Classification of Laser Scanning Data Using Deep Learning

FLORIAN POLITZ[1], BASHIR KAZIMI[1] & MONIKA SESTER[1]

Abstract: In the last couple of years Deep Learning has gained popularity and shown potential in the field of classification. In contrast to 2D image data, Airborne Laser Scanning data is complex due to its irregular 3D structure, which turns the classification into a difficult task.
Classifying point clouds can be separated into pointwise semantic classification and object-based classification. In this paper, we investigate both classification strategies using Convolutional Neural Networks (CNNs). In the first part of this paper, we focus on the semantic classification of 3D point clouds into three classes as required for generating digital terrain models. The second part of this paper deals with classifying archaeological structures in digital terrain models.

1 Introduction

Airborne laser scanning (ALS) is a very versatile technology with many applications such as terrain modelling, hydrology or archaeology. In ALS a laser pulse measures the time a laser beam requires to travel from a remote sensing module to the earth's surface and back. 3D point coordinates can be calculated using the position and direction of the remote sensing module in a global coordinate system as well as the direction and distance of the measured laser beam. An ALS point cloud is a collection of those point measurements for a specific area and is utilised for several applications such as ground point detection, topographic mapping or object recognition.
From the raw point cloud to a final product, the point cloud undergoes several processing steps with different intermediate products. In order to generate a digital terrain model (DTM), a semantic and pointwise classification on the raw point cloud is necessary to distinguish between objects on the ground and the ground itself. Subsequently, through those classified points, a DTM can be interpolated, which is usually saved as a rasterised height map or a triangulated irregular network (TIN). Finally, height maps or TINs can be the basis for further data analysis or classification such as object recognition or object reconstruction.
In the past, traditional machine learning algorithms have been used to classify the data to the requested products. As special type of a machine learning algorithm Convolutional Neural Networks (CNNs) moved into the focus of attention due to winning at the latest image benchmark competitions such as ImageNet and PASCAL VOC for 2D image processing by a large margin (KRIZHEVSKY et al. 2012; GIRSHICK et al. 2013). In contrast to regular 2D images, 3D point clouds possess an irregular data structure, which impede using CNN for semantic or object-based point cloud classification directly.
This paper deals with pointwise and object-based laser scanning point classification using a CNN. Firstly, we briefly overview the related work in order to classify laser scanning data in chapter 2. Secondly, we introduce the CNN and its network layers in chapter 3. Thirdly, we will present and discuss our work classifying ALS data using CNNs in the experiment's section in

[1] Leibniz Universität Hannover, Institut für Kartographie und Geoinformatik, Appelstraße 9a, D-30167 Hannover, E-Mail: [Florian.Politz, Bashir.Kazimi, Monika.Sester]@ikg.uni-hannover.de

chapter 4. The experiment's section is divided in two parts. The first part deals with semantic classification of raw point cloud data. The point cloud shall be classified into the classes "ground", "non-ground" and "others" such as cars, trains and open wires in order to generate a DTM. The second part deals with object-based classification. In this approach, archaeological terrain structures and objects have to be classified from a rasterised height map. Lastly, we finish the paper with a conclusion and outlook in chapter 5 as well as our references in chapter 6.

2 Related Work

For semantic classification using CNNs, laser scanning point clouds can be classified using voxel-based, point-based or projection-based approaches. In voxel-based methods, the point cloud is discretised to equally distributed 3D voxels, which are then fed into a CNN. PROKHOROV (2010) used a 3D CNN to refine coarsely segmented point clouds of cars using a voxel-based density grid. Meanwhile MANTURANA & SCHERER (2015a, 2015b) and HUANG & YOU (2016) calculated a voxel-based occupancy grid from the point cloud as input data for a 3D CNN. While MANTURANA & SCHERER (2015a) wanted to classify if a landing zone for helicopters is safe or not, HUANG & YOU (2016) classified each point in an urban city point cloud into one of seven classes such as buildings, trees, wires and poles. In the point-based methods, the raw point coordinates and some additional information are fed into a CNN. QI et al. (2017) showed the potential of this method by feeding their network called PointNet with raw point coordinates, corresponding colour information and a normalised position.

While voxel-based and point-based methods benefit from highly dense 3D point clouds, ALS point clouds are usually in 2.5D, where the point height can be considered as attribute. This is why, in projection-based methods, an ALS point cloud is projected into 2D raster images. HU & YUAN (2016) projected airborne laser scanning point clouds into 128×128 pixels images with the normalised minimal, average and maximal height for each pixel as channel values. While HU & YUAN (2016) only distinguished between ground and non-ground classes, our network is able to classify a third class for other, artificial structures on the ground such as cars, which must be eliminated from the point cloud for the DTM generation.

