe Photogrammetrie (E120.7)
am Department für Geodäsie and Geoinformation
der Fakultät für Mathematik und Geoinformation der Technischen Universität Wien

Betreuung
Univ.Prof. Dipl.-Ing. Dr.techn. Norbert Pfeifer
Dipl.-Ing. Dr.techn. Gottfried Mandlburger

Wien, 26. Juli 2018

(Unterschrift Verfasser) (Unterschrift Betreuer)

## Erklärung zur Verfassung der Arbeit

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 26. Juli 2018

Lukas Winiwarter

## Acknowledgements

I would like to take this opportunity to extend my gratitude to several people involved in the creation of this thesis and beyond.

First, my thanks go to the University of Stuttgart and the staff at the Institute of Photogrammetry, where I was given the opportunity to conduct research on *PointNet* and *PointNet++* during a stay in February and March 2018. Especially, I thank Prof. Uwe Sörgel for the warm welcome and Stefan Schmohl, Dominik Laupheimer and Patrick Tutzauer for all the help that I received concerning neural networks. It was very valuable for me to talk to experienced developers about hyperparameters of neural networks as well as interpretation of the results.

Of course, I also want to thank my colleagues at TU Wien, especially the Research Group Photogrammetry lead by Prof. Norbert Pfeifer. Michael Wimmer and Moritz Bruggisser always had an open ear for my concerns and provided valuable and recent experiences on diploma theses. Thanks go also to everyone who participated in the discussion following my presentation at the Photo Circle.

My mentor and supervisor, Gottfried Mandlburger, had the idea a research visit to Stuttgart, and provided support whenever needed. It is always a pleasure to work with him, and I would like to thank him for all the inspiration I got.

Aside from academia, my friends and family provided the necessary emotional support. Claudio Navacchi, Manuel Schmitzer and Jakob Thummerer, you three are top! I don't know how I'd have managed pursuing my degree without you. Kim Heintz, thank you for discussions on successes and minor setbacks, both with this and with your own thesis, as well as your support beyond.

My family has provided me with some important prerequisites for science. I was raised in a house where discussions were concluded by searching for the relevant information in textbooks. Also, my education was completely provided for. Thank you for all your support throughout the years.

At the top of the pack is my significant other, Alina, who has given me insights to life beyond anything I could have imagined. Sometimes I have the feeling I know so little about what's important - thank you, expert. This journey with you is wonderful.

Other entities concerned with this thesis that I have benefited from also deserve to be acknowledged. At the University of Stuttgart, *alsNet* was trained and evaluated using hardware sponsored by NVIDIA. In Vienna, the process was continued on compute nodes of the Vienna Scientific Cluster VSC-3. The main dataset, of the federal state of Vorarlberg, was kindly provided by the federal state itself as Open Government Data. I especially want to thank Mag. Peter Drexel for his support concerning the data.

## Abstract

3D point clouds derived with laser scanning and other techniques are always big amounts of raw data which cannot be used directly. To make sense of this data, and allow for the derivation of useful information, a segmentation of the points in groups, units, or classes fit for the specific purpose is required. Since point clouds contain information about the geometric distribution of the points in space, spatial information has to be included in the classification. To assign class labels on a per-point basis, this information is usually represented by means of feature aggregation for each point from a certain neighbourhood. Studies on the relevance of the different features that can be created from such a neighbourhood exist, but they depend very much on the specific case at hand. This thesis aims to overcome this difficulty by implementing a Deep Neural Network (DNN) that automatically optimises the features that should be calculated.

After an introduction into the state-of-the-art methods in both point cloud classification and in neural networks, this novel approach is presented in detail. Three datasets were investigated, including an airborne laser scan (ALS) of a large area (Vorarlberg, 2700 km$^2$), a UAV-based scan (ULS) with a very high point density of a forest (Großgöttfritz) and a benchmark dataset by the ISPRS (Vaihingen/Enz, 3D Semantic Labelling Contest). The transfer of models between these datasets showed that point distribution patterns and point densities had a large influence on the result. However, using a pre-trained model on a new dataset vastly increased convergence of the method.

For the Vorarlberg dataset, the achieved overall accuracy with respect to the reference classification was 82.2 %, with a maximum of 95.8 % in urban areas. The accuracy showed a strong spatial correlation, especially with respect to land cover, suggesting the use of different models for different land covers. On the ISPRS benchmark dataset, the presented method achieved an overall accuracy of 80.6 %, which is comparable to other methods in the benchmark. Tiling of the input dataset into chunks for processing was shown to influence the classification result, especially in areas where the classification was incorrect.

A per-class probability for each point was additionally obtained in the classification process and may be used in further processing steps, e.g. as *a priori* weights in DTM generation.

Future applications of the method include tasks such as tree stem- or deadwood detection in forests. Especially with a growing number of attributes, the approach significantly reduces the input required from the operator (i.e. the selection of features). The method can also be extended to more dimensions, such as time. This would allow the classification of multi-temporal data, including change detection and displacement monitoring.

## Zusammenfassung

3D-Punktwolken, die mittels Airborne Laser Scanning (ALS) oder anderen Methoden erfasst wurden, sind große Mengen an rohen Daten. Um diese Daten zu verstehen, und um daraus weitere Informationen ableiten zu können, ist oft eine Segmentierung in Gruppen, Einheiten oder Klassen entsprechend dem jeweiligen Anwendungsfall notwendig. Da Punktwolken vor allem durch die geometrische Verteilung der Punkte im Raum Information transportieren, muss diese Information auch in die Klassifizierung berücksichtigt werden. Um nun diese Klassen auf Punkt-Basis zuteilen zu können, muss für jeden Punkt diese Information über eine lokale Nachbarschaft gesammelt werden. Es existieren zahlreiche Studien darüber, welche Repräsentationen dieser Information besonders relevant sind, allerdings hängt dies auch vom jeweiligen Anwendungsfall ab. In dieser Arbeit wird ein Ansatz präsentiert, der diese Schwierigkeit zu vermeiden versucht. Dabei kommt ein Deep Neural Network (DNN, zu deutsch: Tiefes Neuronales Netzwerk) zum Einsatz, das automatisch die Repräsentation der Nachbarschaft optimiert.

Zunächst wird eine ausführliche Einführung in die aktuellen Methoden der Punktwolkenklassifizierung und der Neuronalen Netze gegeben, bevor der neue Ansatz im Detail präsentiert wird. Dieser wurde auf drei Datensätzen getestet: Ein ALS-Datensatz mit großer räumlicher Ausdehnung (Vorarlberg, etwa 2700 km$^2$),

ein UAV-basierter Scan eines Waldgebiets (Großgöttfritz) sowie ein Benchmark-Datensatz der ISPRS (Vaihingen/Enz, Semantic Labelling Contest). Die Übertragung trainierter Modelle zwischen den Datensätzen zeigte einen großen Einfluss der unterschiedlichen Punktmuster und Punktdichten. Dennoch konnte durch den Einsatz eines bereits trainierten Modells die Konvergenz der Methode deutlich beschleunigt werden.

Mit dem Vorarlberg-Datensatz wurde eine Genauigkeit von 82,2 % über alle Testgebiete erreicht, wobei in einem urbanen Testgebiet eine Genauigkeit von 95,8 % erzielt wurde. Die Genauigkeit zeigte eine hohe räumliche Korrelation, die insbesondere mit der Landbedeckung zusammenhängt. Dies legt die Verwendung eines auf Landbedeckung angepassten Modells nahe. Der Benchmark-Datensatz konnte mit einer Genauigkeit von 80,6 % klassifiziert werden, was etwa im Mittelfeld der Benchmark-Ergebnisse liegt. Die Kachelung des Datensatzes führte zu Diskrepanzen in der Klassifizierung, insbesondere an jenen Punkten, die gegenüber der Referenz falsch klassifiziert wurden.

Zusätzlich zu der Klassifizierung wurde für jeden Punkt eine Wahrscheinlichkeit pro Klasse berechnet, welche in weiteren Prozessierungschritten, z.B. als *a priori* Gewichtung in der Interpolation von Geländemodellen, verwendet werden kann.

Weitere Anwendungen der Methode sind etwa die Stammdetektion oder Totholzdetektion in Forstbereichen. Mit einer wachsenden Anzahl an Attributen steigt die Stärke der Methode, da weniger Information vom Anwender benötigt wird. Die Methode kann auch auf weitere Dimensionen ausgedehnt werden, insbesondere auf Zeit. Damit wird die Klassifizierung multitemporaler Datensätze ermöglicht, inklusive der Detektion von Änderungen bzw. der Überwachung von Deformationen.

## Nomenclature and colour table

This thesis uses some of abbreviations and symbols, which are explained here:

| Symbol, Abbreviation | Explanation, Definition |
|---|---|
| $p_i$ | single point of a point cloud |
| $x_i$, $y_i$, $z_i$ | x, y, and z-coordinates of point $p_i$ |
| $n_x$, $n_y$, $n_z$ | normal vector components |
| $\bar{a}$ | average values of all $a$ (depending on context) |
| k-NN | k-Nearest Neighbours |
| $I$ | Identity matrix (dimensionality implied by context) |

Table 1: Abbreviations and symbols used in this thesis

## Colour map for point cloud classes

| Colour | ASPRS Class ID | Description |
|---|---|---|
| ⬛ | 0 | Never Classified |
| ⬛ | 1 | Unclassified |
| 🟪 | 2 | Ground |
| 🟩 | 3 | Low Vegetation |
| 🟩 | 4 | Medium Vegetation |
| 🟩 | 5 | High Vegetation |
| 🟥 | 6 | Building |
| ⬜ | 7 | Low Point, Noise |
| 🟨 | 8 | High Point |
| 🟦 | 9 | Water |
| 🟦 | * | any other class |
| 🟩 | n/a | Correctly classified point (in difference plots) |
| 🟥 | n/a | Incorrectly classified point (in difference plots) |

Table 2: Color scheme used for the point cloud plots, representing different classes.

# Contents

**1 Introduction** **7**
1.1 3D Point Clouds and Airborne Laser Scanning . . . 7
1.2 Point Cloud classification . . . 7
1.3 Neural Networks . . . 7
1.4 Research question . . . 8

**2 State of the art** **9**
2.1 Classification . . . 9
2.1.1 Random forest . . . 9
2.1.2 SVM . . . 11
2.1.3 Data preprocessing . . . 11
2.1.4 Grouping . . . 12
2.1.5 Feature extraction . . . 12
2.1.6 Optimization . . . 14
2.1.7 Analysis and Evaluation . . . 14
2.2 History of Neural Networks . . . 15
2.2.1 Copying nature . . . 16
2.2.2 Convolutional Neural Networks (CNNs) . . . 17
2.2.3 Applications in remote sensing and computer vision . . . 18
2.2.4 Neural Networks on Point Clouds . . . 19
2.3 Neural Network Best Practices . . . 23
2.3.1 Dropout . . . 23
2.3.2 Softmax . . . 23
2.3.3 Learning Rate . . . 24
2.3.4 Optimizers . . . 24
2.3.5 Transfer learning . . . 25
2.3.6 Minibatches . . . 25
2.3.7 Parameter search . . . 25

**3 Materials and datasets** **26**
3.1 Vorarlberg ALS 2011 . . . 26
3.2 Auberg ULS 2017 . . . 27
3.3 ISPRS 3D Semantic Labeling Contest . . . 27

**4 alsNet** **29**
4.1 General workflow . . . 29
4.2 Pyramid Scheme . . . 30
4.3 Neighbourhood definitions . . . 30
4.4 Variables in the data . . . 31
4.5 Variables for alsNet . . . 31

**5 Experiments** **32**
5.1 Initial training on Vorarlberg dataset . . . 32
5.2 Training with focus on specific land cover . . . 33
5.3 Thinned out data . . . 34
5.4 ULS data . . . 34
5.5 ISPRS 3D Semantic Labeling Contest . . . 34

**6 Evaluation** **36**
6.1 Different architectures . . . 36

6.2 Spatial correlation of results . . . 37
6.3 Separability of classes . . . 39
6.4 Large-Scale applicability . . . 43
6.5 Moving from ALS to ULS . . . 45
6.6 ISPRS 3D Semantic Labeling Contest . . . 45

**7 Discussion** **50**
7.1 Further experiments . . . 50
7.2 Outlook . . . 51

**8 Conclusions** **52**

# 1 Introduction

The motivation for this thesis is two-fold: Classification of 3D point clouds is (still) a very relevant problem, and the use of neural networks in classification tasks has been successfully proven in many disciplines. Neural networks allow for automatic transformation and extraction of the relevant data from the input. This is especially useful when describing the distribution of a point cloud for which a number of representations exist. This thesis aims to show that a neural network can successfully learn to represent this distribution in order to classify the individual points. The concept eliminates the need for hand-crafted features. While some first results are presented in Section 6, the main focus lies on the presentation and design of *alsNet*, a workflow to effectively pre-process, train and evaluate this neural network.

## 1.1 3D Point Clouds and Airborne Laser Scanning

A 3D point cloud is defined as an unordered, non-regular set of coordinate tuples, possibly annotated with per-point features, also called *attributes* (Otepka et al. 2013). As such, the point clouds investigated in this thesis represent points sampled from the surface of 3D objects. Here, we look at the Earth as the object from the bird's eye perspective , including objects on top of it (buildings, trees, cars, etc.). Such a point cloud can be acquired on a large scale by airborne laser scanning (ALS) (Pfeifer and Mandlburger 2018). With ALS, point densities are usually within 1 to 10 points/m$^2$ (Dorninger and Pfeifer 2008). More recently, laser scanners have also been mounted on unmanned aerial vehicles (UAVs), resulting in higher point densities (100-1000 pts/m$^2$) for smaller areas (Wallace et al. 2016). The high point density is mostly a result of the much lower distance between sensor and object, and higher overlaps. UAV-based laser scanning is sometimes referred to as ULS. The resulting large amounts of points requires efficient handling and further processing (Wieser et al. 2017).

## 1.2 Point Cloud classification

For many applications, a classification, or semantic labelling, of the points is necessary. Essentially, every point is assigned a label corresponding to the type of object it represents. In the context of ALS, the typical classes are Ground, Building, Low-, Medium- and High Vegetation, Water, and Low/High Points (Noise). Using this information, the point cloud can then be segmented or filtered for specific use cases. For example, forest maps can be derived based on vegetation classes, or solar potential maps from roof points. Especially for the interaction boundary between geoscience and other disciplines, point labels are an important feature of a point cloud.

## 1.3 Neural Networks

At the same time, we observe that neural networks have recently gained a lot of popularity in industry and science. The basic idea, originally published in 1943, is to provide a mathematical model to represent the function of information storage and organization in the brain (McCulloch and Pitts 1943). This is achieved by a model of interconnected neurons (representing the nervous cells) in a network. Such a neuron can *learn* how to process inputs.

The recent advances in image recognition tasks are mainly due to efficient handling and processing of convolutional neural networks (CNNs). Such networks learn to recognise local features in images such as corners, edges and isolated points. For CNNs, the data has to be arranged in a regular grid, such as a digital image. To apply CNNs on point clouds, the points have to be transformed to such a regular structure first (see Section 2). This inevitably leads to either very sparse data representations, where many grid cells are empty, which is inefficient, or to the loss of information, when multiple points are aggregated in each cell.

## 1.4 Research question

A desirable classification algorithm would process a 3D point cloud directly. In 2017, a neural network with this property, called *PointNet*, was presented (Qi, Su, et al. 2017). *PointNet* is able to deal with the intrinsic properties of a point cloud presented in Section 1.1.

The research questions that follow from this are:

- Is a neural network based on *PointNet* able to classify points from ALS?
- What parameters of the neural network can be tuned to optimise the classification accuracy?
- What is the effect of different point densities, e.g. coming from a different sensor/platform, on the classification result?
- Which classes can be separated more or less efficiently, and why?
- Is it feasible to provide a trained network for a specific sensor and land cover, e.g. by a sensor manufacturer, as a product?

This thesis aims to answer these questions by implementing a *PointNet*-based algorithm in an attempt to classify 3D point clouds from ALS and ULS. After presenting a short history of neural networks and state-of-the art methods of point cloud processing in Section 2, the data are presented in Section 3. The architecture of the neural network and its application on large datasets is discussed in Section 4. Section 5 presents the experiments carried out using the neural network, which are evaluated in Section 6. An extensive discussion (Section 7) and conclusions (Section 8) lead to further research questions for future work.

# 2 State of the art

Because of the two-fold motivation for this thesis, both state of the art and legacy classification methods as well as a small introduction to neural networks are presented by following a historic approach. Especially the latter is by no means a complete presentation but rather intended to make the reader familiar with the terms used and the reasoning behind it.

## 2.1 Classification

Classification in the context of this thesis is always meant as supervised classification, i.e. independent ground truth data is used to train the classifier. Some approaches for unsupervised segmentation of point clouds exist (e.g. Biosca and Lerma 2008) but are out of the scope of this thesis.

The classification of a point cloud does not necessarily have to be done before any other calculations can be carried out. Sometimes, it can also be useful to first group points together and then assign a label to the point group instead of a single point. This reduces noise, such as salt and pepper points, where single points of one class are surrounded solely by points of another class, but requires more sophisticated features. This grouping can be done by already recognising objects (object-based classification, e.g. classifying roof types with a model catalogue (Englert and Guelch 1996; Haala and Brenner 1999)) or segments, i.e. primitive clusters of points such as flat surfaces, etc.

Traditional approaches to point cloud classification follow somewhat similar paths. The basic path is outlined in Figure 1. Two classification algorithms are presented first, then the individual steps of pre- and post-processing are discussed in Sections 2.1.3 to 2.1.7.

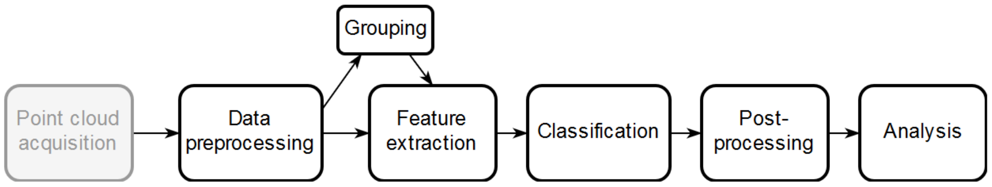


Figure 1: Traditional point cloud classification pipeline. Adapted from J. Zhang, Lin, and Ning 2013

Once features (geometric, radiometric and others) have been calculated for each point, they can be used to classify the point. The most basic classifier is a binary decision tree. Starting at the root node, every node represents a binary separation based on a feature. At the end (on the leaves), the individual classes (in the binary case 1 and 0) are assigned. More complex decision trees also allow the assignment of multiple classes, i.e. having a number of leaf nodes equal to the number of classes.

### 2.1.1 Random forest

Decision trees require an expert to tune the boundaries between the classes, and they do not give any reliability measure. To overcome this, machine learning approaches have been widely used (Weinmann, Jutzi, and Mallet 2013).

A *random forest* is a prominent example of so-called *Ensemble Learners*. It consists of a large number of decision trees (hence "forest"), each of which only operates a little bit better than a completely random classifier. An input is fed through all of the trees simultaneously, and statistics are calculated from the output classes. For example if, 57% of the trees may say that a specific input results in Class A, the output probability for Class A is 57%.

The decision trees are not crafted manually, but rather automatically. For this, different methods exist: ID3 (Iterative Dichotomiser 3) was developed around 1980 and is focused on discrete input data and a binary classification problem. To overcome these restrictions, and to improve the calculation costs, it was developed further into algorithms called C4 and then C4.5. Finally, a randomized variant of C4.5 was published (Dietterich 2000). This method was later on referred to as "randomized trees", and built the basis for random forests. Based on the training data, each node results in a number of samples. The local subset of the data is then divided by means of the "best" one of these samples (Hänsch and Hellwich 2017).

A strength of random forests is that there is no limitation on linear separation. A single decision tree is limited to an orthogonal separation of the input space, but since many decision trees are taken into account in the *forest*, this is overcome. Instead, the borders between the classes are "washed", i.e. smoothed by means of a probability gradient. Figure 2 shows an example of a *random forest* classification, where data is distributed very non-linearly. Still, the results are quite acceptable in most of the cases.

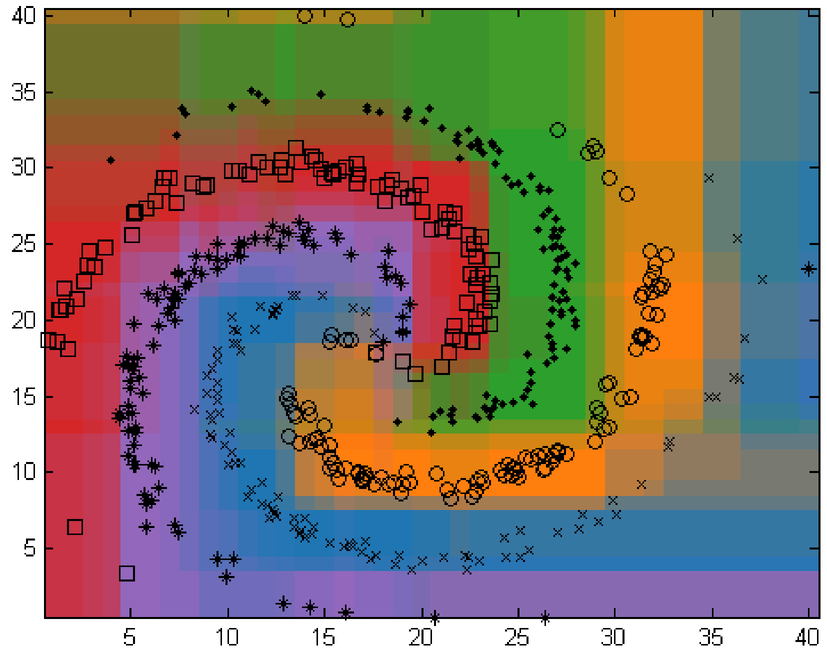


Figure 2: Classification result of a random forest on data that is classified in spiral arms. The markers represent the training samples, the colors the result classes (and their probabilities). Figure taken from https://de.mathworks.com/matlabcentral/fileexchange/44121-randomforest-example by Wasit Limprasert (2013) (last visited on 2018-07-24).

*Random forests* have two main parameters. The first one is the number of trees that are created, which has a direct impact on the runtime of the algorithm. A bigger number of trees ensures that the result is better, i.e. the class probabilities are reliable, because enough randomly generated trees were used to create a meaningful sample. However, the runtime increases with a larger number of trees. The second parameter is the size of an individual tree, which allows to accommodate more complex scenarios, the bigger it is (Hänsch and Hellwich 2017).

### 2.1.2 SVM

Another machine learning classifier is the Support Vector Machine (SVM). SVMs are linear classifiers that maximize the distance of the nearest training dataset from the separating hyperplane (called *support vector*). This already presents two major limitations: SVMs primarily separate only two classes from each other, and the classification of data that is non-linearly separable requires additional steps.

Overcoming the linearity constraint can be done by transforming the input features into a space of higher dimensionality, e.g. $x \to x^2$. In this transformed space, the classes can be separated with a linear boundary (i.e. a hyperplane) (Hearst et al. 1998).

