

FULLY CONVOLUTIONAL NETWORKS FOR GROUND CLASSIFICATION FROM LIDAR POINT CLOUDS

A. Rizaldy^{1,*}, C. Persello¹, C.M. Gevaert¹, S.J. Oude Elberink¹

¹ITC, Faculty of Geo-Information Science and Earth Observation, University of Twente, Netherlands – aldinizaldy@gmail.com, (c.persello, c.m.gevaert, s.j.oudeelberink)@utwente.nl

Commission II, WG II/3

KEY WORDS: LIDAR, Filtering, Ground Classification, DTM Extraction, FCNs, CNNs, Deep Learning

ABSTRACT:

Deep Learning has been massively used for image classification in recent years. The use of deep learning for ground classification from LIDAR point clouds has also been recently studied. However, point clouds need to be converted into an image in order to use Convolutional Neural Networks (CNNs). In state-of-the-art techniques, this conversion is slow because each point is converted into a separate image. This approach leads to highly redundant computation during conversion and classification. The goal of this study is to design a more efficient data conversion and ground classification. This goal is achieved by first converting the whole point cloud into a single image. The classification is then performed by a Fully Convolutional Network (FCN), a modified version of CNN designed for pixel-wise image classification. The proposed method is significantly faster than state-of-the-art techniques. On the ISPRS Filter Test dataset, it is 78 times faster for conversion and 16 times faster for classification. Our experimental analysis on the same dataset shows that the proposed method results in 5.22% of total error, 4.10% of type I error, and 15.07% of type II error. Compared to the previous CNN-based technique and LAsTools software, the proposed method reduces the total error and type I error (while type II error is slightly higher). The method was also tested on a very high point density LIDAR point clouds resulting in 4.02% of total error, 2.15% of type I error and 6.14% of type II error.

1. INTRODUCTION

1.1 Background

Ground point classification (often called filtering) is the task of classifying LIDAR point clouds into two classes: ground and non-ground. Once point clouds have been classified, a Digital Terrain Model (DTM) can be extracted from ground-labeled points.

In the past, point cloud classification was done through unsupervised classification. Sithole and Vosselman (2004) examine several filter algorithms in 15 sites with different terrain characteristics. In general, all filters perform well in relatively flat terrain in a rural area. However, the accuracies dropped significantly when the study area contained steep slopes or complex urban areas. Chen et al. (2017) mention that many different filters had difficulty when applied to complicated and sharply changing landscape.

Deep learning with CNNs has gained popularity for image classification and pattern recognition tasks. In contrast to other machine learning techniques, features are learned from the data itself, not extracted manually. Feature learning is achieved by stacking many layers which contain learnable parameters. In the beginning, those parameters are defined randomly. Then all parameters are adjusted by minimizing the loss function so that the networks provide a correct label for every input image.

CNNs have been proven effective for image classification (Ciresan et al., 2011). The task of image classification is to predict a label for every input image, given many training data.

Following the success of deep learning in many classification tasks, Hu and Yuan (2016) proposed to use Deep CNNs to classify point clouds into the ground and non-ground. The

method starts with a point-to-image conversion. The conversion is based on the height differences in the neighborhood such that there is a clear difference between ground and non-ground points in the extracted images, enabling the network to discriminate between them. After being trained using 17 million points, the method resulted in the lower error rates compared to other filter algorithms in the ISPRS dataset (Hu and Yuan, 2016).

A major limitation of this method is the computational cost. It comes as a result of redundant computation during the conversion because the conversion needs to be done for every single point. This is problematic because if the conversion is slow, it is not practical to handle high-density point clouds characteristic to modern LIDAR datasets.

Another problem related to the conversion is in sloped terrain. Since the conversion relies on the height differences to the neighbors, ground and non-ground points show similar patterns in steep terrain. This condition is not in line with the idea of having high pixel values for a ground point image and low pixel values for a non-ground point image.

The method proposed in this work aims to tackle those limitations. Instead of converting single point into a single image as done by Hu and Yuan (2016), we propose to convert all points into a single image. In this method, a point is represented by a pixel, not an image. Hence it is more efficient and reduces computational time of the conversion and the classification. Furthermore, information regarding the intensity value and return number are also used to deal with steep slopes since both features are invariant to the terrain slope.

In order to classify the pixels, we propose to use Fully Convolutional Networks (FCNs) as FCNs architecture is able to label each pixel in an end-to-end manner. After each pixel has a

label, it is transferred back to the points, so all points in the point cloud are labeled.

1.2 Related Work

In general, there are four methods for filtering based on unsupervised classification. Slope-based filtering (Vosselman, 2000) assumes if there is a large height difference between two nearby points, it is because both points are consisted of ground and non-ground points where ground point is positioned lower than a non-ground point. Progressive densification was proposed by Axelsson (2000). It starts with a selection of the lowest point in certain areas as seed points, then creates Triangulated Irregular Network (TIN). This surface is then densified iteratively by adding the remaining points which are within acceptable angle and distance tolerances. The process gradually builds a more detailed terrain surface. Surface-based filtering (Kraus and Pfeifer, 1998) creates a best-fitted surface to all points and gives lower weight for points below the surface. Then the surface is updated using these weights. The process is repeated iteratively until the change is small. As a result, the final surface will represent the terrain. The last filtering technique is segmentation (Sithole and Vosselman, 2005). Unlike the previous methods, this method works on segments rather than points. First, point clouds are divided into segments based on the assumption that a segment should have a smooth surface, then larger and lower segments are selected as ground.

