RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds

Qingyong Hu[1], Bo Yang[1*], Linhai Xie[1], Stefano Rosa[1], Yulan Guo[2,3],
Zhihua Wang[1], Niki Trigoni[1], Andrew Markham[1]
[1]University of Oxford, [2]Sun Yat-sen University, [3]National University of Defense Technology
`firstname.lastname@cs.ox.ac.uk`

Abstract

*We study the problem of efficient semantic segmentation for large-scale 3D point clouds. By relying on expensive sampling techniques or computationally heavy pre/post-processing steps, most existing approaches are only able to be trained and operate over small-scale point clouds. In this paper, we introduce **RandLA-Net**, an efficient and lightweight neural architecture to directly infer per-point semantics for large-scale point clouds. The key to our approach is to use random point sampling instead of more complex point selection approaches. Although remarkably computation and memory efficient, random sampling can discard key features by chance. To overcome this, we introduce a novel local feature aggregation module to progressively increase the receptive field for each 3D point, thereby effectively preserving geometric details. Extensive experiments show that our RandLA-Net can process 1 million points in a single pass with up to 200× faster than existing approaches. Moreover, our RandLA-Net clearly surpasses state-of-the-art approaches for semantic segmentation on two large-scale benchmarks Semantic3D and SemanticKITTI.*

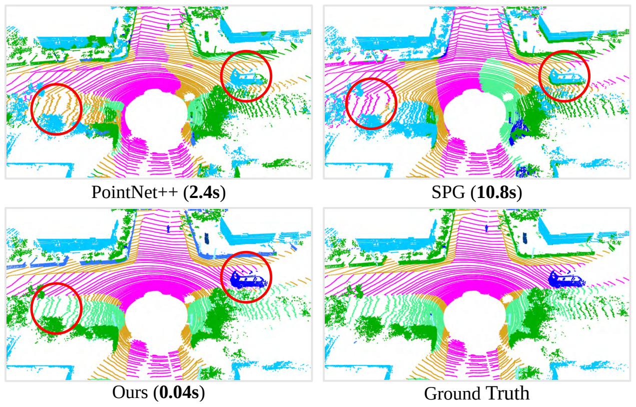

Figure 1. Semantic segmentation results of PointNet++ [38], SPG [23] and our approach on SemanticKITTI [3]. Our RandLA-Net takes only 0.04s to directly process a large point cloud with 10^5 points over $150\times130\times10$ meters in 3D space, which is up to 200× faster than SPG. Red circles highlight the superior segmentation accuracy of our approach.

1. Introduction

Efficient semantic segmentation of large-scale 3D point clouds is a fundamental and essential capability for real-time intelligent systems, such as autonomous driving and augmented reality. A key challenge is that the raw point clouds acquired by depth sensors are typically irregularly sampled, unstructured and unordered. Although deep convolutional networks show excellent performance in structured 2D computer vision tasks, they cannot be directly applied to this type of unstructured data.

Recently, the pioneering work PointNet [37] has emerged as a promising approach for directly processing 3D point clouds. It learns per-point features using shared multi-layer perceptrons (MLPs). This is computationally efficient but fails to capture wider context information for each point. To learn richer local structures, many dedicated neural modules have been subsequently and rapidly introduced. These modules can be generally categorized as: 1) neighbouring feature pooling [38, 28, 19, 64, 63], 2) graph message passing [51, 42, 49, 50, 5, 20, 30], 3) kernel-based convolution [43, 18, 54, 26, 21, 22, 48, 33], and 4) attention-based aggregation [55, 62, 60, 36]. Although these approaches achieve impressive results for object recognition and semantic segmentation, almost all of them are limited to extremely small 3D point clouds (e.g., 4k points or 1×1 meter blocks) and cannot be directly extended to larger point clouds (e.g., millions of points and up to 200×200 meters). The reasons for this limitation are three-fold. 1) The commonly used point-sampling methods of these networks are either computationally expensive or memory inefficient. For example, the widely employed farthest-point sampling [38] takes over 200 seconds to sample 10% of 1 million points. 2) Most existing local feature learners usually rely on computationally

[*]Corresponding author

expensive kernelisation or graph construction, thereby being unable to process massive number of points. 3) For a large-scale point cloud, which usually consists of hundreds of objects, the existing local feature learners are either incapable of capturing complex structures, or do so inefficiently, due to their limited size of receptive fields.

A handful of recent works have started to tackle the task of directly processing large-scale point clouds. SPG [23] preprocesses the large point clouds as super graphs before applying neural networks to learn per super-point semantics. Both FCPN [39] and PCT [7] combine voxelization and point-level networks to process massive point clouds. Although they achieve decent segmentation accuracy, the preprocessing and voxelization steps are too computationally heavy to be deployed in real-time applications.

In this paper, we aim to design a memory and computationally efficient neural architecture, which is able to directly process large-scale 3D point clouds in a single pass, without requiring any pre/post-processing steps such as voxelization, block partitioning or graph construction. However, this task is extremely challenging as it requires: 1) a memory and computationally efficient sampling approach to progressively downsample large-scale point clouds to fit in the limits of current GPUs, and 2) an effective local feature learner to progressively increase the receptive field size to preserve complex geometric structures. To this end, we first systematically demonstrate that **random sampling** is a key enabler for deep neural networks to efficiently process large-scale point clouds. However, random sampling can discard key semantic information, especially for objects with low point densities. To counter the potentially detrimental impact of random sampling, we propose a new and efficient **local feature aggregation module** to capture complex local structures over progressively smaller point-sets.

Amongst existing sampling methods, farthest point sampling and inverse density sampling are the most frequently used for small-scale point clouds [38, 54, 29, 64, 15]. As point sampling is such a fundamental step within these networks, we investigate the relative merits of different approaches in Section 3.2, both by examining their computational complexity and empirically by measuring their memory consumption and processing time. From this, we see that the commonly used sampling methods limit scaling towards large point clouds, and act as a significant bottleneck to real-time processing. However, we identify random sampling as by far the most suitable component for large-scale point cloud processing as it is fast and scales efficiently. Random sampling is not without cost, because prominent point features may be dropped by chance and it cannot be used directly in existing networks without incurring a performance penalty. To overcome this issue, we design a new local feature aggregation module in Section 3.3, which is capable of effectively learning complex local structures by progressively increasing the receptive field size in each neural layer. In particular, for each 3D point, we firstly introduce a local spatial encoding (LocSE) unit to explicitly preserve local geometric structures. Secondly, we leverage attentive pooling to automatically keep the useful local features. Thirdly, we stack multiple LocSE units and attentive poolings as a dilated residual block, greatly increasing the effective receptive field for each point. Note that all these neural components are implemented as shared MLPs, and are therefore remarkably memory and computational efficient.

Overall, being built on the principles of simple **rand**om sampling and an effective **l**ocal feature **a**ggregator, our efficient neural architecture, named **RandLA-Net**[1], not only is up to 200$\times$ faster than existing approaches on large-scale point clouds, but also surpasses the state-of-the-art semantic segmentation methods on both Semantic3D [16] and SemanticKITTI [3] benchmarks. Figure 1 shows qualitative results of our approach. Our key contributions are:

- We analyse and compare existing sampling approaches, identifying random sampling as the most suitable component for efficient learning on large-scale point clouds.
- We propose an effective local feature aggregation module to automatically preserve complex local structures by progressively increasing the receptive field for each point.
- We demonstrate significant memory and computational gains over baselines, and surpass the state-of-the-art semantic segmentation methods on multiple large-scale benchmarks.

2. Related Work

To extract features from 3D point clouds, traditional approaches usually manually hand-craft features [11, 41]. Recent learning based approaches mainly include projection-based, voxel-based and point-based schemes which are outlined here.

(1) Projection and Voxel Based Networks. To leverage the success of 2D CNNs, many works [27, 8, 57, 24] project/flatten 3D point clouds onto 2D images to address the task of object detection. However, many geometric details are lost during the projection. Alternatively, point clouds can be voxelized into 3D grids and then powerful 3D CNNs are applied as in [14, 25, 10, 34, 9]. Although they achieve leading results on semantic segmentation and object detection, their primary limitation is the heavy computation cost, especially when processing large-scale point clouds.

[1]Code and data are available at: https://github.com/QingyongHu/RandLA-Net

(2) Point Based Networks. Inspired by PointNet/PointNet++ [37, 38], many recent works introduced sophisticated neural modules to learn per-point local features. These modules can be generally classified as 1) neighbouring feature pooling [28, 19, 64, 63], 2) graph message passing [51, 42, 49, 50, 5, 20, 30], 3) kernel-based convolution [43, 18, 54, 26, 21, 22, 48, 33], and 4) attention-based aggregation [55, 62, 60, 36]. Although these networks have shown promising results on small point clouds, most of them cannot directly scale up to large scenarios due to their high computational and memory costs. Compared with them, our proposed RandLA-Net is distinguished in three ways: 1) it only relies on random sampling within the network, thereby requiring much less memory and computation; 2) the proposed local feature aggregator can obtain successively larger receptive fields by explicitly considering the local spatial relationship and point features, thus being more effective and robust for learning complex local patterns; 3) the entire network only consists of shared MLPs without relying on any expensive operations such as graph construction and kernelisation, therefore being superbly efficient for large-scale point clouds.

(3) Learning for Large-scale Point Clouds. SPG [23] preprocesses the large point clouds as super graphs to learn per super-point semantics. The recent FCPN [39] and PCT [7] apply both voxel-base and point-based networks to process the massive point clouds. However, both the graph partitioning and voxelisation are computationally expensive. In constrast, our efficient RandLA-Net is end-to-end trainable without requiring any additional pre/post-processing steps.

