SAMBLE: Shape-Specific Point Cloud Sampling for an Optimal Trade-Off Between Local Detail and Global Uniformity

Chengzhi Wu[1] Yuxin Wan[1*] Hao Fu[1*] Julius Pfrommer[2]
Zeyun Zhong[1] Junwei Zheng[1†] Jiaming Zhang[1] Jürgen Beyerer[1,2]

[1]Karlsruhe Institute of Technology, Germany [2]Fraunhofer IOSB, Germany

{chengzhi.wu, zeyun.zhong, junwei.zheng, jiangming.zhang}@kit.edu,
{yuxin.wan, hao.fu}@student.kit.edu, {julius.pfrommer, juergen.beyerer}@iosb.fraunhofer.de

Abstract

*Driven by the increasing demand for accurate and efficient representation of 3D data in various domains, point cloud sampling has emerged as a pivotal research topic in 3D computer vision. Recently, learning-to-sample methods have garnered growing interest from the community, particularly for their ability to be jointly trained with downstream tasks. However, previous learning-based sampling methods either lead to unrecognizable sampling patterns by generating a new point cloud or biased sampled results by focusing excessively on sharp edge details. Moreover, they all overlook the natural variations in point distribution across different shapes, applying a similar sampling strategy to all point clouds. In this paper, we propose a **S**parse **A**ttention **M**ap and **B**in-based **Le**arning method (termed SAMBLE) to learn shape-specific sampling strategies for point cloud shapes. SAMBLE effectively achieves an improved balance between sampling edge points for local details and preserving uniformity in the global shape, resulting in superior performance across multiple common point cloud downstream tasks, even in scenarios with few-point sampling.*

1. Introduction

Point cloud sampling is a less explored research area within the realm of this data representation. Traditional random sampling (RS) and farthest point sampling (FPS) remain the most commonly employed methods when sampling is required for point cloud learning. With the advancement of neural networks, several methods have emerged for point cloud sampling in a downstream task-oriented learning framework, including S-Net [7], SampleNet [14], MOPS-Net [34], etc. However, these methods first generate a new, smaller-sized point cloud as a proxy rather than directly sampling points from the original input, rendering the techniques akin to black-box neural network models with limited interpretability. Consequently, discerning geometric patterns in their qualitative results becomes challenging, as their outcomes closely resemble those obtained through random sampling. More recently, APES [46] pioneers the direction of using neural networks to learn point-wise sampling scores, with which it subsequently samples points whose scores are higher. However, with its score computation design and the Top-M sampling strategy, APES excessively focuses on local details of edge points, resulting in a deficiency in preserving good global uniformity of the input shapes. Consequently, the interpolation operation becomes impractical during the upsampling process, and the sampling quality of few-point sampling is notably subpar (see Fig. 1). In this paper, we introduce a novel point cloud sampling method that addresses the limitations of prior approaches, aiming to achieve a refined balance between capturing local details and preserving global uniformity.

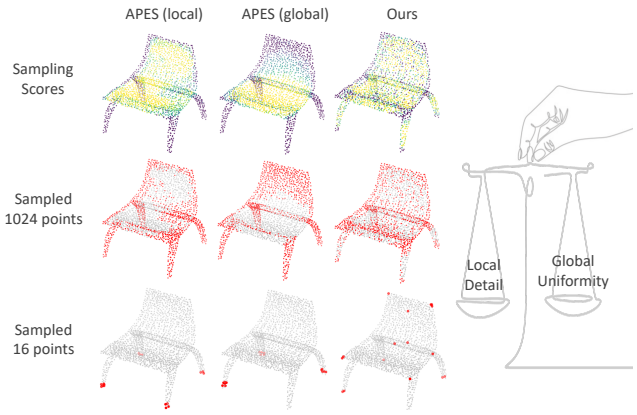

Figure 1. Our method achieves an improved trade-off between sampling local details and preserving global uniformity.

The concept originates from rethinking the mathematical characteristics of local details within point cloud shapes. Typically, these local details are represented by edge points

*Equal contribution. †Corresponding author.

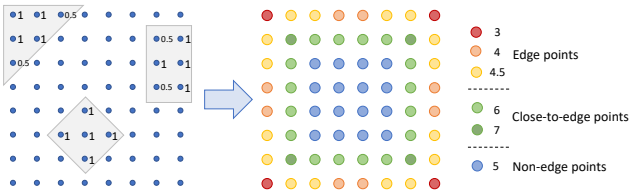

Figure 2. When selecting an equal number of neighbors for each point in the input point cloud, points at different positions are chosen as neighbors with varying frequencies (numbers on the right).

that define the shape's outline and sharpest features. Is there a point property that can easily distinguish between different categories, such as edge points and non-edge points? The answer is affirmative. In our investigation, we have uncovered an extremely fundamental yet crucial observation: if point $\mathbf{p}_i$ is one of the k-nearest neighbors of point $\mathbf{p}_j$, it does not necessarily imply that $\mathbf{p}_j$ is also among the k-nearest neighbors of $\mathbf{p}_i$. Consequently, it leads to the conclusion that the *frequency of each point being chosen as a neighbor* exhibits variation across a single point cloud.

We explore and demonstrate the importance of this point property with a simple example as illustrated in Fig. 2. Assume the input point cloud is a simple grid. When selecting 5 neighbors for each point, all three possible cases are given on the left (center point is self-contained as a neighbor). Note that in the triangular and rectangular cases, they each has a "quantum-entangled" twin point pair, in which two points share the possibility of being chosen as the neighbor. While an equal number of neighbors is selected for each point in the input point cloud, points at different positions are chosen as neighbors with varying frequencies, as listed on the right of Fig. 2. From it, we can observe that in addition to the edge point and non-edge point categories, there is also another noteworthy point category of close-to-edge points. Moreover, within each category, the points can be further grouped into more sub-categories. Overall, this point property effectively captures the local characteristics of a shape, especially for shape outline and sharp details. Building on it, we propose Sparse Attention Map (SAM) and introduce new methods for computing point-wise sampling scores to effectively balance the trade-off between local and global sampling. See more details in Sec. 3.2.

On the other hand, after the point-wise sampling scores are computed, previous methods employ a Top-M sampling strategy for all point cloud shapes, which exacerbates the issue of oversampling edge points. We argue that the naïve top-M sampling strategy may not be optimal across all point cloud shapes for downstream tasks. For example, sampling more non-edge points enhances global uniformity, while sampling more close-to-edge points "thickens" the edge, both of which can potentially improve the performance on downstream tasks [46]. To address this, we introduce a novel bin-based method to explore better sampling strategies shape-specifically by leveraging all point categories. This approach enables the sampling of points with smaller sampling scores, further optimizing the local-global trade-off. As a result, our method dynamically adjusts the sampling strategy for each shape, leading to a more tailored and interpretable sampling process for improved performance.

Our main contributions are summarized as follows:

- We propose a sparse attention map that directly integrates shape local and global information at the attention map level for point cloud sampling, introducing multiple methods for computing point-wise sampling scores.
- We present a novel approach for learning bin boundaries to partition points within individual shapes, enabling shape-specific sampling strategies by incorporating additional bin tokens during the attention computation.
- Our method successfully achieves an improved trade-off between capturing local details and preserving global uniformity for the sampling process, resulting in enhanced performance both qualitatively and quantitatively.

2. Related Work

Point Cloud Sampling. Point cloud sampling is a key process in 3D data handling for simplifying high-resolution dense point clouds. Over the past decades, non-learning-based methods [8, 10, 26] have predominantly been used for point cloud sampling. While Farthest Point Sampling (FPS) [8] is the most widely used one [18, 31, 33, 49, 60], Random Sampling (RS) has also been frequently adopted [10, 32, 61]. More recently, learning-based sampling methods have shown superior performance with task-oriented training. S-Net [7] represents a pioneering work of generating new point coordinates from global representations, while SampleNet [14] introduces a soft projection operation for better point approximation. Following S-Net, multiple learning-based methods have been proposed [21, 28, 40, 41]. MOPS-Net [34] learns a transformation matrix and multiplies it with the original point cloud to generate the sampled one. By employing the attention mechanism to learn point-wise sampling scores, APES [46] captures the edge points in the input point clouds with a strong focus.

Deep Learning on Point Clouds. In contrast to the voxelization-based methods [12, 17, 25] and multi-view-based methods [2, 3, 16, 37], point-based methods deal directly with point clouds. The pioneer studies of PointNet [30] and PointNet++ [31] tackle point clouds through point-wise Multi-Layer Perceptrons (MLPs) and max-pooling operations. Subsequently, other research shifts focus towards constructing more efficient building blocks for local feature extraction, such as convolution-based ones [1, 18, 20, 38, 49, 50, 62] and graph-based ones [4, 15, 22, 24, 36, 42, 54, 58]. More recently, while MLP-based methods like PointNeXt [33] and PointMetaBase [19] have rekin-

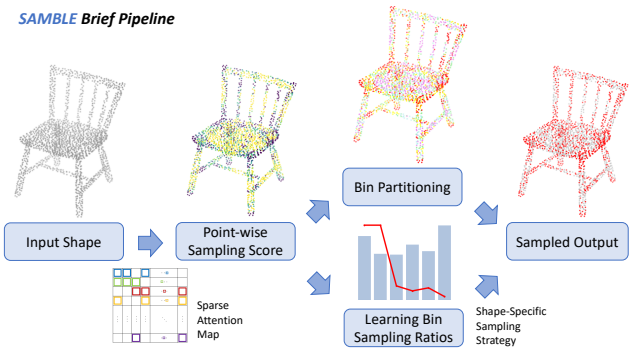

Figure 3. A brief pipeline of our proposed method SAMBLE. It learns shape-specific sampling strategies for point cloud shapes.

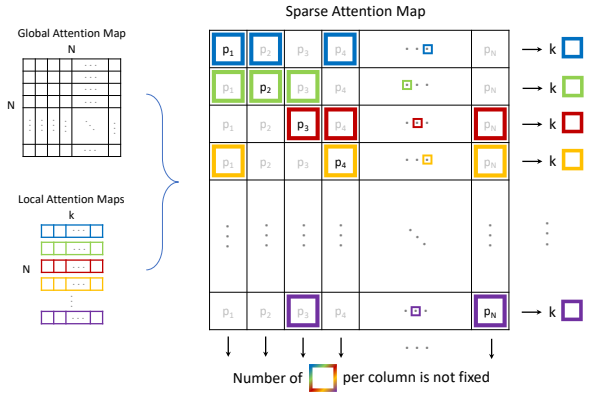

Figure 4. Sparse attention map. In each row, k cells are selected based on the KNN neighbor indexes for each point. The values of other non-selected cells are all set to 0. Note that the number of cells selected within each column is variable.

dled people's interest, the application of attention mechanisms to point cloud analysis has also garnered widespread attention [9, 11, 39, 43, 45, 48, 57, 60]. For example, PT [51, 52, 60] series improve the model performance by introducing subtraction-based attention blocks, and [48] performs a large ablation study over attention module designs for point cloud processing. In addition, approaches that apply Transformers for point cloud self-supervised learning have also been proposed and explored [23, 29, 47, 57, 59].

3. Methodology

A brief pipeline of SAMBLE is illustrated in Fig. 3. It consists of three key steps: constructing a sparse attention map, computing point-wise sampling scores, and learning shape-specific sampling strategies through bin partitioning.

