





Article

# Ground and Multi-Class Classification of Airborne Laser Scanner Point Clouds Using Fully Convolutional Networks

Aldino Rizaldy <sup>1,2,\*</sup>, Claudio Persello <sup>1,\*</sup> , Caroline Gevaert <sup>1</sup> , Sander Oude Elberink <sup>1,\*</sup>  and George Vosselman <sup>1</sup> 

<sup>1</sup> Faculty of Geo-Information Science and Earth Observation, University of Twente, P.O. Box 217, 7514 AE Enschede, The Netherlands; c.m.gevaert@utwente.nl (C.G.); george.vosselman@utwente.nl (G.V.)

<sup>2</sup> Center for Topographic Base Mapping and Toponym, Geospatial Information Agency (BIG), Bogor 16911, Indonesia

\* Correspondence: aldino.rizaldy@big.go.id (A.R.); c.persello@utwente.nl (C.P.); s.j.oudeelberink@utwente.nl (S.O.E.); Tel.: +6221-875-2062 (A.R.); +31-53-487-4343 (C.P.); +31-53-487-4350 (S.O.E.)

Received: 21 September 2018; Accepted: 24 October 2018; Published: 31 October 2018



**Abstract:** Various classification methods have been developed to extract meaningful information from Airborne Laser Scanner (ALS) point clouds. However, the accuracy and the computational efficiency of the existing methods need to be improved, especially for the analysis of large datasets (e.g., at regional or national levels). In this paper, we present a novel deep learning approach to ground classification for Digital Terrain Model (DTM) extraction as well as for multi-class land-cover classification, delivering highly accurate classification results in a computationally efficient manner. Considering the top-down acquisition angle of ALS data, the point cloud is initially projected on the horizontal plane and converted into a multi-dimensional image. Then, classification techniques based on Fully Convolutional Networks (FCN) with dilated kernels are designed to perform pixel-wise image classification. Finally, labels are transferred from pixels to the original ALS points. We also designed a Multi-Scale FCN (MS-FCN) architecture to minimize the loss of information during the point-to-image conversion. In the ground classification experiment, we compared our method to a Convolutional Neural Network (CNN)-based method and LAsTools software. We obtained a lower total error on both the International Society for Photogrammetry and Remote Sensing (ISPRS) filter test benchmark dataset and AHN-3 dataset in the Netherlands. In the multi-class classification experiment, our method resulted in higher precision and recall values compared to the traditional machine learning technique using Random Forest (RF); it accurately detected small buildings. The FCN achieved precision and recall values of 0.93 and 0.94 when RF obtained 0.91 and 0.92, respectively. Moreover, our strategy significantly improved the computational efficiency of state-of-the-art CNN-based methods, reducing the point-to-image conversion time from 47 h to 36 min in our experiments on the ISPRS filter test dataset. Misclassification errors remained in situations that were not included in the training dataset, such as large buildings and bridges, or contained noisy measurements.

**Keywords:** LIDAR; DTM extraction; filtering; classification; deep learning; Convolutional Neural Network

## 1. Introduction

Digital Terrain Models (DTM) can be generated by classifying a point cloud into ground and non-ground classes. This task is also known as filtering [1]. The point cloud is usually derived from an

Airborne Laser Scanner (ALS). Even though a point cloud could also be derived from photogrammetric images using a dense image-matching technique, ALS data offer the advantage of penetrating through the vegetation canopy to reach the ground surface. It is useful for DTM extraction because the ground surface under the vegetation can be detected from ALS data while this is unlikely when photogrammetric point clouds are used. DTMs are not only crucial for geospatial information analysis, they also play a vital role for the further classification of point clouds when the classifier uses the height above the ground as an important feature [2–6].

Traditional algorithms for ground classification are mostly based on unsupervised classification. A filtering test has been conducted to compare the performance of different filtering algorithms on different terrain scenes [7]. Although the algorithms work well in landscapes with low complexity such in the smooth terrain with small and simple buildings, some scenes of terrain cannot be perfectly defined, and lead to inaccurate results [8]. Complex structures in urban areas also lead to the misclassification of ground points [1]. Moreover, various challenges in the DTM extraction are shown in Gevaert et al. [9].

Deep learning strategies using Convolutional Neural Networks (CNNs) have been used massively in recent years. Deep CNNs consistently outperform other classifiers for various image classification tasks [10]. Unlike other machine learning classifiers, CNNs learn spatial–contextual features from the image directly, avoiding the difficult task of feature engineering as commonly done in traditional machine learning techniques. This ability is achieved due to the architecture of CNNs, which employ a set of learnable filters. Initially, all of the filters are randomized; then, the filters are trained during the training phase. As a result, the trained filters capture the important spatial features directly from the input image without the need for feature engineering.

Following the popularity of deep learning, a CNN-based technique was proposed to be used to classify point clouds into ground and non-ground for DTM generation [11]. The method achieved lower error rates compared to other filtering algorithms in an ISPRS (International Society for Photogrammetry and Remote Sensing) filter test dataset [11]. The ISPRS filter test dataset is a benchmark light detection and ranging (LIDAR) dataset for analyzing the performance of filtering algorithms. The dataset consists of 15 areas with various and difficult terrain types to challenge the algorithms. However, their CNN-based method does not process point clouds directly. It converts each point into  $128 \times 128$  pixels of a feature image so that a CNN can process the data. The feature image captures the spatial pattern of the neighbors of each point. The neighbors are defined as all of the neighboring points within a horizontal window of  $96 \times 96$  m. After feature images are extracted for all of the points, a deep CNN was trained using those images in order to separate out ground feature images and non-ground feature images. Although the CNN-based method can produce accurate classifications, the point-to-image conversion is inefficient due to highly redundant calculations. This prevents the application of the CNN-based method to large volumes of ALS data for regional or national level studies.

Recently, deep learning methods have been introduced in the computer vision literature that can operate directly on three-dimensional (3D) points without the conversion to images, e.g., PointNet, PointNet++, SplatNet, etc. [12–14]. These techniques are applied to indoor 3D points or building façades. Their strength is on classifying points of objects that have been captured from various perspectives and scales. On the contrary, ALS data are acquired from a top–down view. Therefore, a projection of the data to a horizontal plane will not result in a significant loss of information, but it will speed up the process tremendously. Our contribution is in the development of an efficient algorithm that is not only able to filter an ALS point cloud into ground and non-ground, but is also able to distinguish buildings from trees. The information needed to do so can be generated by using multi-dimensional two-dimensional (2D) images, instead of 3D point clouds.

We use point-to-image conversion following the approach adopted in Hu and Yuan [11]. However, our method converts all of the points into a multi-dimensional image to accelerate the computational time. The lowest point within a pixel resolution cell is used when calculating each pixel value.

As a result, points' features are represented by pixel values in the extracted image. Consequently, the point classification task is transformed to a pixel-wise image classification task. To address this task, we introduce a Fully Convolutional Network (FCN), which is a CNN variant that can predict the classification labels of every pixel in the image directly. We adopt FCN with dilated kernel (FCN-DK) for the classification [15]. FCN-DK is a no down-sampling network architecture that maintains the spatial size of the feature maps of each layer to be the same as the input. It uses dilated kernels to capture larger spatial contextual information, and therefore increases the receptive field of the network without increasing the number of parameters. We modify the FCN-DK network to perform ground and multi-class classification of an ALS point cloud. We also propose Multi-Scale FCN (MS-FCN) architecture for the classification of high-density point clouds. Using a multi-scale approach, more information is added to the network by employing different pixel sizes simultaneously, which is expected to improve the classification result. In the multi-class classification, we further classify the non-ground points into finer classes (e.g., building and vegetation) by expanding the network architecture. A thorough analysis of the investigated techniques is presented in our experimental section.

## 2. Related Works

This section will focus on reviewing the literature on ground point filtering from LIDAR (light detection and ranging) data. Traditionally, there are four approaches. A fifth strategy based on deep learning has been recently introduced.

1. Slope-based filtering [16]. It is based on the assumption that the terrain is relatively flat. If there is a significant height difference between two nearby points, then it is not caused by the terrain slope; instead, it is caused by both points consisting of ground and non-ground points where a ground point is positioned at a lower height than a non-ground point. Slope-based filtering uses erosion and dilation from mathematical morphology for its implementation. Some revisions have been developed for slope-based filtering by modifying the structuring element [17] or using an adaptive filter [18].
2. Progressive densification [19]. It is based on the assumption that the lowest point in a particular area should be the ground point. These points initially are assigned as seed points; then, the algorithm creates a Triangulated Irregular Network (TIN) from these points and the surrounding points. The neighboring points are decided as ground or non-ground points based on the angle and distance parameters. Next, TINs are created progressively for the next points. These iterative steps gradually build terrain surfaces. A revised version of progressive densification was proposed to avoid the misclassification of a non-ground point into a ground point by changing the angle criterion [20].
3. Surface-based filtering approach [21]. It relies on the assumption that all of the points belong to the ground, and removes the points that do not fit as ground. In the beginning, the algorithm gives the same weights for all of the points and creates best-fitted surfaces for all of the points, and then iteratively changes the weight based on the assumption that points below the surface are ground, and points above surface are non-ground. In the next iteration, all of the points below the surface have higher weight, while points above the surface have lower weight, and a new surface is created based on the new weights. These steps iterate until they converge, and the final surface in the last iteration should be the ground surface. An improvement was proposed by adding a robust interpolation method to deal with large buildings and minimize the computation time [22].
4. Segment-based filtering [23]. Unlike other algorithms, this approach works on segments, rather than points. This approach creates segments of similar points and analyzes which segments belong to the ground. If ground points are grouped into the same segment, while non-ground

points are grouped into their own segments, one can classify which segments belong to either ground or non-ground based on the geometric and topological information.

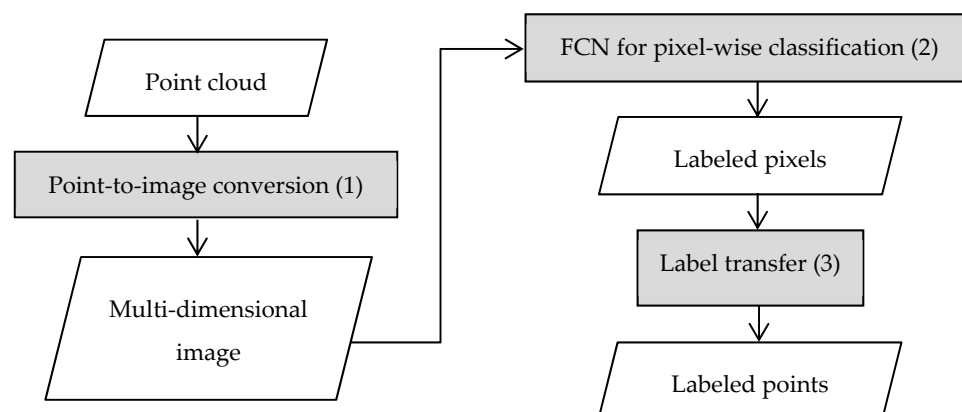
- Since 2016, another filtering algorithm approach was introduced based on deep learning classification [11]. This approach extracts the feature images for every single point to represent the spatial pattern of each point to its neighborhood. A large number of 17 million points are used for training samples to train the deep CNN model in order to successfully discriminate ground and non-ground points in many different landscapes. It proves that if the network is trained properly, the accuracy result of deep learning is considerably high.

Although DTM extraction using deep learning is promising, we see the challenging situation for the point-to-image conversion in areas that include hilly areas combined with low vegetation, for example. Such a conversion also is not efficient due to redundant calculations when converting the neighborhood of each point into a feature image. In this paper, we propose an efficient method by converting all of the points simultaneously into a single image instead of converting a single point into an image patch. In addition to a similar FCN-based for DTM extraction [9], we add a multi-scale network to improve the result.

In recent years, the multi-class classification of an ALS point cloud has been investigated. Various feature sets were proposed to classify a point cloud including height features, echo features, eigenvalue features, local plane features, and full waveform features [24,25]. Contextual information using Conditional Random Field was also used either by point-based [4] or segment-based techniques [6]. A thorough study of different feature sets and classifiers for point cloud classification was also reported [26,27]. In this paper, we expand our ground classification method to further classify the non-ground points into finer classes using the extended network architecture to achieve a multi-class classification.

### 3. Proposed Methods

The general workflow of the proposed approach is shown in Figure 1. We exploit the potential of the FCN in terms of classification accuracy and efficiency in pixel-wise image classification. In order to use the FCN for point cloud analysis, we design our classification system in three steps. The first step is point-to-image conversion. This step is needed in order to process a point cloud with the network. The output of this step is a multi-dimensional image. The second step is a pixel-wise classification using FCN, resulting in labeled pixels. The third step is label transfer from pixels to points, so that all of the points can be labeled.



