TU Delft

Semantic Segmentation of AHN3 Point Clouds with DGCNN

Additional Thesis

Qian Bai

Dept. of Geoscience and Remote Sensing
Delft University of Technology

Supervisors:
Dr. R.C. Lindenbergh
Dr. Liangliang Nan

July 7, 2020 - September 7, 2020

Abstract

Semantic segmentation of aerial point clouds with high accuracy is significant for many geographical applications, but is not trivial since the data is massive and unstructured. In the past few years, deep learning approaches designed for 3D point cloud data have made great progress. Pointwise neural networks, such as PointNet and its extensions, show their ability to process 3D point clouds, especially in classification and semantic segmentation. In this work, we implement DGCNN (Dynamic Graph CNN), which combines PointNet with Graph CNN, and extend its semantic segmentation application from indoor scenes to an aerial point cloud dataset: The Current Elevation File Netherlands (AHN), which was produced by airborne laser scanners for the whole Netherlands. Point clouds from the iteration AHN3 are classified into four classes: ground, building, water and others (including vegetation, railways, etc). Moreover, DGCNN splits the input point cloud into regular blocks before operating on it and processes each block independently, which limits the effective range (receptive field) of the network to some extent. Thus, the second aim of this work is to investigate the impact of the effective range on the performance of DGCNN by adjusting two crucial parameters: the block size and the neighborhood size k in k-NN graphs. It turns out that enlarging the block size or k helps to improve the overall accuracy of DGCNN, but cannot ensure better segmentation results from each individual class. With the block size 50 m and $k = 20$, the most balanced F1 scores for all classes and an overall accuracy of 93.28% are achieved. Based on the evaluation for each setting with a certain block size and k, we also manage to further improve the overall accuracy to 93.51% by combining smaller-scale (with block size 30 m) and larger-scale (with block size 50 m) segmentation results, with $k = 20$.

Acknowledgements

I would like to express my gratitude to Roderik Lindenbergh and Liangliang Nan, who helped me keep the right direction during the research and gave me valuable advice in thesis writing. Besides, this project could not be completed without the cropped dataset provided by Adriaan van Natijne. I would also like to thank Elyta Widyaningrum for giving me useful suggestions with respect to data preprocessing, design of experiments and report writing.

Contents

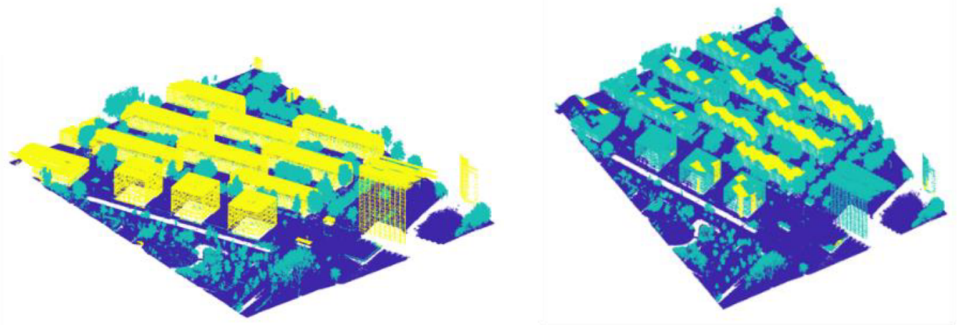
Chapter 1

Introduction

Autonomous and reliable semantic segmentation of 3D point cloud data is an important capability in applications ranging from mapping, 3D modeling, navigation and urban planning, but identifying semantic information from unstructured and unordered point sets is still challenging over the past few decades. Similar to 2D image recognition, this problem has also benefited from deep learning techniques and great progresses have been achieved. PointNet, the first neural network directly consuming raw point cloud data, employs a series of multi-layer perceptrons to learn higher dimensional features for each individual point and concatenates them to obtain the global context within a small 3D block, which shows effective and efficient performance on both classification and semantic segmentation [1]. Soilan et al. extend the application of PointNet to the classification of The Current Elevation File Netherlands (AHN) aerial point clouds and achieve a good overall performance, but there are still problems like a high confusion of vegetation[1] and building classes [2], as shown in Figure 1.1. Considering that PointNet does not exploit local features, limiting its performance in case of 3D scenes with fine details, PointNet++ is then designed to apply PointNet on local regions inside the input point set [3]. By sampling the points in a hierarchical fashion and using PointNet recursively, PointNet++ is also able to combine multi-scale geometric features. However, exploiting spatial relations between points in a neighborhood is still an unsolved problem.

In images, convolutional neural networks (CNNs) can capture the spatially-local correlation effectively [4]. Inspired by the success in 2D image recognition, many CNN-based methods have also been proposed to extract the semantic meaning in 3D data (e.g., point clouds). However, directly convolving on features of the input points will cause permutation variance, bringing the problem of point ordering. Thus, many approaches rely on particular preprocessing techniques when adapting CNN to point clouds. Voxel-based methods extend CNN to 3D cases by first representing 3D shapes with 3D grids (voxels) and then applying 3D CNN

[1]The authors made a misktake when identifying the categories. Here "vegetation" actually refers to the "others" class in the AHN dataset.



(a) Ground Truth (b) Prediction

Figure 1.1: Segmentation results from Soilan et al. show a high confusion between vegetation and buildings, with vegetation shown in green and buildings in yellow [2].

naturally [5]. The voxels can be regular, or obtained with octree [6]. Although these methods are straightforward, a large consumption of memory is needed, especially when the voxels are acquired with small scales. Image-based approaches, however, avoid the usage of 3D CNN by projecting the point cloud onto a series of images that can be classified or segmented using 2D CNN [7]. This approach has also shown success on ALS point clouds. N. Qin et al. propose a network classifying 3D terrain scenes with images obtained from multi-view and multimodal representations [8]. Compared to methods directly operating on point cloud data, both voxel-based and image-based approaches require costly preprocessing and format conversion steps. Recently, graph-based CNN also gains much attention in computer vision domains. Graphs, as a topology structure, are capable of imposing the prior knowledge contained in the input data [9]. In case of 3D point clouds, constructing graphs, such as k nearest neighbor (k-NN) graphs, helps to encode spatial relations among points and capture the local features. Dynamic Graph CNN (DGCNN), proposed by Wang et al., further generates feature descriptors based on *edges* of the k-NN graph of a point cloud [10]. By applying convolution operations on these features, DGCNN accounts for the local features in an effective manner and achieves state-of-the-art performance in the semantic segmentation of point clouds.