Many improvements had been proposed such for slope-based (Kilian et al., 1993), surface-based (Pfeifer et al., 2001), and progressive densification (Nie et al., 2017). Another recent filter algorithm called parameter-free also had been developed (Mongus and Zalik, 2012).

In supervised classification, geometric and contextual features for each point can be extracted to train the classifier. Many machine learning classifiers were used such as Random Forest (Chehata et al., 2009) and Support Vector Machine (Zhang et al., 2013). Conditional Random Field has also been used to improve the result from the classifier (Niemeyer et al., 2012). Weinmann et al. (2015) provide a thorough summary using different classifier, features, and neighborhood definitions. Unlike unsupervised filtering techniques, most of these supervised classifiers not only classify point clouds into the ground and non-ground classes but also classify many other classes.

Deep learning was also used to classify point clouds as conducted on 3D ShapeNets (Wu et al., 2015), VoxNet (Maturana and Scherer, 2015), and PointNet (Qi et al., 2017). However, all of them were designed for 3D object recognition which gives one label for all points as an object and mostly tested on indoor point clouds. For point classification on LIDAR point clouds, Hu and Yuan (2016) proposed to use Deep CNNs. Even though PointNet can also perform single point classification, we prefer to use FCNs approach since we want to focus on optimization of computational cost on CNNs by using FCNs.

2. METHODOLOGY

Our proposed method is different to the previous Deep CNNs for ground classification by Hu and Yuan (2016) in two aspects. Firstly, we propose to convert all points into one large image instead of converting every single point into a separate image as done by Hu and Yuan (2016). Secondly, we designed a Fully

Convolutional Network (FCN), to classify each pixel in the image into the ground or non-ground class.

2.1 Point Clouds to Image Conversion

In order to work with CNNs, the method by Hu and Yuan (2016) converts every single point into an image of 128 x 128 pixels. The pixel value was calculated as the height difference between a point and its neighbors ($Z_{neighbor} - Z_{point}$), while the neighbors were defined as those points within an area of 96 x 96 meters. The purpose is to represent each point as a feature image. It is assumed that ground points are usually lower than the neighbors while the non-ground points are usually higher. If the assumption is true, most of the pixels in the ground point image are high-value pixels, so that ground point images look brighter than non-ground point images.

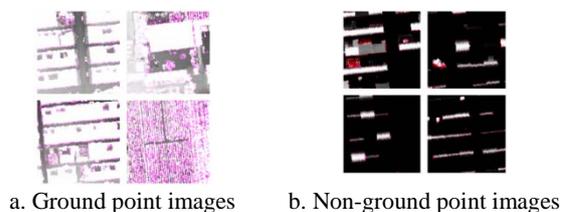


Figure 1. Feature images from (a) ground point and (b) non-ground point (Hu and Yuan, 2016).

If the terrain is relatively flat, the typical result of the conversion is shown in Figure 1. It is very easy to discriminate ground points from non-ground points in those images. However, the conversion is slow due to the highly redundant calculation. Furthermore, the conversion has difficulty in steep terrain when the converted images are no longer different between ground and non-ground points. It could happen because the assumption of ground points always lower than non-ground points is not true anymore. As a result, both ground and non-ground points have a similar pattern as can be seen in Figure 2.

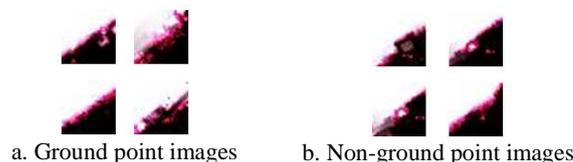


Figure 2. Feature images converted in a steep terrain

In order to overcome the computational cost problem, the proposed method does not convert every single point into a separate image but converts all points into a single large image. In this approach, points are represented by pixels, not by images anymore. As a result, the conversion and classification are faster than if each point is converted into a separate image.

The conversion is done by converting point cloud features into a multi-channel image. LIDAR provides not only a 3-D position but also information regarding the intensity and return number. All the information can be used as features when converting point cloud to image. In the conversion, pixel size is set to 1 x 1 m based on the point cloud density. If there is more than one point within a pixel, then the lowest point was chosen since it is assumed that the lowest point has more possibility to be a ground point than the upper points. On the other hand, if there is no point within one pixel, the pixel value was interpolated from the neighbors, and no label was given for that pixel in the

training image. Figure 3 shows the converted image from point clouds and the corresponding labels on one sample area.

