Point Cloud Segmentation with Guided Sampling and Continuous Interpolation

Gaoyang Zhang and Xinguo Liu

State Key Lab of CAD&CG, Zhejiang University
blurgy@zju.edu.cn
xinguoliu@zju.edu.cn

Abstract. Sampling and interpolation are pivotal in the design of 3D neural networks. Presently, farthest point sampling and k-NN interpolation are the predominant techniques. Nonetheless, the former can lead to information loss in feature-rich regions, while the latter might introduce noticeable discontinuities, compromising neural network performance. In this research, we address information loss with a novel method, **DistrFPS**, that considers the input information distribution during the farthest point sampling. Leveraging DistrFPS, we introduce a guided sampling module to retain crucial information for subsequent network layers. We also propose a continuous interpolation module grounded in barycentric interpolation to ensure spatial coherent feature propagation to higher resolution network layers. Our approach's efficacy in preserving information is demonstrated empirically through signal reconstruction in both 2D and 3D realms. Comprehensive experiments on S3DIS, ScanNet, and ShapeNetPart affirm the advantages of our technique for point-based networks.

Keywords: Point cloud · Semantic segmentation · 3D neural networks · Farthest point sampling · Barycentric interpolation

1 Introduction

Segmentation of point clouds plays a critical role in diverse applications, ranging from robotics and autonomous driving to augmented reality. Given the memory constraints of contemporary computing platforms, most algorithms for 3D point cloud processing employ a hierarchical pipeline [22, 30, 32, 34, 54, 59]. In this approach, the dense point cloud undergoes iterative subsampling to diminish its spatial resolution, while a neural network is concurrently applied to extract high-level features from the sampled points. These sparse features are then interpolated back to the original resolution.

Among the foundational operations in this pipeline is sampling. Extensive research has investigated various sampling techniques for point cloud data, including farthest point sampling (FPS) [22,29,30,59,60], grid sampling [43], random sampling [14], and learning-based sampling [7,20,23,54,55]. Recognized as the most efficacious sampling method, FPS iteratively identifies the point most distant from the current selection as the sampling point [8]. However, this technique assumes a uniform information distribution in the input data. Consequently, identical sampling density is imposed on both feature-rich and feature-sparse regions, resulting in significant information loss in areas with abundant features.

To accommodate the information distribution of the input data, we introduce a method that encodes local feature **dist**ribution statistics directly into point coordinates. These encoded statistics serve as priors to guide the Farthest Point Sampling process, which we refer to as **DistrFPS**. Additionally, we observe that the prevalent k-NN interpolation technique introduces discontinuities into the upsampled domain, adversely affecting prediction accuracy. To mitigate this issue, we employ barycentric interpolation, drawing inspiration from the weight-assigning schemes utilized in n-dimensional linear interpolations and Gouraud shading [10] within computer graphics, thereby enhancing performance.

To validate the efficacy of our approach, we initially demonstrate its capacity to retain input data accurately through signal reconstruction tests on both 2D and 3D signals, which include images and point clouds. In terms of signal reconstruction quality, our approach registers superior scores in both SSIM [48] (for 2D signals) and PSNR (for 2D and 3D signals). When incorporated into neural networks and evaluated against rigorous 3D point cloud segmentation benchmarks, our method exhibits marked improvements, notably surpassing previous results on ShapeNetPart [56] and ScanNet (V2) [6], and delivering a substantial enhancement over earlier methods on S3DIS [1].

In summary, our contributions are as follows:

- We introduce DistrFPS, an innovative sampling technique that leverages the distribution of input information to enhance the preservation of information in subsampled point sets, thereby enabling deeper network layers to concentrate on salient input features.
- We present a continuous interpolation module, which employs barycentric interpolation to engender a continuous feature space in the upsampled domain, subsequently augmenting neural network performance.
- Through comprehensive experiments, we validate the exceptional quality of signal reconstruction and highlight substantial performance gains on demanding 3D point cloud segmentation benchmarks.

2 Related Work

2.1 Point Cloud Learning

Current learning-based techniques for point cloud processing can be grouped into three primary categories: projection-based, discretization-based, and point-based networks.

Projection-based approaches convert 3D point clouds into 2D image planes [3, 16, 19, 21, 36, 41, 46], thereby benefiting from the established methodologies in 2D. While they are capable of handling large-scale inputs, the potential for information loss arises due to occlusions during projection. On the other hand, **Discretization-based techniques** encode the unordered point cloud data in a quantized manner [26, 39]. Subsequently, 3D CNNs, advanced data structures [35, 45] and sparse convolutions [5, 11, 27] are employed to address the challenges of high memory and computational demands. Despite their satisfactory performance, these methods might compromise geometric precision during discretization. In contrast to the aforementioned techniques, our study focuses on processing unordered point sets.

Point-based approaches operate directly on irregular point sets. PointNet [29] initially employs point-wise multi-layer perceptrons (MLPs) to extract per-point features and aggregates this contextual information through a symmetric max-pooling operation. PointNet++ [30] enhances this by introducing set abstraction techniques to capture local features, and it utilizes a hierarchical architecture for global information extraction. Within this architecture, farthest point sampling is employed to subsample point sets to to a more sparse one, and deep features are subsequently learned and upsampled via distance-based interpolation facilitated by skip connections. Subsequent research has largely embraced PointNet++'s efficient learning paradigm, efficient learning paradigm, proposing enhanced modules for local feature extraction and aggregation, including graph-based [18, 47], convolution-based [22, 24, 43, 52, 53], and attention-based [13, 34, 49, 55, 57, 60] methodologies. Our approach diverges from these by re-examining the core modules of PointNet++'s processing pipeline, incorporating guided sampling and continuous interpolation.

2.2 Point Cloud Sampling

Subsampling. The majority of previous studies have employed the farthest point sampling (FPS) approach to produce subsampled point sets [22,25,30,31,34,59,60]. Originally developed as a progressive sampling technique for images, FPS's adaptive version gauges the local function bandwidth across the image, leveraging a weighted distance function to densely sample in areas rich in detail and more sparsely in smoother regions [8]. KPConv [43] utilizes grid sampling to ensure a spatially consistent density among subsampled points. RandLA-Net [14] reduces network latency through the faster random sampling and incorporates a local spatial encoding module to counteract potential information loss. Learning-based subsampling has also been explored, where the subsampling module is designed as a subnetwork to be either jointly optimized [54,55], or learned separately with supervision from downstream tasks [7,20,23]. Our sampling method draws inspiration from [8], in which we directly approximate and encode the local feature information from the input into the distance metric, allowing the sampling method to consider this prior knowledge.