In object-based classification methods, the input data can be processed with two different ways. In the first method, points are segmented into objects and then are fed into a classifier. Using this method, MALLINIS et al. (2008) aimed to delineate forest vegetation polygons in a natural forest in Northern Greece. They segmented homogeneous pixels into objects and applied a Nearest Neighbour algorithm in order to classify these merged objects into different classes. DORREN et al. (2003) also used the first method for forest mapping in steep mountainous terrains. Furthermore, they fed those objects into a decision tree algorithm to classify each object into their specific forest class.

In the second method, image patches are generated from training data by extracting digital terrain models as rasterised height maps. The most common object class within a patch decides its label. Using this method, PALAFOX et al. (2017) clipped squared grids of different sizes from the centre of a visible object to generate training data and applied those grids as an input for a classifier to detect landforms in Mars. The different sized patches are utilised to detect different sizes of the underlying phenomenon and later fed into different CNNs for training. The results of all CNNs are then aggregated to lead to a final classification. We utilise the second method in our work and feed rasterised height maps in a CNN to classify terrain structures originating from

archaeological sites. The input for the CNN are squared grids with a single channel of object heights. The model outputs a single class for the whole input grid.

3 Convolutional Neural Networks

CNN are a subgroup of general Neural Networks, which gained a lot of attention in the past couple of years due to their remarkable results in several benchmark sets. Different linear and non-linear functions build a CNN:
The linear functions can be separated into two different layer types: convolutional and fully-connected layer. A convolutional layer connects units locally using kernel matrices and it calculates a linear convolution. A convolutional layer is position-invariant, since it shares the kernel weights over the whole image. While training, the network optimizes the kernel weights as well as additional bias terms. Another linear function within a CNN is the fully connected layer (FC-layer). In FC-layers each unit of one layer is connected with all units from the previous layer leading to a very dense feature aggregation. Convolutional layers are utilised for feature extraction from the input, while FC-layers gather the most important features for classification and detection purposes at the end of CNNs.
Non-linear functions enhance the capacity of the network and allow non-linear feature aggregation. The Rectified Linear Unit (ReLU) by NAIR & HINTON (2010) is a state-of-the-art non-linear activation function for CNN. ReLU is defined as

$$f(x) = \begin{cases} max(0,x) & if\ x > 0 \\ 0 & else \end{cases}, \tag{1}$$

where x is the output of a convolutional or fully-connected layer. After ReLU only the non-negative gradients with the most valuable features remain. Compared to ReLU, LeakyReLU by MAAS et al. (2013) does not set the activation for $x \leq 0$ to zero, but scales the activation by multiplying with 0,01. As a result, the overall gradient-based optimisation algorithms improves the training's speed. Another non-linear function is max-pooling. In max-pooling two layers are connected, so that only the maximal value within a specified patch of the first layer remains in the next layer. As a result, only the most informative features persist, while spatial invariance is achieved (SCHERER et al. 2010).
Lastly, dropout regularises the network to prevent overfitting. In dropout, network units are randomly dropped while training (SRIVASTAVA et al. 2014). Consequently, in each epoch only a fraction of the network is trained, since weight and bias parameters are only updated on activated units. Therefore, the network is forced to learn more general features for each output class.
For classification purposes, the gathered features have to be transformed into discrete class numbers. In case of binary and multiclass classification, a sigmoid and a softmax function are applied respectively (BISHOP 2006).
The CNN consist of several layers: in the beginning of a CNN, convolutional layers build block structures with ReLU and max-pooling layers. The convolutional part usually finishes with a flattening of the last raster array of size n x n into a $1 \times n*n$ vector, which is necessary to connect convolutional layers with fully connected layers. For classification and detection purposes, fully connected layer with ReLU, dropout and some kind of classification function for their specific problem such as softmax are added at the end of the CNN. Figure 1 shows an exemplary CNN and its structure.