**Figure 1.** The general workflow of the proposed classification approach.

#### 3.1. Point-to-Image Conversion

We propose a more efficient classification system by first projecting a 3D LIDAR point cloud into a 2D image by calculating each pixel value based on the features of the lowest point within

each pixel [28]. Lowest points are more likely to belong to the ground than upper points if there is no outlier in the data. Four features are involved: elevation, intensity, return number, and height difference. The first three features are chosen, because those features are the original information of the LIDAR point cloud. The height difference feature is added and defined between the lowest point in the corresponding pixel and the lowest point in a  $20 \times 20$  m horizontal window centered on the point. The size of  $20 \times 20$  m is selected, as the assumption that most of the buildings are smaller than  $20 \times 20$  m. This feature is added, since non-ground objects are usually located higher than the ground surface; hence, non-ground points are expected to have higher feature values than ground points.

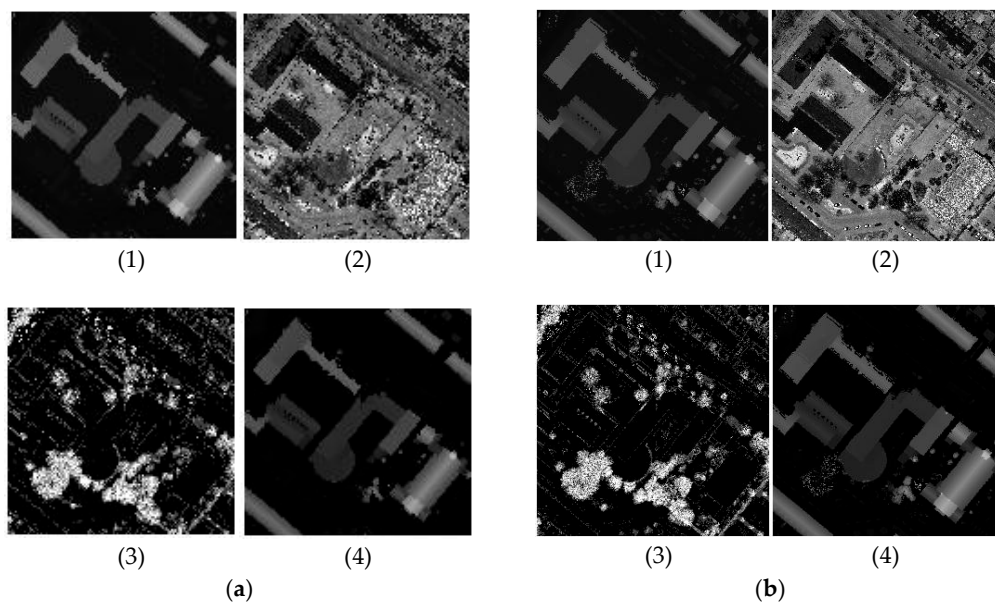
The technique introduced in Hu and Yuan [11] captures the pattern of each point during point-to-image conversion, and uses those patterns to capture the features during CNN training to separate ground and non-ground points. In contrast to that, our method lets the FCN learn the spatial pattern of ground and non-ground objects directly to the image during the training of the network. In Hu and Yuan [11], the point-to-image conversion is done for each point separately, resulting in a largely redundant computation. In our work, we convert the whole ALS point cloud once into a multi-dimensional image. The extracted image can then be used as the input of our FCN. Therefore, our approach results in a significantly faster conversion time. However, projecting 3D points into a 2D image may result in a loss of information. We produce two series of images. The first series is based on the lowest point per pixel for a ground classification. The second series is based on the highest point per pixel to classify above-ground points. As each pixel is only able to accommodate the lowest and the highest points in one pixel, the classification is done at the pixel level. The rest of the points will not be taken into account during the conversion. In order to have all of the points labeled, a simple processing step is applied, as explained in Section 3.3.

The pixel size is defined depending on the point density of the point cloud to be processed. In our experiment, the pixel size is set to  $1 \times 1$  m on the assumption that there is at least one point in one square meter, so that we avoid many empty pixels. However, empty pixels still remain in the image due to the water bodies or data gaps. In these cases, the pixel value is interpolated from the neighboring pixels. It remains unlabeled, so that the network does not use it during the training. On the other hand, a higher point density needs a smaller pixel size in order to capture a smaller structure. Therefore, we also use the  $0.5 \times 0.5$  m pixel size for the Actueel Hoogtebestand Nederland 3 (AHN3) dataset. This is a higher density ALS dataset acquired in the Netherlands that enables us to examine the effect of different pixel sizes on the classification accuracy. We investigate using a multi-scale image (1 m and 0.5 m) as an input to the network. In the classification of multispectral satellite imagery with spectral channels of different spatial resolution (i.e., panchromatic and multispectral bands), the use of a multi-scale network (called FuseNet) was recently proven to be more accurate than traditional networks applied to pan-sharpened images [29]. The network configuration for the multi-scale input can be seen in the Section 3.2.1. The difference of a 1-m pixel size and a 0.5-m pixel size is shown in Figure 2a,b.

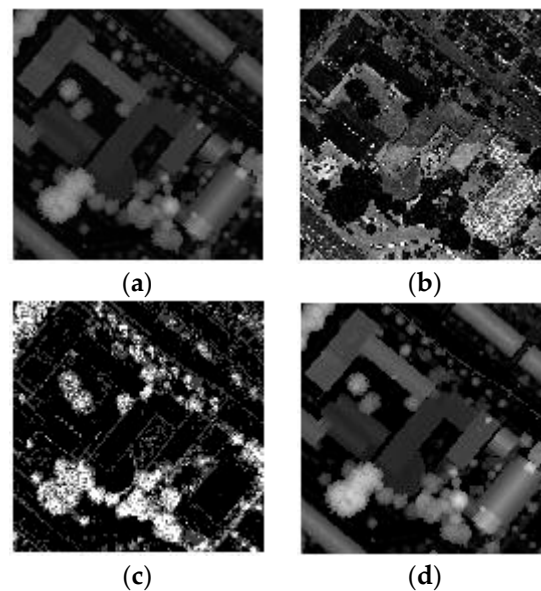
In this paper, we also propose a classification system for multi-class classification. More images are extracted from the point cloud to add more useful information. While the conversion for the ground classification relies on the lowest point within each pixel, the conversion for multi-class classification also uses the highest point within each pixel. The idea is to ‘see’ the scene from above when projecting the 3D point cloud into the 2D image. Similar to ground classification, four features were used in the conversion, creating four channel images. Those are elevation, intensity, the number of returns, and the height difference in a  $20 \times 20$  m horizontal window.

Note that when we convert the point cloud into an image using the highest point, we do not use the return number; instead, we use the number of returns. The reason is that the highest point within each pixel is most likely to be the first return, so the return number will not be helpful in discriminating vegetation from buildings. If this first return refers to vegetation, the same pulse will most likely have multiple returns, whereas objects such as a building will only have a single return. Therefore, by using the number of returns as a feature, we are including information on the penetrability of the

object below the highest point. Figure 3 shows the additional image using the highest points within each pixel.



**Figure 2.** Subset of extracted images from different pixel sizes: (a) 1 m; (b) 0.5 m. Each image has four feature channels: (1) elevation, (2) intensity, (3) return number, and (4) height difference between the lowest point in a  $20 \times 20$  m neighborhood, and the lowest point in a pixel. A higher resolution image captures more detailed structures.



**Figure 3.** Images from the highest point within each pixel: (a) elevation; (b) intensity; (c) the number of returns; (d) height difference between the highest point in a pixel and the lowest point in a  $20 \times 20$  m neighborhood.

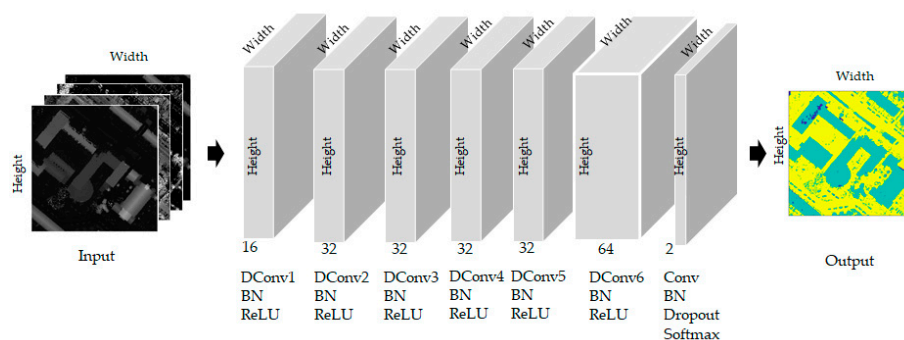
### 3.2. Fully Convolutional Network

Fully Convolutional Networks (FCNs) are a modification of CNNs in which the results are labeled pixels, while CNNs are initially designed to output labeled images. The pixel-wise result can be obtained by using several architectures. Initially, the authors in Long et al. [30] proposed to replace the fully connected layers in CNN architecture with up-sampling layers to change the architecture from

CNNs into FCNs. The up-sampling layer restores the resolution of the feature maps into the original resolution. In this architecture, up-sampling is mandatory, because the output feature maps are smaller after the input is passed through several convolutional and pooling layers. The up-sampling is done using a deconvolutional (or transposed convolution) filter. It works by connecting the coarse output feature maps (from several down-sampling layers) to the dense pixels. The filter itself can be learned instead of using fixed value such as bilinear interpolation. Another popular network for pixel-wise classification was proposed by Badrinarayanan et al. [31]. The network, which is called SegNet, uses an encoder and a decoder to down-sample and up-sample the feature maps, respectively. Unlike the FCN by Long et al. [30], the decoder of SegNet uses pooling indices from the corresponding encoder to create up-sampled maps. While previous networks used the ‘down-sample and up-sample’ approach, a different approach was introduced by maintaining the size of each feature map as the same as the input; hence, it avoids the need for an up-sample layer [15].

### 3.2.1. Network Architectures for Ground Classification

The basis of the adopted network is the FCN-DK network [15]. The network was originally designed for the detection of informal urban areas in satellite images. We have also investigated using various network architectures by modifying the FCN-DK network in order to work with a multi-scale input image, and also label all of the points for multi-class classification. We adapt and fine-tune the architecture of the FCN-DK network for the purpose of ground classification. Max-pooling layers are removed from the original architecture, leaving only convolutional, batch normalization (BN), and Rectified Linear Unit (ReLU) layers. Eliminating the max-pooling layer results in a higher accuracy yet makes the network simpler. Similar to the FCN-DK network, six convolutional layers are involved in the network. In order to maintain the size, the stride is set to one in each layer. While down-sampling a layer enables the network to capture a larger spatial extent in the following layer, a network without down-sampling needs to have a larger filter in the subsequent layer to have the same result. However, a larger filter results in more parameters in the network. To avoid having a vast number of weights, dilated convolutions are introduced [32]. Dilated filters increase the size of the receptive field, but keep the same number of parameters. It could be achieved by adding zero values instead of new parameters when increasing the size of the receptive field. In this way, no more parameters are added, and the network is kept simple. The dilation factor gradually increases from one to six, so that every following layer captures a larger spatial extent. Finally, the last layer accumulates the features of each pixel from the smaller to the larger spatial extent, and is connected to the reference image with the corresponding label on each pixel. Figure 4 illustrates the adopted network.



**Figure 4.** The Fully Convolutional Network with dilated kernel (FCN-DK) network architecture.

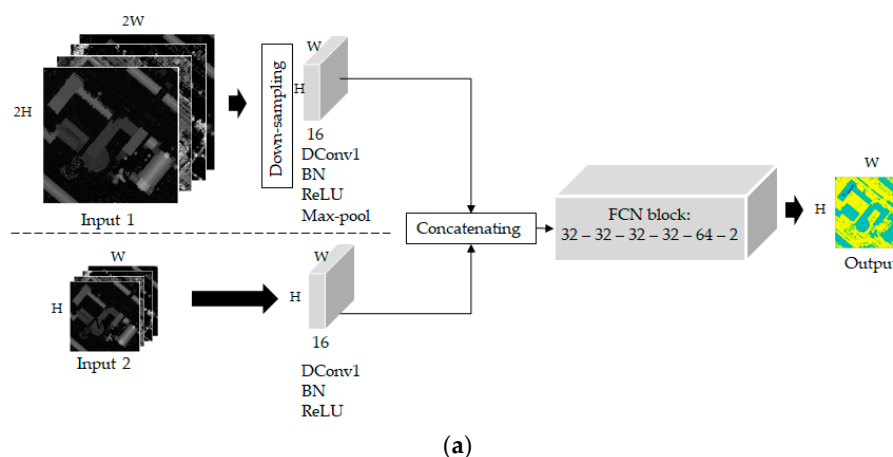
Table 1 shows the details of the network, including the receptive field size in each convolutional layer. Note that the receptive field size increases gradually as the dilation factor increases, but the memory required remains the same, although the receptive field size increases. For a 1-m pixel size, the size of the receptive field on the ground is the same as the size of the pixel.

