Classification of 3D Point Clouds using Deep Neural Networks

LUKAS WINIWARTER[1,2] & GOTTFRIED MANDLBURGER [1,3]

Abstract: *The use of deep neural networks, i.e. neural networks with multiple layers, has significantly improved the accuracy of classification and regression tasks in many disciplines, including computer vision. Here, especially convolutional neural networks (CNNs) that are able to extract features representing local neighborhoods in 2D images are employed. The direct transfer of this approach to 3D point cloud data requires a rasterization and manual selection of attributes that are represented in the raster.*

We present a neural network based on PointNet by QI et al. (2017a) that instead directly uses 3D point clouds, along with any attributes attached to the points, as input, considering neighborhoods of points. On the ISPRS 3D Semantic Labeling Benchmark Vaihingen, we achieve an overall accuracy of 80.6 %. On the airborne laser scanning point cloud of the Federal State of Vorarlberg, Austria, we achieve accuracies in urban areas of up to 95.8 %, observing a strong correlation with land cover. Hence, we conclude that our approach learns a representation of neighborhood mostly equivalent to the current manual selection process, with the option of further improvement by use of additional data such as echo width or RGB information.

1 Introduction

Topographic 3D point clouds representing the surface of the earth and objects thereon can be acquired with different methods. The most established ones are Dense Image Matching (DIM) based on multi-view stereo images and topographic laser scanning or LiDAR (Light Detection And Ranging), respectively. Both methods lead to big amounts of data, often in the order of several terabytes. Especially with advances in sensor technology and automation concerning image matching, these volumes have grown even further. To make use of these data, classification is key. By classification, we refer to the assignment of semantic labels to points, in this case, on a per-point basis (OTEPKA et al. 2013). The nature of these labels depends on the task at hand. In our case, they represent types of objects commonly found on the earth's surface: vegetation, buildings, water, ground, and others (ASPRS 2011).

Section 2 shows that neural networks have been used in point cloud classification, however, current methods do not fully exploit their potential (YOUSEFHUSSIEN et al. 2018). Like with other classifiers, existing neural network approaches rely on the user to select and calculate a number of features characterizing the local neighborhoods of every point. A study on the relevance of these

[1] Technische Universität Wien, Department für Geodäsie und Geoinformation, Gusshausstraße 25-27, A-1040 Wien

[2] Universität Heidelberg, Geographisches Institut, Im Neuenheimer Feld 368, D-69120 Heidelberg, E-Mail: lukas.winiwarter@uni-heidelberg.de

[3] Universität Stuttgart, Institut für Photogrammetrie, Geschwister-Scholl-Straße 24D, D-70174 Stuttgart, E-Mail: gottfried.mandlburger@ifp.uni-stuttgart.de

features was published by WEINMANN et al. (2013), showing that the structure tensor, i.e. the matrix of the second-order moments of the point distribution, is key. To achieve scale-, translation- and rotation-invariance, the moments are centralized and the eigenvalues of the structure tensor are calculated. Ratios or other simple algebraic combinations of these eigenvalues represent measures like *linearity* or *planarity*, with values ranging between zero for non-linear/non-planar objects to one for elongated/planar distributions of points, respectively.
The user does not only have to decide on the specific features, but also on the neighborhood and attributes, these features are calculated from. For the neighborhood, two main approaches exist: a fixed distance (spherical) query, or a fixed number (k-Nearest-Neighbors, kNN) query (WEINMANN et al. 2014). Other attributes in the point cloud may come from the sensor itself (e.g. amplitude, echo width, echo number etc.) or from data fusion (e.g. RGB values from an orthophoto). In addition to the spatial distribution of the points, also the spatial distribution of these attributes can be considered for classification (OTEPKA et al. 2013).

2 State of the Art

Many different classification algorithms are commonly applied to point clouds (GRILLI et al. 2017). Here, we present two prominent machine learning classifiers, Random Forest and Support Vector Machine. Subsequently, we introduce neural networks, including their application on point clouds.

2.1 Supervised Classification

Supervised classification is based on training data. Training data is input data for the classification, where the expected result, which is the class of the data point, is known *a priori*. This reference classification can be provided by other classification algorithms, by manual classification, by a combination of the two, or by simulated data. A classifier trained on such data will attempt to approximate this reference as good as possible by minimizing the training error. This sometimes leads to questionable results, especially if the reference classification contains errors or inconsistencies. The latter is often referred to as classification under label noise. Furthermore, two classes that cannot be separated in feature space will result in outputs where the classes are mixed up or the class with less presence in the training data (i.e. prior probability) does not appear in the output (FRÉNAY & VERLEYSEN 2014).

2.2 Support Vector Machine

Support Vector Machines, or SVMs, are binary classifiers that separate a dataset into two classes. The basic idea is to find a best linear separation boundary (i.e. a hyperplane) between the classes in feature space. Since the restriction to linearly separable classes limits this classifier, the input data is transferred to a space of higher dimensionality. Here, a better linear separation might be possible. To save on computation cost, the "kernel trick" is used, where the data does not have to be transformed itself. In the high-dimensional space, the hyperplane is fitted so that the distance

from the hyperplane to the nearest feature point (the support vector) of each class is minimized (HEARST et al. 1998).