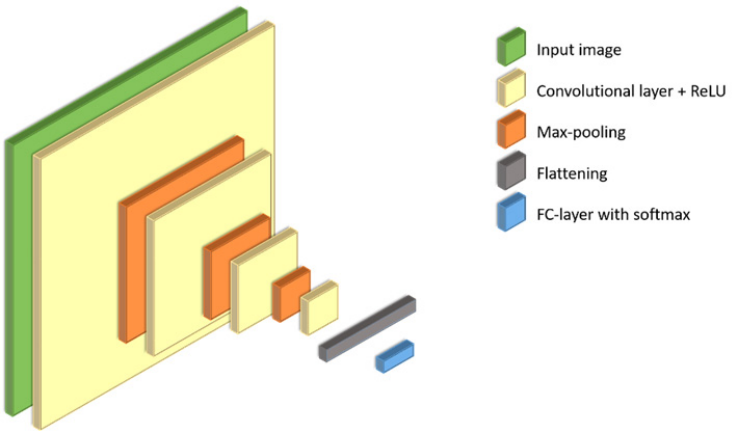

Fig. 1: Exemplary CNN structure with convolutional and ReLU layers, max-pooling, flattening and a FC-layer.

4 Experiments

4.1 Semantic Classification of airborne laser scanning point clouds

4.1.1 Input Data

For the experiments, ALS data from the survey state agency of Mecklenburg-Vorpommern (LAiV-MV) in Germany are given. The ALS data was generated in September 2012 and is located in the southeast of Rostock, Germany. The data includes a 20 km² wide area with a point density of around 4 points/m². The classes "ground", "non-ground" and "cars" are manually classified and are applied as target data for a supervised classification approach. The resulting ground points are the basis to interpolate a Digital Terrain Model (DTM). As dynamic or artificial objects such as cars are not part of a DTM, they have to be eliminated from the point cloud.

A Convolutional Neural Network (CNN) expects input data in a regular structure such as an image. For the CNN, we generate the input data similar to HU & YUAN (2016). For that reason, we project a 3D point and its environment into a 2D image structure. This is done by creating image patches around individual points, which are representing the point's environment. The patch size of 128m was chosen. Choosing such a large environment gives the network the opportunity to distinguish between large areas of flat ground and huge buildings with flat rooftops. Then, all points in that patch are sorted into 1m² raster, leading to a 128×128 pixel image. Additionally, for each raster cell within this image the minimal, average and maximal height are calculated. Finally, the height of the random point i declared as z_i is subtracted from each height value (z_{min}, z_{avg}, z_{max}) and normalized by a sigmoid function

$$Sigmoid(x) = \frac{1}{1+e^{-x}}, \tag{2}$$

where x equals $\{z_{min}, z_{avg}, z_{max}\} - z_i$. The resulting 2D structure is a 128 pixel × 128 pixel image with three channels (normalised minimal, average and maximal height values) for each point in the point cloud. The process chain from the point cloud to the input data is shown in figure 2.

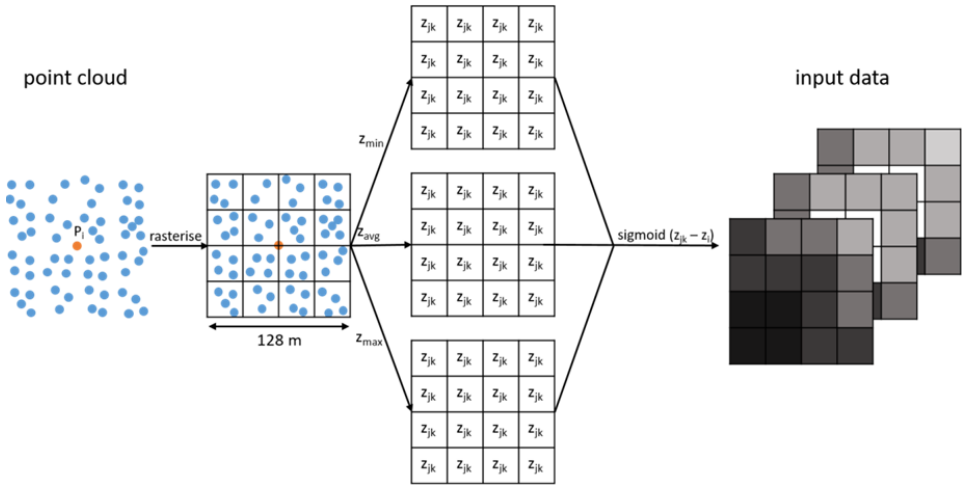

Fig. 2: Process chain to generate rasterised input data from an irregular point p_i and its environment within a point cloud.