**Table 1.** The detailed architecture of the network.

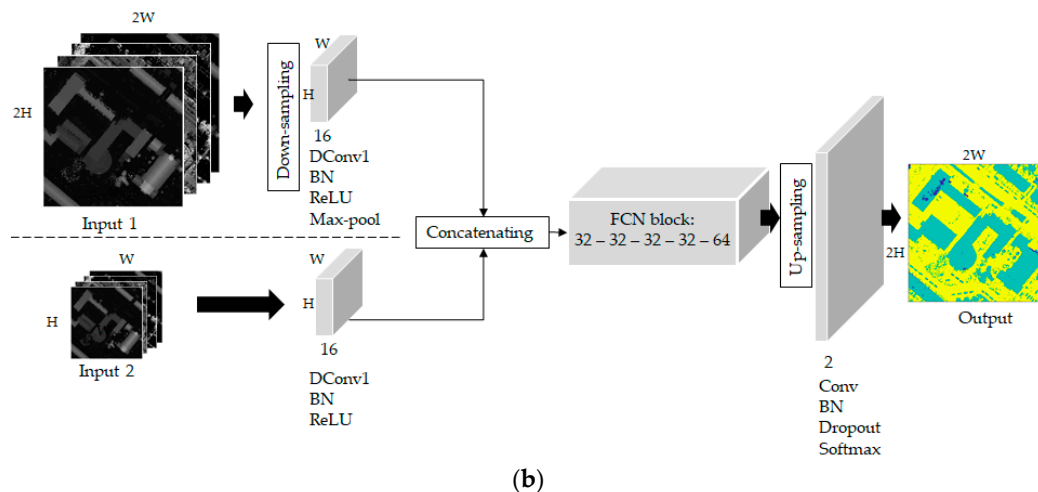
Layer	Filter Size	Number of Filters	Dilation Factor	Receptive Field Size (Pixel)	Memory Required (Megabytes)
DConv1	5 × 5	16	1	5 × 5	22
DConv2	5 × 5	32	2	13 × 13	43
DConv3	5 × 5	32	3	25 × 25	43
DConv4	5 × 5	32	4	41 × 41	43
DConv5	5 × 5	32	5	61 × 61	43
DConv6	5 × 5	64	6	85 × 85	86
Conv	1 × 1	2	-	1 × 1	3

As mentioned earlier, point-to-image conversion may lead to losing some of the information of the 3D point cloud. Hence, a multi-scale image is proposed to minimize the loss of information due to the conversion from a 3D point cloud into a 2D image. Two pixel sizes (1 m and 0.5 m) are used, as explained in Section 3.1.

Down-sampling and up-sampling layers are involved in the network to handle a multi-scale image in the Multi-Scale FCN (MS-FCN) network. At first, the network takes as the input an image of 0.5-m resolution, followed by convolutional, BN, ReLU, and max-pooling layers. In this architecture, a pooling layer is needed in order to process the different image resolutions by down-sampling the image of 0.5-m resolution into the same as an image of 1-m resolution. Therefore, the network can process both images in the following layer. Since the size of the 0.5-m image is always two times larger than the size of the 1-m image, the stride of the pooling layer is set to two, so that the output size is reduced by half. It allows concatenating the output maps of the 0.5-m image to the 1-m image. However, instead of directly concatenating them, the 1-m image is also filtered by a convolutional filter first to have the same depth and the same level of information. The next layers in the network follow the same architecture as the network in Figure 4. There are two options regarding the resolution of the reference image, either using 1 m or 0.5 m. If it uses 1 m, it simply leaves the rest of the network as exactly the same as the plain network; hence, it is called MS-FCN Down, because only a down-sampling layer was involved. However, if it uses 0.5 m, an up-sampling layer is added to restore the resolution; therefore, the name is MS-FCN Down-Up. The up-sampling factor is set to two. The up-sampling layer is modified from Long et al. [30]. Figure 5 shows the architectures of MS-FCN Down and MS-FCN Down-Up.

**Figure 5.** Cont.





**Figure 5.** The proposed multi-scale networks: (a) Multi-Scale (MS)-FCN Down (without up-sampling layer); (b) MS-FCN Down-Up (with up-sampling layer).

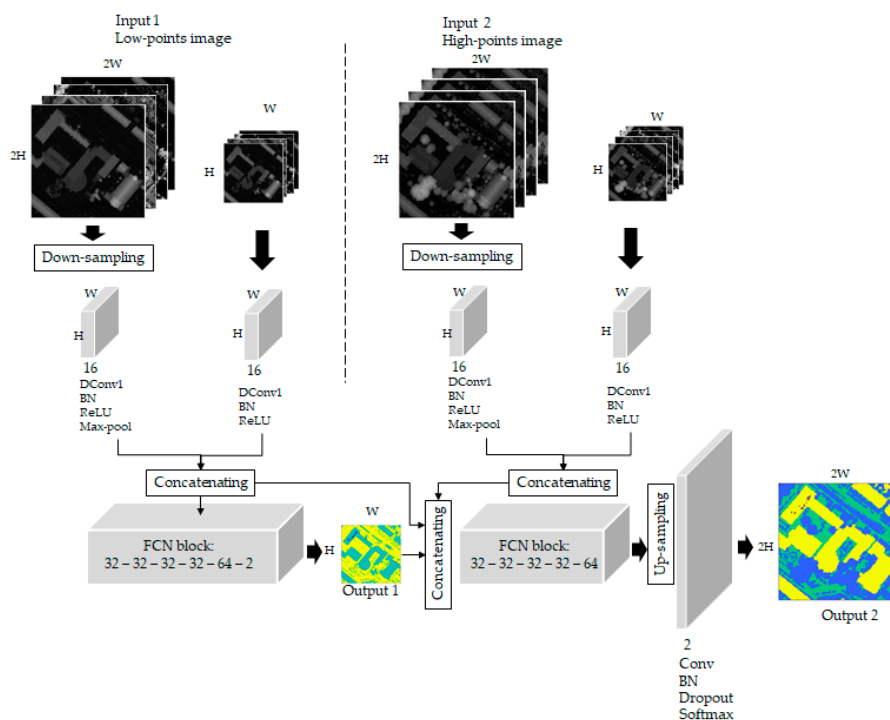
### 3.2.2. Network Architectures for Multi-Class Classification

In our previous networks for ground classification, labels are only given once for each pixel. In that sense, points within one pixel cannot have different labels, while the typical situation in the ALS point cloud is that points on a lower elevation are ground, while the upper points belong to one of the non-ground classes. Given such a case, we expand our network to have different labels in one pixel. The idea is to have labels for ground classification and labels for the further classification of non-ground points.

In order to have two labels in one pixel, we use two loss functions in a Multi-Task Network (MTN), which is named MTN-FCN. Unlike a Single-Task Network (STN), the MTN is used where two different tasks are executed in the same network simultaneously [33]. In our network, the first task is the classification of the lowest points within a pixel as ground or non-ground for DTM extraction. The second task is the multi-class classification of the highest points into vegetation and building. We only add two classes, because the AHN3 dataset only has those two classes for the above-ground points. More classes can be added if the reference data have more classes. The network is constructed by connecting two blocks of FCN, whereas each block has its loss. Unlike Ko et al. [33], where two loss functions are stacked in a parallel way, we design our network by stacking two loss functions in a serial way to have a benefit from the first loss function to the second loss function.

We also use a multi-scale input as shown in the MS-FCN architecture above. The motivation is that the classification is done at the pixel level, whereas there is always a possibility of vegetation and building points being mixed in one pixel. In that case, it makes sense to use a smaller pixel size better differentiate between vegetation and building. However, instead of using a single smaller pixel size, we use a multi-scale resolution to obtain a benefit from different pixel sizes. In other words, we use a higher pixel size to capture the smaller object and a lower pixel size for a faster processing time.

Two sets of images as mentioned in Section 3.1 (namely, low-points image sets and high-points image sets) with two pixel sizes involved as the input of the network. The architecture of the first block is similar to the MS-FCN Down, as shown in Figure 5. The input is the low-points image, and the output is the prediction of two ground classification task classes. The second block is designed for classifying the non-ground points into finer classes. The input is the low-points image and the high-points image, as well as the prediction map from the first block. The motivation of employing the result of the first block is to obtain a benefit from the ground classification task when the ground and non-ground pixels are already labeled. In the training stage, this extended network runs slower due to the complexity and the larger number of parameters. Figure 6 shows the architecture of the MTN-FCN for multi-class classification.



**Figure 6.** The proposed Multi-Task Network (MTN)-FCN network for multi-class classification on a point cloud dataset.

### 3.2.3. FCN Training and Testing

In order to train the network, patches of image are created randomly for all of the training sets. The patch has a size of  $100 \times 100$  pixels of 1-m resolution and  $200 \times 200$  pixels of 0.5-m resolution. Each patch is provided with a corresponding labeled patch. The size of the patch was chosen with an assumption that patch size should be larger than the largest building in the scene. While the size could be set as large as possible, the larger patch leads to heavier computation during the training. We then use both patch sizes for the network with a multi-scale image. The size of the labeled patch is  $200 \times 200$  pixels for a network with an up-sampling layer, while a network without an up-sampling layer has labeled patches of  $100 \times 100$  pixels. In a network for multi-class classification, patches of image are also created for both a low-points image and a high-points image with sizes of  $100 \times 100$  pixels.

The classification task can be seen as the task to predict a label  $y$  given input  $x$ . The input of the network is the patch image. The first layer processes the input image by employing the filters. The results are feature maps. Then, the subsequent layer processes the output from the previous layer and results in the second feature map. This process is called forward-passing, and is repeated through the whole layer until the final output is made. During the forward-passing, given an input image  $x$  with depth  $D$ , and a network with weights  $W$ , bias  $b$ , and ReLU as an activation function, the output maps  $h$  of each layer can be defined in Equation (1). After the input is passed to all of the convolution layers, the softmax function is introduced to distribute the values of each pixel on the output maps into the range  $[0, 1]$ . Since the depth of the final layer represents the number of classes, the class with the highest score is taken as a predicted label for the corresponding pixel.

$$h = \max \left\{ 0, \sum_{d=1}^D W^T_d \times x_d + b \right\} \tag{1}$$

Next, a loss is calculated as the negative log-likelihood between the prediction and the true label. Cross-entropy was chosen for the loss function due to its wide use in the modern neural network [34].

It can be seen from Equation (2) that the closer the prediction  $y$  to the true label  $y'$  for each sample  $j$ , the smaller the loss. Hence, training was performed by minimizing the loss with respect to all of the parameters in the network:

$$L(y, y') = - \sum_j y_j \log(y'_j) \quad (2)$$

Minimizing the loss can be done by adjusting all of the parameters in the network using back-propagation [35]. Stochastic gradient descent (SGD) with momentum is used for the learning of parameters.

The network architecture was implemented in the MatConvNet platform [36]. Learning was performed using SGD with momentum. The learning rate was 0.0001, the momentum was 0.9, and the weight decay was 0.0005, following the parameters set in Persello and Stein [15]. The rate of the dropout layer is 0.5. Each mini-batch has 32 samples. The network was trained for 50 epochs.

Testing was done by forward-passing the image into the network. The output is labeled pixels. The labels correspond to class that has been introduced in the training set. Due to its architecture, the network can consume any size of the input image.

### 3.3. From Labeled Pixels to Labeled Points

Since the output from the FCN is prediction labels at a pixel level, it has not solved the task of the classification of 3D points yet. A further processing step is needed to label the original ALS points. As mentioned in Section 3.1, pixel values are calculated based on the lowest point within a pixel. In that sense, the FCN only labels the lowest point, while the rest of the points remain unlabeled. Let the lowest points in the ground-labeled pixels become the initial ground points. If the ground terrain is defined as a smooth surface, one can densify the labeled points by creating a surface connecting all of the initial ground points, and then label all of the points according to the surface. If the difference of elevation between a point and the surface is within a threshold, then the corresponding point is labeled as a ground point. The threshold value is set to 15 cm based on the typical vertical accuracy of a point cloud from airborne LIDAR [37].