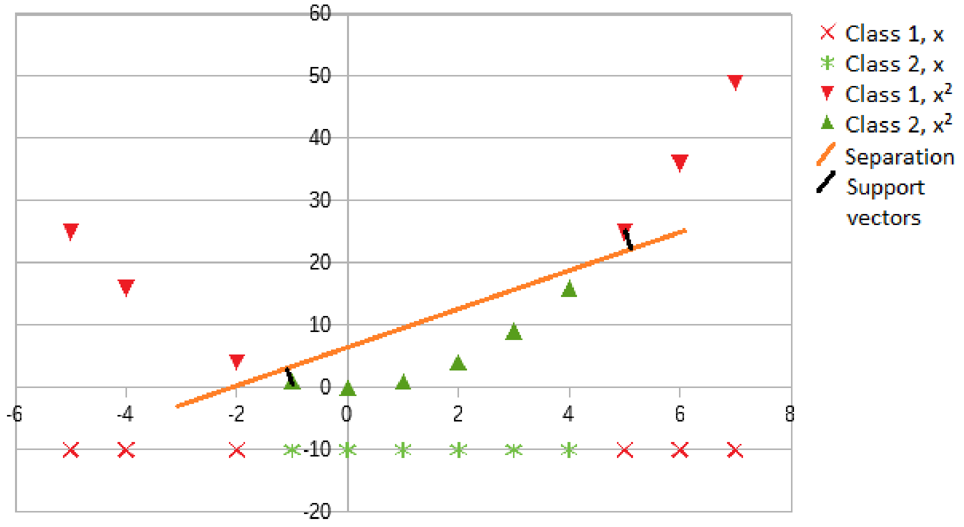


Figure 3: Support vector machine example. The red crosses and green stars represent the input data (1D) in two classes. The triangles represent the transformed data (2D) via the transformation $(x, y) = (x, x^2)$. In this higher dimension, the classes can be separated linearly (i.e. with a line). The support vectors (black lines) are maximized by the SVM. The 1D-dataset is shown at $y = -10$ for a cleaner image.

Using a "kernel trick", this transformation can be done on the separating hyperplane itself, and not on the data, which saves a lot of computational costs. The transformation allows for the separation of classes where not a single boundary might be sufficient, as can be seen in Figure 3. Here, transformation from 1D- to 2D space results in a two-boundary condition on the original dataset.

Many other classifiers exist, but this thesis will focus on *neural networks*, introduced in the next chapter.

In ALS, the classification scheme by the American Society of Photogrammetry and Remote Sensing is often used (ASPRS 2011). The most relevant classes are shown in Table 3.

### 2.1.3 Data preprocessing

Before the classification, the raw point cloud from the sensor has to be preprocessed. This includes a transformation from the scanner coordinate system to a project coordinate system (e.g. UTM/ETRS89), quality control (strip fitting precision, point density assessment), geometric calibration of the sensor model (i.e. strip adjustment) and radiometric calibration (i.e. derivation of sensor-independent reflectance values). Outliers can be removed with a range filter (removing points that are too close/too far away from the sensor) or a spatial filter (if the area of interest is known). Since modern laser scanners allow multiple echoes per laser pulse, this information can also be used for filtering (e.g. to last echoes only for a terrain model).

| Class ID | Description |
|---|---|
| 0 | Created, Never Classified |
| 1 | Unclassified |
| 2 | Ground |
| 3 | Low Vegetation |
| 4 | Medium Vegetation |
| 5 | High Vegetation |
| 6 | Building |
| 7 | Low point (noise) |
| 8 | Model key point (mass point) |
| 9 | Water |

Table 3: Classification scheme from the ASPRS LAS Standard. Table adapted from (ASPRS 2011, p. 11).

### 2.1.4 Grouping

As mentioned above, if the classification is not done for each point individually, segments or objects may be created from a group of points. This can for example be achieved with clustering algorithms (e.g. k-Means, originally by Steinhaus 1956; cf. MacKay 2003, Chapter 20), segmentation algorithms such as plane extraction (Awrangjeb, C. Zhang, and Fraser 2013) or criterion-based region growing (cf. Pöchtrager 2016).

### 2.1.5 Feature extraction

A very important property of point clouds is that points are interrelated, i.e. the position of a point in relation to its neighbourhood is important. A principal component analysis of the local neighbourhood gives an accurate description of such a neighbourhood. Also, higher-order moments or other statistical measures such as the quality of a planar fit or the "Echo Ratio" (Rutzinger, Höfle, and Pfeifer 2007) are important features describing the neighbourhood on a per-point basis.

The covariance matrix used in the principal component analysis, also called *structure tensor*, is a collection of second-order moments and can be calculated as follows (Jutzi and Gross 2009; Gross and Thoennessen 2006):

$$M = \begin{pmatrix} \tilde{m}_{200} & \tilde{m}_{110} & \tilde{m}_{101} \\ \tilde{m}_{110} & \tilde{m}_{020} & \tilde{m}_{011} \\ \tilde{m}_{101} & \tilde{m}_{011} & \tilde{m}_{002} \end{pmatrix} \tag{1}$$

where

$$\tilde{m}_{ijk} = \frac{\sum_{l=1}^{N} (x_l - \bar{x})^i \, (y_l - \bar{y})^j \, (z_l - \bar{z})^k}{R^{i+j+k} N} \tag{2}$$

. $R$ is the radius of the (here spherical) neighbourhood. The second order moments $\tilde{m}_{ijk}$ (where $i+j+k=2$) are invariant to translation, rotation, and scale (Maas and Vosselman 1999).

From this *structure tensor*, the eigenvalues $\lambda_i$ can be obtained by solving the characteristic polynomial of the matrix equation (3).

$$M\vec{v} = \lambda\vec{v} \tag{3}$$

where $\vec{v}$ is any vector but the null-vector by rewriting Eqn. 3 as

$$M\vec{v} - \lambda\vec{v} = 0 \tag{4}$$

$$(M - \lambda I)\vec{v} = 0 \tag{5}$$

This can be interpreted as a homogenous linear equation system $A\vec{x} = 0$, which only has non-trivial solutions ($\vec{v} \neq \vec{0}$) if the determinant of the coefficient matrix $(M - \lambda I)$ is zero (i.e. the matrix is rank deficient) (Goodfellow, Bengio, and Courville 2016, pp. 42 ff.).

For a 3x3-Matrix, up to three different solutions for $\lambda$ exist (depending on the rank deficiency of the matrix $(M - \lambda I)$), and they are ordered descending, with $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$.

On a segment- or object basis, one can calculate such features on basis of the local group, or aggregating the features of all points within the group.

A recent study discusses the most relevant features for point cloud classification. They find that both 2D- and 3D-features have an influence on labelling results. Especially the ratios of eigenvalues from a covariance matrix and normal vector components are important. The five highest ranked features are (Weinmann, Jutzi, and Mallet 2013):

- The Ratio $R_{\lambda,2D}$ in Eqn. 6 is a measure for how well the points are distributed on their 2D footprint. A value close to 1 indicates an even (isotropic) distribution.

$$R_{\lambda,2D} = \frac{\lambda_{2,2D}}{\lambda_{1,2D}} \tag{6}$$

- Verticality $V$ (Eqn. 7) is a measure of how vertical the local neighbourhood is. A value close to zero represents a horizontal surface, a value close to one a vertical one.

$$V = 1 - n_z \tag{7}$$

- The change of Curvature $C_\lambda$ (Eqn. 8) is a good second-order measure, with values close to 0 indicating a linear or flat point distribution, and values up to 1/3 representing omnidirectionality (Rusu 2009).

$$C_\lambda = \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \tag{8}$$

- Finally, the standard deviation $\sigma_{Z,\text{k-NN}}$ (Eqn. 9) and the range $\Delta_{Z,\text{k-NN}}$ (Eqn. 10) give an measure on the vertical distribution of the points in a k-nearest neighbourhood.

$$\sigma_{Z,\text{k-NN}} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{k} (z_i - \bar{z})^2} \tag{9}$$

$$\Delta_{Z,\text{k-NN}} = \max_{i=1..k} z_i - \min_{i=1..k} z_i \tag{10}$$

In their analysis, Weinmann, Jutzi, and Mallet investigate a total of 21 features, with the remaining 16 being less and less relevant to semantic labeling.

While the features mentioned by Weinmann, Jutzi, and Mallet describe a point in a point cloud with respect to its neighbours, it is not clear yet which points constitute this neighbourhood. Multiple different rules to determine neighbourhood exist: *k-NN* simply takes the $k$ nearest neighbours, *fixed radius* takes all points within a sphere of given radius, *quadrant-* (2D) or *octant* (3D) based neighbourhoods ensure a good spatial distribution. These rules can also be combined, e.g. selecting up to a maximum of 20 neighbours within a 1 m search radius.

The individual values that should be used vary highly with local point densities. Also, it might be useful to derive values based on neighbourhoods of different scales. For example, a locally flat surface embedded in a bigger elevated area might be a (flat) roof, while a locally rough patch might represent a chimney. The same rough patch in a different bigger context may represent a tree, and the flat surface a road (not elevated with respect to the terrain).

This example shows the significance of the distribution of features describing a smaller neighbourhood in the context of a bigger neighbourhood. An extension of describing neighbourhood not only with respect to point coordinates, but also with respect to point attributes, can in turn describe the local neighbourhood, if these attributes contain information on the bigger neighbourhood. This is seldomly treated in classification and presents a major advantage of the method presented in Section 4.

Using the features, the points can be assigned individual class labels using a classification algorithm, such as *Random Forest*, *SVM* or a neural network.

#### 2.1.6 Optimization

After classification, post-processing may further improve the result. This is especially the case for classification on point basis. Often, single points may be of a different class within an otherwise homogenous area. *Semantic label smoothing* is a technique which removes this noise. Using the semantics, further optimization is possible, e.g. removing ground points that are higher than vegetation points in a small surrounding (Landrieu et al. 2017).

#### 2.1.7 Analysis and Evaluation

To quantify the success of the classification, multiple metrics exist. They are all calculated from the comparison of evaluation data, where the class has been predetermined by some other means (another classification, possibly with manual post-processing). This class ("ground truth") is compared with the estimation from the classifier. The result can be represented in a confusion matrix. In the rows, the ground truth labels are listed, in the columns the estimation. Each point falls into one of the matrix elements, with the main diagonal representing correctly classified points. An example is shown in Table 4.

| | | Estimation | | |
|---|---|---|---|---|
| | | Class 1 | Class 2 | Class 3 |
| Ground truth | Class 1 | 251 | 0 | 2 |
| | Class 2 | 6 | 583 | 10 |
| | Class 3 | 11 | 120 | 125 |

Table 4: Sample confusion matrix for three classes

The inclusion of a point into a certain class is referred to as *commission*, the absence of a point as *omission*. In the example of Table 4, two points have been omitted from Class 1 and instead committed to Class 3.

While the confusion matrix is the most complete representation of classification success, further values that allow for easier comparison may be derived from it (Fawcett 2006). The numeric values in the following definitions always refer to the confusion matrix in Table 4:

- Accuracy is the most prominent measure, being the quotient of the number of correctly classified points (the trace of the matrix) and the total number of points (sum over rows and columns).

$$\mathrm{Acc} = \frac{\sum_i a_{ii}}{\sum_i \sum_j a_{i,j}} = 86.5\,\% \tag{11}$$

- Intersection over Union (IoU) is a measure very close to accuracy, but can be calculated for each class separately. It is the quotient of the number of correctly classified points (for a single class) divided by the line and column-sum of this class (omitting the diagonal entry for one of the sums).

$$\mathrm{IoU}_i = \frac{a_{ii}}{\sum_k a_{ki} + \sum_l a_{il} - a_{ii}} = \begin{pmatrix} 92.96\,\% \\ 81.08\,\% \\ 46.64\,\% \end{pmatrix} \tag{12}$$

- Sensitivity (also: recall) is a measure for a class describing how many of the points that are actually in the class have been correctly classified. It is given by the quotient of the diagonal element and the sum of the elements in the same row. The commission of points from other classes to this specific class has no influence on the metric.

$$\mathrm{Sensitivity}_i = \frac{a_{ii}}{\sum_k a_{ki}} = \begin{pmatrix} 99.21\,\% \\ 97.33\,\% \\ 48.83\,\% \end{pmatrix} \tag{13}$$

- Specificity shows how well the classifier picks out a specific class, by only taking the omitted points into account. It can be calculated as the quotient of the sum of all matrix elements excluding the current column and the sum of all matrix elements excluding the value at the current line and column (diagonal element).

$$\mathrm{Specificity}_i = \frac{\sum_{k,k\neq i} \sum_l a_{lk}}{\sum_{k,k\neq i} \sum_l a_{lk} + \sum_{k,k\neq i} a_{ki}} = \begin{pmatrix} 98.02\,\% \\ 77.14\,\% \\ 98.78\,\% \end{pmatrix} \tag{14}$$

- Precision shows how many of the points that were classified as class $i$ actually belong to class $i$ by taking the quotient of the diagonal element and the column sum.

$$\mathrm{Precision}_i = \frac{a_{ii}}{\sum_k a_{ik}} = \begin{pmatrix} 93.66\,\% \\ 82.93\,\% \\ 91.24\,\% \end{pmatrix} \tag{15}$$

- The $F1$-score is a combination of precision and sensitivity, and can be obtained by taking the quotient of two times the product of precision and sensitivity divided by the sum of precision and sensitivity.:

$$F1_i = 2 \cdot \frac{\mathrm{Precision}_i \cdot \mathrm{Sensitivity}_i}{\mathrm{Precision}_i + \mathrm{Sensitivity}_i} = \begin{pmatrix} 96.35\,\% \\ 89.55\,\% \\ 63.61\,\% \end{pmatrix} \tag{16}$$

The large number of points belonging to Class 3 in the example that were incorrectly estimated as Class 2 (see Table 4; value 120) has a negative impact on IoU, Sensitivity and $F1$ of Class 3, and a negative impact on IoU, Specificity, Precision and $F1$ of Class 2.

## 2.2 History of Neural Networks

A large body of literature on neural networks exists (Schmohl 2017; Laupheimer 2017; Hemmes 2018; Goodfellow, Bengio, and Courville 2016; Géron 2017). Here, a historic approach is given, showing how different extensions to the original idea as presented by McCulloch and Pitts in 1943 improved the performance of neural networks over time. The focus, especially in more recent years, is on approaches to detect objects in images and in point clouds. In the last paragraphs, common practices and terms used with neural networks are presented.

### 2.2.1 Copying nature

The basis of neural networks comes from looking at nature. A neuron, the nerve cell present in many organisms, consists of a soma (body) and an axon (arm). Multiple neurons are interconnected by having their axons connect to the soma of another neuron (via synapses) (Schünke, Schulte, and Schumacher 2015, p. 268).

This idea is taken up by McCulloch and Pitts. They create a model where multiple inputs are combined. When this combination reaches a certain threshold, the output changes from zero to one, i.e. the neuron fires. A mesh of multiple such neurons into a network allows a logical description of nervous systems (McCulloch and Pitts 1943).

Rosenblatt further concretised this. He used a model of linear combination, i.e. the $n$ inputs $I$ get multiplied by $n$ weights $w$ to form a weighted sum, and included a bias $b$:

$$\sum_{i=1}^{n} I_i \cdot w_i + b \tag{17}$$

This weighted sum is further passed as an argument to an activation function $\varphi$, which decides when to "fire" the output. The domain of the activation function is not restricted, so the output is not necessarily binary.

This model, named *Perceptron*, can be trained, i.e. the weights $w_i$ and the bias $b$ can be learned from a number of labeled input samples. Such a trained model can then be used to classify data which is linearly separable. The infamous XOR example, a basic logical operator which is not linearly separable, shows the limits of the *Perceptron* (Rosenblatt 1958). A solution to break the linearity constraint is using a non-linear activation function $\varphi$ on the result of the weighted summation in Eqn. 17 (Goodfellow, Bengio, and Courville 2016, pp. 169 ff.).

The visual cognitive system has already been of interest to both McCulloch and Pitts as well as Rosenblatt. Applying neural networks to images, however, was more difficult and required a more complex network than what was available and feasible at that time. Especially the training of a neural network with more than one layer or with non-linear activation functions was not possible using the original training algorithm for the *Perceptron*. But in 1986, a group lead by a mathematically trained psychologist developed an algorithm called "backpropagation". Using this algorithm, the gradient of the neural functions can be calculated backwards through all layers, where the weights of the current layer are adjusted accordingly (Rumelhart, Hinton, and Williams 1986).

For a mathematical solution, a loss function (also called cost function) $E$ has to be defined. It gives a scalar value on the optimality of a result, the goal of the learning process is to minimise this function. It can also be interpreted as the energy $E$ required to get from the inferred solution to the ground-truth. For classification purposes, the cross-entropy is a plausible measure, as defined in Eqn. 18. Here, $P(i)$ is an indicator of the current input belonging to Class $i$ (either 0 or 1), and $Q(i)$ is the predicted probability of the input belonging to Class $i$ (Fortuner 2018).

$$E = -\sum_{\text{Classes i}} P(i) \log(Q(i)) \tag{18}$$

To minimise this loss function using backpropagation, weight updates $\Delta w_{ij}$ have to be calculated for each neural connection $i$ in each layer $j$. For this, the chain rule of differentiation is applied:

$$\Delta w_{ij} = -\frac{\partial E}{\partial w_{ij}} = -\frac{\partial E}{\partial \varphi} \frac{\partial \varphi}{\partial w_{ij}} \tag{19}$$

For the last layer, $\frac{\partial\varphi}{\partial w_{ij}}$ is easy to calculate via the loss function. For all other layers, the derivative with respect to the weights $w_{ij}$ requires repeated uses of the chain rule, increasing in complexity (Rumelhart, Hinton, and Williams 1986). The weight updates $\Delta w_{ij}$ are then added to the current value of $w_{ij}$. The bias is updated in a similar manner.

After many training samples are given to the algorithm, an optimum is reached. The most basic example of a network that can be trained with backpropagation is the multi-layer perceptron (MLP). It consists of multiple chained *Perceptrons*, combined with a non-linear activation function. This allows to overcome the problem of solving non-linear problems (Goodfellow, Bengio, and Courville 2016, pp. 5; 163 ff.).

A multi-layer perceptron is shown in Figure 4. The summation from Eqn. 17 is done implicitly by matrix multiplication. In this notation, a whole layer of neurons can be represented with one equation. Input $x$ and output $o$, as well as intermediate output $h_1$ and the biases $b_1$, $b_2$ are vectors. The size of the matrices $A_1$ and $A_2$ determine the ability of the MLP to approximate the target function.

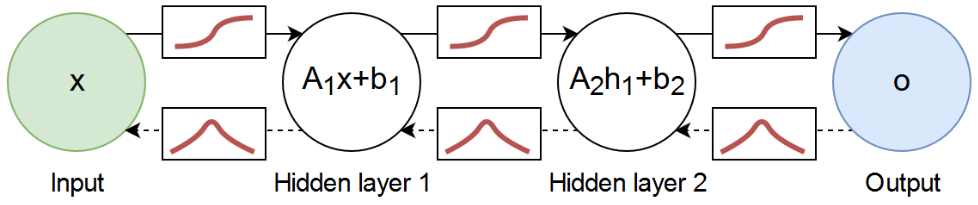


Figure 4: Example of a multi-layer perceptron (MLP). The activation functions (sigmoid) are shown in red on top, their derivatives on the bottom. The dashed arrows indicate backpropagation, the solid ones feed-forward or inference. The parameters that can be learned are contained in the matrices $A_1$ and $A_2$, as well as in the vectors $b_1$ and $b_2$.

Three years later, it was shown that when using a (non-linear) sigmoid function as activation function and enough neurons, any continuous function can be arbitrarily approximated by a neural network (Cybenko 1989). This was also shown later for other popular activation functions such as the tanh or the ReLU (Nair and Hinton 2010,; Sonoda and Murata 2015). These publications demonstrate the advantage of neural networks when compared to e.g. polynomial approximations, where the type of function needs to be known *a priori*.

### 2.2.2 Convolutional Neural Networks (CNNs)

With increasing complexity, an increasing number of neurons is needed to achieve satisfying results. For image analysis, each pixel can be seen as an (R,G,B)-tuple of values, typically ranging from 0 to 255. With large images, the number of neurons needed to process these images rapidly increases. For example, a 100 by 100 pixel with a single hidden layer would already amount to 100,000,000 ($10^8$) connections.

To overcome this problem, it was proposed to use locally shared weights for image recognition. This is implemented by a convolution, a mathematical operation that has been used in image processing for tasks like edge detection or sharpening (LeCun, Boser, et al. 1989; LeCun, Bottou, et al. 1998). Using convolutions, a relatively small number of weights can be used independent of the image size, as a rectangular kernel is moved over the whole image. The pixel under the center of the kernel is then set to the value of the sum of all pixel values multiplied by the kernel values. The *LeNet-5* was able to effectively classify hand-written digits (LeCun, Bottou, et al. 1998).

This idea is also supported by studies of the visual cortex of cats (Hubel 1958; Hubel and Wiesel 1959) and monkeys (Hubel and Wiesel 1968). They showed that neurons have a small receptive field, i.e. a small part of the retina is connected to a neuron (Géron 2017, pp. 357f.).

The matrix in Eqn. 20 shows a kernel for edge detection in x-direction, usually called the Sobel operator. It represents an approximation to the first derivative in x-direction of the image (Sobel and Feldman 1968). In neural networks, the kernel values are represented by the unknown weights.

$$S_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \tag{20}$$

Given the image as a greyscale image $G$, the result of the convolution is then calculated as

$$(G * S)_{i,j} = \sum_{m=0}^{2} \sum_{n=0}^{2} S_{m,n} G_{(i-m),(j-n)} \tag{21}$$

$m$ and $n$ range from 0 to 2 since the convolution kernel in Eqn. 20 has the shape 3x3. This is not a general limitation. When reaching the border of the input image $G$, different methods exist. *Same convolution* extends the input image by rows and colums of zeros so that the resulting convolution will have the same dimension as the original image. *Valid convolution* only returns results at locations where the entire kernel is covering valid pixels of the input. This results in smaller images than the input. *Full convolution*, on the other hand, returns a result whenever at least one element of the kernel overlays the input image, resulting in a picture larger than the input picture (Goodfellow, Bengio, and Courville 2016, pp. 338 f.). To further reduce the number of neurons, convolutional layers may not be evaluated at every position, but rather skip a number of pixels when moving the kernel across the image. This is referred to as a *stride* (Géron 2017, p. 357).

When using a kernel with the same dimensions as the input image, every pixel is treated separately. While this is the most general setting, the strength of convolutions is to *share parameters*. This way, information can be extracted from the image while only using a small number of neurons, thereby reducing computational complexity immensely. The exact kernel size that should be used very much depends on the input resolution and the task at hand (Goodfellow, Bengio, and Courville 2016, pp. 324 ff.).