In this work, we focus on the same AHN3 aerial point cloud dataset used by Soilan et al. [2] but employ DGCNN instead of PointNet. We also investigate how to incorporate multi-scale spatial context (i.e., context with different sizes of receptive field, or effective ranges in case of point clouds), which is proved to be helpful in semantic segmentation [11]. This can be done by processing input data from multiple scales together, or by consolidating the final segmentation results through exchanging information from different scales [12], which is adopted in this work. To achieve the best combination of different effective ranges, we first

explore their impact on the network performance by adjusting the block size and k used in DGCNN.

1.1 Research Questions

The aim of this study can be summarized as follows:

1. Can DGCNN also work well on the semantic segmentation of aerial point clouds (e.g., AHN3)?
2. How do different block sizes and k values used in DGCNN affect the segmentation results and how can we evaluate the performance?
3. Is it possible to improve the semantic segmentation quality by incorporating results from multiple scales?

Chapter 2

Methodology

2.1 DGCNN

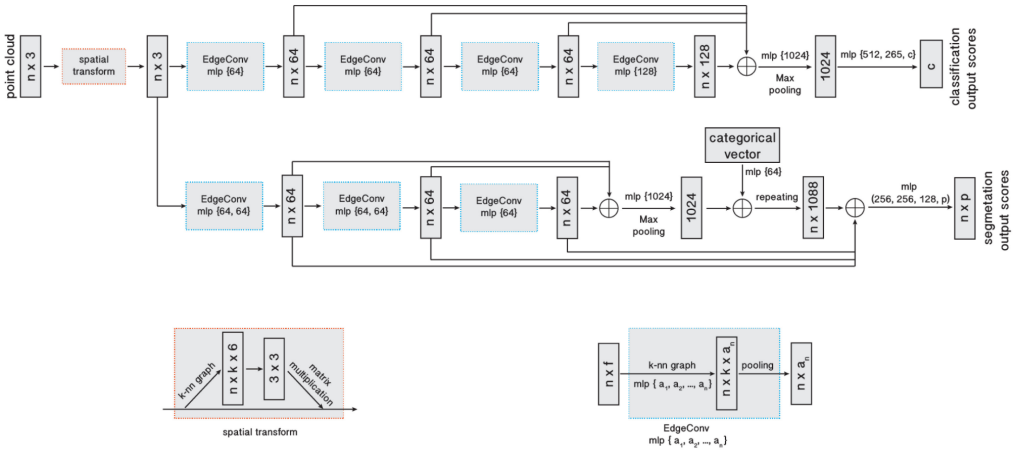
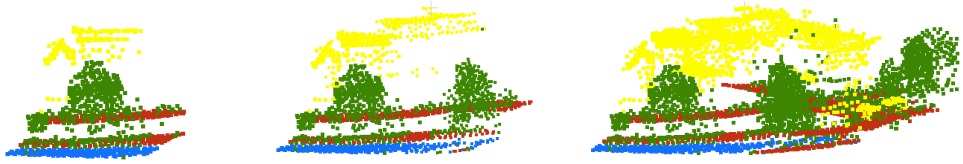


Figure 2.1: Overview of DGCNN architecture [10], with each input point containing 3 features (e.g., x, y and z). The **segmentation model** extends the classification one by combining the global feature vector with all local features generated by EdgeConv. p semantic labels will be output for each point. The **spatial transform** block aligns the input point cloud through a 3×3 matrix, which is expected to be an orthogonal matrix and estimated during training. In the **EdgeConv** operation, edge features for each point with a dimension f are computed by applying a MLP with a number of layer neurons defined as $\{a_1, a_2, ..., a_n\}$. After pooling among edge features, a tensor of shape $(n \times a_n)$ will finally be generated. $\oplus$: Concatenation.

DGCNN (Dynamic Graph CNN) is used for semantic segmentation of point clouds in this study, which takes a whole point set as input and outputs a semantic label for each point [10]. DGCNN uses a basic version of PointNet as the

backbone. Similar to PointNet, a point cloud needs to be split into 3D blocks with a certain block size (see Figure 2.2) before being fed into the neural network [1]. In training mode, N points $X = \{x_1, ..., x_N\} \in \mathbb{R}^F$, where F is the dimensionality of point features, are randomly sampled from a single block and put into the neural network. Afterwards, points from another block are processed until all blocks inside the point cloud are taken. In PointNet, a series of multi-layer perceptrons (MLP) will be applied on each point to obtain point features in a higher dimension. A symmetric function (e.g., $\sum$ or max pooling) is then used to convert features from all N points into a global feature vector, which shows invariance to input permutations. In the segmentation model, the global feature will be further concatenated with point features computed from each MLP to form a final combined vector, which will be used to predict p scores, with p the number of classes. PointNet can summarize both pointwise information and global context inside a single block. However, points are processed only individually in this network and local neighborhood information is not considered.



(a) block size = 30 m (b) block size = 50 m (c) block size = 70 m

Figure 2.2: Labeled point sets from AHN3, with different block sizes. Ground points are shown in brown, building points are in yellow, water points are in blue and points from others are in green.

