Article

Semantic Segmentation of Large-Scale Outdoor Point Clouds by Encoder–Decoder Shared MLPs with Multiple Losses

Beanbonyka Rim ¹ , Ahyoung Lee ² and Min Hong ^{3,*} ¹ Department of Software Convergence, Soonchunhyang University, Asan 31538, Korea; rim.beanbonyka@sch.ac.kr² Department of Computer Science, Kennesaw State University, Marietta, GA 30144, USA; alee146@kennesaw.edu³ Department of Computer Software Engineering, Soonchunhyang University, Asan 31538, Korea

* Correspondence: mhong@sch.ac.kr

Abstract: Semantic segmentation of large-scale outdoor 3D LiDAR point clouds becomes essential to understand the scene environment in various applications, such as geometry mapping, autonomous driving, and more. With an advantage of being a 3D metric space, 3D LiDAR point clouds, on the other hand, pose a challenge for a deep learning approach, due to their unstructured, unordered, irregular, and large-scale characteristics. Therefore, this paper presents an encoder–decoder shared multi-layer perceptron (MLP) with multiple losses, to address an issue of this semantic segmentation. The challenge rises a trade-off between efficiency and effectiveness in performance. To balance this trade-off, we proposed common mechanisms, which is simple and yet effective, by defining a random point sampling layer, an attention-based pooling layer, and a summation of multiple losses integrated with the encoder–decoder shared MLPs method for the large-scale outdoor point clouds semantic segmentation. We conducted our experiments on the following two large-scale benchmark datasets: Toronto-3D and DALES dataset. Our experimental results achieved an overall accuracy (OA) and a mean intersection over union (mIoU) of both the Toronto-3D dataset, with 83.60% and 71.03%, and the DALES dataset, with 76.43% and 59.52%, respectively. Additionally, our proposed method performed a few numbers of parameters of the model, and faster than PointNet++ by about three times during inferencing.

Keywords: semantic segmentation; 3D LiDAR point clouds; deep learning; remote sensing

Citation: Rim, B.; Lee, A.; Hong, M. Semantic Segmentation of Large-Scale Outdoor Point Clouds by Encoder–Decoder Shared MLPs with Multiple Losses. *Remote Sens.* **2021**, *13*, 3121. <https://doi.org/10.3390/rs13163121>

Academic Editors: Thien Huynh-The and Sun Le

Received: 8 July 2021

Accepted: 3 August 2021

Published: 6 August 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The 3D LiDAR point clouds have become one of the most significant 3D data presentations for depth information, and have been deployed in various applications, such as urban geometry mapping, autonomous driving, virtual reality, augmented reality, and more [1–3]. Point cloud is a set of points in a 3D metric space, which provides rich 3D information, such as geometry, color, intensity, normal, and more, to accurately measure the surrounding objects. This information can be utilized for scene understanding. Among tasks that are related to the point cloud scene understanding, a semantic segmentation is a task that has the role of assigning each point to a meaningful label. This means that it does not only tell the location of the object, but it also describes what kind of object is in the scene. In this paper, we propose an outdoor 3D point clouds semantic segmentation.

The deep learning (DL) approach has proved to have an outstanding performance on classification, detection, and segmentation on 2D images [4]. Compared to the 2D images [5–7], point clouds of the outdoor scene are formed in the following properties [1,2]: (1) the points are unstructured, because they are not arranged in a regular grid, and are generally sparse in the 3D world space; (2) they are irregular, because the density of the point coordinates is not uniform, and they generally vary with the distance to the sensor; (3) they are unordered, because the order of storing information of points in the dataset

file does not affect the scene representation; (4) the points are usually collected in a large volume. These properties are potentially challenging for the DL approach to process them directly.

Previous works have proposed several DL approaches to label semantic information to point clouds, such as 2D projection [8], voxelization [9], point-wise multi-layer perceptron (MLP) [10–20], point convolution methods [21–27], and graph-based methods [28–36]. However, those methods confront a trade-off between high accuracy and low computation complexity. The point convolution and the graph-based methods achieve a significant accuracy, but consume a lot of memory and time. On the other hand, the point-wise MLPs methods are vice-versa. The memory and time consumption are also critical when the method is deployed as a real-time application. Therefore, our goal is to define a suitable method to balance this trade-off.

In this paper, inspired by the point-wise MLPs method, we propose an encoder–decoder shared MLP with multiple losses for large-scale outdoor point clouds semantic segmentation. We adopt the point-wise MLP method [11] as a base network. Specifically, we aim for common strategies to segment the large-scale outdoor scene simply, and yet significantly, with a low computational cost. First, we propose random point sampling (RPS), in believing that it speeds up the process of representative point selection. Second, we propose attention-based pooling, which can aggregate the features with attention-weighted summation, to capture the meaningful features. Third, we propose multiple losses, which can regulate the features space to receive a better result than a single loss. Finally, we experiment with our proposed method on the following two large-scale outdoor benchmark datasets: Toronto-3D [37] and DALES [38].

Our contributions are as follows:

- We propose a simple, and yet effective, strategy of the above aforementioned mechanisms, such as a random point sampling, attention-based pooling, and multiple losses summation integrated with the encoder–decoder shared MLPs method, for the large-scale outdoor point clouds semantic segmentation;
- We proof that our method performs good results and has a lower computational cost than PointNet++ [11].

The remainder of the paper is organized as follows. We review previous methods using DL methods in Section 2. Then, we describe our proposed method in Section 3. Next, we describe our experimental setup and analyze our results in Sections 4 and 5, respectively. In Section 6, we discuss and compare our experimental results with other methods. Finally, Section 7 concludes our study.