In a multi-class classification, the procedure is expanded. After ground points are obtained from the first predicted labels of the network, the remaining points (which are expected to be non-ground points) are labeled according to the labels of the corresponding pixels from the second prediction labels of the network.

## 4. Dataset and Results

### 4.1. Dataset

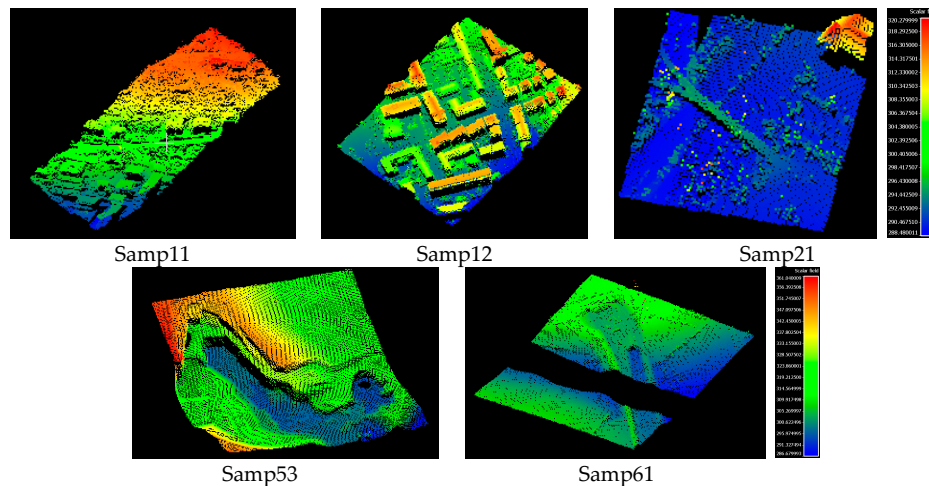
Two datasets were used in the experiment. The ISPRS filter test dataset was used as a benchmark dataset for ground classification, while the AHN3 dataset was used for a modern high point-density dataset, not only for ground classification, but also for the vegetation and building classification.

#### 4.1.1. ISPRS Filter Test Dataset

The ISPRS filter test dataset has 15 sample areas. The point clouds in all of the samples are manually labeled into ground and non-ground classes. Each sample has different terrain characteristics, and was chosen to challenge the algorithm for a specific condition such as a steep terrain or complex buildings. Ten samples were selected as training samples. Due to the limited number of training samples, two-fold cross-validation was performed during the validation phase. For each sample area, 300 patches were extracted randomly. Data augmentation was also conducted to increase the number of training patches by rotating the patch by 90°, 180°, and 270°. Hence, 12,000 patches were obtained to train the network.

Five samples were chosen for the testing set as shown in Figure 7. Samp11 has steep terrain combined with low vegetation and buildings. Samp12 is a flat terrain as a typical scene in an urban

area. Samp21 was chosen to show the result on a bridge. Samp53 has break-lines terrain. It is often difficult for filtering algorithms to handle ground points along the break-line, since those points have a height-jump with respect to the points in the lower terrain level; thus, they are often misclassified as non-ground points. Samp61 is a flat terrain combined with an embankment. An embankment is a man-made structure, but it is considered as a ground surface.



**Figure 7.** Five sample areas of the International Society for Photogrammetry and Remote Sensing (ISPRS) dataset chosen for the testing set.

Due to the low point density, which is around one point per square meter, only the method with  $1 \times 1$  m was executed for the ISPRS dataset. The low point density also makes the dataset more challenging for ground classification, because the terrain is poorly represented. It also has only two returns in contrast to up to five returns, as seen in many modern LIDAR point clouds. In a DTM extraction task, more returns give an advantage when the laser pulse could better penetrate the vegetation canopy to reach the ground surface. The dataset also has some outliers. Since our method is sensitive to low point outliers, we removed the outliers before we converted the point cloud into an image. A bare earth surface was created from the reference ground point; then, all of the points that had elevation of less than 1 m were removed. We used this clean dataset in the experiment for our method and the baseline methods as well. The ISPRS dataset can be accessed at <https://www.itc.nl/isprs/wgIII-3/filtertest/>.

#### 4.1.2. AHN3 Dataset

The Actueel Hoogtebestand Nederland (AHN) dataset covers the entire areas of the Netherlands, although the latest version, AHN3, will be completed in 2019. AHN3 offers a high point density: between eight and 10 points per square meter. It also records up to five returns. Ten sample areas were selected for training set, two were selected for the validation set, and another 10 sample areas were selected for the testing set. Each sample area had a size of  $500 \times 500$  m. To train the network, 300 patches were extracted randomly for each sample area; hence, 3000 patches were used in total. The dataset has a relatively flat terrain, which is a typical situation in the Netherlands. The buildings vary from small houses to large warehouses. A complex road and bridge structure was also added on the testing set to challenge the performance of the proposed method.

The AHN3 dataset is available in Laser (LAS) file format, a common file format to store airborne laser data. It has five labels following the class-code from American Society for Photogrammetry and Remote Sensing (ASPRS). The five classes are ground, vegetation, building, water, and bridge. During ground classification, all of the classes except ground are merged into the non-ground class in order to perform binary classification. In our experiment, points on the water are labeled within the non-ground class, although they could be labeled as ground points, as the geometry of the points on

the water is similar to the ground point. However, we labeled them as non-ground points in order to detect the original ground points only. In the multi-class classification, the labels of all of the original classes are used in the preparation of the training set. Therefore, all of the classes are also used in the prediction (testing). However, due to the limited number of points in the water and bridge classes, we excluded those classes when reporting the metric accuracy. The AHN3 dataset can be downloaded from <https://www.pdok.nl/nl/ahn3-downloads>.

#### 4.2. Results

The results of ground classification on the ISPRS and AHN3 datasets are presented in Sections 4.2.1 and 4.2.2 respectively, while the results of the multi-class classification are presented in Section 4.2.3.

##### 4.2.1. Ground Classification on ISPRS Dataset

The ISPRS dataset has a low point density. Therefore, we only used the  $1 \times 1$  m pixel size for the point-to-image conversion, since we did not see the advantage of using a higher pixel size. The results from our method were compared to deep CNN [11] and LAStools software (<https://rapidlasso.com/lastools/>). In LAStools, a different configuration setting was set between “forest and hill”, “town or flats”, and “city or warehouses”, according to the scene of each testing sample. Figure 8 shows our results compared to those of others.

The quality of ground classification is indicated by calculating the total error, type-I error, and type-II error. The total error is a percentage of misclassified points against all of the points. A type-I error is the percentage of the misclassified ground points into non-ground points, while a type-II error is the percentage of non-ground points misclassified as ground. A higher type-I error indicates that more ground points were labeled incorrectly, while a higher type-II error means that more non-ground points were labeled incorrectly. The total amount of errors, type-I errors, and type-II errors of all of the methods can be seen in Tables 2–4, respectively.

The results show that the FCN resulted in a lower total error than CNN-based classification or LAStools software. It proved that deep learning can be used for the ground classification of a LIDAR point cloud by converting the point cloud in a more efficient way, as shown in this paper, and using FCN architecture to handle the classification. However, it should be noted that the CNN classifier that was used here was trained on the 10 training sample areas of the ISPRS dataset, while when it was trained on an extensive training dataset of 17 million points in mountainous terrain, Hu and Yuan [11] reported better results (0.67% of total error, 2.26% of type-I error and 1.22% of type-II error).

**Table 2.** Total error on ISPRS dataset.

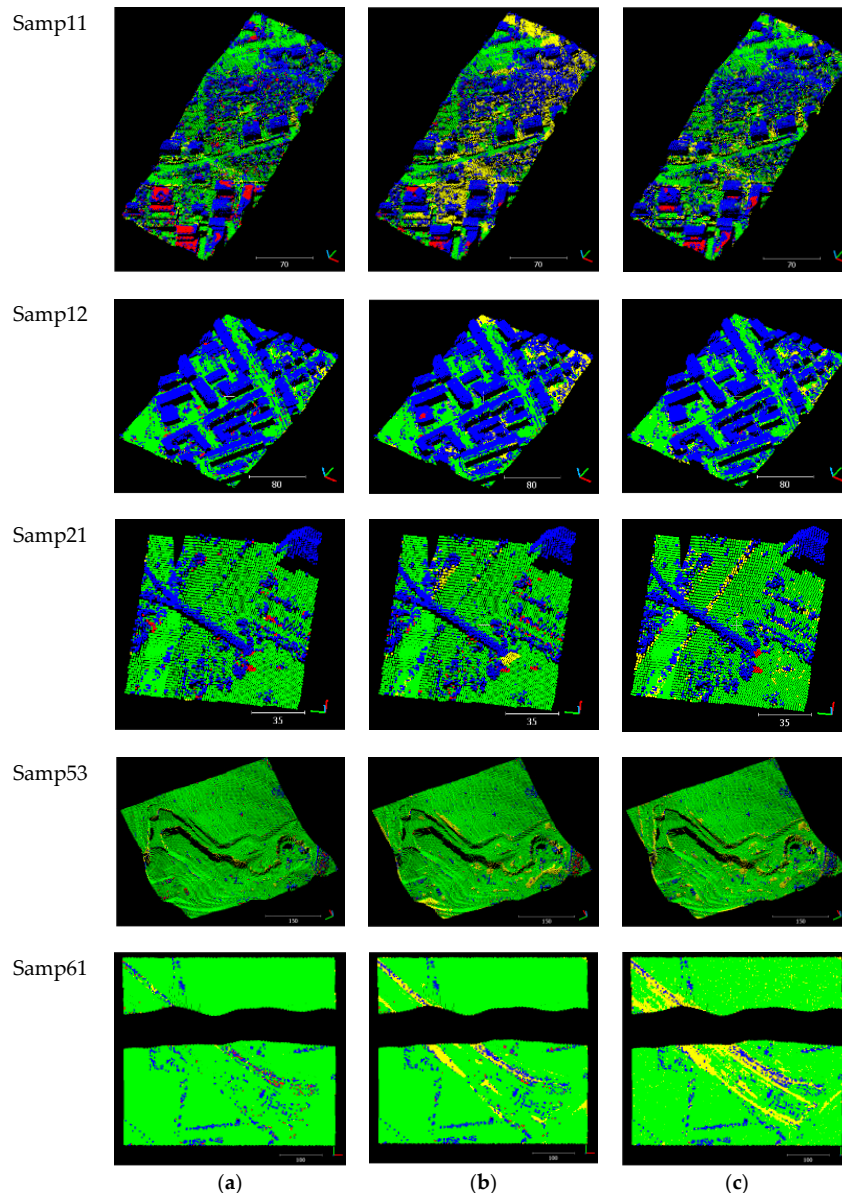
Sample	FCN-DK6	CNN [11]	LAStools
Samp11	15.01	19.47	17.67
Samp12	3.44	7.99	6.97
Samp21	1.6	2.23	6.66
Samp53	4.75	5.67	14.37
Samp61	1.27	4.20	17.24
<b>Average</b>	<b>5.21</b>	<b>7.91</b>	<b>12.58</b>

**Table 3.** Type-I error on ISPRS dataset.

Sample	FCN-DK6	CNN	LAStools
Samp11	14.09	27.10	26.94
Samp12	2.52	13.92	12.87
Samp21	0.24	1.63	7.98
Samp53	3.92	4.44	14.84
Samp61	0.61	3.95	17.85
<b>Average</b>	<b>4.28</b>	<b>10.21</b>	<b>16.10</b>

**Table 4.** Type-II error on ISPRS dataset.

Sample	FCN-DK6	CNN	LAStools
Samp11	16.25	9.20	5.18
Samp12	4.41	1.75	0.77
Samp21	6.53	4.39	1.87
Samp53	24.49	34.79	3.24
Samp61	19.72	11.06	0.40
<b>Average</b>	<b>14.28</b>	<b>12.24</b>	<b>2.29</b>



**Figure 8.** Results on five testing samples of the ISPRS dataset: (a) FCN-DK; (b) Convolutional Neural Network (CNN); (c) LAStools software. Green: correctly labeled ground; Blue: correctly labeled non-ground; Yellow: ground point misclassified as non-ground; Red: non-ground point misclassified as ground.

