GCD-Sampling: A General Cross-scale Decoupled Sampling for Point Cloud

Tao Dai[1,2,4,*], **Yanzi Wang**[3,*], **Jianyu Xiong**[3], **Yaohua Zha**[3], **Shu-Tao Xia**[3], **Zexuan Zhu**[1,2,4,†]

[1]College of Computer Science and Software Engineering, Shenzhen University
[2]National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University
[3]Tsinghua Shenzhen International Graduate School, Tsinghua University
[4]Shenzhen City Key Laboratory of Embedded System Design, Shenzhen, China
{daitao.edu,wangyanzi0314,zyh1614399882}@gmail.com, xiast@sz.tsinghua.edu.cn, zhuzx@szu.edu.cn

Abstract

Sampling strategy (e.g., fixed farthest point sampling) of point cloud has been an essential step for developing practical solutions in 3D computer vision tasks. Previous fixed sampling is simple, but suffer from suboptimal performance for downstream tasks. To adapt to target networks properly, adaptive sampling methods with trainable parameters have been recently developed to enhance the performance. However, existing adaptive sampling methods still suffer from the *over-coupling* problem of target network, and thus become model-specific, which limits their practical applications. To address this issue, we propose a novel general cross-scale decoupled sampling method (GCD-sampling) for point cloud, which consists of original feature cache, cross-scale feature fusion and convex combination learning for better feature extraction. To reduce the coupling relationship with the target task network, our method only utilizes the point cloud coordinates as the input and output of itself. Besides, we introduce an arbitrary scale structure to enable parameter sharing across multi-scale sampling in point cloud networks. Extensive experiments on different architectures demonstrate the effectiveness of our method over other existing adaptive sampling methods.

Code — https://github.com/Yanzi-Wang/GCD-Sampling-A-General-Cross-scale-Decoupled-Sampling-for-Point-Cloud

Introduction

Due to the large number of point cloud captured by 3D LiDAR sensors, point cloud sampling strategy has become crucial in 3D computer vision tasks, such as automatic driving, augmented reality, and environmental modeling.

To date, various sampling strategies, including fixed and adaptive sampling methods, have been developed. Among them, Fixed sampling (e.g., farthest point sampling, FPS) (Moenning and Dodgson 2003) aims to reduce the number of points in the network layer directly (see Figure 1(a)). Although fixed sampling with great simplicity and efficiency is still widely used in various point cloud networks, they usually produce unsatisfying results. To relieve such issue, adaptive sampling methods with learnable parameters have been recently designed by enhancing the learning capability of networks. For example, Nezhadarya et al. (2020) introduced the Critical Points Layer (CPL), which utilizes max-pooling operations to extract the maximum values of feature maps. Yang et al. (2019) proposed the Gumbel Subset Sampling (GSS), employing the Gumbel softmax function for soft sampling. Yan et al. (2020) introduced an adaptive sampling module by leveraging attention mechanisms to extract adjacency relationships among FPS sampled points. Wu et al. (2023) proposed a point cloud contour sampling method based on attention mechanisms.

Despite the success of adaptive sampling methods, they

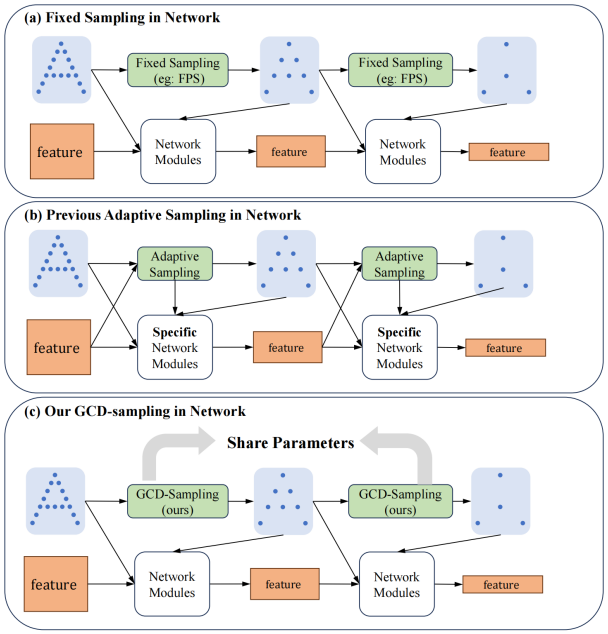

Figure 1: Comparison of various sampling strategies. (a) Fixed sampling is independent on target task network. (b) Existing adaptive sampling model-specific due to over-coupling with target task network. By contrast, (c) our decoupled method is model-agnostic and can be easily embedded into other target task networks.

*The first two authors have equal contributions.
†Corresponding author: Zexuan Zhu

still suffer from *over-coupling* problem for target networks, i.e., sampling methods are highly coupled with target networks. Although such coupling strategy can enhance the performance of target network, it causes sampling become model-specific, which heavily limits the real applications in different kinds of networks. As shown in Figure 1(b), existing adaptive sampling methods not only sample the coordinates of the point cloud, but also participate in the feature mapping in the network, resulting in a high degree of coupling with the original network module. Besides, these methods maintain an independent sampling layer at each scale, resulting in a lack of multi-scale information understanding for each sampling layer.