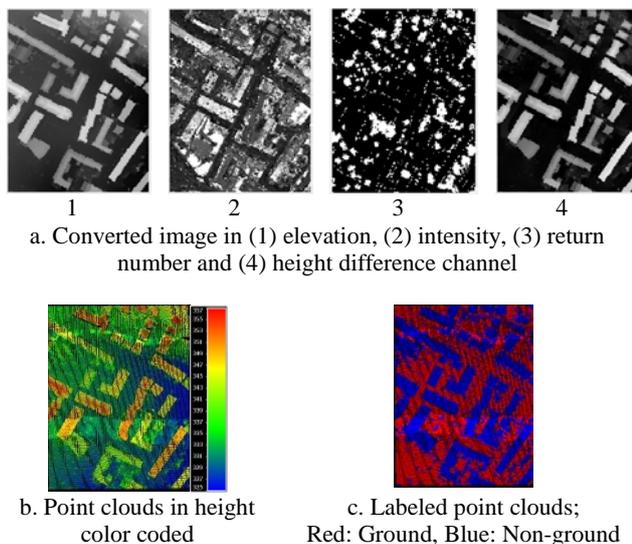


Figure 3. (a) Converted image (b) original point clouds in height color coded and (c) corresponding label for point clouds

Elevation, intensity and return number are original data of LIDAR point clouds. In addition, more features can be extracted. In the literature, height difference was reported as a useful feature to discriminate ground and non-ground points (Chehata et al., 2009). Therefore the proposed method has also used height difference as an additional feature. The feature is defined as the height difference between the point itself and the lowest point in the neighborhood while the neighborhood is set as a window of 20 x 20 m.

Another issue related to the conversion is low point outliers. It is common for LIDAR point clouds to have low point outliers. Those points usually have extremely low elevation so that many filtering algorithms suffer from this type of error. The effect of the outliers has also been examined in this study.

2.2 Fully Convolutional Networks (FCNs)

Common CNN architectures for image classification always predict one label per one input image. In this architecture, the input image is convolved and down-sampled in a series of processing layers which include convolutions, batch normalization, activation functions, pooling and fully connected layers. The last layer of the network predicts class probabilities. Figure 4 illustrates CNN architecture from LeNet-5 for handwritten digits classification.

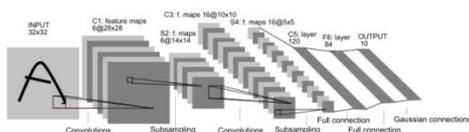


Figure 4. An illustration of CNN architecture LeNet-5 (Le Cun et al., 1998)

In the illustration above, an input image will have a probability for each ten output labels (0 to 9). Then the label with the highest probability is taken as a predicted class for the corresponding input image. Hu and Yuan (2016) used similar CNNs architecture for ground classification; except the final

layer has two output labels for ground and non-ground. Although they also used different network parameters, the general structure is similar.

In contrast to the common CNNs architecture, we propose to use a different architecture. Since point clouds are converted into one image, it means that points are represented by pixels. As a consequence, labels need to be computed for each pixel, not each image. This task is known as semantic labeling or pixel-wise classification. The CNN architecture as illustrated in Figure 4 needs to be modified to accommodate this purpose.

Fully Convolutional Networks (FCNs) are a modification of CNNs. The architecture is structured in such a way to predict a label for each pixel of an input image. Shelhamer et al. (2017) proposed to replace fully connected layer on a common CNNs architecture with an up-sampling layer. The method works by employing deconvolution layer to up-sample the output feature maps to the original resolution. This method was adapted for pixel-wise classification of high-resolution remote sensing imagery by Maggiori et al. (2017), Volpi and Tuia (2017) and Fu et al. (2017). Meanwhile, Badrinarayanan et al. (2015) proposed SegNet that uses encoder and decoder layer. Encoder plays a role as a down-sample layer while decoder as an up-sample layer. The decoder works by using the max-pooling indices from the corresponding encoder to produce the up-sampled feature maps.

In contrast to 'down-sample and up-sample' FCNs, Sherrah (2016) and Persello and Stein (2017) proposed another architecture that maintains the size of each layer in the networks. Hence the architecture is called no-downsampling. Gevaert et al. (2018) adapted the similar no-downsampling networks for DTM extraction from Unmanned Aerial Vehicle imagery.

The no-downsampling architecture maintains the output feature map to have the same size as the input image. It can be achieved by using stride = 1 for both convolutional and pooling layer. If F = filter size, then adding pad = $(F/2) + 1$ is also mandatory to ensure there is no down-sampling in the output feature map. Unlike CNNs architecture, the final layer of FCNs architecture provides the class probabilities of each pixel.

The FCN_DK network by Persello and Stein (2017) was chosen to be adopted in this work to avoid the up-sampling used in the deconvolutional FCNs. The proposed FCN architecture for ground classification is shown in Figure 5. The architecture can consume input images of different sizes. It uses dilated convolutions to increase receptive field size without drastically increasing the number of parameters (Yu and Koltun, 2015). A small modification to FCN_DK networks was taken in order to fit the networks for ground classification purpose. The max-pooling layers were removed, the number of filters and the filter size was changed.