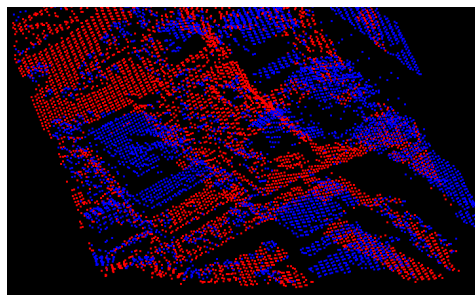
The main advantage of our method compared to LAStools software is the ability to produce more accurate classification and tackle different terrain situations if the network is fed with sufficient training data from various types of terrain. Moreover, LAStools as a rule-based classifier requires

different parameter settings for different types of terrain. In addition, our strategy is significantly more efficient regarding computational cost than state-of-the-art CNN-based techniques [11]. A limitation of our strategy is the need for a point-to-image and image-to-point conversion. Recent computer vision methods can process and classify each point directly [12–14] in the 3D space. Nevertheless, this comes at the expense of an increased computational complexity.

### Qualitative Analysis

Samp11 is steep terrain. When it is combined with low vegetation, the situation makes this sample become trickier for ground classification, as shown in Sithole and Vosselman [7], where two steep slope terrains generate the largest total error. Many filtering algorithms rely on the assumption that the ground points are always lower than the surrounding non-ground points. In Samp11, the assumption is no longer valid, since many low vegetation points are lower than the uphill ground points. The result from the CNN shows many misclassified ground into non-ground classes (type-I errors, which are denoted by striking yellow points in Figure 8), whereas our FCN approach generates less errors on the sloped area. The explanation of this could be that our point-to-image conversion preserves the original information, such as the elevation, and lets the network learn to discriminate ground pixels from non-ground pixels. Meanwhile, the approach using CNN as done in Hu and Yuan [11] only uses the extracted information (the relative elevation of each point with respect to the neighbors) and the feature images extracted from the ground and non-ground points look similar. The result from LAStools also suffered from misclassified ground points on the sloped area, but the misclassification is less striking.

However, the FCN approach resulted in more misclassified non-ground points (type-II error, indicated by the red points in Figure 8), especially in the houses on the downhill part of the area. The houses are located on a sloped terrain, which results in some roofs being coplanar to the neighboring ground and creating a step-like pattern, as seen in Figure 9. This is one of the difficult situations for DTM extraction [9]. Since the roofs have a similar elevation to the surrounding ground, the FCN cannot distinguish the roofs from the ground perfectly.



**Figure 9.** A step-wise pattern on the houses in Samp11. Red: ground; blue: non-ground.

In Samp12, almost all of the points on the building have been correctly labeled as non-ground classes either by FCN or CNN. This is explained by the clear shape of the building and the flat terrain, on which the ground points are located lower than the non-ground points. The result on a bridge in Samp21 is interesting for evaluation, especially regarding the area where the ground surface is connected to the elevated bridge. The CNN produced a noticeable number of misclassified ground points. On the other hand, the FCN generated misclassified non-ground points, but less than the misclassified ground points from LAStools. The result in that particular area is understandable, because the boundary between the ground and the bridge is fuzzy due to the gradual inclination of the road surface.

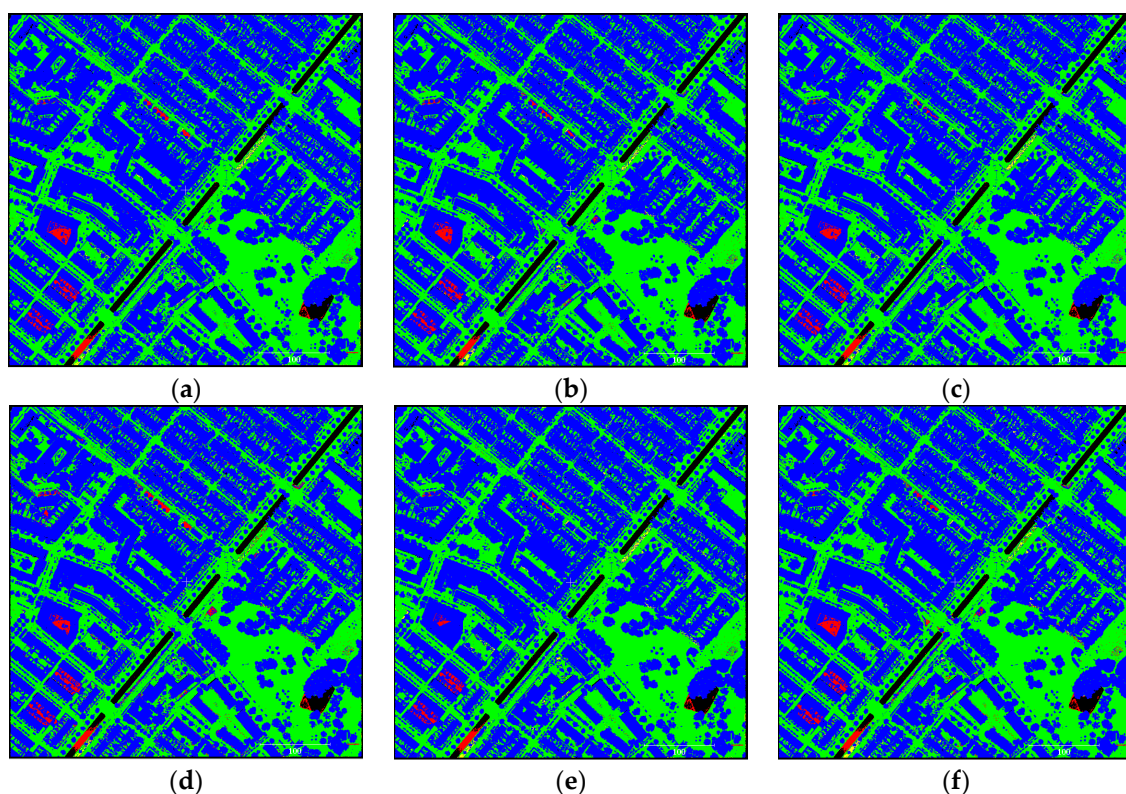
The result on Samp53 proved that the ground points on the break-lines terrain are easily misclassified as non-ground points, since the ground point has a significant height difference over a short distance to the neighboring points on the lower-level terrain. The use of CNN and LAStools

resulted in some misclassified points along this line, while the result from FCN looks more accurate. Similar results were also obtained on the embankment object in Samp61, where more ground points were misclassified as non-ground points in the results from CNN and LAStools, while FCN only produced a few misclassified non-ground points.

Although the visual inspection on Samp53 and Samp61 in Figure 8 shows that the misclassified points of FCN are less noticeable, the type-II error rate on both samples is significantly high. It can be explained by the number of non-ground points in both samples being considerably less than the number of ground points. Samp53 only has 1388 non-ground points in contrast to 32,989 ground points. A similar ratio is also evident on Samp61, where there are only 1247 ground points in contrast to 34,563 ground points. Hence, the type-II error is easily increased by only a few additional misclassified non-ground points.

#### 4.2.2. Ground Classification on AHN3 Dataset

Our method was also tested on an AHN3 dataset to investigate the performance over a larger area. Due to the high point density, we were able to examine the use of smaller pixel size ( $0.5 \times 0.5$  m) and a multi-scale image as an input image for the FCN. We also used our MTN-FCN architecture, but only considered the first output of the network for ground classification. In addition, LAStools software was used as a baseline. Figure 10 shows the results.



**Figure 10.** Ground classification results on testing sample number 7 (TS7) of the Actueel Hoogtebestand Nederland 3 (AHN3) dataset: (a) FCN 1 m; (b) FCN 0.5 m; (c) MS-FCN Down; (d) MS-FCN Down-Up; (e) MTN-FCN; (f) LAStools. Green: correctly labeled ground; Blue: correctly labeled non-ground; Yellow: ground point misclassified as non-ground; Red: non-ground point misclassified as ground.

Again, the quality of the ground classification is indicated by the total errors, type-I errors, and type-II errors. Tables 5–7 show the total errors, type-I errors, and type-II errors, respectively.



**Table 5.** Total errors of the AHN3 dataset.

Sample	FCN 1 m	FCN 0.5 m	MS-FCN Down	MS-FCN Down-Up	MTN-FCN	LAStools
TS 1	4.21	5.77	4.00	4.80	3.78	5.08
TS 2	2.48	3.07	3.02	3.78	2.60	3.44
TS 3	1.94	1.94	1.96	2.93	1.87	2.32
TS 4	2.52	2.08	1.96	3.06	1.96	2.46
TS 5	2.47	5.06	3.69	3.55	4.15	6.23
TS 6	2.59	2.24	2.47	3.13	2.44	4.08
TS 7	1.82	1.79	1.71	2.81	1.71	2.05
TS 8	2.24	2.35	2.40	3.25	2.20	2.86
TS 9	4.48	3.13	2.75	2.93	2.71	3.38
TS 10	3.12	3.79	2.91	3.91	2.97	3.23
<b>Average</b>	<b>2.79</b>	<b>3.12</b>	<b>2.69</b>	<b>3.42</b>	<b>2.64</b>	<b>3.51</b>

**Table 6.** Type-I errors of the AHN3 dataset.

Sample	FCN 1 m	FCN 0.5 m	MS-FCN Down	MS-FCN Down-Up	MTN-FCN	LAStools
TS 1	2.09	3.77	2.63	0.71	2.17	1.70
TS 2	1.49	1.73	2.72	1.69	1.65	1.98
TS 3	0.89	0.38	0.88	0.98	0.86	1.27
TS 4	2.48	0.58	1.42	1.07	1.38	1.03
TS 5	2.15	6.02	4.50	2.70	5.34	6.04
TS 6	1.26	0.36	1.27	0.53	1.23	2.61
TS 7	1.01	0.51	1.03	1.00	1.03	1.72
TS 8	1.42	0.86	1.28	0.99	1.35	2.01
TS 9	5.40	2.55	1.68	0.90	1.68	1.41
TS 10	3.11	1.98	1.84	1.47	2.13	1.55
<b>Average</b>	<b>2.13</b>	<b>1.87</b>	<b>1.93</b>	<b>1.20</b>	<b>1.88</b>	<b>2.13</b>

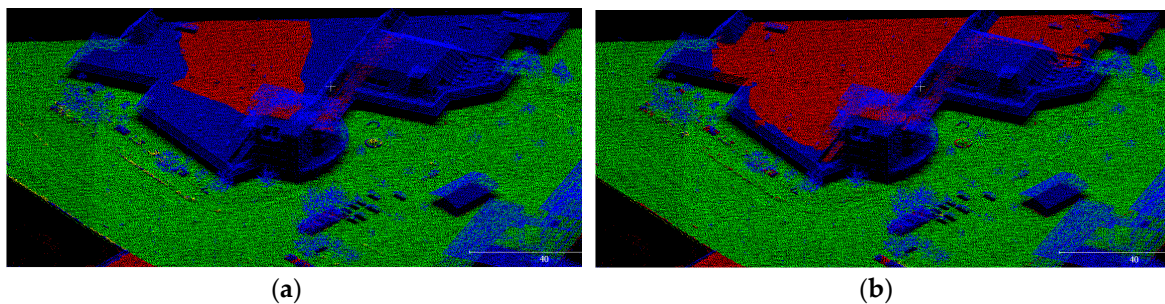
**Table 7.** Type-II errors of the AHN3 dataset.

Sample	FCN 1 m	FCN 0.5 m	MS-FCN Down	MS-FCN Down-Up	MTN-FCN	LAStools
TS 1	6.61	8.03	5.55	9.44	5.61	8.91
TS 2	3.51	4.46	3.32	5.94	3.57	4.94
TS 3	2.83	3.28	2.89	4.59	2.74	3.21
TS 4	2.54	3.24	2.38	4.59	2.40	3.56
TS 5	2.96	3.60	2.46	4.82	2.36	6.54
TS 6	4.06	4.34	3.81	6.02	3.79	5.71
TS 7	2.44	2.78	2.23	4.20	2.23	2.31
TS 8	2.81	3.38	3.18	4.83	2.79	3.46
TS 9	3.56	3.71	3.83	4.97	3.75	5.36
TS 10	3.12	6.20	4.35	7.19	4.09	5.47
<b>Average</b>	<b>3.44</b>	<b>4.30</b>	<b>3.40</b>	<b>5.66</b>	<b>3.33</b>	<b>4.95</b>

It can be seen that a smaller pixel size resulted in a lower amount of type-I error, while a larger pixel size minimized the total number of type-II errors. It can be concluded that the smaller pixel size maintains the ground surface, while the larger pixel size is better when eliminating non-ground objects. The results also show that utilizing the multi-scale resolution slightly improves the accuracy, but only if the up-sampling layer is not employed. It seems that the up-sampling layer cannot interpolate the output feature maps into a more detailed resolution perfectly. Finally, the ground classification result from the MTN-FCN performs better than the others. Employing not only low-point images, but also high-point images to train the network, proved to achieve a better accuracy.