4.1.2 Network structure

We adopt a similar CNN to the VGG16 network of SIMONYAN & ZISSERMAN (2015). In 2014, the VGG16 network achieved first and second place in the ImageNet localisation and detection challenge respectively. Since the detection challenge distinguishes 1000 classes, the capacity of the network is unnecessarily huge for our task to classify three classes. In order to prevent overfitting, we only rebuilt the first four blocks of VGG16 including several convolutional layers with ReLU and Max-Pooling at the end of each block. The kernel size for the convolutional layers is 3x3 and 2x2 for max-pooling layers following VGG16. After the last max-pooling, the aggregated features maps are flattened and two fully-connected layers with dropout are added to the network. A LeakyReLU is applied in between the FC-layers in order to fasten the training process while maintaining all gradients. At the end, a softmax function distinguishes the resulting features into the classes “ground”, “non-ground” and “cars”. Additionally, the input size is set to 128×128 pixel in comparison to the original VGG16 size of 224×224. The complete network structure is shown in table 1. Convolutional layers are abbreviated as “conv”.

4.1.3 Experiments

For the experiments, we split the point cloud data into three separate regions, leading to training, validation and testing data. The training data and their corresponding labels optimise the parameters from the CNN during training. While training, the validation data verifies the network’s capacity to generalise the data in order to prevent overfitting. Finally, the test data tests the optimised network independently from the training process. For the training region 126,000 points (42,000 points of each class) and for the validation and testing regions 18.000 points (6000 points of each class) are randomly picked from their respective regions. The ratio of training: validation: testing is 7:1:1.

Tab. 1: Overview of the network structure

Layer Name	Size (width x length x channel)	Number of Parameters
Input	128 x 128 x 3	0
Conv Block 1		
- Conv and ReLU	128 x 128 x 64	1,792
- Conv and ReLU	128 x 128 x 64	36,928
- Max-pooling	64 x 64 x 64	0
Conv Block 2		
- Conv and ReLU	64 x 64 x 128	73,856
- Conv and ReLU	64 x 64 x 128	147,584
- Max-pooling	32 x 32 x 128	0
Conv Block 3		
- Conv and ReLU	32 x 32 x 256	295,168
- Conv and ReLU	32 x 32 x 256	590,080
- Conv and ReLU	32 x 32 x 256	590,080
- Max-pooling	16 x 16 x 256	0
Conv Block 4		
- Conv and ReLU	16 x 16 x512	1,180,160
- Conv and ReLU	16 x 16 x512	2,359,808
- Conv and ReLU	16 x 16 x512	2,359,808
- Max-pooling	8 x 8 x512	0
- Flattening	1 x 1 x 32768	0
Dropout 1	1 x 1 x 32768	0
FC-layer 1	1 x 1 x 256	8,388,864
LeakyRelu	1 x 1 x 256	0
Dropout 2	1 x 1 x 256	0
FC-layer 2 (with softmax)	1 x 1 x 3	771
	Total	16,024,899

Additionally, a few hyperparameters have to be set for training the network. The specific parameters divide into network related parameters and optimisation related parameters. The network parameters include the amount of units in the first fully connected layer (FC-Layer 1, check table 1) and the drop rate parameter in the dropout layer. The amount of units in FC-Layer 1 decides about the operating network capacity. In case of fully connected layer, the parameter amount calculates as input_channel*output_channel + output_channel for weights and bias respectively. In case of 256 output_channels for FC-layer 1 as shown in table 1, the amount of necessary parameters for this layer reaches over 50% of the total training parameters. The more output units in FC-layer 1, the more the network capacity is increasing, but also the more training parameters are necessary to compute. The drop rate describes how many units are randomly dropped out during training, f.e. using a dropout rate of 0.75 leads to 75% dropped out units in this layer. The higher the drop rate, the sparser is the network while training, which forces the network to learn more general features rather than the training set itself. While testing, dropout is not applied to the network. The optimisation hyperparameter relate to all hyperparameters, which describe the optimisation process during training. The training was executed with a batch size of 50 using the cross entropy as loss function and categorical accuracy as metric. The Adam optimiser is utilized as optimising function (KINGMA & BA 2015). The optimiser consists of different parameters such as learning rate, β_1, β_2, ε and decay. For this experiment, we explore different learning rate values while keeping all other parameters for Adam as mentioned in the original paper by KINGMA & BA (2015). Additionally, we explore different parameter values and their influence towards the resulting classification accuracy.