With convolutional neural networks, the values in the kernel are learned. This way, an image can be analysed with a small number of weights. To detect both local and global features in an image, the image is downscaled after the first convolution, and a second convolution is applied. The downscaling is usually done via max-pooling, meaning every sub-grid of n-by-m pixels is summarized by their maximum value. Whether the maximum or minimum value (or any other scalar value derived from the $nm$ input values) is used does not matter, since the learning process will adjust the respective weights accordingly. A convolutional unit therefore usually consists of three layers: a) a convolutional layer, b) a nonlinearity layer (activation function $\varphi$, e.g. ReLU) and c) a pooling layer (Goodfellow, Bengio, and Courville 2016, pp. 330 ff.).

In the different layers, CNNs have been shown to learn to recognise lines, corners and curves in the first few layers, and higher-order shapes in later layers. This also shows the ability of neural networks to generalise and to separate complicated tasks (such as telling an image of a dog from an image of a cat) into simpler tasks (detecting edges, primitive shapes such as triangles, circles, and so on).

### 2.2.3 Applications in remote sensing and computer vision

Image processing is a common task in both remote sensing and computer vision. CNNs have therefore been used widely in these disciplines. Especially in remote sensing, single pixel classification is important, e.g. for land cover map generation. This requires a label on every pixel of the input image, which requires some adjustments to the output from the CNN, since the CNN only gives one label per image.

Another motivation comes from the computer vision community, especially with respect to autonomous vehicles. Cameras are widely available sensors in self-driving or assisted-driving cars. While monocular systems only allow to derive 3D-information during motion, the detection and interpretation of street signs, road markings and other features can be done on single images without depth information (e.g. Zhu et al. 2016).

A recent publication presents how to *upsample* or *unpool* images in a CNN to get a per-pixel semantic label. The information contained in the image is encoded using a standard CNN, but cut off before the last classification layer. A decoder network then maps the result back to the original resolution by remembering the indices that were used in the max-pooling operation of the encoder. This results in an image where every pixel is labelled according to the class of the object it represents, such as road, road marking, pedestrian, and many more (Badrinarayanan, Kendall, and Cipolla 2017).

### 2.2.4 Neural Networks on Point Clouds

The inherent properties of point clouds make a direct input to a strictly neuron-based system of neural networks difficult. There are a couple of different approaches to overcome this problem, for each of which an example is presented in the following paragraphs.

**Bird's Eye View Pixel-based approaches**
Here, a baseline CNN is used on a raster image generated for every point: A three-channel image is created based on the relation of the maximum, minimum, and mean point heights in a cell to the current point. This image represents the local neighbourhood of the point. The result is a binary decision whether the point is part of the terrain or not. Since a CNN has to be inferred for every point in the point cloud, this approach is quite expensive. Also, the definition of features as well as the raster extent and cell size has an impact on the classification result (Hu and Yuan 2016).

A more recent improvement to this approach does not solely rely on rasterised features per neighbouring cell, but rather selects one representative point per cell. For this point, features are calculated according to the methods presented by Weinmann, Jutzi, and Mallet. However, they also limit their analysis to three values per cell, a limitation that is not necessary. This limitation is possibly introduced to be similar to computer vision approaches with three-channel (Red, Green, Blue) imagery. On the ISPRS Vaihingen (see Section 3.3) benchmark dataset, they achieve an overall accuracy of 84.9 % (Yang et al. 2017).

**Point View Pixel-based approaches**
While being similar to the Bird's Eye View approach, this approach is less focused on airborne laser scanning and topographic point clouds. A virtual view from every point is generated and fed to a CNN. This is achieved by positioning a virtual camera at the point and then rendering an image of the point cloud (or its attributes) from the view of the point. Ideally, this image would be panoramic and therefore represent a complete 3D neighbourhood. Using a fully spherical representation also eliminates the need for decision on what strategy to use at the border of the image, since it wraps around onto itself. The mathematical basis for such a spherical CNN has been published (Cohen et al. 2018).

**Projective View Pixel-based approaches**
This approach combines the semantic segmentation (Badrinarayanan, Kendall, and Cipolla 2017) with virtual view generation. For a scene, multiple virtual images are rendered and individually classified. From the classified pixels, the original point is reconstructed and then labelled. Since a point is visible in multiple images, an accuracy score can be inferred from the redundant information (Lawin et al. 2017).

**Voxel-based approaches**
Convolution can be extended to three dimensions when operating on a regular grid of cubes. As in the 2D image case, a point cloud has to be regularized to a grid structure e.g. by selecting all the points in such a cube (volume element, voxel), and counting them. This leads to a so-called occupancy voxel representation. Alternatively or additionally, other values may be derived on a voxel basis from the point cloud (e.g. mean intensity, roughness measures, ...) (Maturana and Scherer 2015). Like in the 2D case, the baseline CNN outputs a class per input point cloud. This information can then be *unpooled* to give a class information per voxel. Disaggregating this information to individual points however needs more information and cannot be done in the CNN.

An extensive study on the robustness of 3D CNNs with respect to noise was done by Schmohl, showing very promising results on occupancy voxels of CAD models (Schmohl 2017).

**PointNet, PointNet++**
Qi, Su, et al. (2017) presented a novel approach to point cloud processing in neural networks. The idea of their so-called *PointNet* is to directly use point clouds as a set of unordered tuples of coordinates and attributes as input to a neural network. This is achieved by the approximation of a general function $f$ acting on the point cloud by the use of two auxiliary functions $g$ and $h$, as shown in Eqn. 22 (Qi, Su, et al. 2017).

$$f(\{p_1, p_2, \ldots, p_n\}) \approx g(h(p_1), h(p_2), \ldots, h(p_n)) \tag{22}$$

To be independent of the order of the input points $p_i$, the function $g$ has to be symmetrical on its input arguments $h_1, h_2, \ldots, h_n$ (where $h_i = h(p_i)$). This can easily be achieved by using an aggregation function such as $g = \sum_i h_i$, $g = \max_i h_i$ or $g = \bar{h}$ (the average of all $h_i$). Qi, Su, et al. show that the choice of $h$ does not really matter, presumably because the neural network that represents the function $h$ will adjust accordingly.

$h$ is approximated by the use of a ”vanilla” neural network (multi-layer-perceptron, MLP). In this case, a relatively small fully connected network with 3-4 layers and a few hundred neurons per layer is used (Qi, Su, et al. 2017). Since the function is not learned for each point individually, but one single function $h$ is applied to all points, both a generalisation and an independence of the input order are achieved.

*PointNet* further implements a coordinate transformation (via matrix multiplication, similar to a principal component transformation, but also represented by an MLP) to achieve rotation invariance, as well as normalisation of the input values for scale- and translation invariance. For the latter two, the CAD model from which the point cloud is generated is scaled to fit into a sphere of radius 1, situated at the origin.

In a vectorised formulation, not a scalar value, but rather a whole vector of 64 features is derived from the point cloud $P = \{p_i\}$. The neural network in $h$ is fed with the 3D coordinates and the available attributes of the point $p_i$. The output of $h$ is a vector, which is again subjected to a transformation (by means of a 64x64 matrix), from which the entries of the final vector are derived using $g$. This vector describes global features derived from the point cloud, and is fed through another MLP (to allow for the calculation of e.g. interrelations of the features) before being fed to a classifier (Qi, Su, et al. 2017).

Similar to the image recognition tasks and the 3D CNNs, the primary goal of *PointNet* is to output one label per point cloud, e.g. giving information on whether the point cloud represents a coffee mug or an airplane. But, since the features in $h$ have been calculated for every point, this information can be combined with the global feature vector $g$ by concatenating the vectors. This gives a feature vector containing both local and global information for every point, which in turn can be used for per-point classification (Qi, Su, et al. 2017).

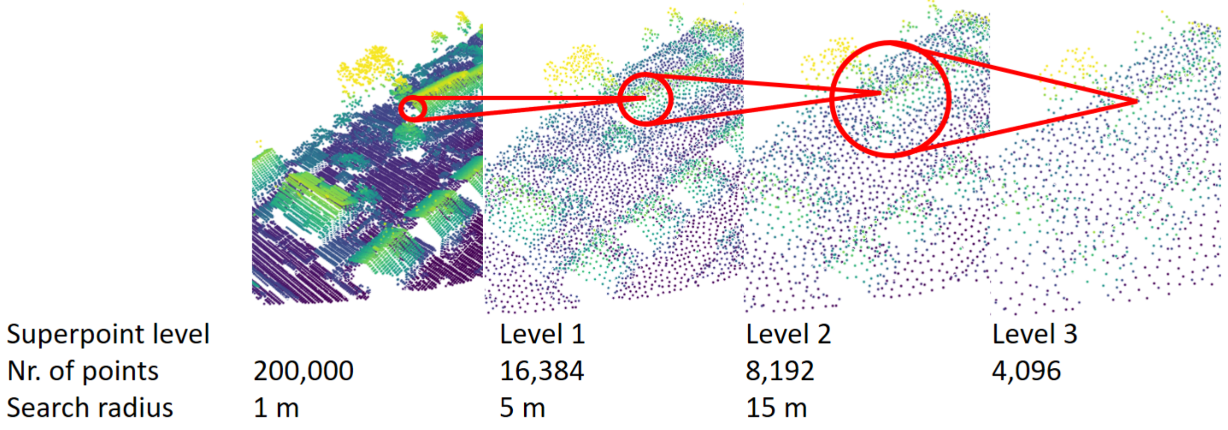
While this approach works well for simple shapes and segments such as a mug and its handle, the two scales do not provide enough information for more complex shapes. Also, for larger amounts of points, the pooling function $g$ greatly reduces the information capacity of the network. However, *PointNet* is able to derive a number of features describing local distribution from a small number of points. These features can therefore be used to describe neighbourhood properties, similar to the hand-crafted ones in Section 2.1.

By defining different levels of neighbourhoods, and subsequently calculating a *PointNet* feature vector from these for every point, many more features, one describing every level of neighbourhood, can be derived. This concatenated feature vector can then be used for per-point classification of the individual points. As it is an extension to the original *PointNet* algorithm, it is referred to as *PointNet++* (Qi, Yi, et al. 2017).

While the *PointNet* feature vector could be derived for all points of the original point cloud (based on the points in the respective neighbourhood for every point), this is not practical due to memory limitations and processing time. Instead, only a subset of the points is selected for both feature calculation and neigbourhood query. This leads to a pyramid-style approach in the follow-up paper. From the original point cloud, a number of points are selected using farthest point sampling. This method randomly selects a point and then samples the next point from the point cloud by selecting the one with the greatest distance from the first one. This operation is repeated by calculating the distance to those two points and minimising the sum of the distances. The sampling process is repeated until a predefined number of points has been reached.

On this subset of points, we will refer to them as *superpoints* of Level 1, the neighbourhood features are calculated, but based on the original points. The *superpoints* are then repositioned to the center of gravity (arithmetic mean) of the neighbourhood points and saved with their feature vector. On the next pyramid level, we call it Level 2, the points are again subsampled using farthest distance sampling, but this time from the Level 1 *superpoints*. Again, the feature vector $g$ is created for Level 2 points only, but now based on the Level 1 points. Since the Level 1 points are not original points of the point cloud, they only have the features created with *PointNet*. However, those features are now used additionally to the point distribution for the Level 2 neighbourhood descriptor. This method is referred to as *Multi-Scale-Grouping*. The sampling process is repeated 3-4 times, as shown in Figure 5. Qi, Yi, et al. refer to this as **set abstraction**.

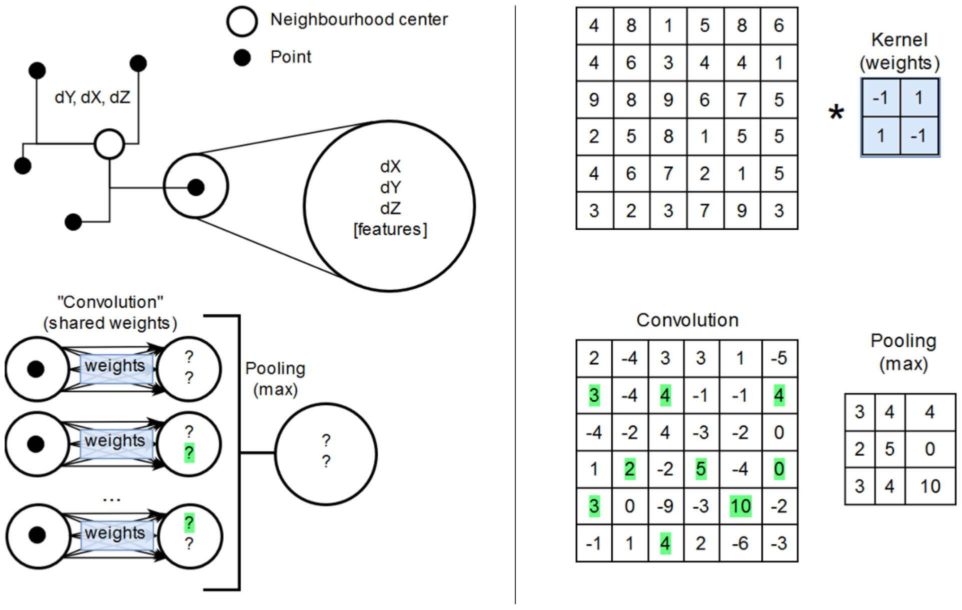
In contrast to *Multi-Scale-Grouping*, *Multi-Resolution-Grouping*, samples the neighbours from different neighbourhood sizes but always at the same positions (Qi, Yi, et al. 2017).



| Superpoint level | | Level 1 | Level 2 | Level 3 |
|---|---|---|---|---|
| Nr. of points | 200,000 | 16,384 | 8,192 | 4,096 |
| Search radius | 1 m | 5 m | 15 m | |

Figure 5: *Multi-Scale-Grouping*; pyramid approach used by *PointNet++*. The red circles symbolically show the selection radius for a *superpoint* of the next level (not to scale).

This approach can be compared to CNNs, as shown in Figure 6: *PointNet* takes points in a local neighbourhood and processes them using shared weights. For the next point, the same weights are used, similar to the convolution kernel moving to the next window position. During pooling, the maximum values in the neighbourhood are selected and used in the next layer, which is again a convolution/*PointNet*-layer, but with different weights (and window size/neighbourhood definition). The way the convolution kernel is moved over the whole image can be compared to the selection of the *superpoints* and the subsequent feature calculation.



| | | | | | |
|---|---|---|---|---|---|
| 4 | 8 | 1 | 5 | 8 | 6 |
| 4 | 6 | 3 | 4 | 4 | 1 |
| 9 | 8 | 9 | 6 | 7 | 5 |
| 2 | 5 | 8 | 1 | 5 | 5 |
| 4 | 6 | 7 | 2 | 1 | 5 |
| 3 | 2 | 3 | 7 | 9 | 3 |

Kernel (weights)

| | |
|---|---|
| -1 | 1 |
| 1 | -1 |

Convolution

| | | | | | |
|---|---|---|---|---|---|
| 2 | -4 | 3 | 3 | 1 | -5 |
| 3 | -4 | 4 | -1 | -1 | 4 |
| -4 | -2 | 4 | -3 | -2 | 0 |
| 1 | 2 | -2 | 5 | -4 | 0 |
| 3 | 0 | -9 | -3 | 10 | -2 |
| -1 | 1 | 4 | 2 | -6 | -3 |

Pooling (max)

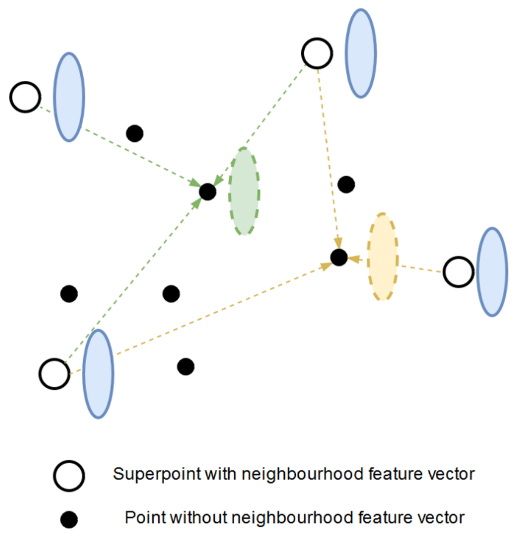
| | | |
|---|---|---|
| 3 | 4 | 4 |
| 2 | 5 | 0 |
| 3 | 4 | 10 |

Figure 6: Comparison between *PointNet* and CNNs. Blue indicates values (weights) that are learned, green indicates values that are picked during max-pooling.

While the classification of the last level *superpoints* is trivial, those points are not even a subset of the original point cloud. Therefore, a strategy for deriving feature vectors at the original point cloud level, called *feature propagation*, is necessary. Going backwards through the *superpoint* levels, the nearest three higher-level *superpoints* are selected and their feature vector is interpolated by inverse distance interpolation (see Eqn. 23) for every point. This interpolated vector is then passed through another MLP (referred to as *reverse_mlp*) and then concatenated with the existing feature vector. On the original level, this results in a stacked vector of original attributes and multiple level neighbourhoods, describing the point and feature distribution at different neighbourhood scales (Qi, Yi, et al. 2017). This method is shown for two points in Figure 7 and is referred to as **feature propagation**.

$$\vec{a}_p = \frac{1}{\sum_{i=0}^{2} \frac{1}{d_i}} \sum_{i=0}^{2} \frac{\vec{a}_i}{d_i} \tag{23}$$

Finally, this combined feature vector for each point is passed through another MLP to allow for cross-neighbourhood-features calculation. After a dropout layer and a softmax regression (see next Section), the probabilities for each class are received.



Superpoint with neighbourhood feature vector
Point without neighbourhood feature vector

Figure 7: Feature propagation by selection of the nearest three *superpoints* and inverse distance interpolation between them.

## 2.3 Neural Network Best Practices

This section deals with some common practices in neural networks, which are also applied in *PointNet* and *PointNet++*.

### 2.3.1 Dropout

Dropout is a special layer that helps to avoid overfitting. Overfitting happens when the network adjusts too well to the training dataset and fails to generalise. In a dropout layer, there is a fixed probability for every neuron that even when the threshold value is reached it will not fire the output. This ensures that no single path through the neurons is responsible for the result, but all neurons need to work together. It also ensures that single neurons do not specialise too much, so that they can "jump in" when a path "dies out" due to dropout. Dropout, of course, only has to be applied in the training phase, and can be skipped during inference (Goodfellow, Bengio, and Courville 2016, pp. 258 ff.).

### 2.3.2 Softmax

Since the values in neural networks mostly have no dimension, it can be difficult to interpret the output. When dealing with a simple classification problem, taking the index of the neuron with the highest value in the last layer can be used as a class label, as shown in Eqn. 24, where $O_i$ are the output neurons of the last layer.

$$\text{Class ID} = \text{argmax}_i O_i \tag{24}$$

However, it is sometimes useful to get a probability measure for each class. To ensure that the probabilities for each point add up to 100%, the output vector is normalized (i.e. having a vector norm of 1). This is usually done using the *softmax* function, which is shown in Eqn. 25.

$$\text{softmax}_i = \frac{\exp(O_i)}{\sum_j \exp(O_j)} \tag{25}$$

The most probable class can still be derived using the argmax function (Eqn. 24), but the softmax gives a better idea of how likely every individual class is (Goodfellow, Bengio, and Courville 2016, p. 81).

### 2.3.3 Learning Rate

When calculating weight updates with backpropagation (see Section 2.2.1) by means of the gradient in Eqn. 19, the weights are usually adapted much too quickly, which results in the optimisation overshooting the minimum. To prohibit this, a learning rate $\lambda$ is introduced, acting as a damping factor to the weight update:

$$\Delta w_{ij} = -\lambda \frac{\partial E}{\partial w_{ij}} \tag{26}$$

This learning rate is an important so-called hyperparameter of the neural network and needs to be fine-tuned. A network with a too big learning rate will always overshoot the minimum and never converge to an optimal solution, whereas the choice of an insufficient learning rate requires much more training data and might result in a local minimum for the loss function.

It is also common to let the learning rate decay, i.e. start with a relatively high learning rate (e.g. $\lambda = 0.1$) and decrease it during training with each update step. This ensures that the algorithm quickly moves to a minimum and then iteratively optimizes the solution towards this minimum (Goodfellow, Bengio, and Courville 2016, p. 425).

### 2.3.4 Optimizers

The backpropagation algorithm is a specialised form of gradient-based training algorithms for neural networks. While it was the first to allow for training of deep neural networks, higher-order methods with quicker convergence have been published, such as the Stochastic Gradient Descent (SGD). The SGD averages the weight updates over a small sample of input datasets (referred to as minibatch) to be less receptive to signals of individual datasets.

*Momentum*-based algorithms further accelerate learning by taking the curvature into account: Using Newtonian physics, a velocity vector of the first update step is calculated. The following updates do not directly influence the weights, but rather the velocity vector, which is then applied to the weights (Goodfellow, Bengio, and Courville 2016, p. 296). This increases convergence speeds and allows the algorithm to overcome minor energy barriers in the loss function.

Finally, also the learning rate $\lambda$ can be adapted during training by the optimizer itself. Perhaps the most prominent example is the *Adam* optimizer (which is short for Adaptive Moments). It uses the *momentum* approach combined with a rescaling of the gradients. This rescaling is also based on a *momentum* calculation. This means that the *Adam* optimizer makes use of first- (mean) and second-order (variance) moments of the gradients. Effectively, this is done with an exponential moving average calculated on the gradients and their squares. The *Adam* optimizer is relatively simple to use, since the default parameters for the decay rates of first- and second order moments can be used in most applications (Goodfellow, Bengio, and Courville 2016, pp. 308 f.; Brownlee 2017).

### 2.3.5 Transfer learning

When starting to train a neural network, the weights and biases need to be initialised with some values. When initialising them all with the same value (e.g. zero), the problem is that the gradients will always be the same for all neurons and the effect of the network is circumvented. Therefore, they are usually initialised with random values drawn from e.g. a Gaussian normal distribution (Goodfellow, Bengio, and Courville 2016, p. 177).

If, however, a trained model with the same network design exists, and it has been trained to perform a similar task (e.g. point cloud classification), *transfer learning*, also called retraining, can be used. Here, the learned parameters from the existing model are used as initial values. This allows the low-level neurons in the first layers to retain their "knowledge" e.g. for detecting simple shapes or lines. Only the last few layers will adapt to the changed input dataset, e.g. other numeric class labels or a different classification scheme. It can also be useful to "freeze" the lower layers, i.e. not to calculate gradients for them. This saves computational power and ensures that training with the new dataset does not result in the network "forgetting" learned properties (Yosinski et al. 2014; Géron 2017).

### 2.3.6 Minibatches

In classification tasks, a single input dataset is not representative for the problem at hand. A weight update based on an error measure from one dataset therefore has a bias towards a specific class. In the worst case, this can mean that the network fails to generalise and will always only output the last trained class for all inputs. To solve this issue, a smaller learning rate can be used. An alternative is mini-batching. Here, multiple datasets (a minibatch) are fed through the neural network, and the error measure is aggregated over them. The weight updates are then calculated to be optimal for the distribution of different inputs in the minibatch. This minibatch should therefore contain a broad representation of different classes in the training dataset. (Goodfellow, Bengio, and Courville 2016, p. 152)

### 2.3.7 Parameter search

Neural networks have a big number of values that can be adjusted to get an optimal performance, both in terms of classification results, and in terms of training time. This concerns all hyperparameters, the training data itself, as well as the architecture (depth, width) of the network.