3. RandLA-Net

3.1. Overview

As illustrated in Figure 2, given a large-scale point cloud with millions of points spanning up to hundreds of meters, to process it with a deep neural network inevitably requires those points to be progressively and efficiently downsampled in each neural layer, without losing the useful point features. In our RandLA-Net, we propose to use the simple and fast approach of random sampling to greatly decrease point density, whilst applying a carefully designed local feature aggregator to retain prominent features. This allows the entire network to achieve an excellent trade-off between efficiency and effectiveness.

3.2. The quest for efficient sampling

Existing point sampling approaches [38, 29, 15, 12, 1, 54] can be roughly classified into heuristic and learning-based approaches. However, there is still no standard sampling strategy that is suitable for large-scale point clouds. Therefore, we analyse and compare their relative merits and complexity as follows.

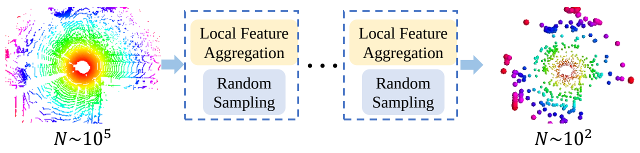

Figure 2. In each layer of RandLA-Net, the large-scale point cloud is significantly downsampled, yet is capable of retaining features necessary for accurate segmentation.

(1) Heuristic Sampling

- *Farthest Point Sampling (FPS):* In order to sample K points from a large-scale point cloud $\boldsymbol{P}$ with N points, FPS returns a reordering of the metric space $\{p_1 \cdots p_k \cdots p_K\}$, such that each p_k is the farthest point from the first $k-1$ points. FPS is widely used in [38, 29, 54] for semantic segmentation of small point sets. Although it has a good coverage of the entire point set, its computational complexity is $\mathcal{O}(N^2)$. For a large-scale point cloud ($N \sim 10^6$), FPS takes up to 200 seconds to process on a single GPU. This shows that FPS is not suitable for large-scale point clouds.

- *Inverse Density Importance Sampling (IDIS):* To sample K points from N points, IDIS reorders all N points according to the density of each point, after which the top K points are selected [15]. Its computational complexity is approximately $\mathcal{O}(N)$. Empirically, it takes 10 seconds to process 10^6 points. Compared with FPS, IDIS is more efficient, but also more sensitive to outliers. However, it is still too slow for use in a real-time system.

- *Random Sampling (RS):* Random sampling uniformly selects K points from the original N points. Its computational complexity is $\mathcal{O}(1)$, which is agnostic to the total number of input points, i.e., it is constant-time and hence inherently scalable. Compared with FPS and IDIS, random sampling has the highest computational efficiency, regardless of the scale of input point clouds. It only takes 0.004s to process 10^6 points.

(2) Learning-based Sampling

- *Generator-based Sampling (GS):* GS [12] learns to generate a small set of points to approximately represent the original large point set. However, *FPS* is usually used in order to match the generated subset with the original set at inference stage, incurring additional computation. In our experiments, it takes up to 1200 seconds to sample 10% of 10^6 points.

- *Continuous Relaxation based Sampling (CRS):* CRS approaches [1, 60] use the reparameterization trick to relax the sampling operation to a continuous domain for end-to-end training. In particular, each sampled point is learnt based on a weighted sum over the full point clouds. It results in a large weight matrix when sampling all the new

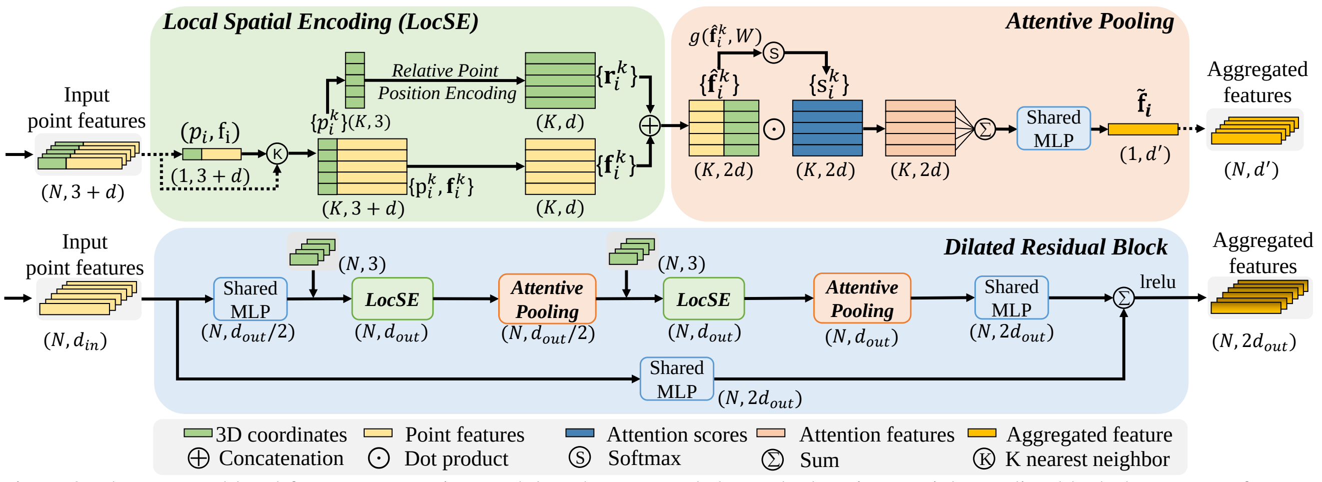

Figure 3. The proposed local feature aggregation module. The top panel shows the location spatial encoding block that extracts features, and the attentive pooling mechanism that weights the most important, based on the local context and geometry. The bottom panel shows how two of these components are chained together, to increase the receptive field size, within a residual block.

points simultaneously with a one-pass matrix multiplication, leading to an unaffordable memory cost. For example, it is estimated to take more than a 300GB memory footprint to sample 10% of 10^6 points.

- *Policy Gradient based Sampling (PGS):* PGS formulates the sampling operation as a Markov decision process [56]. It sequentially learns a probability to sample each point. However, the learnt probability has high variance due to the extremely large exploration space when the point cloud is in large scale. For example, to sample 10% of 10^6 points, the exploration space is $C_{10^6}^{10^5}$ and it is unlikely to learn an effective sampling policy. We empirically find that the network is difficult to converge if PGS is used for large point clouds.

Overall, FPS, IDIS and GS are too computationally expensive to be applied for large-scale point clouds. CRS approaches have an excessive memory footprint and PGS is hard to learn. By contrast, random sampling has the following two advantages: 1) it is remarkably computational efficient as it is agnostic to the total number of input points, 2) it does not require extra memory for computation. Therefore, we safely conclude that random sampling is by far the most suitable approach to process large-scale point clouds compared with all existing alternatives. However, random sampling may result in many useful point features being dropped. To overcome it, we propose a powerful local feature aggregation module as presented in below Section 3.3.

3.3. Local Feature Aggregation

As shown in Figure 3, our local feature aggregation module is applied to each 3D point in parallel and it consists of three neural units: 1) local spatial encoding (LocSE), 2) attentive pooling, and 3) dilated residual block.

(1) Local Spatial Encoding

Given a point cloud $\boldsymbol{P}$ together with per-point features (e.g., raw RGB, or intermediate learnt features), this local spatial encoding unit explicitly embeds the x-y-z coordinates of all neighbouring points, such that the corresponding point features are always aware of their relative spatial locations. This allows the LocSE unit to explicitly observe the local geometric patterns, thus eventually benefiting the entire network to effectively learn complex local structures. In particular, this unit includes the following steps:

Finding Neighbouring Points. For the i^{th} point, its neighbouring points are firstly gathered by the simple K-nearest neighbours (KNN) algorithm for efficiency. Note that, the KNN is based on the point-wise Euclidean distances.

Relative Point Position Encoding. For each of the nearest K points $\{p_i^1 \cdots p_i^k \cdots p_i^K\}$ of the center point p_i, we explicitly encode the relative point position as follows:

$$\mathbf{r}_i^k = MLP\Big(p_i \oplus p_i^k \oplus (p_i - p_i^k) \oplus ||p_i - p_i^k||\Big) \tag{1}$$

where p_i and p_i^k are the x-y-z positions of points, $\oplus$ is the concatenation operation, and $||\cdot||$ calculates the Euclidean distance between the neighbouring and center points. It seems that $\mathbf{r}_i^k$ is encoded from redundant point position information. Interestingly, this tends to aid the network to learn local features and obtains good performance in practice.

Point Feature Augmentation. For each neighbouring point p_i^k, the encoded relative point positions $\mathbf{r}_i^k$ are concatenated with its corresponding point features $\mathbf{f}_i^k$, obtaining an augmented feature vector $\hat{\mathbf{f}}_i^k$.

Eventually, the output of LocSE unit is a new set of neighbouring features $\hat{\mathbf{F}}_i = \{\hat{\mathbf{f}}_i^1 \cdots \hat{\mathbf{f}}_i^k \cdots \hat{\mathbf{f}}_i^K\}$, which explicitly encodes the local geometric structures for the center point p_i. We notice that the recent work [32] also uses point positions to improve semantic segmentation. However, the positions are used to learn point scores in [32], while our LocSE explicitly encodes the relative positions to augment the neighbouring point features.

