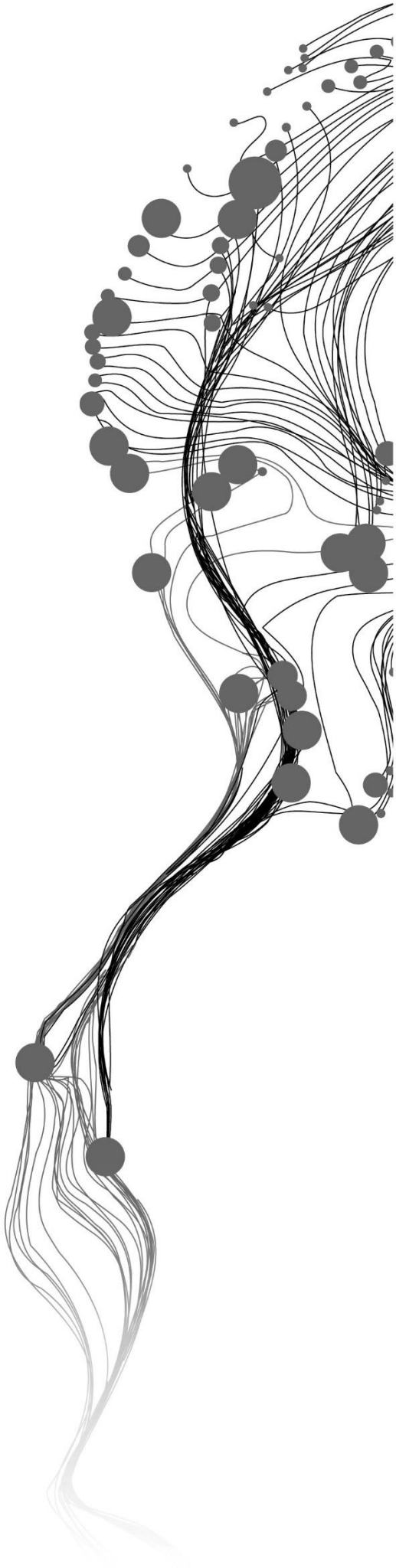
DEEP LEARNING-BASED DTM EXTRACTION FROM LIDAR POINT CLOUD

ALDINO RIZALDY
February, 2018

SUPERVISORS:
dr.ir. S.J. Oude Elberink
dr. C. Persello

ADVISOR:
C.M. Gevaert, MSc



DEEP LEARNING-BASED DTM EXTRACTION FROM LIDAR POINT CLOUD

ALDINO RIZALDY
Enschede, The Netherlands, February, 2018

Thesis submitted to the Faculty of Geo-Information Science and Earth Observation of the University of Twente in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation.
Specialization: Geoinformatics

SUPERVISORS:
dr.ir. S.J. Oude Elberink
dr. C. Persello

ADVISOR:
C.M. Gevaert, MSc

THESIS ASSESSMENT BOARD:
prof.dr.ir. M.G. Vosselman (chair)
dr. R.C. Lindenbergh (External Examiner, Delft University of Technology, Optical and Laser Remote Sensing)

ABSTRACT

In a recent study, a Convolutional Neural Network (CNN) has been used for DTM extraction; following the popularity of deep learning for various classification tasks. Since CNNs are designed to work with images, point-to-image conversion is mandatory in order to process point clouds by CNN. Even though the error rates of the result using CNN are lower than any other methods, the method has a drawback. The point-to-image conversion is slow because each point is converted into a separate image thus leads to highly redundant computation. The objective of this study is to design a more efficient deep learning-based DTM extraction. The goal is achieved by converting the whole point cloud into a single image. The classification itself is performed by employing Fully Convolutional Network (FCN), a modified version of CNN which is specially designed for pixel-wise semantic classification. In the experiment, the proposed method was significantly faster than CNN as the state-of-the-art method. It is 78 times faster for point-to-image conversion and 16 times faster for the testing time. An alternative method was also proposed by extracting features manually and training a Multi-Layer Perceptron (MLP) classifier. Random Forest (RF) was also used as a comparison classifier. The experiment using the ISPRS Filter Test dataset shows that FCN results in 5.22% of total error, 4.10% of type I error, and 15.07% of type II error. It has lower total error and type I error than MLP, CNN, RF and LAStools software. Meanwhile, the alternative method using MLP led to worse accuracies than FCN or CNN. The FCN approach was also tested on AHN dataset, a very high point density LIDAR point cloud, resulting in 3.63% of total error, 0.93% of type I error and 6.03% of type II error. Those error rates are almost similar to the result from LAStools software which has 3.33% of total error, 1.50% of type I error and 5.16 of type II error. Furthermore, the FCN method was extended in order to separate non-ground points into vegetation and building on AHN dataset, so that three classes were obtained in the end. The FCN results in 92.83% correctness and 92.67% completeness. As a comparison, the same dataset was classified by MLP and produces 90.90% correctness and 89.44% completeness.

Keywords
LIDAR, Filtering, DTM Extraction, FCN, CNN, MLP, Deep Learning

ACKNOWLEDGEMENTS

Sakabehing ngelmu iku asale saka Pangeran kang Mahakuwasa
All knowledge comes from God

Sander. Thank you for challenging me to use deep learning in my thesis. You always push me to a new place I can't imagine before.

Claudio. You gave me an insight of the deep learning. Thank you for the opportunity to use your deep learning framework.

Caroline. I always get a new idea after discussing with you. Thank you for always being supportive and reminding me to set back and see the bigger picture.

I would like to thank you to LPDP (Indonesia Endowment Fund for Education) for sponsoring my study in The Netherlands.

Intan Mustika. You always stand by my side; supporting me and keeping me enjoy the life. Thank you for making my life amazing.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Motivation and problem statement

1.1.1. Motivation

A Digital Terrain Model (DTM) is a digital representation of bare earth surface (Briese, 2010). The bare earth is a boundary between ground and objects attached to the ground, thus DTM contains elevation information of solid ground without any objects on it. Many different applications use DTM as important data source for their analysis within Geographic Information System (GIS) scope. Some applications that use a DTM are flood management, infrastructure and engineering planning, and environmental protection. Not to be confused with a DTM, a Digital Surface Model (DSM) contains elevation information of top surface, whether it is solid ground or non-ground objects (i.e. building, car, vegetation). Both digital models are the same in the open area but different in area with any objects attached on the ground.

Light Detection and Ranging (LIDAR) is the most popular method to generate DTM by filtering ground points from the entire point cloud. Compared to traditional photogrammetric DTM generation workflow, LIDAR has some advantages which lead some countries to replace their photogrammetric-based DTM into LIDAR-based DTM (Pfeifer and Mandlburger, 2009). While photogrammetry needs image matching to generate point cloud for filtering purpose, LIDAR obtains point cloud directly without additional processing. Thus, for DTM generation purpose, photogrammetry does not fit well in dense vegetation area because image matching fails to generate points on the ground surface (Rahmayudi and Rizaldy, 2016), so it is not possible to extract accurate and reliable DTM under vegetation canopies.

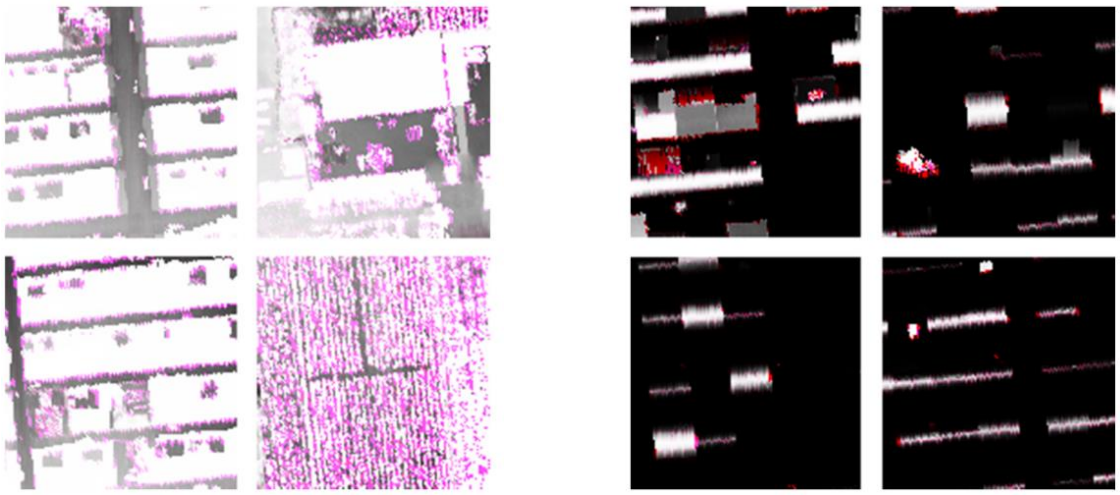
LIDAR overcome the problem because LIDAR relies on a single light trajectory for the 3D position calculation (Beraldin et al., 2010). As long as light can penetrate to the ground, LIDAR can measure the ground accurately. Even in vegetation area, LIDAR gives multiple returns for more advanced application. Since image matching photogrammetry relies on the texture measuring, in poorly textured areas (e.g. forest or desert), it is very hard for image matching techniques to find and match distinctive objects in one image to another image while LIDAR does not encounter this problem. LIDAR also does not need sunlight because LIDAR is an active sensor so it is more versatile. In order to extract DTM automatically from LIDAR, some existing filtering algorithms have been developed. In general, those algorithms filter ground points based on the assumptions that terrain is lying lower than other objects and set of ground points characterized as relatively smooth surfaces. Based on those assumptions, the algorithm can filter ground points from other non-ground points.

However, human intervention is still needed in order to have completely corrected DTM. This is mostly caused by the nature of ground which cannot be defined perfectly in geometric properties (Pfeifer and Mandlburger, 2009). Other main reason is the complexity of terrain structure on urban area (Briese, 2010). Finally, it can be concluded that most of filtering algorithms are unsupervised classification and based on some rules or assumptions. As an alternative, supervised-classification-based technique can be used for filtering. Some algorithms (Chehata et al., 2009; Niemeyer et al., 2012; Niemeyer et al., 2013; Lu et al., 2009; Zhang et al., 2013; Weinmann et al., 2015) have been developed based on supervised classification technique. The main idea is to extract information, so called contextual features, for each point such that those hand-crafted features can discriminate ground and non-ground point and use those features to train a model. Once model has been trained, the model can be used to filter other data. Recently, deep learning

has been used massively for image classification task. However, the implementation of deep learning for DTM extraction had not been conducted until Hu and Yuan (2016) published their research of exploiting Deep Convolutional Neural Network (CNN) for DTM extraction. Therefore, further investigation of the use of deep learning for DTM extraction would be interesting.

CNN has gained popularity for image classification or pattern recognition task. CNN has been proven to classify images for many datasets (Ciresan et al., 2011). The more advance CNN also has been developed which deal not only with spatial information of the image but also the temporal aspect so it is possible to detect an object in the video (Ji et al., 2013). ImageNet Large Scale Visual Recognition Challenge (ILSVRC) contest which held annually (Russakovsky et al., 2015) proved that deep CNN is the most accurate algorithm when dealing with image classification.

Despite the fact that LIDAR point cloud is not an image, it is still possible to treat point cloud as an image hence CNN could be used to classify ground and non-ground point. Hu and Yuan (2016) used Deep CNN to classify Airborne Laser Scanner (ALS) point cloud into ground and non-ground. In order to perform CNN classification, point cloud needs to be converted into images. Each single point is converted into single image based on a spatial pattern of the height difference in its neighborhood. This approach relies on the assumption that ground points have relatively lower elevation value than non-ground points. From the assumption, ground point images are brighter and significantly different to the non-ground point images as shown on Figure 1 below. Compared to TerraSolid software, the CNN-based filtering gives lower total error on ISPRS benchmark dataset (1.22% to 7.61%). This proves deep learning can be used to extract DTM accurately.



a. Ground point images b. Non-ground images

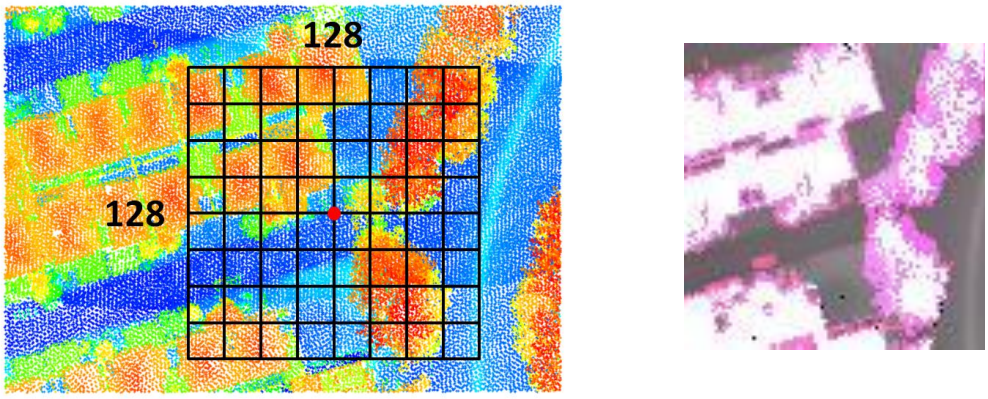
Figure 1. Extracted (a) ground and (b) non-ground images
Source: Hu and Yuan, (2016)

1.1.2. Problems

However the CNN approach on previous research has drawbacks, mainly related to the point-to-image conversion. The first drawback is large computational time. The second one is ground and non-ground images which are extracted in a sloped terrain are not significantly different, in contrast to the original idea to have brighter images for ground points and darker images for non-ground points.

- **Computational time**

As seen in Figure 2.a, a moving window is fitted on one point (red dot) to collect information of the neighbors. This moving window has 128 x 128 cells and the window size is 96 x 96 m (0.75 x 0.75 m for each cell). Next, this moving window is converted into an image (Figure 2.b), one cell becomes one pixel, and thus 128 x 128 pixels of image is representing one single point. Each pixel value of the image is calculated from the height difference ($Z_{neighbor} - Z_{point}$) between neighboring point(s) in the pixel and the corresponding point (red dot). As a result, 128 x 128 = 16,384 calculations of height differences need to be computed, only for converting one point into image. This leads to slow computational speed for real case problem that deals with hundreds of millions points. This approach is different from other supervised-classification-based filtering techniques where height difference feature is only calculated once for every single point.



a. The window in the point-to-image conversion b. Output image

Figure 2. Illustration of (a) point-to-image conversion and (b) output image

- **Sloped terrain**

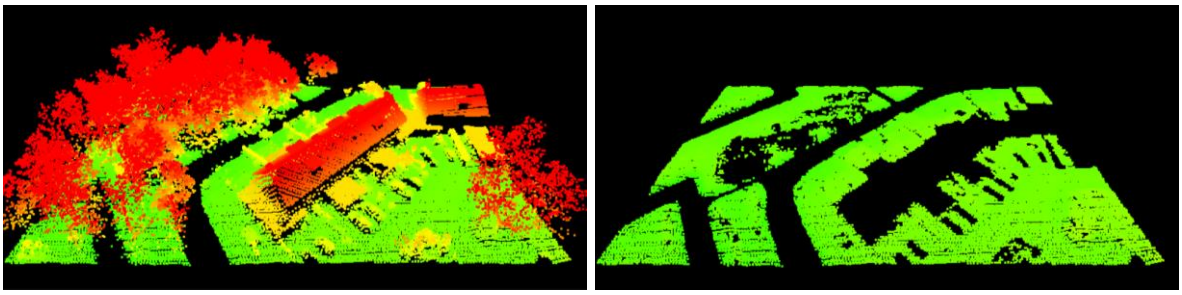
Another drawback is this approach deals only with the height differences feature which works well in a relatively flat terrain. As a consequence, extracted images from ground and non-ground points in a sloped terrain are not significantly different as shown in Figure 3. The situation is not in line with the original purpose of point-to-image conversion when ground point images look brighter than non-ground point images. Therefore, this research will investigate the use of deep learning for filtering ground point from LIDAR point cloud in more efficient way and adapt better in a sloped terrain.

a. Ground point images b. Non-ground point images

Figure 3. (a) Ground and (b) non-ground point images in a sloped terrain

1.2. Research identification

DTM Extraction from LIDAR point cloud basically filters ground points from non-ground points (building, tree, car, etc.) as shown in Figure 4 below. Many filtering algorithms were developed only based on geometric information while LIDAR offers other useful information, for instance intensity value and return number information, which can be used for classification-based filtering.



a. Unclassified point cloud b. Ground points only

Figure 4. A typical LIDAR filtering task: (a) Unclassified point cloud and (b) ground points

Previous work (Hu and Yuan, 2016) has exploited the use of Deep CNN to extract DTM by classifying ground and non-ground points from ALS point cloud. However, it only deals with the height difference information. It also has problems due to the procedure of point-to-image conversion. This study developed a more efficient deep learning-based DTM extraction algorithm which exploits other informative features of LIDAR point cloud and works better in a sloped terrain. Furthermore, the proposed method has been extended to accommodate more classes, such as vegetation and building, in the classification.

1.2.1. Research objectives

The main objective of the proposed research is:

> To develop a more efficient ground classification workflow based on the deep learning for DTM extraction from LIDAR point cloud which works better on sloped terrain

In order to achieve the main objective, the following sub-objectives have to be conducted.

1. To design a deep learning workflow that can handle point cloud datasets
2. To investigate and explore the influence of potential useful features for ground classification purpose
3. To compare the proposed method with the previous CNN-based technique, LAStools software and other supervised-classification techniques.
4. To investigate further application of point cloud classification by adding vegetation and building classes.

1.2.2. Research questions

The following questions have to be answered for the sub-objectives above.

Sub-objective 1:

1. What deep learning architectures have been proposed for general classification problem in the literature?
2. What deep learning architectures that could deal with point cloud dataset?

3. How to convert point cloud such that point cloud can be consumed by image-based deep learning architecture?
4. How to select ground point in case of the output result comes from image-based deep learning architecture?
5. Which deep learning approach is most suitable for the task?

Sub-objective 2:

1. What are LIDAR features that can be used to help classification?
2. What is the influence of those LIDAR features?
3. How to use those LIDAR features in the deep learning model?

Sub-objectives 3:

1. How does the accuracy and performance of the proposed method compared to the previous CNN-based technique, LAStools software and other supervised-classification technique?

Sub-objective 4:

1. What will change if vegetation and building classes are added?
2. How to adopt the proposed method for multi-class classification?

1.2.3. Innovation aimed at

The proposed research aims to develop a new method of DTM extraction by deep learning classification from LIDAR point cloud. Previous deep-learning-based technique has been developed and produces high accuracy compared to the popular software for DTM extraction (Hu and Yuan, 2016). The innovation of this research is to reduce computational time compared to the previous research. As mentioned in the section 1.1.2, the CNN-based approach has a drawback of computational speed when converting points into images because every single point is converted into single image. This algorithm leads to large computational cost.

To solve the problem, this research proposes to create a single large image for the entire point cloud instead of creating a separate single image for every single point as used by Hu and Yuan (2016). This single large image is treated as an image in the image classification problem as usually done in land cover / land use classification. In order to perform classification, the use of Fully Convolution Network (FCN) architecture was proposed. FCN is a modification of CNN architecture; specially designed for pixel-wise classification. In contrast to common CNN architecture that gives only one label for one input image, FCN gives one label for every pixel of the input image (Long et al., 2017). Note that Hu and Yuan (2016) also use a common CNN architecture.

The FCN architecture makes it very suitable for pixel-wise image classification problem. The proposed research investigated the use of FCN for LIDAR ground classification task. Another innovation is, unlike the previous CNN-based approach, the proposed method not only utilizes height difference feature, but also other geometric and non-geometric features such as intensity value and return number information.

As an alternative, another method was also investigated. Weinmann et al. (2015) successfully classified point cloud into many classes. The method extracts 21 geometric features for every point and trains many different classifiers. One of them is deep learning using Multi-Layer Perceptron (MLP). The alternative method in this research adapted the method from Weinmann et al. (2015) but a modification was carried out in terms of the features and the architecture to fit for ground classification purpose.

In contrast to the proposed method that relies on an image-based classification, the alternative method classifies points directly without the need of a point-to-image conversion. In order to do that, hand-crafted features are extracted for every single point to train a deep MLP network in the point-based classification. Extracted features are similar to the features that are used in many literatures for point cloud classification. The MLP network learns from those features to discriminate ground point from non-ground point. In general, this method combines the idea of features engineering with feature learning.

1.3. Project setup

1.3.1. Method adopted

The proposed method relies on image-based classification while the alternative method relies on point-based classification. The idea of the proposed method is to create a single image from point cloud and assign feature value as a pixel value. Deep FCN is trained using the images. It is based on a Deep FCN architecture as used by Persello and Stein (2017). Some modifications in the network were carried out in order to fit the network for DTM extraction purpose. The result of classification is ground and non-ground-labeled pixels. Since the ground is assumed as the lowest point within a certain area (if there is no outliers), the lowest points within ground pixels are assigned as initial ground points. Next, a surface is created connecting those initial ground points. Finally all points within a threshold are labeled as ground. The alternative method extracts features for every single point to train an MLP network and uses the trained MLP network to classify point cloud into ground and non-ground.

1.3.2. Thesis structure

This document is organized in seven chapters. The first chapter describes the motivation of the study. The second chapter gives a briefly description of the existing research in the literature as well as the description of MLP, CNN and FCN. The designed methodologies are explained in detail in the third chapter. The fourth chapter describes the datasets and the experiments in order to fine-tune the network. The comparison methods (CNN, RF and LAStools software) are also explained in the chapter four. The results of all methods are presented in the fifth chapter as well as the accuracies assessments and the computational time performance comparison. The sixth chapter presents the result of further experiment when vegetation and building classes were added in the classification. Lastly, chapter seven recaps the works and lists the possible improvements in the future.

2. LITERATURE REVIEW

2.1. Related work

Traditionally, there are four approaches to filter ground point from LIDAR point cloud. The first approach is slope-based filtering (Vosselman, 2000; G Sithole, 2001). It is based on an assumption that terrain is relatively flat. If there is a large height difference between two nearby points then it is not caused by the terrain slope but it is caused by both points are consisted of ground and non-ground points where ground point is positioned in a lower height than non-ground point. Slope-based filtering uses erosion and dilation from mathematical morphology for its implementation.

The second approach is progressive densification (Axelsson, 2000). It is based on an assumption that the lowest point in particular area should be ground point. These points initially are assigned as seed point then the algorithm creates Triangulated Irregular Network (TIN) from these points and the surrounding points. The surrounding points are decided as ground or non-ground point based on the angle and distance parameters. Next, TINs are created progressively for the next points. These iterative steps gradually build terrain surfaces.

The third approach is surface-based filtering approach (Kraus and Pfeifer, 1998). It relies on the assumption that all points belong to ground and remove points which do not fitted as ground. At the beginning, the algorithm gives same weights for all points and creates best-fitted surfaces to all points then iteratively changes the weight based on the assumption that points below surface are ground and points above surface are non-ground. In the next iteration, all points below surface have higher weight while points above surface have lower height and a new surface is created based on the new weights. These steps iterate until converge and the final surface in the last iteration should be ground surface.

The fourth approach is segment-based filtering (Sithole and Vosselman, 2005). Unlike other algorithms, this approach works on segments not points. This approach creates segments of similar points and analyzes which segments belong to 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.

Improvement of all above algorithms has been done in recent years. Revised version of progressive densification was proposed to avoid misclassification of non-ground point into ground point by changing the angle criterion (Nie et al., 2017). Some revisions also have been developed for slope-based filtering by modifying the structuring element (Kilian et al., 1993) and for surface-based filtering by adding robust interpolation method to deal with large buildings and minimizing computation time (Pfeifer et al., 2001). Another recent algorithm called parameter-free also has been developed by thresholding the standard deviation of top-hat transformation of the points (Mongus and Zalik, 2012).

Since 2016, a new approach of filtering algorithm is introduced by Hu and Yuan (2016) based on deep learning classification. This approach extracts feature image for every single point to represent the spatial pattern of each point to its neighborhood. The large numbers of training samples are used to train the deep CNN model in order to successfully discriminate ground and non-ground points in many different landscapes.

Supervised classification technique, typically used for land cover classification in the image analysis domain, has an opportunity to be used for LIDAR filtering purpose (Chen et al., 2017). Hand-crafted features of point cloud are extracted to train a classifier model. Methods from recent researches extract not only geometric features but also other contextual features such as intensity, echoes information and, if available, properties of full waveform LIDAR. Geometric features not only limited to height difference between point and the neighbors (which is the most important feature for ground classification), but also use other features for instance point density ratio, eigenvalue and local planarity (Chehata et al., 2009).

Many classifiers has been used to train the model such as Random Forest (Chehata et al., 2009) and Support Vector Machine (Zhang et al., 2013). Conditional Random Field (Niemeyer et al., 2012; Lu et al., 2009) also used to improve the result from the classifier. Weinmann et al. (2015) has a comprehensive research using many different definitions of neighbors, features and classifiers to study the effects on the accuracy. A combination of segmentation and machine learning by employing Random Forest followed by Conditional Random Field was also proposed in a recent study (Vosselman et al., 2017). Unlike unsupervised filtering techniques, most of these supervised classifiers not only classify point clouds into ground and non-ground classes but also classify into other classes.