3.1. Sparse Attention Map

Local and Global Attention Maps. Both local and global attention maps are widely used in point cloud analysis. A global attention map is derived from the application of classical self-attention to point features of all points, while a local attention map concentrates on a point-centered area wherein cross-attention is specifically applied to the central point and its neighbors.

Denote $\mathcal{S}_i$ as the set of k-nearest neighbors of point $\mathbf{p}_i$, the local attention map for $\mathbf{p}_i$ is defined as

$$\mathbf{m}_i^l = \text{softmax}\left(Q(\mathbf{p}_i)K(\mathbf{p}_{ij} - \mathbf{p}_i)_{j\in\mathcal{S}_i}^\top / \sqrt{d}\right), \quad (1)$$

where Q and K stand for the linear layers applied on the query and key input, and the square root of the feature dimension count $\sqrt{d}$ serves as a scaling factor [39].

For the global attention map which is equivalent to taking all points as neighbors for each point, it is defined as

$$\mathbf{M}^g = \text{softmax}\left(Q(\mathbf{p}_i)K(\mathbf{p}_j)_{i,j\in\mathcal{S}}^\top / \sqrt{d}\right), \quad (2)$$

where $\mathcal{S}$ denotes the set of all input points.

Sparse Attention Map. Instead of using local or global attention maps solely, we propose sparse attention map, which combines the knowledge from both local and global information, to compute point-wise sampling scores. The idea is illustrated in Fig. 4. After obtaining the global attention map with Eq. 2, k-NN is employed locally to find k neighbors for each point. In this case, k cells are being selected in each row. However, as discussed before, please notice that each point is chosen as a neighbor with varying frequencies. This means while for each row k cells are selected, for each column, the number of selected cells varies. The selected cells are then "carved" out to form the sparse attention map, with the values of other non-selected cells being set to 0. More vividly, consider the global attention map as a grid stone slab of size $N \times N$. For carve-based SAM, values of all cells are pre-computed and hidden in the slab grid cells, and only the selected cells are carved out.

3.2. Computing Point-wise Sampling Score

Indexing Mode. When sampling points, the points are indexed based on the computed point-wise sampling scores. We call the method of computing point-wise sampling scores from the full/sparse attention map as Indexing Mode. With the original full attention map, following APES, there are two possible indexing modes: (i) row standard deviation; and (ii) column sum. For a global attention map $\mathbf{M}^g$ of size $N \times N$, denote m_{ij} as the value of ith row and jth column in $\mathbf{M}^g$. These two indexing modes are formulated as modes (i) and (ii) in Tab. 1. To avoid possible confusion, we use notation $\mathbf{p}_o$ to denote a point only in this subsection.

With the proposed sparse attention map, there are many other possible indexing modes. As discussed in Sec. 1, to achieve an improved trade-off between sampling edge points and preserving global uniformity, the frequency of each point being chosen as a neighbor, i.e., *the number of selected cells in each column* is the key. The following indexing modes are designed and explored for comparison:

Indexing Mode	Attention Map	Formula	Remark
(i) Row standard deviation	Full	$a_{\mathbf{p}_o} = f_{\text{std}}(\{m_{oj} \mid j = 1, 2, \ldots, N\})$	f_{std}: Computes standard deviation for a set of values
(ii) Column sum	Full	$a_{\mathbf{p}_o} = \sum_{i=1}^{N} m_{io}$	
(iii) Row standard deviation	Sparse	$a_{\mathbf{p}_o} = f_{\text{std}}(\{m^s_{oj} \mid j \in S_o\})$	S_o: Set of indices of selected cells in oth row
(iv) Row sum	Sparse	$a_{\mathbf{p}_o} = \sum_{j=1}^{N} m^s_{oj}$	Non-selected cells are all of 0s
(v) Column sum	Sparse	$a_{\mathbf{p}_o} = \sum_{i=1}^{N} m^s_{io}$	
(vi) Column average	Sparse	$a_{\mathbf{p}_o} = \sum_{i=1}^{N} m^s_{io}/n_o$	n_o: Number of selected cells in oth column
(vii) Column square-divided	Sparse	$a_{\mathbf{p}_o} = \sum_{i=1}^{N} m^s_{io}/n_o^2$	n_o: Number of selected cells in oth column

Table 1. Proposed different indexing modes for computing point-wise sampling scores.

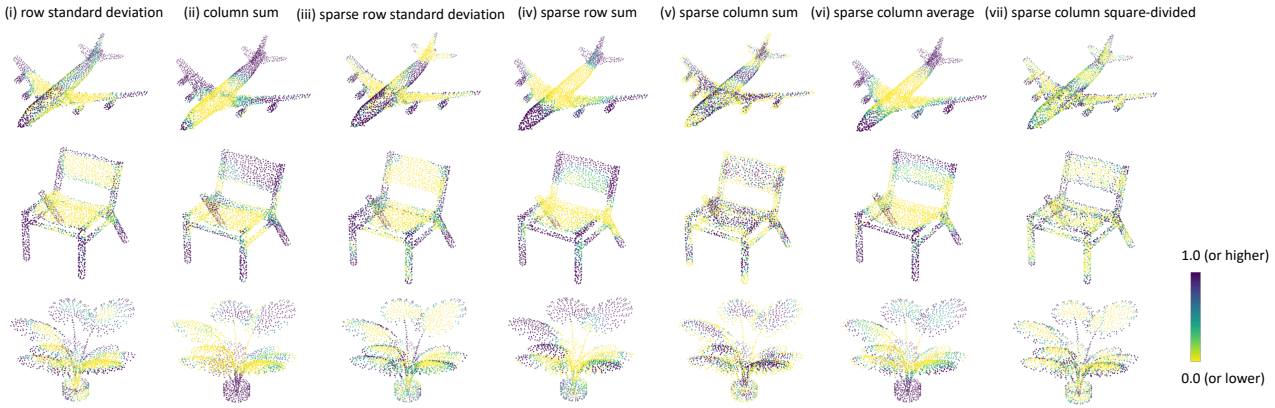

Figure 5. Point sampling score heatmaps under different indexing modes. Scores are normalized to $\mathcal{N}(0.5, 1)$ for better visualization.

(iii) sparse row standard deviation; (iv) sparse row sum; (v) sparse column sum; (vi) sparse column average; and (vii) sparse column square-divided. Again, for a sparse attention map $\mathbf{M}^s$ of size $N \times N$, denote m^s_{ij} as the value of ith row and jth column in $\mathbf{M}^s$. For point $\mathbf{p}_o$, we denote the set of indexes of the selected k cells (indexes of k-nearest neighbors) in oth row as S_o, and denote the number of selected cells in oth column as n_o. Details and respective formulas of the proposed indexing modes are listed in Tab. 1.

Heatmap. To analyze the behavior of each indexing mode, we train a separate model for each mode, ensuring that all other settings remain consistent. The sampling score distributions are visualized as heatmaps in Fig. 5, enhancing the interpretability of our method. From these heatmaps, we can see that both row-standard-deviation-based modes (modes i and iii) concentrate heavily on edge points. However, because they consistently prioritize thin or detailed regions, some areas may be overlooked. In contrast, modes ii and iv show less emphasis on edge points and instead distribute focus across a broader range of points, with a tendency toward other non-edge regions in a biased manner.

More interestingly, the comparison of modes v, vi, and vii, which utilize column-wise information from SAM, reveals distinct sampling preferences and strategies across different point categories. Mode v prioritizes non-edge points, mode vi emphasizes the global shape, and mode vii focuses slightly more on edge points. This is because edge points typically have a smaller number of n_o. Despite these differences and unique characteristics, all three modes capture the overall shape more uniformly compared to the former four. In our case, we aim to sample edge points without over-emphasizing them. For instance, when sampling detailed areas like chair legs, we want to capture some edge points without selecting them all, while also ensuring that non-edge points are sampled to preserve better global uniformity. Given this balance, we chose mode vii as the primary indexing mode for most of the experiments in the following sections. The detailed ablation study over different indexing modes is presented in Sec. 4.4.

3.3. Sampling with Bins

After point-wise sampling scores are computed with SAM, points are sampled according to certain rules. The simplest approach is top-M sampling, where points with the highest scores are sampled. In our case, as we aim to enhance the local-global trade-off and leverage all point categories during the sampling process, we suggest employing a bin-based sampling strategy to allow for the possible sampling of certain close-to-edge points or even non-edge points.

Bin Partitioning. The process begins with processing the distribution of normalized point-wise sampling scores $a_{\mathbf{p}_i}$ across the shapes within the current batch. Let n_b repre-

sent the number of bins used for partitioning, from which $n_b - 1$ bin boundary values need to be derived from the distribution. During each training step, a vector $\boldsymbol{\nu}_c = (\nu_1, \nu_2, \cdots, \nu_{n_b-1})$ is computed based on the point score distribution, ensuring an equitable division of points across all shapes in the current batch. Note that while $\boldsymbol{\nu}_c$ facilitates an even division at the batch level, the points within each individual shape are not necessarily evenly partitioned according to the batch-based bin boundary values.

During the training, for the first iteration, we directly use the boundary values derived from the first batch of data as the dynamic boundary values. Subsequently, since the second iteration, boundaries are updated adaptively in a momentum-based manner:

$$\boldsymbol{\nu}_t = \gamma \boldsymbol{\nu}_{t-1} + (1-\gamma)\boldsymbol{\nu}_c\,, \tag{3}$$

where $\boldsymbol{\nu}_{t-1}$ stands for the bin partitioning boundaries used in the last iteration, and $\boldsymbol{\nu}_t$ is the updated dynamic boundaries used for the current iteration. $\gamma \in (0, 1)$ is the momentum update factor. With updated boundary values $\boldsymbol{\nu}_t$, points in each shape are divided into n_b subsets of $\{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_{n_b}\}$ based on their sampling scores.

The core idea presented here is that, instead of using pre-engineered bin boundary values, we employ adaptive learning for bin boundaries, and they are gradually learned from the entire training dataset. These values are intended to evenly partition the distribution of point-wise sampling scores across all shapes and points in the training dataset. Consequently, for each individual shape, the acquired boundary values can effectively partition its points into bins with a shape-specific strategy, capturing the unique characteristics of the shape while maintaining a degree of proximity to other shapes within the dataset.

Tokens for Learning Bin Weights. With points already being partitioned into bins for each shape, the next step is to learn a shape-specific sampling strategy, i.e., to learn shape-specific sampling weights for each bin. Inspired by ViT [6], VilT [13], and Mask3D [35] — which leverage additional tokens during the computation of attention maps to extract and convey information across the entire feature map or specific groups of points or pixels — we introduce additional tokens specifically for learning bin sampling weights. In our case, attention maps are computed shape-specific during the downsampling process, facilitating the learning of bin sampling weights also in a shape-specific manner.