(2) Attentive Pooling

This neural unit is used to aggregate the set of neighbouring point features $\hat{\mathbf{F}}_i$. Existing works [38, 29] typically use max/mean pooling to hard integrate the neighbouring features, resulting in the majority of the information being lost. By contrast, we turn to the powerful attention mechanism to automatically learn important local features. In particular, inspired by [59], our attentive pooling unit consists of the following steps.

Computing Attention Scores. Given the set of local features $\hat{\mathbf{F}}_i = \{\hat{\mathbf{f}}_i^1 \cdots \hat{\mathbf{f}}_i^k \cdots \hat{\mathbf{f}}_i^K\}$, we design a shared function $g()$ to learn a unique attention score for each feature. Basically, the function $g()$ consists of a shared MLP followed by $softmax$. It is formally defined as follows:

$$\mathbf{s}_i^k = g(\hat{\mathbf{f}}_i^k, \boldsymbol{W}) \tag{2}$$

where $\boldsymbol{W}$ is the learnable weights of a shared MLP.

Weighted Summation. The learnt attention scores can be regarded as a soft mask which automatically selects the important features. Formally, these features are weighted summed as follows:

$$\tilde{\mathbf{f}}_i = \sum_{k=1}^{K} (\hat{\mathbf{f}}_i^k \cdot \mathbf{s}_i^k) \tag{3}$$

To summarize, given the input point cloud $\boldsymbol{P}$, for the i^{th} point p_i, our LocSE and Attentive Pooling units learn to aggregate the geometric patterns and features of its K nearest points, and finally generate an informative feature vector $\tilde{\mathbf{f}}_i$.

(3) Dilated Residual Block

Since the large point clouds are going to be substantially downsampled, it is desirable to significantly increase the receptive field for each point, such that the geometric details of input point clouds are more likely to be reserved, even if some points are dropped. As shown in Figure 3, inspired by the successful ResNet [17] and the effective dilated networks [13], we stack multiple LocSE and Attentive Pooling units together with a skip connection as a dilated residual block.

To further illustrate the capability of our dilated residual block, Figure 4 shows that the red 3D point observes K neighbouring points after the first LocSE/Attentive Pooling operation, and then is able to receive information from up to K^2 neighbouring points i.e. its two-hop neighbourhood after the second. This is a cheap way of dilating the receptive field and expanding the effective neighbourhood through feature propagation. Theoretically, the more units we stack, the more powerful this block as its sphere of reach becomes greater and greater. However, more units would inevitably sacrifice the overall computation efficiency. In addition, the entire network is likely to be over-fitted. In our RandLA-Net, we simply stack two sets of LocSE and Attentive Pooling as the standard residual block, achieving a satisfactory balance between efficiency and effectiveness.

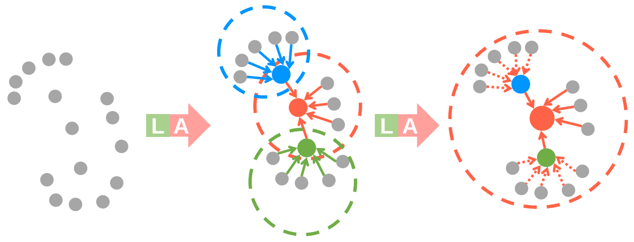

Figure 4. Illustration of the dilated residual block which significantly increases the receptive field (dotted circle) of each point, colored points represent the aggregated features. L: Local spatial encoding, A: Attentive pooling.

Overall, our local feature aggregation module is designed to effectively preserve complex local structures via explicitly considering neighbouring geometries and significantly increasing receptive fields. Moreover, this module only consists of feed-forward MLPs, thus being computationally efficient.

3.4. Implementation

We implement RandLA-Net by stacking multiple local feature aggregation modules and random sampling layers. The detailed architecture is presented in the Appendix. We use the Adam optimizer with default parameters. The initial learning rate is set as 0.01 and decreases by 5% after each epoch. The number of nearest points K is set as 16. To train our RandLA-Net in parallel, we sample a fixed number of points ($\sim 10^5$) from each point cloud as the input. During testing, the whole raw point cloud is fed into our network to infer per-point semantics without any pre/post-processing. All experiments are conducted on an NVIDIA RTX2080Ti GPU.

4. Experiments

4.1. Efficiency of Random Sampling

In this section, we empirically evaluate the efficiency of existing sampling approaches including FPS, IDIS, RS, GS, CRS, and PGS, which have been discussed in Section 3.2. In particular, we conduct the following 4 groups of experiments.

- Group 1. Given a small scale point cloud ($\sim 10^3$ points), we use each sampling approach to progressively downsample it. Specifically, the point cloud is

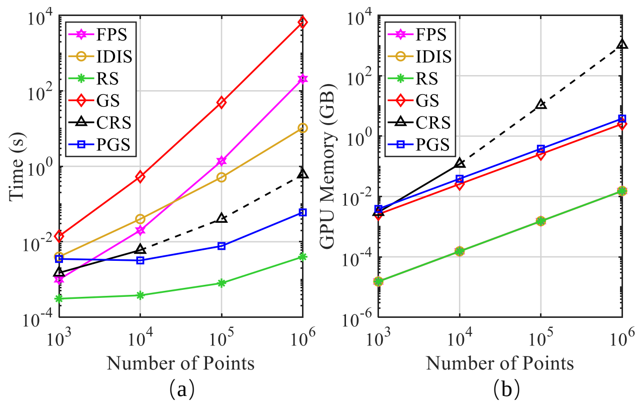

Figure 5. Time and memory consumption of different sampling approaches. The dashed lines represent estimated values due to the limited GPU memory.

downsampled by five steps with only 25% points being retained in each step on a single GPU i.e. a fourfold decimation ratio. This means that there are only $\sim (1/4)^5 \times 10^3$ points left in the end. This downsampling strategy emulates the procedure used in PointNet++ [38]. For each sampling approach, we sum up its time and memory consumption for comparison.

- Group 2/3/4. The total number of points are increased towards large-scale, i.e., around $10^4, 10^5$ and 10^6 points respectively. We use the same five sampling steps as in Group 1.

Analysis. Figure 5 compares the total time and memory consumption of each sampling approach to process different scales of point clouds. It can be seen that: 1) For small scale point clouds ($\sim 10^3$), all sampling approaches tend to have similar time and memory consumption, and are unlikely to incur a heavy or limiting computation burden. 2) For large-scale point clouds ($\sim 10^6$), FPS/IDIS/GS/CRS/PGS are either extremely time-consuming or memory-costly. By contrast, random sampling has superior time and memory efficiency overall. This result clearly demonstrates that most existing networks [38, 29, 54, 32, 64, 60] are only able to be optimized on small blocks of point clouds primarily because they rely on the expensive sampling approaches. Motivated by this, we use the efficient random sampling strategy in our RandLA-Net.

4.2. Efficiency of RandLA-Net

In this section, we systematically evaluate the overall efficiency of our RandLA-Net on real-world large-scale point clouds for semantic segmentation. Particularly, we evaluate RandLA-Net on SemanticKITTI [3] dataset, obtaining the total time consumption of our network on Sequence 08 which has 4071 frames of point clouds in total. We also evaluate the time consumption of recent representative works [37, 38, 29, 23, 48] on the same dataset. For a fair comparison, we feed the same number of points (i.e., 81920) from each scan into each neural network.

	Total time (seconds)	Parameters (millions)	Maximum inference points (millions)
PointNet (Vanilla) [37]	192	0.8	0.49
PointNet++ (SSG) [38]	9831	0.97	0.98
PointCNN [29]	8142	11	0.05
SPG [23]	43584	**0.25**	-
KPConv [48]	717	14.9	0.54
RandLA-Net (Ours)	**176**	0.95	**1.15**

Table 1. The computation time, network parameters and maximum number of input points of different approaches for semantic segmentation on Sequence 08 of SemanticKITTI [3] dataset.

In addition, we also evaluate the memory consumption of RandLA-Net and the baselines. In particular, we not only report the total parameters of each network, but also measure the maximum number of 3D points each network can take as input in a single pass to infer per-point semantics. Note that, all experiments are conducted on the same machine with an AMD 3700X @3.6GHz CPU and an NVIDIA RTX2080Ti GPU.

Analysis. Table 1 quantitatively shows the total time and memory consumption of different approaches. It can be seen that, 1) SPG [23] has the lowest number of network parameters, but takes the longest time to process the point clouds due to the expensive geometrical partitioning and super-graph construction steps; 2) PointNet++ [38] and PointCNN [29] are also computationally expensive mainly because of the FPS sampling operation; 3) PointNet [37] and KPConv [48] are unable to take extremely large-scale point clouds (e.g. 10^6 points) in a single pass due to their memory inefficient operations. 4) Thanks to the simple random sampling together with the efficient MLP based local feature aggregator, our RandLA-Net takes the shortest time (translating to 23 frames per second) to infer the semantic labels for each large-scale point cloud (up to 10^6 points).

4.3. Semantic Segmentation on Benchmarks

In this section, we evaluate the semantic segmentation of our RandLA-Net on three large-scale public datasets: the Semantic3D [16], SemanticKITTI [3], and S3DIS [2].