In the deep learning field, there are recent works that use deep learning to classify point cloud (Wu et al., 2015; Maturana and Scherer, 2015; Qi et al., 2017). However, those deep learning algorithms are designed for 3D object recognition (to predict a class from set of points which represent object such as chair, table, bed, etc.). For ground classification purpose on semantic point classification, Hu and Yuan (2016) has developed a new approach based on the deep CNN as mentioned earlier.

2.2. Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) is a feedforward neural network. It is a basic schema of deep neural network for more sophisticated architecture such as Convolutional Neural Network (CNN). In classification, the purpose of MLP is to predict a label y given input x by learning the parameters θ in such a way so that the learned parameters give the prediction as close as possible to the true label.

$$y = f(x; \theta) \tag{1}$$

The network comes when several functions are composed together. Suppose there are three functions f^1, f^2, f^3, then all functions are stacked in a chain to create a network. In MLP, function f^1 is called the first layer; f^2 is the second layer; and so on. This chain creates a depth of the network thus the name 'deep learning' comes (Goodfellow et al., 2016).

$$f(x) = f^3(f^2(f^1(x))) \tag{2}$$

The architecture of MLP can be seen as a network which contains several layers. In that schema, the last layer is called output layer while the previous layers are called hidden layer. The first layer is connected to the second layer and so on. Each hidden layer contains several hidden units. If all hidden units are connected to all hidden units in the following layer, then the layers are called fully-connected layer. Figure 5 shows a typical MLP architecture consists of three layers.

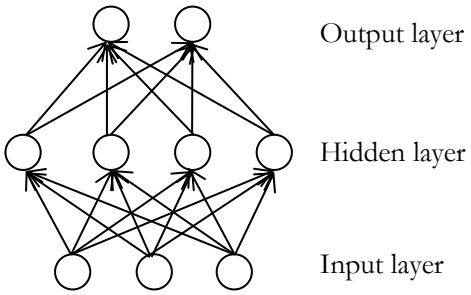


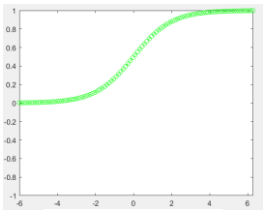
Figure 5. A typical MLP architecture

Each hidden unit has learnable parameters weights W and bias b. The function of W and b can be seen as a linear function. Then nonlinear function is added; usually called as activation function g. Several activation functions are exist and described in the following sub section. Hidden unit h_1 in the first layer is defined in the equation (3). For the next layer, hidden unit h_2 is defined using the same function, but the input is h_1 from the output of the first layer instead of x. This schema holds for the third layer and so on. Finally, input x is mapped through several layers in the network and resulting output unit h_i for i-number of layers.

$$h = g(W^T x + b) \tag{3}$$

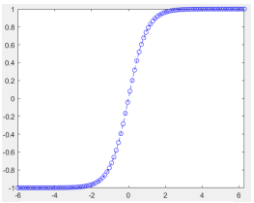
2.2.1. Activation function

Sigmoid (σ) or hyperbolic tangent (tanh) is the common activation function for the most neural networks. As seen in Figure 6 below, the curve of sigmoid and hyperbolic tangent are similar because both are closely related; the different is sigmoid has range of [0,1] while hyperbolic tangent is [-1,1]. Therefore, the derivative of hyperbolic tangent is higher hence the gradient is stronger than the gradient of sigmoid.



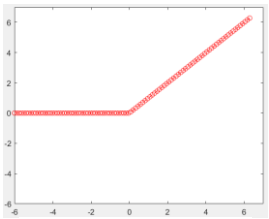
$$f(x) = \frac{1}{1 + e^{-x}}$$

a. Sigmoid



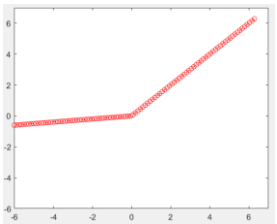
$$f(x) = \frac{2}{1 + e^{-2x}} - 1$$

b. Hyperbolic tanget



$$f(x) = \begin{cases} x & if\ x > 0 \\ 0 & otherwise \end{cases}$$

c. ReLU



$$f(x) = \begin{cases} x & if\ x > 0 \\ 0.01\,x & otherwise \end{cases}$$

d. Leaky ReLU

Figure 6. The different between (a) sigmoid, (b) hyperbolic tangent, (c) ReLU and (d) leaky ReLU as an activation function

Another activation function called rectified linear unit (ReLU) was introduced by Nair and Hinton (2010), and it becomes more popular in recent years (LeCun et al., 2015). ReLU outputs the input into its value if the input is larger than zero, otherwise ReLU outputs zero. ReLU is similar to linear function but the function cuts values below zero hence the range is [0, infinity]. The derivative is 1 when the unit is active and zero if the unit is not active. The advantage is the derivative is always large and consistent whenever the unit is active. It also proved that replacing conventional function such as logistic sigmoid or hyperbolic tangent with ReLU gave better result (Glorot et al., 2011). Furthermore, leaky ReLU (Maas et al., 2013) is variant of ReLU where small value (such as 0.01) is introduced to avoid zero gradient when the unit is not activated.

2.2.2. Gradient-based learning

As mentioned earlier, MLP for classification works by mapping input unit *x* through all layers in the network and resulting the predicted labels in the final layer. In order to predict the correct label, all parameters (*W* and *b*) in the network are learned; mostly using gradient descent.

The predicted label itself is computed by softmax function. It calculates the probability distribution over *j* number of classes, given input *h* from the output of the final layer in the network. For *j* = 1… *K*, softmax turns *K*-dimensional vector of *h* into *K*-dimensional vector of *p(h)* in the range [0,1] which sum up into 1. Most of neural networks use softmax function as a classifier because the output is in probability range. Then the input *x* is labeled as *j*-th class which has the highest probability.

$$p(h)_j = \frac{e^{h_j}}{\sum_{k=1}^{K} e^{h_k}} \quad (4)$$

Table 1 shows an example of how softmax calculates the probabilities from one input unit *x* for each hidden unit in the final layer. Suppose it is a three-class classification, then the final layer should have three hidden unit for three classes. Each hidden unit contains value which is mapped from all previous layers. It can be seen that softmax outputs close-to-one value to largest hidden unit and close-to-zero value to the rest. Hence the name is softmax, because it represents the smooth version of the winner-takes-all model (Bishop, 1995).

Class (j = 3)	Hidden unit h	e^h	p(h)
A	5	148.41	**0.936**
B	2	7.39	0.047
C	1	2.72	0.017
		Σ = 158.52	Σ = 1

Table 1. An example of softmax function for three-class classification

Before the parameters are learned, the loss function is introduced to show how well the network predicts the label. Cross entropy is a common loss function in modern neural networks (Goodfellow et al., 2016). It calculates the negative log-likelihood between prediction label and true label from training samples. Formally, it is defined as

$$L(x, y; \theta) = -\sum_j y_j \, log \, p \, (h_j|x) \quad (5)$$

Then learning is an iterative process to minimize the loss function with respect to all parameters (θ) in the networks. The process is based on the backpropagation algorithm (Rumelhart et al., 1986). For n number of training samples, it is defined in the equation (6).

$$\theta^* = argmin_\theta \sum_{n=1}^{N} L(x^n, y^n; \theta) \quad (6)$$

In order to minimizing the loss function, all parameters are adjusted using stochastic gradient descent with momentum.

In summary, MLP for classification works by building a network that contains learnable parameters which are stacked into several layers. Predicted label *y* is given to each input *x* by mapping *x* through all layers where softmax function calculates the probabilities of *x* in the final layer. Once the predicted label is obtained, loss function is introduced to calculates the cross entropy between the predicted label *y* and the true label *y'*. All learnable parameters are learned using gradient descent in order to minimize the loss. After the loss is minimized, *y* is close to *y'* hence the network is reliable to classify every input *x*. Finally, the network can be used to classify other data (testing samples) by forward passing through all layers until the predicted label is obtained in the final layer.

2.3. Convolutional Neural Network

CNN, also known as Convolutional Network (ConvNet), is a special case of neural network architecture where the input has grid-like topology (Goodfellow et al., 2016). Although it is possible to process 1D grid data i.e. time-series data, CNN raised its popularity to process 2D grid data such as image. When the task is image classification, CNN is better suited than MLP due to its architecture. The main different of CNN architecture compared to MLP is CNN is composed by at least one convolutional layer in the network.

Unlike fully-connected layers in MLP, convolutional layers use parameter sharing. Instead of connecting all units in the hidden layer, CNN only 'see' a local area within a filter and use the same filter to convolve for the entire area in the image. This reduces the number of parameters drastically hence it can avoid overfitting. Network with a huge number of parameters is more likely to have an overfitting (O'Shea and Nash, 2015). On the other hand, MLP can be used to process an image as well. But the number of parameters (weights) can increase drastically following an increase of image size and depth. This would not be a problem for small and black/white images such as MNIST dataset, where the image only has 28 x 28 x 1 pixels (784 weights for one neuron). But this would be a serious problem for large and colorful images such as ImageNet dataset that has size of 224 x 224 x 3 pixels (150,528 weights for one neuron). As mentioned before, this large number of parameters affects to slow computation time and tends to have an overfitting.

CNN architecture is stacked by four types of layers. They are convolutional layer, activation layer (ReLU), pooling layer, and fully-connected layer. The simple CNN architecture can be seen in the Figure 7 below. If the networks have a deeper layer, most of CNN architectures always follow the same order. Convolutional layer, activation layer and pooling layer are repeated several times before fully-connected layer in the final layer.

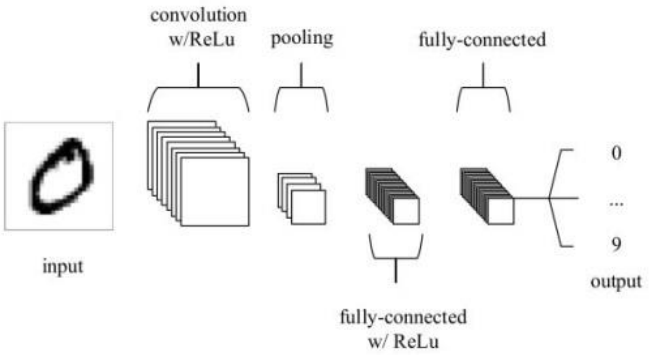


Figure 7. Simple CNN architecture with 9 ouput classes for MNIST dataset classification
Source: O'Shea and Nash (2015)

In a forward pass, the output of each layer is called feature map since it is a raster-grid. The function itself is similar to the function of hidden unit in MLP except that the layer in MLP is vector. For input image x with depth D, weights W, bias b, and ReLU as an activation function, the output feature map h is defined as

$$h = max\{0, \sum_{d=1}^{D} W^T{}_d * x_d + b\} \quad (7)$$

In a CNN, convolutional filter contains parameters W and b. When the filters are convolved in the image, those parameters are shared to the entire area. This is the concept of parameter sharing in CNN as mentioned earlier. When it comes to the learning process, CNN is similar to MLP. It works by forward passing input x through all layers and back-propagating the output to learn all parameters in the network. But, instead of vector, the layers in the CNN are raster-grid with depth as a representation of the number of filters. Similar to MLP, the output of forward passing is the probabilities of each input image x in the final layer. Then the learning process is done similar to learning process of MLP by using gradient descent. After all parameters are learned and the loss is minimized, the network is reliable to classify the testing sample images.

2.3.1. Convolutional layer

Convolutional layer is the most important layer of the network. This layer has learnable filters that containing weights and will be adjusted during training. Common size of this convolutional filter is 3x3 or 5x5 as seen in Simonyan and Zisserman (2014), Szegedy et al. (2015) and Krizhevsky et al. (2012). This size is called receptive field because this is an area where the networks "see" the image. Rather than having different filter over different spatial area within an image, CNN has same filter for different spatial area in the image. This means the weights of convolutional filters will be same for whole area and often called parameter sharing. The reason behind this is if one small area can be used to compute the feature then it will be useful to use the same feature in another area (O'Shea and Nash, 2015). Another reason is for efficiency (Goodfellow et al., 2016).

Convolutional filters on the first layer convolve over an input image and generate an output feature map. For this task, there are three hyper-parameters need to be defined.

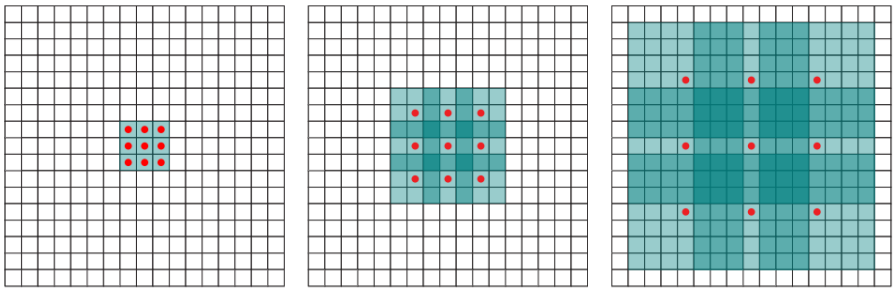
- Filter depth is the number of filters and control how many channel will be created on the output feature map.

- Stride is how the filters will move over an image. Stride 1 means filters will move on the next pixel without any gap, while stride 2 means filters will skip 1 pixel before filters read the next pixel.
- Padding keeps the filter to convolve in the border of the image. This will add pads around the image border and also control the size of output feature map. Usually people use padding to maintain the size of output feature map to have the same size as the input size.

For arbitrary input size (H x H x depth), filter size (F x F), stride (S) and padding (P), the following equation is shown how to calculate the size of output feature map. The depth of the output feature map will follow the depth of the convolutional filter.

$$\frac{(H-F)+2P}{S} + 1 \quad (8)$$

Dilated convolution (Yu and Koltun, 2015) is a variant of convolutional filter and used in the recent development. It is specially designed for dense semantic segmentation. The idea is to aggregate multiscale contextual information without losing its resolution. The dilated convolution will increase receptive field size while maintaining the number of parameters thus avoiding overfitting due to the large number of parameters. This also gives a benefit of efficiency in terms of memory size of the computation. In case of DTM extraction from LIDAR point cloud, the dilated convolution plays an important role because it is important to have a large receptive field to cover large buildings.



a b c

Figure 8. Dilated convolution.
(a) 3 x 3 Receptive field size from 1-dilated convolution; (b) 7 x 7 Receptive field size from 2-dilated convolution; (c) 15 x 15 Receptive field size from 4-dilated convolution
Source: Yu and Koltun (2015)

2.3.2. Pooling layer

Pooling layer is a down-sampling strategy and stacked after convolutional layer and ReLU to summarize the output feature map within a small area. Pooling layer usually combined with stride to drastically reduce the size of output feature map, the number of parameters and the computational complexity of the model (O'Shea and Nash, 2015). Pooling also invariant to small-translation which makes it very useful for image classification (Goodfellow et al., 2016). Pooling layer usually only has a small filter size such as 3x3; otherwise pooling layer will be very aggressive. Max pooling is a typical pooling strategy which takes a maximum value within a small neighborhood as an output feature map. This strategy empirically outperforms other pooling strategy (Scherer et al., 2010). But, even though max pooling is a popular pooling strategy in recent days, a study from Springenberg et al. (2014) shows that removing (max)pooling layer while at the same time increasing the stride of convolutional layer can give a competitive result but in a simpler architecture.

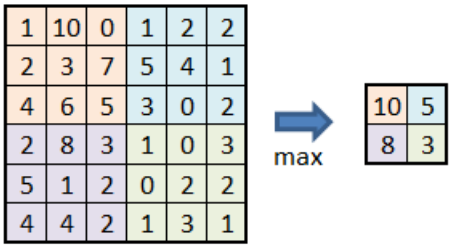


Figure 9. Example of 3x3 max pooling with stride 3

2.4. Fully Convolutional Network

In a CNN, every input image is passed through all layers until the label is obtained in the final layer. Meanwhile, the size of feature map is reduced in the following layer since pooling layers are employed in the network. This down-sampling schema can be seen in Le-Net5 by Le Cun et al. (1998) in the Figure 10 below. As can be seen, the original image has 32x32 pixels then the feature maps are down-sampled into 28x28 in the first layer, 14x14 in the second layer, 10x10 in the third layer and finally 5x5 in the fourth layer.

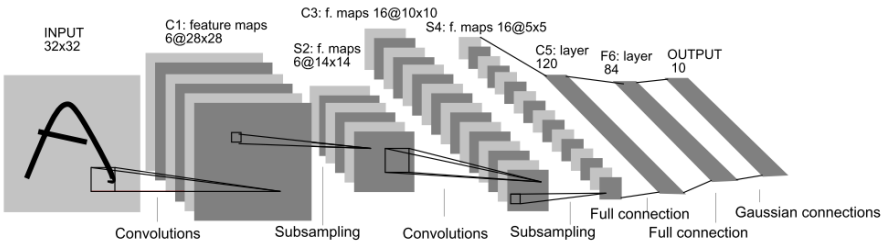


Figure 10. The down-sampling in CNN architecture
Source: Le Cun et al. (1998)

This CNN architecture is designed for image-wise classification where each image is given one label. In the case of pixel-wise classification, where every pixel must be labeled, this kind of architecture cannot provide label for each pixel.

The architecture of CNN needs to be modified for dense labeling. In the past, people use patch-wise training to achieve dense labeling. Ning et al. (2005) modified the architecture by creating patches of 40x40 pixels from the original image, which produces 1x1 pixels in the output. The 40x40 size is chosen due to the size of the object of interest in that case. Every patch has a network and produces one label. Therefore, the full network can be seen as a multiple replicas of all networks. Farabet et al., (2013) proposed to use combination of multi-scale CNN and over-segmentation technique. Input image is transformed into three scales. After each scale produces features maps, the outputs are concatenated. In parallel, the same input image is segmented. Finally the dense labeling is done by combining the features and the segmentations. Another patch-wise approach was designed by (Pinheiro and Collobert, 2014) in a Recurrent CNN architecture. The network consumes patches of images and consists of three stages. In each stage, label is obtained for the pixel at the center of the patch. Then the label is fed to the next stage thus creates recurrent network. Moreover, Bergado et al. (2016) used a patch-wise approach for semantic classification of sub-decimeter resolution aerial images. Patches of 33x33 pixels are extracted for each pixel and CNN takes the patches in order to classify the pixel in the center of the patch. The 33x33 size is about twice the dimension of the smallest object of interest and produces better result than any other size.

Even though CNN can produce dense labels, the patch-wise approaches do not have end-to-end fashion since other processes were needed, either patches creation or segmentation. In order to address the issue, Long et al. (2017) proposed a Fully Convolutional Network (FCN). The network has end-to-end schema which consumes an image, learns the features, and outputs labels for every pixel without additional works.

As CNN architecture down-samples the feature maps (as can be seen in Figure 11), these coarse outputs are needed to be connected to the pixels of the input image in order to have a label for each pixel. One simple approach is by shift-and-stitch trick. But a deconvolutional layer offers more effective and efficient method. Unlike convolutional layer that down-samples the feature maps, deconvolutinal layer up-samples the feature maps hence the coarse outputs can be up-sampled into the original resolution. In training phase, the network learns how to deconvolve the coarse feature maps from the convolutional layer. In order to obtain finer result, Long et al. (2017) not only used the output from the last convolutional layer, but also used the output from the previous convolutional layers. Then the network combines the predictions from all convolutional layers while uses smaller stride for more precise result. The FCN architecture by Long et al. (2017) can be seen in Figure 11. It shows three different schemas from different input of convolutional layers which are used for the deconvolutional layer. In a remote sensing domain, Maggiori et al. (2017), Volpi and Tuia (2017) and Fu et al. (2017) adapted the deconvolutional layer for pixel-wise classification of high resolution remotely sensed imageries.

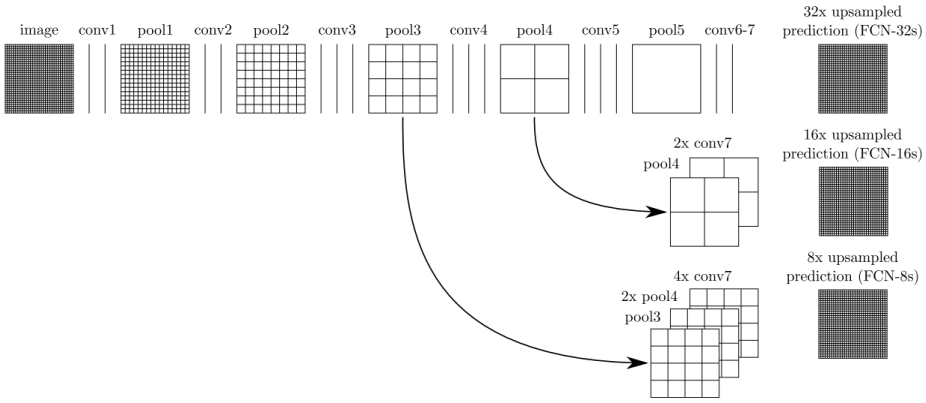


Figure 11. FCN architecture in three different scales. The FCN-8s uses input from the last convolutional layer and two previous pooling layers to achieve finer result.
Source: Long et al. (2017)

In addition to FCN, Badrinarayanan et al. (2015) proposed SegNet for end-to-end architecture of dense labeling. SegNet has encoder network and a corresponding decoder network. The encoder outputs down-sampled feature maps by using convolutional layer while decoder outputs up-sampled feature maps. SegNet is different to FCN in terms of the method for the up-sampling. The decoder of SegNet relies on the result of max-pooling indices from the corresponding encoder. The smaller size feature maps are mapped into the larger size feature maps using those indices. This would create sparse feature maps since the feature maps are partially filled. Suppose 2x2 layer (contains 4 elements) is mapped into 4x4 layer (contains 16 elements), the larger layer only has 4 elements instead of 16. Then the learnable filter is added to densify the feature maps. The architecture of SegNet can be seen in the Figure 12 below.

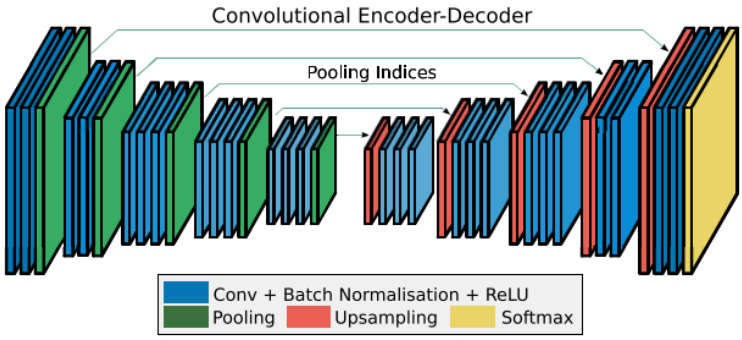


Figure 12. SegNet architecture.
Source: Badrinarayanan et al. (2015)

Another architecture was proposed by Sherrah (2016) and Persello and Stein (2017). Unlike the architectures from Long et al. (2017) and Badrinarayanan et al. (2015) where the layer is down-sampled and up-sampled, Sherrah (2016) and Persello and Stein (2017) used no down-sampling architecture. In this schema, the feature maps of every layer are maintained to have the exactly same size to the input image. Dilated convolution is used in order to capture larger spatial patterns in the image because the architecture does not down-sample the image. In 'down-sample and up-sample' architectures, the convolutional layers do this task. In a no down-sampling architecture, the network only learns in a small spatial extent if dilated convolution is not employed. Indeed, larger filter could be used to capture larger area but increasing filter size would drastically increase the number of parameters.

In order to maintain the size of feature maps, the network uses stride equal to one for all filters. This means the convolutional and pooling filters convolve for each pixel to the next pixel without any gaps hence it ensures the pixel of the input is mapped to the same spatial position in the feature map. In addition, zero padding is also mandatory to allow the filter to start from the boundary of the image. Figure 13 shows the difference between stride equal to one with zero padding and stride equal to two without zero padding. It can be seen that the output in the first configuration has exactly the same size as the input.

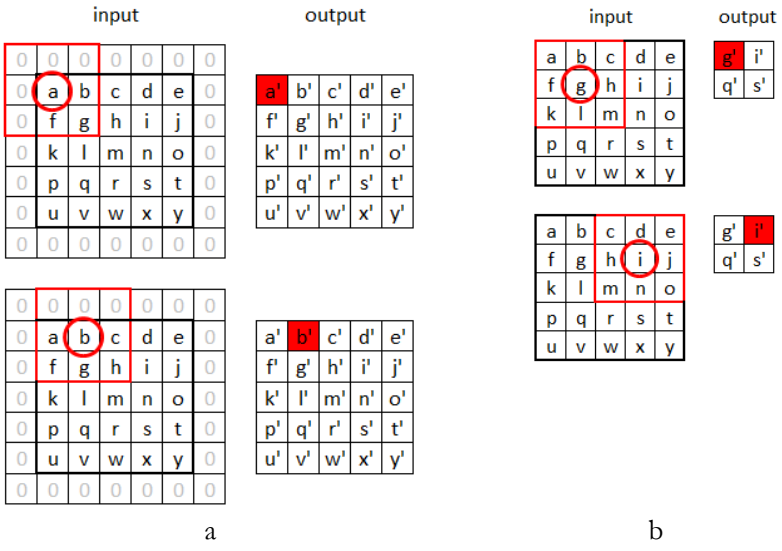


Figure 13. The difference between (a) stride is equal to one and zero padding is added; and (b) stride equal to two and zero padding is not added. The filter size is 3x3 (red square).