During training, in each epoch 50 input images are used to update the parameter values. Each input image is fed into the network during training only once. Therefore, the training process lasts 2520 epochs to process all input images. The Keras API (KERAS-TEAM 2018) provides pre-trained VGG16 weights for the ImageNet data set. We initialised the weights of the convolutional layers with the provided weights and are kept untrainable for the first 20 epochs of training. Consequently, those weights do not lose their feature learning ability while the parameters in the fully connected layers are focused on classifying those extracted features from the convolutional layers to the target classes. After the first 20 epochs, all weights are included in the updating step to approach the given data.
After training, the classification ability of the network can be tested by classifying the test data set. We trained several networks with different parameter sets in order to validate the influence of each parameter and their respective classification accuracy score. Each parameter is tested separately, while the not observed parameter values within each test are set to the following values: number of units in FC-layer 1 256, dropout rate 0.75 and learning rate 0.0001.
The resulting accuracy scores for every tested parameter are shown in table 2. The first column shows the accuracy scores when the amount of units in FC-layer 1 is changed. 512 and 256 units achieve around 3% better results than networks with only 128 or even 1024 units for this layer. Although the resulting network of 512 units achieves a slightly better accuracy score, this layer also consists of 16,777,728 parameters, which have to be optimised, compared to the 8,388,864 parameters for the network with only 256 units in this layer as shown in table 2. For different drop rates in the second column in table 2 the results are not as clear as for the number of units in the fully connected layer. The accuracy score here only differs within 2% for drop rates between 0.70 and 0.85. Despite this trend, when the drop rate is set to 0.90, the accuracy score decreases for more than 5%. Such a drop indicates that the network is no longer be able to achieve correct results, because its structure is too sparse. For the learning rate, a value in the interval between 0.0005 and 0.00005 seems sufficient according to the third column in table 2 with difference of 2% in the accuracy score. Choosing a learning rate of 0.00001 results in an accuracy score of only 84,46%. In this case, the amount of training epochs is simply not enough for the low learning rate to reach the global minimum in only 2520 updates.

Tab. 2: Accuracy scores for different parameter values

Amount of units in FC-layer 1		Drop rate		Learning rate	
Parameter values	Accuracy score	Parameter values	Accuracy score	Parameter values	Accuracy score
1024	86.82	0.70	89.83	0.0005	89.03
512	90.26	0.75	90.00	0.0001	91.14
256	90.00	0.80	88.71	0.00005	89.31
128	85.23	0.85	90.42	0.00001	84.46
		0.90	84.23		

In table 3 the confusion matrix between the classes ground, non-ground and cars for the best network classification using 256 units in the FC-layer 1, a drop rate of 0.75 and a learning rate of 0.0001 are shown. This network achieves an overall accuracy score of 91,14%. The network classified around 94% of the classes ground and cars correctly, but only 84,13% non-ground points were classified correctly. 10,95% of all non-ground points were wrongly classified as ground points.

Tab. 3: Confusion matrix for 18,000 randomly picked test points [%]. Each class is equally distributed.

		Predicted label		
		Ground	Non-ground	Cars
True label	Ground	94.37	3.47	2.17
	Non-ground	10.95	84.13	4.92
	Cars	0.07	4.35	94.92

To identify the location of these misclassifications, we tested our CNN on a 250×250 m² area within the 20 km² point cloud, which have not been used for any previous training, validation or testing of the CNN. In comparison to the randomly picked testing data with an equal class distribution, we picked a test region with a highly unbalanced class distribution. The test region also consists of a challenging terrain with steep slopes as shown in figure 3. Despite the given challenges, the CNN achieved an overall accuracy of 88,41% for the test region.

Fig. 3: Orthophoto of the 250 x 250 m² test region in the southeast of Rostock, Germany in September 2014. The Orthophoto resembles the region, but the ALS data is recorded in September 2012.

Figure 4 shows the misclassifications between ground and non-ground classes. The misclassification of ground as non-ground lies mostly in the forest area in the northwest and in the mound in the southwest. The latter is caused by the lack of training data with mounds. The misclassifications within a forest are to be expected for two reasons. First, most of the ALS points within forest areas are at the first pulse on the treetops, so that points on the ground are sparse. Consequently, the ground cannot be represented correctly by z_{min} in the input images. Second, there is no guarantee, that points from last pulse are representing the ground, leaves or some undergrowth, so even for the manual classification, ground points within a forest are considered challenging. The same explanation applies for the misclassification of non-ground as ground in the forest area. Additionally, there are misclassifications on the field in the east as well as a steep slope in the southeast. The data set is recorded in September, where the fields are usually mowed down and ploughed resulting in a very rough ground structure. Consequently, the differentiation between ground and stubble field is not clear. Steep slopes are always a challenge in DTM generation. Additional training data with steep slopes will help the CNN to learn this special kind of ground better.