Since the relationship between the different parameters is not clear and rather impossible to predict, the optimisation has to be done by trying different combinations. *Grid search* uses a fixed set of possible choices. From those choices, all possible combinations are sampled. While guaranteeing an optimal result (within the choices), the number of runs needed is very high. For example, when having 5 parameters and 10 choices each, the network has to be trained and evaluated $10^5 = 100,000$ times. It is clear that the number of training runs increases quickly with more choices and more parameters (Goodfellow, Bengio, and Courville 2016, p. 432).

*Random search* uses an alternative concept: To reduce the large numbers of parameter sets that need to be tested, only a random subset of those sets is chosen. While the result may not be optimal, it has been proven sufficient for most tasks. This also allows modelling of hyperparameters as a distribution, from which the samples are then drawn. For example, the dropout probability may be uniformly distributed over the interval between 0.45 and 0.55, while the learning rate may be assumed to be exponential-logarithmic (higher probabilities with lower learning rates).

If the effect of the hyperparameters can be described in a mathematical model, it is also possible to calculate the gradient on the hyperparameters to optimise them. This technique is referred to as *Model-based Hyperparameter Optimization* (Goodfellow, Bengio, and Courville 2016, pp. 435f.).

# 3 Materials and datasets

Three datasets were used for point-based classification using a DNN in this thesis. One is a large scale survey of a federal state, one a small scale ULS dataset of coniferous forest, and the third one a benchmark dataset by the ISPRS representing an airborne laser scan of an urban area.

## 3.1 Vorarlberg ALS 2011

The Vorarlberg dataset covers the federal state of Vorarlberg, with approx. 2700 km$^2$. The data has been published as Open Government Data (http://data.vorarlberg.gv.at/ogd/nutzungsbedingungen/nutzungsbedingungen.htm). Data attribution: "Datenquelle: Land Vorarlberg – data.vorarlberg.gv.at"

The dataset contains very different regions, including large cities (Bregenz, Bludenz), suburban areas, forests, valleys, rivers (Rhine, Ill) and high alpine regions. This variety of landscapes makes the dataset very interesting for research. From the exising 543 tiles of 2.5 by 2.5 km$^2$, 10 were chosen, of which 6 were used for training and 4 for validation purposes. Figure 8 shows the tiles in an overview.

The average point density of the dataset is around 20 pts/m$^2$, while the required point density was only between 4-8 pts/$m^2$ (TopoSys, Wiedenhöft, and Vatslid 2014). This results in an overall point count of about $6 \cdot 10^{10}$ points, or $8 \cdot 10^8$ and $6 \cdot 10^8$ for the training and the evaluation areas, respectively. The flights took place between 10-Mar-2011 and 18-Oct-2011, using two different scanners:

- **Harrier 56-009** with a rotating mirror resulting in a scan pattern of parallel lines
- **OPTECH ALTM Gemini** with an oscillating mirror resulting in zig-zag lines of points.

The **Harrier 56-009** was used for most of the areas. The **OPTECH ALTM Gemini** was only used in the high alpine regions in the south of Vorarlberg. Of the 10 selected tiles, the "Vergalda" validation tile and the two southernmost training tiles contain data from the **OPTECH ALTM Gemini**.

| Class ID | Description |
|---|---|
| 0 | Measured, Never Classified |
| 1 | Unclassified (processed but not assigned to a class) |
| 2 | Ground |
| 3 | Low Vegetation ($< 3$ m) |
| 4 | Medium Vegetation (3-10 m) |
| 5 | High Vegetation ($> 10$ m) |
| 6 | Building |
| 7 | Low point (noise, garage ramps, ...) |
| 8 | High point |
| 9 | Water |
| 10 | Bridges |
| 12 | Other structures (dams, ...) |
| 15 | Power lines, cable car cables |
| 16 | Masts for power lines and cable cars |

Table 5: Classification schema used in the Vorarlberg 2011 campaign.

The classification of the dataset was done semiautomatically, using *TerraScan*, *TerraSolid* and *TerraModeler*, products of *Terrasolid Ltd.*. First, gross errors like high- and low-points were detected. Subsequently, ground points were separated, before the vegetation was classified. The different vegetation classes (low, medium and high) were assigned w.r.t. the height above ground ($0 - 3$ m, $3 - 10$ m and $> 10$ m respectively) . The subsequent manual classification was based on the DTM, the preclassified point cloud and orthophotos as well as vector information from surveying. The latter included the classification of power lines and cable-car cables, masts thereof, bridges and dams. An evaluation was carried out at 100 areas of 100x100 m$^2$. Among other criteria, no class was allowed to contain more than 10% incorrectly classified points. In 2013, *BSF Swissphoto* was given the task to correct issues concerning the classification, especially w.r.t. high- and low-points and water points (TopoSys, Wiedenhöft, and Vatslid 2014). The final classes are listed in Table 5.

The dataset contains the following attributes: Amplitude/Intensity of the returned signal, number of echoes for the current pulse, and echo number within a pulse.

## 3.2 Auberg ULS 2017

This dataset was created with a *RIEGL VUX-1* Laser Scanner mounted on the *RIEGL RiCopter*, an unmanned aerial vehicle (UAV). A small area of about 360 x 230 m$^2$ was selected from the dataset, which is situated in Lower Austria, about 1 km south from the municipality of Großgöttfritz. The area covered includes a coniferous forest and a clearing and is covered with approx. 1680 pts/m$^2$. This very high point density results from the UAV-based scanner and the mostly vertical structure of the trees. The data was acquired in 2017 and represents the current state-of-the-art ULS point clouds of forests quite well.

Because the data was not classified, the existing DTM was used to create three threshold-based height classes (see Table 6). There are no buildings present in the selected area.

| Class ID | Description |
|---|---|
| 2 | Ground |
| 3 | Low Vegetation ($< 0.5$ m) |
| 4 | Medium Vegetation (0.5-2 m) |
| 5 | High Vegetation ($> 2$ m) |

Table 6: Classification schema used for the ULS Auberg dataset.

## 3.3 ISPRS 3D Semantic Labeling Contest

In 2014, the International Society for Photogrammetry and Remote Sensing (ISPRS) (Working Group II/4) published a benchmark dataset for comparing and evaluating different classification methods. While the focus for these benchmarks is 2D classification, also a sample ALS dataset exists. The area is split into three parts: A training and two evaluation areas, all situated in the village of Vaihingen/Enz, Germany. While the training dataset is available in a labelled format, the evaluation dataset only contains the coordinates and two attributes, the number of echoes of the pulse and the echo number of the current point. The training dataset also contains around 750,000 points and spans an area of about 360 by 360 meters. For a training dataset of this size, machine learning algorithms are difficult to apply due to their slow convergence.

In Summer 2018, the contest was discontinued and the reference data for the evaluation dataset was provided. The reference classes were created by the authors of Niemeyer, Rottensteiner, and Sörgel 2014, where the points were labelled manually based on an existing 2D-Classification (Gerke 2014). The available classes differ significantly from the ASPRS standard classes.

| Class ID | Description |
|---|---|
| 0 | Power line |
| 1 | Low Vegetation |
| 2 | Impervious surfaces |
| 3 | Car |
| 4 | Fence/Hedge |
| 5 | Roof |
| 6 | Facade |
| 7 | Shrub |
| 8 | Tree |

Table 7: Classification schema used in the ISPRS 3D Semantic Labelling Challenge (Gerke 2014).

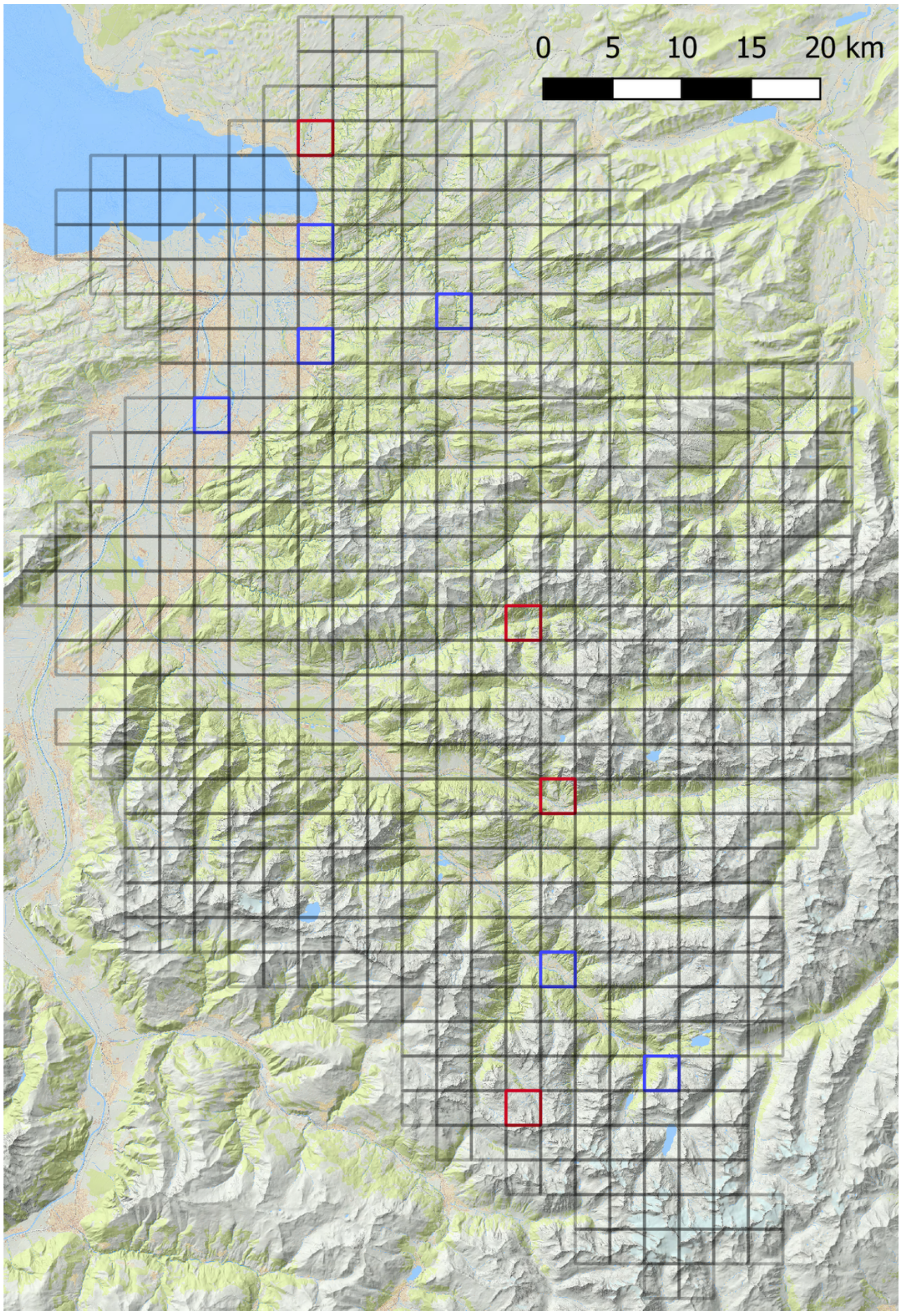


Figure 8: Overview of the Vorarlberg dataset. The areas in blue were used for training, the areas in red for evaluation and validation. From north to south, these red areas cover Bregenz/Lochau, Buchboden, Dalaas, and the Vergalda valley. The basemap is provided by the Land Vorarlberg - data.vorarlberg.gv.at

# 4 alsNet

The neural network created as part of this thesis has been named *alsNet*. While it uses *PointNet* and *PointNet++* as its basis, especially the preprocessing and the architecture of the subnets is different. This section presents the general workflow of creating, training and evaluating an *alsNet* model.

*alsNet* was implemented on basis of the `python`-Code of *PointNet++*, available at `https://github.com/charlesq34/pointnet2` (Qi, Yi, et al. 2017). It was ported to `python3` and subsequently adapted to handling ALS data. The final version of the code is available at `https://github.com/lwiniwar/alsNet`. The code uses `TensorFlow`, a machine learning/neural network library developed and maintained by Google, with support for GPU processing (Géron 2017).

## 4.1 General workflow

Using *alsNet* consists of five parts: a) Data preparation, b) Model creation, c) Training, d) Inference and e) Evaluation.

In *PointNet++*, the nearest neighbours interpolation (in the **feature propagation**), the subsampling and the neighbourhood selection (both in the **set abstraction**) have been implemented to run on a GPU to increase processing speed. This implementation (in CUDA C code) was transferred to be used in *alsNet*. However, this limits the number of points that can be processed simultaneously. Depending on the number of neurons used in the **set abstraction** *PointNet* layers, the memory required on the GPU reaches practical limits after the first MLP layers. To be able to experiment with different *PointNet* architectures, the number of input points was set to 200,000. For preprocessing, this means a tiling of the original input dataset to sets of this size. To ensure isotropy, a circular shape was selected. With a fixed spacing based on the estimated point density (see Eqn. 27), regular (2D) grid points were defined. The distance between two neighbouring centers is referred to as $\Delta_{x,y}$, the number of points as $kNN$ (=200,000), the average point density as $\rho$. The factor 0.95 ensures a margin of 5% when dealing with locally fluctuating point densities.

$$\Delta_{x,y} = \sqrt{\frac{kNN}{\pi\rho}} \cdot \frac{\sqrt{2}}{2} \cdot 0.95 \tag{27}$$

From each of those grid points, the nearest 200,000 points are selected by creating a k-D-tree and selecting the 200,000 nearest neighbours. These points are then saved to a file and represent one input dataset ("batch") for *alsNet*. On average, every point of the original dataset is covered by four of these batches. This ensures that the neighbourhood can be accurately described for all points. The number of 200,000 also eliminated the need for mini-batching, since the input should always contain data with different classes. Only a smaller batch size of e.g. 50,000 points would allow for minibatches of size 4. But smaller batches lead to problems concerning the neighbourhoods, since they cannot be accurately described at the border of the batches. Therefore, the solution with larger batches but no minibatching was chosen.

Additionally, the class counts for classes 2-6 and 9 (the most represented ones) are calculated per batch and written to a .csv-File. To allow for training limited to batches where many classes are represented, a measure of class distribution $cls$ was defined. From the relative frequencies $f_i$ of those classes, the standard deviation is calculated (see Eqn. 28).

$$cls = \sqrt{\frac{1}{n-1} \sum_i (f_i - \bar{f})^2)} \tag{28}$$

In the next step, the model is generated, initialising all variables and reserving the required space in memory. Also, the optimizer is created. In *alsNet*, an Adam optimizer with different learning rates is used. For the loss function, two options exist: (1) A basic cross-entropy function, and (2) a function that treats points that are inferred as ground points, but are non-ground in the reference, with a much higher loss. The motivation behind this is the use of the classified point cloud for DTM generation, where points omitted from the ground class have a much smaller effect than vice-versa, i.e. points in the ground class that do not represent ground. In the evaluation metrics presented in 2.1, this preference is not reflected, therefore another evaluation method should be used in this case. For the experiments, only the basic loss function was used.

After initialisation, the training phase can be started. Using the previously generated .csv-File, the training samples can be limited with respect to frequencies for specific classes (e.g. to train on batches with $> 20\%$ building points only) or to the distribution measure $cls$. Each batch of 200,000 points that fits these requirements is fed into the network, and the loss function is evaluated. The Adam optimizer then calculates the weight adjustments in the network, before the next batch is fed. Every $n$ iterations, the network takes a previously unseen batch, and tests the performance of the network. The output is also saved to a file, which allows for visual inspection of the progress of learning. These previously unseen batches might, however, already have been trained with when doing extended training with multiple passes.

For inference, the previously learned parameters are frozen. Again, point cloud batches of 200,000 points are processed at once. Every point is assigned a probability for each class, and the one with the highest probability is assigned to the point. Since the circular batches overlap each other, one point may get assigned different classes in the same batch. To overcome this issue, the probabilities are averaged over all batches a point appears in, creating a merged dataset. The assigned class is then the one with the highest average probability. While these average probabilities need not necessarily add up to 1, the measure still allows an estimation of the result class.

If the evaluation data already has a classification, it can be used to derive error measures. The estimated (most likely) class is compared to the reference classification using any of the measures presented in Section 2.1.7. This can be done for each batch separately or on the merged dataset.

## 4.2 Pyramid Scheme

As described in Section 2.2.1, *PointNet++* and *alsNet* incorporate neighbourhoods on different levels. Also, the MLP for each neighbourhood can be defined. This leads to a pyramid scheme of increasing level of abstraction. The properties of this pyramid can further be evaluated.

When looking at point count, the base is rather wide with 200,000 points. The first, second and third (and, optionally, the fourth) *superpoint* levels have much less points. These are typically powers of 2, to optimize memory consumption.

The number of neighbours for feature calculation can be combined with the search radius and the number of points in the previous level to get a measure of overlap between the neighbourhoods.

## 4.3 Neighbourhood definitions

There are different ways to define neighbourhoods. The two main categories are *kNN* (k-Nearest Neighbour) and *fixed distance query*. The first one selects a fixed number $k$ of points, by picking the one with the shortest (euclidean) distance to the query point, followed by the one with the second shortest distance, etc.. The fixed distance query takes all points within a sphere (in 3D) or a circle (in 2D) of a fixed radius $r$, so the number of points varies from one query point to another.

These two methods can also be combined, e.g. by selecting up to $k$ neighbours as long as the distance to the query point is less than $r$. This combination is implemented in *alsNet*. In memory, space for up to $k$ points is reserved, if a neighbourhood has less than $k$ points because it is limited by the maximum radius $r$, the rest is filled with zeros. The aggregation function $g$ (see Section 2.2.1) has to deal with this information. In the case of maximum-pooling, the zeros will not have an impact on the result, but with average-pooling, the zeros will have an impact. This is not necessarily a problem, however, since the fact that fewer than $k$ points are present within a radius $r$ also contains information about the neighbourhood. In *alsNet*, both max-pooling and average-pooling are used in parallel, giving two features in $g$ for every feature obtained by applying $h$ on the neighbouring points.

## 4.4 Variables in the data

As already shown in Section 3, data from ALS or ULS can contain dissimilar variables. Differences are evident in a scan pattern due to scanner types, resulting in varying point distributions in the final point cloud, overlaps, point densities, and others. An important variable is the type of land cover the data represents - e.g. an urban or a rural area. In the following experiments, the influence of some of these is investigated.

## 4.5 Variables for alsNet

There are a number of variables that can be changed when using *alsNet*. They can be grouped into two basic categories: Geometry related and neural network related variables.

- **Geometry** (per abstraction level)
  - The **number of superpoints** is the number of points for which the neighbourhood features will be derived.
  - These features are calculated on the **number of neighbours** nearest points.
  - The **search radius** further limits neighbourhood selection to avoid influence from points that are far away.
  - The number of derived features as well as the capacity of the function approximation are influenced by the **width and depth of the DNN**
- **Hyperparameters** (for the whole network)
  - The **dropout rate** can be set to avoid overfitting the training dataset.
  - Using the **learning rate**, generalisation can be achieved. Especially without the use of mini-batches, the learning rate has to be set to a low enough value.
  - To reach adequate results, a suitable **training size** has to be chosen, i.e. enough input datasets have to be provided.
  - When looking at derived data (i.e. DTMs) or at class separabilities, the **loss function** can be used to optimize the result.

# 5 Experiments

This chapter deals with the different training and evaluation sessions that were carried out. The results are presented in the Section 6. All of the experiments were carried out with the simple loss function using cross-entropy.

The compute nodes of the Vienna Scientific Cluster (VSC-3), where most of the processing was carried out, had the following hard- and software properties:

- CPU: 2x Intel Xeon @ 2.6 Ghz (8 Cores each)
- RAM: 256 GB
- GPU: 1x Nvidia GeForce GTX-1080 (8GB RAM)
- python 3.6
- TensorFlow 1.8.0
- CUDA 9.0.103
- cuDNN 7

## 5.1 Initial training on Vorarlberg dataset

Most training was carried out on the Vorarlberg dataset (see Section 3). Different architectures and hyper-parameters were used. In an initial training, an optimal architecture was found using a very small sample randomised parameter search. Tables 8 - 11 show the parameters that were used. While further experiments might lead to even better results, the parameters in Table 10 showed already very promising results and this parameter set was used for further investigation. In the tables, the following parameter names are used:

- nPoint: Number of points that are sub-sampled, for which the neighbourhood vector is calculated.
- radius: Maximum search radius for neighbourhood query.
- nSample: Number of points sampled from the previous level to calculate the neighbourhood feature vector.
- mlp: Dimensions (list of widths for each layer) of the MLP used in set abstraction.
- pooling: Pooling function(s) used to aggregate the features from the neighbourhood points
- reverse_mlp: Dimensions (list of widths for each layer) of the MLP used during feature propagation.

After a training of 3.5 billion points on each of those architectures, the best of the four architectures was found (see Section 6). This model was then evaluated on the four validation tiles. For each batch, the accuracy and the center of gravity were computed and exported to form a vector dataset. These vector datasets are shown in Section 6.2, and were used as the basis for the next experiment.

| Level | 1 | 2 | 3 |
|---|---|---|---|
| nPoint | 4096 | 2048 | 512 |
| radius | 1 | 5 | 15 |
| nSample | 32 | 64 | 64 |
| mlp | [512, 512, 1024] | [128, 128, 256] | [128, 128, 256] |
| pooling | max | max | max |
| reverse_mlp | [128,128] | [256,256] | [256,256] |

Table 8: Architecture 1

| Level | 1 | 2 | 3 |
|---|---|---|---|
| nPoint | 8192 | 4096 | 2048 |
| radius | 1 | 5 | 15 |
| nSample | 16 | 64 | 64 |
| mlp | [64, 64, 128] | [128, 128, 256] | [128, 128, 256] |
| pooling | max, average | max, average | max, average |
| reverse_mlp | [128,64] | [256,256] | [256,256] |

Table 9: Architecture 2

| Level | 1 | 2 | 3 |
|---|---|---|---|
| nPoint | 8192 | 4096 | 2048 |
| radius | 1 | 5 | 15 |
| nSample | 16 | 64 | 64 |
| mlp | [64, 512, 128] | [128, 512, 256] | [128, 512, 256] |
| pooling | max, average | max, average | max, average |
| reverse_mlp | [128,64] | [256,256] | [256,256] |

Table 10: Architecture 3

| Level | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| nPoint | 8192 | 4096 | 2048 | 512 |
| radius | 1 | 2 | 5 | 15 |
| nSample | 16 | 16 | 16 | 32 |
| mlp | [64, 128, 128] | [128, 256, 256] | [128, 256, 256] | [128, 512, 256] |
| pooling | max | max | max | max |
| reverse_mlp | [128,64] | [256,256] | [256,256] | [256,256] |

Table 11: Architecture 4

## 5.2 Training with focus on specific land cover

A spatial correlation of classification accuracies with respect to land cover was found in the first results. To further investigate this, and to address one of the research questions, a training session concentrating on building points, and another one concentrating on low vegetation points were started. It is expected that such focused training further increases accuracies, because a model does not have to adapt to different scenarios itself. To test this theory, training with a focus on batches containing at least 30 % building points or at least 50 % low vegetation points was conducted. The results are presented in Section 6.3.