In the left-side example, the filter starts from pixel *a* in the boundary of image because zero padding is added, and outputs *a'*. Then the filter moves to *b* due to stride is equal to one, and outputs *b'*. In the right-side example, the filter starts from pixel *g* instead of pixel *a,* because zero padding is not added. Then the filter skips pixel *h* and moves to pixel *i* because the stride is equal to two. As a result, the output size in the left-side example is exactly the same as the input (5x5). While in the right-side example, the output size is 2x2. Therefore, stride and zero padding must be set carefully in order to build the no down-sampling architecture. If the network does not down-sample the input image, then the output size is the same as the input hence all pixels are able to be labeled.

The architecture was successfully tested for pixel-wise classification of informal settlements detection (Persello and Stein, 2017) using remote sensing imagery. In addition, Gevaert et al. (2018) adapted the similar no down-sampling method for DTM Extraction from Unmanned Aerial Vehicle (UAV) imagery. Figure 14 below shows the no down-sampling FCN_DK network by Persello and Stein (2017). It can be seen that the pad increases together with the increasing of dilation factor in the following layer.

Layer	module type	dimension	dilation	stride	pad
DK1	convolution	$5 \times 5 \times 4 \times 16$	1	1	2
	lReLU				
	max-pool	5×5		1	2
DK2	convolution	$5 \times 5 \times 16 \times 32$	2	1	4
	lReLU				
	max-pool	9×9		1	4
DK3	convolution	$5 \times 5 \times 32 \times 32$	3	1	6
	lReLU				
	max-pool	13×13		1	6
DK4	convolution	$5 \times 5 \times 32 \times 32$	4	1	8
	lReLU				
	max-pool	17×17		1	8
DK5	convolution	$5 \times 5 \times 32 \times 32$	5	1	10
	lReLU				
	max-pool	21×21		1	10
DK6	convolution	$5 \times 5 \times 32 \times 32$	6	1	12
	lReLU				
	max-pool	25×25		1	12
class.	convolution	$1 \times 1 \times 32 \times 2$	1	1	0
	softmax				

Figure 14. No down-sampling FCN_DK architecture using 6 dilated kernel
Source: Persello and Stein (2017)

3. METHODOLOGY

Two different methods were proposed in this study. The first proposed method was performed on the image-based classification approach. The alternative method was based on extracting features for each point, then the classification was performed based on those point features thus it was named as point-based classification. Other approach (CNN, RF and LAStools) were used as comparison methods.

An overview of both proposed methods is shown in the following flowchart.

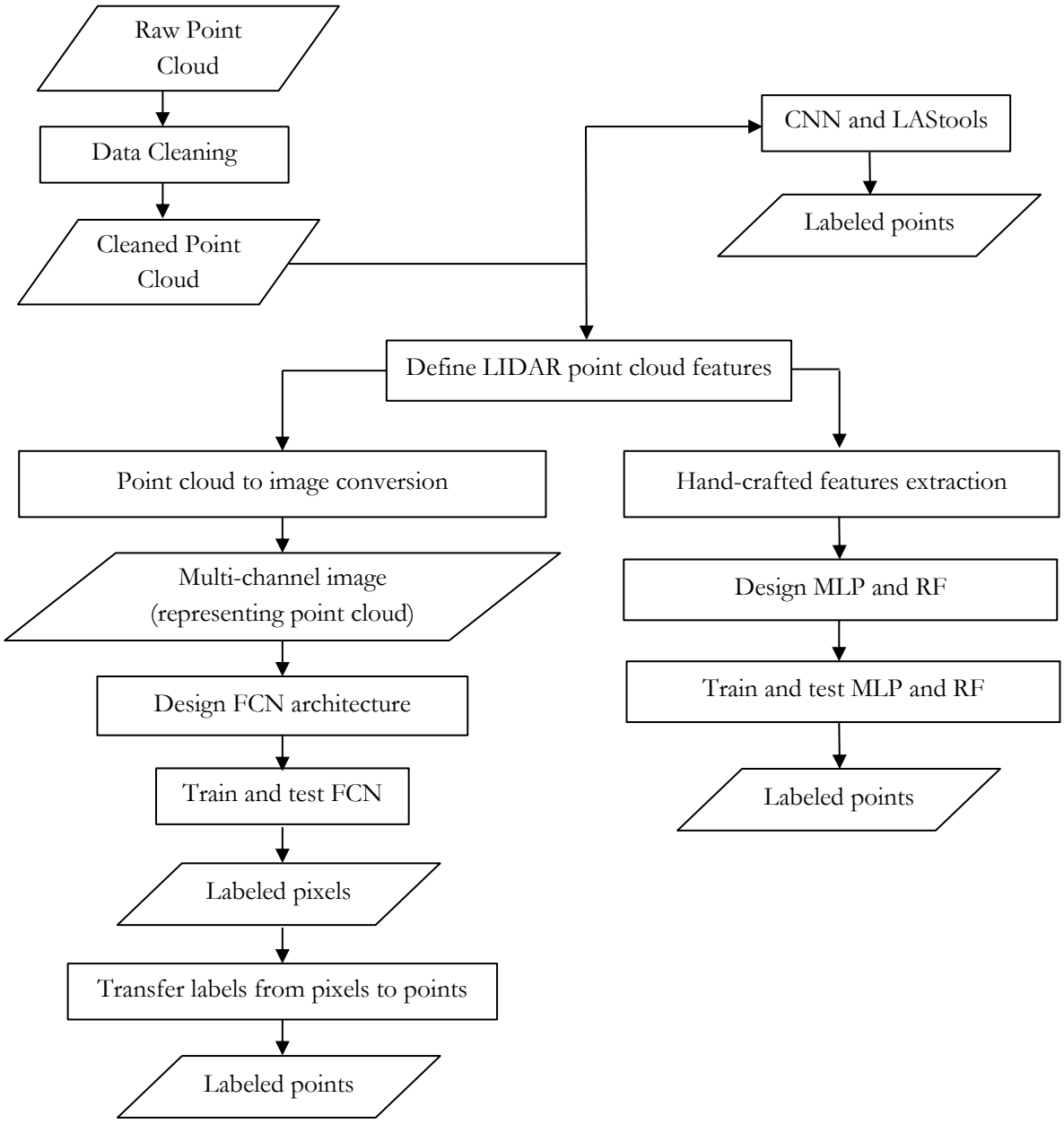


Figure 15. The workflow of the proposed methods

3.1. Fully Convolutional Network

The proposed method in this study is inspired by the method developed by Hu and Yuan (2016). It is believed that their research is the first deep learning-based method for ground classification purpose. In general, the algorithm works by converting every single point into image. Every pixel value of the extracted image from point *i* is calculated using the following equation:

$$F_{red} = [255 * Sigmoid(Z_{max} - Z_i) - 0.5] \quad (9)$$

$$F_{green} = [255 * Sigmoid(Z_{min} - Z_i) - 0.5] \quad (10)$$

$$F_{blue} = [255 * Sigmoid(Z_{mean} - Z_i) - 0.5] \quad (11)$$

$$Sigmoid(x) = (1 + e^{-x})^{-1} \quad (12)$$

It is assumed that ground point has lower elevation than non-ground point. If the assumption is true, the extracted ground point images have higher pixel values than non-ground point images. After all points have been converted into images, deep CNN model is trained. Since image has size of 128 x 128 pixels, it is very obvious that the point-to-image conversion has many redundant calculations.

In order to avoid the redundant calculation, the conversion in this study converts all points into a single large image rather than converting every point into a separate image. However, the proposed conversion leads to another consequence. Since the conversion projects 3D point cloud into 2D raster, not all points are able to be accommodated in the image. Only one point is able to be represented in one pixel. On the other words, point cloud loses its advantage as a 3D data after the conversion. In the classification, labels are predicted for each pixel not each point. In order to have dense labels for all points, the labels are transferred from pixels to points by creating a surface that connects all ground points from the lowest point in a pixel. The motivation and detailed explanation of these additional steps are described in section 3.1.1 and 3.1.3.

After point cloud is converted into image, the task is to classify each pixel either it is belong to ground or non-ground. This task is very similar to dense semantic labeling on pixel-wise classification hence FCN is proposed to tackle the classification.

Another issue that has been improved is the proposed method uses more features for point-to-image conversion while the previous research only uses height difference feature. In a steep terrain, height difference feature is not reliable as the assumption that ground point always lower than the surrounding non-ground point is not always true anymore. The proposed method uses intensity and return number information which is invariant to terrain slope to improve the point-to-image conversion. However, it should be noted that even though Hu and Yuan (2016) only use a height difference feature, the feature keeps the spatial pattern of the surrounding. The feature also calculates the height differences between the point to the maximum, minimum and mean of the neighbors. Although the proposed method also calculates the height difference, but it has lower information level because it only compares elevation of the point to the elevation of lowest point in the neighborhood without keeping the spatial pattern in the neighborhood.

The first proposed method using FCN is described in the following sections. Section 3.1.1 explains the point-to-image conversion, section 3.1.2 explains the proposed FCN architecture, and section 3.1.3 describes the labels transfer for dense points labeling. In addition, the adaptation of the proposed method for multi-class classification is explained in section 3.1.4.

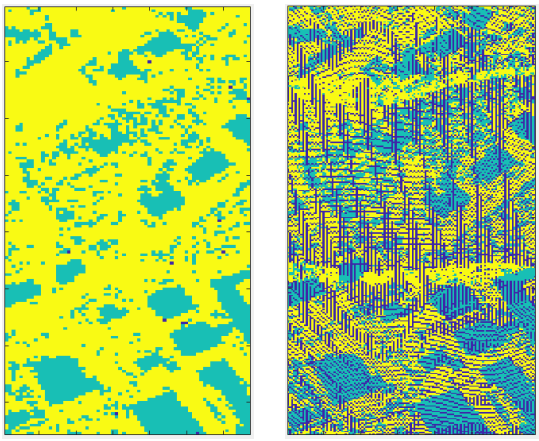
3.1.1. Point-to-image conversion

First task of this method is converting point cloud into image. LIDAR point cloud has at least three original features that can be used for the conversion: elevation (Z), intensity (I) and return number (R). The return number is counted when the laser pulse hits, reflects and penetrates through objects; the first return number is the first reflection, second return number is the second reflection and so on. Not to be confused with the number of returns, that records the total number in a single pulse if the pulse is reflected in several objects.

Furthermore, additional feature was added to help the classification. Height difference (ΔH) between point and the lowest point in the neighborhood is always used as an important feature to separate ground and non-ground point in many point cloud classification using machine learning technique (Chehata et al., 2009; Lu et al., 2009; Mallet et al., 2008; and Niemeyer et al., 2012). This feature gives near-zero value for ground point and high value for non-ground point. However, this assumption is only true on a flat terrain. In this method, the neighbors are defined as all surrounding points lie within 20 x 20 m horizontal rectangle with respect to the corresponding point. For *i*-th point on the point cloud, ΔH is defined as

$$\Delta H_i = Z_i - min(Z_{neighbors\ of\ i}) \quad (13)$$

Pixel size is defined as 1 x 1 m on the conversion. It is based on the assumption that modern LIDAR point cloud always has at least one point per square meter. Larger pixel size avoids an empty pixel but extracts a less detailed image. On the other hand, smaller pixel size creates more detailed image. Indeed, high point density point cloud is more reasonable to be converted into a higher resolution image. The losing information during conversion could be minimized as well. Ideally, the pixel size is chosen so that every point is represented in the image. This can be done by setting the pixel size equal to the point spacing of the point cloud. But this could add a consequence due to the irregularity pattern of point cloud. If the pixel is chosen close to the point spacing, then every point is more likely to be represented in the image but more empty pixels appear. Figure 16 shows that the empty pixels increase when the pixel size is changed into a smaller value.



a b

Figure 16. The difference of (a) 1 m pixel size and (b) 0.5 m pixel size.
Dark blue pixels show the empty pixels.

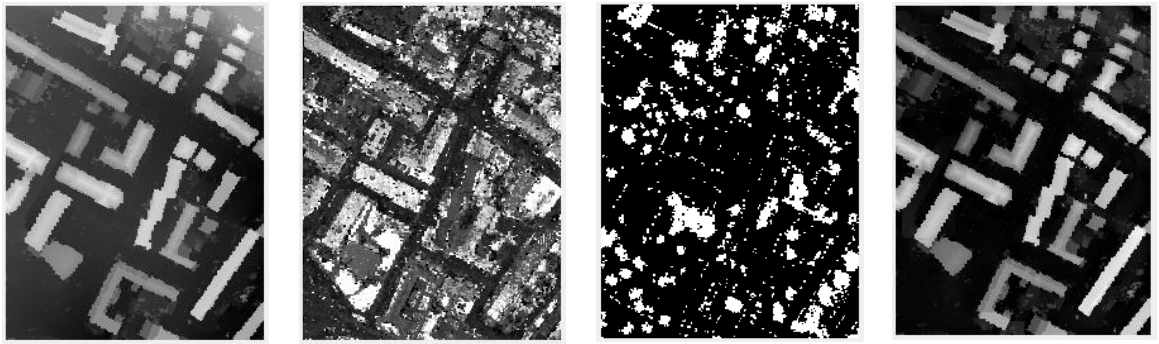
Another issue related to the pixel size is the network should learn on a larger contextual area if the pixel size is set very small in order to accommodate a high point density point cloud. Gevaert et al. (2018) mentioned this issue when dealing with the very high resolution UAV data. The filter in the network should be large enough to capture the largest object in the scene. Such filter would be very large in a case

of UAV data. Even though this could be handled by considering dilated filter in the network (Gevaert et al., 2018), the pixel size must be chosen carefully before designing the network.

The main problem of the conversion is one pixel can only represent one point; consequently any other point information within the same pixel will be lost. Suppose the point cloud has 10 points per square meter, 1 x 1 m pixel size conversion only has a chance to capture one point to be represented in the pixel value and ignores the rest nine points. This problem cannot be avoided due to the nature of 3D to 2D projection.

In that case, it needs to be decided which point is selected to represent the pixel value. Since this study focus on ground and non-ground classification, lowest points within a pixel is chosen if there are more than one point in a pixel. In that way, image is a representation of lowest points in every pixel. This approach was chosen because the classification task is to separate ground point from non-ground point. In most types of terrain, the lowest point of a certain area is more likely to be the ground than the upper point. Using that assumption, it is reasonable to select the lowest points in a pixel to represent the pixel values. After all lowest points is labeled, the upper points will be evaluated either belong to ground or non-ground. Section 3.1.3 describes the procedure of that task.

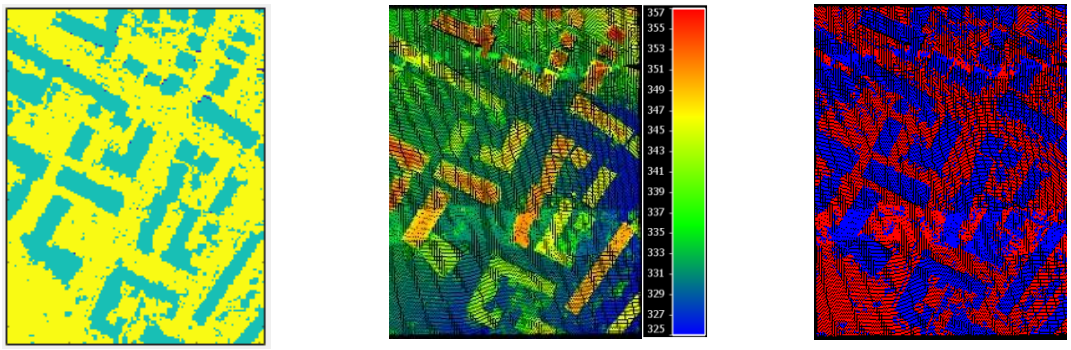
If every feature becomes one channel of the converted image, four-channel image is obtained after conversion. Figure 17 shows the converted point cloud into image in four different channels. The images are normalized in range [0,1] to have the same weight among all channels when the images is consumed by FCN. Hence each feature contributes proportionally in the network. Otherwise, one feature could be more dominant than the other features. Normalization was done by taking the maximum and minimum value in the image and normalized all values with respect to the maximum and minimum.



a. b. c. d.

Figure 17. Converted image in (a) elevation, (b) intensity, (c) return number and (d) height difference feature

After four channels image has been created, a corresponding ground truth label image was also created containing labels for each pixel. Label is derived from the lowest points in a pixel if there are two or more points in a pixel. Figure 18 shows ground truth image from the corresponding point cloud.



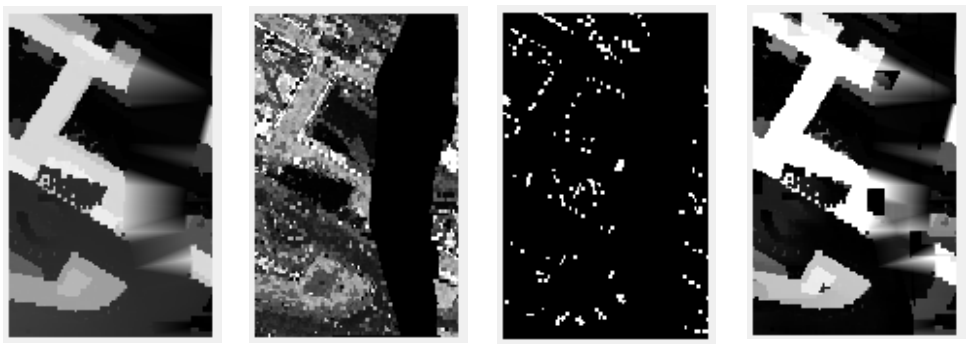
a. Ground truth label image; Yellow: Ground, Green: Non-ground

b. Point clouds in height color coded

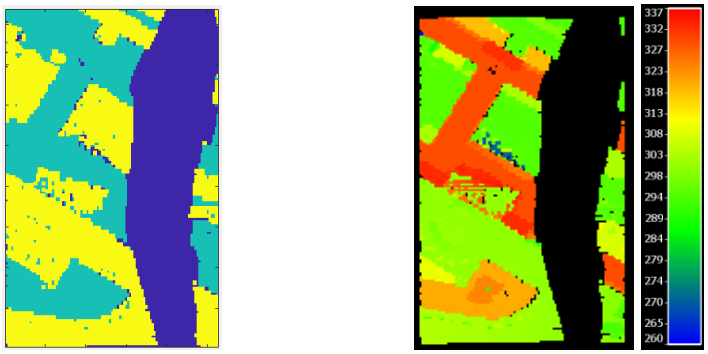
c. Labeled point clouds; Red: Ground, Blue: Non-ground

Figure 18. (a) Corresponding label image, (b) original point clouds in height color coded and (c) corresponding label for point clouds

In a case of image has empty pixels, an interpolation is used to fill in these empty pixels for Z and ΔH images while 0 is used for I and R images. Another label is added to fill in these empty pixels. In this study, label 2 is given for the ground pixel, label 1 for the non-ground pixel and label 0 for the empty pixel. Label 0 is needed only for image creation but it is not considered as a label in the training phase. Hence the predicted image from the classification only has two labels; ground and non-ground. Figure 19 shows how images are obtained if there are data gaps in the area.



a. Converted image in Z, I, R and ΔH features



b. Ground truth label image; Yellow: Ground, Green: Non-ground, Blue: Empty pixels

c. Corresponding point clouds in height color coded

Figure 19. Extracted images in an area with many empty pixels caused by a big gap in the point cloud data

3.1.2. FCN architecture and training

The next task is to train FCN using the extracted image and the corresponding label image. The proposed architecture adapts no down-sampling FCN used by Persello and Stein (2017) rather than the deconvolution FCN architecture by Long et al. (2017). The reason of this choice is to avoid up-sampling interpolation as used in deconvolution network.

In order to construct no down-sampling FCN architecture, stride is set equal to one for all filters. That means the filter moves to the next pixel without any gaps when the filters convolve in the entire image. Another parameter is pad size. The pad size is set to (filter size – 1) / 2 to keep every pixel on the boundary of the image are kept. As a result, the final layer of the network has exactly the same width and height size as the input layer.

The proposed FCN architecture for ground classification is shown in Figure 20 below. It is a variation of deep FCN_DK network proposed by Persello and Stein (2017). The architecture consists of four dilated convolutional layers (DConv) without the pooling layer and one final convolutional layer (Conv). The dilation factor increases from one in the first layer to five in the fourth layer. ReLU and batch normalization layers follow each convolutional layer. Finally, dropout and softmax layer are added in the final layer.

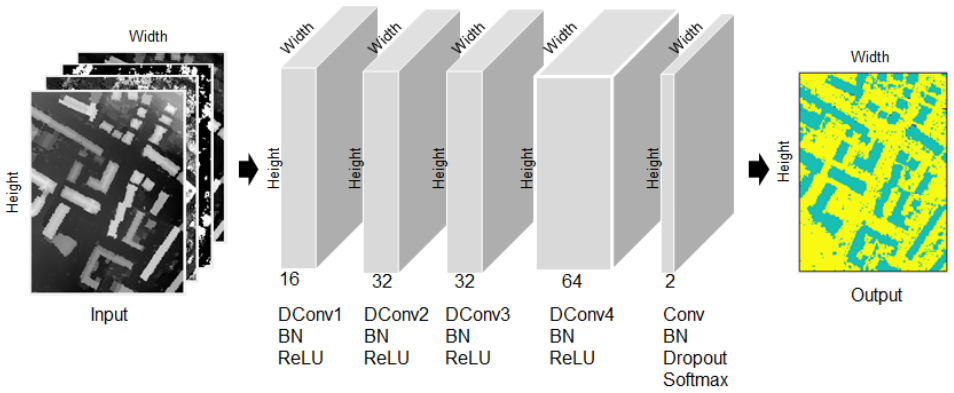


Figure 20. Proposed FCN architecture for ground classification

It can be seen that for arbitrary width and height of the input image, the output size remains the same for every layer. The depth of the input layer corresponds to the number of input image channels while the depth of the final layer is two due to the binary classification.

Dilated convolution was used in order to have a larger receptive field without significantly increase of the number of weights. If the filter size is small, the use of dilated convolutional is very important in a no down-sampling network. Otherwise the network is not able to learn the contextual information in a large spatial extent. Deep FCN_DK network was successfully captured larger contextual information by increasing the receptive field on the deeper layer using dilated convolutional filter. Similar architecture is used in the proposed method. It is important to have large receptive fields for ground classification to capture the largest building in the area.

Table 2 shows how the receptive field increases in the next layer by increasing the dilation factor from one, two, four and five in the first layer until fourth layer respectively. The filter size is 5x5 in the first two layers and 7x7 in the third and fourth layer. The reason of that choice is to have a smaller receptive field in

the earlier layers and, together with the increasing of dilation factor, to have a larger receptive field in the following layers. The number of filters also increases in order to detect more spatial patterns in the larger receptive fields.

Layer	Filter size (pixels)	# filters	Dilation factor	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 2. Receptive field size of the proposed method

After point clouds were converted into images, image patches were generated in order to train the network. The size of this image patch was defined as 105 x 105 pixels. Since pixel size is 1 x 1 m, image patch represents 105 x 105 m of area. This size was chosen with the consideration that the size covers both ground and non-ground object in a typical area where there are ground and non-ground objects in the area. If the size is too small, there is a possibility that the image patch only covers either ground (e.g. in a field) or non-ground objects (e.g. in a large building). In that case, the network cannot learn to discriminate ground from non-ground. If the size is too large, the computation time is too slow.

The training was done by feeding the network with the image patches. The network learns by stochastic gradient descent with momentum. The momentum is 0.9 and the weight decay is 0.0005. Every mini-batch has 32 samples. The dropout in the final layer has rate of 0.5. All parameters are randomized in the first epoch. Then the network was trained for 50 epochs with learning rate of 0.0001. MatConvNet framework was used for the implementation of the network.

3.1.3. Labels transfer from pixels to points

The classification result from FCN is ground and non-ground-labeled pixels whereas what is needed is the labeled point. In order to do so, the labels are transferred to the points. As mentioned in Section 3.1.1., point-to-image conversion only considers the lowest point if there is more than one point within a pixel. According to that rule, labels from pixels are assigned to the lowest points and additional steps are needed to treat the upper points. Figure 21 shows the procedure of the additional steps.