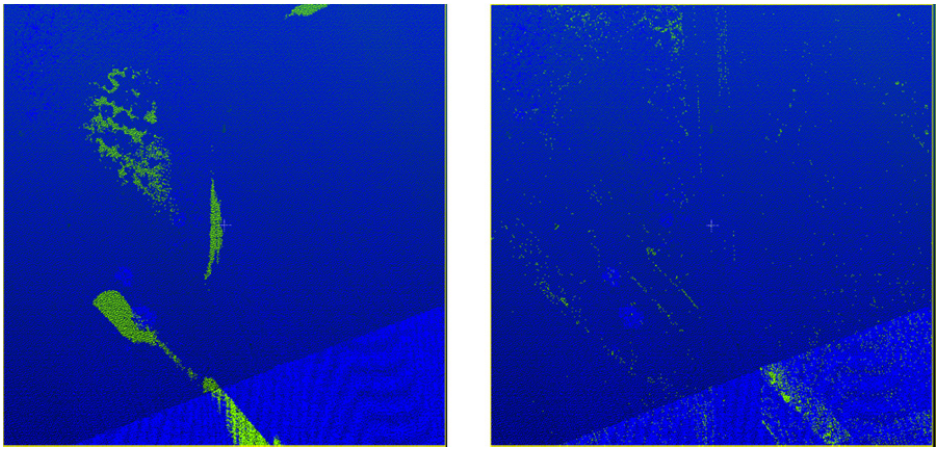

Fig. 4: Misclassifications from our CNN on the test region. Misclassifications are colour-coded as green, while the remaining classification results are blue. (left) The CNN classified ground as non-ground. (right) The CNN classified non-ground as ground.

4.2 Object-based Classification

4.2.1 Input Data

The goal of the second project is to determine archaeological terrain structures such as hollow ways, pits, heaps, shafts, sink holes, but also historical roads and water bodies in DTMs. The data set is a DTM acquired from Harz Mountains region in Lower Saxony, Germany. In our first experiments, we concentrated on two object classes, which are easily distinguishable in DTMs. To generate training data for our network, we labelled known water bodies and roads in the data set in an automatic fashion by intersecting the DTM with GIS data containing these features. Afterwards, we clipped 100×100 pixels images sampled in regular distances on the GIS data, where each pixel represents the elevation value. Figure 5 illustrates the DTM with the overlay of GIS data. The clipping process for generating training data and example inputs for the model are shown in figure 6. The images are normalised according to

$$i' = \frac{i-\mu}{\sigma}, \tag{3}$$

where $i'$is the normalized value for each pixel, i is the previous (original) pixel (elevation), μ is the mean elevation for the whole dataset and σ is the standard deviation for the whole dataset. Normalisation is necessary for the model to be able to learn from patches in one DTM and to identify features in another DTM. Without normalisation, the learnt features are specific to the training data and not transferable to any new data.

The network classifies water bodies and roads. In total, 36,248 images are used in training. 18,927 of those are road images, while the remaining 17,321 images show water bodies. Around 80% of the data is utilized for training and 20% for testing.

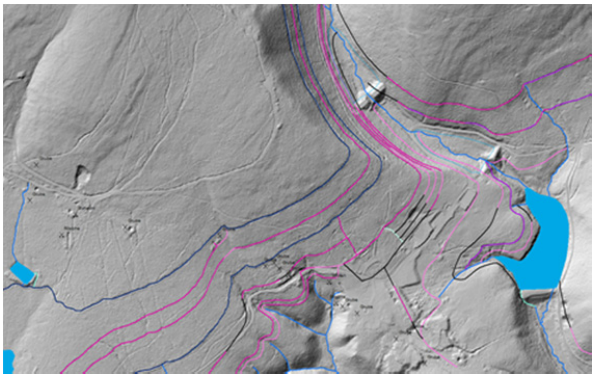

Fig. 5: Overlay of roads and water bodies on top of DTM.

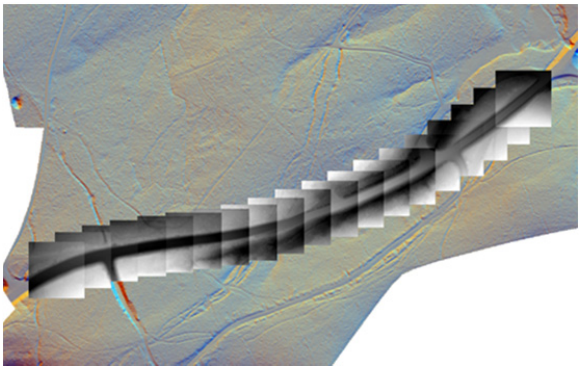

Fig. 6: Clipped roads to create labelled test data.

Table 4 summarizes the characteristics of the data set used in this experiment.