Based on the architecture of PointNet, DGCNN incorporates local features by replacing the MLP with an edge convolution operation (EdgeConv). DGCNN first constructs a directed graph $G = (V, E)$ representing local structures inside a point cloud, where $V = \{1, ..., N\}$ indicates vertices and $E \in V \times V$ refers to edges. In the simplest case, G is the k-nearest neighbor (k-NN) graph of X in $\mathbb{R}^F$. Instead of convolving directly on point features, DGCNN first computes k edge features for each point x_i related to its k nearest neighbors $x_{j_{i1}}$, ..., $x_{j_{ik}}$, as shown in Figure 2.3. The edge feature is defined by $e_{ij} = h_\Theta(x_i, x_j)$, where x_j is one of the neighbors of x_i and h_Θ: $\mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^{F'}$ is a non-linear function with a set of learnable parameters Θ. In DGCNN, an asymmetric function $h_\Theta(x_i, x_j) = \bar{h}_\Theta(x_i, x_j - x_i)$ is adopted to account for both the global shape structure, indicated by x_i, and the local neighborhood information, captured by $x_j - x_i$.

Finally, the EdgeConv operation can be achieved by applying a channel-wise symmetric operation (max pooling in DGCNN) on the edge features associated with each vertex. In addition, the directed graph G is computed with all point

features and will be dynamically updated from layer to layer rather than taken as a fixed constant.

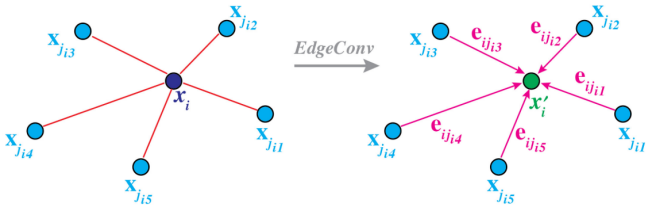


Figure 2.3: The EdgeConv operation [10]. Here the k-NN graph is constructed with $k = 5$. The output x_i' is calculated by aggregating all edge features associated with the vertex x_i through a symmetric function (e.g., max pooling).

2.2 Multi-scale Combination

In 2D cases, e.g., object detection and semantic segmentation using images, the size of the receptive field is crucial to the performance of CNNs [12], which will affect the spatial context of the network. When it comes to point clouds, the receptive field can be referred to as the effective range. In DGCNN, since the k-NN graph is dynamically updated in the feature space from layer to layer, points falling into the same category can belong to the same neighborhood even though they are spatially far away. Thus, it is difficult to compute the effective range using a simple equation. What we know for sure is that the effective range is limited by the block size, and affected by the size of the neighborhood of each point defined in k-NN graphs.

The local features and global features acquired by DGCNN only summarize the information inside a single block. For aerial point clouds, there is no apparent boundary like indoor scenes and objects in one block can be cut off, such as building roofs shown in Figure 2.2. Restricted by the block size, it is possible that there is only one object (or one kind of objects) in a single block, which might cause confusion during training, e.g., the ground and large building roofs can be mislabeled as each other in a small scale if both of them are flat. Enlarging the block size can reduce such mistakes, but detailed information acquired in small scales can be unseen in large scales if the number of points N in each block is unchanged. Besides, the size of the neighborhood (k) will directly influence the scope of the EdgeConv operation and the effective range.

To improve the semantic segmentation quality, it is helpful to cover information

from multiple scales. In this work, we achieve this goal by combining the point descriptors estimated by DGCNN. For two network outputs with different effective ranges (i.e., different block sizes, or different values of k), we first apply the softmax operation on them to obtain "probability" alike values for each class. Considering an output vector $x = \{x_0, x_1, ..., x_{p-1}\}$, the softmax function $\sigma(x)_m$ for the m^{th} class is:

$$\sigma(x)_m = \frac{e^{x_m}}{\sum_k e^{x_k}} \tag{2.1}$$

where $m \in \{0, 1, ..., p-1\}$ and p is the total number of classes [13]. For both point descriptors, if the largest probability value corresponds to the same class m, this point will be classified as m. Otherwise, the point will be labeled with the class which has a larger probability value. We explore several possible realizations of such combination and compare them experimentally in chapter 4.

2.3 Evaluation Measures

We evaluate the performance of all experiments with several metrics. The overall accuracy, computed as dividing the number of correctly classified points from all classes by the total number of predictions, is determined. Since the number of points in different classes are unbalanced, e.g., points from water are much less than ground, the average per class accuracy is also calculated. In addition, we evaluate the mean value of the intersection over union (IoU) from each class, which is a common metric used in semantic segmentation tasks. For point cloud data, IoU from one class is computed as:

$$IoU = \frac{TP}{TP + FP + FN} \tag{2.2}$$

where TP is the number of true positives, FP the number of false positives and FN the number of false negatives [14]. If we assume "building" as the positive class and "not building" as the negative class, a true positive indicates where the model correctly predicts the positive class and a false positive refers to where the model incorrectly predicts the positive class. Moreover, a true negative is an outcome where the model correctly predicts the negative class and a false negative is an outcome where the model incorrectly predicts the negative class. Similar to TP, FP and FN, there is also a measure TN indicating the number of true negatives.

In this work, we also determine the confusion matrix, which contains information of TP, FP, FN and TN from all classes. To quantitatively analyze the segmen-

tation results from each class, we use another three metrics, which are the recall, the precision and the F1 score. These metrics are computed as:

$$recall = \frac{TP}{TP + FN} \tag{2.3}$$

$$precision = \frac{TP}{TP + FP} \tag{2.4}$$

$$F1 = \frac{2 \times recall \times precision}{recall + precision} \tag{2.5}$$

From the above equations, it can be noticed that the recall indicates the proportion of points correctly classified by the model in all actual positive points, while the precision means the percentage of correctly classified points in all positive predictions. The recall can be thought of as a measure of completeness, and the precision is also called the exactness of a model. The F1 score conveys the balance between the precision and the recall.