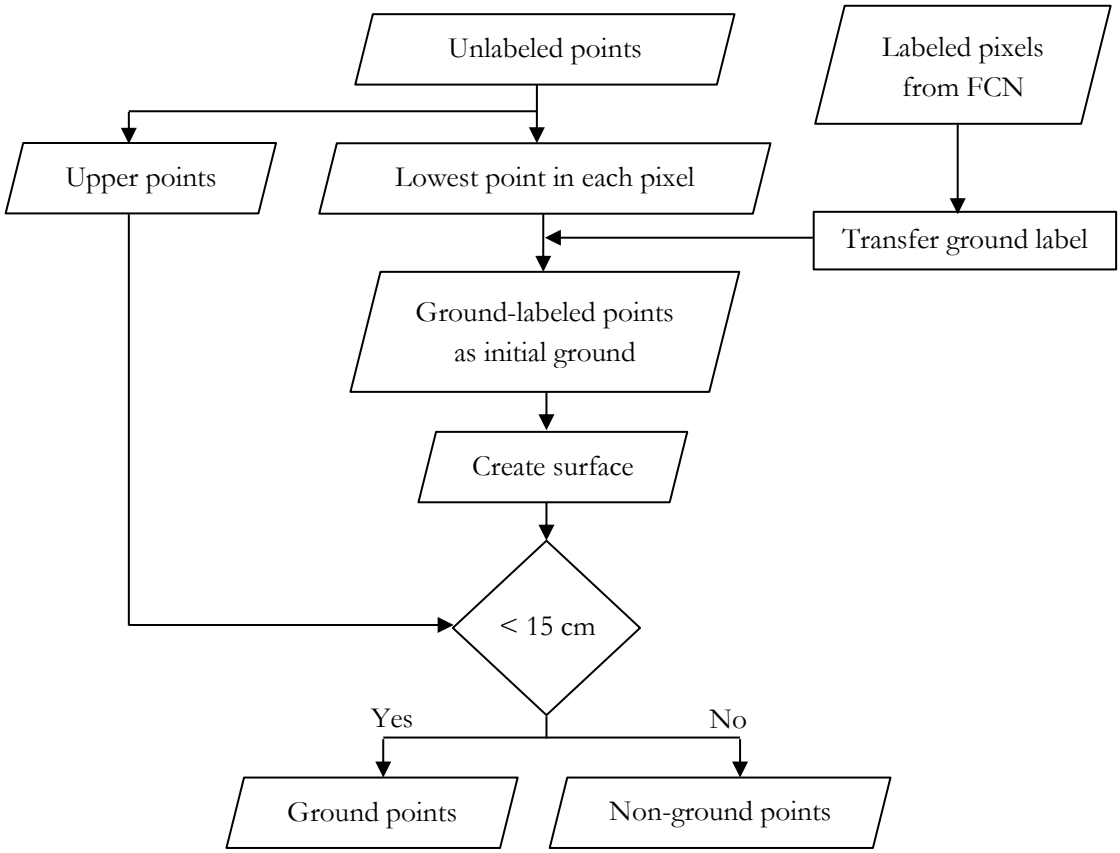


Figure 21. Procedure of transferring labels from pixels to points

The ground-labeled points from the lowest points are taken to create initial ground points. Next, a surface was created connecting all initial ground points. Then all points were labeled as ground point if their elevations are within a threshold to the surface. In this study, the threshold was set to 15 cm based on the typical vertical accuracy of LIDAR point cloud.

However, this approach has a consequence. If the ground surface has a local variation within 1x1 m and the variation is more than 15 cm, then the surface cannot capture that local variation. Another approach is by creating a segment on which the lowest point lies. Then all the points in the segment are labeled as ground point. Although the latter approach could solve the problem, it would add segmentation as a post-processing step; therefore this study used the first approach due to its simplicity.

3.1.4. Multi-class classification procedure

The next question after ground and non-ground labels were obtained is how to adapt the propose method using FCN to add more classes to the classification. Common procedure of multi-class point cloud classification is by extracting features for each point and train a machine learning classifier to give label for each point.

In the context of FCN for point cloud classification, it could be conducted by simply adding more classes to the ground truth label images. Afterwards, the labels of each pixel can be transferred to all points within a pixel. But this simple approach raised a problem because points in one pixel could have different labels, such as the lower points are ground whereas the upper points are vegetation. The problem arises because 3D point cloud is projected into 2D image. But there is still a solution in order to separate labels for points in a pixel.

The idea is to extend the result of the ground classification. After the ground classification provides ground and non-ground labels, points that belong to non-ground class are further classified into finer classes. In this study, building and vegetation classes are considered. AHN dataset is used instead of ISPRS dataset due to the availability of building and vegetation labels in the dataset. In general, two sequential classifications are needed for the task. The first classification is ground or non-ground. The second classification is multi-class classification by labeling non-ground points into building and vegetation. Detailed procedure is shown in Figure 22 below.

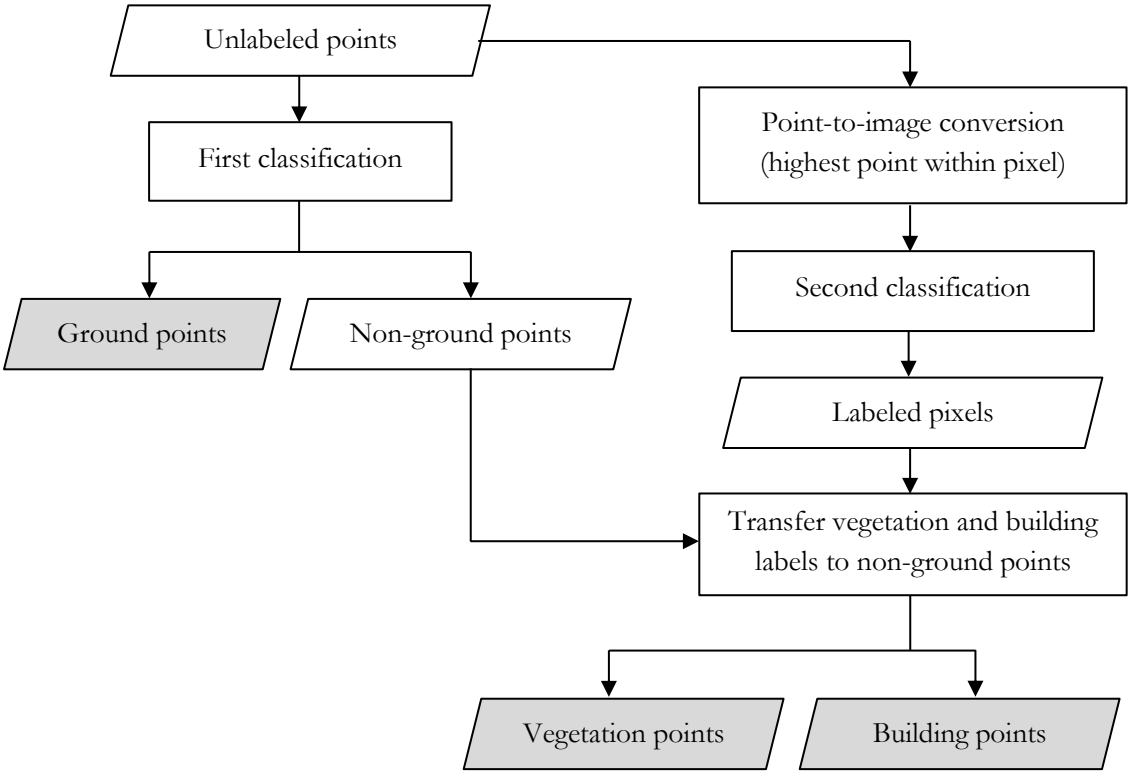


Figure 22. Procedure of adding vegetation and building classes

In order to separate non-ground points into buildings and vegetation, point cloud is converted into image using the highest point within pixel instead of the lowest point as done for the ground classification. The reason is the patterns between vegetation and buildings are clearer if the point cloud is converted using the highest points instead of the lowest points. Next, building and vegetation labels are added in the ground truth label image for training purpose. At this second classification, labels are 0 for empty pixel, 1 for vegetation, 2 for ground and 3 for building.

The result of the second classification is labeled pixels. Then the labels are transferred to points but only to the non-ground points from the result of the ground classification. The labels are transferred in a simple way. For all points within a pixel, the labels are given the same as the corresponding pixel. In the end, all points are labeled into ground, vegetation and building.

In other words, it can be said that the labels from the second classification are attached to the highest points since the conversion only takes the highest point in a pixel. The labels of the rest lower points only follow the labels from the highest points.

Using this approach, it is known that it does not consider the situation if non-ground points in the same pixel have different labels. For example, if the upper point is tree whereas the lower point is roof, then points on the roof will be labeled as tree because the conversion uses the highest point. However, in the other situation where trees and building roofs are clearly separated, this approach could work.

3.2. Multi-Layer Perceptron

In contrast to the proposed method, the alternative method classifies each point directly without needs of point-to-image conversion or any additional steps. It is a combination between feature engineering as used in many machine learning technique with feature learning as used in the deep learning. Multi layer perceptron (MLP) is chosen for the network architecture. Since Weinmann et al. (2015) used MLP as a method for point cloud classification, their method was used as a baseline for the alternative method. Then some modifications were made to fit the method for ground classification task in terms of the features set and the MLP architecture in order to tune the network so that the network is reliable for ground classification task.

3.2.1. Features extraction

Hand-crafted features were extracted manually for each LIDAR point. At the beginning, 21 geometric features from Weinmann et al. (2015) were used as initial features. The features are:

- Eight eigenvalue-based 3D features to describe local shape
- Six 3D features to describe the geometric properties
- Seven 2D features combine the 2D local shape and 2D geometric properties

Three eigenvalues ($\lambda_1 > \lambda_2 > \lambda_3$) were computed for each point from the covariance matrix of k-Nearest-Neighbors (kNN). The number of neighbors was set to 10. The features are linearity (L_λ), planarity (P_λ), scattering (S_λ), omnivariance (O_λ), anisotropy (A_λ), eigenentropy (E_λ), sum of eigenvalue (Σ_λ), and change of curvature (C_λ). Each feature is defined in the following equations.

$$L_\lambda = \frac{\lambda_1 - \lambda_2}{\lambda_1} \quad (14)$$

$$P_\lambda = \frac{\lambda_2 - \lambda_3}{\lambda_1} \quad (15)$$

$$S_\lambda = \frac{\lambda_3}{\lambda_1} \quad (16)$$

$$O_\lambda = \sqrt[3]{\lambda_1 \lambda_2 \lambda_3} \quad (17)$$

$$A_\lambda = \frac{\lambda_1 - \lambda_3}{\lambda_1} \quad (18)$$

$$E_\lambda = -\sum_{i=1}^{3} \lambda_1 \, ln(\lambda_1) \quad (19)$$

$$\Sigma_\lambda = \lambda_1 + \lambda_2 + \lambda_3 \quad (20)$$

$$C_\lambda = \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \quad (21)$$

Six 3D geometric properties features are Z value, radius of kNN, density of kNN, verticality, height difference between the highest and lowest point in the kNN, and standard deviation of Z value of kNN. Seven 2D features are radius of kNN in 2D, density of kNN in 2D, sum of eigenvalues in 2D, ratio of 2D eigenvalues, frequency of points falling in 2D bin, the height difference in a 2D bin, and standard deviation of points in 2D bin.

The 21 features were used to train MLP network in the first experiment. Next, non-geometric features were added because the features from Weinmann et al. (2015) only contain local geometric information. Those features are intensity and return number value.

Lastly, height difference between point and the lowest point in the neighborhood was also used as a feature. This feature separates ground from non-ground point easily in a flat terrain. Similar to Chehata et al., (2009); Lu et al. (2009); Mallet et al. (2008); and Niemeyer et al. (2012), the neighbors are defined as a large cylinder. The radius is set to 10 m. In addition, one additional height difference was added. It has smaller radius (1 m) in order to capture local height variation.

In total, there are 25 extracted features for each point. The selection of features was carried out to use only the relevant features for the ground classification task. According to the experiments, 10 features are selected among all features. The 10 features are two height differences, intensity, return number and six 3D eigen-based features (L_λ, P_λ, S_λ, O_λ, A_λ, C_λ) The detailed experiments are described in section 4.2.2.

3.2.2. MLP architecture and training

The idea of the alternative method using MLP is to classify point cloud from the extracted features. If the features are relevant, those features are able to separate point cloud into ground and non-ground classes. Once the features are selected to form a feature vector, MLP is chosen for the classifier. The first task is designing the MLP architecture, in terms of the number of hidden layers and units. Then the next task is training the network by feeding the network with the feature vectors of the training samples.

MLP architecture with one hidden layer containing 11 hidden units as used by Weinmann et al. (2015) was used as a baseline. Then adding more hidden layer was conducted to create a deeper network. In a pattern recognition task, the deeper network means that the network is more likely to capture more difficult patterns if the network is fed with the sufficient training samples. The proposed architecture has three hidden layers; each layer has 11 hidden units. The reason of stacking more than one hidden layer is to capture simple patterns in the beginning which are combined in the following layer. Figure 23 shows the architecture of the proposed MLP.

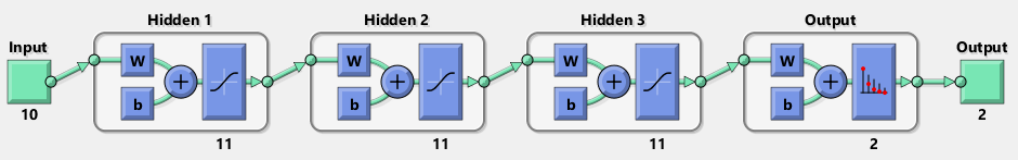


Figure 23. A proposed MLP architecture

The training phase was done by feeding feature vectors of each point to the network. The input was forward-passed through all layers which contain learnable parameters. At the beginning, all parameters in the network are randomized. Then the probabilities are calculated in the final layer by softmax function. As can be seen in the Figure 23, the output has two units to accommodate probabilities of two classes.

The loss is calculated using the cross entropy to show the negative log-likelihood between the predicted labels and the true labels from the training samples. The purpose of the training is to have predicted labels as close as possible to the true labels which is done by learning all parameters with gradient descent. The training was stopped after the error of the validation samples is larger than the error of the training samples that indicates an overfitting. After all parameters were learned, the network is reliable to classify the testing samples by forward-passing the feature vectors in the network.

4. DATASETS AND EXPERIMENTS

4.1. Datasets

4.1.1. ISPRS Dataset

The first dataset is provided by ISPRS and available online on http://www.itc.nl/isprswgIII-3/filtertest/. The dataset has been prepared to compare the performance of various filtering algorithms on extracting DEMs from LIDAR point cloud. Although this contest has been done in 2002 and the dataset seems quite outdated in terms of point density and the return numbers, this dataset is still relevant to test the performance of filtering algorithm due to the various terrain characteristics offered. The dataset itself also contains low point and high point outliers to challenge the algorithms.

The dataset was delivered in TXT format and contains two types of data; namely training and testing. Training data consists of 15 different sites with its specific characteristic as described in the Table 3 below. Points are structured in four columns; the first three columns are X, Y, and Z coordinate while the fourth column is label. The label is either ground (0) or non-ground (1).

No	Site	Characteristic	Outliers
1	Samp11	Steep slope	V
2	Samp12	Flat	V
3	Samp21	Bridge	
4	Samp22	Common	
5	Samp23	Complex	V
6	Samp24	Ramp	
7	Samp31	Outlier	V
8	Samp41	Data gap	V
9	Samp42	Railway	
10	Samp51	Steep slope	
11	Samp52	Slope and low vegetation	
12	Samp53	Break-lines	
13	Samp54	Village	V
14	Samp61	Embankment	
15	Samp71	Bridge	

Table 3. ISPRS training dataset description

Unlike the training data, each point in the testing data does not have a label. Points are structured in eight columns; four first columns for XYZ and intensity value for the first return and four last columns for XYZ and intensity for the last return. The testing data consists of 4 city areas and 4 forest areas.

Even though there are two types of data, the comparison study of the various filtering algorithms (Sithole and Vosselman, 2004) has been done only on the training data. Recent deep learning algorithm for ground classification (Hu and Yuan, 2016) was also tested only on the training data. In order to be able to calculate the accuracy and to have a comparable result with the previous researches, the proposed

methods in this study were tested only on the training data. Furthermore, the proposed methods in this study used supervised classification and needed labeled training sample. The condition makes only training data is available for both training and testing purposes.

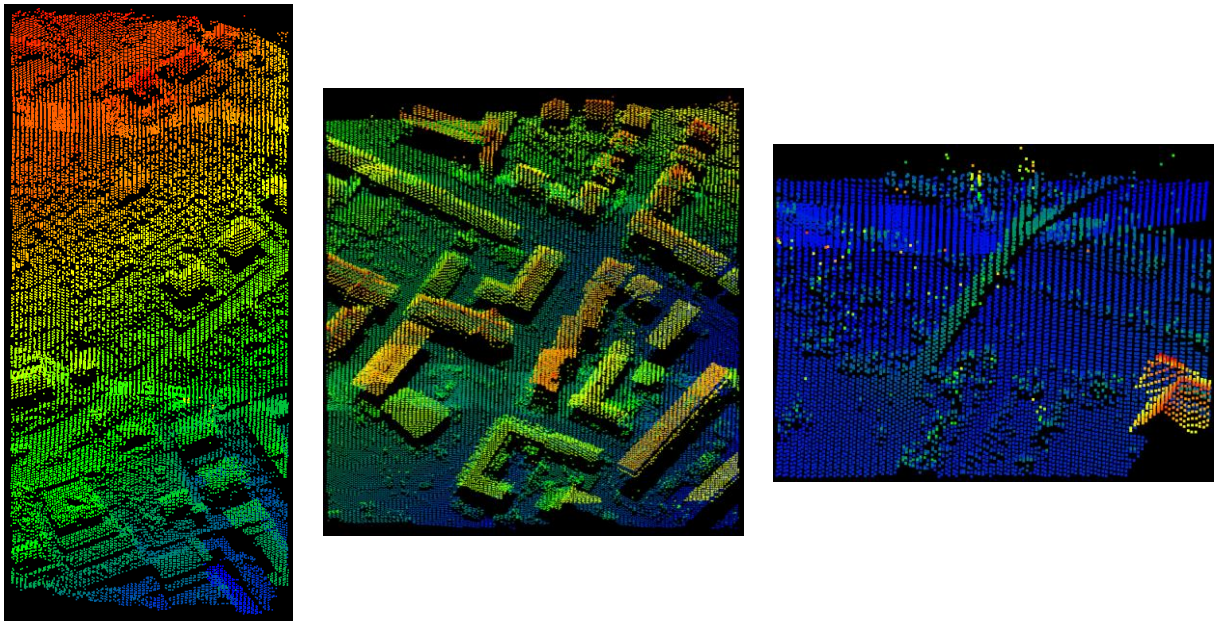
As mentioned in section 3.1.1 and 3.2.1, intensity value and return number is used as features but the training data does not have that information. Fortunately, each training data is a subset from one of the testing data. In that situation, combining training data and testing data makes it possible to have a list of points in the following order: XYZ, intensity value, return number and label. As a result, 15 sample sites from the training data have additional intensity value and return number information. To avoid confusion, the term ISPRS dataset on the following sections refers to those 15 sites.

The 15 sites of ISPRS dataset were divided into three categories. 10 sites were selected for training and five for testing. Due to the limited number of data, the validation step is performed on the 10 training sites to fine-tune the hyper-parameters. Furthermore, the validation use 2-fold cross validation in order to have accuracies for all training sites. 2-fold cross validation means that the training samples are divided into two folds. Each fold has the same number of samples. For ISPRS dataset, each fold has five sites. Then training and testing are performed twice. At first, training uses training samples from the first fold and testing samples from the second fold. Then at second experiment, training uses samples from the second fold and testing uses samples from the first fold. In that manner, each site of 10 ISPRS dataset has a role as training and testing sample, and the classification can be performed to those 10 sites. Table 4 shows the partition between training and testing samples. After the validation step, all 10 training sites were used for training to test the five testing sites.

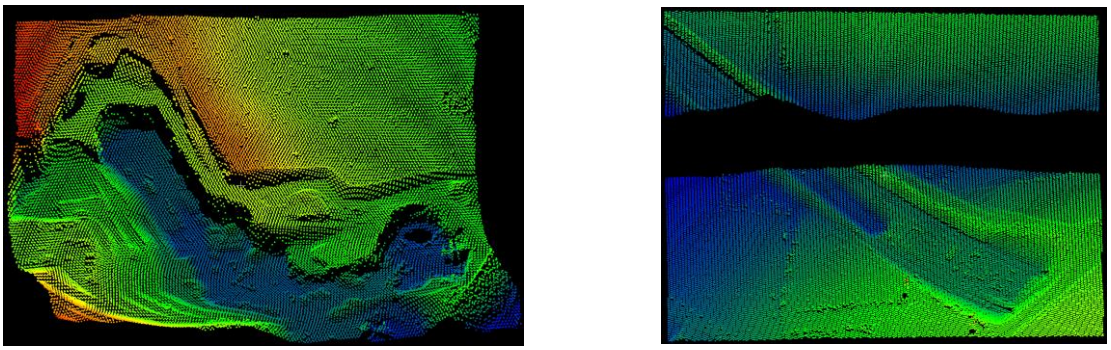
No	Site number	Category		
		Testing	Training	
			1st fold	2nd fold
1	Samp11	X		
2	Samp12	X		
3	Samp21	X		
4	Samp22		X	
5	Samp23		X	
6	Samp24		X	
7	Samp31			X
8	Samp41			X
9	Samp42			X
10	Samp51		X	
11	Samp52			X
12	Samp53	X		
13	Samp54		X	
14	Samp61	X		
15	Samp71			X

Table 4. Training and testing samples in 2-fold cross validation

Figure 24 shows the five ISPRS testing sites in height color coded. It can be seen that the testing sites are varying in terms of the terrain and scene characteristic.



a. Samp11 b. Samp12 c. Samp21



d. Samp53 e. Samp61

Figure 24. ISPRS testing sites on height color coded

The ISPRS dataset has limited number of training sample hence data augmentation was done in order to collect more training samples. As mentioned in section 3.1.2, training process uses image patches as an input data for the network. In order to have more training samples, three more patches are created by rotating the image patch by 90 degree increment for each patch. The motivation is to have a larger variety of angles in training samples so that the trained network is reliable to predict testing sample with any angles of the slope. For instance, the slope angle in the training sites (Samp51 and Samp52) only faces West direction while the slope angle in the testing sites (Samp11) faces South direction.

300 patches are originally extracted from each ISPRS training site. Then each patch is replicated three times in different angles. In total 12,000 patches are obtained to train the network from the 10 ISPRS training sites.

4.1.2. AHN Dataset

Actueel Hoogtebestand Nederland (AHN) dataset covers entire area of The Netherlands. It is provided as an open data in Publieke Dienstverlening op de Kaart (PDOK) portal and can be downloaded at https://www.pdok.nl/nl/ahn3-downloads. AHN dataset has 3 different series of acquisition years, namely AHN1, AHN2 and AHN3. The latest AHN3 is used in this study. AHN3 dataset is delivered in LAS file and has already labeled according to the format from ASPRS (American Society of Photogrammetry and Remote Sensing) as shown on Table 6 below.

Class	ASPRS format	AHN format
0	Created, never classified	-
1	Unclassified	Vegetation
2	Ground	Ground
3	Low vegetation	-
4	Medium vegetation	-
5	High vegetation	-
6	Building	Building
7	Low point (noise)	-
8	Model key point (noise)	-
9	Water	Water
26	Reserved	Bridge

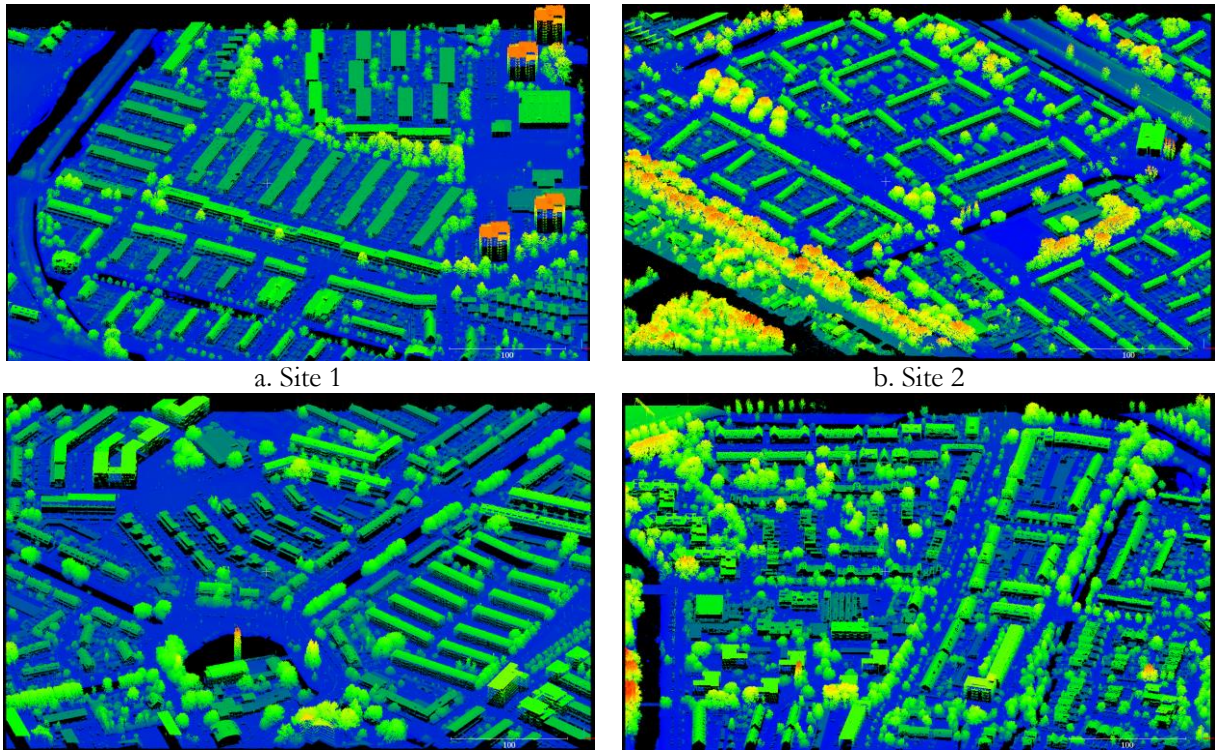
Table 5. Classification format of AHN dataset

However, there is little difference in the class number for vegetation. ASPRS format assigns label 3 to 5 for low, medium and high vegetation while AHN dataset assigns label 1 for vegetation (in fact, it is not only vegetation but also all objects excepts buildings, water and bridge which are not defined in the ASPRS format such as cars, poles, etc.). Since ground classification only focuses on ground and non-ground classes, class 1, 6, 9 and 26 are merged into class 1 as non-ground class while class 2 is kept as ground class.