2. Related Works

Prior works are 2D projection [8] and voxelization [9]. The approach of 2D projection [8] is first to project 3D point clouds, from multiple views, into a collection of 2D images, and then utilize the mature structured 2D CNN. Such projection is simple, but loses 3D geometric information. Thus, it is not suitable for the scene semantic segmentation. Another approach [9] is to voxelize the point clouds into a regular 3D grid, and then utilize the structured 3D CNN. Such voxelizing consumes a lot of computational cost for large-scale dense 3D data, because the computation and memory usage will grow cubically when the data is scaled up. Thus, it is not suitable for the outdoor scene semantic segmentation. Later, the point-based DL approach has been proposed, in which the approach tends to feed the raw point clouds into the DL network directly. The prior works of the point-based DL approach are point-wise multi-layer perceptrons (MLPs) [10–20], in which the methods learn per-point features using shared MLPs as a base network to receive high efficiency. Such shared MLPs extract point-wise features independently and lose point relation information. Therefore, to capture more rich local and global structures, several integrated mechanisms have been introduced. Those include neighborhoods sampling, attention-based pooling, and local–global feature aggregation. Point convolution methods [21–27] tend to define a point-wise convolution operator, to permute local points into canonical

order. The discrete convolution operator uses points to carry kernel weights, but it loses neighboring information. Therefore, a 3D point continuous convolution operator is proposed. However, there are challenges in defining the point convolution operator, learning an accurate permuted matrix, and reducing the computational cost at the preprocessing step. Graph-based methods [28–36] structure point clouds as a super graph, to extract local shape information from the neighbors and feed this to a graph convolution network. Such a graph-based method uses point relationships that are defined as edges, and interpolates them into the network. However, it is not easy for the interpolation function to define a spatial extend of the neighborhood, to extract sufficient local and global point features.

We are interested in DL approaches on raw point clouds. Therefore, we review semantic segmentation methods on point-based DL approaches in detail, as follows.

2.1. Point-Wise MLPs Method

The point-wise MLPs method learns per-point features, using shared MLPs as a base network to receive high efficiency. The method is pioneeringly proposed by PointNet [10]. The shared MLPs learns raw point clouds to extract point-wise features, and then global max pooling aggregates the features of all the points. However, each point is learned individually, and the point relation is not considered. Therefore, to capture more rich local and global structures, PointNet++ [11] proposed hierarchical representation learning, with a farthest point sampling (FPS) layer, a K -nearest neighbor grouping layer, and a max pooling layer. PointSIFT [12] proposed an orientation-encoding module, integrated with shared MLPs. The module captures eight crucial orientation information points of point clouds, to achieve strong scale awareness. Engelmann et al. [13] defined a pairwise distance loss and centroid loss mechanism to the shared MLPs, for better structuring of the learned point feature space. Further, 3DContextNet [14] proposed the Kd-tree structure, to learn representative features progressively. The method enhances the computation cost by learning on implicit partition space and skipping the learning on empty space. ShapeContextNet [15] proposed a shape context kernel, to learn features in concentric shells and aggregate the features using dot-product self-attention for sufficient capturing of both local and global shape information. PointWeb [16] proposed the adaptive feature adjustment (AFA) module, to enhance the local neighborhood features. PAT [17] proposed the group shuffle attention (GSA) module, with self-attention and gumbel subset sampling. The module tends to define a stronger representative feature, with a lower computation cost. LSANet [18] proposed a local spatial awareness layer to learn spatially oriented distribution weights, and integrates it into the shared MLPs. ShellNet [19] proposed a shell convolution, to learn representative features from concentric spherical shell statistics. The method tends to enlarge the receptive fields with less deep layers. RandLA-Net [20] proposed the local spatial-encoding module, to randomly select representative features and attentive pooling modules, to aggregate features using the attention mechanism. The method tends to effectively extract features from a wide neighborhood, with a low computation cost.

2.2. Point Convolution Method

The point convolution method tends to define a point-wise convolution operator, to permute local points into canonical order. PointCNN [21] proposed the X-transformed convolution operator, which can weight input features and permute points into a latent and canonical order. PCCN [22] proposed a continuous convolution operator, which is parameterized by MLPs and tends to support the full continuous vector space. A-CNN [23] proposed an annular convolution operator, which can learn to capture the local feature correlation through a ring-shaped structure and direction. ConvPoint [24] proposed a continuous convolution operation. This method learns the weighted summation from the feature convolution operation and simple MLPs operation of spatial features. KPConv [25] proposed a kernel point convolution operator. The convolution takes radius neighborhoods as the input and learns the weights that are located in the Euclidean space, by a subset of kernel points. DPC [26] proposed a dilated point convolution operator. The convolution

learns the kernel weights over the dilated neighborhood. This method tends to increase the receptive field size of the point convolution. InterpCNN [27] proposed an interpolated convolution operator, to define a permutation and sparsity invariant of the points. The interpolation function takes a set of discrete kernel weights and interpolates the point features into the neighboring kernel weights.

2.3. Graph-Based Method

Graph-based method structures the local shape information by the edges of the graph defined on the point relations, before feeding it into the convolution network. DGCNN [28] proposed an edge convolution, to learn global shape properties by incorporating local neighborhood information. SPG [29] proposed a super point graph, to structure the point clouds into a collection of interconnected shapes. The graph represents edge features of each object's points. Then, the graph is fed into PointNet [10] for embedding, before feeding it to gated recurrent unit (GRU) for the final prediction. Ladrieu et al. [30] proposed a graph-structured contrastive loss, which can learn to embed local points. The local points embedder is a lightweight model that has been adopted from PointNet [10]. GACNet [31] proposed a graph attention convolution. The method tends to properly assign attention weights to different neighboring points, based on both points spatial geometry and features. PAG [32] defined edge-preserved pooling, edge-preserved unpooling, and point atrous convolution. The model is used to exploit multi-scale edge features by introducing a sampling rate parameter to enlarge the receptive fields. HDGCN [33] proposed a depth-wise graph convolution to aggregate the feature channel wisely, and point-wise graph convolution to learn the features across the different channels. Jiang et al. [34] proposed a hierarchical graph framework, to incorporate the point branch and edge branch. The edge branch is used to integrate the point features and generate edge features. Lei et al. [35] proposed a spherical convolution network, to learn point neighborhoods through the spherical convolution kernel. The spherical convolution kernel extracts local features by an octree partitioning structure. DPAM [36] proposed a graph convolution network that can learn a soft points cluster agglomeration by exploring the relation of points in the semantic space. The method tends to define a learning architecture to sample and group the points dynamically.

3. Methodology

In this paper, we propose an encoder–decoder shared MLP with multiple losses, for the large-scale outdoor point clouds semantic segmentation. We adopt the network architecture of PointNet++ [11] as a base network. The PointNet++ [11] is a hierarchical feature, learning through sampling, grouping, shared MLPs, and a pooling layer. We propose random point sampling and attention-based pooling, while PointNet++ [11] used farthest point sampling and max pooling, respectively. However, we fully adopt the K -nearest neighbor grouping and shared MLPs from PointNet++ [11]. Additionally, we propose a summation of multiple loss scores to help to refine the learned feature structure.