To address these issues, we propose a general cross-scale decoupled sampling for point cloud, named GCD-sampling, which is general and can be easily embedded in other hierarchical network architectures. As shown in Figure 1(c), our GCD-sampling solely samples point coordinates, remaining uncoupled from the original network's feature extraction, making it model-independent. It shares parameters across different scales, improving multi-scale feature extraction with fewer additional parameters. In order to achieve its reusability at any scale and improve its cross-scale perception capability, we designed an arbitrary scale structure, which consists of three modules: *original feature cache*, *cross-scale feature fusion* and *convex combination learning*. The original feature cache extracts and stores features from the initial sampling for later use. The cross-scale feature fusion module extracts current scale features, and combines them with those from the cached original point cloud. Finally, the convex combination learning module learns local convex combination coefficients from the fused features to generate the final sampling points. The main contributions are summarized as follows:

- We propose a general cross-scale decoupled point cloud sampling method, named GCD-sampling, which can be embedded in hierarchical network architecture to improve the learning ability of the network.
- To reduce the coupling relationship with target task network, our method only utilizes the point cloud coordinates as the input and output of itself. Besides, we introduce an arbitrary scale structure to enable parameter sharing across multi-scale sampling in point cloud networks. It allows for multi-scale perception and reusability, so as to introduce fewer additional parameters while bringing greater performance improvement.
- Extensive experiments on different downstream tasks under various target networks have shown the superiority of our method over other state-of-the-art sampling methods, in terms of performance and computational cost.

Related Work

Deep Learning on Point Clouds

Recently, learning-based point cloud processing techniques (Qi et al. 2017a; Zhao et al. 2021; Wu et al. 2022; Pang et al. 2022; Liang et al. 2024; Wu et al. 2024) have already achieved great success. PointNet (Qi et al. 2017a) first proposed point cloud analysis that directly uses the 3D coordinates of each point. It employs multi-layer perceptions (MLP) and max-pooling to learn the global features of the point cloud. Building on this, PointNet++ (Qi et al. 2017b) adds set abstraction to hierarchically capture features of local point sets, using farthest point sampling (FPS) for effective feature aggregation. Convolution-based methods (Li et al. 2018; Lin et al. 2020; Thomas et al. 2019; Wu, Qi, and Fuxin 2019; Zhao et al. 2019; Xu et al. 2021) bring the idea of convolution to the point cloud. For example, PointCNN (Li et al. 2018) applies traditional convolution on point clouds by standardizing the order of neighboring points. Graph-based methods (Chen, Fragonara, and Tsourdos 2021; Lin, Huang, and Wang 2020; Zha et al. 2023, 2024) use graph structure to analyze point clouds. For example, DGCNN (Wang et al. 2019) dynamically computes local graphs for extracting geometric information. These networks directly process point cloud data, successfully integrating deep learning into this modality and achieving excellent results in various downstream tasks. The PointMLP framework (Ma et al. 2022) employs a pure MLP network structure without complex local geometric information extraction modules, hierarchically aggregating local features and enhancing robustness with lightweight geometric affine modules. Qian et al. (2022) improved PointNet++ by optimizing training strategy and scaling model size, which enhances scalability through residual structures and bottleneck designs. In these hierarchical networks, point cloud sampling is crucial. Fixed algorithms like FPS are commonly used, but they limit the learning ability of the network.

Adaptive Point Cloud Sampling

Dovrat, Lang, and Avidan (2019) first introduced the S-Net point cloud sampling network, which employs deep learning to adaptively sample point clouds for downstream tasks. Lang, Manor, and Avidan (2020) improved S-Net by proposing SampleNet, which includes a soft projection process. Both S-Net and SampleNet focus on pre-sampling within task networks.

Several adaptive sampling methods have emerged for the sampling process within the network. Nezhadarya et al. (2020) introduced the Critical Points Layer (CPL), which retains important points by counting and sorting the maximum occurrences of features in intermediate layers. Yan et al. (2020) proposed the Adaptive Sampling (AS) module, using an attention mechanism to better capture relationships between adjacent points. Wu et al. (2023) proposed a contour sampling method based on attention, selecting key points directly from a sample set instead of generating new ones. However, these adaptive methods are model-specific and lack multi-scale perception and reusability, limiting their generality and adaptability.

Method

Motivation

Unlike previous adaptive sampling methods (Yang et al. 2019; Nezhadarya et al. 2020; Yan et al. 2020; Wu et al. 2023) that are model-specific, we aim to develop a more