Using the former proposed bin partitioning method, points in each shape are partitioned into n_b subsets of $\{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_{n_b}\}$. The sampling weight ω_j for bin $\mathcal{B}_j (j = 1, 2, \ldots, n_b)$ is established based on the distinctive features of each shape. Fig. 6 gives the network structure of our proposed downsampling layer and illustrates the idea of using additional tokens. n_b bin tokens are introduced during the attention computation, where each token corresponds to a

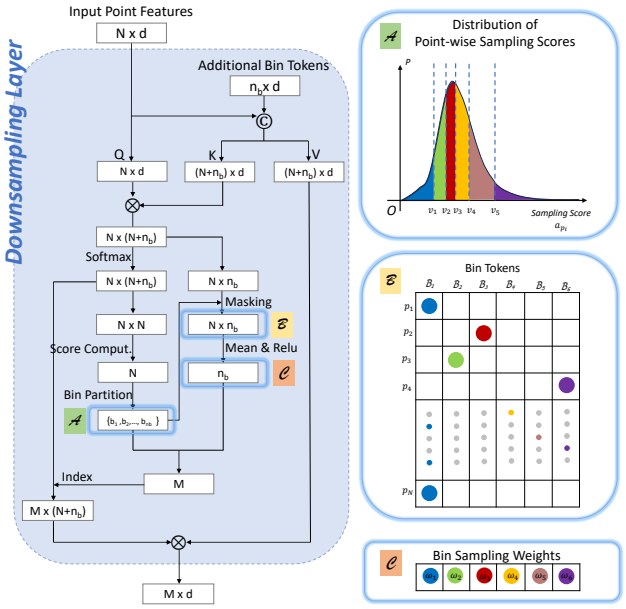

Figure 6. Network structure of our proposed downsampling layer. Block $\mathcal{A}$: Points in each shape are partitioned into n_b bins. Block $\mathcal{B}$: Masking the split-out point-to-token sub-attention map. Block $\mathcal{C}$: Learned bin sampling weights.

specific bin. As shown in Fig. 6, the bin tokens are initially concatenated with the input point-wise features for Key and $Value$. Subsequently, the combined features are subjected to a cross-attention mechanism with the original point-wise features as $Query$. The attention map is split into two parts of a point-to-point sub-attention map and a point-to-token sub-attention map. For the point-to-point attention map, the methods proposed in Sec. 3.1 and Sec. 3.2 are applied to it to obtain point-wise sampling scores. Note that in this case, the row-wise sum is not exactly equal to 1 but still very close to 1 since n_b is of a very small quantity compared to N. With computed point scores, dynamic boundary values $\mathbf{v}_t$ are obtained for bin partitioning. Using the information regarding the allocation of points to respective bins, a mask operation is performed on the point-to-token sub-attention map as illustrated in Block B of Fig. 6. The sampling weights ω_j are then subsequently acquired with

$$\omega_j = \text{ReLU}\left(\frac{1}{\beta_j}\sum_{\mathbf{p}_i \in \mathcal{B}_j} m_{\mathbf{p}_i, \mathcal{B}_j}\right)\,, \tag{4}$$

where β_j stands for the number of points in bin $\mathcal{B}_j$, and $m_{\mathbf{p}_i, \mathcal{B}_j}$ represents the element in the energy matrix corresponding to point $\mathbf{p}_i$ in row and $\mathcal{B}_j$ in column.

In-Bin Point Sampling. For each shape, by considering the number of points contained within bins $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_{n_b})$ alongside the determined bin sampling weights $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_{n_b})$, the specific numbers of points to be sampled from each bin $\boldsymbol{\kappa} = (\kappa_1, \kappa_2, \ldots, \kappa_{n_b})$

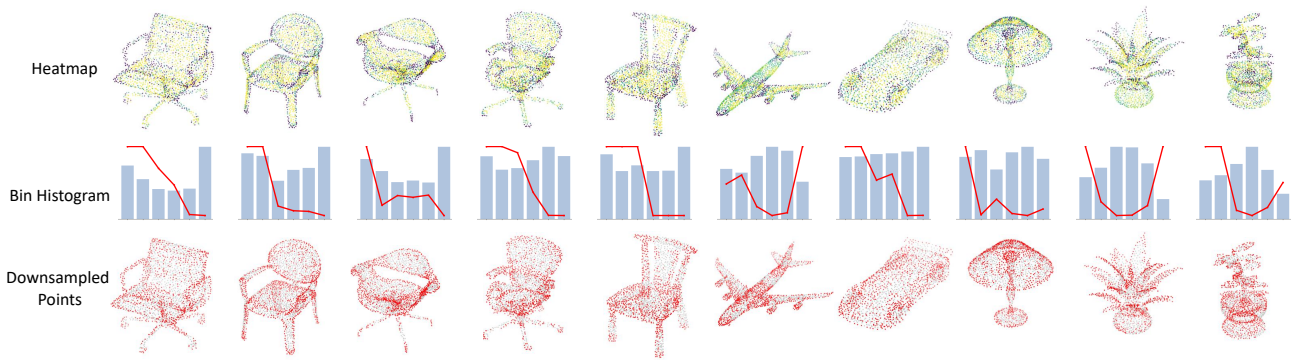

Figure 7. Qualitative results of our proposed SAMBLE. Apart from the sampled results, sampling score heatmaps and bin histograms along with bin sampling ratios are also given. All shapes are from the test set. Zoom in for optimal visual clarity.

need to be determined. Direct multiplication of $\boldsymbol{\beta}$ and $\boldsymbol{\omega}$ does not yield a sum that aligns with the total number of down-sampled points M required by the network structure. To address this discrepancy, a scaling method is applied to first scale bin sampling weights ω_j. Furthermore, to prevent κ_j from surpassing the available point number β_j in any bin, any excess points are proportionately redistributed to other bins that have not been fully sampled. The detailed algorithm is presented in the supplementary materials.

Finally, within bin $\mathcal{B}_j$, κ_j points are selected through random sampling with priors. The sampling probability $\rho_{\mathbf{p}_i}$ is obtained via softmax operation over the normalized point sampling score $a_{\mathbf{p}_i}$ with a temperature parameter τ:

$$\rho_{\mathbf{p}_i} = \frac{e^{a_{\mathbf{p}_i}/\tau}}{\sum_{\mathbf{p}_i \in \mathcal{B}_j} e^{a_{\mathbf{p}_i}/\tau}}. \tag{5}$$

4. Experiments

Like most related works, such as S-NET [7], SampleNet [14], and APES [46], SAMBLE is specifically designed for point cloud shapes. To ensure a fair comparison, we conduct experiments using the same base network architecture as APES [46] on standard point cloud shape datasets, including ModelNet40 and ShapeNet-Part. It is important to note that point cloud sampling is not a standalone task; its effectiveness must be validated through downstream tasks.

4.1. Classification

Experiment Setting. ModelNet40 classification benchmark [53] includes 12,311 CAD models across 40 categories. For a fair comparison, we use the official train-test split, with 9,843 for training and 2,468 for testing. Points are uniformly sampled from the mesh surface and normalized to the unit sphere. Only 3D coordinates are used as input, with random scaling, rotation, and shifting applied for data augmentation. We use $n_b = 6$ bins for point partitioning. The momentum update factor $\gamma = 0.99$ for updating bin boundary values. The temperature parameter $\tau = 0.1$.

Method	Cls.	Seg.	
	OA (%)	Cat. mIoU (%)	Ins. mIoU (%)
PointNet [30]	89.2	80.4	83.7
PointNet++ [31]	91.9	81.9	85.1
SpiderCNN [55]	92.4	82.4	85.3
DGCNN [42]	92.9	82.3	85.2
PointConv [49]	92.5	82.8	85.7
PT[1] [9]	92.8	-	85.9
PT[2] [60]	93.7	83.7	86.6
PCT [11]	93.2	-	86.4
PRA-Net [5]	93.7	83.7	86.3
CurveNet [27]	93.8	-	86.6
DeltaConv [44]	93.8	-	86.6
PointNeXt [33]	93.2	84.4	**86.7**
PointMetaBase [19]	-	84.3	**86.7**
APES (local) [46]	93.5	83.1	85.6
APES (global) [46]	93.8	83.7	85.8
SAMBLE	**94.2**	**84.5**	**86.7**

Table 2. Classification and segmentation results on the ModelNet40 and ShapeNet-Part benchmarks. In comparison with other SOTA methods that also only use raw point cloud data as input.

Qualitative and Quantitative Results. Qualitative results of SAMBLE are presented in Fig. 7, including sampling score heatmaps, learned bin partitioning strategy with bin sampling ratios, and final sampled results. From it, we can observe that SAMBLE effectively samples sufficient edge points to capture the shape structure. Moreover, it maintains better global uniformity by avoiding excessive focus on edge points, particularly in those thin or sharp regions like chair legs. Logged shape bin histograms confirm the learning of shape-specific sampling strategies. More visualization results are provided in the supplementary materials, highlighting an intriguing pattern where shapes of the same category exhibit similar histogram distributions and sampling strategies. Overall, SAMBLE successfully achieves an improved trade-off between sampling edge points and preserving shape global uniformity. Quantitative results are provided in Tab. 2. Our method performs better than previous approaches and achieves state-of-the-art performance.

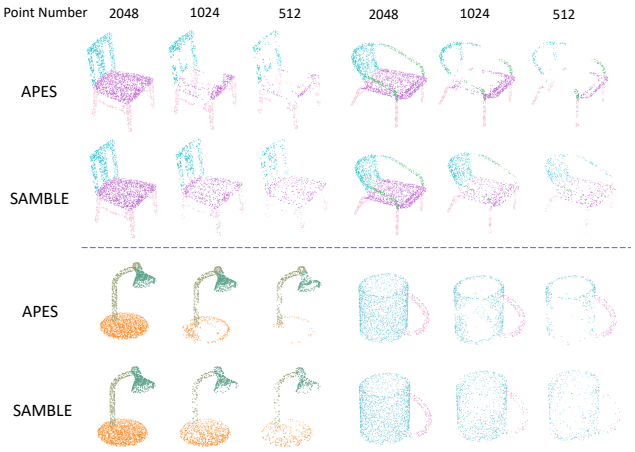

Figure 8. Segmentation results of our proposed SAMBLE in comparison with APES. All shapes are from the test set.

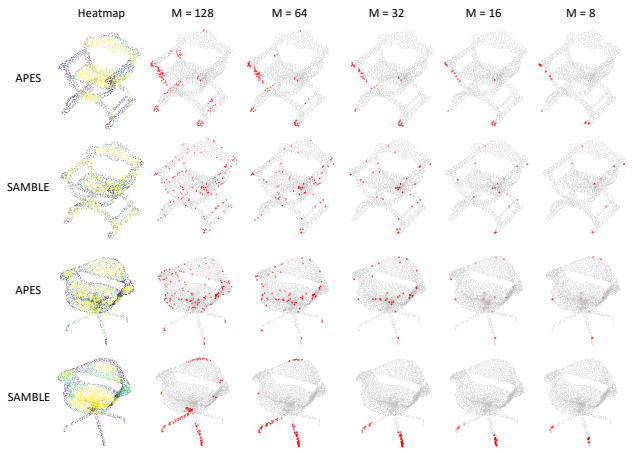

Figure 9. Sampled results of few-point sampling in comparison with APES. Zoom in for optimal clarity.

Method	PointNeXt			SAMBLE		
Point Number	2048	1024	512	2048	1024	512
Cat. mIoU (%)	84.40	83.79	82.77	84.51	84.84	**85.04**
Ins. mIoU (%)	86.70	86.18	85.18	86.67	86.93	**87.12**

Table 3. Additional segmentation performances evaluated on the intermediate downsampled sub-point clouds.

4.2. Segmentation

Experiment setting. The ShapeNet-Part dataset [56] is used for 3D object part segmentation. It includes 16,880 3D models across 16 categories, with 14,006 models for training and 2,874 for testing. Each category contains 2–6 parts, totaling 50 distinct parts. We use the sampled point sets produced in [30] for a fair comparison with prior work. For evaluation metrics, we report both category mIoU and instance mIoU. We use $n_b = 4$ bins for point partitioning. The momentum update factor $\gamma = 0.99$ for updating bin boundary values. The temperature parameter $\tau = 0.1$.