(1) Evaluation on Semantic3D. The Semantic3D dataset [16] consists of 15 point clouds for training and 15 for online testing. Each point cloud has up to 10^8 points, covering up to $160\times240\times30$ meters in real-world 3D space. The raw 3D points belong to 8 classes and contain 3D coordinates, RGB information, and intensity. We only use the 3D coordinates and color information to train and test our RandLA-Net. Mean Intersection of Union (mIoU) and Overall Accuracy (OA) of all classes are used as the standard metrics. For fair comparison, we only include the results of recently published strong baselines [4, 46, 47, 40, 63, 50, 23] and the

	mIoU (%)	OA (%)	man-made.	natural.	high veg.	low veg.	buildings	hard scape	scanning art.	cars
SnapNet_ [4]	59.1	88.6	82.0	77.3	79.7	22.9	91.1	18.4	37.3	64.4
SEGCloud [46]	61.3	88.1	83.9	66.0	86.0	40.5	91.1	30.9	27.5	64.3
RF_MSSF [47]	62.7	90.3	87.6	80.3	81.8	36.4	92.2	24.1	42.6	56.6
MSDeepVoxNet [40]	65.3	88.4	83.0	67.2	83.8	36.7	92.4	31.3	50.0	78.2
ShellNet [63]	69.3	93.2	96.3	90.4	83.9	41.0	94.2	34.7	43.9	70.2
GACNet [50]	70.8	91.9	86.4	77.7	**88.5**	**60.6**	94.2	37.3	43.5	77.8
SPG [23]	73.2	94.0	**97.4**	**92.6**	87.9	44.0	83.2	31.0	63.5	76.2
KPConv [48]	74.6	92.9	90.9	82.2	84.2	47.9	94.9	40.0	**77.3**	**79.7**
RandLA-Net (Ours)	**76.0**	**94.4**	96.5	92.0	85.1	50.3	**95.0**	**41.1**	68.2	79.4

Table 2. Quantitative results of different approaches on Semantic3D (reduced-8) [16]. Only the recent published approaches are compared. Accessed on 15 November 2019.

Methods	Size	**mIoU(%)**	Params(M)	road	sidewalk	parking	other-ground	building	car	truck	bicycle	motorcycle	other-vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic-sign
PointNet [37]		14.6	3	61.6	35.7	15.8	1.4	41.4	46.3	0.1	1.3	0.3	0.8	31.0	4.6	17.6	0.2	0.2	0.0	12.9	2.4	3.7
SPG [23]		17.4	**0.25**	45.0	28.5	0.6	0.6	64.3	49.3	0.1	0.2	0.2	0.8	48.9	27.2	24.6	0.3	2.7	0.1	20.8	15.9	0.8
SPLATNet [43]	50K pts	18.4	0.8	64.6	39.1	0.4	0.0	58.3	58.2	0.0	0.0	0.0	0.0	71.1	9.9	19.3	0.0	0.0	0.0	23.1	5.6	0.0
PointNet++ [38]		20.1	6	72.0	41.8	18.7	5.6	62.3	53.7	0.9	1.9	0.2	0.2	46.5	13.8	30.0	0.9	1.0	0.0	16.9	6.0	8.9
TangentConv [45]		40.9	0.4	83.9	63.9	33.4	15.4	83.4	90.8	15.2	2.7	16.5	12.1	79.5	49.3	58.1	23.0	28.4	**8.1**	49.0	35.8	28.5
SqueezeSeg [52]		29.5	1	85.4	54.3	26.9	4.5	57.4	68.8	3.3	16.0	4.1	3.6	60.0	24.3	53.7	12.9	13.1	0.9	29.0	17.5	24.5
SqueezeSegV2 [53]	64*2048	39.7	1	88.6	67.6	45.8	17.7	73.7	81.8	13.4	18.5	17.9	14.0	71.8	35.8	60.2	20.1	25.1	3.9	41.1	20.2	36.3
DarkNet21Seg [3]	pixels	47.4	25	91.4	74.0	57.0	26.4	81.9	85.4	18.6	**26.2**	26.5	15.6	77.6	48.4	63.6	31.8	33.6	4.0	52.3	36.0	50.0
DarkNet53Seg [3]		49.9	50	**91.8**	**74.6**	**64.8**	**27.9**	**84.1**	86.4	25.5	24.5	**32.7**	22.6	78.3	50.1	**64.0**	36.2	33.6	4.7	**55.0**	38.9	**52.2**
RandLA-Net (Ours)	50K pts	**50.3**	0.95	90.4	67.9	56.9	15.5	81.1	**94.0**	**42.7**	19.8	21.4	**38.7**	**78.3**	**60.3**	59.0	**47.5**	**48.8**	4.6	49.7	**44.2**	38.1

Table 3. Quantitative results of different approaches on SemanticKITTI [3]. Only the recent published methods are compared and all scores are obtained from the online single scan evaluation track. Accessed on 15 November 2019.

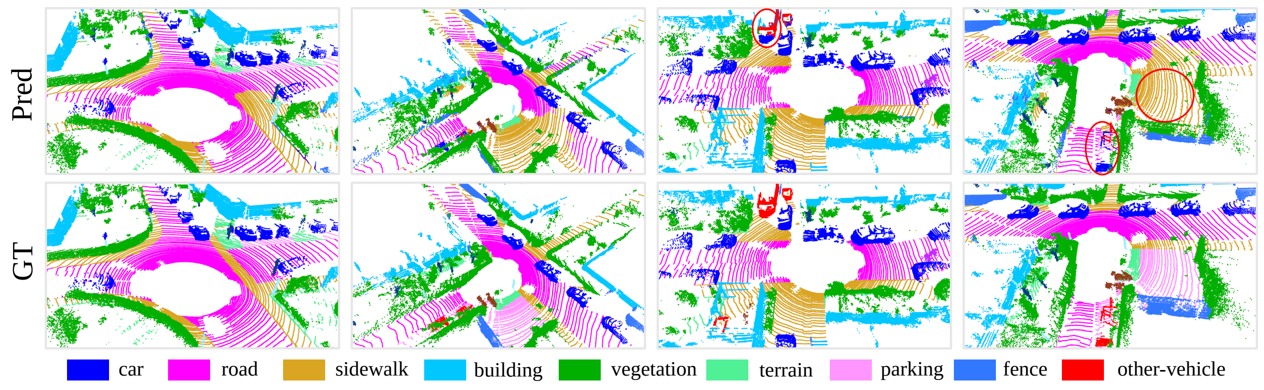

Figure 6. Qualitative results of RandLA-Net on the validation set of SemanticKITTI [3]. Red circles show the failure cases.

current state-of-the-art approach KPConv [48].

Table 2 presents the quantitative results of different approaches. RandLA-Net clearly outperforms all existing methods in terms of both mIoU and OA. Notably, RandLA-Net also achieves superior performance on six of the eight classes, except *low vegetation* and *scanning artifact*.

(2) Evaluation on SemanticKITTI. SemanticKITTI [3] consists of 43552 densely annotated LIDAR scans belonging to 21 sequences. Each scan is a large-scale point cloud with $\sim 10^5$ points and spanning up to $160\times160\times20$ meters in 3D space. Officially, the sequences 00$\sim$07 and 09$\sim$10 (19130 scans) are used for training, the sequence 08 (4071 scans) for validation, and the sequences 11$\sim$21 (20351 scans) for online testing. The raw 3D points only have 3D coordinates without color information. The mIoU score over 19 categories is used as the standard metric.

Table 3 shows a quantitative comparison of our RandLA-Net with two families of recent approaches, i.e. 1) point-based methods [37, 23, 43, 38, 45] and 2) projection based approaches [52, 53, 3], and Figure 6 shows some qualitative results of RandLA-Net on the validation split. It can be seen that our RandLA-Net surpasses all point based approaches [37, 23, 43, 38, 45] by a large margin. We also outperform all projection based methods [52, 53, 3], but not significantly, primarily because DarkNet [3] achieves much better results on the small object category such as *traffic-sign*. However, our RandLA-Net has far fewer network parameters than DarkNet [3] and is more computationally efficient as it does not require the costly steps of pre- and post-projection processing.

(3) Evaluation on S3DIS. The S3DIS dataset [2] consists of 271 rooms belonging to 6 large areas. Each point cloud is a medium-sized single room ($\sim 20\times15\times5$ meters) with dense 3D points. To evaluate the semantic segmentation of our RandLA-Net, we use the standard 6-fold cross-validation in our experiments. The mean IoU (mIoU), mean class Accuracy (mAcc) and Overall Accuracy (OA) of the total 13 classes are compared.

As shown in Table 4, our RandLA-Net achieves on-par or better performance than state-of-the-art methods. Note that, most of these baselines [38, 29, 64, 63, 51, 6] tend to use sophisticated but expensive operations or samplings to optimize the networks on small blocks (e.g., 1×1 meter) of point clouds, and the relatively small rooms act in their favours to be divided into tiny blocks. By contrast, RandLA-Net takes the entire rooms as input and is able to efficiently infer per-point semantics in a single pass.

	OA(%)	mAcc(%)	mIoU(%)
PointNet [37]	78.6	66.2	47.6
PointNet++ [38]	81.0	67.1	54.5
DGCNN [51]	84.1	-	56.1
3P-RNN [61]	86.9	-	56.3
RSNet [19]	-	66.5	56.5
SPG [23]	85.5	73.0	62.1
LSANet [6]	86.8	-	62.2
PointCNN [29]	88.1	75.6	65.4
PointWeb [64]	87.3	76.2	66.7
ShellNet [63]	87.1	-	66.8
HEPIN [20]	**88.2**	-	67.8
KPConv [48]	-	79.1	**70.6**
RandLA-Net (Ours)	87.2	**81.5**	68.5

Table 4. Quantitative results of different approaches on S3DIS dataset [2] (6-fold cross validation). Only the recent published methods are included.