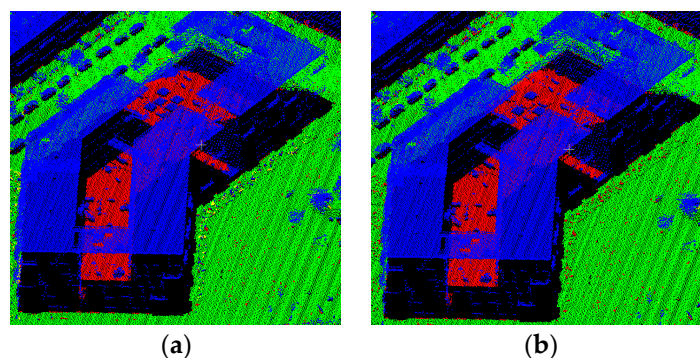
### Qualitative Analysis

Despite the error rates on the 10 testing sets of the AHN3 dataset being considerably low, an evaluation of the error results is needed to investigate the situations in which the algorithm did not perform well. The misclassified points produced by the FCN on the AHN3 dataset were caused by several reasons. The first reason is the large building. As seen in Figure 11, most of the points on the building roof were not classified correctly. The reason was clear, since the size of the building is more than 100 m, which is larger than the receptive field size of the FCN. In other words, the FCN cannot ‘see’ the building along with the surrounding scene. When the FCN only looked at the center of the building roof, it had a flat surface and a similar elevation compared to its surroundings. Hence, it was misclassified as ground. However, a similar result is also derived using LAStools. In the LAStools software, one can change the parameters to obtain a better result, but it needs prior knowledge of the scene and an experienced operator. It may not be a problem in a hierarchical approach or segment-based filtering.



**Figure 11.** Error result on a very large building: (a) FCN; (b) LAStools. Red points are those incorrectly labeled as ground points.

Another building that contributed errors is a large building with a different roof level (Figure 12). Probably, this erroneous result appears due to the receptive field not being large enough to cover the whole building. The building has high roofs on the edge, but a lower roof in the middle. In the extracted image from the point cloud, the roof in the middle has a lower height than the surrounding roofs. Hence, the FCN failed to classify that roof as a non-ground class. The similar result was obtained using LAStools. Thus, it can be concluded that this specific building presents difficulties within a ground classification task.

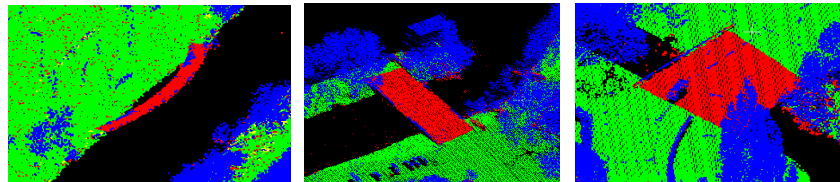


**Figure 12.** Error result on a large building with a lower roof in the middle: (a) FCN; (b) LAStools. Red points are those incorrectly labeled as ground points.

Errors are not only caused by the geometry of the building; they are also caused by the points inside the building. Due to the high point density of the LIDAR dataset, some laser pulses can penetrate inside the building through windows on roofs or facades. Since these points may lie on a ground floor,

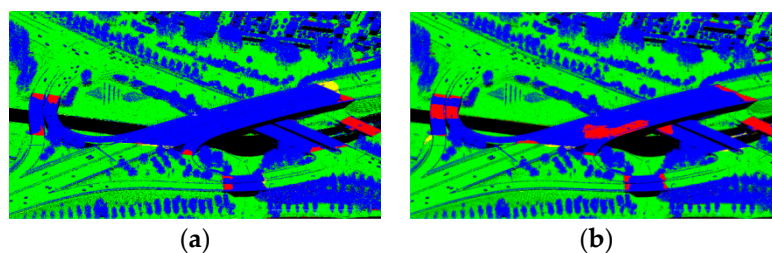
their elevation would be registered as similar to the ground surface outside the building; therefore, those points are labeled as ground as well, which could be considered a correct result.

Another error source is bridges (Figure 13). Although the bridge in Samp21 of the ISPRS dataset can be classified almost perfectly using FCN, many bridges in the AHN3 dataset failed to be labeled within the non-ground class. The reason is that the bridge in Samp21 of the ISPRS dataset is an elevated bridge, while most of the bridges in the AHN3 dataset are flat bridges that are at the level of the ground surface. Since those bridges are coplanar to the ground surface, neither the proposed method nor the LAStools software can label the points on the bridge as a non-ground class.



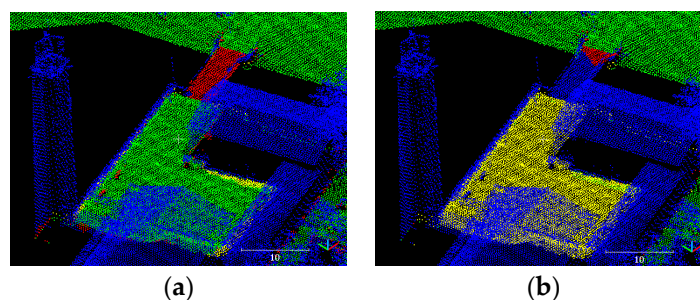
**Figure 13.** Errors on flat bridges. Red points are those incorrectly labeled as ground points.

In order to challenge the algorithms, a quite complex structure composed of a bridge and a road was introduced into the testing set, as shown in Figure 14. It can be seen that the FCN resulted in a better classification than LAStools, where some points in the middle of the bridge were mislabeled as ground points. Nevertheless, both algorithms failed when labeling the transition area between the road and the bridge. In this specific case, the reference data even could not be very accurate, since the exact location of the boundary between the road and the bridge was arbitrary.



**Figure 14.** Result on a mixed structure of road and bridge: (a) FCN; (b) LAStools. Red points are those incorrectly labeled as ground points. Yellow points are those incorrectly labeled as non-ground points.

An interesting result is shown in Figure 15, where LAStools failed to classify the relatively flat ground surface in the middle of the pond, while the FCN successfully labeled almost all of the ground points. In this sample, the parameter setting in LAStools was set to “town or flats” due to the scene being in a relatively flat area. Again, the parameters in the LAStools software could be re-adjusted in order to obtain a better result, but it comes with a need of prior knowledge of the area.

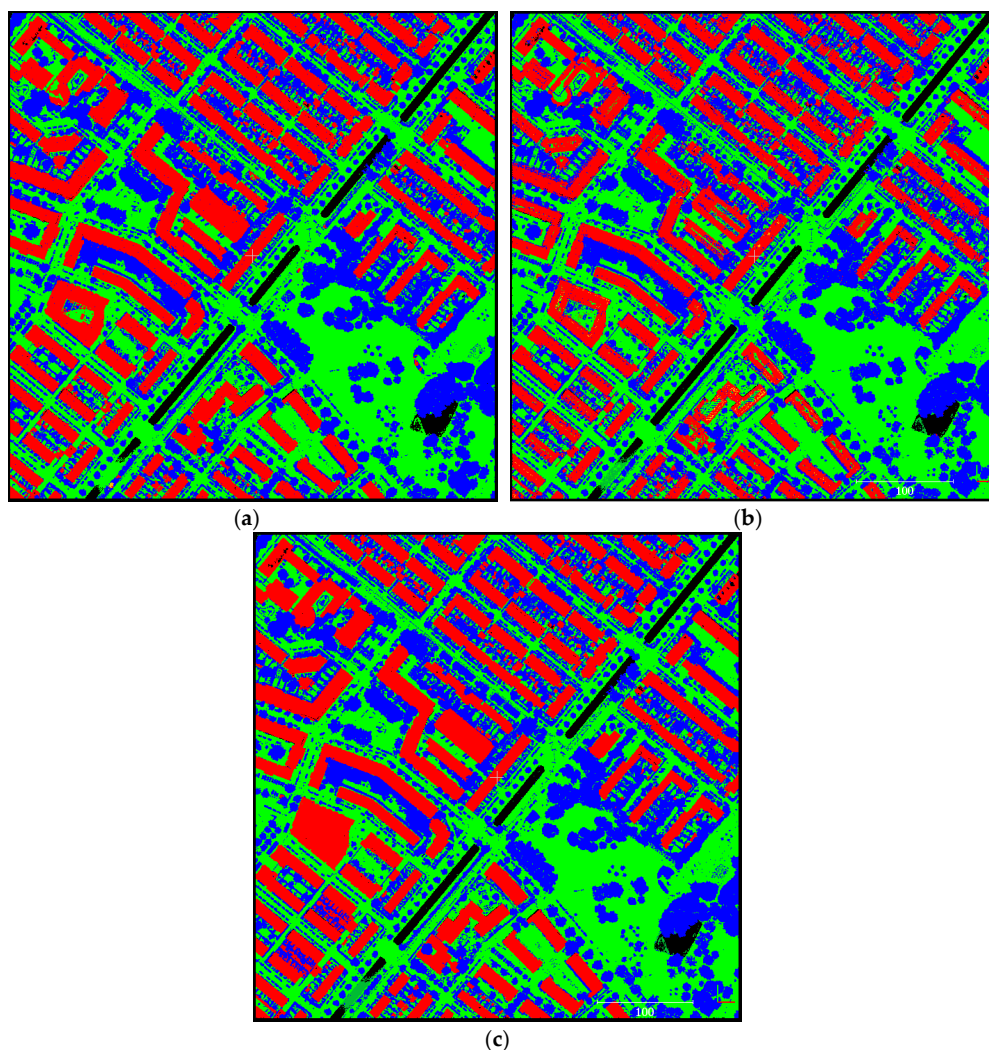


**Figure 15.** A situation when LAStools failed to detect a relatively flat ground surface: (a) FCN; (b) LAStools.

Meanwhile, both algorithms failed when classifying points on the water, as the water is part of the ground surface. In that case, it is difficult to label the points on the water as a non-ground class.

#### 4.2.3. Multi-Class Classification on AHN3 Dataset

Our experiment used the AHN3 dataset to further refine the non-ground class into buildings and vegetation. We also added bridges in the experiments. However, as the number of points on the bridges is not sufficient to train the network properly, we neglected this class in the accuracy assessment. Figure 16 shows our results using the MTN-FCN compared to the results from the Random Forest (RF) classifier. The RF classifier uses 10 features, including local height differences, echo information, linearity, planarity, scattering, omnivariance, anisotropy, and change of curvature [24,26]. The number of trees is set to 100, the minimum sample leaf is set to one, and the maximum number of features is set to four.



**Figure 16.** Multi-class classification results on sample TS7 of the AHN3 dataset: (a) MTN-FCN; (b) Random Forest (RF); (c) Reference. Green: ground; blue: vegetation; red: building; dark green: water.

Precision and recall values were used for quantitative accuracy assessment. Precision is calculated as in Equation (3), while recall is calculated as in Equation (4). Tables 8 and 9 show the precision and recall values on the 10 testing sets of the AHN3 dataset.

$$\text{Precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \quad (3)$$

$$\text{Recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \quad (4)$$

**Table 8.** Precision values of the AHN3 dataset. TS: testing set.

Class	Method	TS 1	TS 2	TS 3	TS 4	TS 5	TS 6	TS 7	TS 8	TS 9	TS 10	Average
Vegetation	MTN-FCN	0.93	0.91	0.96	0.96	0.74	0.89	0.96	0.97	0.96	0.95	0.92
	RF	0.87	0.93	0.92	0.94	0.87	0.89	0.92	0.97	0.96	0.94	0.92
Ground	MTN-FCN	0.95	0.97	0.97	0.97	0.98	0.97	0.97	0.96	0.96	0.97	0.97
	RF	0.95	0.96	0.96	0.95	0.97	0.97	0.95	0.93	0.93	0.95	0.95
Building	MTN-FCN	0.95	0.96	0.96	0.95	0.97	0.98	0.96	0.94	0.94	0.48	0.91
	RF	0.94	0.91	0.97	0.93	0.89	0.97	0.95	0.92	0.63	0.32	0.85
Average Precision	MTN-FCN	0.94	0.95	0.96	0.96	0.90	0.95	0.96	0.96	0.95	0.80	<b>0.93</b>
	RF	0.92	0.93	0.95	0.94	0.91	0.94	0.95	0.94	0.84	0.74	<b>0.91</b>

**Table 9.** Recall values of the AHN3 dataset. MTN: Multi-Task Network, RF: Random Forest.

Class	Method	TS 1	TS 2	TS 3	TS 4	TS 5	TS 6	TS 7	TS 8	TS 9	TS 10	Average
Vegetation	MTN-FCN	0.87	0.95	0.95	0.95	0.95	0.95	0.96	0.97	0.98	0.95	0.95
	RF	0.97	0.91	0.95	0.94	0.89	0.93	0.95	0.95	0.95	0.92	0.94
Ground	MTN-FCN	0.98	0.98	0.99	0.97	0.95	0.99	0.99	0.99	0.98	0.98	0.98
	RF	0.97	0.98	0.99	0.98	0.96	0.99	0.99	0.99	0.92	0.89	0.97
Building	MTN-FCN	0.96	0.86	0.94	0.95	0.77	0.87	0.94	0.86	0.93	0.92	0.90
	RF	0.89	0.88	0.87	0.88	0.87	0.89	0.84	0.80	0.78	0.87	0.86
Average Recall	MTN-FCN	0.94	0.93	0.96	0.96	0.89	0.94	0.96	0.94	0.96	0.95	<b>0.94</b>
	RF	0.94	0.92	0.94	0.93	0.91	0.94	0.93	0.91	0.88	0.89	<b>0.92</b>

It can be seen that the precision and recall values of FCN are higher than RF. The average values for the FCN were 0.93 and 0.94 for both precision and recall respectively, while RF resulted in a slightly lower value of 0.91 for the precision and 0.92 for the recall.