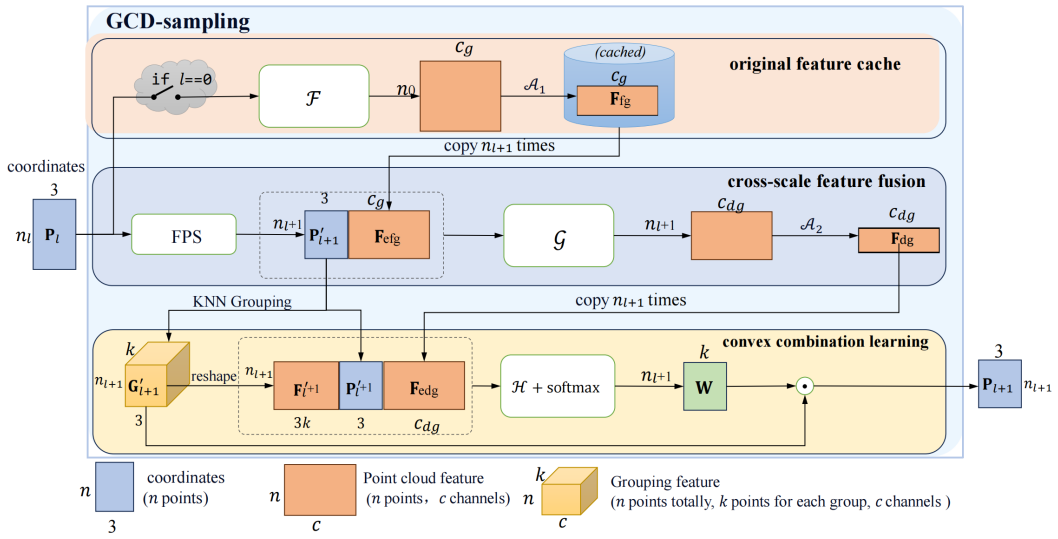

Figure 2: Network structure diagram of our GCD-sampling, which consists of original feature cache, cross-scale feature fusion and convex combination learning.

general adaptive sampling method. In practice, an ideal sampling method should satisfy the following requirements. First, it can serve as a network-agnostic sampling layer that can be plug-and-play without changing the original network structure; second, it can share the same set of parameters across sampling at different scales without extra computational burden.

To this end, we design a simple yet effective sampling method, named GCD-sampling, as below:

1. **GCD-sampling only use the point cloud coordinates as the input and output.** For point cloud networks that use fixed sampling methods, sampling is independent of the intermediate features of the network. Therefore, in order to be able to plug the GCD-sampling into any network immediately, we make this constraint on the GCD-sampling.
2. **For a single GCD-sampling instance, it can handle point clouds at any scale and sampling at any rate.** In a point cloud network, the number of points in the point cloud is different in each layer of the network, and the sampling rate may also be different. Therefore, this arbitrary scale structure design allows us to share the same GCD-sampling instance across different layers of the network, introducing multi-scale awareness while saving additional parameters.

Structure of GCD-sampling

Overview In the l-th sampling layer of network, consider the point coordinate input of the l-th sampling layer is represented by $\mathrm{P}_l \in \mathbb{R}^{n_l \times 3}$, the overall sampling process of GCD-sampling is formalized as :

$$\mathrm{P}_{l+1} = \text{GCD-sampling}_{\theta}\left(\mathrm{P}_l, s\right), l \in 0, 1, \cdots, L-1 \qquad (1)$$

where L represents the number of sampling layers, θ denotes the learnable network parameters within the sampling layer, n_{l+1} represents the number of points for next layer, s represents the sampling rate, and the output $\mathrm{P}_{l+1} \in \mathbb{R}^{n_{l+1} \times 3}$ corresponds to the final sampled points, satisfying $n_{l+1} = \frac{n_l}{s}$.

The network architecture flow of GCD-sampling is illustrated in Figure 2. Within GCD-sampling, we successively design three modules: *original feature cache*, *cross-scale feature fusion* and *convex combination learning*. These three modules work together to extract cross-scale features from point clouds and generate the final sampled points.

Original Feature Cache In the inference of the point cloud network, the point cloud is sampled several times and gradually sparse. If only the point cloud of the current layer is considered in each sampling, the complete semantic information of the point cloud will be lacking. Since the sampling process is progressive, GCD-sampling always receives the densest point cloud (the original point cloud) P_0 firstly, so we design a gating mechanism to extract the original point cloud feature and cache the feature.

The simple gating mechanism ensures that this module will only be enabled if l equals 0. In this case, a lightweight PointNet structure (Qi et al. 2017a) is employed here as the perception network for the entire point cloud, facilitating global feature extraction:

$$\mathrm{F}_{\mathrm{fg}} = \mathcal{A}_1\left(\mathcal{F}\left(\mathrm{P}_0\right)\right), \qquad (2)$$

where $\mathcal{F}$ functions to map features for each point, thus $\mathcal{F}\left(\mathrm{P}_0\right) \in \mathbb{R}^{n_{l+1} \times c_{\mathrm{g}}}$. The structure of $\mathcal{F}$ is designed with three layers, comprising a fully connected layer, a batch normalization (BN) layer, and a ReLU activation function. $\mathcal{A}_1$ denotes the aggregation function, utilizing max pooling for aggregation:

$$\mathcal{A}_1\left(\mathrm{F}\right) = \max(\mathrm{F}[:,0]), ..., \max(\mathrm{F}[:,\mathrm{c_g}-1]), \mathrm{F} \in \mathbb{R}^{n_0 \times c_g}, \qquad (3)$$

where max represents the operation of maximizing the vector. In this way, $\mathrm{F}_{\mathrm{fg}} \in \mathbb{R}^{1 * c_{\mathrm{g}}}$ represents the global feature of

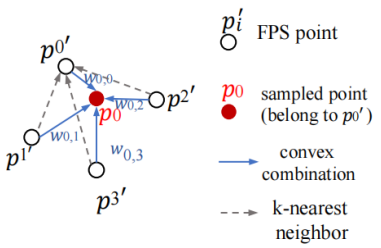

Figure 3: The schematic diagram illustrating the convex learning.

the original point cloud, and then it would be cached. When $l > 0$, the above calculation is not repeated, and F_{fg} is directly fetched from the cache. When a new round of inference is started, since l returns to 0 and the gating condition is satisfied, the cache is naturally updated.