2.3 Random Forest

Random forests are based on decision trees. In a decision tree, an input dataset is fed through threshold classifiers represented by Boolean expressions, so-called nodes (e.g. "amplitude > 50"). Every node has two outputs (true/false), which are connected to new nodes. The end-nodes, called leaves, represent the class the dataset is assigned to, allowing more than two target classes.
Classic decision trees require threshold settings in every node. They also do not provide any measure of uncertainty for their classes. Random forests consist of multiple trees, each having been picked as the best one from a number of randomly generated trees. The input dataset is then passed through all trees simultaneously, which vote on the class. This leads to non-linear boundaries as well as a probability measure of the output class. For example, if 60 out of 100 trees vote for class A, the probability of the point belonging to class A is 60 %. Even though single decision trees are not very good classifiers, the aggregation over the forest leads to reasonable results (BREIMANN 2001).

2.4 Neural Networks

Neural networks represent another type of machine learning algorithms. They emulate the way the nervous system works by connecting multiple atomic units, the neurons. Every neuron has a number of input values I from the previous neurons, which are aggregated into an output O, that is passed to the next neurons. The simple neural model of the *Perceptron* uses a weighted sum of the inputs and a bias. The weights w and the bias b are not known *a priori.* They are initialized randomly and are then adapted in the machine learning process, i.e., during training (ROSENBLATT 1958). To overcome the linearity restraint of the weighted summation, the output value is fed through a non-linear activation function a, such as a sigmoid function, before it is passed to the next neuron. A single neuron can be described mathematically as in Eqn. 1 (GOODFELLOW et al. 2016, p. 171).

$$O = a(\sum_{i=1}^{n} w_i \cdot I_i + b) \quad (1)$$

In a multi-layer perceptron, a very basic neural network, the neurons are organized in layers, where every neuron from the first layer is connected to every neuron from the second layer, and so on. The first layer is referred to as the input, the last layer as the output of the network. All layers in between are so-called hidden layers. The amount of hidden layers defines the depth of the network, coining the term "Deep Learning", while the amount of neurons per layer the width, which may vary from layer to layer (GOODFELLOW et al. 2016: 169). In this simple structure, a whole layer can be represented mathematically as a matrix operation with the weights summarized in a coefficient matrix A, as shown in Eqn. 2. The activation function a is applied to every entry of the vector resulting from the matrix multiplication. For multiple layers, this operation is chained together.

$$\vec{O} = a\left(A \cdot \vec{I} + \vec{b}\right) \quad (2)$$

For training, a reference sample is fed through the network, and the output is compared with the expected output. From this, an error measure ("loss") is created, e.g. using a cross-entropy function. The effect of the weights on this loss can be calculated as the partial derivative of the loss with respect to the weights and biases. For this, repeated application of the chain rule, as well as a differentiable activation function are required. Using gradient descent, the weights are adapted in a way to find a minimum value for the loss (GOODFELLOW et al. 2016: 82 f.). Advanced optimization algorithms such as the Adam optimizer incorporate a momentum approach, where not the weights directly are updated, but rather a velocity vector, which then, in turn, changes the weight values. The Adam optimizer also includes a rescaling of the gradients, making use of the variance of the weight velocity updates, i.e. the second order moments (GOODFELLOW et al. 2016: 308).

2.5 Convolutions and Convolutional Neural Networks

A convolution is a mathematic operation where a function is moved over another function. The result of the convolution is the integral of the multiplied functions with a given offset (i.e. movement position) for one of the functions. In discrete form, the integral is replaced by a sum, as shown in Eqn. 3. Convolutions are commutative, i.e. the order of the functions does not matter.

$$(f * g)(x) = \sum_x (f(x) * g(t - x)) \quad (3)$$

In practice, one function, the so-called kernel, is limited in domain, i.e. the summation is only over a small part of the whole function. Furthermore, the concept can easily be extended into 2D, where the kernel is represented by a small window, a matrix that is moved over the data, e.g. an image (GOODFELLOW et al. 2016, pp. 237f.). These kernels extract information on the local neighborhood of a pixel, e.g. the slope using the well-known Sobel kernel. The values in the kernel are commonly referred to as weights.
In Convolutional Neural Networks (CNNs), the kernel is not selected for a specific purpose *a priori*, instead the weights are adjusted in the training phase. Since the kernel is moved over the input data, the weights are shared between many input values. In a neural network, this leads to fewer values that have to be trained. LECUN et al. (1989) applied this concept on multiple levels (incorporating neighborhoods of different extents) to classify handwritten ZIP codes with previously unseen success. A local pooling function is applied to extract relevant data after each convolution. This function reduces the size of the image by selecting relevant information in a small region. For example, every 2x2-pixel submatrix might be replaced by the maximum value, as shown in the example in Figure 1 (GOODFELLOW et al. 2016: 339-345).