Our proposed network architecture is composed of a feature encoding and a decoding network. The raw point clouds are fed into the feature encoding network to extract local neighboring features. Then, the learned local neighboring features are up-pooled to achieve up-pooled features. The up-pooled features are concatenated with the skip linked features to generate propagated features for semantic point labeling. We describe, in more detail, our complete network, feature encoding network, feature decoding network, and multiple losses summation architecture in Section 3.1, Section 3.2, Section 3.3 and Section 3.4, respectively.

3.1. Network Architecture

Our complete network architecture is shown in Figure 1. The network architecture is an encoder–decoder shared MLP with multiple losses. Firstly, the raw point clouds are fed into the feature encoding network. The feature encoding network consists of the set

abstraction (SA) [11] modules, followed by the attention-based pooling (AP) layers [20,39]. The SA and AP modules are used to down-pool the point features for four levels, to extract high-level local feature abstraction. We enlarge the receptive fields of shared MLPs from 32 to 512 filter sizes, to widely capture the important information.

Figure 1. Our proposed network architecture. Left: feature encoding consists of the set abstraction (SA) modules followed by the attention-based pooling (AP) layers. Right: feature decoding consists of the feature propagation (FP) modules followed by fully connected (FC) layers. The final loss is the summation of four losses. Abbreviation: MLP, multi-layer perceptron model; C, number of categories.

Then, the learned local neighboring features are up-pooled to achieve up-pooled features. The up-pooled features are concatenated with the skip linked features, to generate propagated features for semantic point labeling. The feature decoding network consists of four levels of the feature propagation (FP) modules, followed by the fully connected (FC) layers. We narrow the filter size from 256 to 128. The FC layers condense the filters of size 256 and 128 into the number of categories (C). The filter size of the FC layer is defined with respect to the filter size of the shared MLPs from the FP module. Finally, we sum the four loss scores to receive the final loss, to help to refine the structure of the learned features space.

3.2. Feature Encoding Network

To extract the local features, we adopt the set abstraction module (SA) [11] and attention-based pooling [20,39] for the features encoding, as shown in Figure 2. The SA module [11] learns the point clouds, to encode the local features through sampling, grouping, shared MLP, and a pooling layer. In the sampling layer, we propose random point sampling (RPS), to randomly select representative points. Then, in the grouping layer, the K -nearest neighbor method finds the K nearest points within a radius of each sampled point. Finally, shared MLPs with the attention-based pooling layer aggregate features of the K nearest points that are within the same radius as the local neighboring features.

An input point cloud is represented as $p_i | i = \{1, 2, \dots, n\}$, with $p_i \in \mathbb{R}^F$, where F is the features of the input points, such as xyz coordinates, color, normal, etc.

3.2.1. Sampling Layer

From the set of N input features of dimensionality F , we randomly select the subset of point N . Compared with the farthest point sampling (FPS), the random point sampling (RPS) has less coverage of the entire point set than FPS. However, RPS achieves a low computational cost with $O(1)$, which is suitable for large-scale points. We simply compute the random point sampling with the Python numPy package. We use `numpy.random.choice()` to generate the indices. Then, we select the corresponding point features through these indices.

Figure 2. Our proposed feature encoding network architecture. Left: set abstraction (SA) module consists of sampling, grouping and shared multi-layer perceptron (MLPs) layer. Right: attention-based pooling (AP) layer.

3.2.2. Grouping Layer

Since point clouds are not arranged in a regular grid, similarly to 2D images, computing the neighborhoods allows us to capture the local point relationships. The neighboring features vary depending on the sparsity of the point set. Therefore, we adopt the grouping layer from [11], to compute the neighboring features as follows. For the i^{th} point, we firstly query its neighboring points within a radius of each sampled point, using the K -nearest neighbor method. The K -nearest neighbor method computes based on point-wise Euclidean distances. The K value is set to 32. Then, we compute the relative point position by concatenating the features of each point with its neighboring features. For each of the K neighbor points $\{p_i^1, \dots, p_i^k, \dots, p_i^K\}$ to the centroid point p_i , the relative point position is computed as follows:

$$r_i^k = p_i \oplus (p_i - p_i^k), \quad (1)$$

where p_i and p_i^k are the xyz coordinates of points, $\oplus$ is the concatenate operation, and r_i^k is the relative point position with $r_i^k \in \mathbb{R}^F$.

3.2.3. Shared MLPs Layer

We learn a representation of this relative point position by the point-wise shared multi-layer perceptrons (MLPs). For each of K point positions $\{r_i^1, \dots, r_i^k, \dots, r_i^K\}$ relative to the centroid point P_i , we encode the relative point position as follows:

$$f_i^k = \text{MLP}(r_i^k), \quad (2)$$

where r_i^k is the relative point position and f_i^k is the corresponding learned local features with $f_i^k \in \mathbb{R}^F$.

3.2.4. Attention-Based Pooling Layer

The pooling layer is used to aggregate the set of neighboring point features f_i^k . The PointNet++ [11] used max pooling to hard aggregate the neighboring features. However, it results in losing a lot of information. Inspired by [20,39], we adopted the attention-based pooling layer, which is a robust pooling mechanism used to automatically learn meaningful local features through attention-weighted scores and its summation. Firstly, we compute the attention-weighted scores. From the set of learned local features $\{f_i^1, \dots, f_i^k, \dots, f_i^K\}$,

the attention-weighted scores for each feature are computed by a shared MLP, followed by *Softmax*, as follows:

$$W = \text{MLP}(f_i^k) s_i^k = \text{Softmax}(W), \quad (3)$$

where W is the weights of $\text{MLP}(\cdot)$ and s_i^k is an attention weighted score with $s_i^k \in \mathbb{R}^F$. Then, we sum the dot production of the local features f_i^k and the learned attention scores s_i^k to automatically capture the meaningful features, namely, the encoded features, as follows:

$$g_i = \sum_{k=1}^K (f_i^k \cdot s_i^k), \quad (4)$$

where $(\cdot)$ is the dot product operation and g_i is the encoded features with $g_i \in \mathbb{R}^F$.