## 5.3 Thinned out data

Based on the previous experiments, it was found that large building structures are not classified correctly. One possible reason for this could be that since the building dimensions are in the range the footprint of a batch, the neighbourhood definitions are too small to accurately represent the scene. To overcome the memory limitation (which limits the spatial extent of a batch), the point cloud was thinned out by the factors 2, 10 and 20, meaning every second, tenth, or twentieth point was randomly selected. From these thinned point clouds, 200,000 points were again selected. The footprint of these areas was larger, allowing large buildings to be covered completely.

Training was commenced by transferring the learned model based on architecture 3, but the search radii were adapted to fit the sparser data. The parameters are listed in Table 12. To allow for transfer learning, the other parameters of the architecture (nSamples, nPoints, MLP dimensions) were not changed.

| Factor | Batch diameter (approx.) [m] | Radius Lvl 1 [m] | Radius Lvl 2 [m] | Radius Lvl 3 [m] | Nr of batches training data | Nr of batches validation data |
|---|---|---|---|---|---|---|
| 1 (original) | 110 | 1 | 5 | 15 | 13254 | 8836 |
| 2 | 190 | 2 | 10 | 30 | 6534 | 4358 |
| 10 | 240 | 10 | 50 | 150 | 1350 | 900 |
| 20 | 540 | 20 | 100 | 300 | 600 | 400 |

Table 12: Parameters of the thinned out datasets, and radii used for architecture 3 (see Table 10)

Due to the larger footprint, the relative amount of tiles at the borders of the tiles increased, which led to more deformed batches. Those batches are not circular anymore, but rather semi-circular or even more deformed. A few examples with different thinning factors are shown in Figure 9.

## 5.4 ULS data

To both investigate data with a much higher point density as well as a different scanner, the trained network was evaluated on data from a UAV-based scan. First, the network with the original architecture was directly applied on the data. Subsequently, the search radii were changed to 0.5, 2.5 and 7.5 m for Levels 1-3 respectively. Finally, the ULS data was classified by means of vertical distance to an existing DTM of the Auberg area, and the network was retrained (i.e. transferred) to this data and then evaluated.

## 5.5 ISPRS 3D Semantic Labeling Contest

To allow for comparison with other state-of-the art classification methods, including CNNs, *alsNet* was transferred to the training data that was provided for the ISPRS 3D Semantic Labelling challenge. With 750,000 sample points, the training dataset was quite small for a machine learning approach, and the same batches were used as input to the neural network multiple times, until a saturation in training accuracy could be seen (at around 96.3 %, after about 3 billion points). This trained model is likely overfitting the training samples.

To be comparable to the other participants of the challenge, the resulting point cloud had to be merged from the tiles of 200,000 to one dataset. This was done by averaging the probabilities for each class and then reassigning the estimated class to the most probable one. This aggregation also allowed for the calculation of overlaps, i.e. counting how many times a single input point was classified in different batches.

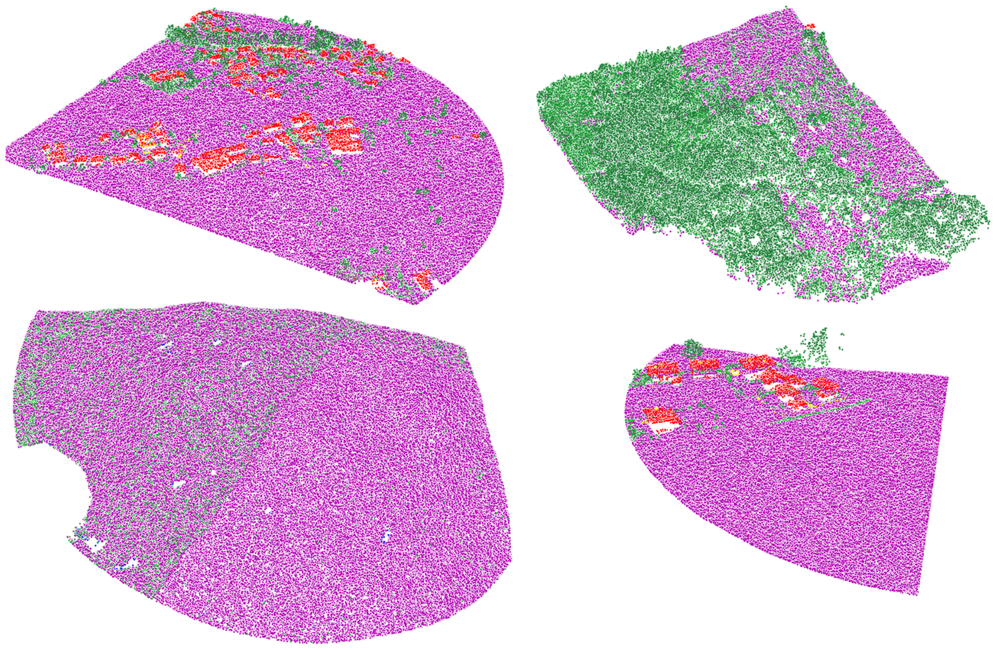


Figure 9: Deformed batches at the borders of the tiles, especially of the thinned datasets. From left to right, top to bottom: Batches 11275100_c0100 (Factor 20), 13245200_c0225 (Factor 10), 13205000_c0032 (Factor 2) and 11275100_c2209 (Factor 1). The point clouds are not to scale, but it can be clearly seen that the Factor 20 (top left) covers a much larger area (less detail) than the Factor 1 point cloud (bottom right).

# 6 Evaluation

The experiments presented in Section 5 were extensively evaluated. Some example plots are presented, as well as overall overviews of the dataset. Most processing was carried out until either the task finished or the hard run-time limit at the VSC of 3 days was reached, which led to a few white spots in the overall processing.

## 6.1 Different architectures

The first experiment concerned the different architectures. The aim was to find a "best" network architecture and use it for the following experiments. Figure 10 shows the achieved accuracies for the first 3.5 billion points. The average performance of the architectures was 76.5 %, 82.7 %, 83.4 %, and 75.9 % for architectures 1-4 respectively. Based on these results, architecture 3 was chosen for the next experiments.

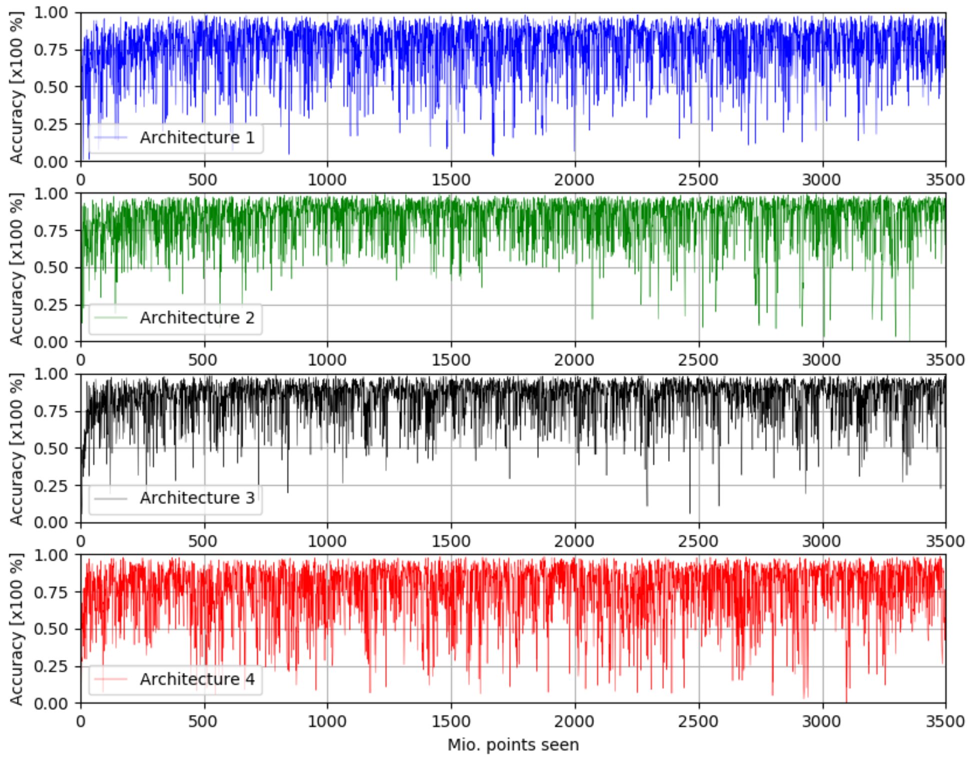


Figure 10: Accuracies achieved over the first 3.5 billion points (17.500 batches) using the different architectures.

Along with a class for each point of the original point cloud, the probability for each class was obtained. This can be used for further investigation, e.g. as input for other algorithms or to use as a measure of classification quality. The probability is visualized for four main classes in Figure 11, for a batch classified with architecture 3.

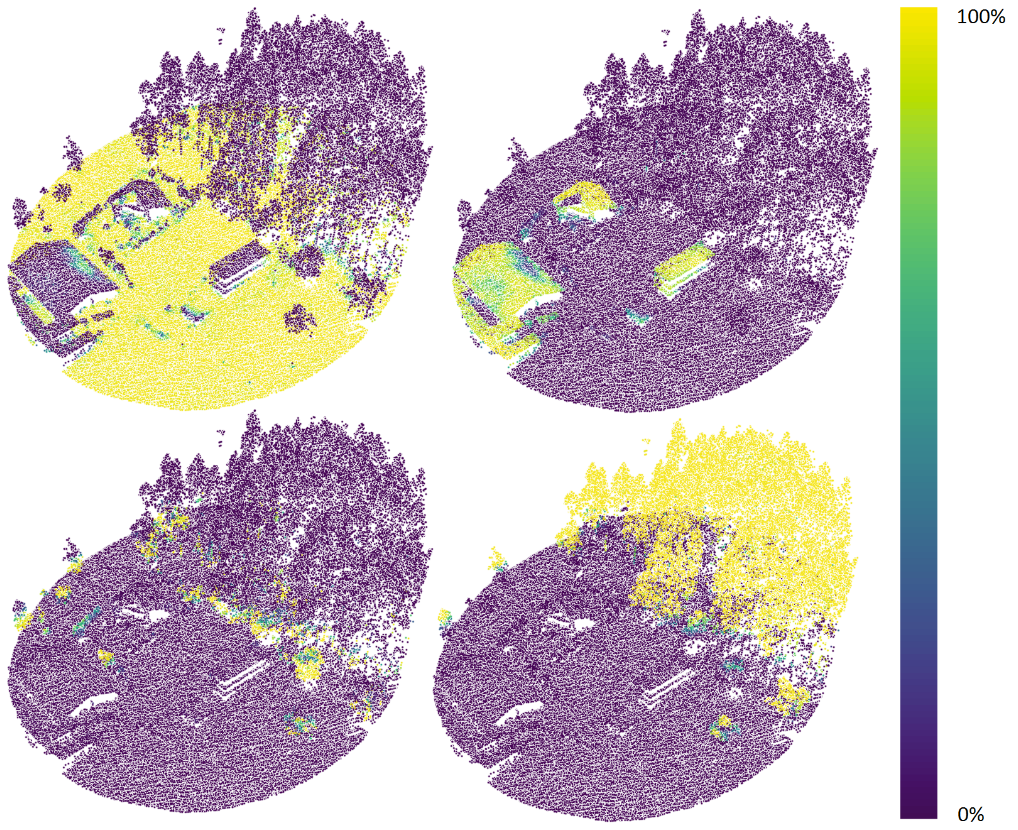


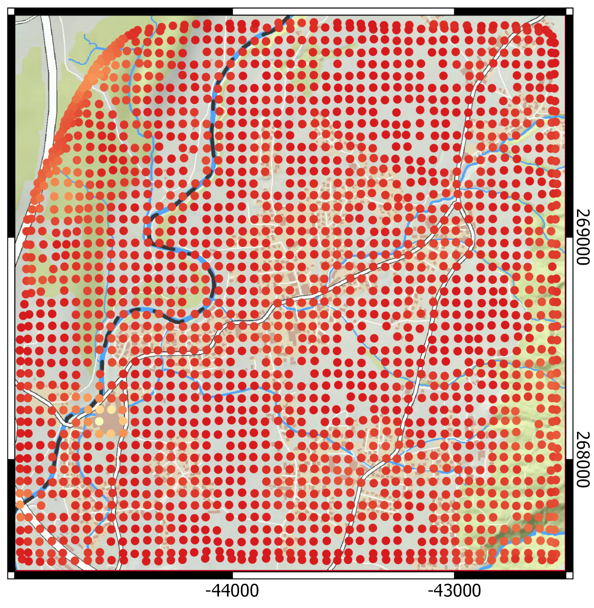
Figure 11: Probabilities for batch 11275100_c1890. Top left: ground; top right: building, bottom left: low vegetation, bottom right: high vegetation.

## 6.2 Spatial correlation of results

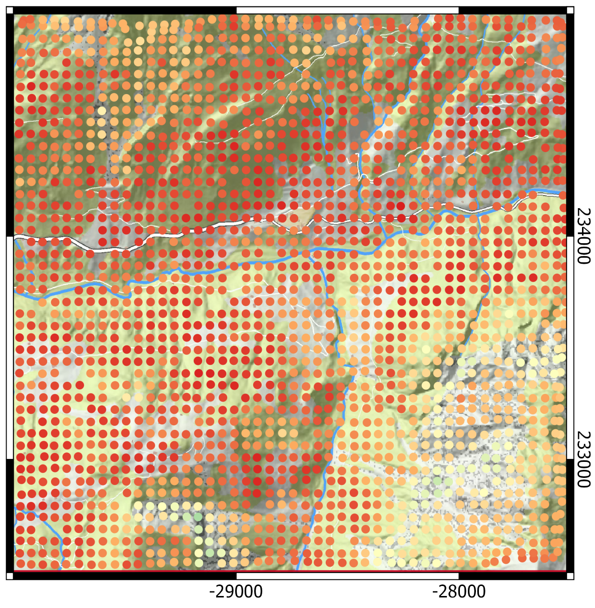
After the application of architecture 3 on the validation tiles, the results were visualized. Figures 12a - 12d show the centers for each batch and the achieved classification accuracy. The average accuracy was 95.8 %, 82.6 %, 86.7 %, and 63.6 % for Bregenz/Lochau, Buchboden, Dalaas and Vergalda valley, respectively. The distorted grid at the borders, especially at the north-western corner of the Bregenz dataset (Figure 12a) is due to the chunks of 200,000 being semi-circles at the border, or due to the edge of the flight strips, since the position of the dot does not represent the sampling center of the kNN query, but the center of gravity of the batch. The missing circles ("holes") are batches that were not processed due to the long runtime of the inference. Inference for all the batches shown in Figure 12 took three days, the hard limit for processing at the VSC.

Spatial correlation can clearly be observed with respect to the land cover of the tile, i.e. the performance is best in the urban area of Bregenz/Lochau (Figure 12a), worse in the rural areas of Buchboden and Dalaas (Figures 12b and 12c) and worst in the alpine region of the Vergalda valley (Figure 12d).

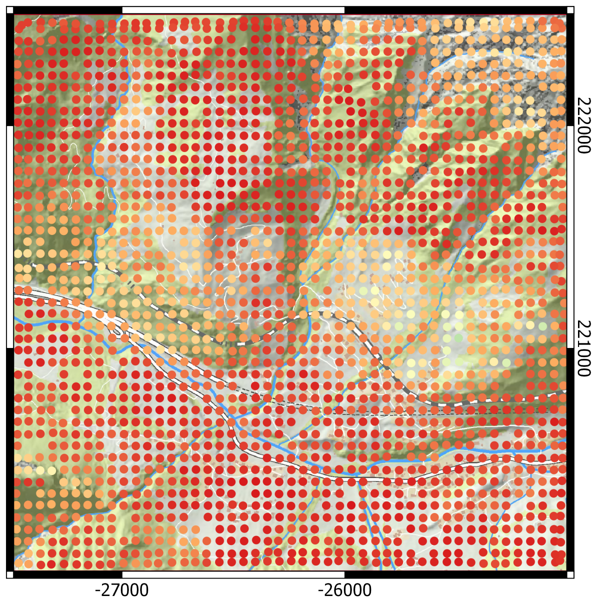
On a finer level, especially in the case of Buchboden and Dalaas, the performance can be seen to drop in rocky, steep areas, while still being on par in populated areas and forests. In Buchboden, there are two areas in the south and south-east exhibiting this behaviour. A comparison of ground-truth and estimated data for one of these batches is shown in Figure 13. Due to the high slope ($\approx 200$ m height difference over $\approx 70$ m horizontal, resulting in an average slope of 285 %), many points are misclassified, resulting in an accuracy



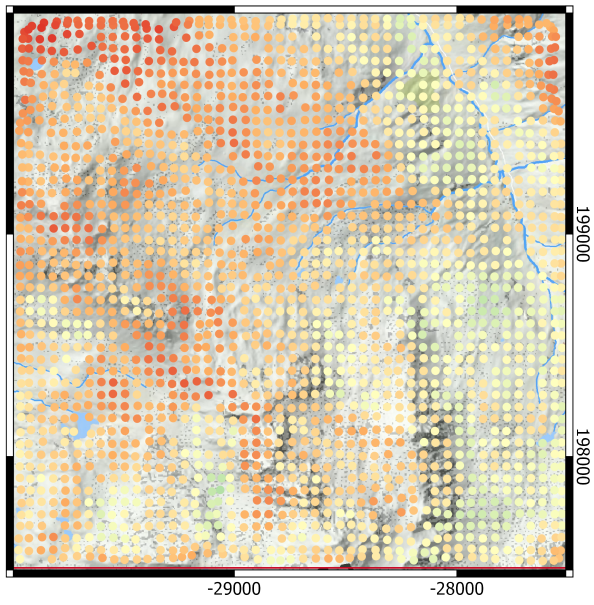
(a) Bregenz/Lochau



(b) Buchboden



(c) Dalaas



(d) Vergalda

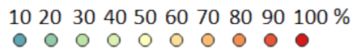


Figure 12: Accuracies per batch in the four validation/evaluation areas. Colours represent the achieved classificaion accuracy according to the colourmap at the bottom. Background map by Land Vorarlberg - data.vorarlberg.gv.at. Coordinates are in MGI/Gauß-Krüger West (EPSG:31254).

of only 25.3 % for this batch. Almost all of the points on the cliff are assigned to "Low Vegetation" in the reference data, but get classified as "Building" when using *alsNet*. The few low vegetation points on the top and bottom planes are misclassified as ground points.

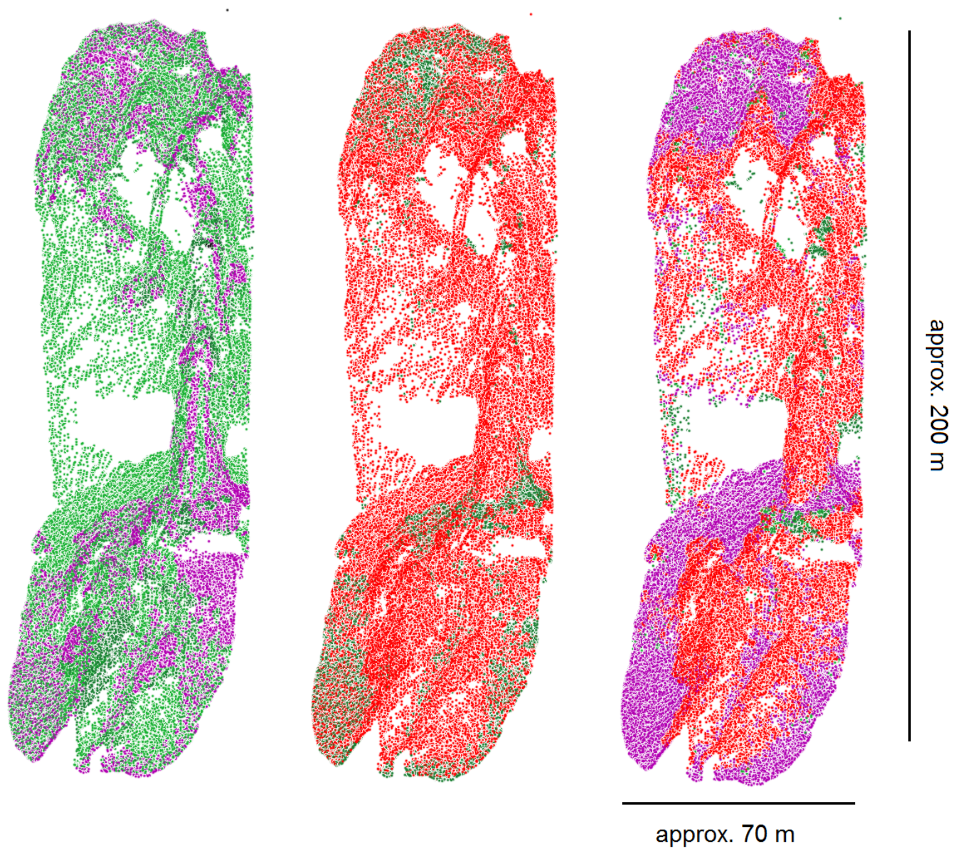


Figure 13: Ground-truth, differences and estimation (from left to right) of classes for batch 13245200_c0710 (center point at RE -29170.426 HO 232742.576). The point cloud is coloured according to Table 2.

In most of the other areas, especially in urban areas, performance is much better. Figure 14 shows such an area. Most objects, including all buildings and trees, are classified correctly. The difference image at the bottom shows the discrepancies between low vegetation (hedges) and ground, as well as the missing chimneys and power lines (in yellow in the reference). The overall accuracy for this batch was 95.4 %.

In moderately steep terrains, even with high vegetation, *alsNet* performs very accurately. In the case of batch 13235203_c1079 (shown in Figure 15), the accuracy is 97.3 %. In this vertical section plot, most incorrectly classified points are shown to be at a certain height above ground. This is the (arbitrary) boundary between low- and medium vegetation, where single points are assigned to the presumably wrong class. Additional incorrectly classified points appear at the ground surface, where some points, which are assigned to low vegetation in the reference, are classified as ground points.

## 6.3 Separability of classes

As discussed in the previous paragraphs, some class pairs are less distinguishable from each other than others. This can be evaluated by means of a confusion matrix. Figures 16a - 16d show these confusion matrix for the most prominent classes per tile.