2.6 PointNet

Point clouds are inherently unordered, irregular sets of points in space. These properties hinder the direct input of points to a neural network, since the order of the neurons matters very much. QI et

al. (2017a) published a method called PointNet able to deal with these properties. They approximate a function *f* acting on a point set by two functions *g*, and *h*, where *g* is a symmetric function (e.g. the maximum or the mean) and *h* is a function applied on the (relative) coordinates and attributes of every point, as shown in Eqn. 4. Furthermore, *h* is represented by a multi-layer perceptron.

$$f(\{x_1, x_2, \dots, x_n\}) = g\big(h(x_1), h(x_2), \dots, h(x_n)\big) = g(h_1, h_2, \dots, h_n) \qquad (4)$$

This ensures that every input point x_i is treated equally, and the output does not depend on the order of the input points. The result of *h* and *g* are vectors, representing local features aggregated from different points in the neighborhood. In the case of the max-function, the first element of *g* may have the value from point *p*, the second element from point *q*, and so on.
We can compare PointNet to a CNN. In both cases, a local neighborhood is used to calculate new features. As Figure 1 shows, weights are applied to the local neighborhood, before a pooling operation selects relevant or representative information from the convoluted neighborhood. These similarities suggest that PointNet may profit from the same benefits as a CNN in applications where local surroundings of a data point (i.e. a pixel or a 3D point) are important.

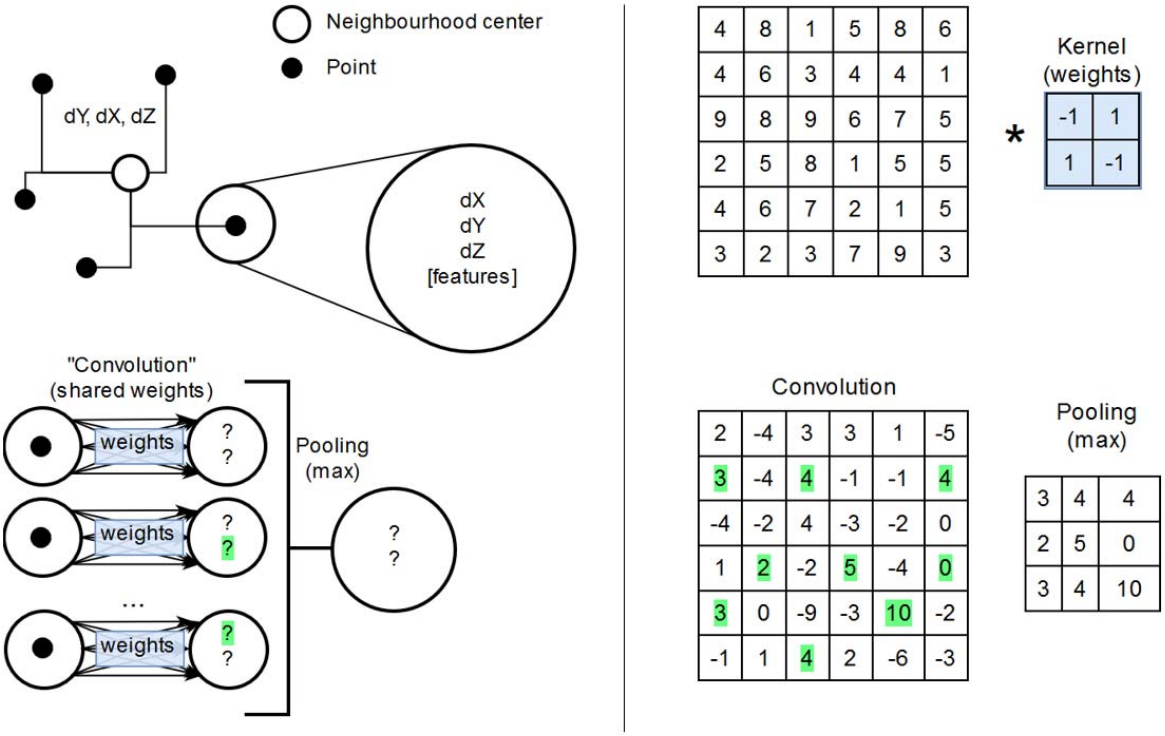

Fig. 1: Comparison of PointNet (left) and a Convolutional Neural Network (right). Blue values are learned in the training phase, while green values are selected to be representative by the pooling function.