3.3. Feature Decoding Network

After the original point set is down-pooled to extract the learned local features in the feature encoding network, we propagate the learned local features for point-wise labeling in the feature decoding network. We fully adopt the feature propagation (FP) module from PointNet++ [11], as shown in Figure 3. The FP module defined the distance-based interpolation, to compute the average of the inverse distance weights. Then, the interpolated features are concatenated with skip linked features, followed by shared MLPs. We denote the decoded features as follows:

$$\hat{f}_i = \text{FP}(g_i^l, g_i^{l-1}), \quad (5)$$

where $\text{FP}(\cdot)$ is the feature propagation module, g_i^l is the encoded features at level l th with $g_i^l \in \mathbb{R}^F$, g_i^{l-1} is the skip linked features at level $(l-1)$ th with $g_i^{l-1} \in \mathbb{R}^F$, and $\hat{f}_i$ is the decoded features with $\hat{f}_i \in \mathbb{R}^F$.

Figure 3. Feature decoding network adopted from PointNet++ [11]. The network consists of the feature propagation (FP) module and the fully connected (FC) layer.

3.4. Multiple Loss Scores

The loss function helps to shape the feature space during the training. We introduce multiple loss scores to enhance the learning structure of the features. We use the cross-entropy function to calculate the error between the predicted probabilities and the ground truth labels. We compute the cross-entropy loss at every level of the FP module, as shown in Figure 1.

$L_j | j = \{1, 2, 3, 4\}$ are the cross-entropy loss scores at level 1, 2, 3 and 4 of the FP modules, respectively. We firstly condense the decoded features in each level of the FP

module to the number of categories by the fully connected layer (FC). Then, we calculate the error between the predicted probabilities $\hat{y}_i$ and the ground truth labels y_i , as follows:

$$\hat{y}_i = FC(\hat{f}_i) L_j = \epsilon(\hat{y}_i, y_i), \quad (6)$$

where $FC(\cdot)$ is the fully connected layer, $\hat{f}_i$ is the decoded features with $\hat{f}_i \in \mathbb{R}^F$, $\epsilon(\cdot)$ is a cross-entropy function, and L_j is the loss score. Then, we sum all losses, as follows:

$$Loss = \sum_{j=1}^4 L_j. \quad (7)$$

4. Experiments

4.1. Experimental Setup

We conducted an experiment on a moderate computer with an Intel Core™ i7-7700 CPU @ 3.60 GHZ, 16.0 GB RAM from Intel Corporation, Seoul, South Korea, and NVIDIA GeForce GTX 1070 from NVIDIA Corporation, Seoul, South Korea. The code was written in Python language and Pytorch framework with cuda library, for accelerating the training. The training was carried out by the optimizer Adam, with learning rate of -1×10^{-3} and weight decay of 2×10^{-5} , a batch size of 8, and a number of epochs of 100. The fully connected layer was followed by ReLU activation, batch normalization, and dropout with ratio of 0.5.

4.2. Datasets

We evaluate our proposed method on the semantic segmentation of point clouds with the following two large-scale outdoor benchmark datasets: Toronto-3D [37] and DALES [38].

4.2.1. Toronto-3D Dataset

The Toronto-3D dataset [37] is an outdoor scene point clouds semantic segmentation benchmark. It contains around 78M points, from 1 km of urban road scene in Canada. The dataset is labeled into eight categories, including road, road marking, natural, building, utility line, pole, car, and fence. Each point provides properties such as *xyz* coordinates, *rgb* color, intensity, GPS time, scan angle rank, and category label. The data format contains four blocks, such as L001, L002, L003, and L004, around 250 m each, with a various number of points. We use L001, L003, and L004 for the training set, and L002 for the testing set, following the guideline from the original paper of the Toronto-3D dataset, as shown in Table 1.

Table 1. Toronto-3D dataset [37] (in thousands).

Set	Road	Road Marking	Natural	Building	Utility Line	Pole	Car	Fence	Unclassified	Total
Training	35,503	1500	4626	18,234	579	742	3733	387	2733	68,037
Testing	6353	301	1942	866	84	155	199	24	360	10,284
Total	41,856	1801	6568	19,100	663	897	3932	411	3093	78,321

4.2.2. DALES Dataset

The DALES dataset [38] is an outdoor scene point clouds semantic segmentation benchmark. It contains around 505M points, from 330 km² of urban road scene in Canada. The dataset is labeled into eight categories, including ground, vegetation, cars, trucks, power lines, poles, fences, and buildings. Each point provides properties such as *xyz* coordinates, reflectance, and class label. The data format contains 40 tiles, around 0.5 km² each, with a various number of points. We use 29 tiles for the training set and 11 tiles for

the testing set, following the guideline from the original paper of the DALES dataset, as shown in Table 2.

Table 2. DALES dataset [38] (in millions).

Set	Ground	Vegetation	Cars	Trucks	Power Lines	Poles	Fences	Buildings	Unclassified	Total
Training	178	121	3	0.75	0.80	0.28	2	57	7	369.83
Testing	69	41	1	0.15	0.23	0.09	0.62	23	0.68	135.77
Total	247	162	4	0.90	1.03	0.37	2.62	80	7.68	505.6

4.3. Data Pre-Processing

We divide the original point data into small patches by their order, storing them in the original data file. Table 3 presents each patch consisting of $8192 \times F$. In the Toronto-3D dataset [37], we use two types of data properties, such as 8192×3 for the xyz coordinates of the points (xyz) and 8192×6 for a combination of the xyz coordinates and rgb colors of the points ($xyz + rgb$). In the DALES dataset [38], we only use one type of data property, such as 8192×3 for the xyz coordinates of the points (xyz). We do not perform the random jitter, rotation, or other data augmentations. The raw point data are used only; however, all the points are normalized into zero mean within a unit patch. Then, we randomly select 1024 points out of 8192 points to suit with the input shape of our network model. The random selection method is the simple Python numPy random choice function.

Table 3. Input shape of our method.

Dataset	Properties	Input Points	Selected Points
Toronto-3D [37]	xyz	8192×3	1024×3
	$xyz + rgb$	8192×6	1024×6
DALES [38]	xyz	8192×3	1024×3

5. Results

5.1. Evaluation Metrics

We follow the evaluation metrics of a general semantic segmentation study. We use the overall accuracy (OA) and mean intersection over union ($mIoU$) as the main evaluation metrics, to evaluate the overall quality of the segmentation. Firstly, we compute the per class IoU , as follows:

$$IoU_c = \frac{TP_c}{TP_c + FP_c + FN_c}, \quad (8)$$

where TP , FP , FN represent true positive, false positive, and false negative, respectively, and c is the c th category label. The $mIoU$ is simply the mean across all eight categories, excluding the unclassified category, as follows:

$$mIoU = \frac{1}{C} \sum_{c=1}^C IoU_c, \quad (9)$$

where C is the number of the category label ($C = 8$) and c is the c th category in C . The overall accuracy (OA) is computed by the sum of all the correct predicted points over the total number of points, as follows:

$$OA = \frac{1}{N} \sum_{c=1}^C TP_c, \quad (10)$$

where N is the total number of points.