4.4. Ablation Study

Since the impact of random sampling is fully studied in Section 4.1, we conduct the following ablation studies for our local feature aggregation module. All ablated networks are trained on sequences 00$\sim$07 and 09$\sim$10, and tested on the sequence 08 of SemanticKITTI dataset [3].

(1) Removing local spatial encoding (LocSE). This unit enables each 3D point to explicitly observe its local geometry. After removing locSE, we directly feed the local point features into the subsequent attentive pooling.

(2$\sim$4) Replacing attentive pooling by max/mean/sum pooling. The attentive pooling unit learns to automatically combine all local point features. By comparison, the widely used max/mean/sum poolings tend to hard select or combine features, therefore their performance may be sub-optimal.

(5) Simplifying the dilated residual block. The dilated residual block stacks multiple LocSE units and attentive poolings, substantially dilating the receptive field for each 3D point. By simplifying this block, we use only one LocSE unit and attentive pooling per layer, i.e. we don't chain multiple blocks as in our original RandLA-Net.

Table 5 compares the mIoU scores of all ablated networks. From this, we can see that: 1) The greatest impact is caused by the removal of the chained spatial embedding and attentive pooling blocks. This is highlighted in Figure 4, which shows how using two chained blocks allows information to be propagated from a wider neighbourhood, i.e. approximately K^2 points as opposed to just K. This is especially critical with random sampling, which is not guaranteed to preserve a particular set of points. 2) The removal of the local spatial encoding unit shows the next greatest impact on performance, demonstrating that this module is necessary to effectively learn local and relative geometry context. 3) Removing the attention module diminishes performance by not being able to effectively retain useful features. From this ablation study, we can see how the proposed neural units complement each other to attain our state-of-the-art performance.

	mIoU(%)
(1) Remove local spatial encoding	45.1
(2) Replace with max-pooling	47.1
(3) Replace with mean-pooling	45.2
(4) Replace with sum-pooling	45.7
(5) Simplify dilated residual block	41.5
(6) The Full framework (RandLA-Net)	**52.0**

Table 5. The mean IoU scores of all ablated networks based on our full RandLA-Net.

5. Conclusion

In this paper, we demonstrated that it is possible to efficiently and effectively segment large-scale point clouds by using a lightweight network architecture. In contrast to most current approaches, that rely on expensive sampling strategies, we instead use random sampling in our framework to significantly reduce the memory footprint and computational cost. A local feature aggregation module is also introduced to effectively preserve useful features from a wide neighbourhood. Extensive experiments on multiple benchmarks demonstrate the high efficiency and the state-of-the-art performance of our approach. It would be interesting to extend our framework for the end-to-end 3D instance segmentation on large-scale point clouds by drawing on the recent work [58] and also for the real-time dynamic point cloud processing [31].

References

[1] Abubakar Abid, Muhammad Fatih Balin, and James Zou. Concrete autoencoders for differentiable feature selection and reconstruction. *arXiv preprint arXiv:1901.09346*, 2019. 3, 11

[2] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 6, 8, 12, 14

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, 2019. 1, 2, 6, 7, 8, 12, 13, 15

[4] Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. Unstructured point cloud semantic labeling using deep segmentation networks. *3DOR*, 2017. 6, 7

[5] Chao Chen, Guanbin Li, Ruijia Xu, Tianshui Chen, Meng Wang, and Liang Lin. ClusterNet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In *CVPR*, 2019. 1, 3

[6] Lin-Zhuo Chen, Xuan-Yi Li, Deng-Ping Fan, Ming-Ming Cheng, Kai Wang, and Shao-Ping Lu. LSANet: Feature learning on point sets by local spatial attention. *arXiv preprint arXiv:1905.05442*, 2019. 8, 11

[7] Siheng Chen, Sufeng Niu, Tian Lan, and Baoan Liu. PCT: Large-scale 3D point cloud representations via graph inception networks with applications to autonomous driving. In *ICIP*, 2019. 2, 3

[8] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3D object detection network for autonomous driving. In *CVPR*, 2017. 2

[9] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point R-CNN. In *ICCV*, 2019. 2

[10] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. *arXiv preprint arXiv:1904.08755*, 2019. 2

[11] Chin Seng Chua and Ray Jarvis. Point signatures: A new representation for 3D object recognition. *IJCV*. 2

[12] Oren Dovrat, Itai Lang, and Shai Avidan. Learning to sample. In *CVPR*, 2019. 3, 11

[13] Francis Engelmann, Theodora Kontogianni, and Bastian Leibe. Dilated point convolutions: On the receptive field of point convolutions. *arXiv preprint arXiv:1907.12046*, 2019. 5

[14] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 2

[15] Fabian Groh, Patrick Wieschollek, and Hendrik P. A. Lensch. Flex-convolution (million-scale point-cloud learning beyond grid-worlds). In *ACCV*, 2018. 2, 3, 11

[16] Timo Hackel, N. Savinov, L. Ladicky, Jan D. Wegner, K. Schindler, and M. Pollefeys. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2017. 2, 6, 7, 12, 13

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[18] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *CVPR*, 2018. 1, 3

[19] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3D segmentation of point clouds. In *CVPR*, 2018. 1, 3, 8, 14

[20] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *ICCV*, 2019. 1, 3, 8

[21] Artem Komarichev, Zichun Zhong, and Jing Hua. A-CNN: Annularly convolutional neural networks on point clouds. In *CVPR*, 2019. 1, 3

[22] Shiyi Lan, Ruichi Yu, Gang Yu, and Larry S Davis. Modeling local geometric structure of 3D point clouds using Geo-CNN. *arXiv preprint arXiv:1811.07782*, 2018. 1, 3

[23] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, 2018. 1, 2, 3, 6, 7, 8, 14

[24] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 2

[25] Truc Le and Ye Duan. Pointgrid: A deep network for 3D shape understanding. In *CVPR*, 2018. 2

[26] Huan Lei, Naveed Akhtar, and Ajmal Mian. Octree guided cnn with spherical kernels for 3D point clouds. In *CVPR*, 2019. 1, 3

[27] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3D lidar using fully convolutional network. *arXiv preprint arXiv:1608.07916*, 2016. 2

[28] Jiaxin Li, Ben M Chen, and Gim Hee Lee. SO-Net: Self-organizing network for point cloud analysis. In *CVPR*, 2018. 1, 3

[29] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on x-transformed points. In *NeurIPS*, 2018. 2, 3, 5, 6, 8, 11, 14

[30] Jinxian Liu, Bingbing Ni, Caiyuan Li, Jiancheng Yang, and Qi Tian. Dynamic points agglomeration for hierarchical point sets learning. In *ICCV*, 2019. 1, 3

[31] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3D point cloud sequences. In *ICCV*, 2019. 8

[32] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, 2019. 5, 6, 11

[33] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3D point cloud understanding. In *ICCV*, 2019. 1, 3

[34] Hsien-Yu Meng, Lin Gao, Yu-Kun Lai, and Dinesh Manocha. Vv-net: Voxel vae net with group convolutions for point cloud segmentation. In *ICCV*, 2019. 2

[35] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*, 2014. 11

[36] Anshul Paigwar, Ozgur Erkent, Christian Wolf, and Christian Laugier. Attentional pointnet for 3d-object detection in point clouds. In *CVPRW*, 2019. 1, 3

[37] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017. 1, 3, 6, 7, 8, 14

[38] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 1, 2, 3, 5, 6, 7, 8, 11

[39] Dario Rethage, Johanna Wald, Jurgen Sturm, Nassir Navab, and Federico Tombari. Fully-convolutional point networks for large-scale point clouds. In *ECCV*, 2018. 2, 3

[40] Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette. Classification of point cloud scenes with multiscale voxel deep network. *arXiv preprint arXiv:1804.03583*, 2018. 6, 7

[41] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3D registration. In *ICRA*, 2009. 2

[42] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *CVPR*, 2018. 1, 3

[43] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SPLATNet: sparse lattice networks for point cloud processing. In *CVPR*, 2018. 1, 3, 7

[44] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*, 2000. 11

[45] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3D. In *CVPR*, 2018. 7

[46] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3D point clouds. In *3DV*, 2017. 6, 7

[47] Hugues Thomas, François Goulette, Jean-Emmanuel Deschaud, and Beatriz Marcotegui. Semantic classification of 3D point clouds with multiscale spherical neighborhoods. In *3DV*, 2018. 6, 7

[48] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. *arXiv preprint arXiv:1904.08889*, 2019. 1, 3, 6, 7, 8, 14

[49] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *ECCV*, 2018. 1, 3

[50] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *CVPR*, 2019. 1, 3, 6, 7

[51] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019. 1, 3, 8

[52] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3D lidar point cloud. In *ICRA*, 2018. 7

[53] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *ICRA*, 2019. 7

[54] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3D point clouds. *arXiv preprint arXiv:1811.07246*, 2018. 1, 2, 3, 6, 11

[55] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *CVPR*, 2018. 1, 3

[56] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015. 4

[57] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Realtime 3D object detection from point clouds. In *CVPR*, 2018. 2

[58] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning object bounding boxes for 3D instance segmentation on point clouds. *arXiv preprint arXiv:1906.01140*, 2019. 8

[59] Bo Yang, Sen Wang, Andrew Markham, and Niki Trigoni. Robust attentional aggregation of deep feature sets for multiview 3D reconstruction. *IJCV*, 2019. 5

[60] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *CVPR*, 2019. 1, 3, 6, 11

[61] Xiaoqing Ye, Jiamao Li, Hexiao Huang, Liang Du, and Xiaolin Zhang. 3D recurrent neural networks with context fusion for point cloud semantic segmentation. In *ECCV*, 2018. 8, 14

[62] Wenxiao Zhang and Chunxia Xiao. PCAN: 3D attention map learning using contextual information for point cloud based retrieval. In *CVPR*, 2019. 1, 3

[63] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. *arXiv preprint arXiv:1908.06295*, 2019. 1, 3, 6, 7, 8, 14

[64] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019. 1, 2, 3, 6, 8, 11, 14

Appendices

A. Details for the Evaluation of Sampling.

We provide the implementation details of different sampling approaches evaluated in Section 4.1. To sample K points (point features) from a large-scale point cloud $\boldsymbol{P}$ with N points (point features):

1. *Farthest Point Sampling (FPS):* We follow the implementation [2] provided by PointNet++ [38], which is also widely used in [29, 54, 32, 6, 64]. In particular, FPS is implemented as an operator running on GPU.