PointNet allows the creation of a neighborhood feature vector for a local set of points. This vector can be created on a number of scales (i.e. different neighborhood input sets) for every point in the point cloud. In PointNet++, QI et al. (2017b) suggested, however, that the values representing large-scale neighborhoods are not subject to volatile changes. Therefore, they sub-sample the point cloud using farthest distance sampling to save processing cost. On this sub-sampled point cloud, the neighborhood is defined with respect to the original point set. This process is repeated on multiple levels, each using the sub-sampled points of the previous level for the neighborhood aggregation, as shown in Figure 2. In addition, the local neighborhood feature vectors are used in the computation of the more global ones.
To obtain a classification on the original point cloud, the features have to be propagated back to the original point set. This is done on each level, where the three nearest points of the higher-level point cloud are selected and the features are interpolated, weighted by the inverse distance from the point. These propagated values are fed through another multi-layer perceptron. The result is a set of neighborhood feature vectors for each point of the original point cloud.
These vectors are concatenated and fed through a fully connected layer. To avoid overfitting the training dataset, a dropout layer is applied during training. Here, every neuron has a fixed probability of returning nothing (i.e. zero) to the next layer. Finally, a softmax regression is carried out to obtain probability values for each class per point. The class with the highest probability is then assigned to the point along with the per-class probabilities.

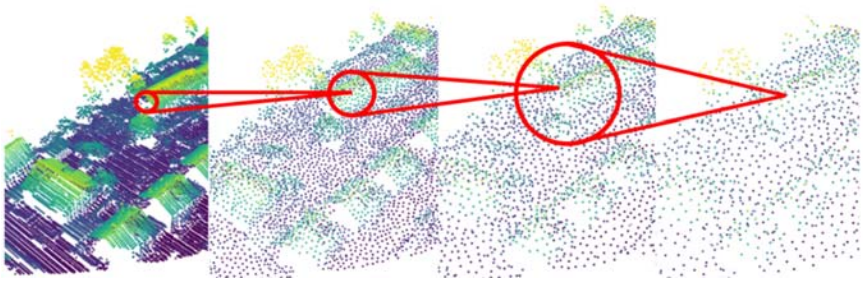

Fig. 2: Subsampling of the point cloud using farthest point sampling and calculation of feature vectors based on neighborhoods of increasing size.

3 Methodology

We adapted the algorithm of PointNet++ to work with data from Airborne Laser Scanning (ALS) in a workflow titled alsNet. The code for this workflow is available at https://github.com/lwiniwar/alsNet. While PointNet and PointNet++ mainly focused on the classification of CAD-objects represented by point clouds, the semantic segmentation was only a side-result of their study. They also applied their methods to scanned datasets of indoor environments. Their applications, however, remain in a local scope (QI et al. 2017b).
To apply PointNet++ to ALS data, we exploited a property of this data: ALS data follows the earth's surface, which is close to a 2-manifold; therefore, regular 2D-batches were cut out of the

dataset. To minimize possible edge effects, these batches were circular in shape, created by sampling the nearest 200,000 points around a grid point. We spaced the grid points such that every ALS point was covered by at least one batch, but four batches on average. The limitation of 200,000 points is introduced by the processing: Since the neighborhood query, the interpolation and the subsampling are performed on the graphics card, the GPU Memory (11 GB on an NVidia GeForce GTX 1080 Ti) was a limiting factor.
After the per-batch classification, we averaged the class probabilities for each point over the batches, and assigned the class with the highest average probability to the point.
To avoid long training times, we also tested using an already trained model as initial values for training on a new dataset (transfer learning).

4 Data

We tested the algorithm on different datasets, including a large-scale airborne laser scan and the 3D Semantic Labeling Challenge Benchmark of the ISPRS (GERKE 2014).

4.1 ALS Vorarlberg

The ALS dataset of the federal state of Vorarlberg covers about 2700 km² with a point density between 10 pts/m² and 20 pts/m². This results in about 10^{10} points, of which about $8 \cdot 10^8$ were selected for training and $6 \cdot 10^8$ for validation. The covered areas include urban, suburban, agricultural, forested, and alpine terrain. Along with the point coordinates, the amplitude of the returned signal, the number of echoes, and the echo number were provided for each point.
The reference classification in this dataset was derived using the automatic workflow in *TerraScan*, *TerraSolid*, and *TerraModeler* (*Terrasolid Ltd.*). In addition, classification was checked manually and corrected if needed, especially with respect to cable car lines. The classification follows the ASPRS LAS Specification (TOPOSYS 2014; ASPRS 2011).
Because of the unbalanced class frequency in the dataset, only batches containing a large variety of classes were used for training. This was achieved by calculating the standard deviation of the class histogram for each batch. A threshold on this standard deviation was set at 20 %, effectively selecting 3,781 of the original 13,255 batches for training.

4.2 ISPRS Benchmark Vaihingen

The Vaihingen dataset, provided as a benchmark for 3D semantic labelling (i.e. classification), consists of about 750,000 data points for training, spanning about 360 by 360 meters. NIEMEYER et al. (2014) manually created reference classes based on an existing 2D classification. As in the Vorarlberg dataset, amplitude information, the number of echoes and the echo number are provided with the dataset (GERKE 2014). We note here that the relatively small number of samples makes the application of machine learning algorithms, especially neural networks, difficult. Therefore, we did not apply any filtering on the training set batches.