Upsampling. For segmentation tasks, which fundamentally involve dense prediction, an upsampling module is vital to relay the acquired high-level features to each input point. The inverse-distance weighted k-NN interpolation stands as the predominant method for upsampling in point-based techniques [14,17,23,25,30,54,60], attributed to its computational efficiency and straightforward implementation. However, as we demonstrate subsequently, this weight allocation strategy induces discontinuities in the upsampled feature space, compromising the performance of neural networks. Consequently, we advocate for an interpolation module adept at producing a continuous feature space, ensuring the accurate propagation of deep features to their original spatial granularity.

3 Method

Notations. Let $\mathcal{X} = \{(\boldsymbol{p}_i, \boldsymbol{f}_i)\}_i$ be a set of points in an n-dimensional Euclidean space $\mathbb{R}^n$, where the i-th point $\boldsymbol{x}_i$ is comprised of a position component $\boldsymbol{p}_i \in \mathbb{R}^n$ and a feature

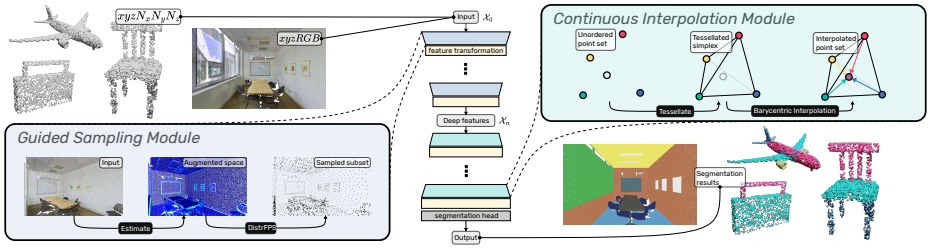


Fig. 1: **Overview of our method.** We adopt a U-Net [30, 37] like architecture (middle) for the point cloud segmentation task. Our method is compatible with any point cloud data that holds features (normals, colors, *etc*.) on each point. In the contracting path (left), we employ a guided sampling module (Sec. 3.2) to allow each subsampling step to be aware of the feature distribution of the input point cloud. In the expansive path (right), we enforce correct feature propagation from the low-resolution deep point features to higher resolutions with a continuous interpolation module (Sec. 3.3).

component $\boldsymbol{f}_i \in \mathbb{R}^f$. For the iterative sampling methods discussed here, we denote $\mathcal{S}$ as the subsampled set of points and $\mathcal{S}_i$ as the subsampled point set after the i-th point $\boldsymbol{x}_i$ is added to the point set, *i.e*. $\mathcal{S}_1 = \{\boldsymbol{x}_1\}, \mathcal{S}_{i+1} = \mathcal{S}_i \cup \{\boldsymbol{x}_{i+1}\}$.

Overview. The method we employ is depicted in Fig. 1. We utilize a hierarchical network design tailored for point cloud segmentation, comprising n encoding levels. This transforms the initial point cloud $\mathcal{X}_0$ into a concise representation on a sparse set of points $\mathcal{X}_n$. Subsequently, the deeply learned features from $\mathcal{X}_n$ are upsampled to the original spatial resolution through n decoding phases. The encoder integrates our proposed guided sampling module (as seen on the left of Fig. 1), while the decoder incorporates the continuous interpolation module (shown on the right of Fig. 1). A shared network is applied after each encoding/decoding block for local feature extraction and aggregation.

In the following, we first revisit the FPS method in Sec. 3.1, laying the groundwork for our guided sampling approach. In Sec. 3.2, we describe details of our sampling technique. Sec. 3.3 highlights the suboptimality of the prevalent k-NN interpolation method in point cloud segmentation tasks, leading us to conceptualize our continuous interpolation module.

3.1 Motivation

FPS Uniformity. As the most successful sampling method in point-based methods, FPS effectively generates a spatially uniform subset of the input [8]. We conceptualize the sampling process as a *cardinality reduction* operation that strives to *preserve information* from the original data. In a scenario where the information in the input data is uniformly distributed, the process of FPS can be interpreted as greedily identifying a point $\boldsymbol{x}_{i+1}$ such that $\boldsymbol{p}_{i+1}$ lies at the location where the already selected point set $\mathcal{S}_i$ holds the least information, thereby ensuring the inclusion $\boldsymbol{x}_{i+1}$ maximizes the information gain to $\mathcal{S}_i$. While this is correct in the uniform distribution case, it does not hold

(a) Original 512^2 image (b) Lifted by δ (c) FPS (stripped color, 4096 points) (d) DistrFPS (stripped color, 4096 points)

Fig. 2: **Sampling result of DistrFPS compared to FPS on a 2D point set.** Significant structures are well preserved from the input data by sampling in the augmented space.

when it comes to intricate visual data, such as 2D imagery or 3D point clouds, where the information distribution is spatially variable. Resorting to this form of sampling can exacerbate data loss in the subsampled set, potentially constraining performance in hierarchical data analysis due to recurrent sampling stages.

Sampling density considerations. Optimally, the density of sampling ought to be contingent upon the high-level semantics inherent in the input data. For semantic segmentation tasks, given sufficient training data, a sampling method should be able to sample densely around regions where the ground-truth label changes — *i.e.* near the boundaries of different semantic regions — while adopting a sparser approach in areas with stable labels. This targeted density allocation enables the neural network to focus its representational power on patterns that distinguish different regions. As ground-truth labels are not available a priori during forward propagation, we propose to guide the sampling process by the local feature information surrounding each point. High-level semantics frequently correspond to changes in these input features, such as colors, normals, and curvatures.

3.2 Guided Sampling

We present the DistrFPS method, depicted in Fig. 1 (left). For a given input point cloud, we first estimate the local feature significance for every point, then leverage this information to guide the sampling, yielding a representative subset of the point cloud.

Sampling density estimation. Given the challenge of analytically determining the information distribution within an input point set, we propose an approximation. We associate a feature significance value, $\delta_{\boldsymbol{p}}$, along with the location $\boldsymbol{p}$ of point $\boldsymbol{x}$. This is estimated using the unbiased standard deviation $\sigma_{\boldsymbol{f}}$ of the normalized features of its neighboring points:

$$\delta_{\boldsymbol{p}} = \sqrt{\frac{\sum_{i=0}^{|\mathcal{N}_{\boldsymbol{x}}|} (\boldsymbol{f}_i - \boldsymbol{\mu}_{\boldsymbol{f}}(\mathcal{N}_{\boldsymbol{x}}))^2}{|\mathcal{N}_{\boldsymbol{x}}| - 1 + \varepsilon}}. \tag{1}$$

Here, $\mathcal{N}_{\boldsymbol{x}}$ represents the set of points proximal to point $\boldsymbol{x}$ based on a specific metric (e.g., k-NN, ball query). $\boldsymbol{\mu}_{\boldsymbol{f}}(\mathcal{N}_{\boldsymbol{x}}) \in \mathbb{R}^f$ is the mean feature value across $\mathcal{N}_{\boldsymbol{x}}$. Additionally, ε is a small positive constant introduced to maintain numerical stability.