2. *Inverse Density Importance Sampling (IDIS):* Given a point p_i, its density ρ is approximated by calculating the summation of the distances between p_i and its nearest t points [15]. Formally:

$$\rho(p_i) = \sum_{j=1}^{t} \left|\left| p_i - p_i^j \right|\right|, p_i^j \in \mathcal{N}(p_i) \tag{4}$$

   where p_i^j represents the coordinates (i.e. x-y-z) of the j^{th} point of the neighbour points set $\mathcal{N}(p_i)$, t is set to 16. All the points are ranked according to the inverse density $\frac{1}{\rho}$ of points. Finally, the top K points are selected.

3. *Random Sampling (RS):* We implement random sampling with the python numpy package. Specifically, we first use the numpy function *numpy.random.choice()* to generate K indices. We then gather the corresponding spatial coordinates and per-point features from point clouds by using these indices.

4. *Generator based Sampling (GS):* The implementation follows the code[3] provided by [12]. We first train a ProgressiveNet [12] to transform the raw point clouds into ordered point sets according to their relevance to the task. After that, the first K points are kept, while the rest discarded.

5. *Continuous Relaxation based Sampling (CRS): CRS* is implemented with the self-attended gumbel-softmax sampling [1][60]. Given a point cloud $\boldsymbol{P} \in \mathbb{R}^{N\times(d+3)}$ with 3D coordinates and per point features, we firstly estimate a probability score vector $\mathbf{s} \in \mathbb{R}^N$ through a score function parameterized by a MLP layer, i.e., $\mathbf{s} = softmax(MLP(\boldsymbol{P}))$, which learns a categorical distribution. Then, with the Gumbel noise $\mathbf{g} \in \mathbb{R}^N$ drawn from the distribution $Gumbel(0,1)$. Each sampled point feature vector $\mathbf{y} \in \mathbb{R}^{d+3}$ is calculated as follows:

$$\mathbf{y} = \sum_{i=1}^{N} \frac{\exp\left((log(\mathbf{s}^{(i)}) + \mathbf{g}^{(i)})/\tau\right)\boldsymbol{P}^{(i)}}{\sum_{j=1}^{N}\exp\left((log(\mathbf{s}^{(j)}) + \mathbf{g}^{(j)})/\tau\right)}, \tag{5}$$

   where $\mathbf{s}^{(i)}$ and $\mathbf{g}^{(i)}$ indicate the i^{th} element in the vector $\mathbf{s}$ and $\mathbf{g}$ respectively, $\boldsymbol{P}^{(i)}$ represents the i^{th} row vector in the input matrix $\boldsymbol{P}$. $\tau > 0$ is the annealing temperature. When $\tau \to 0$, Equation 5 approaches the discrete distribution and samples each row vector in $\boldsymbol{P}$ with the probability $p(\mathbf{y} = \boldsymbol{P}^{(i)}) = \mathbf{s}^{(i)}$.

6. *Policy Gradients based Sampling (PGS):* Given a point feature set $\boldsymbol{P} \in \mathbb{R}^{N\times(d+3)}$ with 3D coordinates and per point features, we first predict a score $\mathbf{s}$ for each point, which is learnt by a MLP function, i.e., $\mathbf{s} = softmax(MLP(\boldsymbol{P})) + \epsilon$, where $\epsilon \in \mathbb{R}^N$ is a zero-mean Gaussian noise with the variance $\boldsymbol{\Sigma}$ for random exploration. After that, we sample K vectors in $\boldsymbol{P}$ with the top K scores. To properly update the score function, we apply REINFORCE algorithm [44] as the gradient estimator. By modeling the entire sampling operation as a sequential Markov Decision Process (MDP), we formulate the policy function π as:

$$a_i \sim \pi(a|\boldsymbol{P}^{(i)}; \theta, \boldsymbol{\Sigma}) \tag{6}$$

   where a_i is the binary decision of whether to sample the i^{th} vector in $\boldsymbol{P}$, θ is the network parameter of the MLP. Then we apply the segmentation accuracy R as the reward value for the entire sampling process and maximize our reward function $\mathcal{J} = R$ with the following estimated gradients:

$$\frac{\partial \mathcal{J}}{\partial \theta} \approx \frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{N}\frac{\partial}{\partial\theta}\log\pi(a_i|\boldsymbol{P}^{(i)}; \theta, \boldsymbol{\Sigma}) \times (R - b^c - b(\boldsymbol{P}^{(i)})), \tag{7}$$

   where M is the batch size, b^c and $b(\boldsymbol{P}^{(i)})$ are two control variates [35] for alleviating the high variance problem of policy gradients.

[2] https://github.com/charlesq34/pointnet2

[3] https://github.com/orendv/learning_to_sample

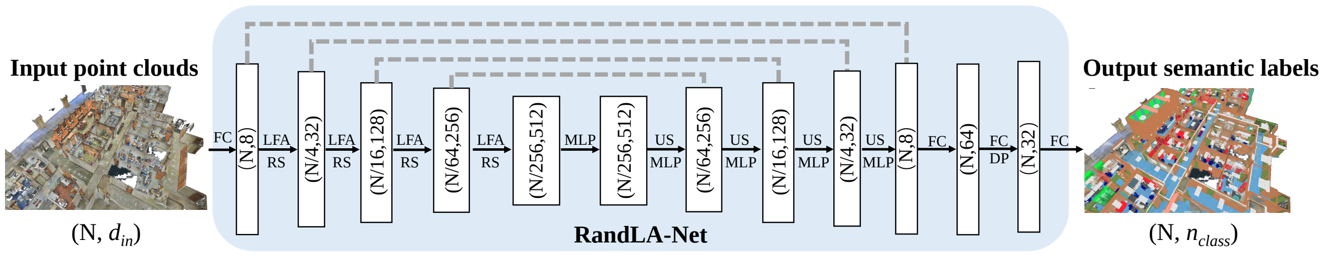

Figure 7. The detailed architecture of our RandLA-Net. (N, D) represents the number of points and feature dimension respectively. FC: Fully Connected layer, LFA: Local Feature Aggregation, RS: Random Sampling, MLP: shared Multi-Layer Perceptron, US: Up-sampling, DP: Dropout.

B. Details of the Network Architecture

Figure 7 shows the detailed architecture of RandLA-Net. The network follows the widely-used encoder-decoder architecture with skip connections. The input point cloud is first fed to a shared MLP layer to extract per-point features. Four encoding and decoding layers are then used to learn features for each point. At last, three fully-connected layers and a dropout layer are used to predict the semantic label of each point. The details of each part are as follows:

Network Input: The input is a large-scale point cloud with a size of $N \times d_{in}$ (the batch dimension is dropped for simplicity), where N is the number of points, d_{in} is the feature dimension of each input point. For both S3DIS [2] and Semantic3D [16] datasets, each point is represented by its 3D coordinates and color information (i.e., x-y-z-R-G-B), while each point of the SemanticKITTI [3] dataset is only represented by 3D coordinates.

Encoding Layers: Four encoding layers are used in our network to progressively reduce the size of the point clouds and increase the per-point feature dimensions. Each encoding layer consists of a local feature aggregation module (Section 3.3) and a random sampling operation (Section 3.2). The point cloud is downsampled with a four-fold decimation ratio. In particular, only 25% of the point features are retained after each layer, i.e., $(N \rightarrow \frac{N}{4} \rightarrow \frac{N}{16} \rightarrow \frac{N}{64} \rightarrow \frac{N}{256})$. Meanwhile, the per-point feature dimension is gradually increased each layer to preserve more information, i.e., $(8 \rightarrow 32 \rightarrow 128 \rightarrow 256 \rightarrow 512)$.

Decoding Layers: Four decoding layers are used after the above encoding layers. For each layer in the decoder, we first use the *KNN* algorithm to find one nearest neighboring point for each query point, the point feature set is then upsampled through a nearest-neighbor interpolation. Next, the upsampled feature maps are concatenated with the intermediate feature maps produced by encoding layers through skip connections, after which a shared MLP is applied to the concatenated feature maps.

Final Semantic Prediction: The final semantic label of each point is obtained through three shared fully-connected layers (N, 64) → (N, 32) → (N, n_{class}) and a dropout layer. The dropout ratio is 0.5.

Network Output: The output of RandLA-Net is the predicted semantics of all points, with a size of $N \times n_{class}$, where n_{class} is the number of classes.