5 Results

On the Vorarlberg dataset, we observed a strong spatial correlation of the classification accuracy. This lead to overall accuracies ranging between 95.8 % for urban areas (average over batches in a 2.5 x 2.5 km² tile), between 82.6 % and 86.7 % in villages surrounded by forested mountains and down to 63.6 % in the case of high alpine terrain. Additionally, we observed the spatial correlation on a smaller scale when looking at the accuracies per batch, as shown in Figure 3.
Looking at the results in more detail, Figure 4 shows a batch containing a power line, a ditch with vegetation and a bus terminal with buses and street furniture. While the buses and vegetation are classified well, especially the street furniture (yellow) does not appear in the estimation at all. Similarly, Figures 5, 6, 7 and 8 show the classification results on different areas and land cover types of the Vorarlberg dataset. On the ISPRS dataset, we achieved an overall accuracy of 80.2 %, a mid-level result compared to the competitors in the benchmark. A full confusion matrix of the classification result on this dataset is shown in Figure 9.
An interesting observation was made on the variance of the class probabilities per batch for the points: Points with a high variance (receiving different class labels from different batches) seem to correlate with erroneous classification. We conclude that we can use this variance as a measure of the quality of the classification, possibly along with probability surplus (i.e. the difference in probability of the most likely and the second most likely class).

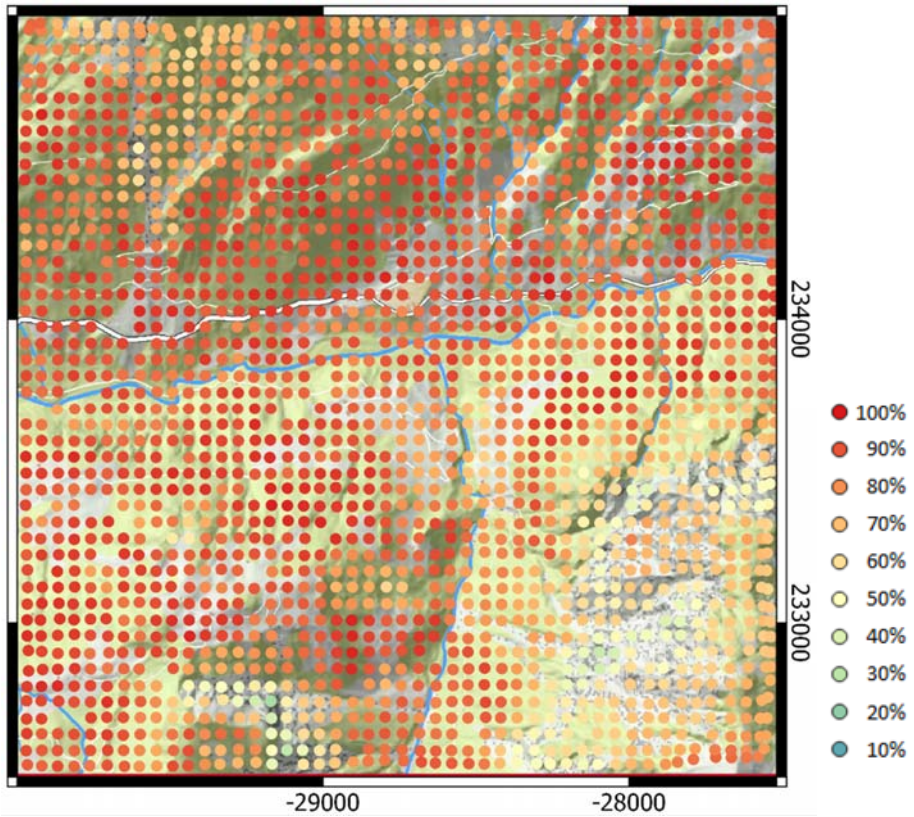

Fig. 3: Overall accuracies per batch (200,000 points each) in "Buchboden". A strong spatial correlation can be observed in the rocky terrain in the south and southeast. Background map: Land Vorarlberg – data.vorarlberg.gv.at. Coordinates: MGI/GK West (EPSG: 31254)