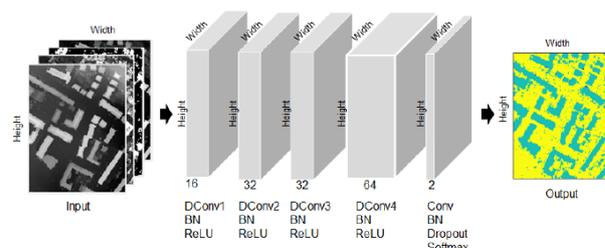


Figure 5. Proposed FCN architecture for ground classification

Layer	Filter Size (pixels)	# filters	Dilation	Receptive field size (pixels)
DConv1	5 x 5	16	1	5 x 5
Batch Norm				
ReLU				
DConv2	5 x 5	32	2	13 x 13
Batch Norm				
ReLU				
DConv3	7 x 7	32	4	37 x 37
Batch Norm				
ReLU				
DConv4	7 x 7	64	5	67 x 67
Batch Norm				
ReLU				
Conv	1 x 1	2	1	67 x 67
Batch Norm				
Dropout				
Softmax				

Table 1. Filter number and receptive field size

As can be seen in Figure 5 and Table 1, the size of the output feature map on each layer is always the same as the input image, while the receptive field is small in the earlier layers but gradually increases in the next layers. A larger dilation factor in the deeper layer causes a larger receptive field. In ground classification, it is important to have a large receptive field so that the network can capture both ground and non-ground features especially when large buildings are present in the area.

2.3 Training and Testing

Training and testing follow the common procedure of CNNs. In general, training was conducted by forward passing the input image through all layers so that each input unit (pixel) has a prediction label. Then it back propagates to learn the parameters of the network. The process is repeated until the prediction label is close to the true label. Testing was performed by forward passing input units through all layers which contain learned parameters from the training stage.

Forward passing on the network is done based on the same schema as used on many CNNs architectures. For input unit (x), weights (W), bias (b), non-linear activation function (g), then the output unit (h) is computed based on Equation (1).

$$h = g(W^T x + b) \quad (1)$$

In the CNNs architecture using Rectified Linear Unit (ReLU) as an activation function, the output unit (h) of input image (x) with depth (D) is computed as

$$h = \max\{0, \sum_{d=1}^D W^T_d * x_d + b\} \quad (2)$$

In a forward pass, every input unit (pixel) is passed through all layers. Next, the probability is computed in the final layer using a softmax classifier as defined in Equation (3). For $j = 1 \dots K$, softmax turns K -dimensional vector of h in the range $[0,1]$.

$$P(h)_j = \frac{e^{h_j}}{\sum_{k=1}^K e^{h_k}} \quad (3)$$

After that, a loss function is introduced. A common loss function used in the modern neural network is cross entropy (Goodfellow et al., 2016). It calculates the negative log-likelihood between the prediction from the network and the true

label from training data. The loss function is defined in Equation (4).

$$L(x, y; \theta) = -\sum_j y_j \log p(h_j|x) \quad (4)$$

Once the loss function has been calculated, learning is performed by minimizing the loss with respect to all parameters in the network as defined in Equation (5).

$$\theta^* = \operatorname{argmin}_{\theta} \sum_{n=1}^N L(x^n, y^n; \theta) \quad (5)$$

Then all parameters are adjusted using a stochastic gradient descent with momentum.

Image patches were created to train the network. Each patch has a size of 105 x 105 pixels. The training was done by feeding the image patches to the networks. The MatConvNet (<http://www.vlfeat.org/matconvnet/>) framework was used for the implementation. The networks were trained for 50 epochs with learning rate of 0.0001. The momentum is 0.9 and the weight decay is 0.0005. Each mini-batch has 32 samples. The dropout layer has rate of 0.5. After the networks have been trained, the testing samples were classified by passing testing images through the network to label each pixel as ground or non-ground.

2.4 Selecting ground points

The output of FCNs is a predicted label for each pixel while what is needed is a predicted label for each point in the point cloud. As mentioned earlier, a pixel is a representation of point so that the predicted label can be transferred to the corresponding points. In case a pixel has more than one point, the label is transferred to the lowest point within that pixel. It was done because the lowest point is chosen if there is more than one point in a pixel when converting point clouds to an image. Once the label was transferred to the lowest point, a surface is created connecting all points labeled as ground.

Then points are labeled as ground points if their elevation is within a threshold to the surface. The threshold was chosen based on the typical vertical accuracy of LIDAR point clouds, in this study the threshold is set to 15 cm.

3. DATASET

Two datasets were used for the experiment: ISPRS Filter Test and *Actueel Hoogtebestand Nederland* (AHN). The two datasets have different characteristics in terms of point density and multiple-return ability.

3.1 ISPRS Filter Test Dataset

The ISPRS dataset has 15 sample areas. From the 15 samples, ten sample areas were chosen for training and five sample areas for testing. Figure 6 shows the five testing samples.

The dataset has a low point density. This causes the terrain to be poorly represented, especially on a combination of steep terrain and low vegetation. There are only two returns, first and last, in contrast to five multiple returns on a typical modern LIDAR point cloud. Fewer returns mean less information is available to separate vegetation from the ground. This situation makes the dataset more challenging for ground classification.