Chapter 3

Dataset

The data used in this work comes from The Current Elevation File Netherlands (AHN), which contains detailed and precise altitude data for the whole Netherlands, including raster and 3D point cloud [15]. Classification labels have already been assigned to each individual point in the 3D point cloud. Each point belongs to one of the following classes: ground, buildings, water, bridges and others. The "others" class consists mostly of vegetation, and also objects like railways. In our study, we use point clouds from the iteration AHN3. Moreover, since points classified as bridges are much less than other categories, bridge points are merged into ground class for better training of DGCNN.

Figure 3.1: AHN3 sections selected for study, with training sections shown in purple and the test area in orange.

The whole AHN3 point cloud has been divided into smaller rectangular sections, with each of them covering a surface area of 6.25×5 km^2 and assigned with a unique ID. For training and test purposes, we select 4 sections from the original AHN3 dataset (see Figure 3.1), which are from the surroundings of Utrecht and Delft. Table 3.1 shows the total number of points in each section. Besides, the original dataset has an average point density of 20 points/m^2. To make it easier to handle the point cloud with a normal computer, each section is further split into 25 tiles and downsampled uniformly with a point interval of 1 m. In our work, 12 tiles from 38FN1, 37EN2 and 31HZ2 are finally used. In 32CN1, only 8 tiles are used since this section contains a large amount of vegetation, which

(a) Rendered with false color composed of return number, number of returns and intensity

(b) Rendered with RGB colors that are taken from aerial images acquired with another flight

(c) Labeled with ground (brown), building (yellow), water (blue) and others (green)

Figure 3.2: A subset of point cloud from the section 31HZ2 [15].

leads to much more points than other sections due to the multipath effect. The number of finally used points is also summarized in Table 3.1, with a total number about 109.3 million.

Section ID	# all points ($\cdot 10^6$)	# used points ($\cdot 10^6$)	Usage
38FN1	47.5	25.4	Training
37EN2	50.6	28.8	Training
32CN1	87.4	27.8	Training
31HZ2	55.9	27.4	Test

Table 3.1: Overview of used data from AHN3

Apart from x, y and z coordinates, all points from AHN3 are also provided with extra attributes, e.g., return number, intensity, GPS time, etc. 9 features are used in this work, including 3D coordinates (x, y, z), LiDAR features (return number, number of returns and intensity) and normalized coordinates (nx, ny and nz). The normalized x, y and z are obtained through subtracting central 3D coordinates of each tile, which can make the learning of the network more robust.

Chapter 4

Experiments

To investigate the feasibility of DGCNN on aerial point clouds and the influence of different effective ranges, we experiment with three block sizes (30, 50 and 70 m) and three k values (15, 20 and 25). To compare the performance with different settings, we first use $k = 20$ as the default neighborhood size and vary the block size. Afterwards, using the block size achieving the best segmentation result, the effect of different k is explored. Having results with the above **single-scale settings**, we further consolidate the segmentation through **multi-scale settings**, in which test results using different block sizes or k values are combined by comparing the output point descriptors from DGCNN. The choices of block size and k depend on performances of previous single-scale experiments.

4.1 Training Settings

For each experiment, 4096 points are randomly sampled from each block during training before being fed into the neural network. To ensure some overlap between different blocks, we use 1.5 as the sample rate for each point. In the test stage, the sample rate is set as 1.0 and all points in each block are used.

All our experiments are performed on a High Performance Computing (HPC) environment of Delft University of Technology, which consists of 26 computing nodes. We use one available NVIDIA GeoForce RTX 2080 Ti GPU in the cluster. During training, the batch size is 8, which means 8 blocks can be processed at the same time. However, we use a batch size of only 1 during test, since the number of points is different in each block when no random sampling is used and different blocks cannot be stacked together. Moreover, the network is optimized by the Adam optimizer with an initial learning rate of 0.001, which is suggested as default in DGCNN [10]. For all experiments, the model used in test is obtained by choosing the best model after training with 50 epochs.

4.2 Results and Analysis

4.2.1 Single-scale Settings

Table 4.1 summarizes the quantitative results of point cloud semantic segmentation over the test area. When k is fixed as 20 and different block sizes are investigated, the best overall accuracy (93.28%) and mean IoU (81.73%) are achieved with a block size of 50 m, while the highest average per class accuracy (90.90%) is achieved with the block size 70 m. It can also be noticed that the average per class accuracy and mean IoU with block size 30 m are much lower than the best results, although the overall accuracy under all three settings are similar, indicating that more balanced results among different classes can be obtained when the block size is larger. Considering the recall and precision values shown in Table 4.3 and Table 4.4, segmentation results from each individual class are better when the block size is 50 m. Thus, for experiments with different k values, a block size of 50 m is finally used.

In the middle part of Table 4.1, comparisons between different k values are provided. Similar to experiments with the block size, larger k values, which correspond to a larger neighborhood, help to improve the performance of the model. Besides, results from $k = 20$ and $k = 25$ are close to each other.

Block size (m)	k	OA (%)	PA (%)	mIoU (%)
30	20	91.72	81.53	74.94
50	20	**93.28**	89.39	**81.73**
70	20	92.97	**90.90**	79.51
50	15	92.38	88.51	79.98
50	20	93.28	**89.39**	81.73
50	25	**93.30**	89.01	**82.10**
50 & 30	20	**93.51**	**91.60**	82.34
50	15 & 20	93.37	90.48	**82.46**

Table 4.1: Comparison of the overall accuracy (OA), the average per class accuracy (PA) and the mean IoU (mIOU) with different block sizes and k values. "50 & 30" and "15 & 20" in the lowest part indicate results with multi-scale combinations.