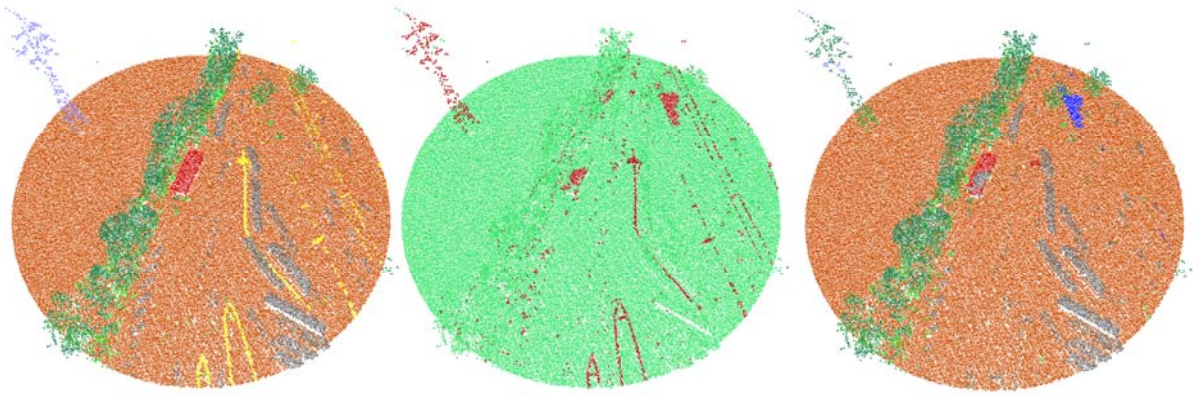

Fig. 4: A batch showing a bus terminal and a power line: reference (left), differences (center) and estimation (right). The buses (grey) are classified correctly, the station building (red) partially correct. There is a patch of water (blue) present in the estimation, and the power line pylon (light blue) is mostly classified as vegetation (shades of green). Overall, these pylons were misclassified, but the power lines (where present) were classified quite well.

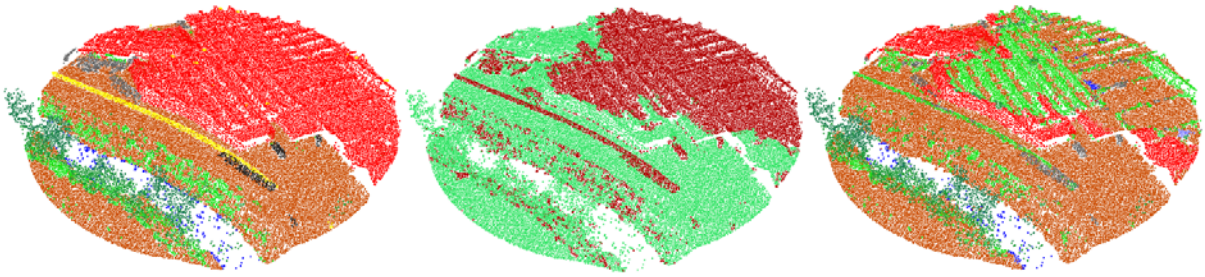

Fig. 5: Large factory building that is mostly misclassified as ground and low vegetation: reference (left), differences (center) and estimation (right). The maximum search radius of 15 m does not cover the building. In a strip of around 7.5 m along the border of the building, the classification is correct. Furthermore, street furniture (yellow, in this case a fence) is misclassified as vegetation, and some vegetation points along the riverbank are misclassified as ground.

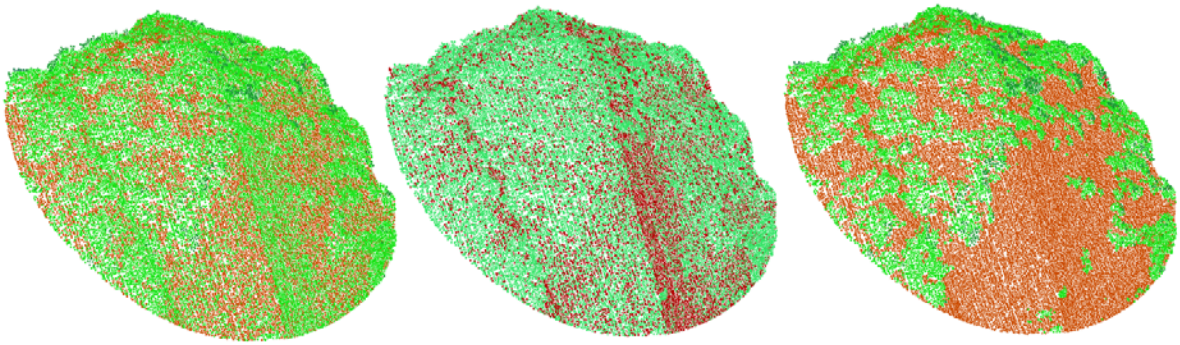

Fig. 6: All-vegetation batch showing the problematic separation between low vegetation and ground: reference (left), differences (center) and estimation (right). alsNet largely overestimates the ground points, whereas the reference shows much more vegetation.

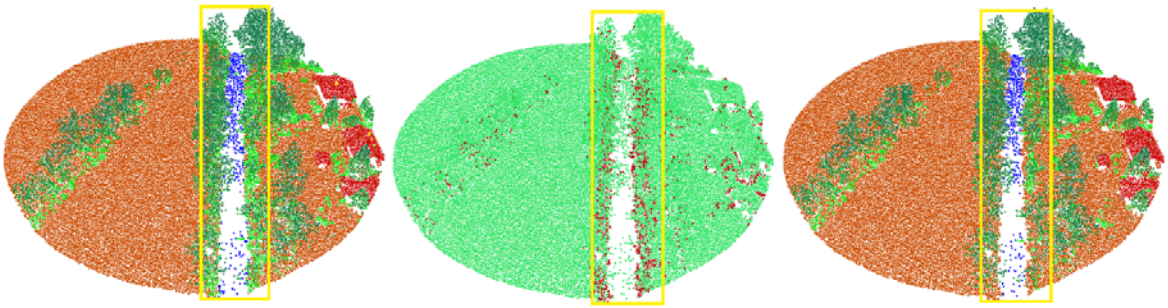

Fig. 7: Reference (left), differences (center) and estimation (right). On the river banks (yellow rectangle), the estimation shows much more ground points than the reference classification. The other areas in this batch (single buildings, trees) are classified well.