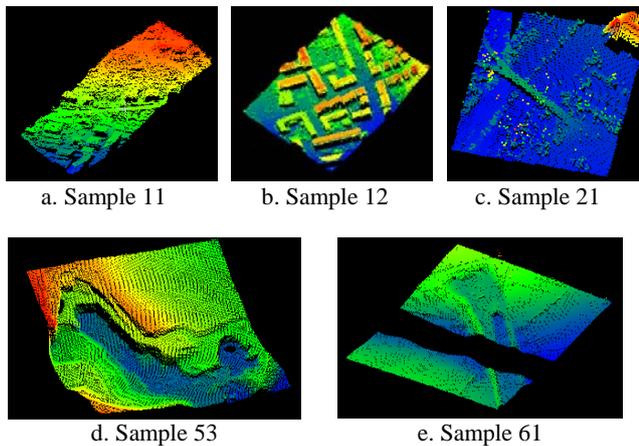


Figure 6. ISPRS testing samples: (a) steep terrain, (b) dense buildings, (c) bridge, (d) height discontinuity and (e) embankment and data gap

3.2 AHN Dataset

In contrast to ISPRS dataset, AHN dataset offers very high point density and high penetration from the multiple returns. It has more than 20 points per square meter and records up to 5 returns. The dataset already has reference labels for all points. The labels are ground, vegetation, building, water, and bridge. In this work, all labels except ground are re-labeled as non-ground. AHN dataset covers the entire area of The Netherlands, although the latest dataset, AHN-3 (can be accessed on <https://www.pdok.nl/nl/ahn3-downloads>), only covers half of the Netherlands. The dataset has relatively flat terrain which is the typical terrain in The Netherlands. Twelve sample sites were selected. Each sample has a size of 500 x 500 meters. Nine samples were chosen for training, two for validation and one for testing. Figure 7 shows the testing sample which includes small, medium and large buildings.

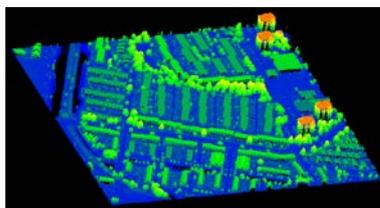


Figure 7. AHN-3 Testing sample

4. RESULTS AND DISCUSSIONS

This section explains the results from hyper-parameters tuning, accuracy assessment, and computational cost comparison. The optimum hyper-parameter configuration was used in the testing data and compared to the results from Deep CNN-based ground classification (Hu and Yuan, 2016) and unsupervised classification from LAStools software.

4.1 Hyper-parameters tuning

Different FCN architectures were carried out to find the optimum configuration. Due to the limited number of samples, all experiments were done with 2-fold cross-validation data. Ten training samples were divided into two folds.

Three experiments with a different number of convolutional layers were conducted. The first experiment had three convolutional layers, the second experiment had four convolutional layers, and the third experiment had five convolutional layers. The results in Table 2 below indicate that having more convolutional layers results in higher accuracy. In this architecture, having more convolutional layers increases not only the number of weights but also the receptive field size since it involves dilated convolution. Adding more weights solves more complex problems while the larger receptive field size allows the networks to learn on a bigger area. Deeper networks also usually achieve higher accuracy on many data sets (Simonyan and Zisserman, 2014; Sherrah, 2016; and Persello and Stein, 2017).

Validation set	Number of Convolutional layers		
	3	4	5
ISPRS	84.98	83.97	86.22
AHN	95.09	95.31	95.46

Table 2. Results from a different number of convolutional layers

Others experiments were performed to find out the effect of pooling layer. The first experiment had max-pooling layer while the second experiment had no max-pooling layer. The results in Table 3 show that removing pooling layer in the networks gives better accuracy for ISPRS validation set but slightly reduces the accuracy of AHN validation set. Another advantage of removing pooling layer is that it simplifies the network and faster computation.

Validation set	Pooling layer	
	Yes	No
ISPRS	85.44	88.54
AHN	95.35	95.09

Table 3. Results of employing pooling layer and not

Although the pooling layer is important in image-wise classification task because it makes networks invariant to small translation (Goodfellow et al., 2016), it seems that it is better to remove it in a pixel-wise classification. The nature of pooling layer is summarizing value in a small spatial unit thus reduce the output for each layer. In other words, pooling layers down-sample the output feature maps. Hence it becomes irrelevant in a no down-sampling architecture as used in this work.

Besides hyper-parameter tuning, experiments were also conducted to see the effects of adding height difference features and removing outliers. Experiments indicate that adding the height difference feature improves the accuracy by 1.31% while cleaning low point outliers before the point-to-image conversion improves the accuracy by 2.62%.

4.2 Accuracy assessment and comparison to other methods

The optimal architecture from the hyper-parameter tuning was selected to test the five testing samples. The architecture with five convolutional layers without max-pooling layer was chosen. The accuracy value was examined from the total error (percentage of misclassified points), type I error (rejection of ground points) and type II error (acceptance of non-ground points as ground points). Deep CNN-based ground classification (Hu and Yuan, 2016) and LAStools software (that uses progressive TIN densification) were chosen for comparison.

For faster point-to-image conversion, a modification is carried out on the Deep CNN approach. Every point is converted into 32 x 32 pixels, instead of 128 x 128 pixels as originally proposed by Hu and Yuan (2016). However, the window size is still the same 96 x 96 m. As a consequence of having a smaller image, the output feature maps are smaller for each layer. But the number of layers and filter size were still the same. Table 4 shows the difference between the Deep CNN architecture used here and the original version. The networks were trained using the same training set as used by the proposed method to have a fair comparison.