5.2. Results on Toronto-3D Dataset

Table 4 shows our results on the Toronto-3D dataset. We experimented with our method, with the properties that are mentioned in the above Section 4.3. For experiment on the xyz coordinates of the points, our proposed method achieved an overall accuracy (OA) and a mean intersection over union ($mIoU$) of 72.55% and 66.87%, respectively. We noticed that our method performed well in the road, natural, building, and utility line categories, with $IoUs$ of 92.74%, 88.66%, 93.52%, and 81.03%, respectively. The pole and fence category were achieved around the average scores, with $IoUs$ of 67.71% and 56.90%, respectively.

Table 4. Ablation study on Toronto-3D dataset [37].

Method	OA	$mIoU$	Road	Road Marking	Natural	Building	Utility Line	Pole	Car	Fence
Ours (xyz)	72.55	66.87	92.74	14.75	88.66	93.52	81.03	67.71	39.65	56.90
Ours ($xyz + rgb$)	83.60 ¹	71.03	92.84	27.43	89.90	95.27	85.59	74.50	44.41	58.30

¹ The bold number represents the highest score.

Additionally, we conducted an ablation study on our proposed method, between the xyz coordinates of the points (xyz), and the combination of xyz coordinates and rgb colors of the points ($xyz + rgb$), as shown in Table 4. We noticed that the results that were predicted on the $xyz + rgb$ of the points outperformed the prediction on only the xyz of the points, in both OA and $mIoU$, with 83.60% and 71.03%, respectively. Additionally, the predicted results of the $xyz + rgb$ of the points also outperformed the prediction of the xyz of the points in all the categories. Our weakness is on the road marking and car categories. The $IoUs$ of the prediction on the xyz of the points were 14.75% and 39.65%; and the prediction of the $xyz + rgb$ of the points were 27.43% and 44.41%, which were under an average score. Because they have a fewer numbers of points than the others, we can assume that this was the case, which it is the imbalance data challenge.

The qualitative assessment is illustrated in Figure 4. The L002 scene, which was used as the testing dataset, is rendered in rgb colors, and its ground truth label is rendered in categorical color code, as shown in Figure 4a,b, respectively. Figure 4c demonstrates the qualitative result of our proposed method that was predicted on the xyz coordinates of the points. We can observe that our method performed well in the road, building, and natural categories. However, the road marking category was confused by the road category, and the pole category was also confused by one block of the road category. Additionally, only some points of the car, fence, and utility line categories were able to be predicted. Figure 4d illustrates our results that were predicted on the combination of xyz coordinates and rgb colors of the points ($xyz + rgb$). We noticed that the road, natural, and building categories were still well predicted. Additionally, the pole, utility line, car, and fence categories were improved. They were predicted better than using only the xyz of the points. However, the nature category was confused by some part of the building category.

Figure 4. Our experimental results on L002 data as the testing set of Toronto-3D dataset [37]: (a) original point clouds are rendered in *rgb* colors; (b) ground truth labels are rendered in categorical color code; (c) predicted labels were predicted on the *xyz* coordinates of points; (d) predicted labels were predicted on the combination of *xyz* coordinates and *rgb* colors of points.

5.3. Results on DALES Dataset

Table 5 shows our results on the DALES dataset. We experimented with our method, with the properties mentioned in the above Section 4.3. For the experiment on the xyz coordinates of the points, our proposed method achieved an overall accuracy (OA) and a mean intersection over union ($mIoU$) of 76.43% and 59.52%, respectively. We noticed that our method performed well in the ground, vegetation, and fences categories, with $IoUs$ of 86.78%, 85.40%, and 84.89%, respectively. The power lines, pole and car categories achieved around average scores, with $IoUs$ of 67.47%, 50.76%, and 50.63%, respectively. However, our weakness is on the trucks and buildings categories, with $IoUs$ of 32.59% and 17.66%, respectively, which were under the average score.

Table 5. Our results on DALES dataset [38].

Method	OA	mIoU	Ground	Vegetation	Cars	Trucks	Power Lines	Poles	Fences	Buildings
Ours (xyz)	76.43	59.52	86.78	85.40	50.63	32.59	67.47	50.76	84.89	17.66

The qualitative assessment is illustrated in Figure 5. The 5100_54440 scene, which was used as the testing dataset, can be rendered for its ground truth label in the categorical color code only, since the DALES dataset does not provide the rgb colors of the points, as shown in Figure 5a,b, which demonstrates the qualitative result of our proposed method that was predicted on the xyz coordinates of points. The DALES dataset defined 29 tiles or scenes as the training set, and 11 scenes as the testing set. We followed this guideline, and conducted the testing on all 11 scenes of the testing set. Table 5 shows our testing results of all 11 scenes. The detailed explanation of how we prepared the DALES dataset is described in Section 4.2.2. However, we randomly selected one scene among 11 scenes, for displaying this qualitative assessment. The 5100_54440 scene was randomly selected, and conducted the semantic segmentation. With this scene point cloud, our method could predict well in the ground and vegetation categories only. We could not detect the other categories. This challenge occurs because the DALES dataset has a huge volume (around 505 M points), while the Toronto-3D dataset is around 78 M points. Our method used the input shape of 1024 points, which was small and could not capture the large neighboring points very well. We can assume that this was the case. On the other hand, our method performed well on the Toronto-3D dataset. Therefore, we can assume that our method, with the small input shape of 1024 points, can segment the point clouds up to around 78 M points.

Figure 5. Our experimental results on the 5100_54440 data of the testing set of DALES dataset [38]: (a) ground truth labels are rendered in categorical color code; (b) predicted labels were predicted on the xyz coordinates of points.