In general, AHN3 dataset has different characteristics with ISPRS dataset. It offers more than 20 points per square meter in contrast to ISPRS dataset which offers only 1 point per square meter (or even less in some samples). The terrains are relatively flat and no outliers on the data. Since the point density is high, the pixel size could be chosen at a smaller pixel. But the pixel size is kept at 1x1 m in this study because if the pixel size is reduced, let say into 0.5 x 0.5 m, the filter in the network should be twice as large as the filter size for 1x1 m pixel size in order to have the same receptive field size.

16 sites with area of 500x500 m were used. Nine sites are for training samples, two sites for validation and four sites for testing. Since the terrain is relatively flat and the scenes are heterogenic, it was not necessary to augment the data. 300 patches were extracted for each sites, hence 2,700 patches were obtained in total.

Figure 25 shows the AHN testing sites in height color coded while Figure 26 shows the Z images of the nine training sites.



a. Site 1 b. Site 2

c. Site 3 d. Site 4

Figure 25. AHN testing sites

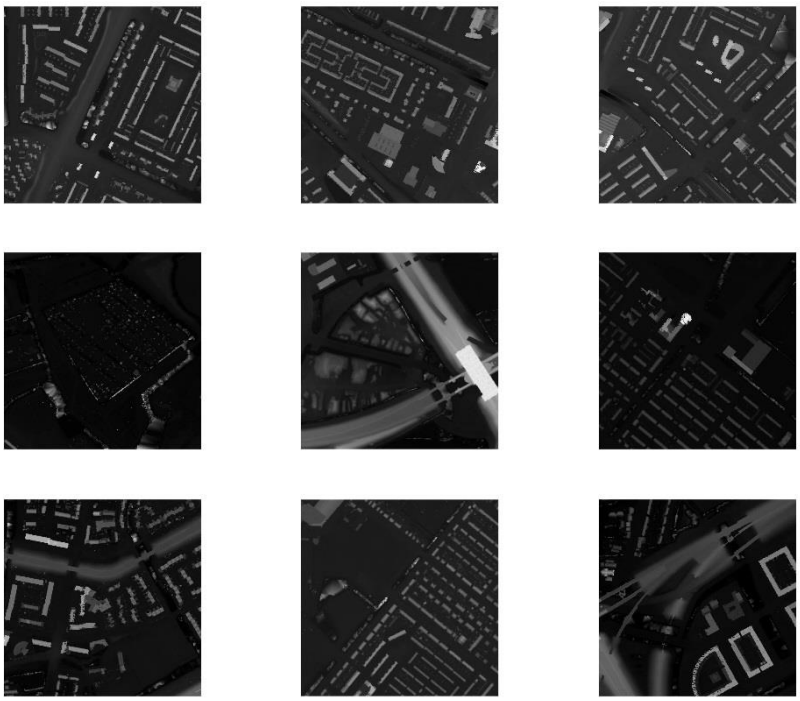


Figure 26. Nine training sites of AHN dataset

4.2. Sensitivity analysis

4.2.1. Hyper-parameters tuning

Hyper-parameters tuning on FCN

Besides the learnable parameters, there were other parameters which were not trained during training process but defined to construct the architecture. Those are called hyper-parameters. Different hyper-parameters configurations were tested to find the most optimum configuration for ground classification purpose. In order to do that, all samples were divided into training, validation and testing samples. Training samples were used to train the network, validation samples were used to fine-tune the hyper-parameters configurations and testing samples are used to test the accuracy from the fine-tuned network. In a validation step, different number of convolutional layers and the use of max-pooling layer were tested.

The number of convolutional layers

The number of convolutional layers may be varying in CNN or FCN architecture. However, Simonyan and Zisserman (2014) proved that CNN with deeper network gives better accuracy. Furthermore, Sherrah (2016) and Persello and Stein (2017) also proved that deeper network gives better result on FCN architecture for the considered problem and dataset.

In this study, three different configurations were tested. Adapting a network from Gevaert et al. (2018), two dilated convolutional layers were used on the first configuration. The first convolutional layer has receptive field size of 5 x 5 pixels and the receptive field is increased significantly on the second convolutional layer into 33 x 33 pixels by employing dilated convolutional factor of four.

Then, following the success of deep FCN_DK network from Persello and Stein (2017), deeper networks were used on the next experiment. In the second configuration, one convolutional layer with two dilated convolutional factor was added between the first and the second convolutional layer from configuration one. After adding one convolutional layer, the receptive field size in the final layer was increased from 37x37 pixels into 49x49 pixels. In the third configuration, the receptive field size was increased drastically almost twice as the receptive field size from configuration two. This was achieved by adding one convolutional layer with dilated convolution factor of five. Detailed architecture from the three configurations can be seen on Table 6.

Layer	Configuration											
	1				2				3			
	Filter size	Dilated	RF size	# filters	Filter size	Dilated	RF size	# filters	Filter size	Dilated	RF size	# filters
Conv1	5 x 5	1	5 x 5	16	5 x 5	1	5 x 5	16	5 x 5	1	5 x 5	16
Max-pooling	5 x 5	-	9 x 9	-	5 x 5	-	5 x 5	-	5 x 5	-	5 x 5	-
Conv2	7 x 7	4	33 x 33	32	5 x 5	2	17 x 17	32	5 x 5	2	17 x 17	32
Max-pooling	5 x 5	-	37 x 37	-	5 x 5	-	21 x 21	-	5 x 5	-	21 x 21	-
Conv3	1 x 1	1	37 x 37	2	7 x 7	4	45 x 45	32	7 x 7	4	45 x 45	32
Max-pooling					5 x 5	-	49 x 49	-	5 x 5	-	49 x 49	-
Conv4					1 x 1	1	49 x 49	2	7 x 7	5	79 x 79	64
Max-pooling									5 x 5	-	83 x 83	-
Conv5									1 x 1	1	83 x 83	2

Table 6. Different configurations in terms of the number of convolutional layers

The results from all experiments are shown in Table 7. The results are shown in error rates. Total error shows the percentage of misclassified points of both classes. Type I error indicates the percentage of ground points are misclassified into non-ground points, and type II error is the other way around.

The network gave the best result as the receptive field has the largest size. Deepest network that having largest receptive field proves to give a better result. Besides that, the deeper network also has more parameters; hence it is more likely to solve a more complex problem.

Error rates (%)	Different number of dilated convolutional layers		
	2	3	4
Total error	15.01	16.02	13.77
Type I error	7.55	10.24	7.80
Type II error	30.13	30.07	29.38

Table 7. Error rates from different number of layers

It was predictable that the network with largest receptive field achieves the lowest error rates. However, it was unpredictable when the error from three dilated convolution layers is worse than the error from two dilated convolution layers. This is not in line with the research from Persello and Stein (2017) when the accuracies always increase along with the increasing of the receptive field. But in sloped areas, the error rate reduces consistently as the receptive field size increases. Samp51 and Samp52 are two sites which have sloped terrain. The total error rates on both sites are shown in Table 8.

Sites	Total error on different number of dilated convolutional layers		
	2	3	4
Samp51	17.03 %	16.22 %	15.84 %
Samp52	16.21 %	10.92 %	8.82 %

Table 8. Total error rates on sloped terrain

Besides ISPRS dataset, three experiments were also performed on AHN dataset. The accuracies are calculated on validation sample areas. The result is shown in Table 9 below. Similar to the experiments on ISPRS dataset, the lowest total error on AHN dataset is achieved on a deepest network that having largest receptive field among the other networks. However, it must be noted that increasing receptive field size minimizes type II error, but on the other hand it increases type I error.

Error rates (%)	Different number of dilated convolutional layers		
	2	3	4
Total error	4.91	4.69	4.54
Type I error	0.27	0.41	0.51
Type II error	8.87	8.34	7.97

Table 9. Error rates on AHN validation sample

Pooling layer

Pooling layer is the second hyper-parameter which was tested during fine-tuning step. Pooling layer with stride more than one is useful on CNN architecture to reduce the output feature maps in each layer. The same schema is also seen in deconvolution FCN architecture while no down-sampling FCN architecture uses stride equal to one to maintain the size. In the fine-tuning step, two configurations were performed. First configuration has max-pooling layer with 5 x 5 pixels of filter size while the second configuration does not have pooling layer. The latter configuration is worth a try as Springenberg et al., (2014) showed that removing pooling layer makes the network simpler but its accuracy is maintained.

Two experiments were performed to analyze the impact of max-pooling layer in the network on ISPRS dataset. The results are shown in Table 10 below. It can be seen that the error rates reduces after max-pooling layer is removed.

Error rates (%)	Max-pooling layer	
	Yes	No
Total error	14.56	11.45
Type I error	5.67	7.26
Type II error	34.42	20.96

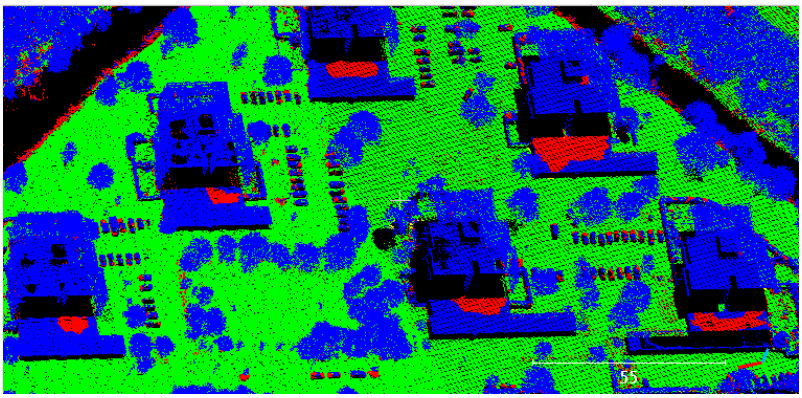
Table 10. Error rates between network that uses max-pooling layer and not

In ISPRS dataset, network without max-pooling layer gives a better result than network with pooling layer. One of the reasons is pooling layer was originally designed to reduce the dimension of output feature maps yet makes a network invariant to translation. This purpose is suitable for CNN architecture when the size of the following layer is gradually reduced. But this purpose becomes irrelevant for FCN architecture with no down-sampling as used in this study. In a ground classification task, it seems better to leave the output feature maps as an original result from the convolutional filter without summarizing them in each local area as done by max-pooling layer. In addition, the network architecture becomes simpler so the computational time is faster.

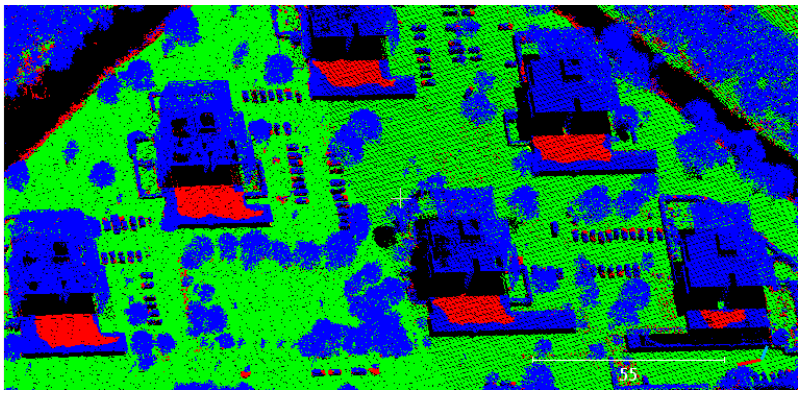
Similar experiments were conducted as well on AHN dataset. Table 11 shows the error rates from the two experiments. Unlike the result from ISPRS dataset, the error slightly increases as the max-pooling layer is removed. The possible reason is that the network without max-pooling layer has smaller receptive field (29x29) than the network with max-pooling layer (37x37). Meanwhile, some buildings in AHN dataset are considered as large buildings. Figure 27 shows the impact of involving max-pooling layer in the network to the type II error in building roofs. It is visible that the network without max-pooling cannot capture the plan and large roofs, while the network with max-pooling classifies the roofs better.

Error rates (%)	Max-pooling layer	
	Yes	No
Total error	4.65	4.91
Type I error	0.29	0.27
Type II error	8.36	8.87

Table 11. Error rates on AHN validation site



a. Result with max-pooling layer



b. Result without max-pooling layer

Figure 27. Type II errors on buildings (red points) in networks (a) with max-pooling layer and (b) without max-pooling layer

Hyper-parameters tuning on MLP

Three experiments were performed to find the optimum network architecture as mentioned in section 3.2.2. First experiment uses one hidden layer with 11 hidden units to follow the architecture from Weinmann et al. (2015). Second and third experiments each use two and three hidden layers to make the network deeper; each layer has the same 11 hidden units. The configuration for each experiment is shown in Table 12 below.

Configuration	Architecture (Input – Hidden layer – Output)
1	Input – 11 – Output
2	Input – 11 – 11 – Output
3	Input – 11 – 11 – 11 – Output

Table 12. Different configuration of MLP architectures

Table 13 shows the accuracy results on different configurations. Even though the third experiment has the highest total error, but type I error and type II error are more "balanced".

No	Network	Total error (%)	Type I error (%)	Type II error (%)
1	In – 11 – Out	10.28	10.64	17.11
2	In – 11 – 11 – Out	11.37	12.09	15.72
3	In – 11 – 11 – 11 – Out	12.39	13.59	14.52

Table 13. Accuracy assessment using different network architectures

4.2.2. Features selection

Features selection on FCN

Four features were used to convert point cloud into image. Elevation, intensity and return number are the original information from the point cloud while height difference is extracted manually from the data. Three experiments were performed to investigate the influence of the features. The first experiment used elevation image only. In the second experiment, intensity and return number were added to see the influence of non-geometric features. Finally, height difference feature was added in the third experiment to include all features.

Table 14 shows the accuracy results from all experiments. All experiments were performed on the 10 ISPRS training sites.

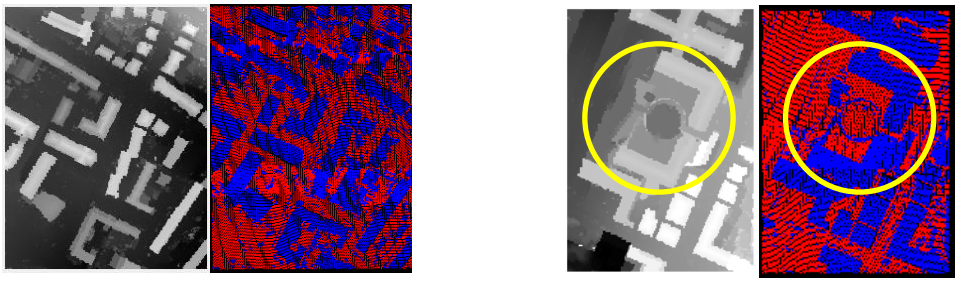
No	Features	Total error (%)	Type I error (%)	Type II error (%)
1	Z	12.48	3.22	33.32
2	Z I R	14.56	5.67	34.32
3	Z I R ΔH	13.18	8.01	22.54

Table 14. Accuracy assessment using different combinations of elevation (Z), intensity (I), return number (R), and height difference (ΔH) features

From Table 14 above, the lowest total error is achieved on the first experiment. Adding non-geometric features on the second experiment makes the accuracy worse. But the additional height difference feature in the third experiment apparently improves the result from the second experiment. Even though the first configuration gives a lowest total error, the third experiments offers more balance combination between type I error and type II error. The influences of each feature are discussed in the following sections.

Elevation (Z)

This feature has important role to discriminate ground and non-ground. In a local flat area, ground points always have lower elevation values than non-ground points. If the assumption is true, elevated object such as buildings always look brighter than the surrounding ground (Figure 28.a). Meanwhile, more difficult situation appears on the example in Figure 28.b. In this example (yellow circle) the scene creates step-like pattern and is mixed between ground surface and buildings. There are three levels of surface. While the highest surface must be a building and the lowest surface is more likely to be a ground surface, the difficult part is the surface in the middle. Since the convolutional filter learns the spatial patterns, it is not easy for the network to classify that middle surface.



a. b.

Figure 28. Examples of extracted Z image when the assumption is either (a) true or (b) not always true, and the corresponding true label (red: ground; blue: non-ground)

Intensity value (I)

When laser pulse hits object, it returns with different response according to the material of the object. This amount of energy reflected by the object is called intensity value. Highly reflective objects such as cars have higher intensity value than less reflective object such as trees. Typically, point in the ground has lower intensity value than point in the elevated object such as building because mostly ground point is a bare soil or grass that has low intensity value. But this assumption is never true in all condition. As can be seen in Figure 29, some buildings have high intensity value thus it helps to distinguish them from the surrounding ground. But, there is a building with low intensity value (yellow ellipse). The intensity values of the building are similar to the ground.

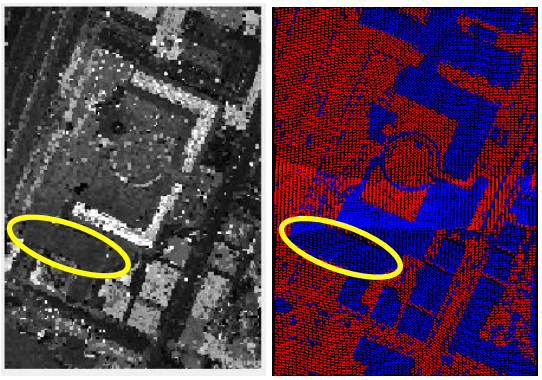
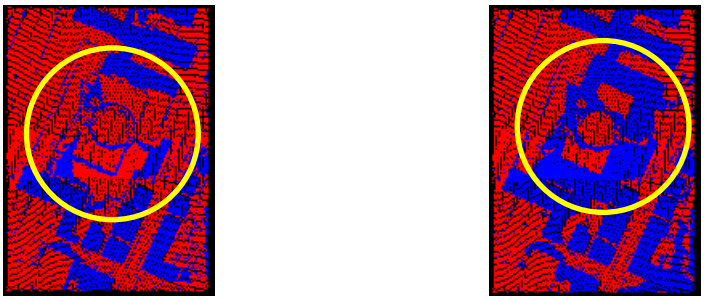


Figure 29. Intensity image and the corresponding true label (red: ground; blue: non-ground)

However, the intensity value can help the difficulty faced in Figure 28.b. If intensity image is used, it is more obvious which pixels belong to buildings and which pixels belong to ground. In the experiment, the total error of this site reduces from 15.71% into 10.06% after intensity and return number were added. The detailed classification result is shown in Figure 30. It can be seen the buildings were not correctly labeled if the classification uses elevation image only. This fact shows the benefit of employing non-geometric feature over geometric feature.



a. b.

Figure 30. Point cloud with predicted label from (a) Z image and (b) Z I R image

Return number (R)

Similar to the intensity value, return number is non-geometric features. LIDAR point cloud records the return number for each returned pulse. There is only one return in the open area or building roof while vegetation or building façades have more than one return. ISPRS dataset only offers two return numbers in contrast to AHN dataset that offers up to 5 return numbers.

Laser pulse can be reflected as single return or multiple returns. If the laser hits solid object such as roofs or roads, the laser is only reflected once and recorded as a single return. Meanwhile, laser pulse is recorded as multiple returns if the laser hits an object, records a first return, then it is partly reflected and partly penetrated. The penetrated pulse continues until it is reflected by the next object and recorded as a second return. In a context of filtering, the last returns are more likely to be the ground points. But this is not always valid especially in the dense vegetated area where laser pulses cannot reach ground. In that situation, last returns could be branches or stems.

But still, the return number is valuable information for filtering. This makes some filtering algorithms work directly only on single and last returns. Hence the return number gives a good indication for ground classification and might work with other features to help the classification.

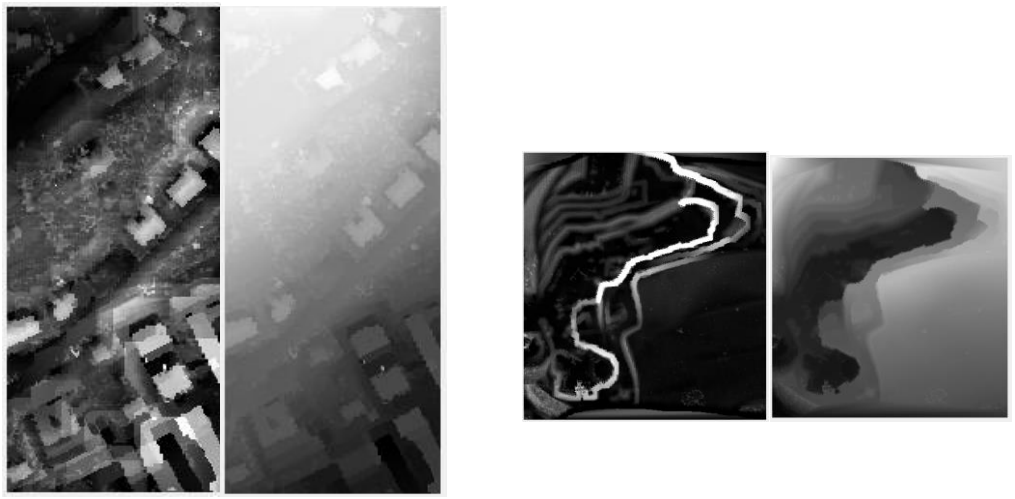
Height difference (ΔH)

In order to separate non-ground objects to the ground, the value of non-ground pixels should be high enough to the surrounding ground pixels in a local area. Even though this purpose has been already achieved using elevation image, height difference feature was still extracted to strengthen the difference between ground and non-ground pixel values. In the experiment, adding ΔH feature improved the accuracy compared to the result of Z I R feature set.

The ΔH feature was calculated by subtracting height of every point to the lowest point in the neighborhood. As mentioned in section 3.1.1., the neighborhood is defined as all surrounding points lie within 20 x 20 m horizontal rectangle. The size was chosen to capture the largest building in the area but not large enough to cut the small undulating terrain.