Cross-scale Feature Fusion In this module, we fuse the cross-scale features of point clouds. FPS is applied to P_l firstly, which is regarded as the initialization of final sampled point:

$$\mathrm{P}'_{l+1} = \mathrm{FPS}(\mathrm{P}_l, s), \tag{4}$$

where the result $\mathrm{P}'_{l+1} \in \mathbb{R}^{n_{l+1} \times 3}$, and s denotes the sampling rate, since farthest point sampling is a non-learnable sampling algorithm, it can be employed with any rate s. Next, the cached original features F_{fg} could be obtained in the previous module, are duplicated n_{l+1} times along the point dimension to yield $\mathrm{F}_{\mathrm{efg}} \in \mathbb{R}^{n_{l+1} \times c_{\mathrm{g}}}$. Then, $\mathrm{F}_{\mathrm{efg}}$ is concatenated with the sampled three-dimensional spatial coordinates P'_{l+1} along the feature dimension, serving as the input to later perception network for the sampled point cloud:

$$\mathrm{F}_{\mathrm{dg}} = \mathcal{A}_2\left(\mathcal{G}\left([\mathrm{P}'_{l+1}; \mathrm{F}_{\mathrm{efg}}]\right)\right). \tag{5}$$

This process fuses the original features with the current scale features, where $\mathrm{F}_{\mathrm{dg}} \in \mathbb{R}^{1 * c_{\mathrm{dg}}}$ contains the fused feature, and [;] denotes concatenation along the feature dimension; the structure of $\mathcal{G}$ is similar to $\mathcal{F}$, also consisting of three layers, comprising a fully connected layer, a batch normalization layer, and a ReLU activation function. $\mathcal{A}_2$ represents the feature aggregation function, which is defined as below:

$$\mathcal{A}_2\left(\mathrm{F}\right) = \left[\frac{1}{n_0}\sum_{i=1}^{n_{l+1}} \mathrm{F}_{i,0}, \ldots, \frac{1}{n_0}\sum_{i=1}^{n_{l+1}} \mathrm{F}_{i,c_{dg-1}}\right]^T, \mathrm{F} \in \mathbb{R}^{n_{l+1} \times c_{dg}}. \tag{6}$$

Notably, inspired by Huang et al. (2022), in order to enable the network to perceive the specific scale of the sampling, a summation and dividing by the number original point cloud diversity n_0 instead of an average pooling operations (dividing by n_{l+1}). This allows the network to introduce a prior on the sampling scale through the difference in feature magnitudes: when the number of points after sampling is large, the aggregated feature values are relatively small, and vice versa.

Convex Combination Learning The final step involves obtaining the positions of the final sampled points. To confine the range of sampled points without introducing additional loss functions, this module models the positions of the sampled points as the convex combination of the k-nearest neighbor positions of each point in the farthest point sampled points P'_{l+1}, with the coefficients of this convex combination learned by a neural network. Specifically, this process performs k-nearest neighbor queries and indexing operations for each point in the P'_{l+1} point cloud, obtaining grouped point cloud coordinates $\mathrm{G}'_{l+1} \in \mathbb{R}^{n_{l+1} \times k \times 3}$, where the k nearest neighbor points are sorted by distance from nearest to farthest. Subsequently, their coordinates are concatenated with the respective point's coordinates to form the feature vectors, which can be implemented as a reshape operation:

$$\mathrm{F}'_{l+1} = \mathrm{reshape}(\mathrm{G}'_{l+1}) \tag{7}$$

$\mathrm{F}'_{l+1} \in \mathbb{R}^{n_{l+1} \times 3k}$. Furthermore, the features F_{dg} obtained in earlier module are replicated n_{l+1} times in the point dimension, yielding $\mathrm{F}_{\mathrm{edg}} \in \mathbb{R}^{n_{l+1} \times c_{\mathrm{dg}}}$, which are then concatenated with P'_{l+1} and F'_{l+1}. Through network layer $\mathcal{H}$, the feature dimensions are mapped to k, and convex combination coefficients are learned via a softmax layer, as seen in Figure 3. This process is formalized as below:

$$\begin{aligned} \mathrm{H} &= \left\{h_{i,j}\right\}_{n_{l+1} \times k} = \mathcal{A}\left(\mathcal{H}\left([\mathrm{F}'_{l+1}; \mathrm{P}'_{l+1}; \mathrm{F}_{\mathrm{edg}}]\right)\right), \\ \mathrm{W} &= \left\{w_{i,j}\right\}_{n_{l+1} \times k}, \text{where} \quad w_{i,j} = \frac{e^{h_{i,j}}}{\sum_m e^{h_{i,m}}}, \quad \forall i, \end{aligned} \tag{8}$$

where, $\mathcal{H}$ is similiar to $\mathcal{F}$, which consists of a fully connected layer, a batch normalization layer, and a ReLU activation function. $\mathrm{H}, \mathrm{W} \in \mathbb{R}^{n_{l+1} \times k}$, and W is the final convex combination coefficients. Finally, weighted by W, the k nearest neighbor grouping G'_{l+1} of the FPS point cloud yields the sampled points P_{l+1}, denoted as:

$$\begin{aligned} \mathrm{P}_{l+1} &= \left\{p_{i,j}\right\}_{n_{l+1} \times 3}, \\ \text{where } p_{i,j} &= \sum_m w_{i,j} \cdot g'_{i,m,j}, \quad \forall i, j. \end{aligned} \tag{9}$$