A clear problem can be identified in the second row, first column, where points that are classified as ground

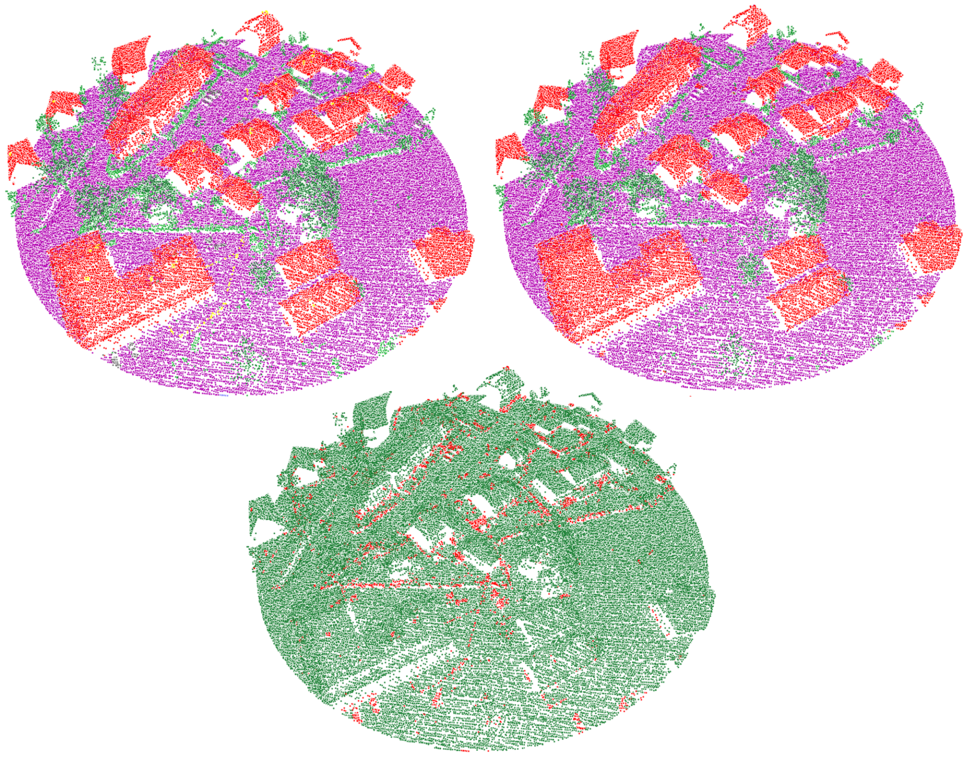


Figure 14: Ground-truth, differences and estimation (left, bottom, right) of classes for batch 11275100_c1293 (center point at RE -43530.011 HO 268760.794). The point cloud is coloured according to Table 2.

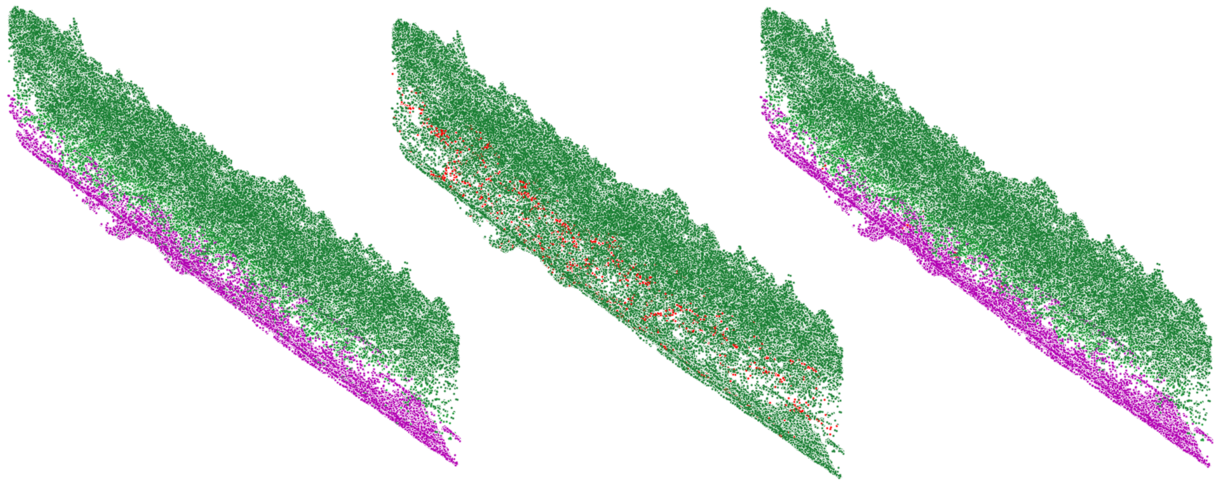
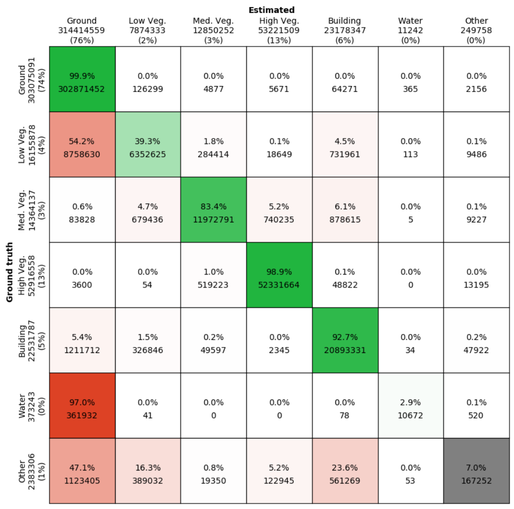


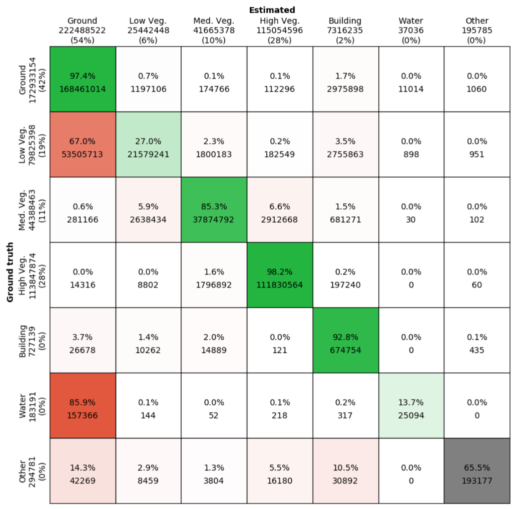
Figure 15: Ground-truth, differences and estimation (left, bottom, right) of classes for batch 13235203_c1079 (center point at RE -26291.544 HO 222386.41), shown from the side. The point cloud is coloured according to Table 2.

points while actually representing low vegetation are shown. This problem is more apparent the more rural the area is, which cannot only be attributed to the higher occurrence of low vegetation points in these



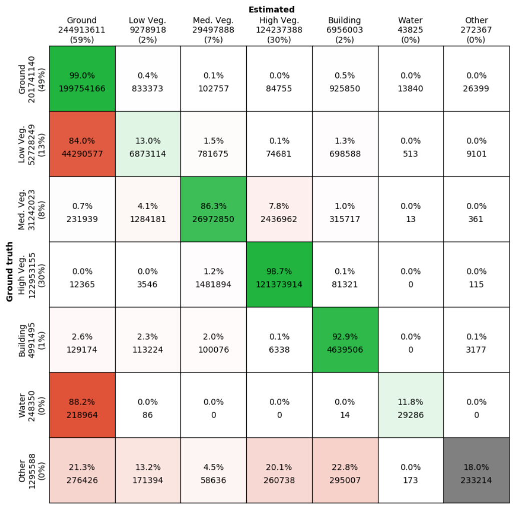
| Ground truth / Estimated | Ground 314414559 (76%) | Low Veg. 7874333 (2%) | Med. Veg. 12850252 (3%) | High Veg. 53221509 (13%) | Building 23178347 (6%) | Water 11242 (0%) | Other 249758 (0%) |
|---|---|---|---|---|---|---|---|
| Ground 303075091 (74%) | 99.9% 302871452 | 0.0% 126299 | 0.0% 4877 | 0.0% 5671 | 0.0% 64271 | 0.0% 365 | 0.0% 2156 |
| Low Veg. 16155878 (4%) | 54.2% 8758630 | 39.3% 6352625 | 1.8% 284414 | 0.1% 18649 | 4.5% 731961 | 0.0% 113 | 0.1% 9486 |
| Med. Veg. 14364137 (3%) | 0.6% 83828 | 4.7% 679436 | 83.4% 11972791 | 5.2% 740235 | 6.1% 878615 | 0.0% 5 | 0.1% 9227 |
| High Veg. 52916558 (13%) | 0.0% 3600 | 0.0% 54 | 1.0% 519223 | 98.9% 52331664 | 0.1% 48822 | 0.0% 0 | 0.0% 13195 |
| Building 22531787 (5%) | 5.4% 1211712 | 1.5% 326846 | 0.2% 49597 | 0.0% 2345 | 92.7% 20893331 | 0.0% 34 | 0.2% 47922 |
| Water 373243 (0%) | 97.0% 361932 | 0.0% 41 | 0.0% 0 | 0.0% 0 | 0.0% 78 | 2.9% 10672 | 0.1% 520 |
| Other 2383306 (1%) | 47.1% 1123405 | 16.3% 389032 | 0.8% 19350 | 5.2% 122945 | 23.6% 561269 | 0.0% 53 | 7.0% 167252 |

(a) Bregenz/Lochau



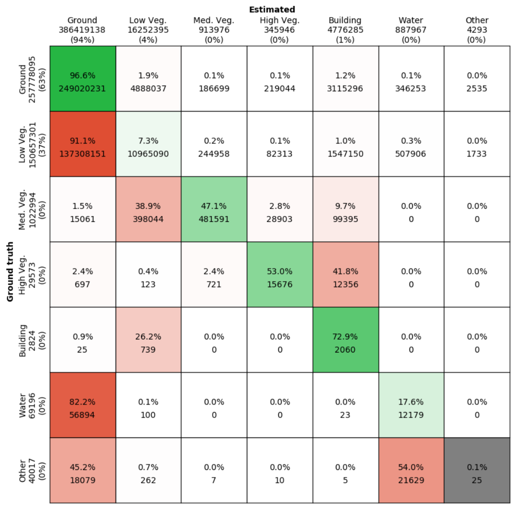
| Ground truth / Estimated | Ground 222488522 (54%) | Low Veg. 25442448 (6%) | Med. Veg. 41665378 (10%) | High Veg. 115054596 (28%) | Building 7316235 (2%) | Water 37036 (0%) | Other 195785 (0%) |
|---|---|---|---|---|---|---|---|
| Ground 172933154 (42%) | 97.4% 168461014 | 0.7% 1197106 | 0.1% 174766 | 0.1% 112296 | 1.7% 2975898 | 0.0% 11014 | 0.0% 1060 |
| Low Veg. 79825398 (19%) | 67.0% 53505713 | 27.0% 21579241 | 2.3% 1800183 | 0.2% 182549 | 3.5% 2755863 | 0.0% 898 | 0.0% 951 |
| Med. Veg. 44388463 (11%) | 0.6% 281166 | 5.9% 2638434 | 85.3% 37874792 | 6.6% 2912668 | 1.5% 681271 | 0.0% 30 | 0.0% 102 |
| High Veg. 113847874 (28%) | 0.0% 14316 | 0.0% 8802 | 1.6% 1796892 | 98.2% 111830564 | 0.2% 197240 | 0.0% 0 | 0.0% 60 |
| Building 727139 (0%) | 3.7% 26678 | 1.4% 10262 | 2.0% 14889 | 0.0% 121 | 92.8% 674754 | 0.0% 0 | 0.1% 435 |
| Water 183191 (0%) | 85.9% 157366 | 0.1% 144 | 0.0% 52 | 0.1% 218 | 0.2% 317 | 13.7% 25094 | 0.0% 0 |
| Other 294781 (0%) | 14.3% 42269 | 2.9% 8459 | 1.3% 3804 | 5.5% 16180 | 10.5% 30892 | 0.0% 0 | 65.5% 193177 |

(b) Buchboden



| Ground truth / Estimated | Ground 244913611 (59%) | Low Veg. 9278918 (2%) | Med. Veg. 29497888 (7%) | High Veg. 124237388 (30%) | Building 6956003 (2%) | Water 43825 (0%) | Other 272367 (0%) |
|---|---|---|---|---|---|---|---|
| Ground 201741140 (49%) | 99.0% 199754166 | 0.4% 833373 | 0.1% 102757 | 0.0% 84755 | 0.5% 925850 | 0.0% 13840 | 0.0% 26399 |
| Low Veg. 52728249 (13%) | 84.0% 44290577 | 13.0% 6873114 | 1.5% 781675 | 0.1% 74681 | 1.3% 698588 | 0.0% 513 | 0.0% 9101 |
| Med. Veg. 31242023 (8%) | 0.7% 231939 | 4.1% 1284181 | 86.3% 26972850 | 7.8% 2436962 | 1.0% 315717 | 0.0% 13 | 0.0% 361 |
| High Veg. 122953155 (30%) | 0.0% 12365 | 0.0% 3546 | 1.2% 1481894 | 98.7% 121373914 | 0.1% 81321 | 0.0% 0 | 0.0% 115 |
| Building 4991495 (1%) | 2.6% 129174 | 2.3% 113224 | 2.0% 100076 | 0.1% 6338 | 92.9% 4639506 | 0.0% 0 | 0.1% 3177 |
| Water 248350 (0%) | 88.2% 218964 | 0.0% 86 | 0.0% 0 | 0.0% 0 | 0.0% 14 | 11.8% 29286 | 0.0% 0 |
| Other 1295588 (0%) | 21.3% 276426 | 13.2% 171394 | 4.5% 58636 | 20.1% 260738 | 22.8% 295007 | 0.0% 173 | 18.0% 233214 |

(c) Dalaas



| Ground truth / Estimated | Ground 386419138 (94%) | Low Veg. 16252395 (4%) | Med. Veg. 913976 (0%) | High Veg. 345946 (0%) | Building 4776285 (1%) | Water 887967 (0%) | Other 4293 (0%) |
|---|---|---|---|---|---|---|---|
| Ground 257778095 (63%) | 96.6% 249020231 | 1.9% 4888037 | 0.1% 186699 | 0.1% 219044 | 1.2% 3115296 | 0.1% 346253 | 0.0% 2535 |
| Low Veg. 150657301 (37%) | 91.1% 137308151 | 7.3% 10965090 | 0.2% 244958 | 0.1% 82313 | 1.0% 1547150 | 0.3% 507906 | 0.0% 1733 |
| Med. Veg. 1022994 (0%) | 1.5% 15061 | 38.9% 398044 | 47.1% 481591 | 2.8% 28903 | 9.7% 99395 | 0.0% 0 | 0.0% 0 |
| High Veg. 29573 (0%) | 2.4% 697 | 0.4% 123 | 2.4% 721 | 53.0% 15676 | 41.8% 12356 | 0.0% 0 | 0.0% 0 |
| Building 2824 (0%) | 0.9% 25 | 26.2% 739 | 0.0% 0 | 0.0% 0 | 72.9% 2060 | 0.0% 0 | 0.0% 0 |
| Water 69196 (0%) | 82.2% 56894 | 0.1% 100 | 0.0% 0 | 0.0% 0 | 0.0% 23 | 17.6% 12179 | 0.0% 0 |
| Other 40017 (0%) | 45.2% 18079 | 0.7% 262 | 0.0% 7 | 0.0% 10 | 0.0% 5 | 54.0% 21629 | 0.1% 25 |

(d) Vergalda

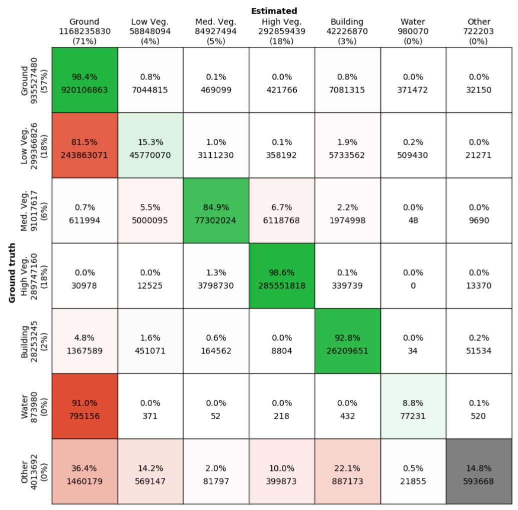
Figure 16: Confusion matrices for the four test areas. Values in the main diagonal (green) are correctly classified, the percentages are with respect to the ground truth of the respective class.

areas (although this is also the case). In fact, the separability between ground and low vegetation seems to decrease with a higher occurrence of low vegetation points. Interestingly, this inseparability only holds in one direction: ground points are very seldom classified as low vegetation points. A different choice of loss function (high loss for the misclassified low vegetation points) might change these metrics.

Another effect that can be seen in all four tiles is the poor separability of ground and water. This is probably due to the overall low amount of water points. More training with a focus on this class might improve these results.

In the Vergalda valley (Figure 16d), the confusion matrix also shows a high amount of high vegetation points being classified as buildings. The same goes for ground points, with a much higher (1.2%) rate of ground points estimated to be building points. The latter, also present in Buchboden (Figure 16b), can be explained by flat, smooth rock surfaces, having similar geometry (and amplitude) as a flat roof. Overall, there are very few building points in the Vergalda valley tile. The Vergalda valley tile is also the only one of the evaluation tiles that was flown with the OPTECH ALTM Gemini Sensor. The different scan pattern may also in part explain the worse performance in this area.

Figure 17 shows the overall confusion matrix for all four datasets, after training on architecture 3. The same problems as discussed with the individual tiles surface: low vegetation and water points are often misclassified as ground points. Especially when using *alsNet* as a pre-process in the DTM generation, this might yield a problem.



| Ground truth \ Estimated | Ground 1168235830 (71%) | Low Veg. 58848094 (4%) | Med. Veg. 84927494 (5%) | High Veg. 292859439 (18%) | Building 42226870 (3%) | Water 980070 (0%) | Other 722203 (0%) |
|---|---|---|---|---|---|---|---|
| Ground 935527480 (57%) | 98.4% 920106863 | 0.8% 7044815 | 0.1% 469099 | 0.0% 421766 | 0.8% 7081315 | 0.0% 371472 | 0.0% 32150 |
| Low Veg. 299366826 (18%) | 81.5% 243863071 | 15.3% 45770070 | 1.0% 3111230 | 0.1% 358192 | 1.9% 5733562 | 0.2% 509430 | 0.0% 21271 |
| Med. Veg. 91017617 (6%) | 0.7% 611994 | 5.5% 5000095 | 84.9% 77302024 | 6.7% 6118768 | 2.2% 1974998 | 0.0% 48 | 0.0% 9690 |
| High Veg. 289747160 (18%) | 0.0% 30978 | 0.0% 12525 | 1.3% 3798730 | 98.6% 285551818 | 0.1% 339739 | 0.0% 0 | 0.0% 13370 |
| Building 28253245 (2%) | 4.8% 1367589 | 1.6% 451071 | 0.6% 164562 | 0.0% 8804 | 92.8% 26209651 | 0.0% 34 | 0.2% 51534 |
| Water 873980 (0%) | 91.0% 795156 | 0.0% 371 | 0.0% 52 | 0.0% 218 | 0.0% 432 | 8.8% 77231 | 0.1% 520 |
| Other 4013692 (0%) | 36.4% 1460179 | 14.2% 569147 | 2.0% 81797 | 10.0% 399873 | 22.1% 887173 | 0.5% 21855 | 14.8% 593668 |

Figure 17: Overall confusion matrix. Values in the main diagonal (green) are correctly classified, the percentages are with respect to the ground truth of the respective class.

To further investigate the inter-class separabilities, the neural network was transferred to another training session focused i) on low vegetation and ii) on buildings. While the first one did not improve the separation of these classes even during training, and was therefore not investigated, the latter one was evaluated on the four validation tiles. The overall accuracy decreased in all tiles, especially in the Vergalda valley, where many batches were fully classified as building instead of ground. In Bregenz/Lochau, most batches performed slightly better than without focused training, especially the ones containing a big industry complex. In Buchboden and Vergalda, all batches had a lower accuracy when applying the focused model, whereas in Dalaas, a few batches' accuracy improved. Figure 19 shows the differences in terms of classification accuracy for tiles Bregenz/Lochau and Dalaas.

In all datasets and tests, low vegetation could not be reliably separated from ground. These two classes form two layers that run mostly parallel and with a horizontal boundary in between. The selection of the *superpoints* for these classes will have points in both layers, distributed far in horizontal and close in vertical direction, as shown in Figure 18. Here, the interpolation by selection of the nearest three points results in wrong information on the boundary between the classes (approx. halfway between the surfaces), which should actually be sought very close to the ground (i.e. only the lowest points belong to the ground class).

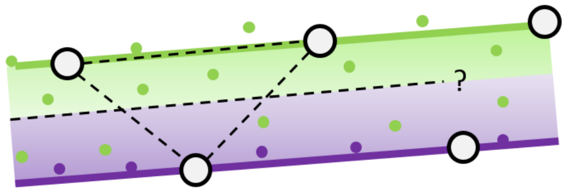
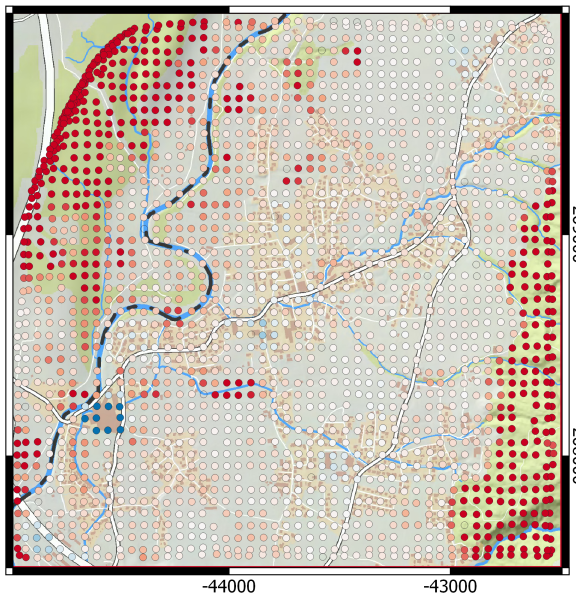
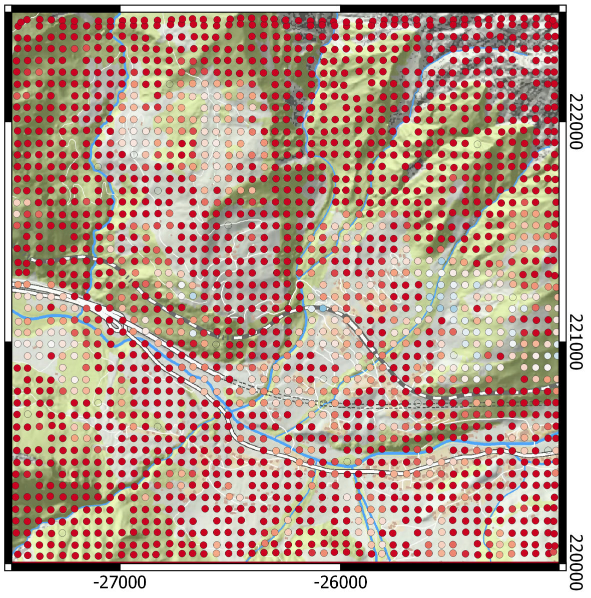


Figure 18: Schematic cross section of low vegetation and ground (green and purple lines), representing two mostly parallel surfaces. The superpoints here do not adequately represent the boundary between the classes.