Figure 31 shows the converted image from the height difference feature in two sample areas and the corresponding elevation image. The feature works quite well in a sloped terrain as seen in the Figure 31.a where buildings look clearer on the ΔH image than the Z image. But in the terrain with height

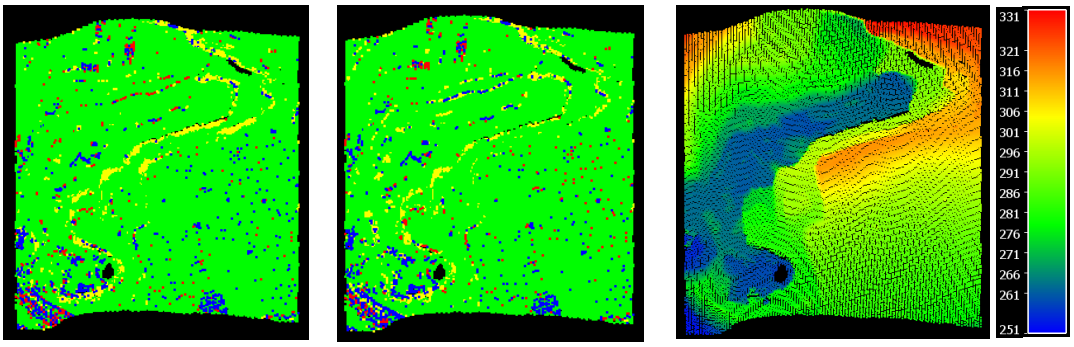
discontinuities such in Figure 31.b, ΔH feature creates artefacts along the break-lines terrain. Pixels within the artefacts could be easily misclassified as non-ground points due to the high pixel values.



a. Sloped terrain b. Break-lines terrain

Figure 31. Comparison between ΔH images (left) and Z image (right) in different terrain characteristics (a) Sloped terrain and (b) break-lines terrain

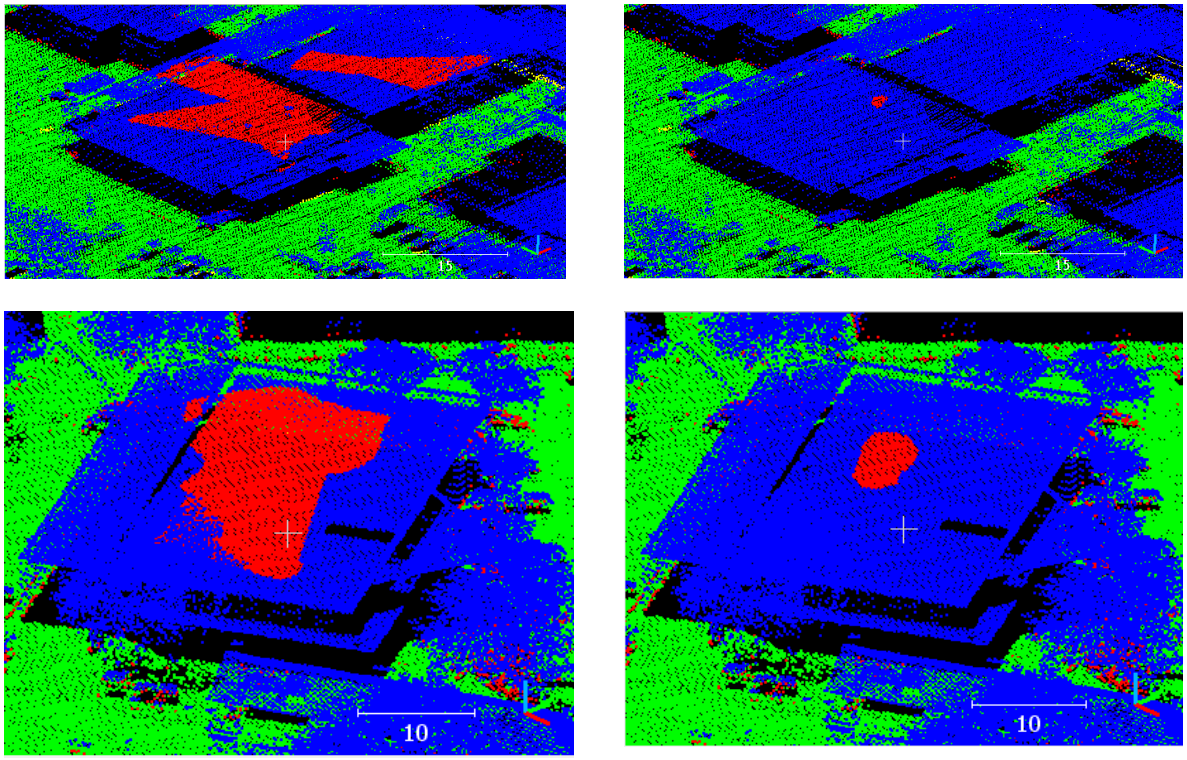
In order to see the impact of employing ΔH feature, classification was performed on a site with break-lines terrain. Figure 32.a shows labeled points from Z I R ΔH image set, while Figure 32.b shows labeled points without using ΔH images. Yellow points indicate ground points are misclassified into non-ground points. It can be seen that there are more yellow points along the break-lines from the result of the classification using ΔH image.



a. with ΔH image b. without ΔH image c. corresponding point cloud

Figure 32. Labeled points from different image set on break-lines terrain.
(Green: correctly labeled ground; Blue: correctly labeled non-ground;
Yellow: ground misclassified as non-ground; Red: non-ground misclassified as ground)

In AHN dataset, adding ΔH reduces total error from 4.54 % into 4.41 %. Even though the percentage is small, the improvement is clearly visible on a labeled point cloud as can be seen in Figure 33. The type II errors on the building roofs are minimized after ΔH feature was added.



a. Without ΔH feature b. With ΔH feature

Figure 33. The improvement on the building roofs after ΔH feature was added. Red points indicate type II errors

Features selection on MLP

The alternative method extracted 25 features as mentioned in section 3.2.1. Three experiments were done in order to select the most appropriate features for ground classification. In the first experiment, 21 features from Weinmann et al. (2015) were employed. Four more features (two height differences, intensity and return number) were added in the second experiment. The third experiment used 10 features, six from Weinmann et al. (2015) and four additional features as mentioned before. The error rates from all experiments are shown in the Table 15 below.

No	Feature set	Total error (%)	Type I error (%)	Type II error (%)
1	21	26.24	38.89	31.14
2	25	28.69	45.25	12.89
3	10	10.28	10.64	17.11

Table 15. Accuracy assessment using different features set

Even though Weinmann et al. (2015) achieved overall accuracy of 87.29 % using 21 features on Oakland 3D Point Cloud dataset with MLP classifier, the same feature set does not give a satisfactory result when tested on ISPRS dataset. All error rates are considerably high especially type I error; indicating many ground points fail to be correctly classified. This might be due to different scenes between Oakland 3D Point Cloud dataset and the ISPRS filtering dataset.

After four more features were added, type II error reduced drastically. This shows the potential of the additional height differences and non-geometric features. Given the fact that the task in Oakland 3D Point Cloud dataset is to classify point cloud in five classes, while ground classification task only has two classes, it is clear that the task of multi-class classification is more complex than ground classification hence it needs more features to separate more classes. As a consequence, reducing features for ground classification is a reasonable action.

Third experiment kept the most common features for ground classification. Height difference was used in many literatures hence two features with different neighbor definitions are kept. Intensity is also kept due to its sensitivity to detect different surface material. Return number is kept as original information from LIDAR point cloud that shows the indication of point to the ground surface (e.g. last return points are more likely to lie in the ground surface than the first return points). Six more features are chosen from Weinmann et al. (2015) which include linearity, planarity, scattering, omni variance, anisotropy, and change of curvature. Besides representing 3D local shape, those features are commonly used in other literatures (Chehata et al., 2009; Niemeyer et al., 2012). The rest features are not used as they are considered irrelevant for ground classification.

The combination of the selected features is useful. For instance, planarity and scattering help to discriminate flat ground to vegetation. Although building roof has planarity value similar to ground, but both objects could be separated easily by using height difference feature. As a result, ground can be separated among buildings and vegetation.

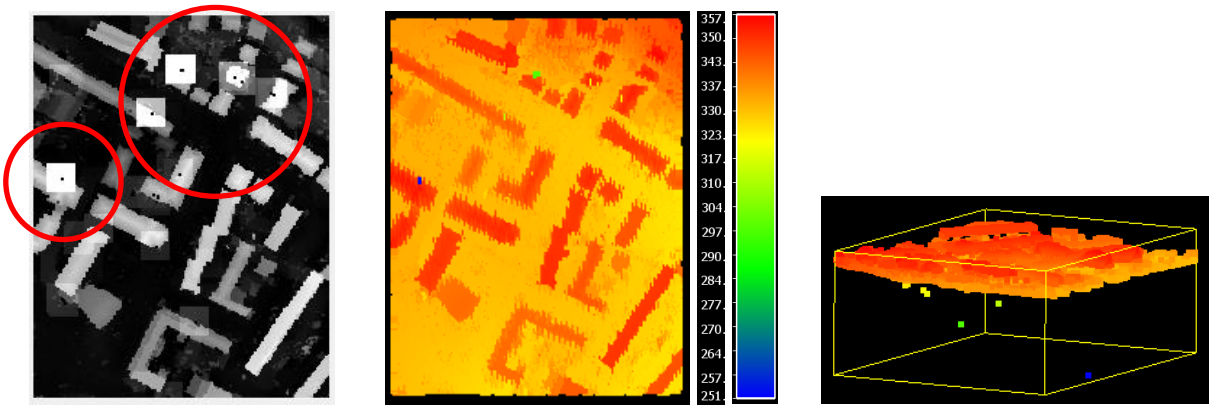
In the third experiment, all 2D features from Weinmann et al. (2015) were no longer used. The underlying reason is 2D features are only calculated in horizontal extent while ground is commonly lies under non-ground object hence it is not makes sense to use 2D feature for the task. There is no relevancy of using 2D feature when the task needs to analyze objects in a vertical extent.

The six 3D geometric property features were also no longer used in the third experiment. Unlike eigenvalue based features, the features do not describe the shape of object on which the point is located. For instance, one of the features is the elevation value itself. This feature is considerably meaningless especially in a varying elevations terrain. In that area, either ground or non-ground objects could be lying at any elevation. Other examples of the unselected features are radius of kNN and the point density because the relevance of those features to the ground classification task is unclear.

4.2.3. Low point outliers

It is common to have low point outliers in point clouds from airborne LIDAR. But, since many filtering algorithms start with an assumption that the lowest point within certain size of area must be a ground point, these types of algorithms are very sensitive to low point outliers and usually needs pre-processing step to clean the outliers before performing the algorithms.

The proposed method might has a similar effect depends on the choice of input image features. Obviously, the height difference (ΔH) feature suffers to this error as the extracted image always has artefacts where low point outliers appear as shown in Figure 34 below. The low point outliers can be seen in Figure 34.b and 34.c as blue and green points meaning the elevations are extremely low.



a. Converted image b. Point cloud in color coded c. Point cloud in perspective view

Figure 34. (a) Artefacts caused by outliers on ΔH feature images; (b) and (c) corresponding point cloud in nadir and perspective view

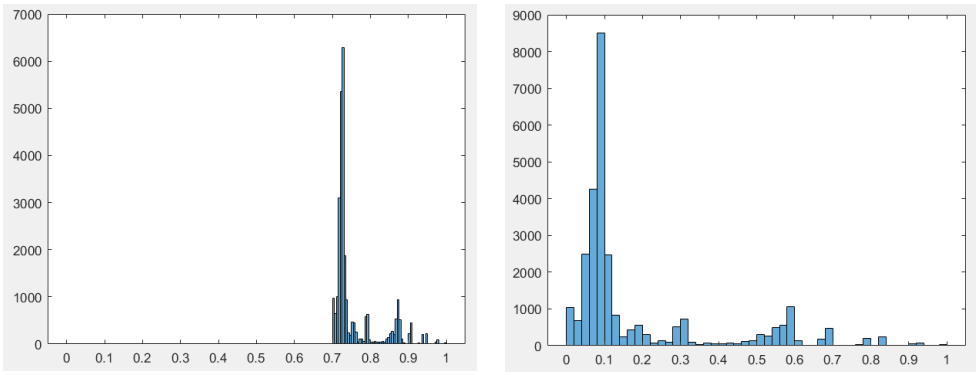
Ten ISPRS training sites were tested in order to find the effect to the accuracy. Two experiments were performed. The first experiment was done on the original dataset containing the outliers. In the second experiment, the outliers were removed by calculating the height of each point compared to the ground surface derived from the reference data. Points which have elevation less than 1 m to the ground surface were removed. Both experiments have four features consist of elevation, intensity, return number and height difference. The result shows that the overall accuracy increases 1.93 % as the outliers were removed.

The artefacts caused by low point outliers in the ΔH image are predictable as the feature calculates the height difference to the lowest point in the neighborhood. Once the lowest point is extremely low, the height difference is extremely high. As a consequence, the classification accuracy is getting lower.

The more interesting issue is whether the low point outliers affect the classification accuracy if the image is extracted without ΔH feature, only by elevation, intensity, and return number. Using those features, the outliers only appear as small black pixels in the image as seen in Figure 34.a.

Two experiments were performed similar to the previous experiments; with and without outliers. The overall accuracy still increases 1.49 % after the outliers were removed. The reason behind this is although the outliers only affect small pixels in the image, the image contrast is worse than the image contrast if the image is extracted without outliers (Figure 35.a). The difference is due to the normalization with maximum and minimum value. In the image containing outliers, the normalization takes lowest outlier point as a minimum value. If the minimum value is very low, then the histogram distribution is not balanced (Figure 35.b). From the experiment, it can be concluded that the network learns better on the better contrast.

a. Elevation image with outliers (left) and without outliers (right)



b. Corresponding histogram of image with outliers (left) and without outliers (right)

Figure 35. (a) Images created in case of low point outliers exist and (b) the corresponding histograms

4.3. Comparison of methods

The accuracies of two proposed method are presented in section 5.2, along with the accuracies of state-of-the-art algorithms. Deep CNN-based approach (Hu and Yuan, 2016) is the baseline of deep learning for ground classification. It is the pioneer of CNN for ground classification yet provides the highest accuracy than others. Since the MLP method extracts some contextual features, Random Forest (RF) classifier was also chosen as a baseline method. The same features were used but the classification was performed using RF instead of MLP. The last comparison method is LAStools software. It uses a modified of progressive TIN densification algorithm (Axelsson, 2000). All comparison methods were performed using the same cleaned dataset to avoid the influence of outliers and each algorithm has a result on its best performance.

CNN-based classification

A modification was carried out for the deep CNN approach since the point-to-image conversion takes a very long time before all points could be converted into images. Hu and Yuan (2016) originally converted point into 128 x 128 pixels image. In this study, the size was reduced into 32 x 32 pixels. Reducing the image size significantly fasten the conversion from 6.5 to 0.4 seconds for each point. However, the window size is still the same 96 x 96 m to keep the contextual feature are captured on a same spatial extent. As can be seen in Table 16 below, the output feature maps are smaller than the original ones as a consequence of having smaller image. But the number of layers and filter size are the same. In general, the architectures are similar; only differ on the feature map sizes. Lastly, the network was trained using the same training samples and having data augmentation as well 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 16. A modification of width, height and depth compared to the original version.

RF-based classification

RF is an ensemble of bootstrap-aggregated (bagged) decision trees. It combines many weak learners together in order to obtain a better classifier.

The same 10 features set with MLP were used to train the RF model. The number of trees was set to 200. Maximum number of features was set to four. The selected parameters gave the best result in the experiments. The number of tree defines how many decision trees will be made while the maximum number of features defines how many features are considered to find the best split at each tree.

5. DTM RESULT AND DISCUSSION

Results and discussions from both proposed method are explained in the following sub sections. The qualitative results and accuracies assessment are presented. The comparison of the computational time is also shown. Both proposed methods use the most optimum hyper-parameters configuration. The FCN method uses four dilated convolution layers and no max-pooling layer, while the MLP method uses 10 selected features and three hidden layers.

5.1. Qualitative analysis

5.1.1. ISPRS dataset

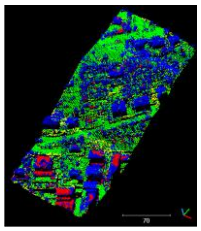
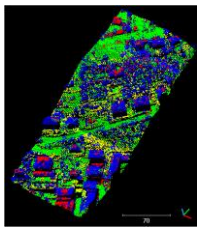
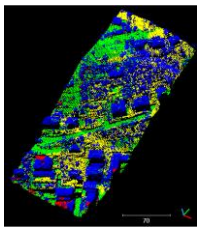
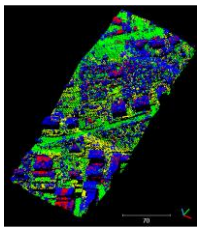
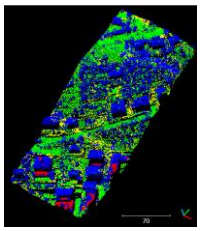
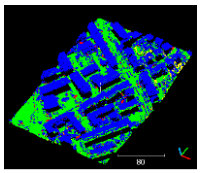
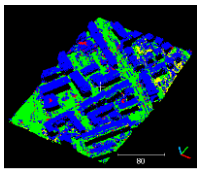
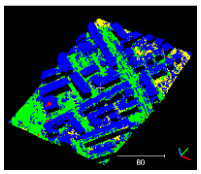
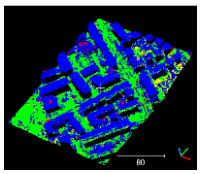
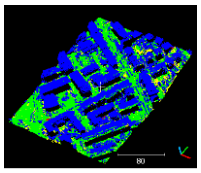
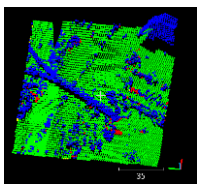
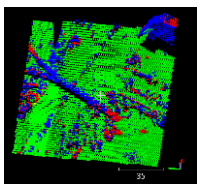
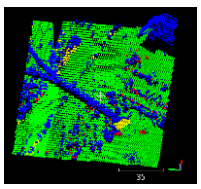
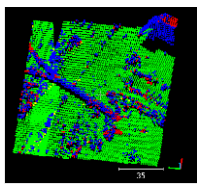
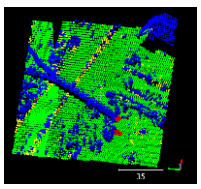
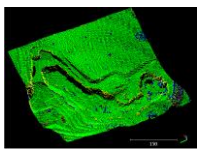
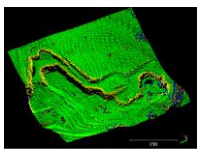
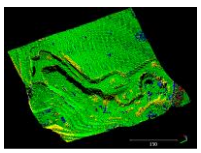
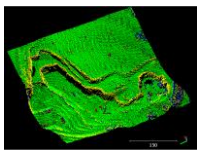
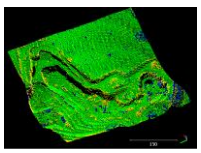
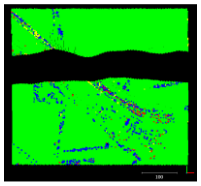
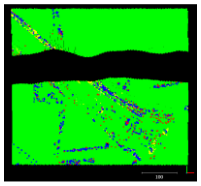
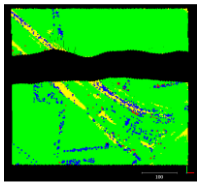
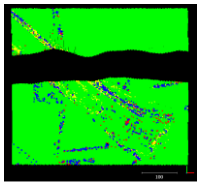
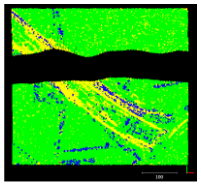
Site	FCN	MLP	CNN	RF	LAStools
Samp11					
Samp12					
Samp21					
Samp53					
Samp61					

Figure 36. Visual inspection on five ISPRS testing sites.
Green: correctly labeled ground, Blue: correctly labeled non-ground,
Yellow: type I error, Red: type II error.

Figure 36 shows the labeled points of all method at each ISPRS testing sites for a qualitative checking purpose. Colors indicate either point is correctly or incorrectly labeled. Green points mean correctly labeled ground points. Blue points mean correctly labeled non-ground points. Yellow points correspond to ground points which are misclassified as non-ground points (type I error). Red points correspond to non-ground points which are misclassified as ground points (type II error).

Meanwhile, DTM as a final product is shown in Figure 37. The DTMs are generated from FCN and MLP methods on the five testing sites, and compared to the reference DTM by manual editing.

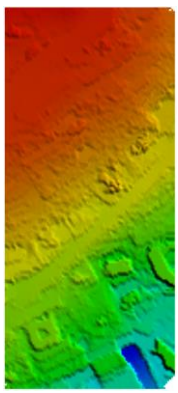
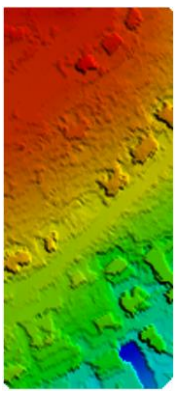
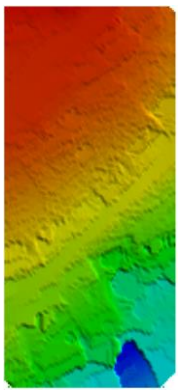
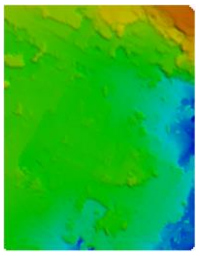
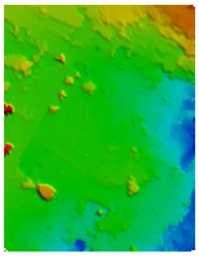
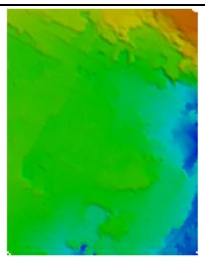
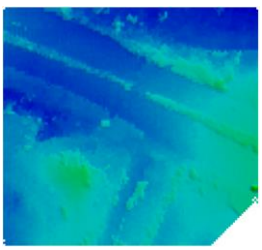
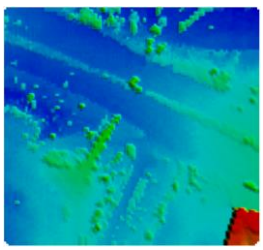
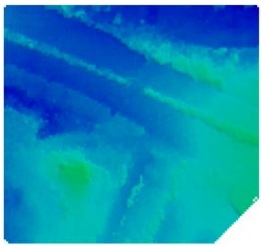
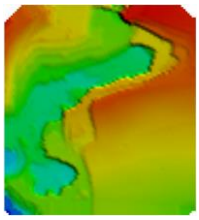
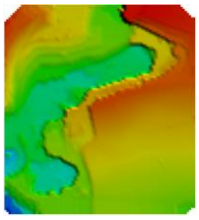
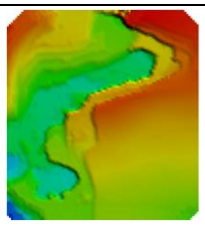
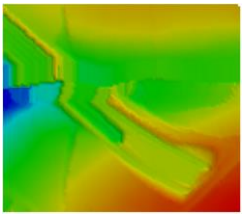
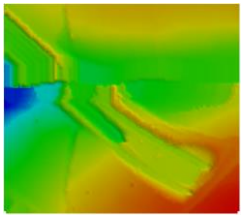
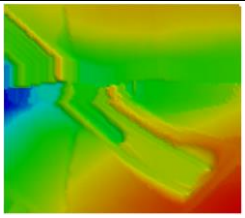
Site	FCN	MLP	Reference DTM
Samp11			
Samp12			
Samp21			
Samp53			
Samp61			

Figure 37. DTMs on five ISPRS testing sites

Samp11 is the most difficult area to handle among the others. Recall the general assumption that ground points are usually lower than non-ground points, the combination of steep terrain and low vegetation makes the assumption is no longer valid. Most of the algorithms failed to separate ground to the low vegetation. However, the result from CNN is the worse than the others since many ground points are misclassified (indicated by more yellow points). This might be happened since CNN approach only takes height difference feature for the classification while it is known that it might not works in this kind of situation. One of the motivations of this study is to improve the result from CNN approach when deals with sloped terrain. This sample could be the evident of how FCN approach can produce a better result than CNN.

More than that, another difficulty comes from buildings which have similar elevation to the surrounding ground as shown in Figure 38. Buildings and ground also make a similar pattern. Gevaert et al. (2018) mentioned this situation as one of the difficulties on DTM extraction. The situation raises caused by a step-like pattern and the buildings are coplanar to the ground. In this area, none of the algorithms can remove the buildings perfectly, not even LAStools software that usually successfully minimizing type II error.

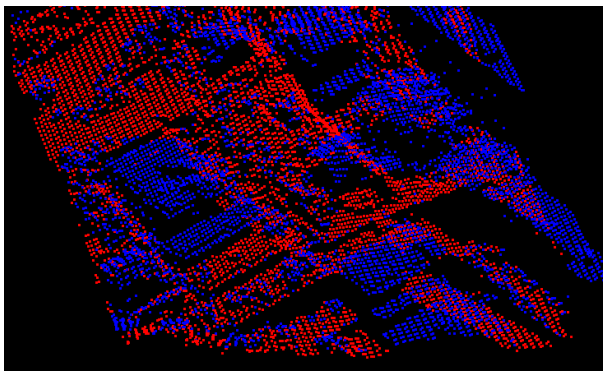


Figure 38. Difficult situation caused by buildings have similar height to surrounding ground in Samp11. (Red: ground points, Blue: non-ground points)

Samp12 is the situation when the ground is typically lying below non-ground objects. Flat terrain and clear building shape make most algorithms work well. This sample was chosen to test algorithms on the situation on a normal scene which is easy yet frequently faced. In this site, both FCN and MLP perform well and type II error is not really noticeable. However, it can be seen that type I error on MLP is worse than FCN.

Samp21 has bridge and one large building to challenge the algorithms. It is clear that MLP and RF have more type II errors. Errors on a building roof certainly come from the height difference feature as the radius of the cylinder is not large enough. Meanwhile, FCN and CNN look better although type I errors are still visible on a result from CNN. In a ground classification task, bridge always gives a difficulty especially on a connection between bridge and ground due to both objects has a fuzzy boundary to each other. As a result, most algorithms have errors on a connection between bridge and ground.

Results on Samp53 show that ground points along the break-lines are easily misclassified as non-ground points as indicated by the yellow points. Due to its shape, break-lines in Samp53 are difficult to capture especially if the algorithm assumes lower points are ground. MLP and RF which use two height difference features look as the worst method when dealing with the ground along the break-lines.

Samp61 has embankment as shown in Figure 39 below. It is a man-made structure but is considered as the ground surface. Similar to break-lines in Samp53, this object could be easily misclassified as non-ground points. FCN, MLP and RF fail to remove non-ground objects on top of the embankment. On the other hand, CNN and LAStools fail to capture the embankment itself as a ground surface.