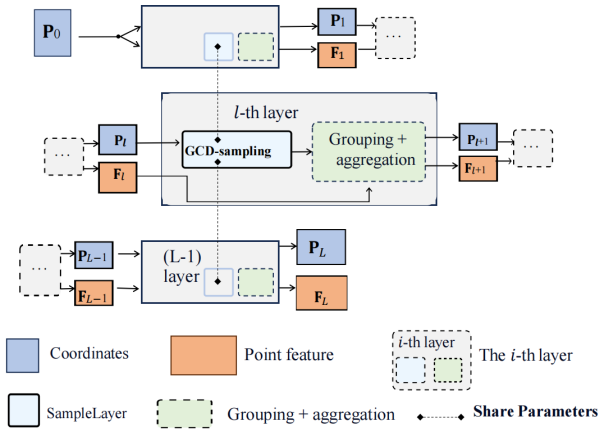

Figure 4: The position diagram of GCD-sampling

GCD-sampling in Point Cloud Networks

This subsection delineates the position of GCD-sampling within task networks, as seen in Figure 4. Although the details of point cloud hierarchical network structures (Qi et al.

2017b; Li et al. 2018; Ma et al. 2022; Qian et al. 2022) may vary across different categories, they share a certain common form in each layer: Consider the input of the l-th layer is represented by $(\mathrm{P}_l, \mathrm{F}_l)$, where $\mathrm{P}_l \in \mathbb{R}^{n_l \times 3}$ is the spatial coordinates of the point cloud for this layer the n_l represents the number of points for this layer, and $\mathrm{F}_l \in \mathbb{R}^{n_l \times c_l}$ represents the feature vectors of individual points in the input point cloud for this layer. The responsibility of each layer involves point sampling and feature mapping, serving as input for the next layer: $(\mathrm{P}_l, \mathrm{F}_l) \longrightarrow (\mathrm{P}_{l+1}, \mathrm{F}_{l+1})$. And inside each layer are three typical processes: point cloud sampling, local grouping and feature mapping/aggregation (Qi et al. 2017b).

The usage pattern of GCD-sampling is embedding it within these hierarchical network architectures to replace conventional point cloud sampling. For each layer of the hierarchical network architecture, GCD-sampling substitutes the conventional point cloud sampling process. Across different network layers, the proposed GCD-sampling shares network parameters among layers. The properties of GCD-sampling, as described earlier, including arbitrary point number inputs and sampling at arbitrary rates, enable it to utilize the same set of network parameters to complete the sampling process at each layer within the network.

Experiments

To validate the effectiveness of GCD-sampling, we employ four representative networks with hierarchical architectures as baseline models, including PointNet++ (Qi et al. 2017b), PointCNN (Li et al. 2018), PointMLP (Ma et al. 2022), and PointNeXt (Qian et al. 2022).

To further demonstrate the superiority of GCD-sampling, we compared it with two types of adaptive sampling layers: ASNL-AS (Wu et al. 2023) and APES (Yan et al. 2020). Both of these methods lack the ability for multiscale reuse, thus requiring the deployment of independently parameterized network layers for sampling at each level.

Classification

Main Results. We evaluate our model on the datasets ModelNet40 (Wu et al. 2014) and ScanObjectNN (Uy et al. 2019) for classification. We use Overall Accuracy (OA) and Mean Class Accuracy (mAcc) to evaluate the classification performance. Specifically, ASNL-AS (Wu et al. 2023), APES (Yan et al. 2020), and our GCD-sampling are embedded into the respective encoder parts of the network structures. The networks are trained from scratch, with the same number of epochs, learning rate, optimizer, and other hyperparameters as the baseline experiments without embedding GCD-sampling. Table 1 presents the evaluation of the classification performance in the ModelNet40 and ScanObjectNN datasets. Across the experiments with the four typical network architectures, the classification accuracy achieved when using GCD-sampling exceeds that achieved without GCD-sampling on both datasets, demonstrating the effectiveness of GCD-sampling in classification tasks. Furthermore, in comparison with the other two types of adaptive sampling layers, GCD-sampling also demonstrates superior performance.

Efficiency Compared to fixed sampling processes, adaptive sampling methods introduce additional parameters. The statistical results about parameter statistic are alse presented in Table 1. For ASNL-AS and APES, a significant increase in parameter count is observed after their incorporation, as each sampling operation entails establishing adaptive sampling layers with independent parameters. In contrast, for the proposed GCD-sampling, since parameters are shared across layers, the addition of GCD-sampling to the four different networks results in an increase of only 0.08M parameters, equivalent to the parameter count of a single GCD-sampling. This increment is consistently lower than the parameter count observed with ASNL-AS and APES sampling schemes. As for time efficiency, we report the FLOPS (G) and inference time (ms per instance) in Table 1 respectively. These experiments are tested on one NVIDIA TITAN X (Pascal) GPU and 16 cores Intel Xeon CPU E5-2697 v4 @ 2.30GHz CPU. All the results are measured using 32×1024 (batch size 32, number of points 1024) as input. Our method achieves more efficient results than previous methods in both FLOPS and inference time.

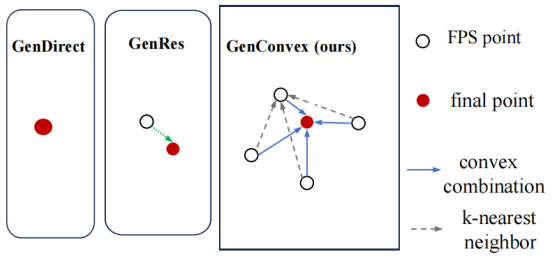

Figure 5: The schematic diagram illustrating the design of comparative experiments for different sampling point generation modes.