Tab. 4: Data set used for object-based classification

	Dataset 1: DTM data
# training images	28,997
# testing images	7,251
Input size	100 x 100 x 1
# classes	2 (water bodies & roads)

4.2.2 Network Structure

The CNN model for object-based classification takes a height map of size N x N as input and outputs a single label for the entire input image. The model consists of three blocks with a convolutional layer, ReLU and a max-pooling layer. Another convolutional layer and ReLU follows these blocks. The model finishes with a fully connected layer and a softmax function to classify the aggregated features into the respective classes. Table 5 summarizes the structure of the model for the data set. Convolutional layers are abbreviated as “conv”.

Tab. 5: Overview of the network structure for object-based classification

Layer Name	Size (width x length x channel)	Number of Parameters
Input	100 x100 x 1	0
Conv and ReLU	100 x 100 x 16	592
Max-pooling	50 x 50 x 16	0
Conv and ReLU	50 x 50 x 32	115,232
Max-pooling	25 x 25 x 32	0
Conv and ReLu	25 x 25 x 16	12,816
Max-pooling	12 x 12 x 16	0
Conv and ReLu	12 x 12 x 8	2,056
Flattening	1 x 1 x 1152	0
FC-layer (with softmax)	1 x 1 x 2	2,306
	Total	133,002

4.2.3 Experiments

For training the CNN, we also chose the loss function as the categorical cross entropy and Adam as optimiser (KINGMA & BA 2015). The batch size is 32 and the learning rate is 0.001. After completing the training process, the model was tested. The CNN achieves a test accuracy of 75%. To provide a better insight into the results, we list the confusion matrix for the test set in table 6.

Tab. 6: Confusion matrix for test data [%]

		Predicted label	
		Water bodies	Roads
True label	Water bodies	72.45	27.55
	Roads	22.68	77.32

As observed in table 6, 72.45% of water body images were classified correctly while the remaining 27.55% were misclassified as roads. In case of road images, 77.31% were classified correctly and 22.68% were misclassified as water bodies. There are two main reasons for the misclassifications. First, the training data set is small and deep neural network models usually require many training examples. The second reason for the misclassification is probably in the way the training images have been clipped. While clipping grids of 100×100 pixels from the DTM, some of the examples of water bodies might contain roads, and vice versa. Figure 7 shows some example inputs and their predictions by our model.

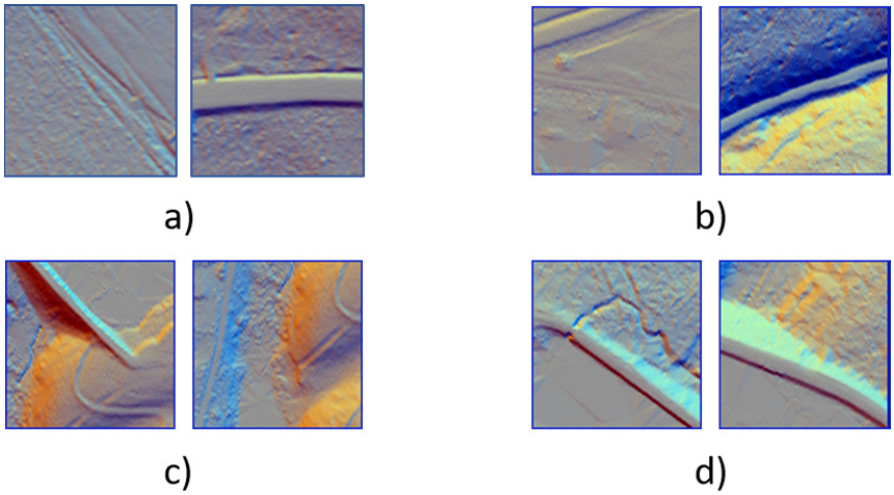

Fig. 7: Example inputs and predictions by our model. The input size is 100 x 100 pixel. a) Road input classified correctly as road. b) Road input classified incorrectly as water body. c) Water body input classified correctly as water body. d) Water body input classified incorrectly as road.

5 Conclusion and Outlook

In this research, we explored and discussed different methods to classify airborne laser scanning. In the first method, each point in the point cloud is classified. The second method deals with object classification using digital terrain models, which are a derived product from airborne laser scanning point clouds. A convolutional neural network (CNN) was applied as classifier in both methods. A CNN requires a regular structure for their input, which irregular point clouds from airborne laser scanning does not provide.

For this reason, the 3D point clouds were projected into 2D images in the first method. For each point, the normalized minimal, average and maximal height is taken into account as input images. The trained network classifies the point cloud into the classes ground, non-ground and car and achieves an overall accuracy of 91,14% while testing, which is sufficient for practical use to derive a digital terrain model. From the scientific point of view, this is not enough. As mentioned in experiments in chapter 4.1.3 there are still some errors left in the trained network resulting in around 10% of the non-ground points being misclassified as ground points. In future work, we will reduce those misclassifications. Equally, we will increase the number of classes in our network in order to distinguish buildings from vegetation in the non-ground class as well as streets from simple ground points in the ground class.