Qualitative and Quantitative Results. Qualitative results are presented in Fig. 8, where we observe that, compared to APES, which tends to focus excessively on edge points, SAMBLE achieves a significantly improved balance between sampling edge points and preserving shape global uniformity. For example, SAMBLE demonstrates a more balanced use of non-edge points, as seen in the chair seat and lamp base, reflecting a thoughtful sampling strategy that accounts for different point categories and provides a more comprehensive representation of the overall shape. The quantitative results in Tab. 2 further highlight that SAMBLE achieves state-of-the-art performance.

For the part segmentation benchmark, we further report the performance on the intermediate downsampled sub-point clouds in Tab. 3. Additionally, results from PointNeXt [33] are also presented, which is a prominent point cloud learning method that employs FPS for downsampling. It is evident that FPS-based methods exhibit poorer performance when evaluated on intermediate downsampled sub-point clouds. In contrast, a notable observation is that SAMBLE achieves superior performance on intermediate downsampled points. This indicates that the learned sampled points contribute more to the overall performance, while the upsampling layer cannot fully reconstruct the features of the discarded points. Although SAMBLE enables the interpolation-based upsampling and it outperforms the upsampling layer used in APES (see details in Sec. 4.4), there remains potential for further improvement by designing a more meticulously crafted upsampling layer. However, this is another topic and beyond the scope of this paper.

4.3. Few-Point Sampling

Experiment setting. We further compare our sampling method to previous approaches, including RS, FPS, and more recent learning-based methods such as S-Net, SampleNet, LightTN, APES, and others. The evaluation follows the same framework as in [7, 41, 46]. First, the point cloud is downsampled to a limited number of points, and the resulting subset is then fed into a task network for evaluation. For this comparison, we use the ModelNet40 classification task with the vanilla PointNet network. All sampling methods are evaluated across multiple sampling sizes of M.

Qualitative and Quantitative Results. Quantitative results are presented in Tab. 4. Note that APES [46] uses FPS to pre-process the input into $2M$ points while we do not. For a fair comparison, additional results of APES without the pre-processing step are also tested and reported. Nonetheless, even without pre-processing, SAMBLE achieves state-of-the-art results in the few-point sampling task as the number of sampled points decreases to extremely smaller ones.

M	Voxel	RS	FPS [8]	S-NET [7]	PST-NET [40]	SampleNet [14]	MOPS-Net [34]	DA-Net [21]	LighTN [41]	APES [46] (w/ pre-pro.)	APES [46] (w/o pre-pro.)	SAMBLE
512	73.82	87.52	88.34	87.80	87.94	88.16	86.67	89.01	89.91	**90.81**	89.81	90.58
256	73.50	77.09	83.64	82.38	83.15	84.27	86.63	86.24	88.21	**90.40**	86.78	90.18
128	68.15	56.44	70.34	77.53	80.11	80.75	86.06	85.67	86.26	89.77	84.87	**90.02**
64	58.31	31.69	46.42	70.45	76.06	79.86	85.25	85.55	86.51	89.57	79.23	**89.81**
32	20.02	16.35	26.58	60.70	63.92	77.31	84.28	85.11	86.18	88.56	75.63	**89.45**

Table 4. Comparison with other sampling methods. Evaluated on the ModelNet40 classification benchmark with multiple sampling sizes. For APES, we additionally report its performance when pre-processing is not performed for a fair comparison.

Indexing Mode		i	ii	iii	iv	v	vi	vii
Cls.	OA (%)	93.92	93.78	93.63	93.66	93.40	**94.11**	94.08
Seg.	Cat. mIoU (%)	83.98	83.85	83.62	83.51	83.47	84.12	**84.22**
	Ins. mIoU (%)	86.16	85.99	85.74	85.60	85.49	86.38	**86.46**

Table 5. Classification and segmentation performance with different indexing modes.

Number of Bins		1	2	4	6	8	10	12
Cls.	OA (%)	93.95	93.91	93.98	**94.18**	94.02	93.80	93.84
Seg.	Cat. mIoU (%)	84.22	84.14	**84.51**	84.40	84.19	83.98	84.36
	Ins. mIoU (%)	86.46	86.28	**86.67**	86.61	86.48	86.23	86.43

Table 6. Classification and segmentation performance with different number of bins.

Qualitative results are presented in Fig. 9. For few-point sampling, APES relies on FPS to pre-sample the input into $2M$ points due to its limitations . In contrast, our method preserves better global uniformity, allowing direct few-point sampling from the input while still achieving satisfactory sampled results, as demonstrated in Fig. 9. When sampling very few points, APES tends to concentrate on the sharpest regions, whereas our SAMBLE method preserves better global uniformity throughout the point cloud shape.

4.4. Ablation Study

In this subsection, our emphasis is directed toward the novel designs introduced within this paper, excluding common topics such as network width. More ablation studies and further design justifications are provided in the supplementary materials to enhance our method's interpretability.

Different Indexing Modes. Apart from the visualized heatmaps given in Fig. 5, we also report their respective experimental results in Tab. 5. The tests are performed using top-M as the sampling strategy. From it, we can observe that indexing modes vi and vii achieve best performances.

Number of Bins. As a key parameter in SAMBLE, an ablation study is performed over the number of bins n_b. The results are presented in Tab. 6. Remarkably, increasing the number of bins does not yield improved performance. This phenomenon is likely attributable to the subdivision of shapes into an excessive number of point categories, leading to the gradual diminishment of score disparities across the bins. In our case, $n_b = 6$ and 4 yield the best performance for the classification and segmentation tasks respectively, and we use it for the corresponding experiments.

Upsample	Interpolation			Cross-Attention
	$K_{up} = 3$	$K_{up} = 8$	$K_{up} = 16$	
APES (local)	82.89 / 85.40	82.95 / 85.44	82.96 / 85.42	83.11 / 85.58
APES (global)	83.16 / 85.53	83.19 / 85.59	83.17 / 85.55	83.67 / 85.81
SAMBLE	**84.51 / 86.67**	84.35 / 86.48	84.31 / 86.43	84.36 / 86.44

Table 7. Segmentation results with different upsampling layers on ShapeNet-Part. The number before "/" is the category mIoU, and the number after is the instance mIoU.

Upsampling layer. An important aspect to highlight is the upsampling layer. Most point cloud network models employ neighbor-based interpolation [31, 33, 60] for upsampling, as FPS is used during the downsampling process. However, APES introduces a cross-attention layer for upsampling to address its limitations of overemphasizing edge points, which renders traditional neighbor-based interpolation impractical. In contrast, our method achieves an improved balance between sampling edge points and preserving global uniformity, allowing the use of interpolation operations during upsampling. An ablation study for evaluating various upsampling layers and interpolation with different K_{up} values is conducted, and the results are presented in Table 7. The results show a performance drop for APES when interpolation is used in place of cross-attention, while SAMBLE demonstrates superior performance with it.

5. Conclusion

In this paper, a novel point cloud sampling method SAMBLE is proposed to learn shape-specific sampling strategies for point cloud shapes. Based on a sparse attention map that integrates both local and global information, multiple indexing modes are designed and explored. By partitioning the points in each shape into bins and learning respective sampling ratios for each bin, shape-specific sampling strategies are acquired for individual point cloud shapes. SAMBLE achieves an optimal trade-off between sampling local details and preserving global uniformity, resulting in improved performance on downstream tasks. For future directions, advancements in upsampling layers could further improve the model's performance. Additionally, adapting the proposed method for point cloud scenes is another promising area to explore.

References

[1] Pyunghwan Ahn, Juyoung Yang, Eojindl Yi, Chanho Lee, and Junmo Kim. Projection-based point convolution for efficient point cloud segmentation. *IEEE Access*, 10:15348–15358, 2022. 2

[2] Nicolas Audebert, Bertrand Le Saux, and Sébastien Lefèvre. Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. In *Asian Conference on Computer Vision*, pages 180–196. Springer, 2016. 2

[3] Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. Unstructured point cloud semantic labeling using deep segmentation networks. *3DOR@ Eurographics*, 3, 2017. 2

[4] Can Chen, Luca Zanotti Fragonara, and Antonios Tsourdos. GAPointNet: Graph attention based point neural network for exploiting local feature of point cloud. *Neurocomputing*, 438:122–132, 2021. 2

[5] Silin Cheng, Xiwu Chen, Xinwei He, Zhe Liu, and Xiang Bai. PRA-Net: Point relation-aware network for 3d point cloud analysis. *IEEE Transactions on Image Processing*, 30: 4436–4448, 2021. 6

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 5

[7] Oren Dovrat, Itai Lang, and Shai Avidan. Learning to sample. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2760–2769, 2019. 1, 2, 6, 7, 8

[8] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997. 2, 8

[9] Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer. Point transformer. *IEEE Access*, 9:134826–134840, 2021. 3, 6

[10] Fabian Groh, Patrick Wieschollek, and Hendrik PA Lensch. Flex-convolution: Million-scale point-cloud learning beyond grid-worlds. In *Asian Conference on Computer Vision (ACCV)*, pages 105–122. Springer, 2018. 2

[11] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. PCT: Point cloud transformer. *Computational Visual Media*, 7:187–199, 2021. 3, 6

[12] Mingyang Jiang, Yiran Wu, Tianqi Zhao, Zelin Zhao, and Cewu Lu. PointSift: A sift-like network module for 3d point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*, 2018. 2

[13] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*, pages 5583–5594. PMLR, 2021. 5

[14] Itai Lang, Asaf Manor, and Shai Avidan. SampleNet: Differentiable point cloud sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7578–7588, 2020. 1, 2, 6, 8

[15] Itai Lang, Dvir Ginzburg, Shai Avidan, and Dan Raviv. DPC: Unsupervised deep point correspondence via cross and self construction. In *2021 International Conference on 3D Vision (3DV)*, pages 1442–1451. IEEE, 2021. 2

[16] Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Deep projective 3d semantic segmentation. In *International Conference on Computer Analysis of Images and Patterns*, pages 95–107. Springer, 2017. 2

[17] Truc Le and Ye Duan. PointGrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9204–9214, 2018. 2

[18] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on x-transformed points. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018. 2

[19] Haojia Lin, Xiawu Zheng, Lijiang Li, Fei Chao, Shanshan Wang, Yan Wang, Yonghong Tian, and Rongrong Ji. Meta architecture for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17682–17691, 2023. 2, 6

[20] Yiqun Lin, Zizheng Yan, Haibin Huang, Dong Du, Ligang Liu, Shuguang Cui, and Xiaoguang Han. FPConv: Learning local flattening for point convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4293–4302, 2020. 2

[21] Yanan Lin, Yan Huang, Shihao Zhou, Mengxi Jiang, Tianlong Wang, and Yunqi Lei. DA-Net: Density-adaptive downsampling network for point cloud classification via end-to-end learning. In *2021 4th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, pages 13–18. IEEE, 2021. 2, 8

[22] Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang. 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 (CVPR)*, pages 1800–1809, 2020. 2

[23] Haotian Liu, Mu Cai, and Yong Jae Lee. Masked discrimination for self-supervised learning on point clouds. In *European Conference on Computer Vision (ECCV)*, pages 657–675. Springer, 2022. 3

[24] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8895–8904, 2019. 2

[25] Daniel Maturana and Sebastian Scherer. VoxNet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. 2

[26] Carsten Moenning and Neil A Dodgson. Fast marching farthest point sampling. Technical report, University of Cambridge, Computer Laboratory, 2003. 2