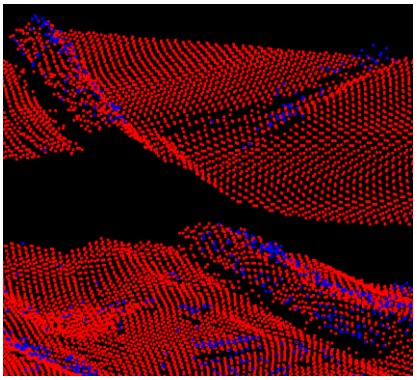
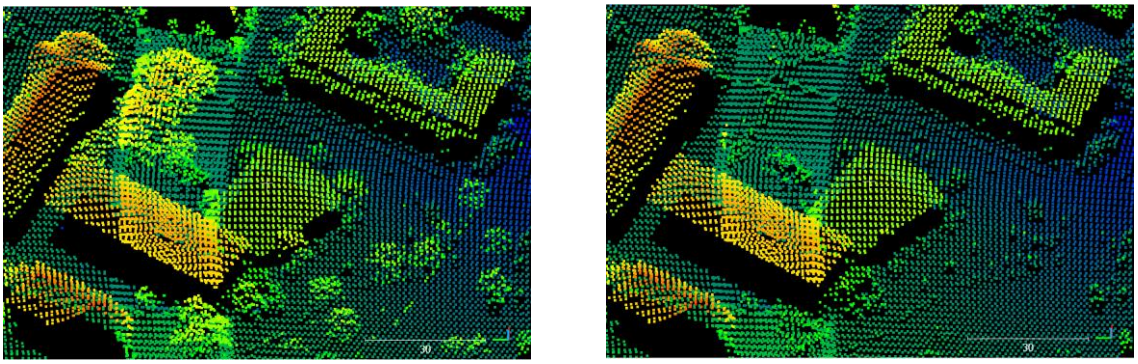


Figure 39. Perspective view of embankment on Samp61.
(Red: ground points, Blue: non-ground points)

According to Figure 36, it appears that Samp53 and Samp61 have very few non-ground points compared to the number of ground points. Samp53 has 32,989 ground points and only 1,388 non-ground points while Samp61 has 34,563 ground points and only 1,247 non-ground points. The situation makes type II error raises easily only by a small number of misclassified non-ground points.

As mentioned earlier, the alternative method using MLP is less reliable than the first or the baseline methods. In the beginning, it was believed that combining height difference features and eigenvalue-based features could separate ground from non-ground perfectly. The idea is eigenvalue-based features separate vegetation to ground and buildings then height difference features separate ground to buildings. However, the combination did not work well. One of the reasons probably comes from the dataset itself. 15 sites of ISPRS dataset only contain single and last returns while the original point cloud has the first returns as well. Probably, ISPRS did not provide the first returns for the labeled points because most of the filtering algorithms do not consider the first return points to work with. Figure 40 shows the different of point cloud which contains all returns and point cloud that has single and last returns only. Points on vegetation are mostly removed, leaving only points on the lower part of the vegetation.



a. All returns b. Single and last returns only

Figure 40. The difference of point cloud with (a) all returns and (b) single and last returns only

As a consequence, the eigenvalue-based features, which are calculated from the k-Nearest-Neighbors, fail to represent the shape of the vegetation. Figure 41 illustrates the problem when the red point is tested using two eigenvalue-based features. In this scenario, red point is part of tree (lower stem) and recorded as last return. According to the idea, vegetation should have low planarity and high scattering values in opposite to flat ground or buildings. If all returns are involved, the idea could be achieved. But if the eigenvalues are calculated without the first return, the red point gives high planarity and low scattering values which are similar to the values of the surrounding ground. Hence the red point could be misclassified as ground instead of non-ground point.

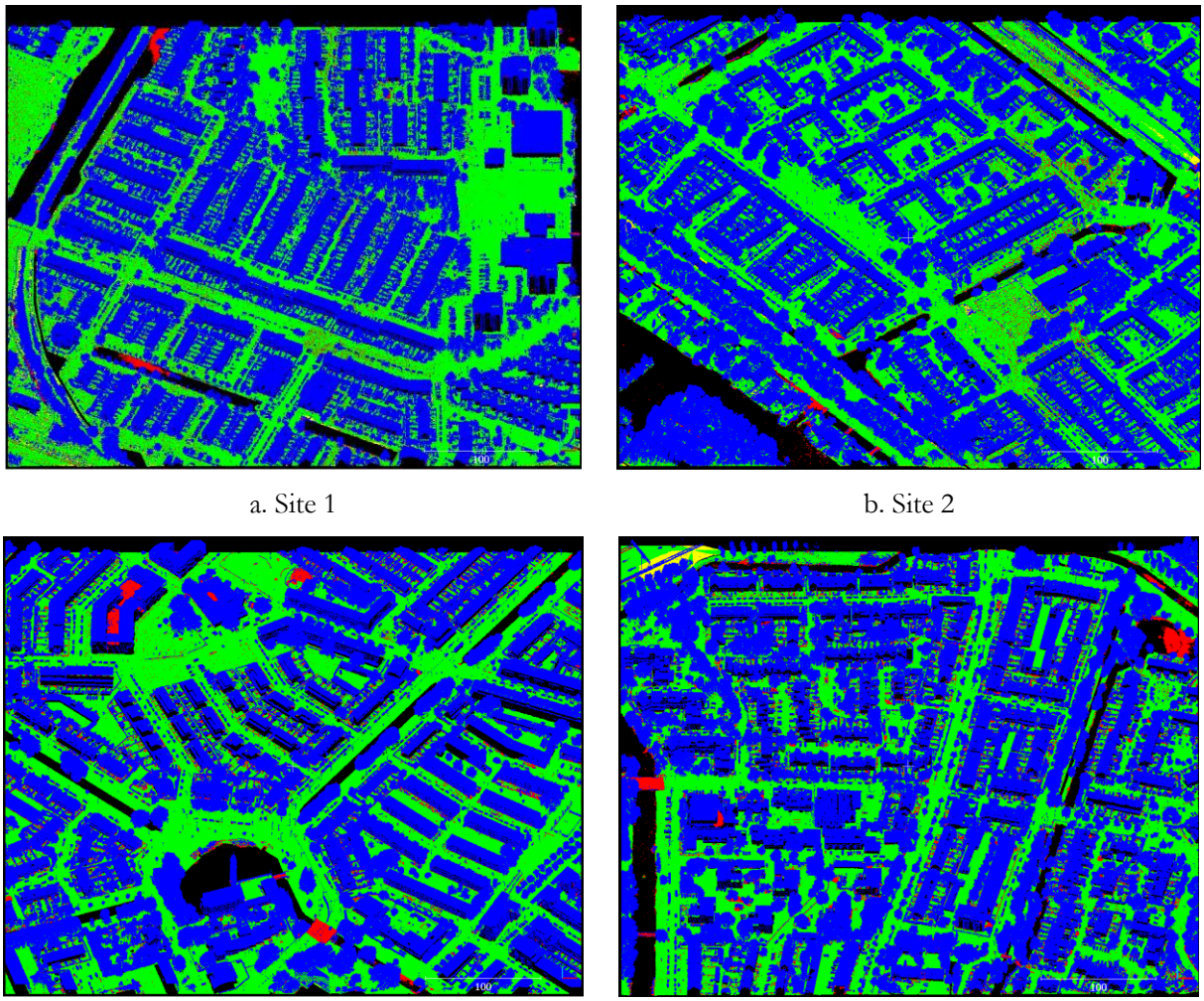
a. Scene illustration (Left: all returns; Right: single and last returns only)

b. 10-Nearest-Neighbors of the red point (Left: all returns; Right: single and last returns only)

Figure 41. The difference of eigenvalues caused by the different definition of the neighbors

5.1.2. AHN dataset

Figure 42 shows the labeled points from the proposed method using FCN on four AHN testing sites for qualitative checking purpose. Green points are correctly labeled grounds, blue points are correctly labeled non-grounds, yellow points are type I errors and red points are type II errors.



a. Site 1 b. Site 2

c. Site 3 d. Site 4

Figure 42. Labeled point clouds on four AHN testing sites.
Green: correctly labeled of ground, Blue: correctly labeled of non-ground, Yellow: ground misclassified as non-ground, and Red: non-ground misclassified as ground

It can be seen that most of all ground and non-ground points are classified correctly, indicated by the huge number of green and blue points over the red and yellow points. However, there is a few number of type II errors (red points) on some areas.

In testing site 3, two buildings on the left side of the area failed to be classified correctly. Those are large buildings with 80x40 m and 40x35 m in size. It looks like the receptive field size of FCN is still not large enough to handle the building. While building on the right has planar roof, building on the left has different roof level. The roof on the center is much lower than the surrounding roofs, causing that roof is misclassified as ground, as shown in Figure 43 (red points).

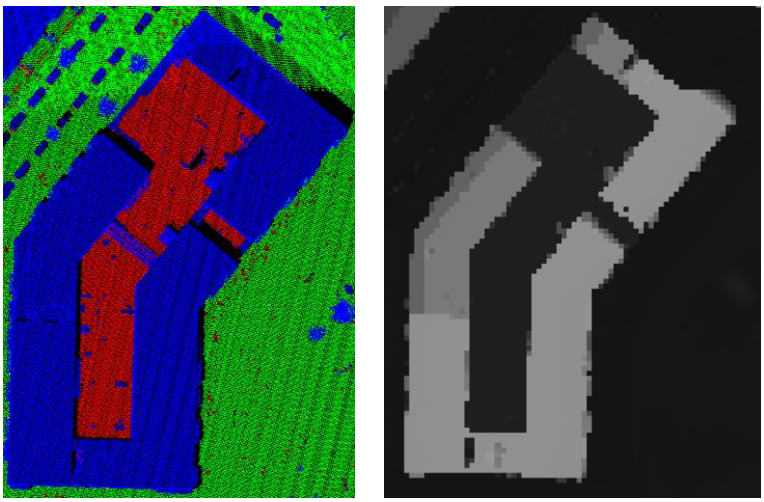


Figure 43. Error on building roof (red points) and the corresponding Z image

Points on the water also contribute type II error because the geometry is similar to points on the ground surface. It seems that no automatic algorithms can handle the situation perfectly. In this case, manual editing is needed. Moreover, since the errors on the water do not affect the geometry of the extracted DTM, those errors can be ignored.

Other type II errors are sourced by points inside buildings as seen in Figure 44. Small number of laser pulses penetrated the building through the small gaps on the roof, window, or wall, due to the high density of AHN dataset. Hence, many points are lying on the ground floor inside the building. As a result, those points are misclassified as ground. The reason is the elevation of those points is similar to the height of the surrounding terrain. If this is considered to be a problem, manual editing using building footprints is needed.

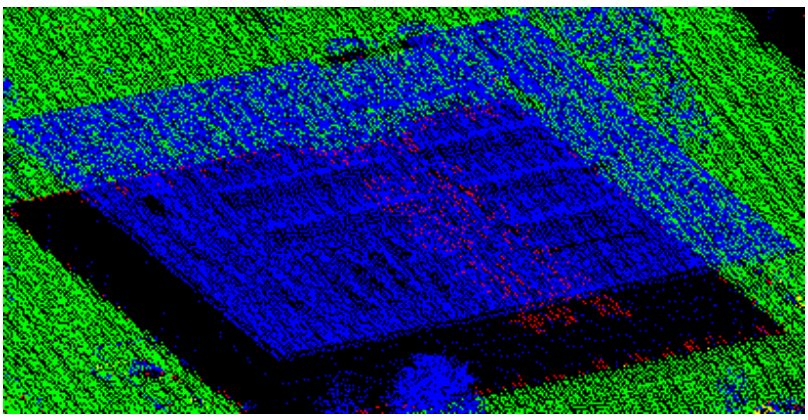


Figure 44. Errors on point cloud inside building (red points)

Bridge also gives type II errors as shown in Figure 45. Although bridge in Samp21 of ISPRS dataset is successfully classified as non-ground, bridges in AHN dataset are misclassified as ground as can be seen in testing sites 1, 3 and 4. This is caused by the different type of bridge. The bridge in the ISPRS dataset is an overpass bridge which has a higher elevation than the surrounding ground while the bridges in the AHN dataset are considered as a flat bridge and has a similar elevation to the surrounding ground. However, it can be seen that a car on the bridge is nicely classified as a non-ground point although the bridge itself is misclassified as ground.

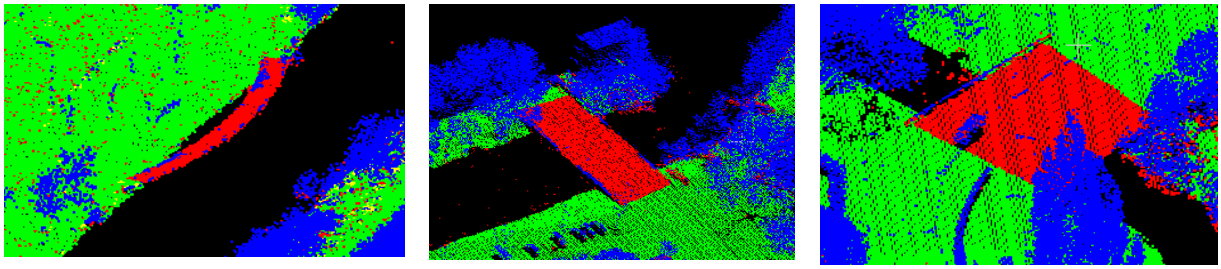
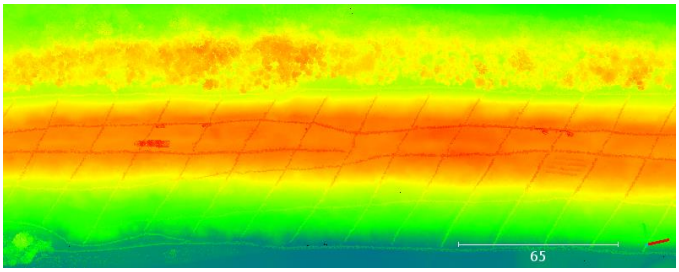
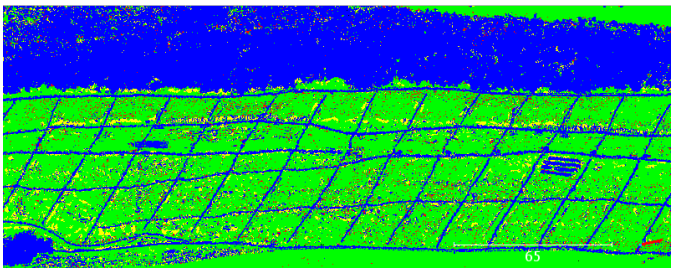


Figure 45. Type II errors caused by the bridge

While ISPRS dataset offers various types of terrain, terrain in AHN dataset is mostly flat hence makes it difficult to see the performance of the method on a sloped terrain. However, there is area in the southwest of The Netherlands that offers sloped terrain. Terrain slope in the area caused by dune, a man-made structure along the shoreline. The classification result is shown in Figure 46. It can be seen that FCN successfully classifies ground and non-ground even in the sloped terrain. Furthermore, FCN also captured the pattern of vegetation on the terrain.



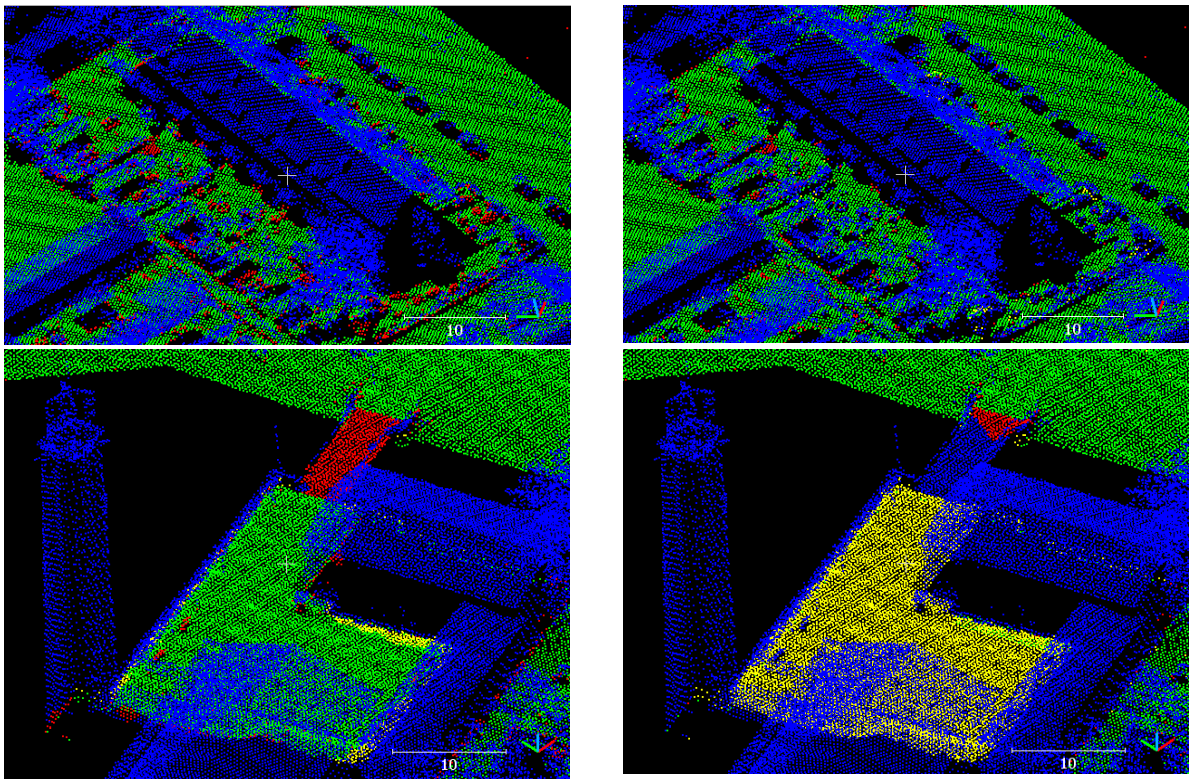
a. Point cloud in dune, color coded by height



b. Labeled points on dune.

Figure 46. Ground classification result on dune
Green: correctly labeled of ground, Blue: correctly labeled of non-ground, Yellow: ground misclassified as non-ground, and Red: non-ground misclassified as ground

Compared to the results from LAStools, FCN has more errors on low vegetation. It seems that the proposed method cannot capture small and low objects. This is most likely caused by the pixel size that is too large to handle the high density point cloud. However, FCN works better in other situations when LAStools fails to classify ground surface correctly, such in the middle of the pond as shown in Figure 47.



a. FCN b. LAStools

Figure 47. Comparison results between (a) FCN and (b) LAStools
Green: correctly labeled of ground, Blue: correctly labeled of non-ground, Yellow: ground misclassified as non-ground, and Red: non-ground misclassified as ground

5.2. Accuracy assessment

Accuracy assessment was performed on ISPRS and AHN testing sites. Three error rates are used to show the accuracy. Total error is the percentage of misclassified points from both ground and non-ground classes. Type I error is the percentage of misclassified ground points into non-ground points. Type II error is the percentage of misclassified non-ground points into ground points.

Site	**Total Error (%)**				
	FCN	MLP	CNN	RF	LAStools
Samp11	14.54	26.14	19.47	27.69	17.67
Samp12	3.97	8.67	7.99	10.93	6.97
Samp21	1.55	4.19	2.23	6.61	6.66
Samp53	4.89	17.11	5.67	10.95	14.37
Samp61	1.16	5.86	4.20	3.98	17.24
Average	**5.22**	12.39	7.91	12.03	12.58

Site	**Type I Error (%)**				
	FCN	MLP	CNN	RF	LAStools
Samp11	12.86	33.18	27.10	35.03	26.94
Samp12	3.03	11.58	13.92	14.89	12.87
Samp21	0.20	0.64	1.63	4.13	7.98
Samp53	3.88	16.78	4.44	10.30	14.84
Samp61	0.55	5.76	3.95	3.41	17.85
Average	**4.10**	13.59	10.21	13.55	16.10

Site	**Type II Error (%)**				
	FCN	MLP	CNN	RF	LAStools
Samp11	16.80	16.67	9.20	17.80	5.18
Samp12	4.96	5.60	1.75	6.78	0.77
Samp21	6.43	16.99	4.39	15.57	1.87
Samp53	29.10	24.78	34.79	26.58	3.24
Samp61	18.04	8.58	11.06	15.39	0.40
Average	15.07	14.52	12.24	16.42	**2.29**

Table 17. Error rates of all methods.
Compared to previous deep CNN approach by Hu and Yuan (2016), FCN as the first proposed method gives lower total error.

The results on five ISPRS testing sites are shown in Table 17. FCN approach as the proposed method achieves the lowest total error and type I error rates among all algorithms. Even though the type II error is worse than the results of CNN and LAStools software, it is easier to fix type II error than type I error in the manual editing (Sithole and Vosselman, 2004). Visually, type II errors are striking while type I errors are very difficult to identify. Type I errors make holes in the terrain which is difficult to decide whether the holes come from type I errors or the removed non-ground objects.

In contrast to the FCN, MLP as the alternative method turns out to be disappointing especially for type I error. Even though type II error is lower than FCN, post manual editing is more difficult to do if many ground points are misclassified. Moreover, MLP error rates are similar to error rates from RF. This could be an indication that the extracted features are more important than the choice of the classifier.

Compared to CNN as the baseline method, FCN gives a better result even in the sloped terrain (Samp11 and Samp53). Since one of the motivations of this study is to deal with slope terrain as described in section 1.1, it can be concluded that the problem might be solved by using FCN. Meanwhile, MLP as the alternative method cannot prove to be an alternative to CNN.

However, it should be noted that the accuracy of CNN in this study was derived from the modified network architecture and the different training set. Hu and Yuan (2016) reported 1.22 % of total error, 0.67 % of type I error and 2.262 % of type II error on 15 ISPRS sample sites. Those results are achieved as the CNN was trained on 17 million points of various types of terrain while the reported accuracies of CNN in this study were obtained only using 106,074 points of 10 ISPRS training sites.

Tested on AHN dataset, the results of the proposed method using FCN are shown in Table 18 below. As a comparison, Table 19 shows the results of LAStools software. Although LAStools achieved lower error rates, the difference is small. Compared to the results in ISPRS datset, LAStools performs better in AHN dataset probably because the terrains are not difficult as seen in ISPRS dataset. It also shows that most of filtering algorithms such as LAStools assume that ground surface is a relatively smooth surface and the non-ground objects are elevated to the surrounding ground. Such assumption is valid on AHN dataset thus LAStools performed very well on this dataset. Meanwhile, if the terrains are not similar to the assumption, such in ISPRS dataset, LAStools did not perform as good as in AHN dataset.

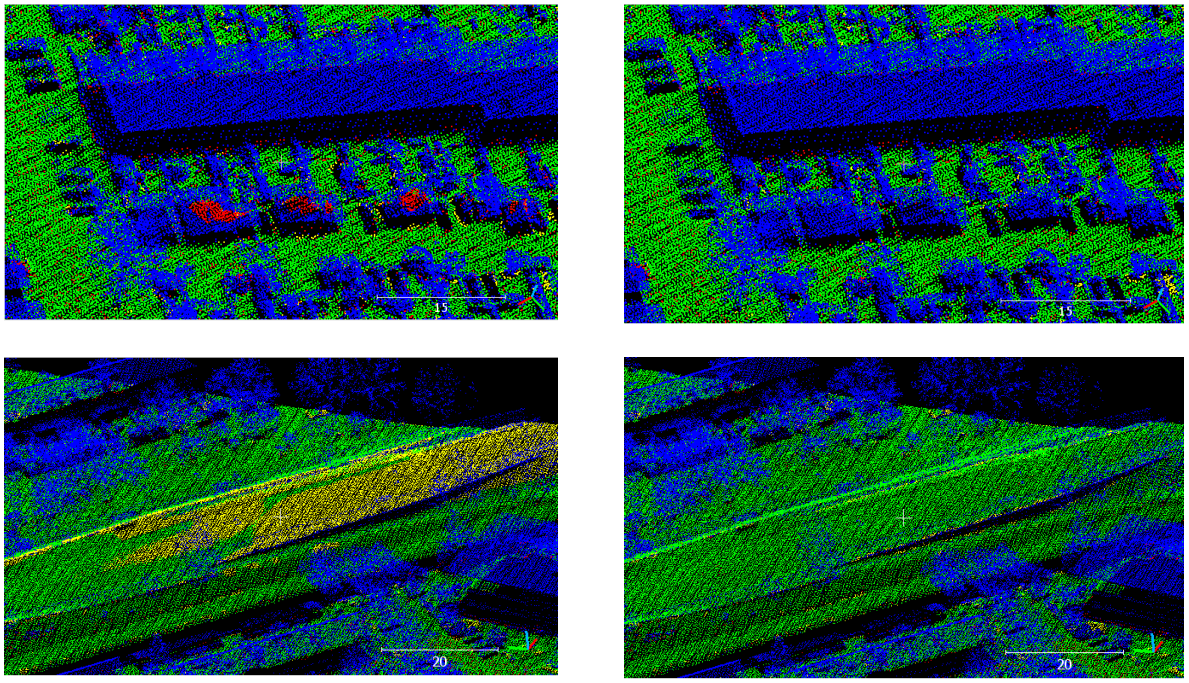
Site	Total error (%)	Type I error (%)	Type II error (%)
1	3.79	2.00	5.83
2	2.36	1.54	3.21
3	4.07	0.07	7.50
4	4.31	0.09	7.56
AVG	**3.63**	**0.93**	**6.03**

Table 18. Accuracies of FCN on AHN dataset

Site	Total error (%)	Type I error (%)	Type II error (%)
1	5.08	1.70	8.91
2	3.44	1.98	4.94
3	2.32	1.27	3.21
4	2.46	1.03	3.56
AVG	**3.33**	**1.50**	**5.16**

Table 19. Accuracies of LAStools on AHN dataset

As noted before, the results from CNN which was trained using a large number of training samples (Hu and Yuan, 2016) are more accurate than the CNN which was trained using a small number of training samples as used in this study. Hence it is worth to train the FCN using larger number of training samples in order to improve the result. In AHN dataset, an experiment was done by adding eight additional training sites so that there are 17 training sites in total. After the FCN was re-trained, the error rates reduce 0.24 % for total error, 0.44 % for type I error and 0.07 % for type II error. Although the improvement is small, some misclassified points are correctly classified after adding more training samples as can be seen in Figure 48.



a. Original results b. After more training samples were added

Figure 48. The improvements after more training samples were added.
(a) using nine training sites and (b) using 17 training sites

5.3. Computational cost

The main motivation of this study is to propose a more efficient deep learning approach on ground classification as previously proposed by Hu and Yuan (2016) using CNN. As mentioned in section 1.1, ground classification using CNN is slow due to the redundancies on the conversion. Table 20 shows the computational time of FCN to CNN and LAStools software on ISPRS dataset. All experiments were 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)
FCN	36 minutes	12.0 hours	7.8 seconds
CNN	47 hours	2.5 hours	126.0 seconds
LAStools	-	-	3.8 seconds

Table 20. Computational time comparison

As predicted, method using FCN is significantly faster than CNN either for the conversion or the testing. CNN approach is slow because every single point is converted into separate image while the proposed method converts all points into one image. The different of conversion technique is due to CNN as an image-wise classification while FCN as a pixel-wise classification. However, training on FCN is slower. But once the network has been trained, it can be used in the future to classify another data without needs to train the network anymore. In other words, it is prefer to have a faster conversion but slower training time than the other way around.