This feature significance value $\delta_{\boldsymbol{p}}$ serves as a proxy to guide the sampling density in our approach. Specifically, let $\mathcal{P}_{\text{const}}$ represent the set of locations where the δ values remain relatively constant, and $\mathcal{P}_{\text{var}}$ denote those where δ fluctuates rapidly. A sparse sampling strategy is preferable for locations in $\mathcal{P}_{\text{const}}$ as a consistent standard deviation among neighboring features typically indicates homogeneity. In contrast, locations in $\mathcal{P}_{\text{var}}$ warrant denser sampling due to the distinguishing patterns they exhibit, which help differentiate various homogeneous regions in the input data.

Guided sampling. To optimize the FPS process using the feature significance δ, it's desirable to increase the distance between points within $\mathcal{P}_{\text{var}}$ from others, thus "diluting" the space from $\mathcal{P}_{\text{var}}$. To facilitate this, we enhance the spatial coordinate $\boldsymbol{p} \in \mathbb{R}^n$ by adding a new dimension Δ with the value $\lambda \cdot \delta_{\boldsymbol{p}}$, resulting in an augmented coordinate $\boldsymbol{p}' \in \mathbb{R}^{n+1}$:

$$\boldsymbol{p}' = [\boldsymbol{p}; \lambda \cdot \delta_{\boldsymbol{p}}]. \tag{2}$$

Here, λ represents a constant scalar determining the extent to which points are displaced. It's noteworthy that a λ value of 0 reverts to the conventional FPS method. The augmented point coordinate $\boldsymbol{p}'$ encapsulates both the original spatial location $\boldsymbol{p}$ and the inferred local feature significance $\delta_{\boldsymbol{p}}$. A representative example, with the input point set positioned on a dense 2D grid, is illustrated in Fig. 2. By elevating each point $\boldsymbol{p}$ to a height defined by $\delta_{\boldsymbol{p}}$ (as seen in Fig. 2b), points with fluctuating δ values are distanced from their neighboring points, whereas those with stable δ values remain proximate.

We extend the FPS method to *operate on the augmented $\mathbb{R}^{n+1}$ space rather than the original $\mathbb{R}^n$ space*, designating this modified approach as **DistrFPS**. Owing to the increased dispersion of points at locations with fluctuating δ values, DistrFPS tends to sample more points in these areas, thereby promoting uniformity in the augmented space. As illustrated in Fig. 2, despite having an identical point count of 4096, the subset sampled from the original space (Fig. 2c) is uniformly distributed but lacks distinct features, whereas the subset obtained from the augmented space (Fig. 2d) is readily discernible, capturing structural nuances from the input data.

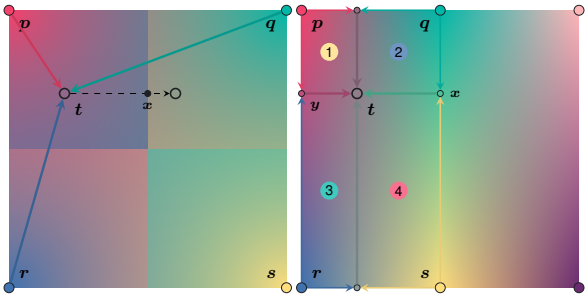
3.3 Continuous Interpolation

Background. Although the guided sampling module can seamlessly integrate with point set encoders, empirical evidence suggests that its direct implementation adversely affects both signal reconstruction quality and segmentation accuracy. This performance decline is attributed to the discontinuities introduced by the interpolation method.

In the decoding stage of U-Net-like architectures [37], as depicted in Fig. 1, the majority of existing works employ k-NN interpolation for upsampling due to its straightforwardness and computational efficiency. Specifically, k-NN interpolation computes the weighted average of the features of the top-k nearest neighbors at a given point $\boldsymbol{p}$ to generate its upsampled features. Let $\boldsymbol{n}_i$ represent the i-th neighboring point of $\boldsymbol{p}$, its assigned weight $w_{\boldsymbol{n}_i}$ is defined as:

$$w_{\boldsymbol{n}_i} = \frac{\frac{1}{d(\boldsymbol{n}_i, \boldsymbol{p})}}{\sum_{j=1}^{k} \frac{1}{d(\boldsymbol{n}_j, \boldsymbol{p})}}, \tag{3}$$

where $d(\cdot, \cdot)$ calculates the distance between two points.



(a) k-NN ($k = 3$) (b) Bilinear

Fig. 3: **Results of color interpolation from four points using different methods on a 2D grid.** Point $\boldsymbol{x}$ lies on the perpendicular bisector of the squares. In the k-NN case, the interpolated color value at point $\boldsymbol{x}$ is ambiguous, while bilinear interpolation generates smoothly transitioning colors across grid boundaries. Colored arrows indicate paths of interpolation, longer arrows have smaller influences on the target point.

Discontinuity of k-NN interpolation. For the sake of clarity, consider the task of interpolating the color $\boldsymbol{f_t}$ of a point $\boldsymbol{t}$ from four reference points $\boldsymbol{p}$, $\boldsymbol{q}$, $\boldsymbol{r}$, and $\boldsymbol{s}$ on a 2D grid. As illustrated in Fig. 3a, when applying k-NN interpolation with $k = 3$, the color determination for point $\boldsymbol{t}$ as it nears $\boldsymbol{x}$ is calculated as follows:

$$\lim_{\boldsymbol{t}\to\boldsymbol{x}^-} \boldsymbol{f_t} = w_{\boldsymbol{p}}\boldsymbol{f_p} + w_{\boldsymbol{q}}\boldsymbol{f_q} + \lim_{\boldsymbol{t}\to\boldsymbol{x}^-} w_{\boldsymbol{r}}\boldsymbol{f_r}, \tag{4}$$

$$\lim_{\boldsymbol{t}\to\boldsymbol{x}^+} \boldsymbol{f_t} = w_{\boldsymbol{p}}\boldsymbol{f_p} + w_{\boldsymbol{q}}\boldsymbol{f_q} + \lim_{\boldsymbol{t}\to\boldsymbol{x}^+} w_{\boldsymbol{s}}\boldsymbol{f_s}. \tag{5}$$

Given that $\lim_{\boldsymbol{t}\to\boldsymbol{x}^-} w_{\boldsymbol{r}} = \lim_{\boldsymbol{t}\to\boldsymbol{x}^+} w_{\boldsymbol{s}}$, the interpolated color at point $\boldsymbol{x}$ remains continuous if, and only if, $\boldsymbol{f_r} = \boldsymbol{f_s}$. Consequently, k-NN interpolation exhibits discontinuity when one among the top-k neighbors transitions to an alternate point. While setting k equivalent to the total number of input points ensures continuity by precluding the alteration of any neighboring point, this approach is computationally prohibitive for inputs with a substantial point count. As we will demonstrate in the signal reconstruction experiments (Sec. 4.2), such discontinuities are further magnified along edges when the input is not uniform.