[27] AAM Muzahid, Wanggen Wan, Ferdous Sohel, Lianyao Wu, and Li Hou. CurveNet: Curvature-based multitask learning

deep networks for 3d object recognition. *IEEE/CAA Journal of Automatica Sinica*, 8(6):1177–1187, 2020. 6

[28] Ehsan Nezhadarya, Ehsan Taghavi, Ryan Razani, Bingbing Liu, and Jun Luo. Adaptive hierarchical down-sampling for point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12956–12964, 2020. 2

[29] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *European Conference on Computer Vision (ECCV)*, pages 604–621. Springer, 2022. 3

[30] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 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)*, pages 652–660, 2017. 2, 6, 7

[31] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017. 2, 6, 8

[32] Haozhe Qi, Chen Feng, Zhiguo Cao, Feng Zhao, and Yang Xiao. P2B: Point-to-box network for 3d object tracking in point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6329–6338, 2020. 2

[33] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. PointNext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:23192–23204, 2022. 2, 6, 7, 8

[34] Yu Qian, Junhui Hou, Yiming Zeng, Qijian Zhang, Sam Tak Wu Kwong, and Ying He. MOPS-Net: A matrix optimization-driven network fortask-oriented 3d point cloud downsampling. *ArXiv*, abs/2005.00383, 2020. 1, 2, 8

[35] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3d: Mask transformer for 3d semantic instance segmentation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8216–8223. IEEE, 2023. 5

[36] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3693–3702, 2017. 2

[37] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3887–3896, 2018. 2

[38] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6411–6420, 2019. 2

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017. 3

[40] Xu Wang, Yi Jin, Yigang Cen, Congyan Lang, and Yidong Li. PST-Net: Point cloud sampling via point-based transformer. In *11th International Conference on Image and Graphics (ICIG)*, pages 57–69. Springer, 2021. 2, 8

[41] Xu Wang, Yi Jin, Yigang Cen, Tao Wang, Bowen Tang, and Yidong Li. LighTN: Light-weight transformer network for performance-overhead tradeoff in point cloud downsampling. *IEEE Transactions on Multimedia*, 2023. 2, 7, 8

[42] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):1–12, 2019. 2, 6

[43] Cheng Wen, Baosheng Yu, and Dacheng Tao. Learnable skeleton-aware 3d point cloud sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17671–17681, 2023. 3

[44] Ruben Wiersma, Ahmad Nasikun, Elmar Eisemann, and Klaus Hildebrandt. DeltaConv: anisotropic operators for geometric deep learning on point clouds. *ACM Transactions on Graphics (TOG)*, 41(4):1–10, 2022. 6

[45] Chengzhi Wu, Xuelei Bi, Julius Pfrommer, Alexander Cebulla, Simon Mangold, and Jürgen Beyerer. Sim2real transfer learning for point cloud segmentation: An industrial application case on autonomous disassembly. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4531–4540, 2023. 3

[46] Chengzhi Wu, Junwei Zheng, Julius Pfrommer, and Jürgen Beyerer. Attention-based point cloud edge sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5333–5343, 2023. 1, 2, 6, 7, 8

[47] Chengzhi Wu, Qianliang Huang, Kun Jin, Julius Pfrommer, and Jürgen Beyerer. A cross branch fusion-based contrastive learning framework for point cloud self-supervised learning. In *2024 International Conference on 3D Vision (3DV)*, pages 528–538. IEEE, 2024. 3

[48] Chengzhi Wu, Kaige Wang, Zeyun Zhong, Hao Fu, Junwei Zheng, Jiaming Zhang, Julius Pfrommer, and Jürgen Beyerer. Rethinking attention module design for point cloud analysis. In *International Conference on Pattern Recognition (ICPR)*, 2024. 3

[49] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9621–9630, 2019. 2, 6

[50] Wenxuan Wu, Li Fuxin, and Qi Shan. PointConvFormer: Revenge of the point-based convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21802–21813, 2023. 2

[51] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:33330–33342, 2022. 3

[52] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang

Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4840–4851, 2024. 3

[53] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 6

[54] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. Grid-GCN for fast and scalable point cloud learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5661–5670, 2020. 2

[55] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. SpiderCNN: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018. 6

[56] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016. 7

[57] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-BERT: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19313–19322, 2022. 3

[58] Kuangen Zhang, Ming Hao, Jing Wang, Xinxing Chen, Yuquan Leng, Clarence W de Silva, and Chenglong Fu. Linked dynamic graph cnn: Learning through point cloud by linking hierarchical features. In *2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pages 7–12. IEEE, 2021. 2

[59] Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-M2AE: multi-scale masked autoencoders for hierarchical point cloud pre-training. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:27061–27074, 2022. 3

[60] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268, 2021. 2, 3, 6, 8

[61] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4490–4499, 2018. 2

[62] Wei Zhu, Yue Ying, Jin Zhang, Xiuli Wang, and Yayu Zheng. Point cloud registration network based on convolution fusion and attention mechanism. *Neural Processing Letters*, pages 1–21, 2023. 2

SAMBLE: Shape-Specific Point Cloud Sampling for an Optimal Trade-Off Between Local Detail and Global Uniformity

Supplementary Material

6. Carve-based SAM and Insert-based SAM

Based on different information basis, we propose two different sparse attention maps (SAM), carve-based SAM and insert-based SAM. In the main paper, we used global information as the basis and introduced carve-based SAM. Using local information as the basis, insert-based SAM is introduced as follows.

Insert-based Sparse Attention Map. Carve-based sparse attention map starts from the global information, and then merges the local information. It can also be done in a reverse way: starting with the local information first, then considering it in a global situation. To be more specific, local-based attention maps are first computed with the equation presented in the main paper, subsequently, the values in the local attention map of each point are inserted in the corresponding cells of each row in an empty (initialized as all 0s) global $N \times N$ attention map based on the k-nearest neighbor indexes. We term it Insert-based Sparse Attention Map. Again, for each row, k cells are inserted; and for each column, the number of inserted cells is not fixed.

Relation between Carve and Insert-based SAM. More vividly, consider the global attention map as a grid stone slab of size $N \times N$. For carve-based SAM, values of all cells are pre-computed and hidden in the slab grid cells, and only the selected cells are carved out; for insert-based SAM, only values of certain cells are pre-computed in the mosaic tile strings (the local-based attention maps), and they are then inserted into the slab according to the corresponding KNN indexes, like inserting mosaic tiles into an empty grid slate. The final outputs from carve-based SAM and Insert-based SAM are quite similar since they have the same places of non-zero cells. For both methods, the number of selected cells in each row is always k, while the number of selected cells in each column is variable. Their main difference is that the row-wise sum in insert-based SAM is always 1, while in carve-based SAM is not.

Different Indexing Modes with Insert-Based SAM. Based on carve-based SAM, all seven indexing modes proposed are compatible. However, when insert-based SAM is used, since the sparse row-wise sum is always 1, only indexing modes iii, v, vi, vii are compatible.

To investigate how each indexing mode works, we train a separate model for each indexing mode with all other settings consistent. In the paper, we have visualized the sampling score distributions as heatmaps for carve-based SAM to improve the method's interpretability. As supplementary materials, we also present such heatmap results with insert-based SAM as in Fig. 10. Note that only four indexing modes are applicable for insert-based SAM. The visualized results are quite similar to those of carve-based SAM except for indexing mode vi, with which insert-based SAM shows smaller differences between point sampling scores. On the other hand, indexing mode vii still achieves a better trade-off between sampling edge points and preserving global uniformity. However, the performance of insert-based SAM on downstream tasks is mostly not on par with carve-based SAM as presented in Tab. 8, hence we use carve-based SAM as default for most experiments in our paper.

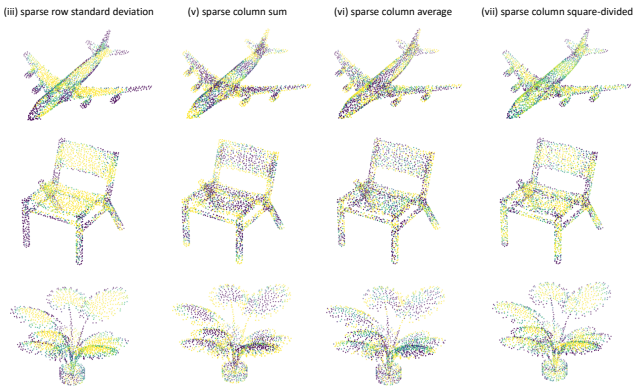

Figure 10. Heatmaps under different indexing modes with insert-based sparse attention map.

SAM	Indexing Mode	Cls. OA (%)	Seg.	
			Cat. mIoU (%)	Ins. mIoU (%)
Carve	i	93.92	83.98	86.16
	ii	93.78	83.85	85.99
	iii	93.63	83.62	85.74
	iv	93.66	83.51	85.60
	v	93.40	83.47	85.49
	vi	**94.11**	84.12	86.38
	vii	94.08	**84.22**	**86.46**
Insert	iii	93.67	83.71	85.86
	v	93.44	83.42	85.51
	vi	93.46	83.64	85.78
	vii	**93.83**	**84.09**	**86.15**

Table 8. Classification and segmentation performance of different indexing modes with different SAMs. Top-M sampling is adopted as the sampling strategy.

7. Determining Number of Sampled Points for Each Bin

For each shape, by considering the number of points contained within bins $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_{n_b})$ alongside the determined bin sampling weights $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_{n_b})$, the specific numbers of points to be sampled from each bin $\boldsymbol{\kappa} = (\kappa_1, \kappa_2, \ldots, \kappa_{n_b})$ are computed with the following algorithm.

Algorithm 1 Determining $\boldsymbol{\kappa}$ from $\boldsymbol{\beta}$ and $\boldsymbol{\omega}$

Require: number of total points to be selected: M, Sampling weights $\boldsymbol{\omega}$: $[\omega_1, \omega_2, \ldots, \omega_{n_b}]$, number of points in bins $\boldsymbol{\beta}$: $[\beta_1, \beta_2, ..., \beta_{n_b}]$

1: $\boldsymbol{\kappa} \leftarrow \mathbf{0}$
2: $\mathbf{x} \leftarrow \boldsymbol{\omega} \cdot \boldsymbol{\beta} + \epsilon$
3: $M_r \leftarrow M$
4: **while** $M_r > 0$ **do**
5: $\quad s \leftarrow \frac{M_r}{\sum x_j}$
6: $\quad$ **for** $j = 1$ to n_b **do**
7: $\quad\quad \kappa_j \leftarrow \text{round}(\kappa_j + s x_j)$
8: $\quad\quad$ **if** $\kappa_j \geq \beta_j$ **then**
9: $\quad\quad\quad \kappa_j \leftarrow \beta_j$
10: $\quad\quad\quad x_j \leftarrow 0$
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: $\quad M_r \leftarrow M - \sum \kappa_j$
14: **end while**
15: **return** $\boldsymbol{\kappa}$

In the above algorithm, s in line 5 is the scaling factor used to ensure a desired number of total sampled points. It is also evident that our sampling method is scalable to any desired sampling ratio. In line 2, ϵ is a minimal value (here we use 1×10^{-8}) to prevent the denominator part from being zero in a later step. Moreover, to prevent κ_j from surpassing the available number β_j in any bin, any excesse points are proportionately redistributed to other bins that have not been fully selected. The redistribution process is further illustrated in Fig. 11 for better comprehensibility.