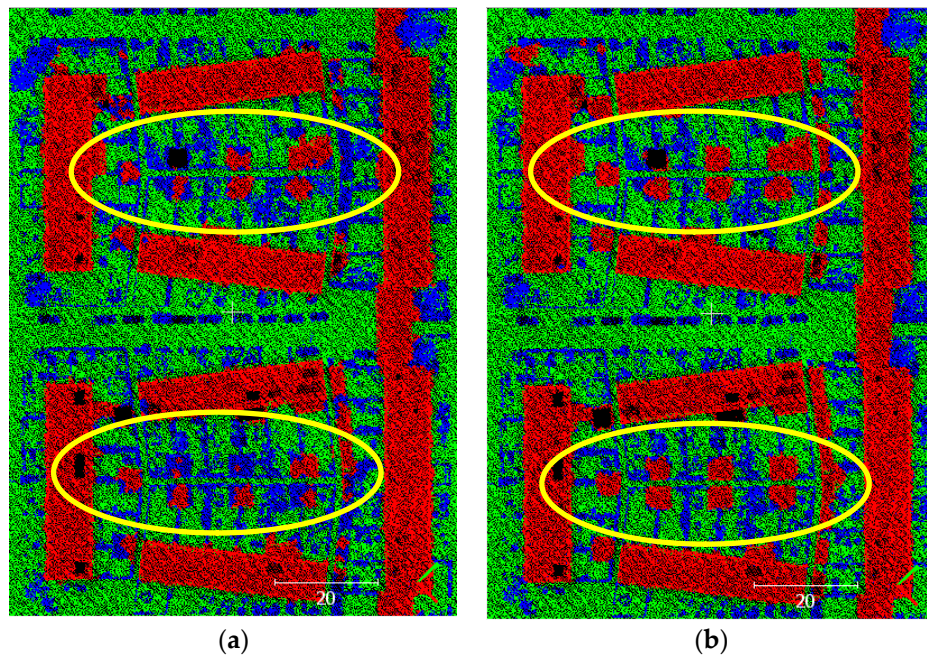
### Qualitative Analysis

In our experiment, the use of a multi-scale network in the MTN-FCN compared to a single pixel (1-m) MTN-FCN only increased the precision and recall values by 0.0043 and 0.0147, respectively. However, in a visual inspection, the use of a multi-scale network improved the result by a lot, especially where small building roofs were surrounded by vegetation. In the single-pixel size MTN-FCN, many points on the small building roof were labeled as vegetation, while those points were labeled correctly in a multi-scale MTN-FCN. Figure 17 shows the differences between the single pixel MTN-FCN and the multi-scale MTN-FCN.

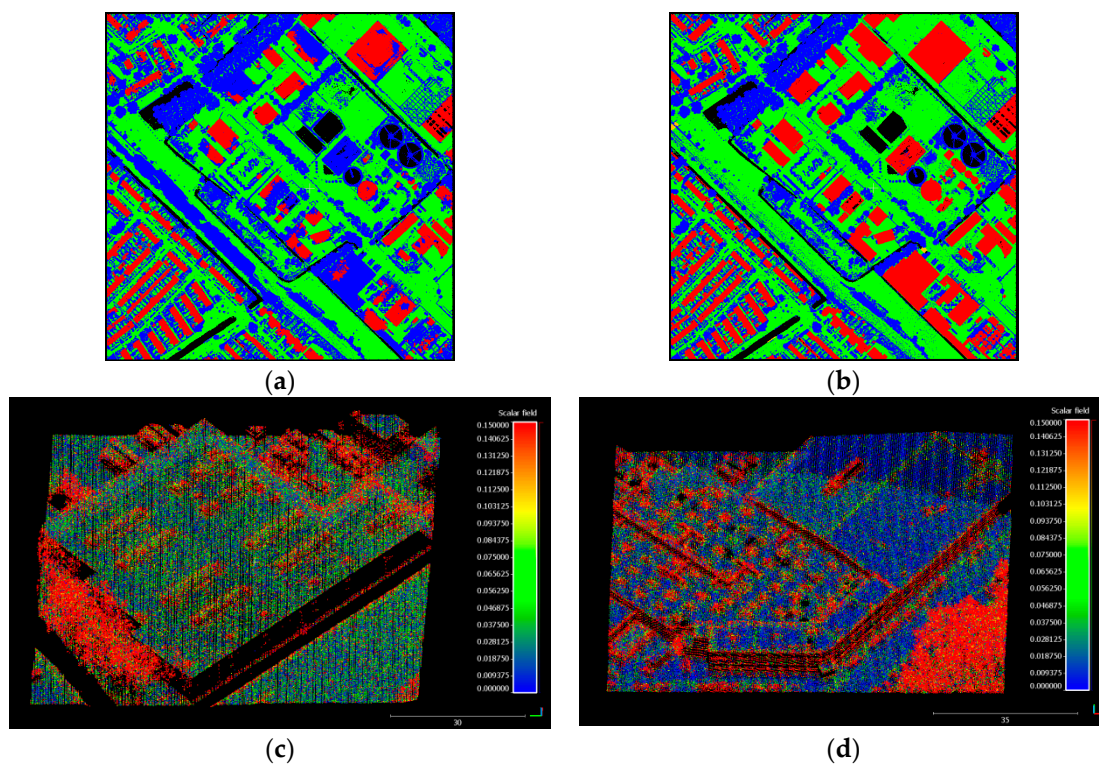
Although the metric accuracies of the FCN and RF were quite comparable, it is interesting to examine the performance of each method qualitatively. Even though the FCN resulted in a better accuracy, it has several drawbacks. As mentioned in Section 3.3, the labels are transferred from pixels to points. The drawback is apparent if there is a situation where vegetation and building are mixed in one pixel. Then, all of the points are labeled within the same class.

Another erroneous result was in the sample TS5. The FCN failed when classifying points on relatively large buildings; instead, those points were labeled as vegetation. It is not very clear in the beginning why the FCN labeled those points as vegetation, but a further investigation on the point cloud revealed that the point cloud on this sample is a bit noisy compared to the other samples. Figure 18c shows the noise of each point on one building from sample TS5 and another building from the training set. The noise is calculated by the distance of the corresponding point and the local plane.

In a flat roof, if the point cloud is very precise, then the noise should be small enough: typically less than 5 cm. However, it turned out that the noise in sample TS5 was relatively larger (up to 10 cm) than the noise in the building from the training set (Figure 18d). This could come from the overlapped area of two strips, which accumulated the noise of the two strips. It is likely that the noise was the reason that the flat roof was misclassified into the vegetation class.

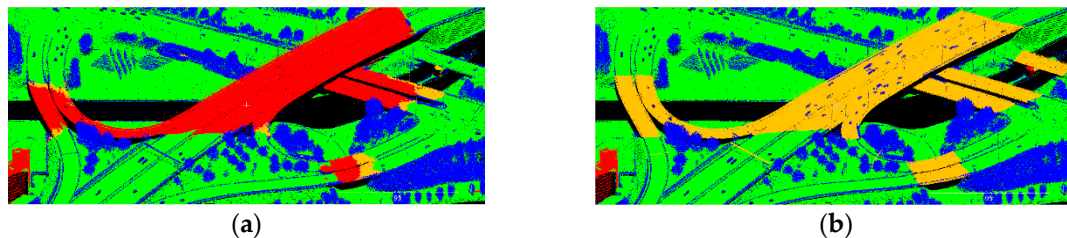


**Figure 17.** Difference between: (a) single pixel (1-m) MTN-FCN; (b): multi-scale MTN-FCN. Note the shape of the small building roofs.



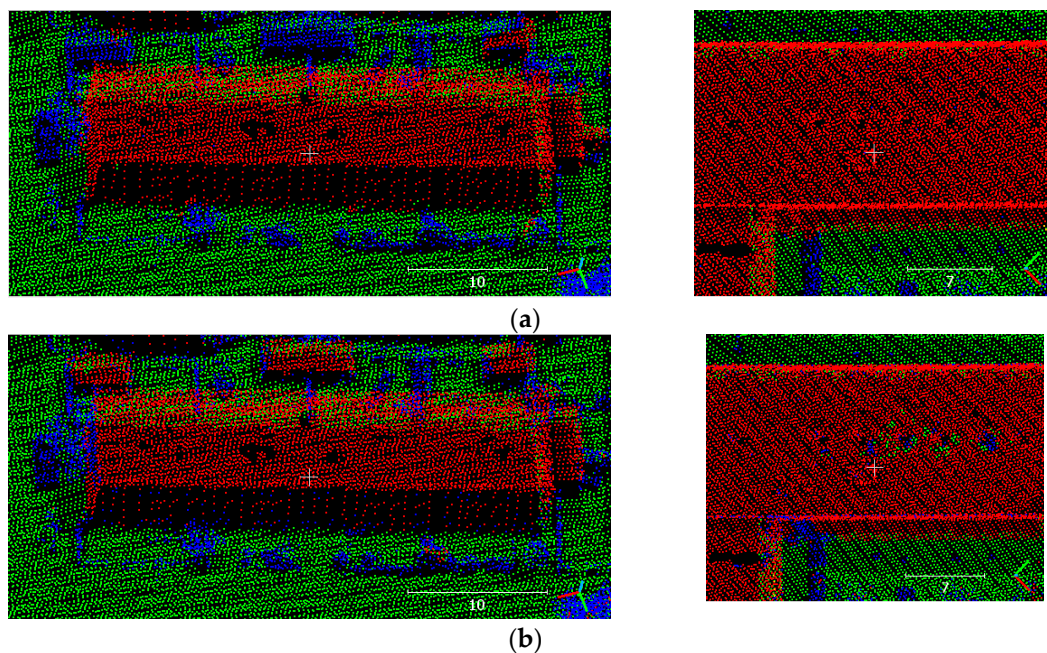
**Figure 18.** Error on FCN result due to noise in sample TS5: (a) prediction; (b) reference; (c) noise in one building in TS5; (d) noise in one building in the training set. Note that noise in (c) is higher than in (d).

In a ground classification, points on an elevated bridge can be labeled correctly as non-ground. However, in a multi-class classification, most of those points are labeled as a building. Only a few parts of the bridge could be labeled correctly. This is probably due to the lack of an elevated bridge sample in the training set. Since the bridge is an elevated object, then it could be easily mislabeled as a building. Figure 19 shows the result on that particular area.



**Figure 19.** Elevated bridge: (a) FCN; (b) reference. Green: ground; blue: vegetation; red: building; yellow: bridge.

The FCN performed better than RF in the area shown in Figure 20. In the RF result, some points on the building facades are labeled incorrectly as vegetation. Some points on the building roofs are also mislabeled as ground, while some points on the vegetation are mislabeled as buildings. Furthermore, an approach using RF needs a careful procedure of handcrafted feature extraction, while our method using a FCN skips the feature extraction and directly feeds the data into the network.