C. Additional Ablation Studies on LocSE

In Section 3.3, we encode the relative point position based on the following equation:

$$\mathbf{r}_i^k = MLP\Big(p_i \oplus p_i^k \oplus (p_i - p_i^k) \oplus ||p_i - p_i^k||\Big) \tag{8}$$

We further investigate the effects of different spatial information in our framework. Particularly, we conduct the following more ablative experiments for LocSE:

- 1) Encoding the coordinates of the point p_i only.
- 2) Encoding the coordinates of neighboring points p_i^k only.
- 3) Encoding the coordinates of the point p_i and its neighboring points p_i^k.
- 4) Encoding the coordinates of the point p_i, the neighboring points p_i^k, and Euclidean distance $||p_i - p_i^k||$.
- 5) Encoding the coordinates of the point p_i, the neighboring points p_i^k, and the relative position $p_i - p_i^k$.

Table 6 compares the mIoU scores of all ablated networks. We can see that: 1) Explicitly encoding all spatial information leads to the best mIoU performance. 2) The relative position $p_i - p_i^k$ plays an important role in this component, primarily because the relative point position enables the network to be aware of the local geometric patterns. 3) Only encoding the point position p_i or p_i^k is

LocSE	mIoU(%)
(1) (p_i)	40.7
(2) (p_i^k)	41.1
(3) (p_i, p_i^k)	42.5
(4) $(p_i, p_i^k, \|\|p_i - p_i^k\|\|)$	44.1
(5) $(p_i, p_i^k, p_i - p_i^k)$	48.8
(6) $(p_i, p_i^k, p_i - p_i^k, \|\|p_i - p_i^k\|\|)$ **(The Full Unit)**	52.0

Table 6. The mIoU result of RandLA-Net by encoding different kinds of spatial information.

unlikely to improve the performance, because the relative local geometric patterns are not explicitly encoded.

D. Additional Ablation Studies on Dilated Residual Block

In our RandLA-Net, we stack two LocSE and Attentive Pooling units as the standard dilated residual block to gradually increase the receptive field. To further evaluate how the number of aggregation units in the dilated residual block impact the entire network, we conduct the following two more groups of experiments.

- 1) We simplify the dilated residual block by using only one LocSE unit and attentive pooling.
- 2) We add one more LocSE unit and attentive pooling, i.e., there are three aggregation units chain together.

Dilated residual block	mIoU(%)
(1) one aggregation unit	41.9
(2) three aggregation units	48.7
(3) two aggregation units (**The Standard Block**)	52.0

Table 7. The mIoU scores of RandLA-Net regarding different number of aggregation units in a residual block.

Table 7 shows the mIoU scores of different ablated networks on the validation split of SemanticKITTI [3] dataset. It can be seen that: 1) Only one aggregation unit in the dilated residual block leads to a significant drop in segmentation performance, due to the limited receptive field. 2) Three aggregation units in each block do not improve the accuracy as expected. This is because the significantly increased receptive fields and the large number of trainable parameters tend to be overfitted.

E. Visualization of Attention Scores

To better understand the attentive pooling, it is desirable to visualize the learned attention scores. However, since the attentive pooling operates on a relatively small local point set (i.e., K=16), it is hardly able to recognize meaningful shapes from such small local regions. Alternatively, we visualize the learned attention weight matrix W defined in Equation 2 in each layer. As shown in Figure 8, the attention weights have large values in the first encoding layers, then gradually become smooth and stable in subsequent layers. This shows that the attentive pooling tends to choose prominent or key point features at the beginning. After the point cloud being significantly downsampled, the attentive pooling layer tends to retain the majority of those point features.

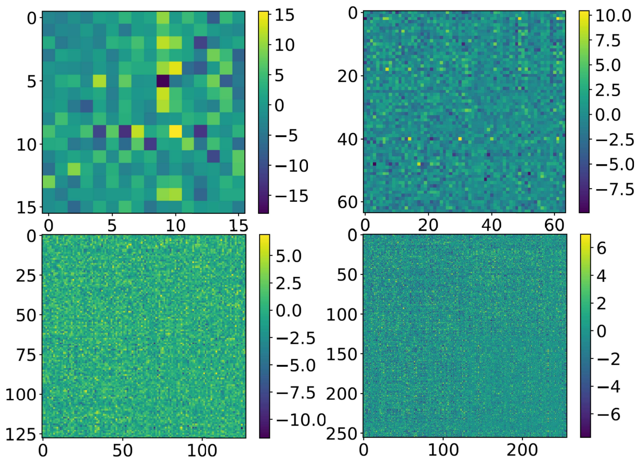

Figure 8. Visualization of the learned attention matrix in different layers. From top left to bottom right: 16×16 attention matrix, 64×64 attention matrix, 128×128 attention matrix, 256×256 attention matrix. The yellow color represents higher attention scores.

F. Additional Results on Semantic3D

More qualitative results of RandLA-Net on Semantic3D [16] dataset (*reduced-8*) are shown in Figure 9.

G. Additional Results on SemanticKITTI

Figure 10 shows more qualitative results of our RandLA-Net on the validation set of SemanticKITTI [3]. The red boxes showcase the failure cases. It can be seen that, the points belonging to *other-vehicle* are likely to be misclassified as *car*, mainly because the partial point clouds without colors are extremely difficult to be distinguished between the two similar classes. In addition, our approach tends to fail in several minority classes such as *bicycle*, *motorcycle*, *bicyclist* and *motorcyclist*, due to the extremely imbalanced point distribution in the dataset. For example, the number of points for *vegetation* is 7000 times more than that of *motorcyclist*.

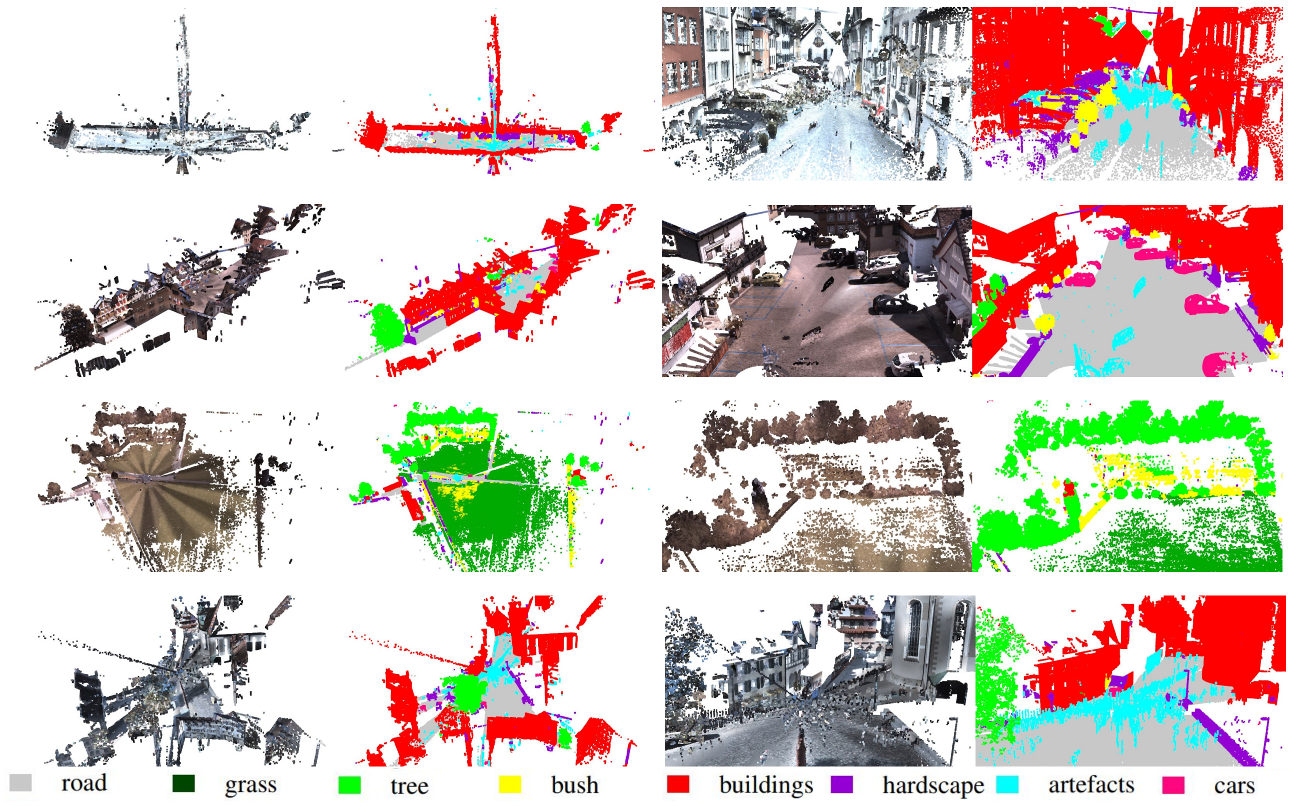

Figure 9. Qualitative results of RandLA-Net on *reduced-8* split of Semantic3D. From left to right: full RGB colored point clouds, predicted semantic labels of full point clouds, detailed view of colored point clouds, detailed view of predicted semantic labels. Note that the ground truth of the test set is not publicly available.