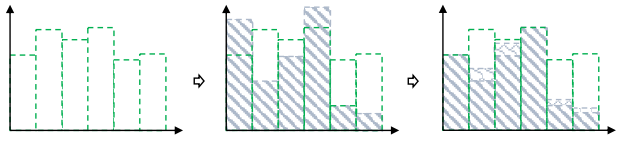

Figure 11. Illustration of redistributing excess points to other bins that have not been fully selected.

8. Relationship between Bin Sampling Weights and Bin Sampling Ratios

For the sake of brevity and improved visual clarity, in the paper, the axis labels of the histograms have been omitted. We further provide the full version of the histogram, in which the number of points and the sampling ratio in each bin are given. A demo is provided in Fig. 12. More detailed histogram results are provided in Sec. 16.

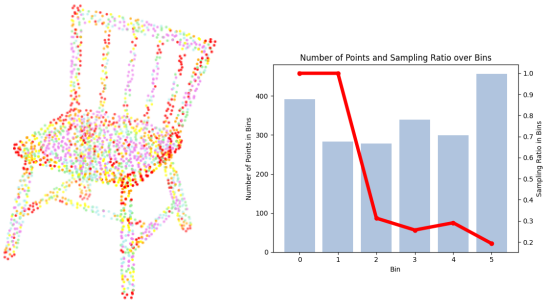

Figure 12. Left: bin partitioning, each color represents the points belonging to this bin. Right: the learned sampling strategy.

One thing worth noting is that the indicated sampling ratios $\mathbf{r}$ in the histogram are not simply re-scaled sampling weights $\boldsymbol{\omega}$. As in the algorithm we presented before, apart from the re-scaling operation, a redistribution operation is also applied to prevent κ_j surpassing the available point number β_j in one bin. Given the point number in each bin $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_{n_b})$ and the number of points to be sampled from each bin $\boldsymbol{\kappa} = (\kappa_1, \kappa_2, \ldots, \kappa_{n_b})$, the sampling ratios presented in the histogram is $\mathbf{r} = \boldsymbol{\kappa}/\boldsymbol{\beta}$ and $\mathbf{r} \in [0, 1]$.

Bin Index	0	1	2	3	4	5
Possibilities of All Points Being Sampled	53.69%	27.11%	8.02%	2.11%	0.85%	4.98%

Table 9. Possibilities of all points being sampled in bins, across all shapes from the test dataset.

The redistribution operation only happens when κ_j is about to surpass β_j, this means all points in jth bin have been selected and $r_j = 1$. We additionally count and document the likelihood of this occurrence for all bins across all shapes from the test dataset. The numbers are reported in Tab. 9, from which we can find that for around 54% of the shapes, all points in the first bin are selected and sampled. Note that the first bin corresponds to the points of higher sampling scores which are mostly edge points with indexing mode vii. This observation underscores the significance of edge points. On the other hand, there are still around 46% shapes that do not sample all edge points. It suggests that an excessive emphasis on edge points might have adverse effects on subsequent downstream tasks for these shapes, which also aligns with the conclusion drawn by APES.

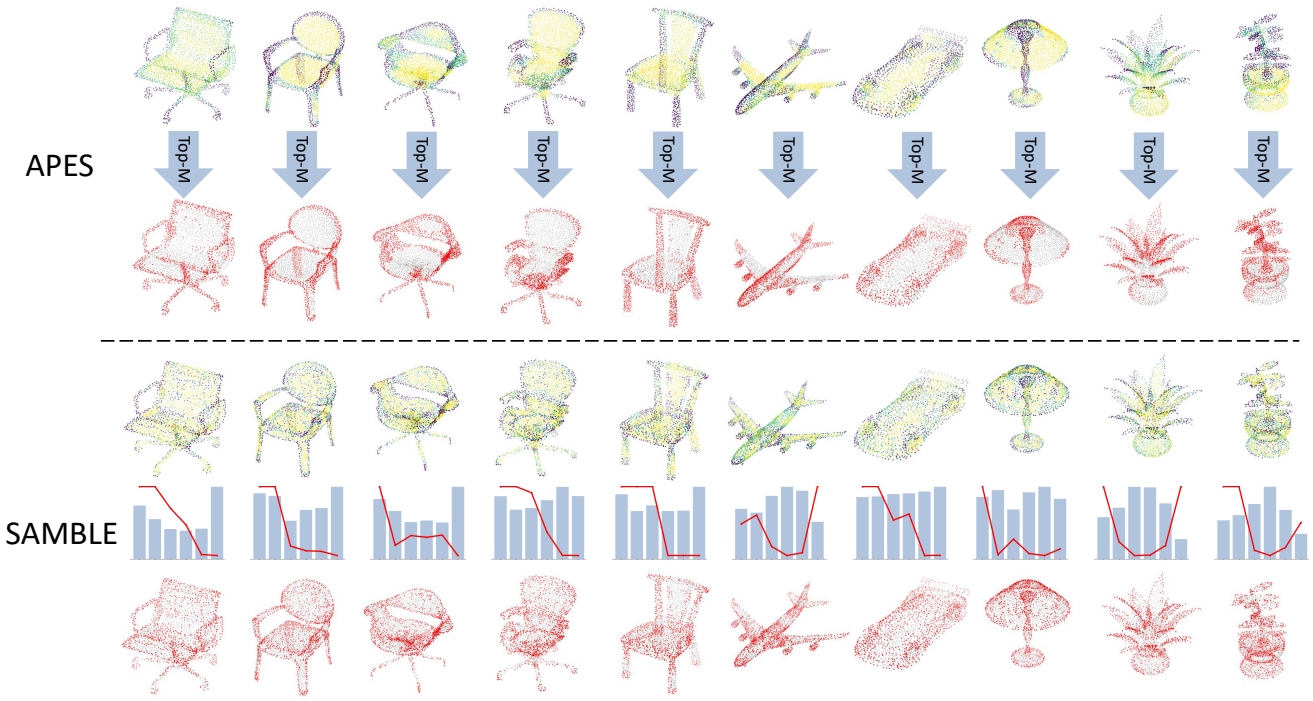

Figure 13. Qualitative results of our proposed SAMBLE, in comparison with APES. In addition to the sampled results, sampling score heatmaps and sampling strategies are also provided. All shapes are from the test set.

9. Network Architecture

For a fair comparison, the same basic network architectures from APES are used in our experiments, as illustrated in Fig. 14. The downsampling layers are replaced with our proposed ones, and the upsampling layers are replaced with the classical interpolation-based ones.

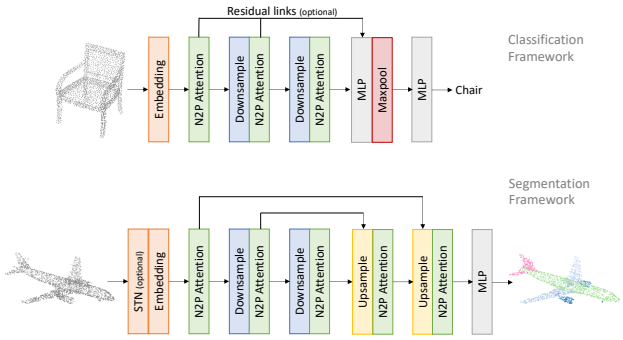

Figure 14. Network architectures for the classification task and the segmentation task.

10. More Training Details

Classification Tasks. AdamW is used as the optimizer. The learning rate starts from 1×10^{-4} and decays to 1×10^{-8} with a cosine annealing schedule. The weight decay hyperparameter for network weights is set as 1. Dropout with a probability of 0.5 is used in the last two fully connected layers. We use $n_b = 6$ bins for point partitioning. The momentum update factor $\gamma = 0.99$ for updating boundary values. The temperature parameter $\tau = 0.1$. The network is trained with a batch size of 8 for 200 epochs.

Segmentation Tasks. AdamW is used as the optimizer. The learning rate starts from 1×10^{-4} and decays to 1×10^{-8} with a cosine annealing schedule. The weight decay hyperparameter for network weights is 1×10^{-4}. We use $n_b = 4$ bins for point partitioning. The momentum update factor $\gamma = 0.99$ for updating boundary values. The temperature parameter $\tau = 0.1$. The network is trained with a batch size of 16 for 200 epochs.

11. Sampling Results in Comparison with APES

Additional qualitative results in comparison with APES are provided in Fig. 13 and Fig. 15. Both figures indicate that APES focuses excessively on edge points, while SAMBLE successfully achieves a much better trade-off between sampling edge points and preserving global uniformity, leading to better performance on downstream tasks.

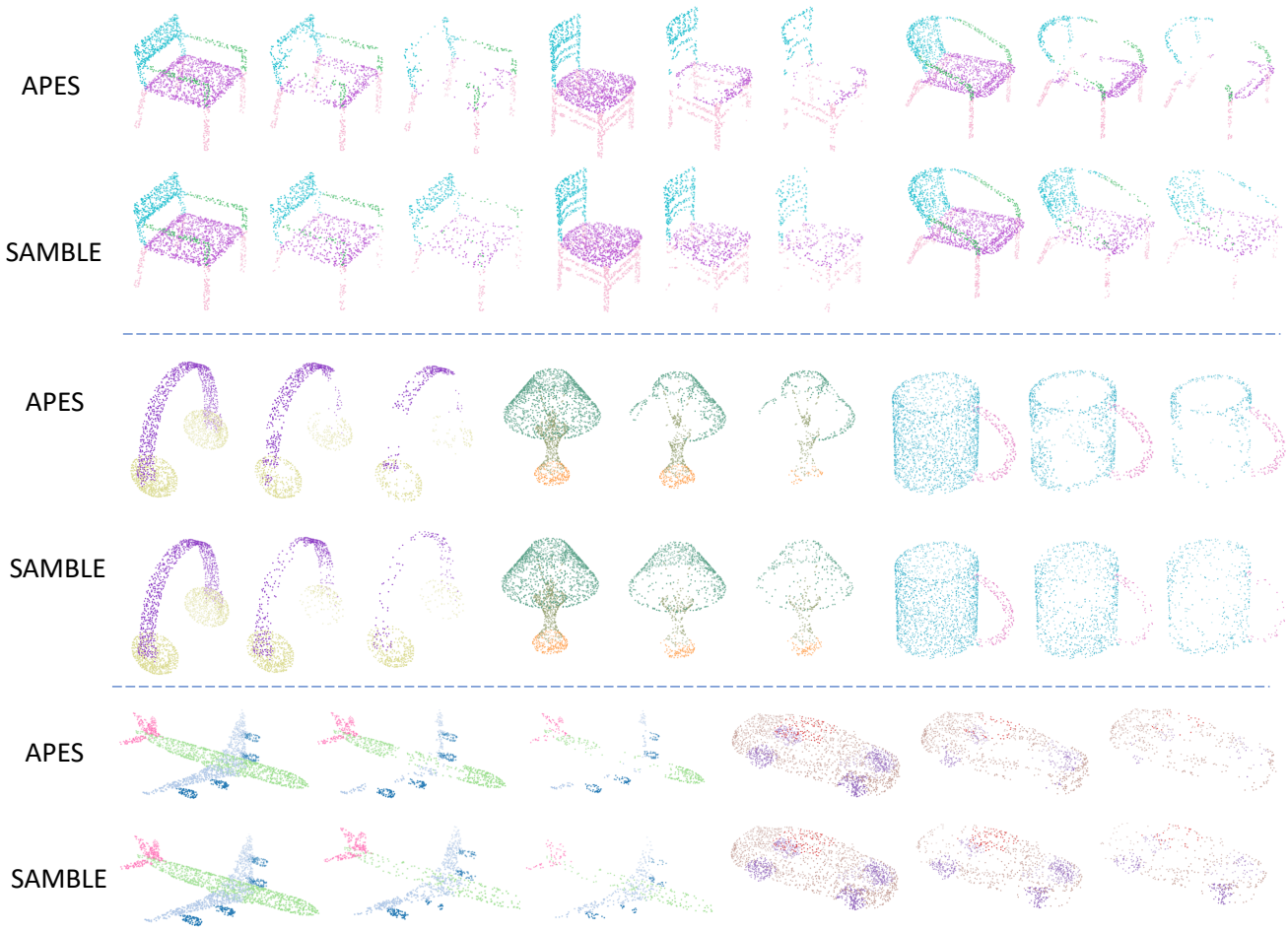

Figure 15. Segmentation results of our proposed SAMBLE, in comparison with APES. All shapes are from the test set.