**Figure 20.** Comparison results: (a) FCN; (b) RF.

## 5. Discussion

The proposed method is designed to improve the computational efficiency of a state-of-the-art deep learning approach [11]. In the experiment, FCN-based ground classification works significantly faster than the CNN-based ground classification, both in the point-to-image conversion and the testing. We performed the experiments using a computer equipped with an Intel Core i7-6700HQ 2.6GHz, with 16 GB of RAM and Nvidia Quadro M1000M 2GB GPU. Our implementation takes advantage of the Graphics Processing Unit (GPU) processing adopting the MatConvnet deep learning library [36]. The point-to-image conversion is reduced from 47 h to 36 min (on all 15 samples), while testing

time was reduced from 126 s to 7.8 s (on the five testing samples). However, the training is slower. The reason is that our FCN without down-sampling maintains the size of the feature maps in each layer. Hence, the filters will convolve on a larger extent during the forward pass than in CNN architecture. Table 10 shows the comparison with the ISPRS filter test dataset.

**Table 10.** Computational time comparison. Computer specifications: Intel Core i7-6700HQ 2.6GHz, 16 GB RAM, and Nvidia Quadro M1000M 2GB.

Method	Point-to-Image Conversion	Training	Testing
FCN	36 min	12 h	7.8 s
CNN	47 h	2.5 h	126 s

In the architectural design, we keep the six layers of the FCN architecture as proposed by Persello and Stein [15]. The experiments on the ISPRS validation set show that reducing the number of layers affected the accuracy. The accuracy reduces from 86.37% on architecture with six layers into 86.22% and 83.97% on architecture with five and four layers, respectively. The results of the AHN3 validation set show a similar trend. When the number of layers is reduced, the accuracy decreases from 95.46% to 95.31% and 95.09%. Our results are consistent with other research [38] where the higher accuracy could be achieved through a deeper network. In a deeper configuration, the network has more features to capture.

We removed the max-pooling layers, resulting in a higher accuracy and a simpler network. Similar results were obtained in Springenberg et al. [39] when removing the pooling layer while maintaining the accuracy. In our experiment, removing the max-pooling layer increased the accuracy from 85.44% to 88.54% on ISPRS validation set. The results on the AHN3 validation set showed a different behavior. It turns out that the accuracy decreased slightly from 95.35% to 95.09% when the max-pooling layer was removed. However, we prefer to remove the max-pooling layer to keep the network simple; therefore, the computational time is faster.

In a LIDAR point cloud, it is common to have outlier points. Those points usually have an extreme height, which could be either very high or very low. Such outliers affect our method during the point-to-image conversion. Removing the outliers before the data conversion increases the accuracy result by 2.62%.

## 6. Conclusions

This paper introduces an FCN-based approach to classify ALS point clouds into ground, building, and vegetation. The proposed method is more efficient than state-of-the-art ones, and is therefore suitable for real-world applications to process ALS point clouds at a national or regional level. Experimental results show that our method has the potential for large-scale applications.

We show that the FCN-based classification resulted in a higher accuracy yet faster testing time than CNN-based classification in an ISPRS filter test dataset. It runs faster due to the more efficient point-to-image conversion. The accuracy increases, since we keep the original information from the LIDAR point cloud instead of only using the relative difference in height, so that the network could learn the features from the original information. However, our conversion also leads to a drawback. While CNN-based classification is able to give a label for each point, FCN is only able to do the labeling for each pixel. FCN also performs better than LAsTools software in the AHN3 dataset.

We extend our network for multi-class classification task by stacking another FCN block to build a multi-task network so that it could predict the second label for each pixel. In the experiments, our method resulted in a precision value of 0.93 and recall value of 0.94, which was slightly higher than the results from RF (0.91 and 0.92). We have shown that our multi-scale resolution approach, e.g., using both 50-cm and 1-m pixel sizes, improves the detection of small buildings. This advantage is crucial for automatic building extraction. Despite the point-to-image conversion being more efficient and leading



to better accuracies than those of the baselines, our method has some drawbacks. Firstly, it still needs a point-to-image conversion in order to process the point cloud. Secondly, the classification is done at the pixel level. In other words, the labels are given to a pixel instead of to the point directly. A potential limitation of our method, especially for multi-class classification problems, is in the situation when more than one point from different non-ground classes is in the same pixel, such as for instance a tree branch above a building roof. In that case, only one class label is assigned to all of the non-ground points within that pixel.

In the future, several strategies can be investigated to improve the classification of ALS point clouds. In the context of FCN-based strategies, a better point-to-image conversion can be explored, including additional features. As an alternative to point-to-image conversion, voxelization is a potential promising direction. Unlike pixel-based analyses, voxels retain their 3D structure, thus minimizing the loss of information during the conversion. Using voxels for the conversion would thus resolve the limitation of assigning the same class to all of the points within a pixel (as described above). However, processing with voxels would also increase the computational requirements of the algorithm.

**Author Contributions:** A.R. adopted the FCN code developed by C.P. A.R. set-up and performed all experiments, under the supervision of S.O.E., C.P. and C.G. A.R. is the main author of the manuscript, but all co-authors supported the development of the paper and revised it over several rounds.

**Funding:** This research received no external funding.

**Acknowledgments:** The research is supported by LPDP scholarship from Ministry of Finance, Indonesia and ITC Excellence Program scholarship from Faculty of Geo-Information Science and Earth Observation, University of Twente.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Briese, C. Extraction of Digital Terrain Models. In *Airborne and Terrestrial Laser Scanning*; Vosselman, G., Maas, H.-G., Eds.; Whittles Publishing: Dunbeath, UK, 2010; pp. 135–167. ISBN 978-1-4398-2798-7.
2. Niemeyer, J.; Rottensteiner, F.; Soergel, U. Conditional random fields for LiDAR point cloud classification in complex urban areas. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2012**, *I-3*, 263–268. [[CrossRef](#)]
3. Niemeyer, J.; Rottensteiner, F.; Soergel, U. Classification of urban LiDAR data using conditional random field and random forests. In Proceedings of the Joint Urban Remote Sensing Event (JURSE) 2013, Sao Paulo, Brazil, 21–23 April 2013; pp. 139–142. [[CrossRef](#)]
4. Niemeyer, J.; Rottensteiner, F.; Soergel, U. Contextual classification of lidar data and building object detection in urban areas. *ISPRS J. Photogramm. Remote Sens.* **2014**, *87*, 152–165. [[CrossRef](#)]
5. Yang, Z.; Jiang, W.; Xu, B.; Zhu, Q.; Jiang, S.; Huang, W. A Convolutional Neural Network-Based 3D Semantic Labeling Method for ALS Point Clouds. *Remote Sens.* **2017**, *9*, 936. [[CrossRef](#)]
6. Vosselman, G.; Coenen, M.; Rottensteiner, F. Contextual segment-based classification of airborne laser scanner data. *ISPRS J. Photogramm. Remote Sens.* **2017**, *128*, 354–371. [[CrossRef](#)]
7. Sithole, G.; Vosselman, G. Experimental comparison of filter algorithms for bare-Earth extraction from airborne laser scanning point clouds. *ISPRS J. Photogramm. Remote Sens.* **2004**, *59*, 85–101. [[CrossRef](#)]
8. Pfeifer, N.; Mandlburger, G. LiDAR Data Filtering and DTM Generation. In *Topographic Laser Ranging and Scanning: Principles and Processing*; Shan, J., Toth, C.K., Eds.; CRC Press: Boca Raton, FL, USA, 2009; pp. 307–333. ISBN 978-1-4200-5142-1.
9. Gevaert, C.M.; Persello, C.; Nex, F.; Vosselman, G. A deep learning approach to DTM extraction from imagery using rule-based training labels. *ISPRS J. Photogramm. Remote Sens.* **2018**, *142*, 106–123. [[CrossRef](#)]
10. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [[CrossRef](#)]
11. Hu, X.; Yuan, Y. Deep-learning-based classification for DTM extraction from ALS point cloud. *Remote Sens.* **2016**, *8*, 730. [[CrossRef](#)]

12. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 June 2017.
13. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv* **2017**, arXiv:1706.02413.
14. Su, H.; Jampani, V.; Sun, D.; Maji, S.; Kalogerakis, E.; Yang, M.-H.; Kautz, J. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 19–21 June 2018; pp. 2530–2539.
15. Persello, C.; Stein, A. Deep Fully Convolutional Networks for the Detection of Informal Settlements in VHR Images. *IEEE Geosci. Remote Sens. Lett.* **2017**, *14*, 2325–2329. [[CrossRef](#)]
16. Vosselman, G. Slope based filtering of laser altimetry data. *Int. Arch. Photogramm. Remote Sens.* **2000**, *33*, 678–684. [[CrossRef](#)]
17. Kilian, J.; Haala, N.; Englich, M. Capture and evaluation of airborne laser scanner data. *Int. Arch. Photogramm. Remote Sens.* **1993**, *31*, 383–388.
18. Sithole, G. Filtering of Laser Altimetry Data Using a Slope Adaptive Filter. *Int. Arch. Photogramm. Remote Sens.* **2001**, *34*, 203–210.
19. Axelsson, P. DEM Generation from Laser Scanner Data Using adaptive TIN Models. *Int. Arch. Photogramm. Remote Sens.* **2000**, *33*, 110–117. [[CrossRef](#)]
20. Nie, S.; Wang, C.; Dong, P.; Xi, X.; Luo, S.; Qin, H. A revised progressive TIN densification for filtering airborne LiDAR data. *Measurement* **2017**, *104*, 70–77. [[CrossRef](#)]
21. Kraus, K.; Pfeifer, N. Determination of terrain models in wooded areas with airborne laser scanner data. *ISPRS J. Photogramm. Remote Sens.* **1998**, *53*, 193–203. [[CrossRef](#)]
22. Pfeifer, N.; Stadler, P.; Briese, C. Derivation Of Digital Terrain Models In The Scop++ Environment. In Proceedings of the OEEPE Workshop on Airborne Laserscanning and Interferometric SAR for Digital Elevation Models, Stockholm, Sweden, 1–3 March 2001.
23. Sithole, G.; Vosselman, G. Filtering of airborne laser scanner data based on segmented point clouds. In Proceedings of the ISPRS Workshop Laser Scanning 2005, Enschede, The Netherlands, 12–14 September 2005; pp. 66–71.
24. Chehata, N.; Guo, L.; Mallet, C. Airborne Lidar feature Selection for urban classification using Random Forests. In Proceedings of the ISPRS Workshop Laser Scanning 2009, Paris, France, 1–2 September 2009; pp. 207–212.
25. Zhang, J.; Lin, X.; Ning, X. SVM-Based classification of segmented airborne LiDAR point clouds in urban areas. *Remote Sens.* **2013**, *5*, 3749–3775. [[CrossRef](#)]
26. Weinmann, M.; Jutzi, B.; Hinz, S.; Mallet, C. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS J. Photogramm. Remote Sens.* **2015**, *105*, 286–304. [[CrossRef](#)]
27. Hackel, T.; Savinov, N.; Ladicky, L.; Wegner, J.D.; Schindler, K.; Pollefeys, M. Semantic3D.net: A New Large-Scale Point Cloud Classification. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2017**, *IV*, 91–98. [[CrossRef](#)]
28. Rizaldy, A.; Persello, C.; Gevaert, C.M.; Elberink, S.J.O. Fully Convolutional Networks for Ground Classification from LIDAR Point Clouds. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2018**, *IV*, 231–238. [[CrossRef](#)]
29. Bergado, J.; Persello, C.; Stein, A. Recurrent Multiresolution Convolutional Networks for VHR Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 6361–6374. [[CrossRef](#)]
30. Long, J.; Shelhamer, E.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 640–651. [[CrossRef](#)]
31. Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv* **2015**, arXiv:1511.00561.
32. Yu, F.; Koltun, V. Multi-Scale Context Aggregation by Dilated Convolutions. *arXiv* **2015**, arXiv:1511.07122.
33. Ko, C.; Kang, J.; Sohn, G. Deep Multi-task Learning for Tree Genera Classification. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2018**, *IV*, 153–159. [[CrossRef](#)]
34. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.

35. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [[CrossRef](#)]
36. Vedaldi, A.; Lenc, K. MatConvNet—Convolutional Neural Networks for MATLAB. In Proceedings of the 23rd ACM International Conference on Multimedia, Brisbane, Australia, 26–30 October 2015.
37. Reutebuch, S.E.; Mcgaughey, R.J.; Andersen, H.; Carson, W.W. Accuracy of a high-resolution lidar terrain model under a conifer forest canopy. *Can. J. Remote Sens.* **2003**, *29*, 527–535. [[CrossRef](#)]
38. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015; pp. 1–14.
39. Springenberg, J.T.; Dosovitskiy, A.; Brox, T.; Riedmiller, M. Striving for Simplicity: The All Convolutional Net. *arXiv* **2014** arXiv:1412.6806.



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).