Confusion matrices from the above experiments are shown from Table 4.2 to Table 4.6. Points from others and ground classes are identified well in all cases, with high values in both recall and precision. This is not surprising considering points in both classes are way more than other categories. When the network processes point clouds in a smaller scale (e.g., the block size is 30 m or $k = 15$),

block size = 30 m $k = 20$		Prediction				Recall (%)
		others	ground	building	water	
Gt	others	12605814	377425	263017	3993	95.14
	ground	130083	8242652	72551	8205	97.51
	building	1154213	215135	4187805	497	75.35
	water	4171	35463	9	55021	58.21
Precision (%)		90.73	92.92	92.58	81.25	

Table 4.2: Confusion matrix of segmentation results with block size = 30 m and $k = 20$ (Gt: Ground truth)

block size = 50 m $k = 20$		Prediction				Recall (%)
		others	ground	building	water	
Gt	others	12491522	373611	380528	4485	94.27
	ground	101575	8207967	131128	12932	97.09
	building	725797	91307	4740282	140	85.29
	water	1244	16794	68	76723	80.90
Precision (%)		93.78	94.46	90.26	81.38	

Table 4.3: Confusion matrix of segmentation results with block size = 50 m and $k = 20$

block size = 70 m $k = 20$		Prediction				Recall (%)
		others	ground	building	water	
Gt	others	12286168	85676	635932	622	94.45
	ground	393010	8265197	106283	31115	93.97
	building	565762	95030	4815031	2	87.93
	water	2199	6800	226	63099	87.24
Precision (%)		92.75	97.78	86.64	66.53	

Table 4.4: Confusion matrix of segmentation results with block size = 70 m and $k = 20$

block size = 50 m $k = 15$		Prediction				Recall (%)
		others	ground	building	water	
Gt	others	12449649	505497	288765	6235	93.96
	ground	77090	8270088	91649	14775	97.83
	building	904223	177600	4475384	319	80.53
	water	1060	16248	16	77505	81.73
Precision (%)		92.96	92.20	92.17	78.42	

Table 4.5: Confusion matrix of segmentation results with block size = 50 m and $k = 15$

block size = 50 m $k = 25$		Prediction				Recall (%)
		others	ground	building	water	
Gt	others	12485661	382648	377977	3860	94.23
	ground	95490	8264229	84858	9025	97.76
	building	697189	161876	4698144	317	84.54
	water	682	18564	168	75415	79.53
Precision (%)		94.03	93.62	91.03	85.10	

Table 4.6: Confusion matrix of segmentation results with block size = 50 m and $k = 25$

the predicted building points have a much higher precision than the recall, which means the model misses a large number of points from this class, although most predictions are correct. For the water class, when the block size is very large (70 m) or the network looks at a small neighborhood ($k = 15$), the precision can be worse than the recall rate, indicating that the model is not accurate enough for detecting water points in this case. Since the total number of water points is much less than other classes, the segmentation of water can be tricky. Only when the block size is 50 m and $k = 20$, the recall rate and precision show balanced high values.

4.2.2 Multi-scale Settings

Apparently, the segmentation results are improved when we combine results with different block sizes or k values, compared to the original single-scale settings, as shown in Table 4.1. An average per class accuracy of 91.60% can be achieved when combining results with the block size as 30 m and 50 m, which is the best result in all experiments. In addition, $k = 20$ is chosen in this case since $k = 25$ means much more parameters and brings the overfitting problem, although we achieve similar results under both single-scale settings in subsection 4.2.1.

block size = 30 & 50 m $k = 20$		Prediction				Recall (%)
		others	ground	building	water	
Gt	others	12611735	87874	843299	1807	93.11
	ground	372161	8281938	104242	18251	94.36
	building	261449	74927	4609489	4	93.20
	water	4031	8141	240	74591	85.73
Precision (%)		95.19	97.98	82.95	78.80	

Table 4.7: Confusion matrix of combined segmentation results (block size = 30 m & 50 m), with $k = 20$

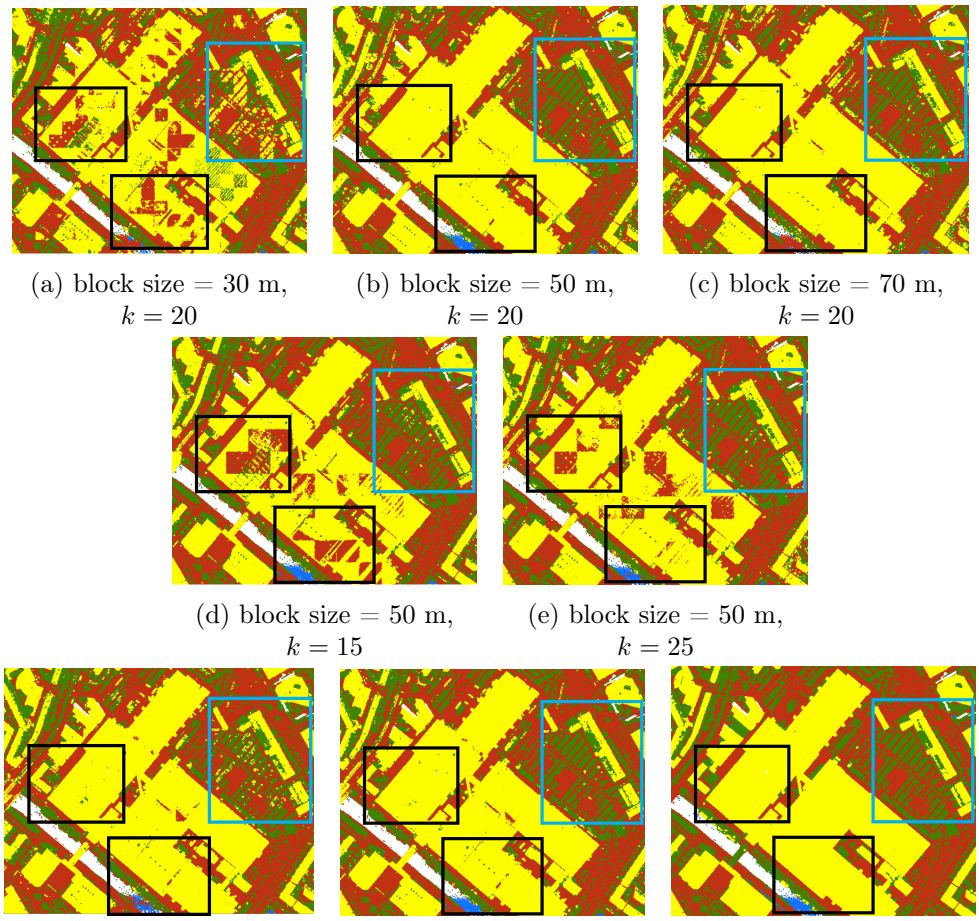
block size = 50 m k = 15 & 20		Prediction				Recall (%)
		others	ground	building	water	
Gt	others	12518431	75101	761904	897	93.73
	ground	431562	8252793	105192	15019	93.73
	building	294763	113751	4689956	13	91.99
	water	4889	11636	247	78900	82.47
Precision (%)		94.48	97.63	84.39	83.20	