12. Design Justifications of the Bin Token Idea - Devil Is in the Details.

Adding Bin Tokens to Q or K/V? A critical point in the idea of bin tokens lies in determining the specific branches to which the tokens should be concatenated. In order to match the tensor dimension for later computation in the attention mechanism, the tensor size of Key and Value should be the same. Hence if tokens are being added to the Key branch, they also have to be added to the Value branch. Overall, there are two possibilities of adding bin tokens to (i) the Query branch, or (ii) the Key and the Value branches.

It is crucial to emphasize that, due to the nature of the sampling operation where indexes are selected, gradients cannot be propagated back through the sampling operation during the backward propagation process. As a result, regardless of the selected structure, it is essential to establish an alternative pathway to convey the information contained within the bin tokens, which have a size of $n_b \times N$, to the downsampled features, which have a size of $M \times d$. This pathway should ensure the flow of relevant information despite the inability to directly backpropagate gradients through the sampling operation.

As illustrated in the left of Fig. 16, in the former case, an attention map of tensor size $(N + n_b) \times N$ is obtained. After M indexes of the points to be sampled are learned with SAMBLE, M rows in the attention map are extracted to form a new tensor for the next steps. However, note that the sub-tensor of $n_b \times N$ will never be delivered to the next steps since they do not correspond to points, hence no gradient will be backpropagated to the tokens during the training.

On the other hand, as illustrated in the right of Fig. 16, adding bin tokens to the Key and Value branches does not have this problem and successfully enables gradient backpropagation. One thing worth mentioning is that in this scenario, the row-wise sum is not exactly equal to 1 but still very close to 1 due to the significantly smaller magnitude of n_b relative to N. Therefore, this is unlikely to significantly

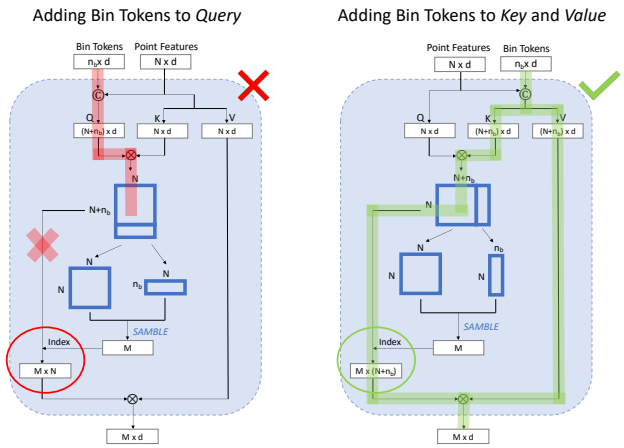

Figure 16. Adding bin tokens to Query leads to no gradient being backpropagated to the tokens, while adding bin tokens to Key and Value enables the gradient backpropagation.

impact the calculation of point-wise sampling scores. Concerning the design of adding bin tokens to all branches of Query, Key, and Value, it is equivalent to case ii since the sub-tensor of n_b rows in the attention map will never be sampled and propagated.

Order of Mean-pooling and ReLU Operations. Within our design, the ReLU operation is used to prevent the learned sampling weight from being negative. It can be performed either after Mean-pooling (as done in the main paper), or before Mean-pooling:

$$\omega_j = \frac{1}{\beta_j} \sum_{\mathbf{p}_i \in \mathcal{B}_j} \text{ReLU}(m_{\mathbf{p}_i, \mathcal{B}_j}) \,. \tag{6}$$

However, the inherent distribution of values within tensors often results in a non-negligible proportion being negative, especially those corresponding to points of lower importance. Directly setting too many values to zero would result in a significant loss of features, which is regrettable considering the potential information discarded. Therefore, instead of performing the ReLU operation before the mean-pooling operation, we do it the other way around, i.e., first mean-pooling, then, after this information fusion, ReLU is performed over the pooled results.

Fig. 17 gives the learned sampling strategies with the mean-pooling and ReLU operations applied in different orders. Although both orders yield shape-specific sampling strategies, the sampling ratios over bins learned with the order of ReLU first are mostly around 40% - 60%, leading to a worse sampling performance. On the other hand, the order of mean-pool first yields better sampling strategies as less potential information is discarded.

We additionally count and document the likelihood of ReLU being effective, which indicates the former pooled result is negative, for all bins across all test shapes. From the numbers reported in Tab. 10, we can see that the likelihood of the pooled results being negative is extremely small (less than 1%) for the first half of bins, while it goes higher for the latter bins yet the number is still relatively acceptable.

Bin Index	0	1	2	3	4	5
Possibilities of ReLU Being effective	0.45%	0.28%	0.57%	4.25%	11.63%	13.53%

Table 10. Possibilities of ReLU being effective in bins, across all test shapes.

Pre-softmax or Post-softmax Attention Map for Splitting The Point-to-Token Sub-Attention Map. When addressing the bin tokens, our initial approach involved splitting the point-to-token sub-attention map from the post-softmax attention map $\mathbf{M}_{\text{post}}$, which seemed intuitively appropriate. Furthermore, all elements within $\mathbf{M}_{\text{post}}$ are inherently positive, eliminating any concern for negative sampling weights and obviating the need for an additional ReLU operation. However, experimental findings revealed that this method proved ineffective, as it resulted in overly uniform sampling weights across different bins.

The underlying cause of this issue was identified after we explored the underlying mathematical principles and examined the values in the tensors during runtime. Tensors in a well-trained network tend to exhibit diminutive feature values as they propagate through layers. Denote m_{ij} as one element in the pre-softmax attention map $\mathbf{M}_{\text{pre}}$, given its minute magnitude, we apply the Taylor expansion formula to yield:

$$e^{m_{ij}} = 1 + m_{ij} + \frac{m_{ij}^2}{2} + \cdots \approx 1 + m_{ij} \,. \tag{7}$$

Therefore, the corresponding element m'_{ij} in the post-softmax attention map is

$$m'_{ij} = \frac{e^{m_{ij}}}{\sum_{j=1}^{N+n_b} e^{m_{ij}}} \approx \frac{1 + m_{ij}}{N + n_b + \sum_{j=1}^{N+n_b} m_{ij}} \,. \tag{8}$$

In our case, the values of the elements m_{ij} in $\mathbf{M}_{\text{pre}}$ are approximately within the magnitude of 10^{-3} to 10^{-5}. After a softmax operation, the resultant values m'_{ij} in $\mathbf{M}_{\text{post}}$ exhibit minimal variation, leading to closely similar sampling weights across bins in a later step.

Efforts were undertaken to address this issue before we turned to using $\mathbf{M}_{\text{pre}}$ for sampling weights acquisition. We attempted to use the logarithmic operation to restore the lost information:

$$\ln(m'_{ij}) = \ln(\frac{e^{m_{ij}}}{\sum_{j=1}^{N+n_b} e^{m_{ij}}}) = m_{ij} - \ln(\sum_{j=1}^{N+n_b} e^{m_{ij}}) \tag{9}$$

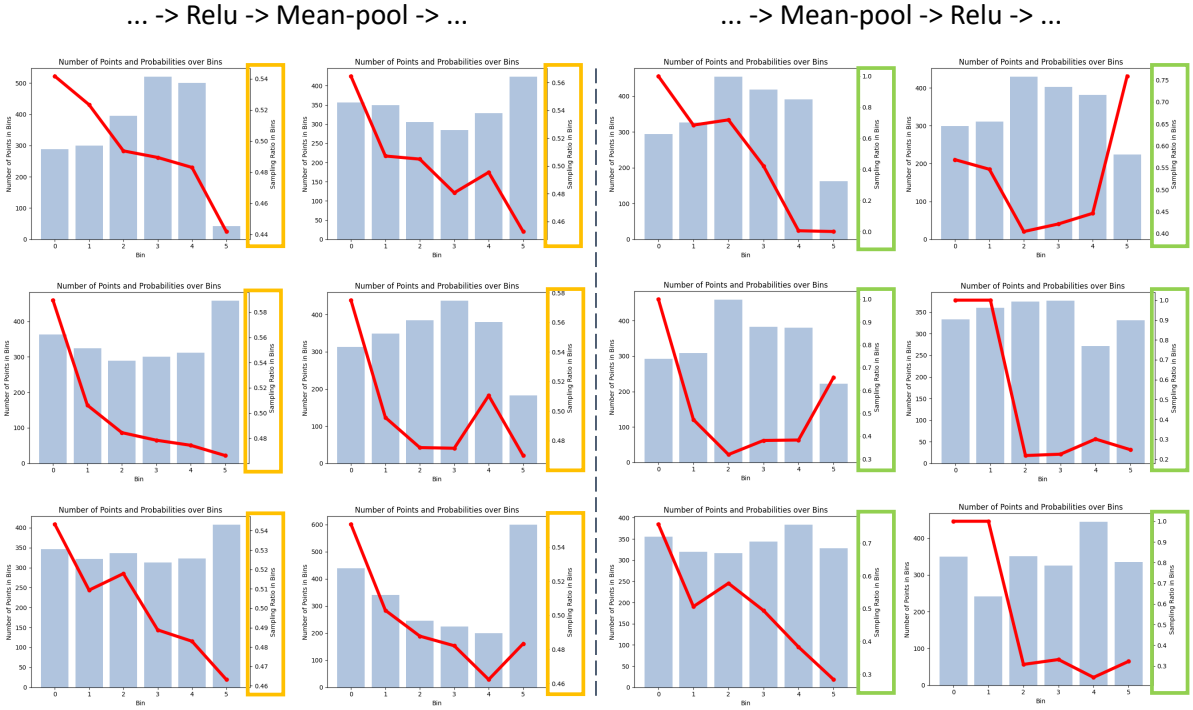

Figure 17. Learned sampling strategies with the mean-pooling and ReLU operations applied in different orders.

After the logarithmic operation, every value in the sub-attention map is negative. Therefore, a normalization operation is necessary. However, as shown in Fig. 18, the common normalization methods, such as z-score and centering, will result in too many negative elements (more than half), leading to too much information loss when passing through subsequent ReLU modules. Even if we successfully identify or meticulously design a superior normalization method that enables manual control over the proportion of negative elements to an applicable value, such manual intervention strays from the original intention of this thesis, which is to discover a learning-based mapping from sampling score to sampling probability.

Through the analysis, we observed that the term m_{ij} in Eq. (9) is exactly the elements in the pre-softmax attention map and is what we are interested in. Therefore, to avoid the potential loss of information that could arise from the softmax operation, we opted to directly use the results from $\mathrm{M}_{\mathrm{pre}}$ for bin sampling weights acquisition.