Most ALS sensors nowadays provide more information than only coordinates and intensity, but also a measure of how much the outgoing pulse is broadened by the echo, i.e. how permeable the object returning the pulse is. This is a very good measure to separate these classes, since a ground point will always have a very sharp, narrow return, but low vegetation (e.g. leaves, grass) will have the pulse stretched over a longer timespan. This can be explained by differences in the returned laser signals. Ground forms a uniform surface for the whole footprint of the laser beam whereas the height distribution of leaves, etc. is much higher, and returns from objects at multiple levels will summate to form the return pulse (Höfle et al. 2008).



(a) Bregenz/Lochau



(b) Dalaas

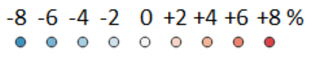


Figure 19: Differences in accuracy for the focused model and the overall model (+ means the overall model performed better). The differences are in percentage points. The other two areas (Buchboden and Vergalda) show worse performance using the focused model in all batches. Empty circles are batches processed for one model, but not for the other one. Background map by Land Vorarlberg - data.vorarlberg.gv.at. Coordinates are in MGI/Gauß-Krüger West (EPSG:31254).

## 6.4 Large-Scale applicability

The effect of the spatial dimensions of a single batch was researched by thinning out the point cloud by the factors 2, 10 and 20, and then sampling 200,000 points. The resulting accuracies are shown in Table 13.

| | Bregenz/Lochau | Buchboden | Dalaas | Vergalda | Overall |
|---|---|---|---|---|---|
| Factor 1 | 95.8 % | 82.6 % | 86.7 % | 63.6 % | 82.2 % |
| Factor 2 | 94.0 % | 82.1 % | 86.2 % | 63.3 % | 81.4 % |
| Factor 10 | 94.4 % | 78.8 % | 84.3 % | 61.7 % | 79.8 % |
| Factor 20 | 92.7 % | 75.0 % | 81.7 % | 60.0 % | 77.4 % |

Table 13: Performance in terms of accuracy with respect to different thinning factors and areas, as well as overall performance.

A clear trend towards worse performance is seen with increasing factors. This can be explained by the less detailed geometry which allows for a less accurate description of points by means of their neighbourhood. A lesser recognition in thinner datasets is not unexpected, human perception shows comparable perceptive limits.

This experiment was also conducted to research the possibility of classifying large industrial building complexes. An example is shown in Figure 20. Unfortunately, the building is almost completely classified as ground in the thinned out dataset, whereas in the original dataset, at least the edges of the building were classified accurately. In the test areas, this is the only building complex of its sort, whereas the training areas contain a few more, but maybe not enough for the neural network to be able to generalise.

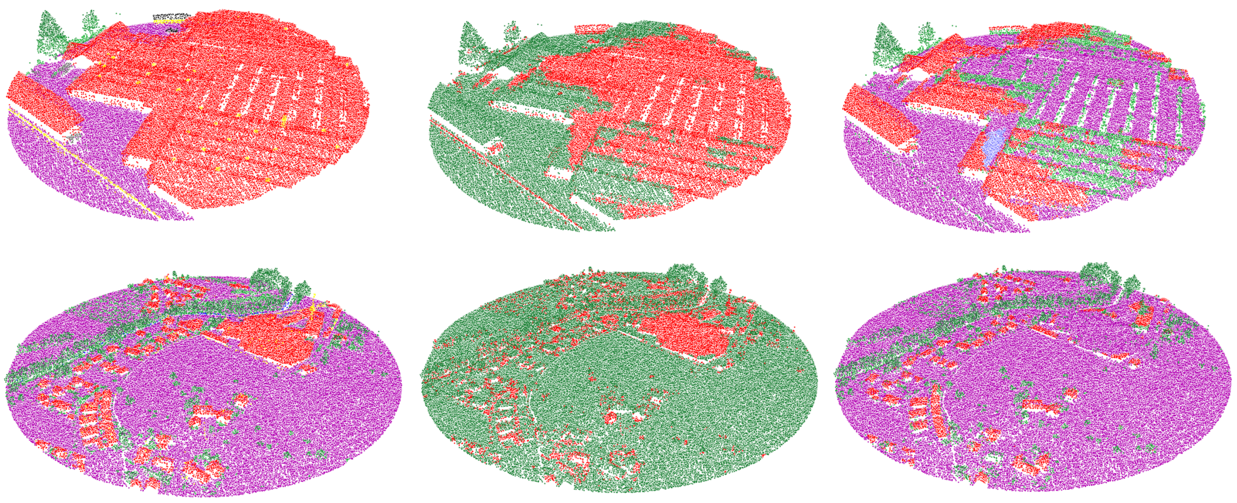


Figure 20: Comparison of classification of an industrial complex in Bregenz/Lochau. Left: reference classification, middle: differences, right: estimation. Top: Original point density, bottom: thinned out point cloud by a factor of 20. The complex shown in the original point cloud is on the top right of the thinned out point cloud.

## 6.5 Moving from ALS to ULS

This experiment was undertaken to evaluate the transferability of a trained model to a completely different point density. Neither the model from the Vorarlberg dataset nor a change in search radii resulted in satisfactory results, as Figure 22 shows. The results were different but both completely wrong, classifying many ground points and most of the trees as building, as well as adding water points. The trees themselves show a high amount of noise, especially with ground- and water points.

The transferred model that was trained on classified ULS data, however, adapted quickly (after only around 500 mio. points) to the new data. It shows that *alsNet* is able to work with ULS data as well as ALS data, if trained on the respective models.

For the Auberg dataset, the overall accuracy was 97.0 %, but as evaluation was performed on the same data that the model was trained with, the result cannot be compared to those obtained with different data sets. The performance on unseen data is expected to be lower. Also, only four classes were used in training and evaluation. The confusion matrix is shown in Figure 21 and shows similar results to the Vorarlberg dataset: Mainly the inclusion of low vegetation points in the ground class.

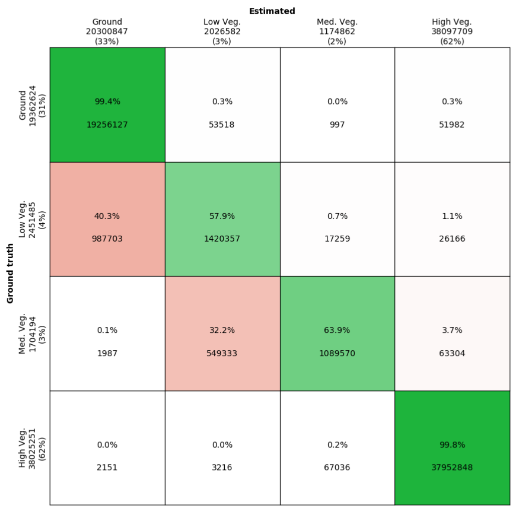


Figure 21: Confusion matrix of the Auberg ULS dataset, containing only four classes. Values in the main diagonal (green) are correctly classified, the percentages are with respect to the ground truth of the respective class.

## 6.6 ISPRS 3D Semantic Labeling Contest

For the ISPRS 3D Semantic Labeling Contest, the data was merged from the individual batches to one reference file, which was then used for evaluation. The main classes (Low Vegetation, Impervious Surfaces, Roof and Tree) were assigned quite accurately and completely. In comparison to the other algorithms that participated in the challenge, the performance ranks in the middle. 10 of the 24 submissions performed better in terms of overall accuracy. The overall accuracy of *alsNet* was 80.6 %, the highest two competitors (CNN approaches by Zhao and Pang (Nanjing Normal University) by Yang et al. 2017) resulted in overall accuracies of 85.2 % and 84.9 % respectively. The other submissions can be accessed at `http://www2.isprs.org/commissions/comm2/wg4/vaihingen-3d-semantic-labeling.html`.

The correctness, completeness and F1-Scores are listed in Table 14.

| | Power | Low Veg. | Imp. Surf. | Car | Fence/Hedge | Roof | Facade | Shrub | Tree |
|---|---|---|---|---|---|---|---|---|---|
| Correctness [%] | 77.2 | 82.0 | 89.2 | 94.8 | 83.9 | 94.9 | 77.0 | 30.9 | 66.6 |
| Completeness [%] | 64.2 | 79.1 | 91.1 | 30.1 | 4.0 | 91.3 | 34.1 | 39.5 | 84.5 |
| F1 [%] | 70.1 | 80.5 | 90.2 | 45.7 | 7.6 | 93.1 | 47.3 | 34.7 | 74.5 |

Table 14: Accuracy measures of *alsNet* for the ISPRS dataset, evaluated by Dr. Markus Gerke for the ISPRS working group IV / Commission 2.

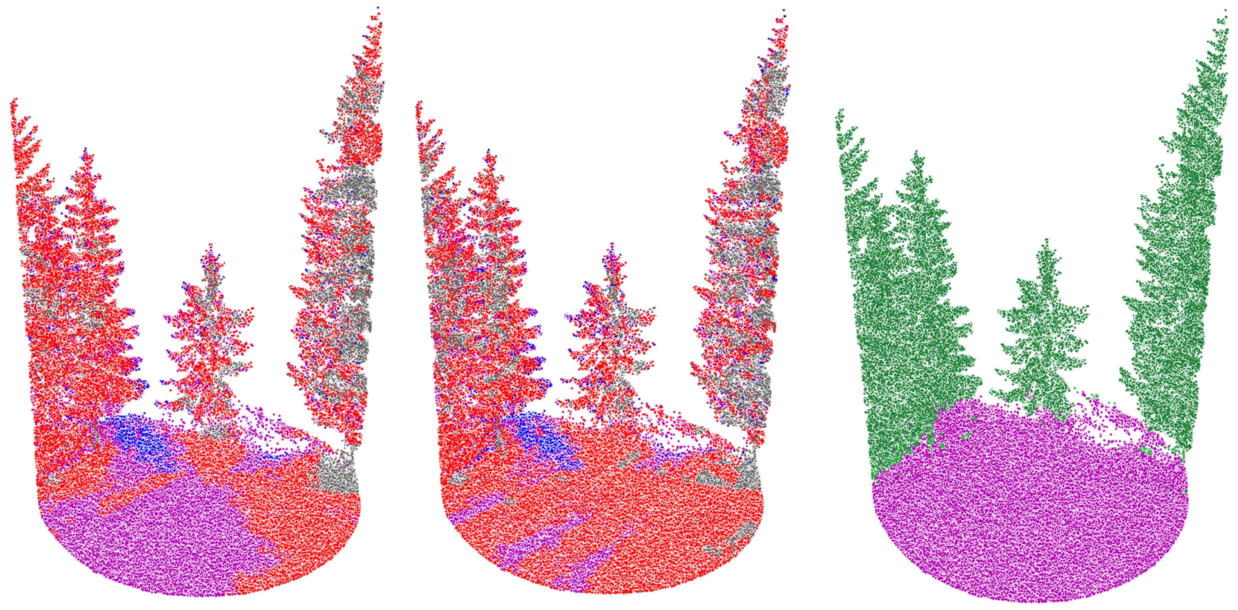
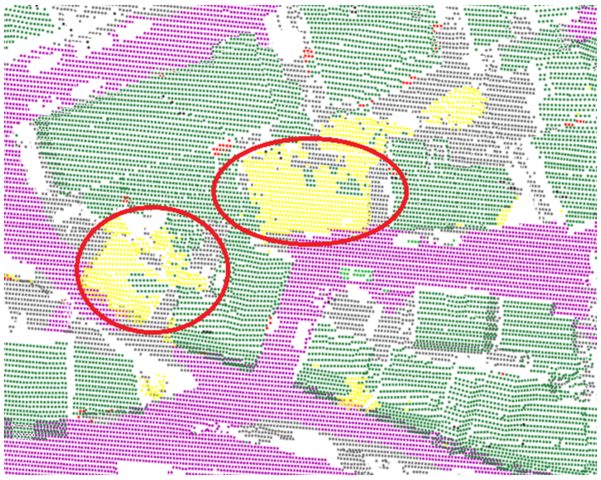
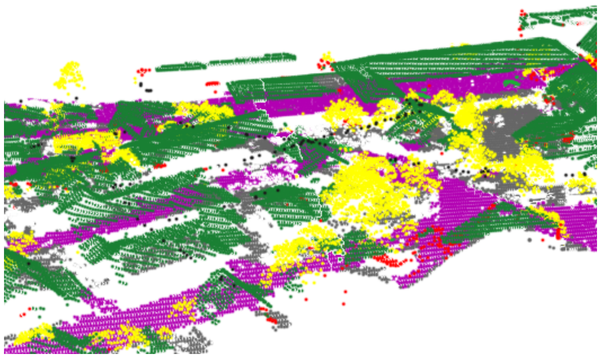


Figure 22: Comparison of different attempts to classify ULS data (from left to right): Original model (architecture 3), trained on Vorarlberg; Original model with adapted search radii; transferred and retrained model on the height-based classification scheme.

In the classified data, the effect of the multi-scale neighbourhood interpolation can be seen quite well. In the example of a building that is misclassified as low vegetation, this misinformation is carried to almost the whole building - as opposed to single points being in the wrong class. Linear structures such as the power lines are captured quite accurately. These examples are shown in Figure 23.



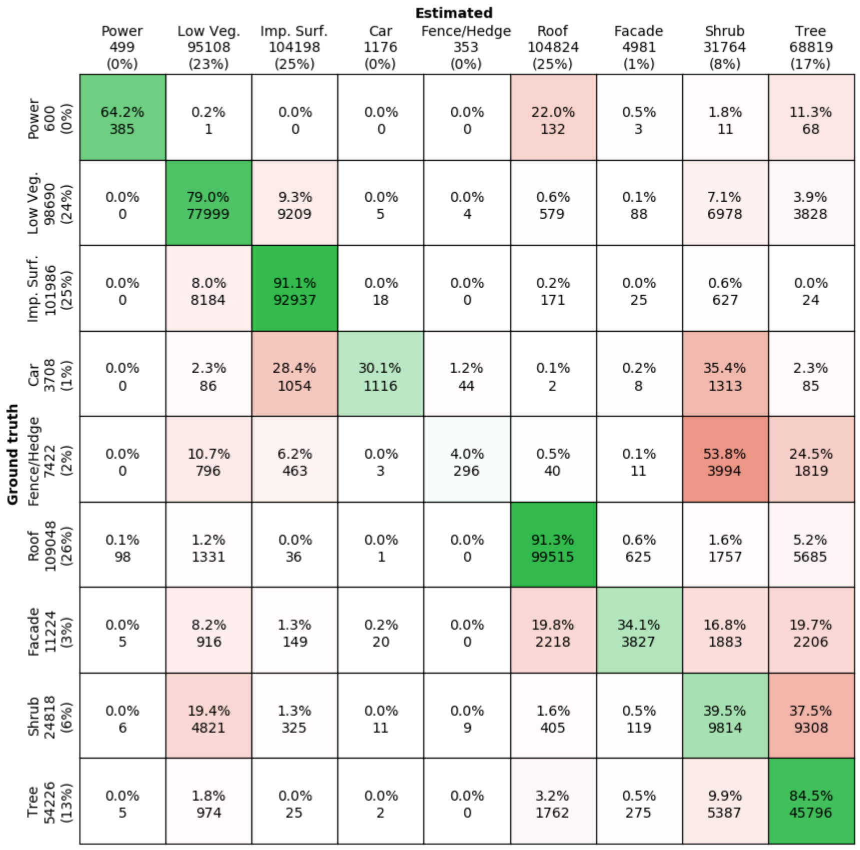
(a) Whole buildings (circled in red) classified as trees (yellow points) are mostly responsible for the high number of roof and facade points in the tree class.



(b) Power lines (black) can be classified quite well due to their linear features. The two buildings shown in Figure 23a can be seen in the background on the left side.

Figure 23: Some examples from the estimated classification of the ISPRS dataset. Due to the different classification scheme, the colours do not represent the same classes as in the Vorarlberg and Auberg datasets. Here, roofs are shown in green, trees in yellow, facades in red and power lines in black.

In the confusion matrix (Figure 24), the most problematic class can be identified as ”Fence/Hedge”, with over 50 % of the points misclassified as ”Shrub”. The accurate separation of these classes is not trivial, even with manual classification. Also, the training data already showed some ambiguities in classes, where points possibly belonging to a roof were classified as facade. Which of those classes is correct cannot be said with complete certainty. This example is shown in Figure 25.



| Ground truth \ Estimated | Power 499 (0%) | Low Veg. 95108 (23%) | Imp. Surf. 104198 (25%) | Car 1176 (0%) | Fence/Hedge 353 (0%) | Roof 104824 (25%) | Facade 4981 (1%) | Shrub 31764 (8%) | Tree 68819 (17%) |
|---|---|---|---|---|---|---|---|---|---|
| Power 600 (0%) | 64.2% 385 | 0.2% 1 | 0.0% 0 | 0.0% 0 | 0.0% 0 | 22.0% 132 | 0.5% 3 | 1.8% 11 | 11.3% 68 |
| Low Veg. 98690 (24%) | 0.0% 0 | 79.0% 77999 | 9.3% 9209 | 0.0% 5 | 0.0% 4 | 0.6% 579 | 0.1% 88 | 7.1% 6978 | 3.9% 3828 |
| Imp. Surf. 101986 (25%) | 0.0% 0 | 8.0% 8184 | 91.1% 92937 | 0.0% 18 | 0.0% 0 | 0.2% 171 | 0.0% 25 | 0.6% 627 | 0.0% 24 |
| Car 3708 (1%) | 0.0% 0 | 2.3% 86 | 28.4% 1054 | 30.1% 1116 | 1.2% 44 | 0.1% 2 | 0.2% 8 | 35.4% 1313 | 2.3% 85 |
| Fence/Hedge 7422 (2%) | 0.0% 0 | 10.7% 796 | 6.2% 463 | 0.0% 3 | 4.0% 296 | 0.5% 40 | 0.1% 11 | 53.8% 3994 | 24.5% 1819 |
| Roof 109048 (26%) | 0.1% 98 | 1.2% 1331 | 0.0% 36 | 0.0% 1 | 0.0% 0 | 91.3% 99515 | 0.6% 625 | 1.6% 1757 | 5.2% 5685 |
| Facade 11224 (3%) | 0.0% 5 | 8.2% 916 | 1.3% 149 | 0.2% 20 | 0.0% 0 | 19.8% 2218 | 34.1% 3827 | 16.8% 1883 | 19.7% 2206 |
| Shrub 24818 (6%) | 0.0% 6 | 19.4% 4821 | 1.3% 325 | 0.0% 11 | 0.0% 9 | 1.6% 405 | 0.5% 119 | 39.5% 9814 | 37.5% 9308 |
| Tree 54226 (13%) | 0.0% 5 | 1.8% 974 | 0.0% 25 | 0.0% 2 | 0.0% 0 | 3.2% 1762 | 0.5% 275 | 9.9% 5387 | 84.5% 45796 |

Figure 24: Confusion matrix for the ISPRS 3D labeling challenge. Values in the main diagonal (green) are correctly classified, the percentages are with respect to the ground truth of the respective class.

A by-product of the data merging process was the count of how often each point was classified. This over-classification is intended and can be used to examine the influence of the batch size on the classification result. Figure 26a shows the number of times each point was classified. It can be clearly seen that the points at the edge of the dataset are classified only a few times (appear in only a few batches), while the points in the middle of the dataset are covered very often. The average count of 27.2 also corresponds to the fact of $\sim$ 411,000 points of the evaluation dataset being distributed into 56 batches of 200,000 points.

In Figure 26b, a measure for the determinism of the method is shown. This is achieved by first taking the standard deviation of the class probability for each class at each point, coming from different batches, and then averaging this standard deviation over the classes. The result shows how different the batches estimate the classes for the same point in the same neighbourhood. This also shows the influence of batching on the results. The maximum discrepancies with around 16% standard deviation appear where groups of points were misclassified, e.g. the two houses that were classified as trees, and the courtyard to the left of them. In general, vegetation has a much higher mean standard deviation than e.g. impervious surface. The measure can be interpreted as how well a class can be understood by the network and may further be used to determine reliability of the estimated classes.

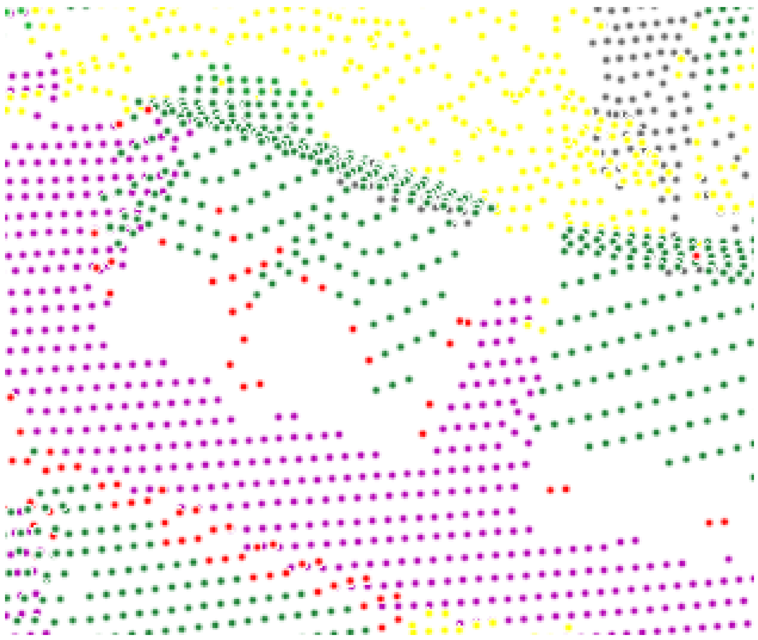
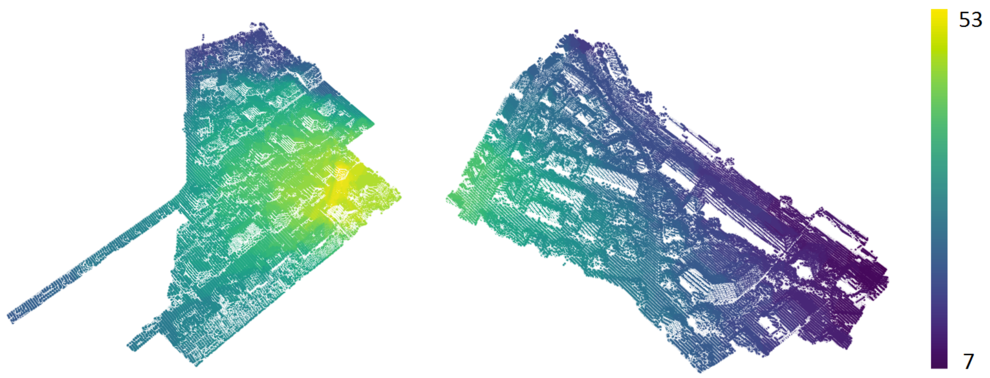
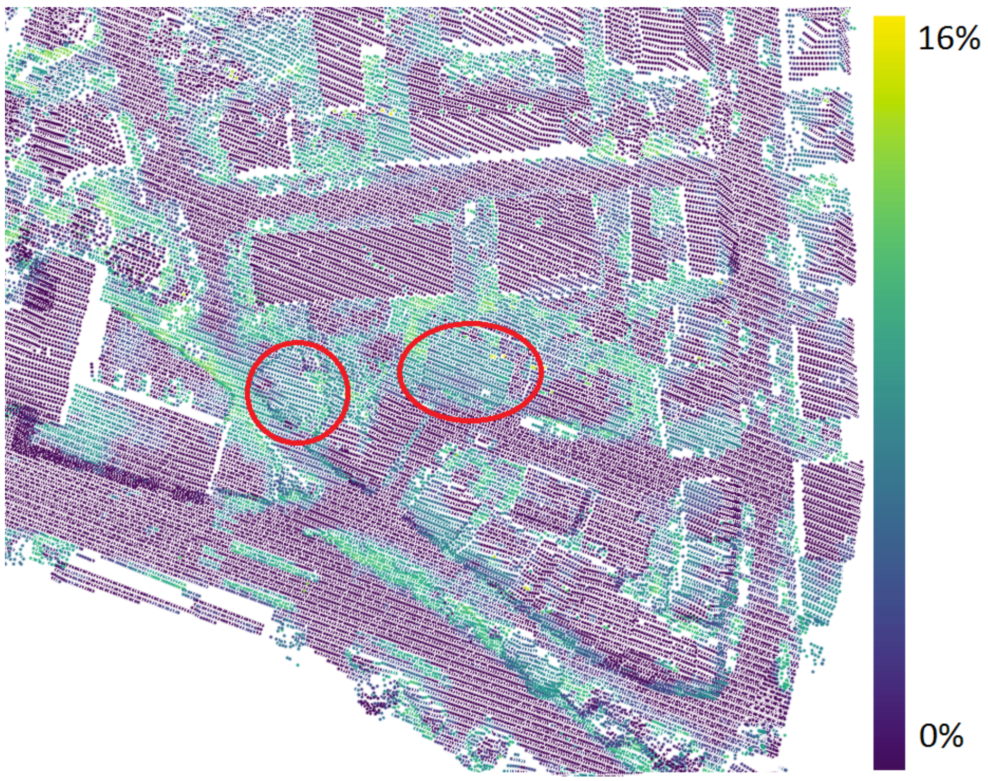


Figure 25: Reference classification of the ISPRS dataset. Issues with red facade points can be identified at the eaves, where points may as well belong to the green roof class.