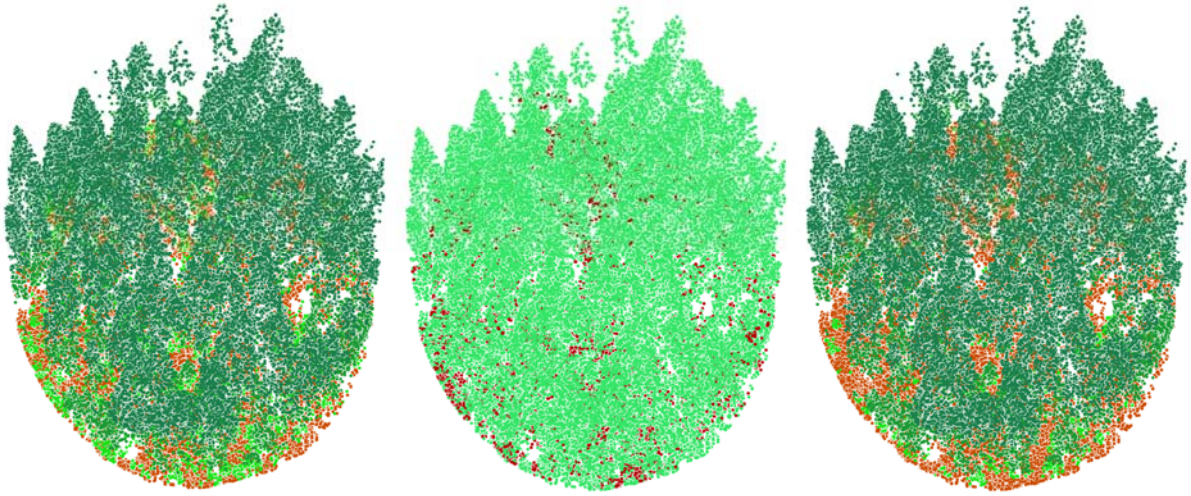

Fig. 8: A batch covering coniferous forest: reference (left), differences (center) and estimation (right). The few misclassified points again appear where low vegetation points are close to the ground. This is also caused by shadowing effects in the forest, where single points do not have close neighbors, leading to ambiguity.

6 Discussion and Outlook

alsNet has shown the ability of a neural network acting directly upon the point cloud to learn a representation of neighborhood information comparable to manually selected and extracted features. The benefit of end-to-end Deep Learning is that the features will automatically be optimized for the task at hand, given by the training samples. In addition, alsNet allows the computation of distribution parameters not solely on the geometry, but also on the other attributes of the point cloud.

The highest misclassification rates appear at the intersection of low vegetation and ground points, especially in areas where occlusion leads to isolated points. We expect that the use of more attributes that might be available from the sensor, namely the *echo width*, would aid in the separation of these two classes. Furthermore, attributes acquired by different sensors such as RGB values from a camera may be fused with the point cloud to provide additional information.