Layer	Original			Modified		
	W	H	Depth	W	H	Depth
Input	128	128	3	32	32	3
CONV, BN, RELU, Pool	64	64	64	16	16	64
CONV, BN, RELU, Pool	32	32	128	8	8	128
CONV, BN, RELU	16	16	256	4	4	256
CONV, BN, RELU	16	16	256	4	4	256
CONV, BN, RELU	16	16	256	4	4	256
CONV, BN, RELU, Pool	16	16	128	4	4	128
FC, BN, RELU	1	1	4096	1	1	1028
FC, BN, RELU	1	1	4096	1	1	1028
FC	1	1	2	1	1	2

Table 4. A small modification of Deep CNNs architecture in terms of width, height and depth of the layers.

The results from ISPRS dataset as seen in Table 5 show that the proposed method using FCN has a lower total error and type I error, but it has higher type II error. The results are consistent in the five testing samples; no matter whether it consists of flat or sloped terrain.

Sample	Total Error		
	FCN	CNN	LAStools
Sample 11	14.54	19.47	17.67
Sample 12	3.97	7.99	6.97
Sample 21	1.55	2.23	6.66
Sample 53	4.89	5.67	14.37
Sample 61	1.16	4.20	17.24
Average	5.22	7.91	12.58

Sample	Type I Error		
	FCN	CNN	LAStools
Sample 11	12.86	27.10	26.94
Sample 12	3.03	13.92	12.87
Sample 21	0.20	1.63	7.98
Sample 53	3.88	4.44	14.84
Sample 61	0.55	3.95	17.85
Average	4.10	10.21	16.10

Sample	Type II Error		
	FCN	CNN	LAStools
Sample 11	16.80	9.20	5.18
Sample 12	4.96	1.75	0.77
Sample 21	6.43	4.39	1.87
Sample 53	29.10	34.79	3.24
Sample 61	18.04	11.06	0.40
Average	15.07	12.24	2.29

Table 5. A comparison between the accuracies obtained by the proposed method, the previous CNNs approach, and LAStools.

Compared to previous deep learning algorithm using CNN proposed by Hu and Yuan (2016), FCN approach gives a lower total error.

It should be noted that Hu and Yuan (2016) reported better results on the ISPRS datasets (0.67% type I error, 2.26% type II error and 1.22% total error on all samples) based on a large

training dataset of 17 million points in mountainous terrain. What can be deduced is that the use of more representative training data improves the test results, so it is worth to train our FCN with more training data in future work. Qualitative evaluation was also conducted by visual inspection. Figure 8 shows labeled point clouds from the proposed method for five testing samples. Green points mean correctly labeled ground points. Blue points correspond to correctly labeled non-ground points. Yellow points (type I error) are misclassified as non-ground, where the true label is ground. Red points (type II error) are misclassified as ground but should be non-ground.

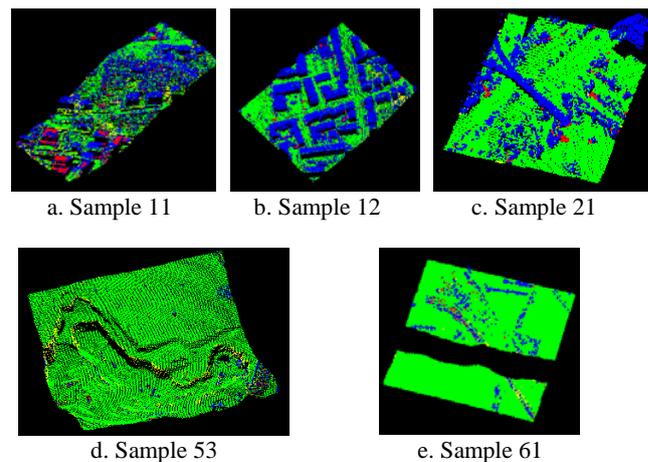


Figure 8. Labeled point clouds on five ISPRS testing samples (Green: correctly labeled of ground, Blue: correctly labeled of non-ground, Yellow: ground misclassified as non-ground, and Red: non-ground misclassified as ground).

Sample 11 is the most difficult area to handle compared to other samples. It has steep terrain combined with low vegetation and buildings. It can be seen that the proposed method using FCN can correctly remove buildings and vegetation in the upper and middle area without losing ground points. However, some buildings cannot be correctly removed in the bottom area.

Sample 12 is flat terrain with dense buildings. In general, FCN can perform well in this type of area. Sample 21 has a bridge and one large building. The proposed method also works well in this area especially as it removed the bridge and the large building.

Sample 53 has height discontinuity in the terrain while sample 61 has an embankment object. Ground points in both samples could be easily misclassified as non-ground points due to their shape. In sample 53, almost all break-line terrains were captured correctly by the proposed method. This kind of terrain is usually difficult to capture especially for filters that assume higher points in the local area are a non-ground object. The same result can be seen in sample 61. Most of the ground points were correctly classified. However, the FCN fails to classify a small number of non-ground points in both sample 53 and 61. Since both samples only have a few non-ground points (1,388 points for sample 4 and 1,247 for sample 5) compared to ground points (32,989 points for sample 53 and 34,563 points for sample 61), the type II error is relatively large.