Continuous interpolation. To provide a foundation for our continuous interpolation approach, we first delve into the continuity properties of linear interpolation. In the realm of 1D linear interpolation, when point $\boldsymbol{p}$ is positioned on the segment between points $\boldsymbol{a}$ and $\boldsymbol{b}$, the corresponding weights are determined as:

$$w_{\boldsymbol{a}} = \frac{d(\boldsymbol{p},\boldsymbol{b})}{d(\boldsymbol{a},\boldsymbol{b})},\ w_{\boldsymbol{b}} = \frac{d(\boldsymbol{p},\boldsymbol{a})}{d(\boldsymbol{a},\boldsymbol{b})}. \tag{6}$$

This ensures the continuity of the interpolated value since $w_{\boldsymbol{a}}$ diminishes to zero as $\boldsymbol{p}$ converges to $\boldsymbol{b}$, and vice versa. For a 2D grid, illustrated in Fig. 3b, employing bilinear interpolation allows the color $\boldsymbol{f_t}$ at point $\boldsymbol{t}$ to be derived by executing linear interpolations across both dimensions. Through algebraic manipulations (details provided in the

supplementary material[1]), $\boldsymbol{f_t}$ can be represented as:

$$\boldsymbol{f_t} = \frac{A_1 \boldsymbol{f_s} + A_2 \boldsymbol{f_r} + A_3 \boldsymbol{f_q} + A_4 \boldsymbol{f_p}}{A}, \tag{7}$$

where A_i denotes the area of the i-th rectangle highlighted in Fig. 3b and A denotes the total area enclosed by points $\boldsymbol{p}$, $\boldsymbol{q}$, $\boldsymbol{r}$, and $\boldsymbol{s}$. This underlines that bilinear interpolation serves as the 2D grid analog of the barycentric interpolation technique, a method prominently employed in computer graphics, with Gouraud Shading [10] being one of its notable applications. Barycentric interpolation, being well-structured for n-dimensional simplices, aptly aligns with our unordered point set scenario. As depicted in Fig. 1 (right), our continuous interpolation framework begins by tessellating the input point set into simplices — triangles for 2D and tetrahedrons for 3D. Subsequently, for each target point, we identify its encompassing simplex and compute barycentric weights to interpolate within a continuous feature domain. We employ Delaunay triangulation [12] for tessellation. In the occasional instance where a target point is external to all simplices, we resort to k-NN interpolation for its upsampled feature computation.

Incorporating barycentric interpolation within point cloud learning has been explored in [38,40]. Notable distinctions between their strategies and ours include: [38,40] perform the interpolation on an auxiliary lattice structure, and require two interpolations during learning: the first one embeds the input point cloud onto the lattice; the second one maps the learned features from the lattice back to the point cloud. In contrast, our method directly tessellates the unordered point cloud, and performs interpolation with barycentric weights once.

4 Experiments

We begin by examining the information preservation abilities of our method through direct signal reconstruction on data that can be represented as point sets (Sec. 4.2). Following this, we evaluate the method's effectiveness on point cloud segmentation tasks, which include semantic segmentation (Sec. 4.3) and object part segmentation (Sec. 4.4). Lastly, we verify our underlying assumptions and design choices in Sec. 4.5.

4.1 Experimental Setup.

Implementation details. In the implementation, the feature significance $\delta_{\boldsymbol{p}}$ is computed by evaluating the per-channel standard deviation of the top 16 proximal points to $\boldsymbol{p}$, followed by averaging these values across all channels. For inputs featuring unoriented normal vectors, the absolute value of each element is taken prior to local feature significance estimation. The point sets are tessellated into simplices using the Delaunay triangulation implementation provided by the SciPy [44] library. The feature aggregation module employed in our primary experiments is PointTransformer [60], while an ablation study is conducted using PointNet [29] as an MLP-based backbone for comparison. Additionally, a learning-based sampling method [54] is used for comparison.

[1] Supplementary material at: https://doi.org/10.5281/zenodo.10494633

All experiments are executed using the PyTorch [28] framework on a single NVIDIA GTX 1080 Ti GPU. Further details on network architectures and hyperparameters can be found in the supplementary material.

Training / Evaluation protocol. Our training and evaluation approach aligns with that of our base model, PointTransformer, as well as other high-performing techniques such as PointNeXt. Before processing, large point clouds are split into smaller, more sparse subparts, allowing the network to remain independent of scene scale. The official implementation of PointTransformer operates with subpart sizes of 80k and a batch size of 16 on four TITAN RTX GPUs. For compatibility with our single-GPU configuration, we have adjusted this to work with 50k points and a batch size of four.

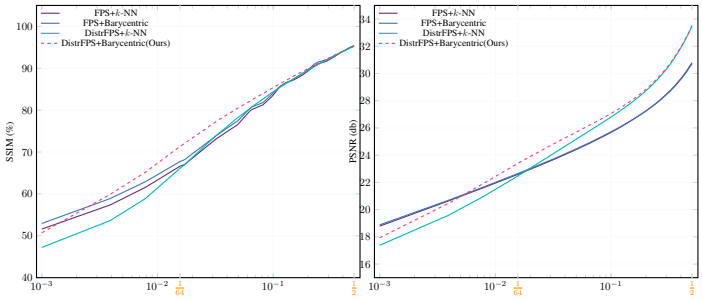
Datasets. Our method has been assessed across several datasets: S3DIS [1], ScanNet (V2) [6], and ShapeNetPart [56] for segmentation tasks, and ModelNet40 [50] for an additional classification experiment within our ablation study. Both S3DIS and ScanNet offer robust benchmarks for indoor scene semantic segmentation. Specifically, S3DIS encompasses 272 rooms across six distinct regions, with each point labeled from one of 13 semantic categories. ScanNet provides a 3D semantic labeling benchmark with 1,613 indoor scenes, divided into 1,201 for training, 312 for validation, and 100 for testing. Each point within these scenes is categorized under one of the 20 semantic labels. ShapeNetPart offers a segmentation benchmark of 16,881 models across 16 categories, each labeled with two to six part labels, summing up to a total of 50 labels. ModelNet40 presents a classification benchmark with 12,311 CAD models distributed among 40 categories, 9,843 of which are designated for training and 2,468 for testing.