(a) Number of batches a point appears in. Due to the high overlap the center points are classified multiple times.



(b) Mean variance per class for each point. This is used as a measure of the influence of batching on the dataset. Subset showing the two houses that were misclassified.

Figure 26: Additional information on the classification as a by-product of the data merging process.

# 7 Discussion

The comparison enabled by the ISPRS Semantic Labeling Contest shows that a classifier based on a deep neural network effectively learns a representation of neighbourhood geometry. This representation is mostly equivalent in performance to the structure tensor presented in Section 2.1. A major advantage to state-of-the-art classifiers is that no hand-crafted features need to be generated. This is especially useful when dealing with new classification tasks (see possible applications in Section 7.2), where it is not known *a priori*, which features best separate the classes. *alsNet* will always try to transform the input point cloud to an optimal representation of the neighbourhood.

## 7.1 Further experiments

While the main aim of this thesis was to determine the feasibility of a deep neural network as a general neighbourhood descriptor, many further questions surfaced during the evaluation of the datasets. Some of these questions as well as possible answers are presented in the following paragraphs. Their full examination, however, is out of scope for this thesis.

The comparison between ALS and ULS data, especially the attempt to transfer a model without the need for training, has shown that the sensor has a big influence on the performance of *alsNet*. This can further be evaluated. To get reliable, unambiguous and sufficient training and validation data, a simulator (such as HELIOS [1]) could be introduced. The advantage of such a dataset would be the possibility to cover the same areas with different scanners, a very rare property due to the high costs of ALS and ULS campaigns. Also, the effect of different land cover (urban, forest, bare ground) could be investigated even better.

Using simulated data to pretrain a model (e.g. for a new sensor type), and then only using a small amount of real training data also reduces the need for large-scale manual classification as reference. The basic geometric features that are learned by the model are not expected to differ much between real data and data from simulations.

Neural Networks perform best with tasks, where a mathematical function can describe the phenomenon, but the type of this function (the model) is not precisely known. Such an example could also be bathymetry. While the interaction of a laser beam on the air-water boundary of water bodies is well known and formulated in Snell's law, absorption and propagation of the beam within the water body is more complex. Especially suspended materials (silts) and floating objects in the water result in ambiguous signals. A neural network might learn to discriminate such objects and reliably determine the bottom of the water body. Still, the run-time and geometric corrections (cf. Mandlburger et al. 2015) should be carried out outside of the neural network. Here, the physical model very accurately describes reality and a neural network is neither required nor adequate.

Ambiguous signals are also very common in new-era high sensitive ALS systems, such as Single Photon or Geiger-Mode LiDAR. Due to the sensitivity of the sensor, many points are recorded in the air column between the sensor and the object. This problem, and the high amount of data generated by these systems call for a reliable classification and pre-filtering of these point clouds. While the energy and time-costs required for *alsNet* are currently not competitive, an optimization of the pre- and postprocessing steps as well as the DNN model itself is expected to realise a model where the application is feasible.

The geometric parameters presented in Section 4.5 are currently supplied by the operator of *alsNet* and are based on expert knowledge about point density and derivation of useful neighbourhood descriptors. The experiments with thinned out- and ULS data showed that the definition of these values, especially the search radii, are crucial for the success of *alsNet*. Ideally, these values would also be learned by the neural network. In this context, the importance of classification at different scales also plays a role. The thinned-out datasets

[1]HELIOS: Heidelberg Lidar Operations Simulator. See https://www.geog.uni-heidelberg.de/gis/helios.html

in Section 5.3 show that the accuracy drops when looking at fewer points - this information could, however, be used as a base for a classification on the full data. The current limitation of *alsNet*, only allowing 200,000 points at once, limits the maximum size of neighbourhood that can be considered.

## 7.2 Outlook

This chapter outlines further possibilities for application of *alsNet*. They are offered as a basis for experiment and remain sketches.

Since *alsNet* does not only produce a discrete classification, the information that is present can be used in further processing. For example, the probability of a point belonging to the ground class could be used as weights in terrain model interpolation algorithms such as the hierarchic robust interpolation (Kraus and Pfeifer 2001). Points with a high probability get assigned a large weight, points with low probability a small weight. This probability can further be investigated to be used in post-processing (i.e. looking at the second-most probable class in addition of the most probable one, and at the difference of probability) as a measure of reliability.

In foresty, a huge field of applications for ALS and ULS data, automatic stem detection is an important task (Wang et al. 2016). While no training data for stems was available for the datasets covered, one of the intermediate training results of *alsNet* is shown in Figure 27. Clearly, the tree stem and major branches have been detected and (incorrectly) assigned to the "Building" class. While being a wrong result, the correct identification of stem and major branches shows the power of neural networks. Especially when combining neural networks with hyperspectral LiDAR, the task of tree species detection may be rendered possible. In contrast to most other methods, *alsNet* does not only aggregate geometry information on neighbourhoods, but also any other attributes and any combination thereof.

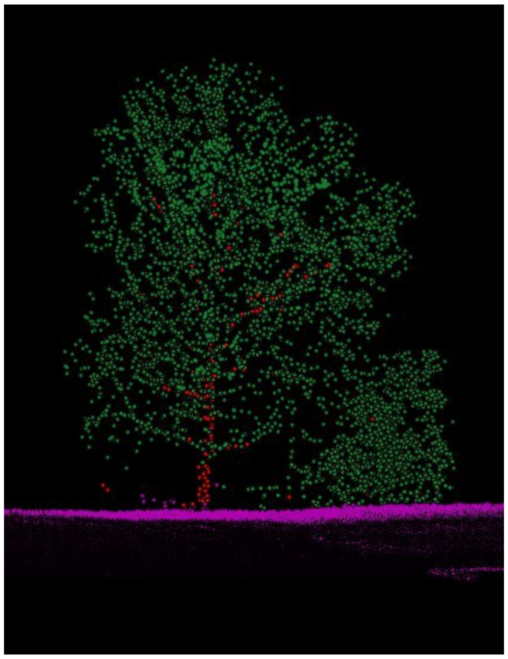


Figure 27: Tree stem detected and classified as "building" - sample output during training.

A recent trend in point cloud processing has been towards 4D-data, incorporating time as the fourth dimension. In this context, the neighbourhood definition could be extended to incorporate time and attributes can be calculated across multiple epochs. This information can then be used to e.g. classify points that are part of a deformation process such as a landslide, as well as giving a probability measure thereof. With this added fourth dimension, the manual crafting of features is incomparably more cumbersome whereas a neural network could accommodate this information simply by extending the dimensions of the layers.

Finally, one does not have to stop at point classes or their probabilities. A neural network could, for example, also be trained to the whole processing chain from raw points to a terrain model. The neighbourhood information would not be calculated for every point of the input point cloud, but rather for every grid point of the output model. The output value would then be the terrain height at the respective position. While *alsNet* focused very much on the classification task, the *PointNet* feature aggregation can be used in any task that aggregates information from the spatial distribution of points.

# 8 Conclusions

With *alsNet*, it is possible to achieve results comparable to state-of-the-art methods, but without the need for hand-crafted features. This justifies the comparably large effort needed in training and inference. Also, *alsNet* allows the combination of geometry and radiometry by deriving neighbourhood information not solely on the coordinates but also on all other attributes of the point cloud, if present. For example, a point with high reflectance may have a different significance when surrounded by points with low reflectance, than if it is just one of many points with high reflectance. When hand-crafting, the calculation of such features is often too cumbersome, especially with a large number of attributes, whereas with *alsNet*, only the size of the MLP has to be adapted to an adequate capacity of the problem.

Still, the problem of neighbourhood selection remains. With point cloud data being more readily available (e.g. through smartphones), methods to use these data without having to set many parameters would be desirable. It would be ideal if the network were able to learn an optimal neighbourhood definition on top of the parameters created from that neighbourhood.

This leads to an evaluation of the success of this thesis in terms of the research questions presented in Section 1.4:

- Is a neural network based on *PointNet* able to classify points from airborne laser scanning?
  Yes, *alsNet* is able to classify point clouds from both ALS and ULS with accuracies comparable to other state-of-the-art methods.
- What parameters of the neural network can be tuned to optimise the classification accuracy?
  There are many parameters that can be adjusted. The most important architecture parameter was the number of neurons in the layers of the *set abstraction* MLP. Other than that, the choice of training datasets proved important for efficient training of the network.
- What is the effect of different point densities, e.g. coming from a different sensor/platform, on the classification result?
  When using a trained model on a dataset from a different scanner or platform, the results are very unsatisfactory. However, even short retraining with small datasets quickly allowed the model to adapt to the changed circumstances.
- Which classes can be separated more or less efficiently, and why?
  Most issues were seen with points of low vegetation that were estimated to be ground points. This might be explained by a) the data distribution, being rather horizontal, where the boundary between these classes is also horizontal, which cannot be accurately represented by the *superpoints*, and by b) that more attributes, especially with respect to the width of the returned echo, were not present in the data. Such information is expected to allow for a better separation of these two classes.
- Is it feasible to provide a trained network for a specific sensor and land cover, e.g. by a sensor manufacturer, as a product?
  Currently, this looks like the most promising approach. Since *alsNet* is susceptible to changes in point density and/or distribution, a trained model can only be used for a certain dataset. This problem could be overcome by introducing simulated data based on the parameters of the real data. Since a user might not have enough training data available to make a machine-learning approach feasible, the sensor manufacturer could provide a trained model as a service.

## References

ASPRS (2011). *LAS Specification.* Version 1.4 - R13. https://www.asprs.org/wp-content/uploads/2010/12/LAS_1_4_r13.pdf.

Awrangjeb, Mohammad, Chunsun Zhang, and Clive S Fraser (2013). „Automatic extraction of building roofs using LIDAR data and multispectral imagery“. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 83, pp. 1–18.

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla (2017). „Segnet: A deep convolutional encoder-decoder architecture for image segmentation“. In: *IEEE transactions on pattern analysis and machine intelligence* 39.12, pp. 2481–2495.

Biosca, Josep Miquel and José Luis Lerma (2008). „Unsupervised robust planar segmentation of terrestrial laser scanner point clouds based on fuzzy clustering methods“. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 63.1, pp. 84–98.

Brownlee, Jason (2017). *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning.* URL: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (visited on 06/17/2018).

Cohen, Taco S, Mario Geiger, Jonas Koehler, and Max Welling (2018). „Spherical CNNs“. In: *arXiv preprint arXiv:1801.10130*.

Cybenko, George (1989). „Approximation by superpositions of a sigmoidal function“. In: *Mathematics of Control, Signals and Systems* 2.4, pp. 303–314.

Dietterich, Thomas G (2000). „An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization“. In: *Machine Learning* 40.2, pp. 139–157.

Dorninger, Peter and Norbert Pfeifer (2008). „A comprehensive automated 3D approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds“. In: *Sensors* 8.11, pp. 7323–7343.

Englert, Roman and Eberhard Guelch (1996). „One-eye stereo system for the acquisition of complex 3D building descriptions“. In: *Journal for Spatial Information and Decision Making* 9, pp. 16–21.

Fawcett, Tom (2006). „An introduction to ROC analysis“. In: *Pattern Recognition Letters* 27.8, pp. 861–874.

Fortuner, Brendan (2018). *Loss Functions.* URL: http://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html (visited on 06/17/2018).

Gerke, Markus (2014). *3D Semantic Labeling Contest.* URL: http://www2.isprs.org/commissions/comm3/wg4/3d-semantic-labeling.html (visited on 06/17/2018).

Géron, Aurélien (2017). *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media, Inc.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning.* http://www.deeplearningbook.org. MIT Press. ISBN: 978-0-262-03561-3.

Gross, Hermann and Ulrich Thoennessen (2006). „Extraction of lines from laser point clouds“. In: *Symposium of ISPRS Commission III: Photogrammetric Computer Vision PCV06. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences.* Vol. 36, pp. 86–91.

Haala, Norbert and Claus Brenner (1999). „Extraction of buildings and trees in urban environments“. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 54.2-3, pp. 130–137.

Hänsch, Ronny and Olaf Hellwich (2017). „Random Forests“. In: *Handbuch der Geodäsie - Photogrammetrie und Fernerkundung.* Ed. by Jan Fagerberg, David C. Mowery, and Richard R. Nelson. Berlin: Springer Spektrum. Chap. 17, pp. 603–643. ISBN: 978-3-662-47093-0.

Hearst, Marti A., Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf (1998). „Support vector machines“. In: *IEEE Intelligent Systems and their Applications* 13.4, pp. 18–28.

Hemmes, Tom (2018). „Classification of Large Scale Outdoor Point Clouds using Convolutional Neural Networks“. MSc Thesis. The Netherlands: Delft University of Technology.

Höfle, Bernhard, Markus Hollaus, Hubert Lehner, Norbert Pfeifer, Wolfgang Wagner, et al. (2008). „Area-based parameterization of forest structure using full-waveform airborne laser scanning data“. In: *Proceedings of SilviLaser* 2008, 8th.

Hu, Xiangyun and Yi Yuan (2016). „Deep-learning-based classification for DTM extraction from ALS point cloud“. In: *Remote Sensing* 8.9, p. 730.

Hubel, David H (1958). „Single unit activity in striate cortex of unrestrained cats“. In: *The Journal of Physiology* 147.2, pp. 226–238.

Hubel, David H and Torsten N Wiesel (1959). „Receptive fields of single neurones in the cat's striate cortex“. In: *The Journal of Physiology* 148.3, pp. 574–591.

— (1968). „Receptive fields and functional architecture of monkey striate cortex“. In: *The Journal of Physiology* 195.1, pp. 215–243.

Jutzi, Boris and Hermann Gross (2009). „Normalization of LiDAR intensity data based on range and surface incidence angle“. In: *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci* 38, pp. 213–218.

Kraus, Karl and Norbert Pfeifer (2001). „Advanced DTM generation from LIDAR data“. In: *International Archives Of Photogrammetry, Remote Sensing And Spatial Information Sciences* 34.3/W4, pp. 23–30.

Landrieu, Loic, Hugo Raguet, Bruno Vallet, Clément Mallet, and Martin Weinmann (2017). „A structured regularization framework for spatially smoothing semantic labelings of 3D point clouds“. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 132, pp. 102–118.

Laupheimer, Dominik (2017). „Deep Learning for the Classification of Building Facades“. MSc Thesis. Stuttgart: Universität Stuttgart.

Lawin, Felix Järemo, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg (2017). „Deep projective 3D semantic segmentation“. In: *International Conference on Computer Analysis of Images and Patterns*. Springer, pp. 95–107.

LeCun, Yann, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel (1989). „Backpropagation applied to handwritten zip code recognition“. In: *Neural computation* 1.4, pp. 541–551.

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). „Gradient-based learning applied to document recognition“. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Maas, Hans-Gerd and George Vosselman (1999). „Two algorithms for extracting building models from raw laser altimetry data“. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 54.2-3, pp. 153–163.

MacKay, David JC (2003). *Information theory, inference and learning algorithms.* Cambridge University Press.

Mandlburger, Gottfried, Christoph Hauer, Martin Wieser, and Norbert Pfeifer (2015). „Topo-bathymetric LiDAR for monitoring river morphodynamics and instream habitats—A case study at the Pielach River“. In: *Remote Sensing* 7.5, pp. 6160–6195.

Maturana, Daniel and Sebastian Scherer (2015). „Voxnet: A 3d convolutional neural network for real-time object recognition“. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on.* IEEE, pp. 922–928.

McCulloch, Warren S and Walter Pitts (1943). „A logical calculus of the ideas immanent in nervous activity“. In: *The Bulletin of Mathematical Biophysics* 5.4, pp. 115–133.

Nair, Vinod and Geoffrey E Hinton (2010). „Rectified linear units improve restricted boltzmann machines“. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814.

Niemeyer, Joachim, Franz Rottensteiner, and Uwe Sörgel (2014). „Contextual classification of lidar data and building object detection in urban areas“. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 87, pp. 152–165.

Otepka, Johannes, Sajid Ghuffar, Christoph Waldhauser, Ronald Hochreiter, and Norbert Pfeifer (2013). „Georeferenced point clouds: A survey of features and point cloud management“. In: *ISPRS International Journal of Geo-Information* 2.4, pp. 1038–1065.

Pfeifer, Norbert and Gottfried Mandlburger (2018). „Lidar Data Filtering and Digital Terrain Model Generation“. In: *Topographic Laser Ranging and Scanning - Principles and Processing, Second Edition.* Ed. by Jie Shan and Charles K. Toth. invited. Boca Raton: CRC Press, pp. 349–378. ISBN: 9781498772273.

Pöchtrager, Markus (2016). „Segmentation of Huge Point Clouds using Region Growing“. MSc Thesis. Wien: TU Wien.

Qi, Charles R, Hao Su, Kaichun Mo, and Leonidas J Guibas (2017). „Pointnet: Deep learning on point sets for 3d classification and segmentation“. In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* 1.2, p. 4.

Qi, Charles R, Li Yi, Hao Su, and Leonidas J Guibas (2017). „Pointnet++: Deep hierarchical feature learning on point sets in a metric space“. In: *Advances in Neural Information Processing Systems*, pp. 5105–5114.

Rosenblatt, Frank (1958). „The perceptron: a probabilistic model for information storage and organization in the brain.“ In: *Psychological Review* 65.6, p. 386.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). „Learning representations by back-propagating errors“. In: *Nature* 323.6088, p. 533.

Rusu, Radu Bogdan (2009). „Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments“. Dissertation. München: Technische Universität München.

Rutzinger, Martin, Bernhard Höfle, and Norbert Pfeifer (2007). „Detection of high urban vegetation with airborne laser scanning data“. In: *Proceedings of ForestSat* 7.

Schmohl, Stefan (2017). „Study on Noise Robustness of 3D Shape Recognition with Convolutional Neural Networks“. MSc Thesis. Stuttgart: Universität Stuttgart.

Schünke, Michael, Erik Schulte, and Udo Schumacher (2015). *PROMETHEUS - Kopf, Hals und Neuroanatomie - LernAtlas Anatomie*. 4th ed. Stuttgart: Thieme. ISBN: 978-3-131-39544-3.

Sobel, Irwin and Gary Feldman (1968). „A 3x3 isotropic gradient operator for image processing“. Stanford Artificial Intelligence Project (SAIL).

Sonoda, Sho and Noboru Murata (2015). „Neural network with unbounded activation functions is universal approximator“. In: *arXiv preprint arXiv:1505.03654*.

Steinhaus, Hugo (1956). „Sur la division des corp materiels en parties“. In: *Bull. Acad. Polon. Sci* 1.804, p. 801.

TopoSys, Anja Wiedenhöft, and Svein G Vatslid (2014). *Technischer Abschlussbericht LiDAR und RGB - Land Vorarlberg*.

Wallace, Luke, Arko Lucieer, Zbyněk Malenovský, Darren Turner, and Petr Vopěnka (2016). „Assessment of forest structure using two UAV techniques: A comparison of airborne laser scanning and structure from motion (SfM) point clouds“. In: *Forests* 7.3, p. 62.

Wang, Di, Markus Hollaus, Eetu Puttonen, and Norbert Pfeifer (2016). „Automatic and Self-Adaptive Stem Reconstruction in Landslide-Affected Forests“. In: *Remote Sensing* 8.12, p. 974.

Weinmann, Martin, Boris Jutzi, and Clément Mallet (2013). „Feature relevance assessment for the semantic interpretation of 3D point cloud data“. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 5, W2.

Wieser, Martin, Gottfried Mandlburger, Markus Hollaus, Johannes Otepka, Philipp Glira, and Norbert Pfeifer (2017). „A Case Study of UAS Borne Laser Scanning for Measurement of Tree Stem Diameter“. In: *Remote Sensing* 9.11. ISSN: 2072-4292. DOI: 10.3390/rs9111154. URL: http://www.mdpi.com/2072-4292/9/11/1154.

Yang, Zhishuang, Wanshou Jiang, Bo Xu, Quansheng Zhu, San Jiang, and Wei Huang (2017). „A Convolutional Neural Network-Based 3D Semantic Labeling Method for ALS Point Clouds“. In: *Remote Sensing* 9.9, p. 936.

Yosinski, Jason, Jeff Clune, Yoshua Bengio, and Hod Lipson (2014). „How transferable are features in deep neural networks?“ In: *Advances in neural information processing systems*, pp. 3320–3328.

Zhang, Jixian, Xiangguo Lin, and Xiaogang Ning (2013). „SVM-based classification of segmented airborne LiDAR point clouds in urban areas“. In: *Remote Sensing* 5.8, pp. 3749–3775.

Zhu, Zhe, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu (2016). „Traffic-sign detection and classification in the wild“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2110–2118.