Table 4.8: Confusion matrix of combined segmentation results (k = 15 & 20), with the block size as 50 m

Table 4.8 shows that combining different k values helps to achieve higher and more balanced values of the precision and recall of the water class. However, the segmentation result is not always better under multi-scale settings for all classes. For detected building points, the precision values in Table 4.7 and Table 4.8 are much worse than the corresponding single-scale settings, while the recall rates are increased. This might be caused by some incorrect but confident predictions with small scales (i.e., block size = 30 m or $k = 15$), which have higher "probability" values in the output descriptor of DGCNN.

Figure 4.1, Figure 4.2 and Figure 4.3 illustrate the segmentation results with different **single-scale settings** and **multi-scale settings** from one tile of the test area. Compared to large scales, smaller block size or k can cause some confusion to the classification of building points, e.g., the middle of the large roof can be detected as ground points (see black boxed areas in Figure 4.1). When the block size is 30 m, points from others can also be mislabeled as buildings, as shown in the blue boxed area in Figure 4.1a. Figure 4.3a and Figure 4.3d also indicate that points on building facades can be labeled as others (trees). With small scales, there exist some "block effect" in the test results, which means the edges of some blocks can be clearly seen in the visualization of the segmentation. Most of these problems can be solved when we increase the block size or the neighborhood size k[1]. Beyond that, smaller scales also show advantages in the segmentation of areas, where objects from different classes are highly mixed (see Figure 4.2a, which indicates the central station in Utrecht, and Figure 4.2d).

[1]The "block effect" is not removed with $k = 25$, which might be due to the randomness of the test, or the overfitting problem of the model when there are much more parameters.