The object-based classification also achieves reasonable results. Only by considering the elevation values in a grid containing the whole object in our data set, the model learns to identify objects with an accuracy of around 75%. In summary, the object-based classifier implemented in this research proves to be appreciable. With more training data, the model is expected to learn better and gain a higher accuracy. To this end, additional training data will be included from other areas; furthermore, also perturbations of the existing training data (e.g. rotation or scaling) will be applied to create more data with a higher variation. Additionally, we will extend the model to detect more classes such as hollow ways, pits, heaps, shafts, sink holes, etc. After classification, the next task is to delineate the object geometry and create 3D object models.

6 References

BISHOP, C., 2006: Pattern Recognition and Machine Learning. Springer Science+Business Media, LLC. Singapore. ISBN 978-0-387-31073-2, 198.

DORREN, L.K.A., MAIER, B. & SEIJMONSBERGEN, A.C., 2003: Improved Landsat-based forest mapping in steep mountainous terrain using object-based classification. Forest Ecology and Management 183 (1-3), 31-46.

GIRSHICK, R., DONAHUE, J., DARRELL, T. & MALIK, J., 2013: Rich features hierarchies for accurate object detection and semantic segmentation. Extended version of the published conference paper at CVPR 2014. arXiv: 1311.2524v5.

HU, X. & YUAN, Y., 2016: Deep-Learning-Based Classification for DTM Extraction from ALS Point Cloud. Remote Sens. 2016 (8), 730.

KERAS-TEAM, 2018: Keras: Deep Learning for humans. GitHub repository, https://github.com/keras-team/keras

KINGMA, D.P & BA, J.L., 2015: Adam: A method for stochastic optimization. Published as a conference paper at ICLR 2015. arXiv:1412.6980v9.

KRIZHEVSKY, A., SUTSKEVER, I., & HINTON, G.E., 2012: ImageNet classification with deep convolutional neural networks. Proceedings of the 25th International Conference on Neural Information (NIPS) 2012 (1), 1097-1105.

MAAS, A., HANNUN, A.Y. & NG, A.Y, 2013: Rectifier Nonlinearities Improve Neural Network Acoustic Models. ICML'13 Proceedings of the 30th International Conference on Machine Learning. JMLR: W&CP (28).

MALLINIS, G., KOUTSIAS, N., TSAKIRI-STRATI, M. & KARTERIS, M., 2008: Object-based classification using Quickbird imagery for delineating forest vegetation polygons in a Mediterranean test site. Published in ISPRS Journal of Photogrammetry and Remote Sensing, **63**(2), 237-250.

MANTURANA, D. & SCHERER, S., 2015a: 3D Convolutional Neural Networks for Landing Zone Detection from LiDAR. Published in IEEE International Conference on Robotics and Automation (ICRA) 2015, 3471-3478.

MANTURANA, D. & SCHERER, S., 2015b: VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. Published in IEEE International Conference on Intelligent Robots and Systems (IROS), 922-928.

NAIR, V. & HINTON, G. E. 2010: Rectified Linear Units Improve Restricted Boltzmann Machines. ICML'10 Proceedings of the 27th International Conference on Machine Learning, 807-814.

PALAFOX, L. F., HAMILTON, C. W., SCHEIDT, S. P., & ALVAREZ, A. M., 2017: Automated detection of geological landforms on Mars using Convolutional Neural Networks. Computers and Geosiences, **101**, 48-56.

PROKHOROV, D., 2010: A convolutional Learning System for Object Classification in 3D Lidar Data. IEEE Transactions on Neural Networks, **21**(5), 858-863.

SCHERER, D., MÜLLER, A. & BEHNKE, S., 2010: Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. International conference on artificial neural networks. Springer, Berlin, Heidelberg, 92-101.

SIMONYAN, K. & ZISSERMAN, A., 2015: Very Deep Convolutional Networks For Large-Scale Image Recognition. Published as a conference paper at ICLR 2015. arXiv:1409.1556v6.

SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I., & SALAKHUTDINOV, R., 2014: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, **15**, 1929-1958.

QI, C.R., SU, H., MO, K. & GUIBAS, L.J., 2017: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. Proc. IEEE Computer Vision and Pattern Recognition (CVPR), **1**(2), 4, arXiv: 1612.00593v2.