6. Discussion

6.1. Discussion on Toronto-3D Dataset

We conducted a comprehensive comparison study on the point-wise MLPs method (i.e., PointNet++ [11] and RandLA-Net [20]), point convolution method (i.e., KPConv [25]), and graph-based method (i.e., DGCNN [28]). The results of the Toronto-3D dataset are demonstrated in Table 6. The results of the PointNet++ [11], RandLA-Net [20], KPConv [25], and DGCNN [28], were recorded from the original paper of the Toronto-3D dataset. It was reported that they were trained on the *xyz* coordinates of the points. We applied only our method on the Toronto-3D dataset. For fair comparison, we compare our method performance using the *xyz* coordinates of the points with them. Compared with those methods, our proposed method achieved the lowest overall accuracy (OA), with 72.55%, but we achieved the second highest mean IoU (*mIoU*), with 66.87%, which was lower than the RandLA-Net, with 77.71%. Our method led to the highest *IoU* in the building and fence categories, and the second highest *IoU* in the road and road marking categories. Even though the natural, utility line, and pole categories did not achieve the highest or the second highest *IoU* among them, they were predicted well, with *IoUs* of 88.66%, 81.03%, and 67.71%, respectively.

Table 6. Comparison results on Toronto-3D dataset [37].

Method	OA	mIoU	Road	Road Marking	Natural	Building	Utility Line	Pole	Car	Fence
PointNet++ [11]	91.21	56.55	91.44	7.59	89.80	74.00	68.60	59.53	53.97	7.54
RandLA-Net [20]	92.95 ¹	77.71	94.61	42.62	96.89	93.01	86.51	78.07	92.85	37.12
KPConv [25]	91.71 ²	60.30	90.20	0.00	86.79	86.83	81.08	73.06	42.85	21.57
DGCNN [28]	89.00	49.60	90.63	0.44	81.25	63.95	47.05	56.86	49.26	7.32
Ours (<i>xyz</i>)	72.55	66.87	92.74	14.75	88.66	93.52	81.03	67.71	39.65	56.90

¹ The bold red number represents the highest score. ² The bold blue number represents the second highest score.

6.2. Discussion on DALES Dataset

We conducted a comprehensive comparison study on the point-wise MLPs method (i.e., PointNet++ [11]), point convolution method (i.e., KPConv [25]), and graph-based method (i.e., SPG [29]). The results on the DALES dataset are demonstrated in Table 7. The results of the PointNet++ [11], KPConv [25], and SPG [29], were recorded from the original paper of the DALES dataset. It was reported that they were trained on the *xyz* coordinates of the points. We applied only our method on the DALES dataset. Compared to those methods, our proposed method achieved the lowest score in both overall accuracy (OA), with 76.43%, and mean IoU (*mIoU*), with 59.52%. However, our method led to the highest *IoU* in the fence category, and the second highest *IoU* in the trucks and poles categories. Even though the ground, vegetation, and power lines categories did not achieve the highest or the second highest *IoU* among them, they performed well, with *IoUs* of 86.78%, 85.40%, and 67.47%, respectively. The cars and poles categories achieved *IoU* scores of 50.63% and 50.76%, respectively, which were at the average score. Our weakness is on the trucks and buildings categories, with 32.59% and 17.66%, respectively, which were under the average score. We noticed that our method achieved better performance on the Toronto-3D dataset than on the DALES dataset.

Table 7. Comparison results on DALES dataset [38].

Method	OA	mIoU	Ground	Vegetation	Cars	Trucks	Power Lines	Poles	Fences	Buildings
PointNet++ [11]	95.70 ²	68.30	94.10	91.20	75.40	30.30	79.90	40.00	46.20	89.10
KPConv [25]	97.80 ¹	81.10	97.10	94.10	85.30	41.90	95.50	75.00	63.50	96.60
SPG [29]	95.50	60.60	94.70	87.90	62.90	18.70	65.20	28.50	33.60	93.40
Ours (<i>xyz</i>)	76.43	59.52	86.78	85.40	50.63	32.59	67.47	50.76	84.89	17.66

¹ The bold red number represents the highest score. ² The bold blue number represents the second highest score.

6.3. Discussion on Computational Cost

Table 8 shows a comparison of computational complexity and inference time per forward pass. The dash represents information that we could not study. For the neighboring strategy, PointNet++ [11] used farthest point sampling (FPS), with a complexity of $O(N^2)$, while our method and RandLA-Net [20] used the random point sampling (RPS), with a complexity of $O(1)$. KPConv [25] used the Kd-tree for projecting, with a complexity of $O(KN \log N)$. We could not figure out the complexity of DGCNN [28]. We tested only PointNet++ [11] on the Toronto-3D and DALES dataset, of which the input shape is set to 1024 points and the batch size is 8, while we could not test for the others. For the number of parameters of the model, our model has around 1.98 M parameters, which is the second fewest numbers, among others. Additionally, our experimental comparison with PointNet++ (its inference time is 370.37 ms) shows that our method (its inference time is 102.45 ms) is faster than about three times.

Table 8. Computational cost comparison.

Method	Neighboring	Complexity	No. of Parameters	Inference Time
PointNet++ [11]	FPS	$O(N^2)$	8.70 M	370.37 ms
RandLA-Net [20]	RPS	$O(1)$	1.24 M	- ¹
KPConv [25]	Kd-tree	$O(KN \log N)$	14.90 M	-
DGCNN [28]	-	-	21 M	-
Ours	RPS	$O(1)$	1.98 M	102.45 ms ²

¹ The dash represents undefined study. ² The bold number represents the lowest score.

6.4. Effect of Our Proposed Mechanism

We conducted a study on the effect of random point sampling (RPS), attention-based pooling (AP), and multiple losses (ML), on the Toronto-3D dataset, as shown in Table 9. We studied four scenarios, as follows: (1) We applied farthest point sampling (FPS), AP, and ML. The FPS + AP + ML gave an overall accuracy (OA) and a mean intersection over union (mIoU) of 70.88% and 65.67%, respectively. (2) We applied the RPS, max pooling (MP), and ML. The RPS + MP + ML gave an OA and mIoU of 64.79% and 65.37%, respectively. (3) We applied the RPS, AP, and a single loss score (SL). The RPS + AP + SL gave an OA and mIoU of 61.42% and 60.12%, respectively. (4) We applied the RPS, AP, and ML. The RPS + AP + ML gave an OA and mIoU of 72.55% and 66.87%, respectively. Through this study, we can see that the fourth scenario (RPS + AP + ML) gave the highest OA and mIoU.

Table 9. Comparison effects of our proposed mechanism on Toronto-3D dataset.

Mechanism	OA	mIoU
FPS + AP + ML	70.88	65.67
RPS + MP + ML	64.79	65.37
RPS + AP + SL	61.42	60.12
RPS + AP + ML	72.55 ¹	66.87

¹ The bold number represents the highest score.