Metrics. For signal reconstruction, the metrics employed are PSNR and SSIM [48]. For semantic segmentation, we report mean intersection-over-union across categories (mIoU), mean accuracy across categories (mAcc) and overall segmentation accuracy (oAcc). For object part segmentation, both category mIoU and instance mIoU are reported, in line with established conventions. In the classification task evaluated in the ablation study, mAcc and oAcc are the chosen metrics. Comprehensive definitions of these metrics are provided in the supplementary material.

4.2 Signal Reconstruction

Table 1: **Signal reconstruction.** Metrics are reported with sampling ratio = $\frac{1}{64}$.

Sampling	Interpolation	SSIM↑		PSNR↑		
		camera	lena	camera	lena	Area5
Grid	Bilinear	69.05	64.03	22.85	24.72	-
FPS	k-NN ($k = 3$)	69.87	63.33	21.84	23.86	22.59
	Barycentric	71.00	64.41	21.85	23.86	22.65
DistrFPS	k-NN ($k = 3$)	70.30	61.87	22.65	23.80	22.47
	Barycentric	**75.21**	**67.43**	**24.00**	**25.12**	**23.40**



(a) Images (`lena` and `camera`) (b) Point clouds (`Area5` of S3DIS)

Fig. 4: **Quantitative comparison of signal reconstruction qualities.** Combining guided sampling with continuous interpolation, our method (dashed red curves) shows the best reconstruction quality spanning almost all of reasonable ($\gtrapprox \frac{1}{100}$) sampling ratios.

Fig. 5: **Qualitative results of reconstructing 512^2 images from 4096 sampled points.**

In the signal reconstruction experiments, the data sources are as follows: for 2D point set data, images of "`lena`" and "`camera`" are used, whereas for 3D point set data, we analyze all 68 point clouds from Area 5 of the S3DIS dataset and present the averaged metrics.

For each input, sampling and interpolation are executed to reconstruct the data, utilizing 40 uniformly sampled ratios ranging from $\frac{1}{1000}$ to $\frac{1}{2}$ in logarithmic space. Specifically, four combinations of sampling and interpolation methods are examined: FPS+k-NN, DistrFPS+k-NN, FPS+Barycentric, and DistrFPS+Barycentric. Additionally, the Grid+Bilinear method is applied to 2D signals.

Performance comparison. Fig. 4 presents the average signal reconstruction qualities for images and point clouds. Additionally, with the sampling ratio fixed at $\frac{1}{64}$, reconstruction statistics are displayed in Tab. 1. The findings are consistent across both 2D and 3D inputs. As previously highlighted in Sec. 3.3, the direct application of DistrFPS without modifying the interpolation method offers limited benefits and can occasionally diminish the reconstruction quality due to discontinuities from k-NN interpolation. While

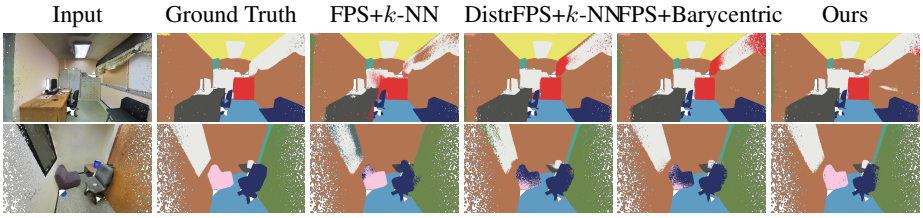


Fig. 6: **Qualitative results on the S3DIS** [1] **dataset.** More visualization results can be found in the supplementary material.

the FPS+Barycentric approach slightly surpasses the FPS+k-NN baseline, its performance remains inferior to grid sampling combined with bilinear interpolation. Notably, DistrFPS+Barycentric significantly outperforms all other methods in both PSNR and SSIM metrics, underscoring its superior information preservation abilities.

The observations above merit further examination. Particularly notable is the fact that when only one of the proposed modifications is applied, the performance improvement is not as substantial as when both are incorporated. The significant performance gain observed when both improvements are incorporated can be attributed to barycentric interpolation's ability to accurately restore features in rapidly changing regions. **1.** For FPS+Barycentric, FPS distributes sampled points uniformly, limiting the full potential of barycentric interpolation. **2.** Conversely, while DistrFPS samples densely in high-frequency regions, k-NN struggles to restore the original features due the discontinuities we previously analyzed.

Visualization. Fig. 5 illustrates the results of 2D signal reconstruction. The FPS+k-NN approach exhibits pronounced jittering artifacts along image edges, a phenomenon that is even more intensified in images reconstructed via DistrFPS+k-NN. In contrast, while the Grid+Bilinear method avoids such jittering, its reconstructed images appear excessively blurred. FPS+Barycentric continues to manifest discontinuities along edges due to inadequate sampling density at these locations. Notably, our DistrFPS+Barycentric approach excels in reconstructing sharp edges and achieves the highest scores in both PSNR and SSIM metrics, thereby demonstrating unparalleled information preservation abilities among the evaluated methods.

4.3 Semantic Segmentation

Performance comparison. Tab. 2 provides quantitative segmentation results for both the S3DIS 6-fold and ScanNet datasets. Equipping our guided sampling method, DistrFPS, coupled with a continuous interpolation module, our technique enables the network to outperforms in all of the three metrics on S3DIS, leading to a noteworthy enhancement, surpassing the baseline model PointTransformer [60] by and absolute 2.6 percent mIoU points. Additionally, our method improves the mIoU by 1.1% on the ScanNet validation set.

Table 2: **Quantitative results of indoor scene semantic segmentation on S3DIS** [1] **6-fold cross validation and ScanNet** [6].