6. MULTI-CLASS CLASSIFICATION RESULT

More classes were introduced as the follow up from the ground classification using FCN on AHN dataset. As mentioned in section 3.1.4, two classifications were employed in order to derive ground, vegetation and building. Figure 49 shows the images in the first and second classification and the corresponding labeled pixels. Images are different depending on which point is chosen to represent the pixel value. First classification uses lowest point while second classification uses highest point. This schema is chosen so that the first classification captures ground points while second classification captures vegetation and building points. It can be seen that images in the second classification show the different patterns between vegetation and buildings clearly.

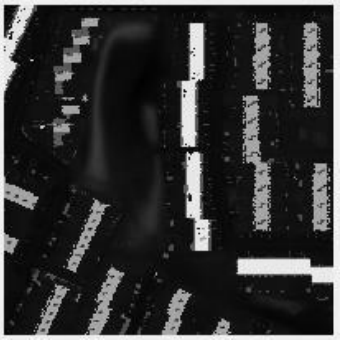
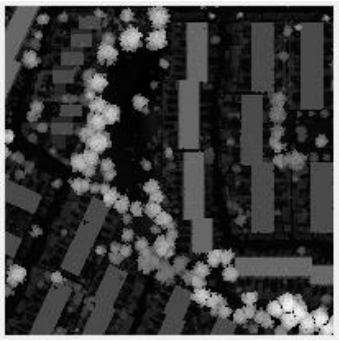
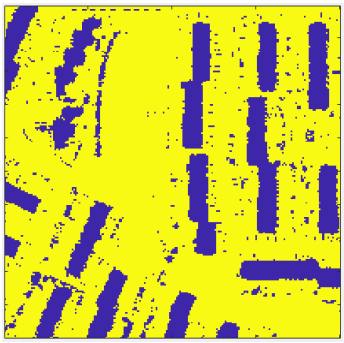
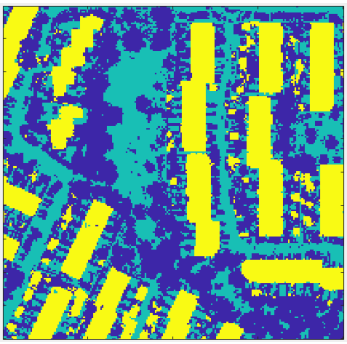
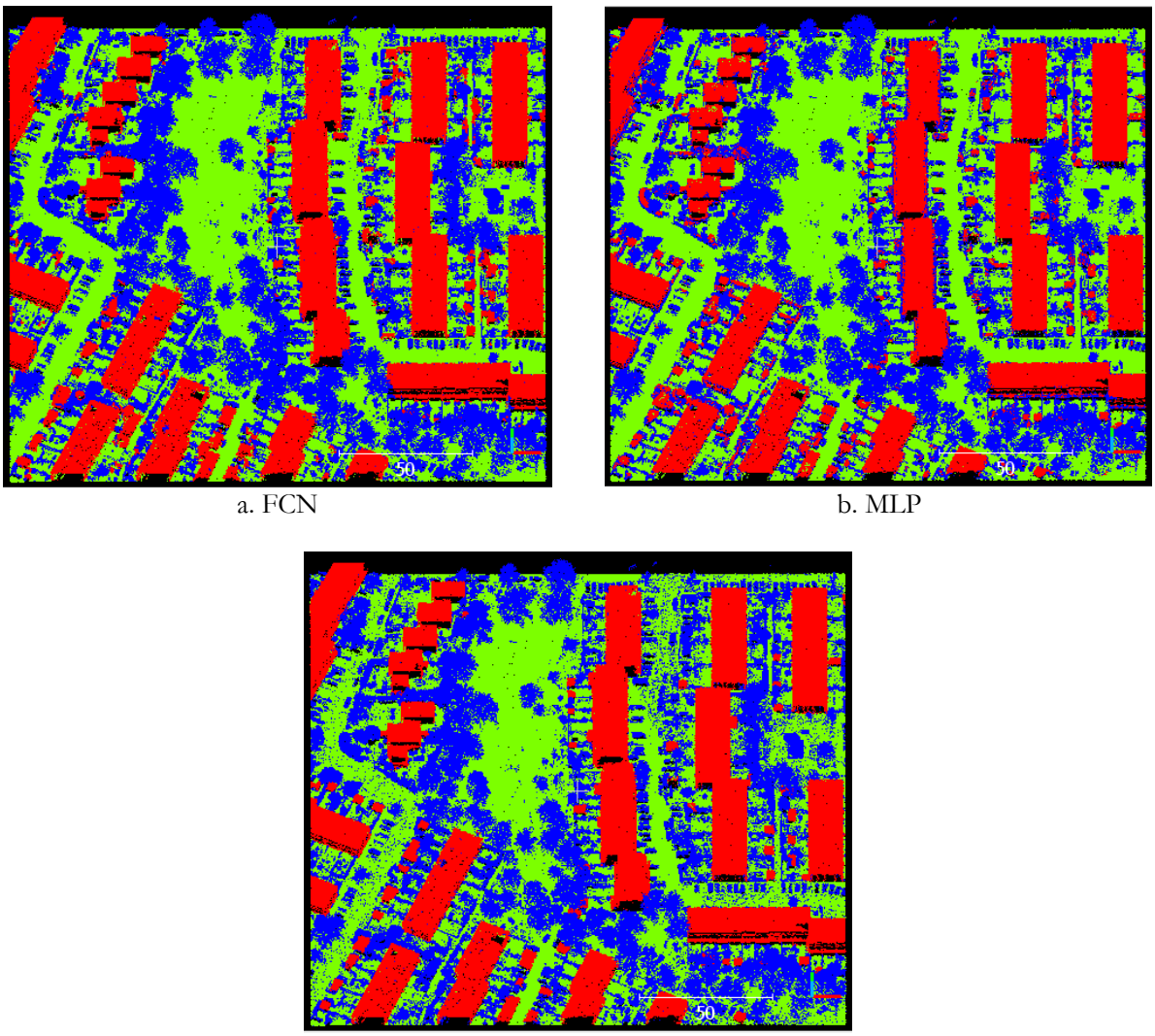
	First classification	Second classification
Converted image	 Lowest point	 Highest point
Labeled pixels	 - Ground (yellow) - Non-ground (blue)	 - Ground (cyan) - Vegetation (blue) - Building (yellow)

Figure 49. Converted images and labeled pixels from two classifications

The predicted labeled points using FCN and the corresponding true labeled points are shown in Figure 50 below. It is clear that most of ground, vegetation and buildings are correctly classified. All of large buildings are perfectly separated from the vegetation. However, some small buildings are still classified as vegetation and the other way around. As a comparison, MLP was used to classify the same dataset. It used the same 10 features and architecture as done in a ground classification. The number of neighbors (kNN) was increased from 10 as used in ISPRS dataset into 50 due to the different point density.



a. FCN b. MLP

c. Reference labels

Figure 50. Predicted labels from (a) FCN, (b) MLP and (c) the reference labels of three-class classification (Green: ground, Blue: vegetation, Red: building)

Table 21 and 22 show the accuracy assessment using correctness and completeness from the results of FCN and MLP. Correctness calculates the number of correctly labeled points (true positive) over the number of points detected (true positive + false positive). Completeness calculates the number of correctly labeled points (true positive) over the number of reference points (true positive + false negative).

Class	Correctness	Completeness
Vegetation	92.91 %	85.35 %
Ground	92.09 %	97.77 %
Building	90.49 %	94.89 %
AVG	**92.83 %**	**92.67 %**

Table 21. Accuracies of FCN on multi-class classification

Class	Correctness	Completeness
Vegetation	90.37 %	80.97 %
Ground	92.47 %	98.98 %
Building	89.87 %	88.37 %
AVG	**90.90 %**	**89.44 %**

Table 22. Accuracies of MLP on multi-class classification

7. CONCLUSION AND FUTURE WORK

7.1. Conclusion

In this study, two methods for ground classification were designed based on deep learning domain. The first proposed method uses FCN whereas the alternative method uses MLP. Both methods are evaluated over CNN, RF and LAStools software as the state-of-the-art methods. In general, the first proposed method works by converting point to image then labeling each pixel in the image using FCN and transferring labels from pixels to points. The alternative method works simpler in terms of the network architecture. It starts with feature extraction then train an MLP classifier in order to label the testing samples. The accuracy assessment on five ISPRS testing samples shows that the FCN approach achieves the lowest total error and type I error rates while the lowest type II error is obtained by LAStools software. The alternative method using MLP is worse than CNN and LAStools software, and it has similar accuracies to RF due to the same features involved. In this case, it can be concluded that the features play a more important role than the choice of the classifier.

The motivation of this study is to design a more efficient ground classification using deep learning since the previous CNN based method (Hu and Yuan, 2016) is slow due to the redundancies in point-to-image conversion. In the experiment, FCN is significantly faster than CNN for the conversion and testing time. Although training time is slower, it can be say that it is preferred to have a faster conversion and testing time than training time. In a real case, let say using AHN dataset for The Netherlands, training is done only one at the beginning. If the training samples represent all terrain characteristic in The Netherland, once the network is trained and gives a reliable result on the testing areas, training is not needed anymore. But if the network fails to produce an expected result, that could be an indication to re-train the network. However, it is hard to say when the network needs to be trained again.

In general, if the network will be used to classify an area with different scene characteristic to the training data, the network must be re-trained using transfer learning to adapt the network with the current condition. Based on the experiments in AHN dataset, the network has a difficulty when classifying the large buildings. Although the errors are possibly caused by the insufficient size of the receptive field, the errors also possible come from insufficient of large buildings in the training samples. In that case, one can say that the network should be re-trained with more training samples.

A second motivation of this study is to have a more reliable result on a sloped terrain. As can be seen in Table 5, total error rate of FCN is lower than CNN in two testing samples with sloped terrain; Samp11 and Samp53. But total error rate of MLP is worse than CNN, even in all testing samples. Although the method using MLP is simpler and faster, if the result is not reliable then it is not suggested to use MLP for an alternative method to CNN. However, MLP may work well depends on its features set and the scene types. According to the results on ISPRS testing sites, MLP works well on Samp12 where the terrain is relatively flat and the buildings are relatively small and clearly separate from the ground surface or vegetation. MLP also works well on AHN dataset for the three-class classification. That could happen due to the similar characteristic between AHN testing sites and Samp12 of ISPRS dataset.

Furthermore, the method using FCN was extended to classify more classes. In this study, two more classes were introduced; vegetation and building. The idea is to split non-ground labeled points into more classes. The result shows that most vegetation and buildings are successfully captured even though some small buildings or vegetation are still misclassified into each other. The overall correctness is 92.83% and completeness is 92.67%. As a comparison, MLP was used and results in 90.90% correctness and 89.44% completeness. The results prove that both methods are promising to be used in a more detailed classification, depending on the availability of the labels in the training samples.

Research questions: answered

The following section discusses the solutions of the research questions that were asked in the beginning of the study.

1. What deep learning architectures have been proposed for general classification problem in the literature?
   Chapter 2 discusses the deep learning approaches that already exist in the literature. CNN is widely used for image classification task while FCN is the solution for pixel-wise classification. The proposed method of this study was designed under FCN framework in contrast to the state-of-the-art method that uses CNN. Moreover, an alternative method was also proposed using MLP, a deep feedforward neural network, to give an alternative approach.

2. What deep learning architectures that could deal with point cloud dataset?
   Neither FCN nor MLP can be used to consume point cloud dataset directly. FCN works in the raster domain hence needs point-to-image conversion while MLP needs a feature extraction for training and testing. However, few deep learning architectures exist and are able to directly consume point cloud as an input as described in section 2.1. Even though those frameworks are promising, most of them are designed for object recognition instead of semantic point classification. Although one of them, PointNet, is able to label every single point individually, the framework still works in combination of object classification and semantic classification thus the difficulty arises when preparing the training samples.

3. How to convert point cloud such that point cloud can be consumed by image-based deep learning architecture?
   Section 3.1.1 deals with point-to-image conversion for the FCN method. The conversion of the CNN method proposed by Hu and Yuan (2016) is also explained in order to have a better understanding of the innovation of this study. In general, the proposed method converts all points into single image thus it is more efficient because it avoids redundancies as appears if all point are converted into separate images. Section 3.1.1 also explains the difficulties such as empty pixel or more than one point within a pixel. Image contains four channels; each channel represents elevation, intensity, return number and height difference feature.

4. How to select ground point in case of the output result comes from image-based deep learning architecture?
   Since FCN works in raster domain, labels which are derived from the classifier are embedded to pixels. Transferring label is needed in order to obtain labeled points. Section 3.1.4 explains how the labels are transferred from pixels to points. It starts with transferring labels to the lowest point within each pixel and creating a surface connecting all ground labeled points. Next, all points within a threshold to the surface are assigned as ground points. This post-processing step is needed in order to have label for each point.

5. Which deep learning approach is most suitable for the task?
   It is hard to say which one is the most suitable for ground classification task. Between the two proposed methods, FCN seems better since images have a better representation of the point cloud than features of MLP. Extracting features manually costs a risk especially when the features are not relevant to the task. However, using FCN needs additional steps such as point-to-image conversion, and the labels are originally assigned for the pixels not the points.

6. What are LIDAR features that can be used to help classification?
   Section 4.1.2 describes features which are used for the FCN method. Elevation, intensity, return number and height difference are stacked into channels of the image. The alternative method using MLP extracts more features. In total, 25 features are employed including eight 3D eigenvalue based features, six 3D geometric properties, seven 2D features, two height differences features, intensity and return number.

7. What is the influence of those features?
   All features used in the FCN method are described in section 4.1.2 whereas section 4.2.1 explains the influence of features from the alternative method using MLP. Each feature works in different responses for ground classification task thus all features are combined to work together. However, some features are a bad influence to the classification result. For instance, height difference feature introduces more type I error in case of break-lines terrain. In the MLP method, many features are also irrelevant so that those features are removed to achieve better accuracy and leaving 10 features which are considered to be correlated to the task.

8. How to use those features in the deep learning model?
   The FCN method uses features as an image channel while the MLP method uses them as point's features. Section 3.1.1 explains the use of the features in the image in the FCN method. Each pixel value in each channel is a normalized feature value of the lowest point in the pixel. Next, FCN is trained after the feature images are obtained. The MLP method follows the machine learning procedure when features are extracted for each sample and train a classifier using the features.

9. How does the accuracy and performance of the proposed method compared to the previous CNN-based technique, LAStools software and other supervised-classification technique?
   Table 5 in section 4.3 shows the errors of all methods used in the study. The proposed method using FCN method gives lowest total error (5.22%) and type I error (4.10%) among all methods. But type II error (15.07%) is worse than CNN and LAStools. On the other hand, the alternative method using MLP gives worse total error (12.39%), type I error (13.59%) and slightly better type II error (14.52%) than FCN. Moreover, MLP has similar error rates to RF which is an indication that, in this task, the extracted features are more influential than the choice of the classifier.

10. What will change if vegetation and building classes are added?
    Dense points labeling on ground classification using FCN approach could be done by creating a surface connecting all initial ground points. This simple approach only works for ground and non-ground classes. In a multi-class classification, other approaches should be introduced.

11. How to adopt the proposed method for multi-class classification?
    The idea is to extend the result of ground classification. Points are labeled as ground and non-ground in the ground classification. Then non-ground points are divided into vegetation and building in the next classification. Detailed procedure and the output result are explained in section 3.1.5 and section 4.6.

7.2. Recommendations

The following suggestions are recommended for the future.

- In case of image-based classification, investigation of a better point-to-image conversion is needed including the use of different pixel size in a multi-scale image. This might help to deal with empty pixel or many points in a pixel. Adding more features as is also welcome and needs to be evaluated to achieve a better accuracy. It also needs a better approach when transferring labels to points.
- If point-to-image conversion is to be avoided, deep learning architecture that is especially designed for point clouds can be utilized. Both FCN and CNN are working under raster domain. The drawback is they need to convert point clouds into images before the network can consume the data. Recent research such as PointNet (Qi et al., 2017) that uses deep learning directly to process point clouds without need a conversion is worth to be investigated for ground classification.
- Although this study has added vegetation and building classes, more classes can be added in the future.

LIST OF 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. https://doi.org/10.1016/j.isprsjprs.2005.10.005

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.

Beraldin, J. A., Blais, F., & Lohr, U. (2010). Laser Scanning Technology. In G. Vosselman & H.-G. Maas (Eds.), *Airborne and Terrestrial Laser Scanning* (pp. 1–42). Whittles Publishing.

Bergado, J. R. A., Persello, C., & Gevaert, C. (2016). A Deep Learning Approach to the Classification of Sub-Decimetre Resolution Aerial Images. In *2016 IEEE Geoscience and Remote Sensing Symposium (IGARSS)* (pp. 1516–1519). https://doi.org/10.1109/IGARSS.2016.7729387

Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Briese, C. (2010). Extraction of Digital Terrain Models. In G. Vosselman & H.-G. Maas (Eds.), *Airborne and Terrestrial Laser Scanning* (pp. 135–167). Whittles Publishing.

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. https://doi.org/10.3390/s17010150

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. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-210

CS231n Convolutional Neural Networks for Visual Recognition. (n.d.). Retrieved November 13, 2017, from http://cs231n.github.io/convolutional-networks/

Farabet, C., Couprie, C., Najman, L., & Lecun, Y. (2013). Learning Hierarchical Features for Scene Labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1915–1929. https://doi.org/10.1109/TPAMI.2012.231

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. https://doi.org/10.3390/rs9050498

Gevaert, C. M., Persello, C., Nex, F., & Vosselman, G. (2018). A Deep Learning Approach to DTM Extraction from Imagery Using Rule-Based Training Labels.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. *AISTATS '11: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, *15*, 315–323. https://doi.org/10.1.1.208.6449

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. https://doi.org/10.3390/rs8090730

Ji, S., Yang, M., & Yu, K. (2013). 3D Convolutional Neural Networks for Human Action Recognition. *Pami*, *35*(1), 221–31. https://doi.org/10.1109/TPAMI.2012.59

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. https://doi.org/10.1016/S0924-2716(98)00009-4

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, 1–9. https://doi.org/http://dx.doi.org/10.1016/j.protcy.2014.09.007

Le Cun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. https://doi.org/10.1109/5.726791

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature*, *521*. https://doi.org/10.1038/nature14539

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. https://doi.org/10.1109/TPAMI.2016.2572683
Lu, W., Murphy, K. P., Little, J. J., Sheffer, A., & Fu, H. (2009). A Hybrid Conditional Random Field for Estimating the Underlying Ground Surface From Airborne LiDAR Data, *47*(8), 2913–2922.
Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. *Proceedings of the 30 Th International Conference on Machine Learning*, *28*, 6. Retrieved from https://web.stanford.edu/~awni/papers/relu_hybrid_icml2013_final.pdf
Maggiori, E., Tarabalka, Y., Charpiat, G., & Alliez, P. (2017). Convolutional Neural Networks for Large-Scale Remote-Sensing Image Classification, *55*(2), 645–657.
Mallet, C., Soergel, U., & Bretar, F. (2008). Analysis of Full-Waveform Lidar Data for Classification of Urban Areas. *Photogrammetrie - Fernerkundung - Geoinformation*, *XXXVII*(Part B3a), 2–4. https://doi.org/10.1080/01431160701736448
Maturana, D., & Scherer, S. (2015). VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. *Iros*, 922–928. https://doi.org/10.1109/IROS.2015.7353481
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. https://doi.org/10.1016/j.isprsjprs.2011.10.002
Nair, V., & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on Machine Learning*, (3), 807–814. https://doi.org/10.1.1.165.6419
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. https://doi.org/10.1016/j.measurement.2017.03.007
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 Spatioal Information Sciences*, *I-3*(September), 263–268. https://doi.org/10.5194/isprsannals-I-3-263-2012
Niemeyer, J., Rottensteiner, F., & Soergel, U. (2013). Classification of urban LiDAR data using conditional random field and random forests. *Proceedings of the JURSE 2013*, *856*(1), 139–142. https://doi.org/10.1109/JURSE.2013.6550685
Ning, F., Delhomme, D., LeCun, Y., Piano, F., Bottou, L., & Barbano, P. E. (2005). Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, *14*(9), 1360–1371. https://doi.org/10.1109/TIP.2005.852470
O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks, 1–11. Retrieved from http://arxiv.org/abs/1511.08458
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. https://doi.org/10.1109/LGRS.2017.2763738
Pfeifer, N., & Mandlburger, G. (2009). LiDAR Data Filtering and DTM Generation. In J. Shan & C. K. Toth (Eds.), *Topographic Laser Ranging and Scanning: Principles and Processing* (pp. 307–333). CRC Press.
Pfeifer, N., Stadler, P., & Briese, C. (2001). Derivation Of Digital Terrain Models In The Scop++ Environment. *OEEPE Workshop on Airborne Laserscanning and Interferometric SAR for Digital Elevation Models, ,* (March 2014), 13.
Pinheiro, P., & Collobert, R. (2014). Recurrent convolutional neural networks for scene labeling. *Proceedings of The 31st International Conference …*, *32*(June), 82–90. https://doi.org/10.1109/ICCV.2015.221
Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv*. Retrieved from http://arxiv.org/abs/1706.02413
Rahmayudi, A., & Rizaldy, A. (2016). Comparison of semi automatic DTM from image matching with DTM from Lidar. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *41*(July), 373–380. https://doi.org/10.5194/isprsarchives-XLI-B3-373-2016
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0
Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., … Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y
Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6354 LNCS*(PART 3), 92–101.

https://doi.org/10.1007/978-3-642-15825-4_10
Sherrah, J. (2016). Fully Convolutional Networks for Dense Semantic Labelling of High-Resolution Aerial Imagery, 1–22. Retrieved from http://arxiv.org/abs/1606.02585
Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition, 1–14. https://doi.org/10.1016/j.infsof.2008.09.005
Sithole, G. (2001). Filtering of Laser Altimetry Data Using a Slope Adaptive Filter. *International Archives of Photogrammetry and Remote Sensing*, *34*(3(WG4)), 203–210.
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. https://doi.org/10.1016/j.isprsjprs.2004.05.004
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.
Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for Simplicity: The All Convolutional Net, 1–14. https://doi.org/10.1163/_q3_SIM_00374
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., … Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07–12–June*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594
Volpi, M., & Tuia, D. (2017). Dense Semantic Labeling of Subdecimeter Resolution Images With Convolutional Neural Networks, *55*(2), 881–893. https://doi.org/10.1109/TGRS.2016.2616585
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. https://doi.org/10.1016/S0924-2716(98)00009-4
Vosselman, G., Coenen, M., & Rottensteiner, F. (2017). Contextual segment-based classification of airborne laser scanner data. *ISPRS Journal of Photogrammetry and Remote Sensing*, *128*, 354–371. https://doi.org/10.1016/j.isprsjprs.2017.03.010
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. https://doi.org/10.1016/j.isprsjprs.2015.01.016
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. https://doi.org/10.1109/CVPR.2015.7298801
Yu, F., & Koltun, V. (2015). Multi-Scale Context Aggregation by Dilated Convolutions. https://doi.org/10.16373/j.cnki.ahr.150049
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. https://doi.org/10.3390/rs5083749