We also conducted a study on the effect of random point sampling (RPS), attention-based pooling (AP), and multiple losses (ML), on the DALES dataset, as shown in Table 10. We also studied four scenarios, as follows: (1) We applied farthest point sampling (FPS), AP, and ML. The FPS + AP + ML gave an overall accuracy (OA) and a mean intersection over union (mIoU) of 81.19% and 51.94%, respectively. (2) We applied the RPS, max pooling (MP), and ML. The RPS + MP + ML gave an OA and mIoU of 62.80% and 39.52%, respectively. (3) We applied the RPS, AP, and a single loss score (SL). The RPS + AP + SL gave an OA and mIoU of 57.77% and 48.65%, respectively. (4) We applied the RPS, AP, and ML. The RPS + AP + ML gave an OA and mIoU of 76.43% and 59.52%, respectively.

Through this study, we can see that the fourth scenario (PRS + AP + ML) gave an OA of 76.43%, which is lower than the first scenario (FPS + AP + ML). However, PRS + AP + ML gave an *mIoU* of 59.52%, which is the highest score among other scenarios. For the semantic segmentation task, the *mIoU* metrics are considered to be more precise than OA. Therefore, we can assume that PRS + AP + ML is the best choice among those scenarios.

Table 10. Comparison effects of our proposed mechanism on DALES dataset.

Mechanism	OA	<i>mIoU</i>
FPS + AP + ML	81.19 ¹	51.94
RPS + MP + ML	62.80	39.52
RPS + AP + SL	57.77	48.65
PRS + AP + ML	76.43	59.52

¹ The bold number represents the highest score.

7. Conclusions

In this paper, we proposed random point sampling, attention-based pooling, and multiple losses summation, integrated with the encoder–decoder shared MLPs method, for the large-scale outdoor point clouds semantic segmentation. We experimented and demonstrated significant computational gains and results on the following two large-scale outdoor benchmark datasets: Toronto-3D and DALES dataset. We achieved an overall accuracy (OA) and a mean intersection over union (*mIoU*), of both the Toronto-3D dataset, with 83.60% and 71.03%, and the DALES dataset, with 76.43% and 59.52%, respectively. Our experimental results shows that our method performed well on the segmentation of the road, nature, building, utility line, and pole categories of the Toronto-3D dataset; and the ground and vegetation categories of the DALES dataset.

We proved that our proposed method can (1) speed up the point selection neighboring process, through the random point sampling layer, by the computational complexity of $O(1)$; (2) capture the important information for aggregation, through the attention-based pooling layer, by the attention-weighted summation; and (3) refine the structure of the learned features space, through the summation of multiple losses. Additionally, comparing the computational complexity, our method performed a few numbers of parameters of the model, and faster than PointNet++, about three times during inferencing. However, there are limitations of our method, due to imbalance, unorganized, and huge volume data challenges, which is our future research.

Author Contributions: Conceptualization and supervision, M.H.; methodology and writing—original draft, B.R. and A.L.; writing—review and editing, B.R., A.L. and M.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by BK21 FOUR (Fostering Outstanding Universities for Research) (No. 5199990914048) and was supported by the Soonchunhyang University Research Fund.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

A-CNN	Annular convolution neural network
AFA	Adaptive feature adjustment
AP	Attention-based pooling
CNN	Convolutional neural network
ConvPoint	Convolutional point
DGCNN	Dynamic graph convolutional neural network
DL	Deep learning
DPAM	Dynamic point agglomeration
DPC	Dilated point convolution
FC	Fully connected layer
FN	False negative
FP	Feature propagation
FP	False positive
FPS	Farthest point sampling
GACNet	Graph attention convolution network
GRU	Gated recurrent unit
GSA	Group shuffle attention
HDGCN	Hierarchical depth-wise graph convolution network
InterpCNN	Interpolated convolutional neural network
KPConv	Kernel point convolution
LSANet	Local spatial awareness network
mIoU	mean intersection over union
MLPs	Multi-layer perceptrons
OA	Overall accuracy
PAG	Point atrous graph
PCCN	Point continuous convolution network
PointCNN	Point convolutional neural network
RandLA-net	Random and Large-scale network
RPS	Random point sampling
SA	Set abstraction
SPG	Super point graph
TP	True positive

References

1. Bello, S.A.; Yu, S.; Wang, C.; Adam, J.M.; Li, J. Review: Deep Learning on 3D point clouds. *Remote Sens.* **2020**, *12*, 1729. [[CrossRef](#)]
2. Guo, Y.; Wang, H.; Hu, Q.; Liu, H.; Liu, L.; Bennamoun, M. Deep Learning for 3D point clouds: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**. [[CrossRef](#)] [[PubMed](#)]
3. Gwak, J.; Jung, J.; Oh, R.; Park, M.; Rakhimov, M.A.K.; Ahn, J. A review of intelligent self-driving vehicle software research. *KSII Trans. Internet Inf. Syst.* **2019**, *13*, 5299–5320. [[CrossRef](#)]
4. Mu, R.; Zeng, X. A Review of Deep Learning research. *KSII Trans. Internet Inf. Syst.* **2019**, *13*, 1738–1764. [[CrossRef](#)]
5. Jung, J.; Park, M.; Cho, K.; Mun, C.; Ahn, J. Intelligent hybrid fusion algorithm with vision patterns for generation of precise digital road maps in self-driving vehicles. *KSII Trans. Internet Inf. Syst.* **2020**, *14*, 3955–3971. [[CrossRef](#)]
6. Yin, J.; Qu, J.; Huang, W.; Chen, Q. Road damage detection and classification based on multi-level feature pyramids. *KSII Trans. Internet Inf. Syst.* **2021**, *15*, 786–799. [[CrossRef](#)]
7. Zhao, X.; Liu, W.; Xing, W.; Wei, X. DA-Res2Net: A novel Densely connected residual attention network for image semantic segmentation. *KSII Trans. Internet Inf. Syst.* **2020**, *14*, 4426–4442. [[CrossRef](#)]
8. Lawin, F.J.; Danelljan, M.; Tosteberg, P.; Bhat, G.; Khan, F.S.; Felsberg, M. Deep projective 3D semantic segmentation. In *Computer Analysis of Images and Patterns*; Felsberg, M., Heyden, A., Krüger, N., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 95–107.
9. Meng, H.; Gao, L.; Lai, Y.; Manocha, D. VV-net: Voxel VAE net with group convolutions for point cloud segmentation. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea, 27 October–2 November 2019; pp. 8500–8508.
10. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep Learning on point sets for 3D classification and segmentation. *arXiv* **2017**, arXiv:1612.00593v2.

11. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4 December 2017; pp. 5105–5114.
12. Jiang, M.; Wu, Y.; Zhao, T.; Zhao, Z.; Lu, C. Pointsift: A sift-like network module for 3D point cloud semantic segmentation. *arXiv* **2018**, arXiv:1807.00652.
13. Engelmann, F.; Kontogianni, T.; Schult, J.; Leibe, B. Know What Your Neighbors Do: 3D Semantic Segmentation of Point Clouds. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8 September 2018.
14. Zeng, W.; Gevers, T. 3DContextNet: Kd tree guided hierarchical learning of point clouds using local and global contextual cues. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8 September 2018.
15. Xie, S.; Liu, S.; Chen, Z.; Tu, Z. Attentional ShapeContextNet for point cloud recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 4606–4615.
16. Zhao, H.; Jiang, L.; Fu, C.W.; Jia, J. Pointweb: Enhancing local neighborhood features for point cloud processing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 5565–5573.
17. Yang, J.; Zhang, Q.; Ni, B.; Li, L.; Liu, J.; Zhou, M.; Tian, Q. Modeling Point Clouds with Self-Attention and Gumbel Subset Sampling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 3323–3332.
18. Chen, L.Z.; Li, X.Y.; Fan, D.P.; Wang, K.; Lu, S.P.; Cheng, M.M. LSA-Net: Feature learning on point sets by local spatial aware layer. *arXiv* **2019**, arXiv:1905.05442.
19. Zhang, Z.; Hua, B.S.; Yeung, S.K. ShellNet: Efficient point cloud convolutional neural networks using concentric shells statistics. In Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 1607–1616.
20. Hu, Q.; Yang, B.; Xie, L.; Rosa, S.; Guo, Y.; Wang, Z.; Trigi, N.; Markham, A. RandLA-Net: Efficient semantic segmentation of large-scale point clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual Conference, Seattle, WA, USA, 14–19 June 2020; pp. 11108–11117. Available online: https://www.youtube.com/channel/UC0n76gicarsN_Y9YShWwhw (accessed on 18 June 2020).
21. Li, Y.; Bu, R.; Sun, M.; Wu, W.; Di, X.; Chen, B. PointCNN: Convolution on x-transformed points. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 820–830.
22. Wang, S.; Suo, S.; Ma, W.C.; Pokrovsky, A.; Urtasun, R. Deep parametric continuous convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 2589–2597.
23. Komarichev, A.; Zhong, Z.; Hua, J. A-CNN: Annularly Convolutional Neural Networks on Point Clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 7421–7430.
24. Boulch, A. ConvPoint: Continuous convolutions for point cloud processing. *Comput. Graph.* **2020**, *88*, 24–34. [[CrossRef](#)]
25. Thomas, H.; Qi, C.R.; Deschaud, J.E.; Marcotegui, B.; Goulette, F.; Guibas, L.J. KPConv: Flexible and Deformable Convolution for Point Clouds. In Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 6411–6420.
26. Engelmann, F.; Kontogianni, T.; Leibe, B. Dilated point convolutions: On the receptive field size of point convolutions on 3D point clouds. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Virtual Conference, Paris, France, 31 May–31 August 2020; pp. 9463–9469.
27. Mao, J.; Wang, X.; Li, H. Interpolated Convolutional Networks for 3D Point Cloud Understanding. In Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 1578–1587.
28. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J. Dynamic graph CNN for learning on point clouds. *ACM Trans. Graph.* **2019**, *38*, 1–12. [[CrossRef](#)]
29. Landrieu, L.; Simonovsky, M. Large-scale point cloud semantic segmentation with superpoint graphs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 4558–4567.
30. Landrieu, L.; Boussaha, M. Point Cloud Oversegmentation with Graph-Structured Deep Metric Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 7440–7449.
31. Wang, L.; Huang, Y.; Hou, Y.; Zhang, S.; Shan, J. Graph attention convolution for point cloud semantic segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 10296–10305.
32. Pan, L.; Chew, C.M.; Lee, G.H. PointAtrousGraph: Deep hierarchical encoder-decoder with point atrous convolution for unorganized 3D points. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Virtual Conference, Paris, France, 31 May–31 August 2020; pp. 1113–1120. Available online: <https://www.ieee-ras.org/students/events/event/1144-icra-2020-ieee-international-conference-on-robotics-and-automation-icra/> (accessed on 30 June 2020).

33. Liang, Z.; Yang, M.; Deng, L.; Wang, C.; Wang, B. Hierarchical depthwise graph convolutional neural network for 3D semantic segmentation of point clouds. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 8152–8158.
34. Jiang, L.; Zhao, H.; Liu, S.; Shen, X.; Fu, C.W.; Jia, J. Hierarchical Point-edge interaction network for point cloud semantic segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 10433–10441.
35. Lei, H.; Akhtar, N.; Mian, A. Spherical convolutional neural network for 3D point clouds. *arXiv* **2018**, arXiv:1805.07872.
36. Liu, J.; Ni, B.; Li, C.; Yang, J.; Tian, Q. Dynamic points agglomeration for hierarchical point sets learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 7546–7555.
37. Tan, W.; Qin, N.; Ma, L.; Li, Y.; Du, J.; Cai, G.; Yang, K.; Li, J. Toronto-3D: A large-scale mobile lidar dataset for semantic segmentation of urban roadways. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR), Virtual Conference, Seattle, WA, USA, 14–19 June 2020; pp. 202–203. Available online: https://www.youtube.com/channel/UC0n76gicaarsN_Y9YShWwhw (accessed on 18 June 2020).
38. Varney, N.; Asari, V.K.; Graehling, Q. DALES: A large-scale aerial LiDAR data set for semantic segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR), Virtual Conference, Seattle, WA, USA, 14–19 June 2020; pp. 186–187. Available online: https://www.youtube.com/channel/UC0n76gicaarsN_Y9YShWwhw (accessed on 18 June 2020).
39. Yang, B.; Wang, S.; Markham, A.; Trigoni, N. Robust attentional aggregation of deep feature sets for multi-view 3D reconstruction. *Int. J. Comput. Vis.* **2020**, *128*, 53–73. [[CrossRef](#)]