(a) block size = 30 m, $k = 20$

(b) block size = 50 m, $k = 20$

(c) block size = 70 m, $k = 20$

(d) block size = 50 m, $k = 15$

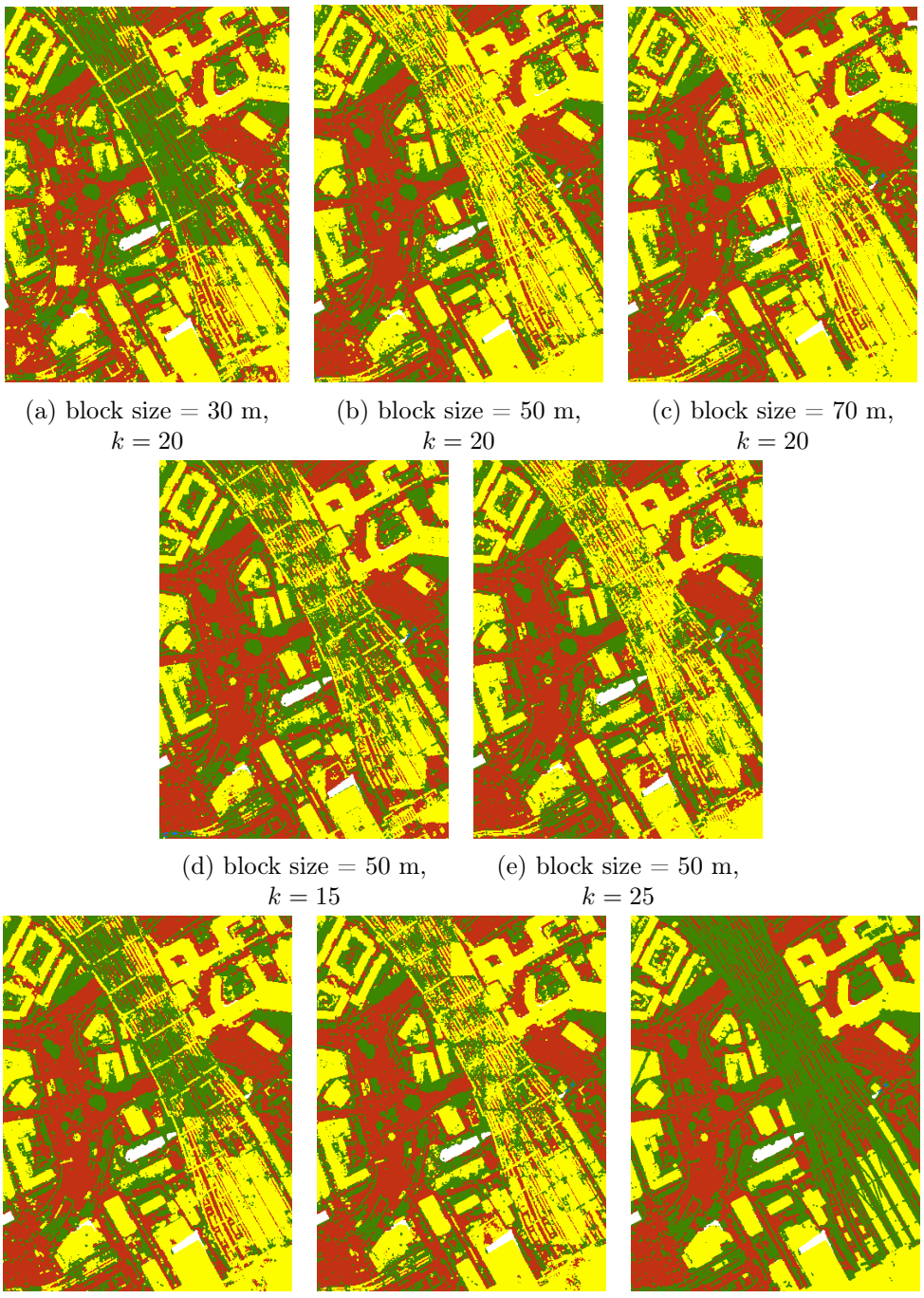
(e) block size = 50 m, $k = 25$

(f) block size = 30 & 50 m, $k = 20$

(g) block size = 50 m, $k = 15$ & 20

(h) Ground Truth

Figure 4.1: Segmentation results from a tile in the test area, with ground points in brown, building points in yellow, water points in blue and points from others in green. With small scales, confusion of building and ground classes can be seen in black boxed areas. A large number of points from others are labeled as buildings when the block size is 30 m, as indicated by the blue boxed area.



(a) block size = 30 m, $k = 20$

(b) block size = 50 m, $k = 20$

(c) block size = 70 m, $k = 20$

(d) block size = 50 m, $k = 15$

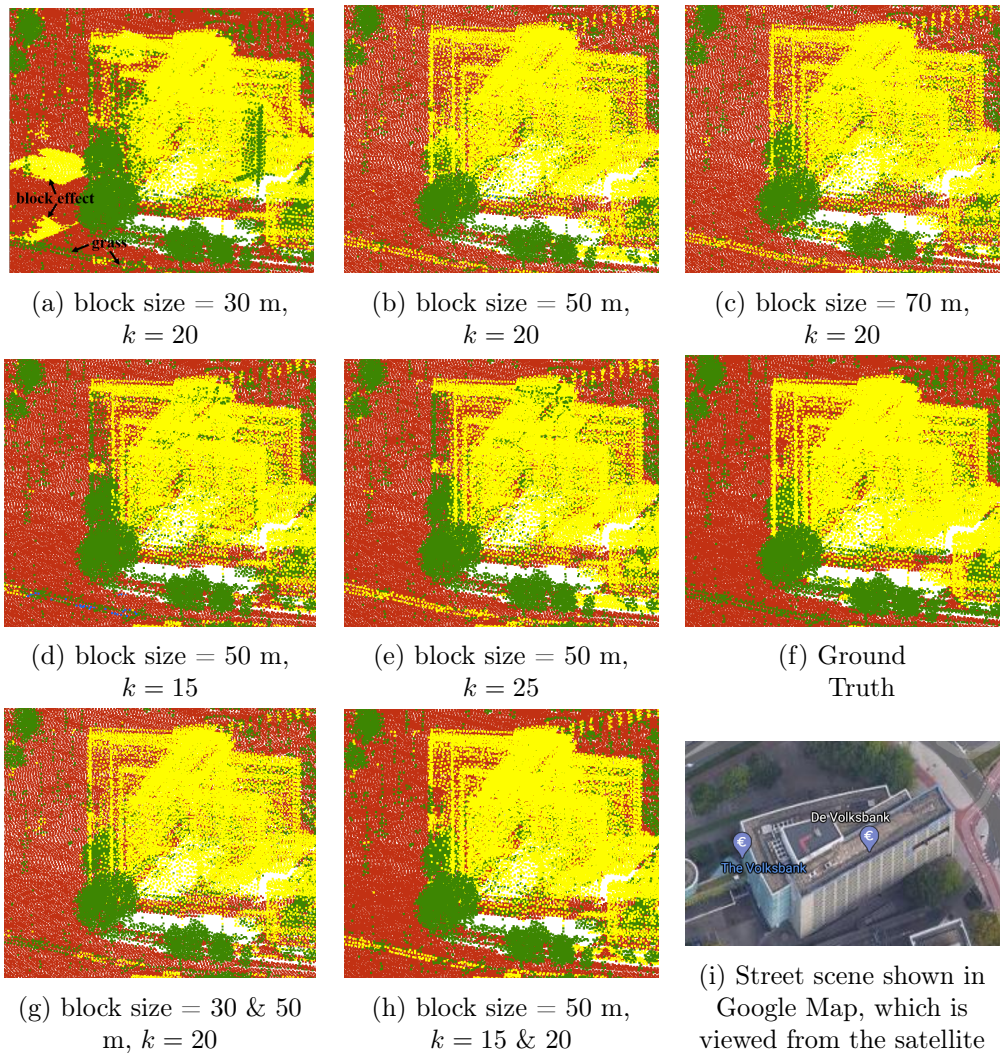
(e) block size = 50 m, $k = 25$

(f) block size = 30 & 50 m, $k = 20$

(g) block size = 50 m, $k = 15$ & 20

(h) Ground Truth

Figure 4.2: Segmentation results around Utrecht central station in the test area, with ground points in brown, building points in yellow, water points in blue and points from others in green. With small scales, highly mixed points from ground and others are easier to distinguish.