Plug-and-Play. As the sampling layer designed in this chapter can replace farthest point sampling (FPS) without altering any other network structures within the task network, further tests on the network compatibility of GCD-sampling were conducted in this section. Specifically, experiments were designed to fine-tune the sampling layer by loading pre-trained network weights. In these experiments, the pre-trained parameters of the task network were loaded, and a randomly initialized GCD-sampling replaced its FPS layer. The remaining network parameters were kept fixed, and only the parameters of GCD-sampling were trained. The experiments were conducted on the ScanObjectNN dataset, and the results are presented in Table 2. It can be observed that when only the GCD-sampling parameters (with trainable parameter count of 0.08M) were fine-tuned, the performance was further improved, even surpassing that achieved when training the entire network from scratch. This further demonstrates the enhancement in network learning capability facilitated by the proposed adaptive sampling. In contrast, the performance of training the entire network with pre-trained parameters deteriorated compared to training only GCD-sampling. This is attributed to the instability of randomly

Network structure	Sampling method	ModelNet40		ScanObjectNN (PB_T50_RS)		Efficiency metrics		
		OA(%)	mAcc(%)	OA(%)	mAcc(%)	Params(M)	FLOPS(G)	Inference Time(ms/ins.)
PointNet++	FPS	91.82	87.94	77.83	75.37	1.51	1.71	1.04
	ASNL-AS	92.03	87.94	77.91	75.38	1.66(+0.15)	2.63(+0.92)	1.16(+0.12)
	APES	92.06	87.98	78.07	75.43	1.70(+0.19)	2.75(+1.04)	1.18(+0.14)
	GCD-sampling(ours)	**92.10**	**88.18**	**78.35**	**75.56**	**1.59(+0.08)**	**2.54(+0.83)**	**1.14(+0.10)**
PointCNN	FPS	92.06	88.90	78.66	76.55	0.60	0.86	7.05
	ASNL-AS	92.18	89.92	78.71	76.54	0.69(+0.09)	1.70(+84)	9.58(+2.53)
	APES	92.38	89.58	78.90	76.61	0.79(+0.19)	1.84(+0.98)	10.49(+3.44)
	GCD-sampling(ours)	**92.59**	**89.98**	**79.11**	**76.75**	**0.68(+0.08)**	**1.53(+0.67)**	**7.39(+0.34)**
PointMLP	FPS	92.99	89.77	87.23	85.91	13.20	31.35	10.36
	ASNL-AS	93.03	89.79	87.34	85.81	13.77(+0.57)	32.45(+1.10)	11.83(+1.47)
	APES	93.11	90.05	87.30	85.66	13.39(+0.19)	32.42(+1.07)	11.61(+1.25)
	GCD-sampling(ours)	**93.31**	**90.31**	**87.54**	**86.13**	**13.28(+0.08)**	**32.32(+0.97)**	**11.48(+1.12)**
PointNeXt	FPS	93.15	90.22	87.40	86.10	1.42	3.64	1.03
	ASNL-AS	93.19	90.23	87.44	86.21	2.01(+0.59)	4.66(+1.02)	1.15(+0.12)
	APES	93.27	90.07	87.65	86.26	1.71(+0.29)	4.64(+1.00)	1.14(+0.11)
	GCD-sampling(ours)	**93.35**	**90.33**	**87.86**	**86.28**	**1.50(+0.08)**	**4.55(+0.91)**	**1.13(+0.10)**

Table 1: The experimental results of GCD-sampling on the ModelNet40 and ScanObjectNN classification datasets

Network structure	Load weights of (1)	unfreeze	trainable params (M)	OA(%)	mAcc(%)
(1) PointNeXt	-	-	1.42	87.40	86.10
(2) PointNeXt +GCD	✗	✗	1.50(+0.08)	87.86	86.28
(3) PointNeXt +GCD	✓	✗	1.50(+0.08)	83.07	80.04
(4) PointNeXt +GCD	✓	✓	0.08	**88.41**	**87.44**

Table 2: Fine-tuning sampling layer experimental results for loading trained networks. GCD: GCD-sampling.

initialized GCD-sampling during the initial training stages, which affects the update of the well-trained parameters in other parts of the network.

Segmentation

We assess our model using the ShapeNetPart dataset citeyi2016scalable for the point cloud part segmentation task. Similar to the classification task, baseline models including PointNet++ (Qi et al. 2017b), PointCNN (Li et al. 2018), PointMLP (Ma et al. 2022), and PointNeXt (Qian et al. 2022) were trained without any adaptive sampling methods. This experiment utilized Instance Mean Intersection over Union (ins.mIoU) and Class Mean Intersection over Union (cls.mIoU) to measure the performance of segmentation tasks.

Note that in segmentation tasks, the task network is typically divided into two parts: an encoder and a decoder. In this experiment, GCD-sampling is only embedded into the encoder part to replace the farthest point sampling without altering the structure of the decoder. Additionally, this experiment also compared GCD-sampling with two other types of adaptive sampling layers, ASNL-AS (Yan et al. 2020) and APES (Wu et al. 2023). The experimental results are shown in Table 3. The experiments on these four network structures demonstrate that GCD-sampling can also provide benefits to downstream tasks in part segmentation and outperforms the other two types of adaptive sampling layers.