Samples 11 and 53 have sloped terrain. Total errors in both samples are lower than CNNs-based approach. It proves the proposed method solves the problem of CNNs-based approach on sloped terrain as mentioned in the beginning.

Tested on AHN dataset, the proposed method has 4.02% of total error, 2.15% of type I error and 6.14% of type II error. Figure 9 shows the labeled points from AHN testing samples.

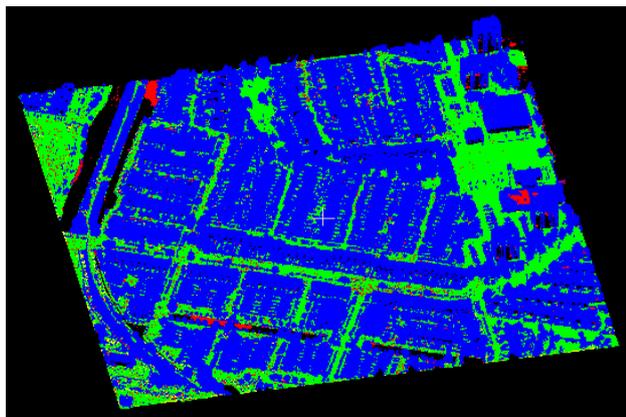


Figure 9. Labeled point clouds on AHN testing sample. (Green: correctly labeled ground, Blue: correctly labeled non-ground, Yellow: ground misclassified as non-ground, and Red: non-ground misclassified as ground)

From Figure 9, it can be seen that most ground points were classified correctly. Buildings and vegetation were also removed correctly. However, there are noticeable type II errors (red points) in the result. In most cases, those type II errors come from points on the water. It seems that no automatic classification can handle the situation perfectly. Manual editing is needed. If misclassification of water is not considered as an error since it does not affect the geometry for DTM extraction, then type II error reduces to 5.85%. Another noticeable type II error is on a building on the right side of the area. That is a large building with 65 x 35 m in size. It looks like the receptive field size of FCN still not large enough to handle this building.

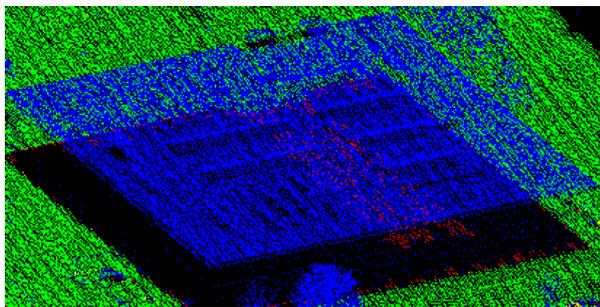


Figure 10. Error on point clouds inside building (red points)

Another error source is points inside buildings as seen in Figure 10. Due to the point cloud density, many laser pulses penetrated inside building through the small gaps on the roof, window, or wall. It can be seen many points inside building on the ground floor. Those points were misclassified as ground. This error is due to the similarity between the elevation of those points and the surrounding terrain. If this is considered to be a problem, manual editing using building footprints is needed.

Overall, the proposed method works well for minimizing total error and type I error, but it has higher type II error than the other methods. The conversion is an important step, but on the other hand, the conversion leads to some errors. It could happen since data conversion always loses some information. For instance, if there is more than one point within a pixel, the

conversion only takes the lowest point while ideally the conversion should manage all points.

4.3 Computational cost

Computational cost is the main motivation behind proposing this method. Table 6 shows the computational time compared to the previous Deep Learning approach and LAStools software in the ISPRS dataset. The comparison was performed on Intel Core i7-6700HQ 2.6GHz, 16 GB RAM, and Nvidia Quadro M1000M 2GB.

Method	Point to image conversion (15 samples)	Training (10 samples)	Testing (5 samples)
FCNs	± 36 minutes	± 12 hours	7.8 sec
CNNs	± 47 hours	± 2.5 hours	126 sec
LAStools	-	-	3.85 sec

Table 6. Time Comparison

The proposed method was significantly faster when converting the point cloud to image as the conversion does not convert each point into a separate image as done in previous work. Testing was also faster because fewer numbers of images were involved. Training was slower, but once the networks have been trained, it can be used to classify another dataset without the need to train the network anymore.

5. CONCLUSIONS AND FUTURE WORK

The motivation of this work is to improve the performance of ground classification using deep learning in terms of computational cost. Previously, deep learning was used by classifying feature images which were created from every point. This approach is slow, especially when converting points to feature images. The proposed method is more efficient because all points were converted into one image; it avoids redundant calculation as in the previous approach. As a result, conversion is 78 times faster while classification is 16 times faster. Furthermore, the proposed method gives a better accuracy; the total error is 2.69% lower than the previous deep learning approach based on the same limited training data set. The use of more representative training data may improve the test results, so it is worth to train our FCN with more training data in future work. The proposed method performs better because it has more features rather than one height difference feature as used in the previous method.

In the future, the point-to-image conversion could be more thoroughly investigated since the conversion is a critical step in the proposed workflow. Different parameter settings such as pixel size might affect the results. A better approach is also needed to handle cases of more than one point within a pixel.

Finally, all deep learning schemas here are based on CNN architectures. The drawback is that it needs to convert point clouds into an image before CNNs can consume the data. Recently, an architecture that allow the network to process the point cloud directly has been proposed (Qi et al., 2017). It can be implemented for ground classification if point-to-image conversion is to be avoided.