(a) block size = 30 m, $k = 20$

(b) block size = 50 m, $k = 20$

(c) block size = 70 m, $k = 20$

(d) block size = 50 m, $k = 15$

(e) block size = 50 m, $k = 25$

(f) Ground Truth

(g) block size = 30 & 50 m, $k = 20$

(h) block size = 50 m, $k = 15$ & 20

(i) Street scene shown in Google Map, which is viewed from the satellite

Figure 4.3: A point cloud subset in 3D view. With smaller scales (i.e., block size = 30 m or $k = 15$), many points from building facades are detected as trees from "others".

When we combine results from two single-scale settings, drawbacks from both scales can be mitigated by taking more confident predictions. For example, when the block size 30 m and 50 m are combined (see Figure 4.3g), much less points on building facades are detected as trees and the "block effect" is removed compared to Figure 4.3a, which is achieved with the block size 30 m. Additionally, grass points shown in Figure 4.3b (block size = 50 m) are mislabeled as buildings, which is also largely corrected in Figure 4.3g.

Table 4.9 also summarizes F1 scores from different classes in all experiments. For

the ground class, the F1 score varies not much with the block size or k value. Increasing the scale helps to detect points from buildings and others. The F1 score varies the most with different scales for the water class and larger scales do not always improve the segmentation. When the block size is too large, the F1 score even drops a lot. For water points, the model performs the best when the block size is 50 m and $k = 25$. Moreover, combining results from different scales improve the F1 score, but not very much.

Block size (m)	k	F1 score (%)			
		others	ground	building	water
30	20	92.88	95.16	83.08	67.77
50	20	94.03	95.76	**87.71**	81.14
70	20	93.59	**95.84**	87.28	75.49
50	15	93.32	94.93	85.95	80.04
50	25	**94.13**	95.65	87.66	**82.22**
30 & 50	20	94.14	96.14	87.78	82.12
50	15 & 20	94.10	95.64	88.03	82.83

Table 4.9: Comparison of per class F1 scores under different single-scale and multi-scale settings

Chapter 5

Conclusion

This work investigates the feasibility of DGCNN on the semantic segmentation of ANH3 point clouds using 3D coordinates and LiDAR features (return number, number of returns and intensity), and discusses how different choices of the block size and the neighborhood size k in k-NN graphs affect the segmentation results.

Three block sizes (30 m, 50 m and 70 m) and three k values (15, 20 and 25) are studied. The best overall accuracy (93.30%) and mean IoU (82.10%) are achieved when the block size is 50 m and $k = 25$, while the highest average per class accuracy is obtained with a block size of 70 m and $k = 20$. It can be concluded that increasing the block size or k is beneficial for the overall performance of DGCNN on AHN3 point clouds. For each individual class, i.e., ground, building, water and others, we also discover that a certain single-scale setting cannot ensure the best segmentation results from all classes. However, we can achieve a balanced and good F1 scores for all kinds of objects when both values of the block size and k are appropriate (e.g., when the block size is 50 m and $k = 20$ or 25). Besides, combining segmentation results from single-scale settings helps to consolidate the overall performance of the model by exploiting advantages from both settings, although confusion for some classes (e.g., buildings and vegetation from "others") can also be caused.

Chapter 6

Recommendations

In our experiments, DGCNN is proved to be an effective way of classifying aerial point clouds, but there are still many problems indicating future research and further investigation. First, during test, the number of points inside a block is largely increased when we set the block size as 70 m, resulting in the out-of-memory problem in relevant experiments. Thus, although using a larger block size can improve the overall accuracy, the best choice is also limited by the hardware. Second, we downsampled the original AHN3 point clouds uniformly before applying DGCNN, which lowers the point density and causes certain loss of details. Since downsampling is necessary in some cases considering the memory usage, it might be significant to improve the sampling method at the same time, e.g., sampling with octree to account for different detail levels within a point cloud. In this way, it is possible to add object-level context such as more detailed division of tree and building facade points. Similar problems also happen during training. The block usage of DGCNN can cause artefacts in the segmentation results, which means in some area the edges of the blocks are very obvious. This problem can be less serious using a larger block size, but in this case the point density inside a block is harmed due to random sampling. It is beneficial to apply more refined sampling technique in each block, or to sample for several times. Third, using a larger k value increases the overall performance of the model, but the model can be overfitted when k is too large (e.g., $k = 25$). This can be explained by the usage of much more parameters, and the more overlapping neighborhood of different points. We could account for more points in a block when increasing the neighborhood size. Additionally, in multi-scale combinations, some predictions at one scale are correct but less "confident" compared to the other. A correction method should be found. We could also consider using boosting strategy to improve the result, or incorporating the multi-scale combination inside the network. Lastly, we notice that the "others" class of AHN3 dataset contains points from grass, trees, railways, etc., which actually are distinct in features like height (z coordinate). We suggest a finer division of classes in the next iteration of AHN point cloud data, which might improve the performance of network further. For the future research, it will also be beneficial to apply DGCNN on aerial point clouds with more complex categories.

Bibliography

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[2] M. Soilán Rodríguez, R. Lindenbergh, B. Riveiro Rodríguez, A. Sánchez Rodríguez *et al.*, "Pointnet for the automatic classification of aerial point clouds," 2019.

[3] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108.

[4] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," in *Advances in neural information processing systems*, 2018, pp. 820–830.

[5] D. Griffiths and J. Boehm, "A review on deep learning techniques for 3d sensed data classification," *Remote Sensing*, vol. 11, no. 12, p. 1499, 2019.

[6] G. Riegler, A. Osman Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3577–3586.

[7] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[8] N. Qin, X. Hu, and H. Dai, "Deep fusion of multi-view and multimodal representation of als point cloud for 3d terrain scene recognition," *ISPRS journal of photogrammetry and remote sensing*, vol. 143, pp. 205–212, 2018.

[9] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv preprint arXiv:1812.08434*, 2018.

[10] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[11] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 891–898.

[12] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3d semantic segmentation of point clouds," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 716–724.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[14] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[15] "Pdok - anh3 downloads," https://downloads.pdok.nl/ahn3-downloadpage/.