Method	S3DIS			ScanNet	
	mIoU	mAcc	oAcc	Val mIoU	Test mIoU
PointNet [29]	47.6	66.2	78.6	53.5	55.7
DGCNN [47]	56.1	-	84.1	-	-
RSNet [15]	56.5	66.5	-	-	39.4
SPGraph [18]	62.1	73.0	85.5	-	-
PAT [55]	64.2	76.4	-	-	-
PointCNN [22]	65.4	75.6	88.1	-	45.8
PointWeb [59]	66.7	76.1	87.3	-	-
ShellNet [58]	66.8	-	87.1	-	-
JointPointBased [4]	-	-	-	69.2	63.4
PointASNL [54]	68.7	79.0	88.8	63.5	66.6
RandLA-Net [14]	70.0	82.0	88.0	-	-
KPConv [43]	70.6	79.1	-	69.3	68.6
SCF-Net [9]	71.6	82.7	88.4	-	-
CBL [42]	73.1	83.1	89.6	71.3	**70.5**
RepSurf-U [33]	74.3	82.6	90.8	-	-
DeepViewAgg [36]	74.7	-	-	71.0	-
PointNeXt [32]	74.9	83.0	90.3	-	-
PointTransformer [60]	73.5	81.9	90.2	70.6	-
Ours	**76.1**	**84.2**	**91.0**	**71.7**	70.1
Improvement	2.6	2.3	0.8	1.1	-

Visualization. Fig. 6 offers a visualization of semantic segmentation results, benchmarked against three other sampling and interpolation combinations. Notably, our approach excels in distinguishing distinct areas, yielding more refined segmentation results than the baseline. For an extended set of visualization results, readers are directed to the supplementary material.

4.4 Object Part Segmentation

Table 3: **Quantitative results of object part segmentation on the ShapeNetPart** [56] **dataset.**

Method	cat. mIoU	inst. mIoU
StratifiedFormer [17]	**85.1**	86.6
CurveNet [51]	-	86.8
PointNeXt [32]	-	**87.2**
PointTransformer [60]	83.7	86.6
Ours	84.0	86.7
Improvement	0.3	0.1

Performance comparison and analysis. Tab. 3 details quantitative results on the ShapeNetPart dataset [56]. We note that the improvements in this context are less pronounced than

Table 4: **Ratio of points using fall-back k-NN interpolation on different datasets.**

Dataset	#points per input	Network Depth			
		1	2	3	4
ScanNet	~50k	0.72%	1.84%	5.10%	13.00%
S3DIS	~50k	1.30%	3.29%	8.23%	18.86%
ShapeNetPart	~2.6k	6.56%	13.04%	25.74%	54.14%

those observed in the semantic segmentation experiments. Tab. 4 elucidates this by displaying the statistics of points that resort to fallback k-NN interpolation because they do not reside within any simplices (as outlined in Sec. 3.3). Given that the input point clouds from ShapeNetPart comprise significantly fewer points — approximately an order of magnitude less than those from S3DIS/ScanNet — a larger proportion (over 2 times) of points rely on k-NN interpolation. This dependency curtails the impact of our proposed continuous interpolation module. Nonetheless, our approach enhances the baseline by 0.3% category mIoU.

4.5 Ablation Study

We conduct ablation studies to validate our design choices and the effectiveness of the proposed modules.

Table 5: **Ablation study: scaling factor λ.**

λ ($\times\bar{d}$)	0	1	2	4	8	16
mIoU	69.0	69.2	**70.0**	69.5	69.6	69.3

Scaling factor λ. We sought the optimal value for λ as detailed in Sec. 3.2 using the S3DIS fold-5 benchmark. These findings are presented in Tab. 5. Optimal performance is observed when λ equals $2\bar{d}$, where $\bar{d}$ represents the average nearest point distance among the input points. Fig. 7 illustrates the impact of varying λ using a simplified example of sampling a point cloud of a cap, with colors indicating the per-point normal direction. When $\lambda = 0$, DistrFPS is equivalent to FPS, resulting in a uniformly sampled point cloud. This uniformity prevents the network from adequately capturing the information distribution prior of the input. Conversely, for λ values of $4\bar{d}$ or higher, DistrFPS predominantly samples the intersection between the cap's crown and brim, leaving other areas sparsely populated. This irregular point distribution may introduce significant variations in the inputs to the feature learning network, adversely affecting its learning efficiency. We determined that a λ value of $2\bar{d}$ strikes an optimal balance between these considerations and employed this value across all experiments.

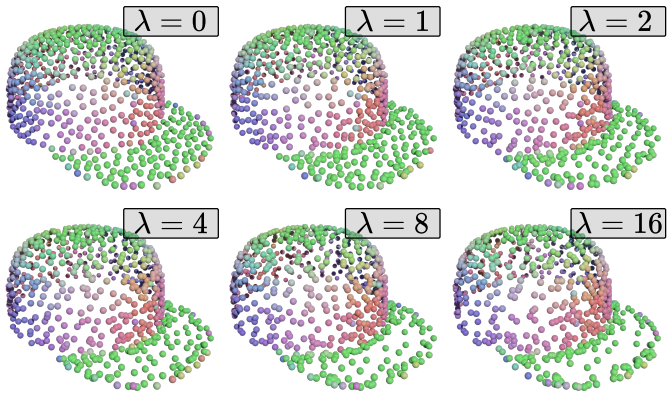


Fig. 7: **Effects of λ ($\times\bar{d}$) on the sampling results in DistrFPS.** Each visualized point cloud contains 655 points (sampled from 2620 points). Notice how the brim contains fewer points with larger values of λ.

Table 6: **Ablation studies: MLP-based backbone and guided sampling.** * denotes our replicated model according to the methods' official implementation.

(a) **MLP-based backbone**

Method	mIoU	mAcc	oAcc
PointNet++ [30]	53.5	-	83.0
PointNet++*	67.5	74.1	89.8
PointNet++* w/ ours	**68.1**	**74.6**	**90.0**

(b) **Guided sampling**

Method	mAcc	oAcc
FPS [60]	90.6	**93.7**
FPS*	90.5	92.3
DistrFPS (Ours)	**91.9**	93.2

MLP-based backbone. In addition to employing an attention-based backbone [60] as described in our primary experiments, Tab. 6a presents the segmentation performance on the S3DIS fold-5 benchmark using PointNet [29] as an MLP-based backbone. By integrating PointNet++ [30] with our proposed guided sampling and continuous interpolation modules, we achieved a 0.6 mIoU point improvement over our replicated model and surpassed the original PointNet++ performance by 14.6 mIoU points.

Guided sampling. To examine the impact of the guided sampling module, we employed only the encoding pathway of the network architecture for evaluation on the ModelNet40 shape classification benchmark. Utilizing the same data preprocessing methods and training/evaluation protocols as our FPS-based baseline [60], we present the mean accuracy across shape categories (mAcc) and overall accuracy (oAcc) in Tab. 6b. By simply substituting the FPS operation with DistrFPS, we observed a significant improvement in mAcc over the baseline by 1.3%, indicating that the guided sampling module effectively enhances the network's ability to discriminate between different inputs.