REFERENCES

- Axelsson, P., 2000. DEM Generation from Laser Scanner Data Using adaptive TIN Models. *International Archives of Photogrammetry and Remote Sensing*, 23(B4), 110–117.
- Badrinarayanan, V., Kendall, A., & Cipolla, R., 2015. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12), 2481–2495.
- Chehata, N., Guo, L., & Mallet, C., 2009. Airborne Lidar feature Selection for urban classification using Random Forests. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII(Part 3 / W8), 207–212.
- Chen, Z., Gao, B., & Devereux, B., 2017. State-of-the-Art: DTM Generation Using Airborne LIDAR Data. *Sensors*, 17(1), 150.
- Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J., 2011. Flexible, high performance convolutional neural networks for image classification. *IJCAI International Joint Conference on Artificial Intelligence*, 1237–1242.
- Fu, G., Liu, C., Zhou, R., Sun, T., & Zhang, Q., 2017. Classification for high resolution remote sensing imagery using a fully convolutional network. *Remote Sensing*, 9(5), 1–21.
- Gevaert, C. M., Persello, C., Nex, F., & Vosselman, G., 2018. A Deep Learning Approach to DTM Extraction from Imagery Using Rule-Based Training Labels.
- Goodfellow, I., Bengio, Y., & Courville, A., 2016. *Deep Learning*. MIT Press.
- Hu, X., & Yuan, Y., 2016. Deep-Learning-Based Classification for DTM Extraction from ALS Point Cloud. *Remote Sensing*, 8(730), 1–16.
- Kilian, J., Haala, N., & Englich, M., 1993. Capture and evaluation of airborne laser scanner data. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 31, 383–388.
- Kraus, K., & Pfeifer, N., 1998. Determination of terrain models in wooded areas with airborne laser scanner data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 53(4), 193–203.
- Le Cun, Y., Bottou, L., Bengio, Y., & Haffner, P., 1998. Gradient-Based Learning Applied to Document Recognition.
- Long, J., Shelhamer, E., & Darrell, T., 2017. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4), 640–651.
- Maggiori, E., Tarabalka, Y., Charpiat, G., & Alliez, P., 2017. Convolutional Neural Networks for Large-Scale Remote-Sensing Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2), 645–657.
- Maturana, D., & Scherer, S., 2015. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. *Iros*, 922–928.
- Mongus, D., & Zalik, B., 2012. Parameter-free ground filtering of LiDAR data for automatic DTM generation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 67(1), 1–12.
- Nie, S., Wang, C., Dong, P., Xi, X., Luo, S., & Qin, H., 2017. A revised progressive TIN densification for filtering airborne LiDAR data. *Measurement: Journal of the International Measurement Confederation*, 104, 70–77.
- Niemeyer, J., Rottensteiner, F., & Soergel, U., 2012. Conditional random fields for LiDAR point cloud classification in complex urban areas. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1-3(September), 263–268.
- Persello, C., & Stein, A., 2017. Deep Fully Convolutional Networks for the Detection of Informal Settlements in VHR Images. *IEEE Geoscience and Remote Sensing Letters*, 14(12), 2325–2329.
- Pfeifer, N., Stadler, P., & Briese, C., 2001. Derivation Of Digital Terrain Models In The Scop++ Environment. In *OEEPE Workshop on Airborne Laserscanning and Interferometric SAR for Digital Elevation Models*.
- Qi, C. R., Yi, L., Su, H., & Guibas, L. J., 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv*.
- Sherrah, J., 2016. Fully Convolutional Networks for Dense Semantic Labelling of High-Resolution Aerial Imagery. *arXiv*, 1–22.
- Simonyan, K., & Zisserman, A., 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proc. Int. Conf. Learn. Represent.* (pp. 1–14).
- Sithole, G., & Vosselman, G., 2004. Experimental comparison of filter algorithms for bare-Earth extraction from airborne laser scanning point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 59(1–2), 85–101.
- Sithole, G., & Vosselman, G., 2005. Filtering of airborne laser scanner data based on segmented point clouds. *ISPRS WG III/3, III/4, V/3 Workshop "Laser Scanning 2005,"* 66–71.
- Volpi, M., & Tuia, D., 2017. Dense Semantic Labeling of Subdecimeter Resolution Images With Convolutional Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2), 881–893.
- Vosselman, G., 2000. Slope based filtering of laser altimetry data. *International Archives of Photogrammetry and Remote Sensing*, Vol. 33, Part B3/2, 33(Part B3/2), 678–684.
- Weinmann, M., Jutzi, B., Hinz, S., & Mallet, C., 2015. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105, 286–304.
- Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J., 2015. 3D ShapeNets: A deep representation for volumetric shapes. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07–12–June, 1912–1920.
- Yu, F., & Koltun, V., 2016. Multi-Scale Context Aggregation by Dilated Convolutions. In *Proc. Int. Conf. Learn. Represent.* (pp. 1–13).
- Zhang, J., Lin, X., & Ning, X., 2013. SVM-Based classification of segmented airborne LiDAR point clouds in urban areas. *Remote Sensing*, 5(8), 3749–3775.