	OA(%)	mAcc(%)	mIoU(%)	ceil.	floor	wall	beam	col.	wind.	door	table	chair	sofa	book.	board	clut.
PointNet [37]	78.6	66.2	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	54.1	42.0	9.6	38.2	29.4	35.2
RSNet [19]	-	66.5	56.5	92.5	92.8	78.6	32.8	34.4	51.6	68.1	59.7	60.1	16.4	50.2	44.9	52.0
3P-RNN [61]	86.9	-	56.3	92.9	93.8	73.1	42.5	25.9	47.6	59.2	60.4	66.7	24.8	57.0	36.7	51.6
SPG [23]	86.4	73.0	62.1	89.9	95.1	76.4	62.8	47.1	55.3	68.4	73.5	69.2	63.2	45.9	8.7	52.9
PointCNN [29]	**88.1**	75.6	65.4	94.8	97.3	75.8	63.3	51.7	58.4	57.2	71.6	69.11	39.1	61.2	52.2	58.6
PointWeb [64]	87.3	76.2	66.7	93.5	94.2	80.8	52.4	41.3	64.9	68.1	71.4	67.1	50.3	62.7	62.2	58.5
ShellNet [63]	87.1	-	66.8	90.2	93.6	79.9	60.4	44.1	64.9	52.9	71.6	84.7	53.8	64.6	48.6	59.4
KPConv [48]	-	79.1	**70.6**	93.6	92.4	83.1	63.9	54.3	66.1	76.6	57.8	64.0	69.3	74.9	61.3	60.3
RandLA-Net(Ours)	87.1	**81.5**	68.5	92.7	95.6	79.2	61.7	47.0	63.1	67.7	68.9	74.2	55.3	63.4	63.0	58.7

Table 8. Quantitative results of different approaches on S3DIS [2] (6-fold cross-validation). Overall Accuracy (OA, %), mean class Accuracy (mAcc, %), mean IoU (mIoU, %), and per-class IoU (%) are reported.

H. Additional Results on S3DIS

We report the detailed 6-fold cross validation results of our RandLA-Net on S3DIS [2] in Table 8. Figure 11 shows more qualitative results of our approach.

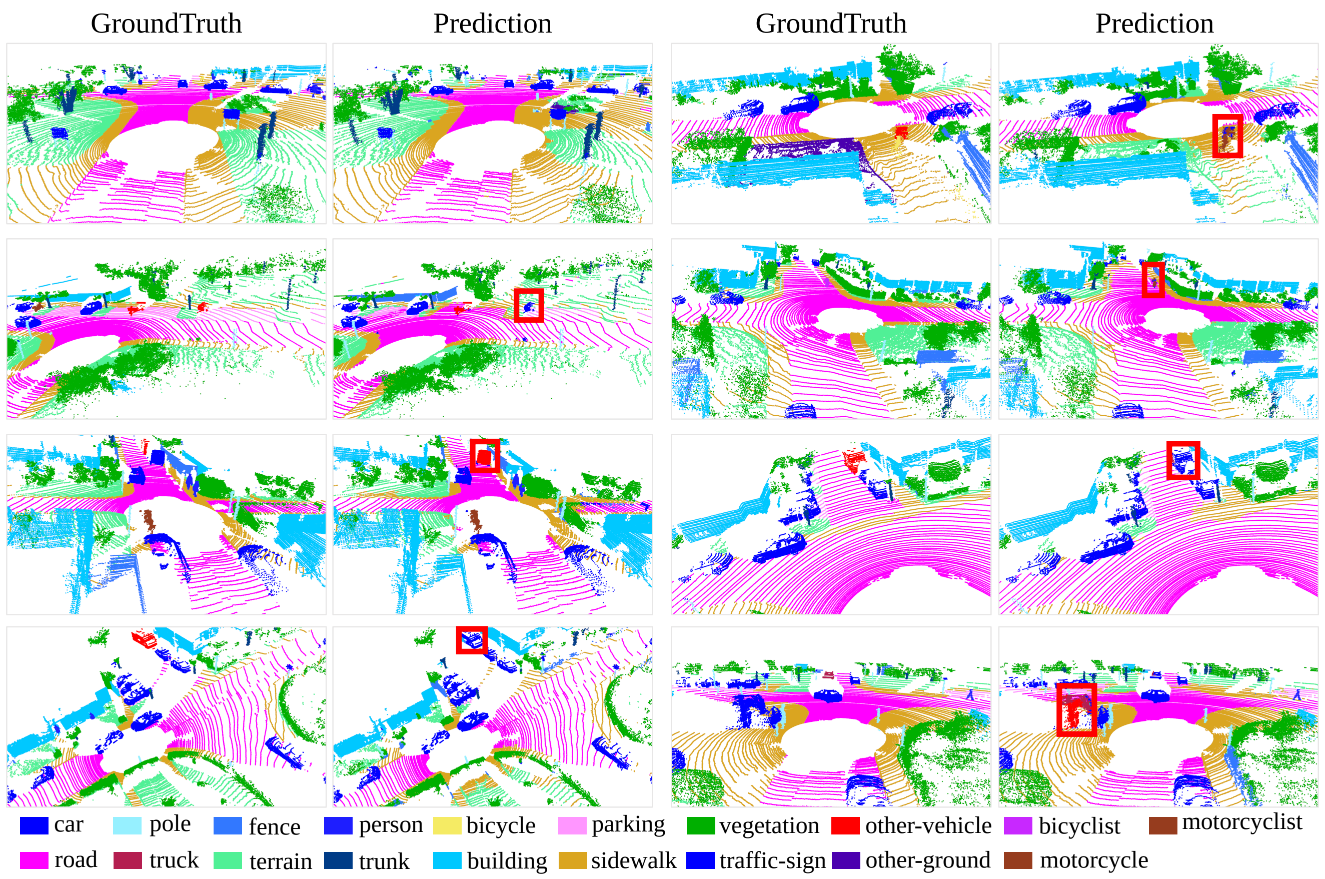

Figure 10. Qualitative results of RandLA-Net on the validation split of SemanticKITTI [3]. Red boxes show the failure cases.

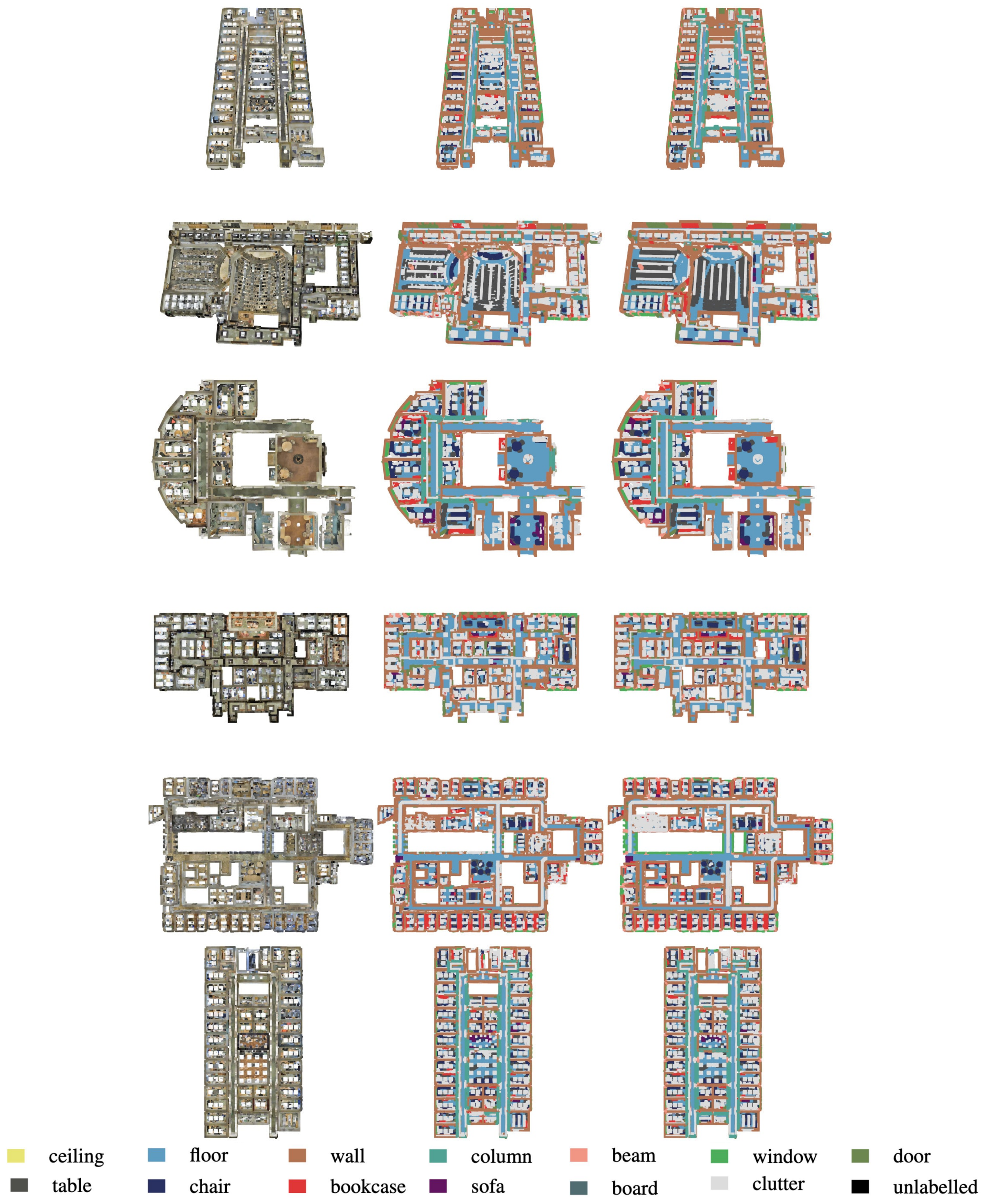

Figure 11. Semantic segmentation results of our RandLA-Net on the complete point clouds of Areas 1-6 in S3DIS. Left: full RGB input cloud; middle: predicted labels; right: ground truth.