Table 7: **Ablation Study: continuous interpolation.**

DistrFPS	Barycentric Interpolation	Training Epochs (Loss↓)					6-fold mIoU↑
		1-200	201-400	401-600	601-800	800 (final)	
-	-	36.0	25.0	14.7	6.34	5.89	74.96
✓	-	36.1	25.1	14.6	6.38	5.98	73.33
-	✓	35.5	24.4	13.8	5.98	5.60	75.37
✓	✓	**35.3**	**24.2**	**13.6**	**5.87**	**5.51**	**76.14**

Continuous interpolation. To further demonstrate the impact of our continuous interpolation module, we detail training losses on S3DIS fold-5 and testing mIoU on the S3DIS 6-fold benchmark in Tab. 7. Networks utilizing barycentric interpolation consistently display reduced training losses and superior testing accuracy compared to those using k-NN interpolation. Additionally, Tab. 7 highlights a significant observation: the combination of DistrFPS with k-NN (as shown in the second row) results in the least favorable segmentation accuracy and training loss. Given that (1) Tab. 6b has previously confirmed the efficacy of DistrFPS and (2) the implementation of barycentric interpolation (as seen in the last two rows) elevates the testing mIoU over its k-NN counterparts (the first two rows), this indicates that the discontinuities introduced by the k-NN method, as discussed in Sec. 3.3, indeed constrain segmentation performance.

Table 8: **Ablation study: Learning-based sampler.** AS is the adaptive sampler proposed by Yan *et al*. [54], AS* is our implementation. All the networks are trained on point clouds *without outliers*.

	AS [54]	AS*+k-NN	AS*+bary.	DistrFPS+bary.
+0.1% outliers	-	67.9	**68.0**	54.4
+0.01% outliers	-	68.9	**69.2**	**69.2**
no outlier	62.6	68.7	69.3	**70.0**

Learning-based sampler. Previous studies have investigated the application of learning-based samplers in point cloud segmentation. Yan et al. introduced an outlier-resistant learnable sampler [54] that adjusts each FPS-sampled point's position based on the features of its neighboring points. These adjusted positions are determined using a weighted average of the neighbors, thereby minimizing outlier effects. We integrated the official implementation [2] of this sampler into our framework and report results from tests conducted both with and without intentionally introduced outliers in Tab. 8. The findings indicate that our approach is more vulnerable to previously unseen outliers during training compared to the learning-based sampler. The rationale behind this sensitivity is clear: our technique employs FPS for sampling in the augmented space, and

[2] https://github.com/yanx27/PointASNL

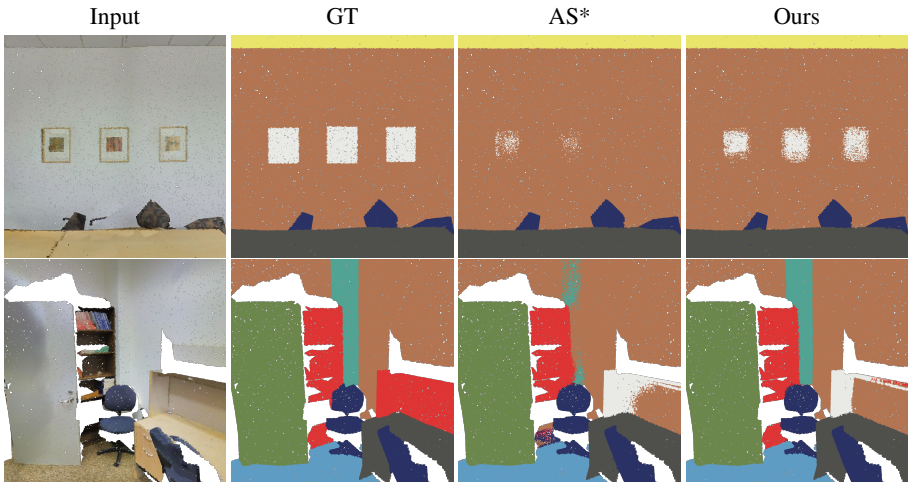


Fig. 8: **Comparison with the learnable sampler by [54].**

FPS is susceptible to outliers, primarily because outliers often lie at greater distances from other points. In the no-outlier setting, Fig. 8 illustrates that our method effectively identifies the boundaries between different semantic regions.

5 Conclusion and Discussion

We introduced a guided sampling and a continuous interpolation module, both designed for integration into point cloud segmentation networks. The guided sampling module employs the DistrFPS technique, leveraging the information distribution of the input point set to minimize information loss during sampling. The continuous interpolation module enhances spatial continuity in interpolated features, resulting in superior neural network performance. Experimental results from signal reconstruction, semantic segmentation, and object part segmentation demonstrate that our modules excel in information preservation and surpass previous methods in point cloud segmentation benchmarks.

Limitations and future work. The presented method exhibits several limitations. Firstly, since DistrFPS conducts FPS in the augmented space, it retains the outlier-sensitivity inherent to FPS. Secondly, our current implementation utilizes SciPy [44] for Delaunay triangulation, which can be computationally expensive for larger inputs (taking approximately 300ms to tessellate 50k points in our experiments). However, since barycentric weights, rather than Delaunay triangulation, is the key to ensuring continuity of the upsampled domain, alternative tessellation methods or implementations, like those suggested by [2], could be adopted to accelerate this process. Lastly, we incorporated an additional parameter, λ, to aid the feature aggregation module in handling inputs of varying spatial densities. In future work, it may be worthwhile to investigate a feature

aggregation design that is resilient to, or independent of, the spatial densities of the input points.

Acknowledgements. The authors would like to thank the CVM 2024 reviewers and program chairs for their valuable feedback. This work is supported by National Natural Science Foundation of China (No. 62032011).