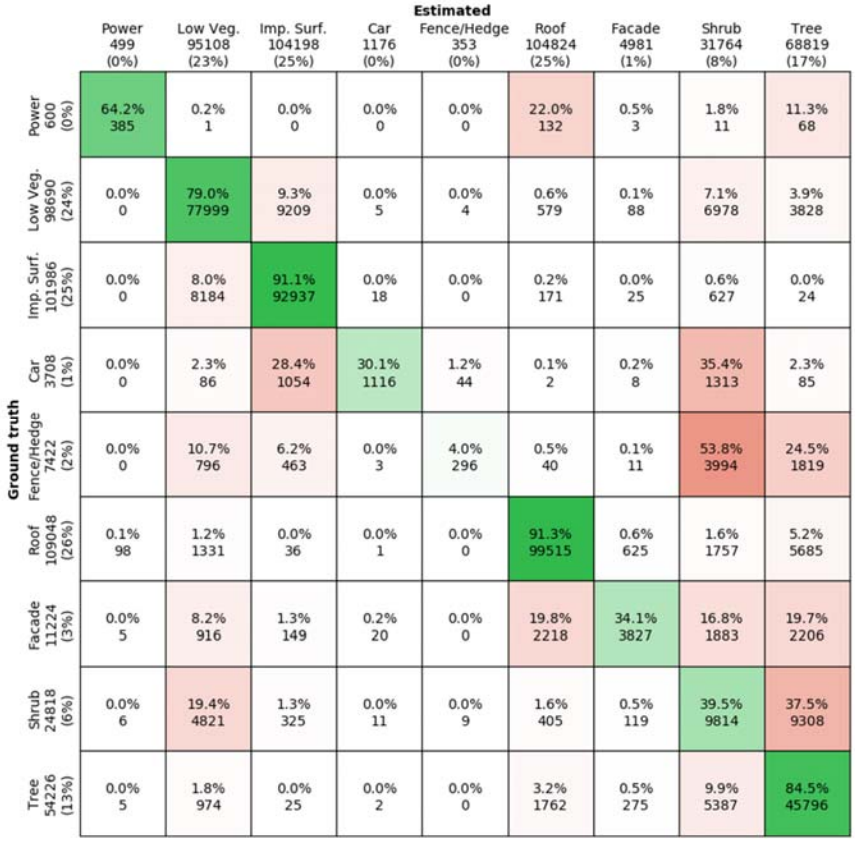

Fig. 9: Confusion matrix for the ISPRS Benchmark dataset. Correctly represented classes are shown on the main diagonal in green, whereas off-diagonal elements show misclassification. For example, 22 % of the power line points are misclassified as roof points (first row, sixth column). The main classes of low vegetation (24 % of the dataset), impervious surface (25 %), roof (26 %) and tree (13 %) are each classified with an accuracy of over 79 %.The highest misclassification in terms of absolute points is the estimation of shrubs as trees, followed by low vegetation classified as impervious surface.

The transfer of a trained model showed that the network is very susceptible to changes in point density and point patterns. When re-training the network, however, only a relatively small dataset is needed to achieve adequate results. The need for big training datasets could therefore be overcome by the use of simulated data (for which the reference is known exactly) and only very little real training data.

alsNet can be seen as a proof-of-concept for applying Deep Learning directly on topographic 3D point clouds by using an adapted version of PointNet++, considering characteristics of ALS data. The additional information obtained (probability, variance of probability over batches) can be further investigated. The extension of the neighborhood concept to 4D (i.e. time series of 3D point clouds) could be applied to analyses of terrain deformations and quantification thereof.

7 Acknowledgments

The neural network was trained and evaluated in part on hardware donated by NVIDIA at the Universities of Stuttgart and Heidelberg and on the Vienna Scientific Cluster VSC-3.

8 References

ASPRS, 2011: LAS Specification. Version 1.4 - R13. URL: https://www.asprs.org/wp- content/uploads/2010/12/LAS_1_4_r13.pdf, last access 17.06.2018.

BREIMAN, L., 2001: Random forests. Machine learning, **45**(1), 5-32.

FRÉNAY, B. & VERLEYSEN, M., 2014: Classification in the presence of label noise: a survey. IEEE transactions on neural networks and learning systems, **25**(5), 845-869.

GERKE, M., 2014: 3D Semantic Labeling Contest. URL: http://www2.isprs.org/commissions/comm3/wg4/3d-semantic-labeling.html, last access 17.06.2018.

GOODFELLOW, I., BENGIO, Y. & COURVILLE, A., 2016: Deep Learning. URL: http://www.deeplearningbook.org. MIT Press. ISBN: 978-0-262-03561-3.

GRILLI, E., MENNA, F. & REMONDINO, F., 2017: A review of point clouds segmentation and classification algorithms. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, **42**(2/W3), 339-344.

HEARST, M.A., DUMAIS, S.T., OSUNA, E., PLATT, J. & SCHOLKOPF, B., 1998: Support vector machines. IEEE Intelligent Systems and their Applications, **13**(4), 18-28.

NIEMEYER, J., ROTTENSTEINER, F. & SÖRGEL, U., 2014: Contextual classification of Lidar data and building object detection in urban areas. ISPRS Journal of Photogrammetry and Remote Sensing, **87**, 152-165.

OTEPKA, J., GHUFFAR, S., WALDHAUSER, C., HOCHREITER, R. & PFEIFER, N., 2013: Georeferenced point clouds: A survey of features and point cloud management. ISPRS International Journal of Geo-Information, **2**(4), 1038-1065.

QI, C.R., SU, H., MO, K. & GUIBAS, L.J., 2017a: PointNet: Deep learning on Point Sets for 3D Classification and Segmentation. IEEE Computer Vision and Pattern Recognition, **1**(2), 4.

QI, C.R., YI, L., SU, H. & GUIBAS, L.J., 2017b: PointNet++: Deep hierarchical feature learning on point sets in a metric space. Advances in Neural Information Processing Systems, 5105-5114.

ROSENBLATT, F., 1958: The Perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review, **65**(6), 386.

TOPOSYS (WIEDENHÖFT, A. & VATSLID, S.G., 2014: Technischer Abschlussbericht LiDAR und RGB - Land Vorarlberg.

WEINMANN, M., JUTZI, B. & MALLET, C., 2013: Feature relevance assessment for the semantic interpretation of 3D point cloud data. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, **5**(W2).

WEINMANN, M., JUTZI, B. & MALLET, C., 2014: Semantic 3D scene interpretation: a framework combining optimal neighborhood size selection with relevant features. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, **2**(3), 181.

YOUSEFHUSSIEN, M., KELBE, D.J., IENTILUCCI, E.J. & SALVAGGIO, C., 2018: A multi-scale fully convolutional network for semantic labeling of 3D point clouds. ISPRS Journal of Photogrammetry and Remote Sensing, **143**, 191-204.