Network	Sampling method	ins.mIoU(%)	cls.mIoU(%)
PointNet++	FPS	85.06	81.71
	ASNL-AS	85.25	81.91
	APES	84.97	81.62
	GCD-sampling	**85.54**	**82.09**
PointCNN	FPS	86.04	83.85
	ASNL-AS	86.31	83.87
	APES	86.02	83.75
	GCD-sampling	**86.50**	**83.96**
PointMLP	FPS	86.10	84.52
	ASNL-AS	86.18	84.58
	APES	85.98	84.43
	GCD-sampling	**86.45**	**84.78**
PointNeXt	FPS	86.65	84.41
	ASNL-AS	86.78	84.60
	APES	86.61	84.30
	GCD-sampling	**86.90**	**84.72**

Table 3: The experimental results of GCD-sampling on the ShapeNetPart part segmentation dataset

Visualization

We have added a comparison of the visual results of GCD-sampling and FPS in Figure 6. We note that compared with FPS sampling, points sampled by GCD-sampling have two characteristics: (1) Sampled points are not limited to the original point cloud; (2) The sampled points focus on some areas while maintaining the basic outline. It is consistent

Network structure +Sampling method	Method of generated points	w/ CD loss	w/o CD loss
PointNet++ +GCD-sampling	GenDirect	75.67	53.73
	GenRes	78.07	58.67
	GenConvex	78.10	**78.35**
PointCNN +GCD-sampling	GenDirect	76.97	54.17
	GenRes	78.78	59.65
	GenConvex	78.80	**79.11**
PointNeXt +GCD-sampling	GenDirect	86.60	59.17
	GenRes	87.23	64.55
	GenConvex	87.25	**87.54**
PointNeXt +GCD-sampling	GenDirect	86.92	60.51
	GenRes	87.06	66.24
	GenConvex	87.40	**87.86**

Table 4: Comparative experimental results of sampling point generation methods

with the original intention of our design.

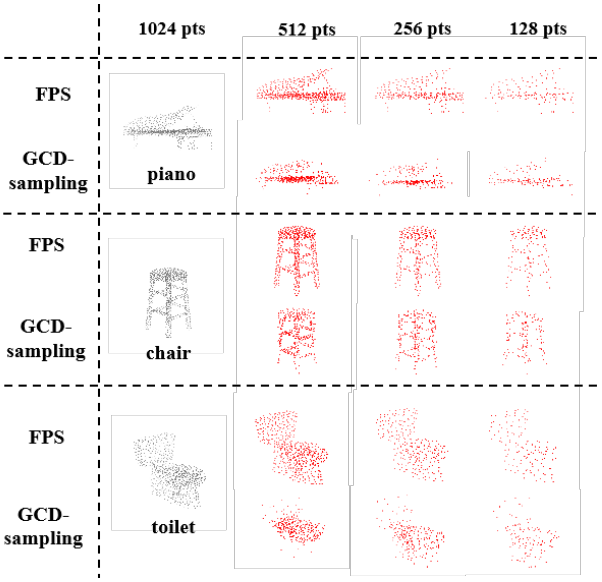

Figure 6: Visualization results of GCD-sampling and FPS.

Ablation Study

Generation Mode of Sampled Points. In this subsection, we conducted experiments to test the usability and effectiveness of our method for learning convex combination coefficients (referred to as GenConvex). As illustrated in Figure 5, we compared it with two other generation modes: GenDirect, which directly generates sampled points, and GenRes, which generates residuals to the farthest sampling positions. Each method was tested in two scenarios: one with a shape constraint loss function (the chamfer distance loss) and one without.

The results are shown in Table 4. Without convex combination coefficients (in GenDirect and GenRes), the downstream task performance suffers significantly due to the unconstrained spatial generation of points. Adding the chamfer distance loss improves results. GenDirect, which directly learns the positions of sampled points, exhibits the poorest performance among the tested configurations, as it faces the greatest difficulty in learning absolute coordinates. In contrast, the proposed method of learning convex combination coefficients maintains satisfactory performance without the need for the chamfer distance loss function. Conversely, the introduction of the chamfer distance loss function restricts the learning space of the sampled points, resulting in a slight performance decline.

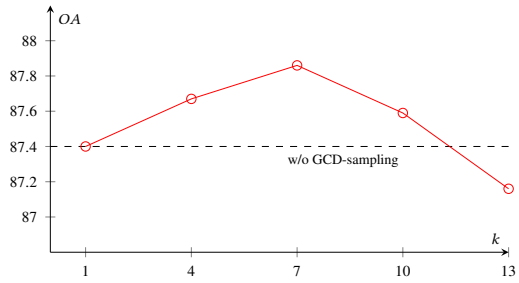

Figure 7: Experimental results of the influence of convex combination coefficients of k-nearest neighbors on performance

***k*-nearest Neighbors in Convex Learning.** As the GCD-sampling learns the convex combination coefficients of the k nearest neighbors of the farthest point sampled points, this section designs experiments to investigate the impact of the choice of k on the network. When $k = 1$, GCD-sampling only learns the combination coefficients of the nearest neighbors, keeping the baseline FPS points unchanged. As expected, it achieves performance consistent with not using GCD-sampling. However, as the value of k gradually increases, a larger range is assigned to the convex hull where the final sampled points lie, leading to performance improvement facilitated by the adaptive framework. Yet, when k becomes excessively large, the increased freedom of the final sampled points imposes a heavier learning burden on the network, resulting in performance degradation. It can be observed that within the range of k values from 1 to 10, the proposed method consistently provides gains to the network, with the sensitivity to the choice of parameter k being manageable.