References

1. Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S.: 3d semantic parsing of large-scale indoor spaces. In: CVPR (2016) 2, 9, 11, 12
2. Cao, T.T., Nanjappa, A., Gao, M., Tan, T.S.: A GPU accelerated algorithm for 3D Delaunay triangulation. In: I3D (2014) 16
3. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. In: CVPR (2017) 2
4. Chiang, H.Y., Lin, Y.L., Liu, Y.C., Hsu, W.H.: A unified point-based framework for 3d segmentation. In: 3DV (2019) 12
5. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: CVPR (2019) 2
6. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: CVPR (2017) 2, 9, 12
7. Dovrat, O., Lang, I., Avidan, S.: Learning to sample. In: CVPR (2019) 1, 3
8. Eldar, Y., Lindenbaum, M., Porat, M., Zeevi, Y.Y.: The farthest point strategy for progressive image sampling. TIP (1997) 1, 3, 4
9. Fan, S., Dong, Q., Zhu, F., Lv, Y., Ye, P., Wang, F.Y.: SCF-net: Learning spatial contextual features for large-scale point cloud segmentation. In: CVPR (2021) 12
10. Gouraud, H.: Continuous shading of curved surfaces. TC (1971) 2, 8
11. Graham, B., Engelcke, M., Van Der Maaten, L.: 3d semantic segmentation with submanifold sparse convolutional networks. In: CVPR (2018) 2
12. Guibas, L., Stolfi, J.: Primitives for the manipulation of general subdivisions and the computation of voronoi. TOG (1985) 8
13. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: PCT: Point cloud transformer. CVM (2021) 3
14. Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A.: RandLA-net: Efficient semantic segmentation of large-scale point clouds. In: CVPR (2020) 1, 3, 12
15. Huang, Q., Wang, W., Neumann, U.: Recurrent slice networks for 3d segmentation of point clouds. In: CVPR (2018) 12
16. Kanezaki, A., Matsushita, Y., Nishida, Y.: Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In: CVPR (2018) 2
17. Lai, X., Liu, J., Jiang, L., Wang, L., Zhao, H., Liu, S., Qi, X., Jia, J.: Stratified transformer for 3d point cloud segmentation. In: CVPR (2022) 3, 12
18. Landrieu, L., Simonovsky, M.: Large-scale point cloud semantic segmentation with superpoint graphs. In: CVPR (2018) 3, 12
19. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: CVPR (2019) 2
20. Lang, I., Manor, A., Avidan, S.: Samplenet: Differentiable point cloud sampling. In: CVPR (2020) 1, 3
21. Li, B.: 3d fully convolutional network for vehicle detection in point cloud. In: IROS (2017) 2

22. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: PointCNN: Convolution on x-transformed points. In: NeurIPS (2018) 1, 3, 12
23. Lin, Y., Chen, L., Huang, H., Ma, C., Han, X., Cui, S.: Task-aware sampling layer for point-wise analysis. TVCG (2022) 1, 3
24. Liu, Y., Fan, B., Xiang, S., Pan, C.: Relation-shape convolutional neural network for point cloud analysis. In: CVPR (2019) 3
25. Ma, X., Qin, C., You, H., Ran, H., Fu, Y.: Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In: ICLR (2022) 3
26. Maturana, D., Scherer, S.: VoxNet: A 3d convolutional neural network for real-time object recognition. In: IROS (2015) 2
27. Nekrasov, A., Schult, J., Litany, O., Leibe, B., Engelmann, F.: Mix3D: Out-of-Context Data Augmentation for 3D Scenes (2021) 2
28. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: NeurIPS (2019) 9
29. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep learning on point sets for 3d classification and segmentation. In: CVPR (2017) 1, 3, 8, 12, 14
30. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS (2017) 1, 3, 4, 14
31. Qian, G., Hammoud, H., Li, G., Thabet, A., Ghanem, B.: ASSANet: An Anisotropic Separable Set Abstraction for Efficient Point Cloud Representation Learning. In: NeurIPS (2021) 3
32. Qian, G., Li, Y., Peng, H., Mai, J., Hammoud, H.A.A.K., Elhoseiny, M., Ghanem, B.: PointNeXt: Revisiting PointNet++ with improved training and scaling strategies. In: NeurIPS (2022) 1, 12
33. Ran, H., Liu, J., Wang, C.: Surface representation for point clouds. In: CVPR (2022) 12
34. Ran, H., Zhuo, W., Liu, J., Lu, L.: Learning inner-group relations on point clouds. In: ICCV (2021) 1, 3
35. Riegler, G., Osman Ulusoy, A., Geiger, A.: Octnet: Learning deep 3d representations at high resolutions. In: ICCV (2017) 2
36. Robert, D., Vallet, B., Landrieu, L.: Learning multi-view aggregation in the wild for large-scale 3d semantic segmentation. In: CVPR (2022) 2, 12
37. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI (2015) 4, 6
38. Rosu, R.A., Schütt, P., Quenzel, J., Behnke, S.: LatticeNet: fast spatio-temporal point cloud segmentation using permutohedral lattices. Autonomous Robots (2022) 8
39. Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M., Funkhouser, T.: Semantic scene completion from a single depth image. In: CVPR (2017) 2
40. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: Splatnet: Sparse lattice networks for point cloud processing. In: CVPR (2018) 8
41. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: ICCV (2015) 2
42. Tang, L., Zhan, Y., Chen, Z., Yu, B., Tao, D.: Contrastive Boundary Learning for Point Cloud Segmentation. In: CVPR (2022) 12
43. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.: KPConv: Flexible and deformable convolution for point clouds. In: ICCV (2019) 1, 3, 12
44. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J.: SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods (2020) 8, 16

45. Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X.: O-cnn: Octree-based convolutional neural networks for 3d shape analysis. TOG (2017) 2
46. Wang, Y., Ji, R., Chang, S.F.: Label propagation from ImageNet to 3d point clouds. In: CVPR (2013) 2
47. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. TOG (2019) 3, 12
48. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. TIP (2004) 2, 9
49. Wu, X., Lao, Y., Jiang, L., Liu, X., Zhao, H.: Point Transformer V2: Grouped Vector Attention and Partition-based Pooling. In: NeurIPS (2022) 3
50. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: CVPR (2015) 9
51. Xiang, T., Zhang, C., Song, Y., Yu, J., Cai, W.: Walk in the cloud: Learning curves for point clouds shape analysis. In: ICCV (2021) 12
52. Xu, M., Ding, R., Zhao, H., Qi, X.: PAConv: Position Adaptive Convolution with Dynamic Kernel Assembling on Point Clouds. In: CVPR (2021) 3
53. Xu, Y., Fan, T., Xu, M., Zeng, L., Qiao, Y.: Spidercnn: Deep learning on point sets with parameterized convolutional filters. In: ECCV (2018) 3
54. Yan, X., Zheng, C., Li, Z., Wang, S., Cui, S.: PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In: CVPR (2020) 1, 3, 8, 12, 15, 16
55. 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: CVPR (2019) 1, 3, 12
56. Yi, L., Kim, V.G., Ceylan, D., Shen, I.C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L.: A scalable active framework for region annotation in 3d shape collections. TOG (2016) 2, 9, 12
57. Zhang, C., Wan, H., Shen, X., Wu, Z.: PatchFormer: An efficient point transformer with patch attention. In: CVPR (2022) 3
58. Zhang, Z., Hua, B.S., Yeung, S.K.: Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In: ICCV (2019) 12
59. Zhao, H., Jiang, L., Fu, C.W., Jia, J.: Pointweb: Enhancing local neighborhood features for point cloud processing. In: CVPR (2019) 1, 3, 12
60. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: ICCV (2021) 1, 3, 8, 11, 12, 14