Conclusion

In this paper, we propose a general cross-scale decoupled point cloud sampling method, named GCD-sampling. Unlike the previous methods that suffer from over-coupling problem with target task networks, our method is decoupled from the original network's feature extraction by solely sampling point coordinates, thus making it model-agnostic. Besides, we present an arbitrary scale structure in GCD-sampling to enable it to have multi-scale perception and reusability, and bring greater semantic perception performance improvement while introducing fewer additional parameters when embedding into point cloud neural networks. Extensive experiments on various point cloud networks with our GCD-sampling verify the superiority of ours over other adaptive sampling methods.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China, under Grant (62302309, 62471310) and the Shenzhen Science and Technology Program (JCYJ20220818101014030).

References

Chen, C.; Fragonara, L. Z.; and Tsourdos, A. 2021. GAPointNet: Graph attention based point neural network for exploiting local feature of point cloud. *Neurocomputing*, 438: 122–132.

Dovrat, O.; Lang, I.; and Avidan, S. 2019. Learning to sample. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2760–2769.

Huang, T.; Zhang, J.; Chen, J.; Liu, Y.; and Liu, Y. 2022. Resolution-free point cloud sampling network with data distillation. In *European Conference on Computer Vision*, 54–70. Springer.

Lang, I.; Manor, A.; and Avidan, S. 2020. Samplenet: Differentiable point cloud sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7578–7588.

Li, Y.; Bu, R.; Sun, M.; Wu, W.; Di, X.; and Chen, B. 2018. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31.

Liang, D.; Zhou, X.; Wang, X.; Zhu, X.; Xu, W.; Zou, Z.; Ye, X.; and Bai, X. 2024. PointMamba: A Simple State Space Model for Point Cloud Analysis. *arXiv preprint arXiv:2402.10739*.

Lin, Y.; Yan, Z.; Huang, H.; Du, D.; Liu, L.; Cui, S.; and Han, X. 2020. Fpconv: Learning local flattening for point convolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4293–4302.

Lin, Z.-H.; Huang, S.-Y.; and Wang, Y.-C. F. 2020. Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 1800–1809.

Ma, X.; Qin, C.; You, H.; Ran, H.; and Fu, Y. 2022. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In *International Conference on Learning Representations (ICLR)*.

Moenning, C.; and Dodgson, N. A. 2003. Fast marching farthest point sampling. Technical report, University of Cambridge, Computer Laboratory.

Nezhadarya, E.; Taghavi, E.; Razani, R.; Liu, B.; and Luo, J. 2020. Adaptive hierarchical down-sampling for point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12956–12964.

Pang, Y.; Wang, W.; Tay, F. E.; Liu, W.; Tian, Y.; and Yuan, L. 2022. Masked autoencoders for point cloud self-supervised learning. Tel Aviv, Israel.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 652–660.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30.

Qian, G.; Li, Y.; Peng, H.; Mai, J.; Hammoud, H.; Elhoseiny, M.; and Ghanem, B. 2022. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 35: 23192–23204.

Thomas, H.; Qi, C. R.; Deschaud, J.-E.; Marcotegui, B.; Goulette, F.; and Guibas, L. J. 2019. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, 6411–6420.

Uy, M. A.; Pham, Q.-H.; Hua, B.-S.; Nguyen, T.; and Yeung, S.-K. 2019. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, 1588–1597.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (ToG)*, 38(5): 1–12.

Wu, C.; Zheng, J.; Pfrommer, J.; and Beyerer, J. 2023. Attention-based point cloud edge sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5333–5343.

Wu, W.; Qi, Z.; and Fuxin, L. 2019. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9621–9630.

Wu, X.; Jiang, L.; Wang, P.-S.; Liu, Z.; Liu, X.; Qiao, Y.; Ouyang, W.; He, T.; and Zhao, H. 2024. Point transformer v3: Simpler, faster, stronger. volume 35. Seattle, USA.

Wu, X.; Lao, Y.; Jiang, L.; Liu, X.; and Zhao, H. 2022. Point transformer v2: Grouped vector attention and partition-based pooling. volume 35, 33330–33342. New Orleans, Louisiana, USA.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2014. 3D ShapeNets: A Deep Representation for Volumetric Shapes.

Xu, M.; Ding, R.; Zhao, H.; and Qi, X. 2021. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 3173–3182.

Yan, X.; Zheng, C.; Li, Z.; Wang, S.; and Cui, S. 2020. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5589–5598.

Yang, J.; Zhang, Q.; Ni, B.; Li, L.; Liu, J.; Zhou, M.; and Tian, Q. 2019. Modeling point clouds with self-attention and gumbel subset sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3323–3332.

Zha, Y.; Ji, H.; Li, J.; Li, R.; Dai, T.; Chen, B.; Wang, Z.; and Xia, S.-T. 2024. Towards compact 3d representations via point feature enhancement masked autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 6962–6970.

Zha, Y.; Wang, J.; Dai, T.; Chen, B.; Wang, Z.; and Xia, S.-T. 2023. Instance-aware dynamic prompt tuning for pre-trained point cloud models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 14161–14170.

Zhao, H.; Jiang, L.; Fu, C.-W.; and Jia, J. 2019. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5565–5573.

Zhao, H.; Jiang, L.; Jia, J.; Torr, P. H.; and Koltun, V. 2021. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, 16259–16268.