13. Additional Ablation Studies

Momentum Update Factor. The momentum update strategy is widely used within contrastive learning frameworks in self-supervised learning. In our case, we aim to derive the bin boundary values $\boldsymbol{\nu}$ from the entirety of shapes within the training dataset. These values aim to evenly partition the

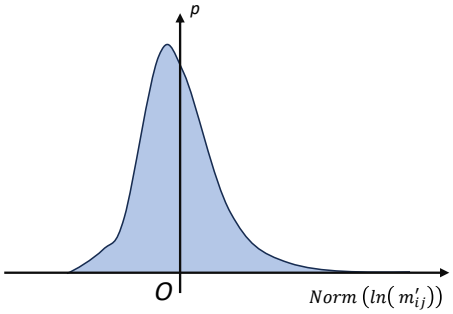

Figure 18. Illustrative figure of the distribution of the element values in the post-softmax attention map, after normalization.

distribution of point sampling scores across all shapes and points in the training data. Hence such an adaptive learning method is used.

An ablation study over the momentum update parameter γ is performed and the numerical results are reported in Tab. 11. From it, we can see that $\gamma = 0.99$ yields the best performance. This actually aligns with most current contrastive learning frameworks, where a majority use a value of $\gamma = 0.99$.

We additionally provide the bin partitioning results over the test dataset with the learned boundary values $\boldsymbol{\nu}$ in Fig. 19. It demonstrates that the boundary values adaptively

	γ	0.9	0.99	0.999	0.9999
Cls.	OA (%)	93.80	**94.18**	94.02	93.95

Table 11. Classification performance with different values of the momentum update factor γ.

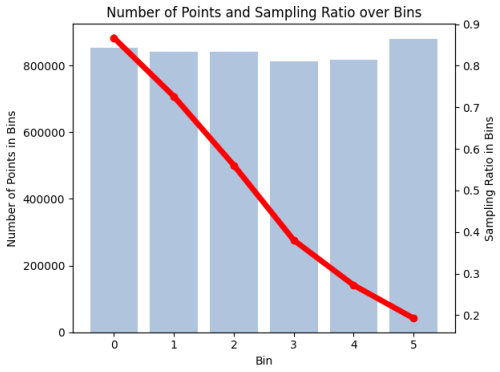

Figure 19. Partitioning the distribution of point sampling scores of all shapes and points in the test dataset into bins with the learned boundary values.

learned from the training dataset can also effectively partition the distribution of point sampling scores evenly across all shapes and points in the test dataset.

Temperature Parameter. The sampling strategy is determined with the point number in each bin $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_{n_b})$ and the number of points to be sampled from each bin $\boldsymbol{\kappa} = (\kappa_1, \kappa_2, \ldots, \kappa_{n_b})$. Within each bin, instead of applying the top-M sampling method simply, we suggest employing random sampling with priors. The idea is quite straightforward: process the point-wise sampling scores into point-wise sampling probabilities, and M non-repeated points are sampled randomly based on their sampling probabilities:

$$\rho_{\mathbf{p}_i} = \frac{e^{a_{\mathbf{p}_i}/\tau}}{\sum_{i=1}^{N} e^{a_{\mathbf{p}_i}/\tau}}, \tag{10}$$

where the temperature parameter τ controls the distribution of the sampling probabilities.

An ablation study over τ has been conducted. For a better illustration, the pre-softmax point sampling score heatmap and the post-softmax point sampling probability heatmap are visualized in Fig. 20. However, since the softmax operation is performed within each bin, it would be impossible to visualize the post-softmax sampling probabilities of different bins in the same figure if multi-bins are used. Hence in Fig. 20 only a single bin is used. From it, we can observe that the sampling probabilities of points go from having a large deviation to being uniformly distributed, just as we designed. Numerical results are reported in Tab. 12, where $\tau = 0.1$ achieves the best performance.

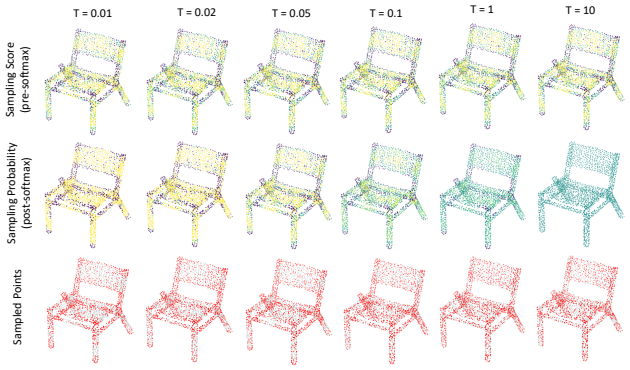

Figure 20. Different sampling results using different τ in the softmax with temperature during the sampling process. The indexing mode is the sparse column square-divided.

	τ	0.01	0.02	0.05	0.1	0.2	0.5	1	10
Cls.	OA (%)	93.84	93.96	94.06	**94.18**	93.89	93.84	93.74	93.70
Seg.	Cat. mIoU (%)	84.10	84.23	84.38	**84.51**	84.26	84.13	84.02	83.88
	Ins. mIoU (%)	86.44	86.48	86.60	**86.67**	86.51	86.42	86.29	86.23

Table 12. Classification and segmentation performance of the model with different τ values.

14. Sampling Policy Comparison

Three different sampling policies are illustrated in Fig. 21, including Top-M sampling, prior-based sampling, and bin-based sampling. The Top-M sampling policy is the simplest one and it samples the points with larger sampling scores directly. The prior-based sampling policy first converts the sampling scores into sampling probabilities and then samples randomly based on those probabilities. This method introduces the possibility of allowing the sampling of points with smaller sampling scores. The bin-based sampling policy further builds upon that. It first partitions the points into bins, and then learns bin sampling weights to determine the number of points to be sampled within each bin, In this way, it guarantees the sampling of some smaller-score points if the model thinks they are helpful for downstream tasks. In each bin, either top-M sampling or prior-based sampling can employed. In our case, we use the prior-based sampling. The bin-based sampling policy allows for more fine-grained control over the sampling process, tailoring it to the specific characteristics of each shape.

15. Model Complexity and Runtime Efficiency

To evaluate SAMBLE's efficiency, we assess its model complexity in comparison with APES and report the results in Tab. 13. Results from the traditional FPS and SAMBLE's variations are also reported. For a more direct and detailed comparison, we report both the number of parameters and FLOPS of a single downsampling layer. In order

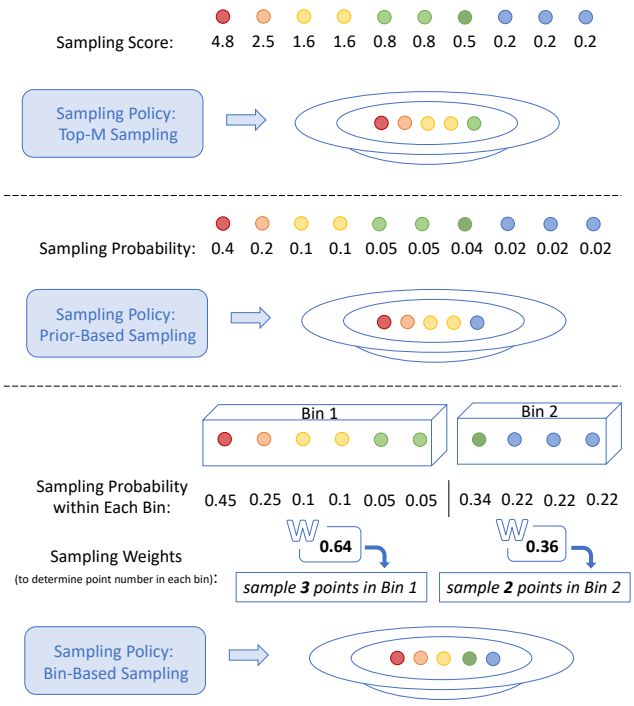

Figure 21. An illustration of different sampling policies. Note for bin-based sampling, either top-M sampling or prior-based sampling may be used within each bin.

to assess inference efficiency, experiments were carried out using a trained ModelNet40 classification model on a single NVIDIA GeForce RTX 3090. The tests were conducted with a batch size of 8, evaluating a total number of 2468 shapes from the test set.

As shown in Tab. 13, SAMBLE has a slightly larger number of model parameters compared to APES, primarily due to the incorporation of additional bin tokens. Notably, when $n_b = 1$, the number of parameters and FLOPs of SAMBLE are identical to that of global-based APES. This is quite reasonable as in this case, using additional bin tokens is unnecessary and the multi-bin-based sampling policy degrades into the simple prior-based sampling policy (see Fig. 21). On the other hand, SAMBLE's inference throughput is reduced due to the introduction of bin partitioning operations. Notably, the process of determining the number of points to be sampled within each bin involves a CPU-intensive loop computation (the redistribution process in Algorithm 1), which can lead to increased inference time.

Overall, using a sparse attention map instead of a full attention map improves performance on downstream tasks, though it slightly decreases inference throughput. Similarly, replacing top-M sampling with prior-based sampling for the sampling policy shows comparable results—enhancing performance at the cost of a minor reduction in efficiency. In

Method	Attention Map	Sampling Policy	Params.	FLOPs	Throughput (ins./sec.)	OA (%)
FPS + kNN	-	-	20.90k	2.71G	102	92.80
APES (local)	Local	Top-M	49.15k	1.09G	488	93.47
APES (global)	Global	Top-M	49.15k	0.05G	**520**	93.81
		Bin-based	49.92k	0.38G	128	<u>94.02</u>
SAMBLE ($n_b = 1$)	SAM	Top-M	49.15k	0.05G	<u>506</u>	93.92
		Prior-based	49.15k	0.05G	473	93.95
SAMBLE ($n_b = 6$)		Bin-based	49.92k	0.38G	125	**94.18**

Table 13. For model complexity, we report the number of parameters and FLOPs of one downsampling layer for a more detailed comparison. We also report the inference throughput (instances per second) and the classification performance.

contrast, when bin-based sampling is employed, there is a significant boost in model performance, but this comes with a notable decrease in efficiency. Despite this, SAMBLE consistently outperforms FPS in both performance and speed. Note that the FPS results presented here already use a GPU-accelerated version, whereas its standard implementation achieves a much lower throughput (around 12).

Considering the trade-off between model performance and runtime efficiency, the sampling method choice should be based on specific needs. For a balance between decent performance and high inference throughput, it is advisable to use SAM for point-wise score computation paired with a straightforward sampling policy such as Top-M or a prior-based approach (i.e., SAMBLE with n_b=1). This setup delivers good results without significantly impacting speed. On the other hand, if the focus is on achieving an optimal performance on downstream tasks, SAMBLE with the bin-based sampling policy is the ideal choice. However, this method may result in reduced inference throughput due to the complexity of the sampling strategy.

16. More Visualization Results

Learned Shape-Specific Sampling Strategies. We present additional extensive results in Fig. 22, Fig. 23, Fig. 24, and Fig. 25 with various categories. From them, we can observe that shape edge points are mostly partitioned into the first two bins. Furthermore, in addition to learning shape-wise sampling strategies for individual shapes, it is observed that analogous shapes within the same category exhibit similar histogram distributions and sampling strategies. Conversely, point clouds from different shape categories are sampled by distinct sampling strategies.

Few-Point Sampling. We further provide more visualization results of few-point sampling in Fig. 26 and Fig. 27. No pre-processing with FPS into $2M$ points was performed. From them, we can observe that when sampling very few points from the input directly, APES can only sample points from the sharpest regions in a concentrated manner, while our SAMBLE keeps better global uniformity.

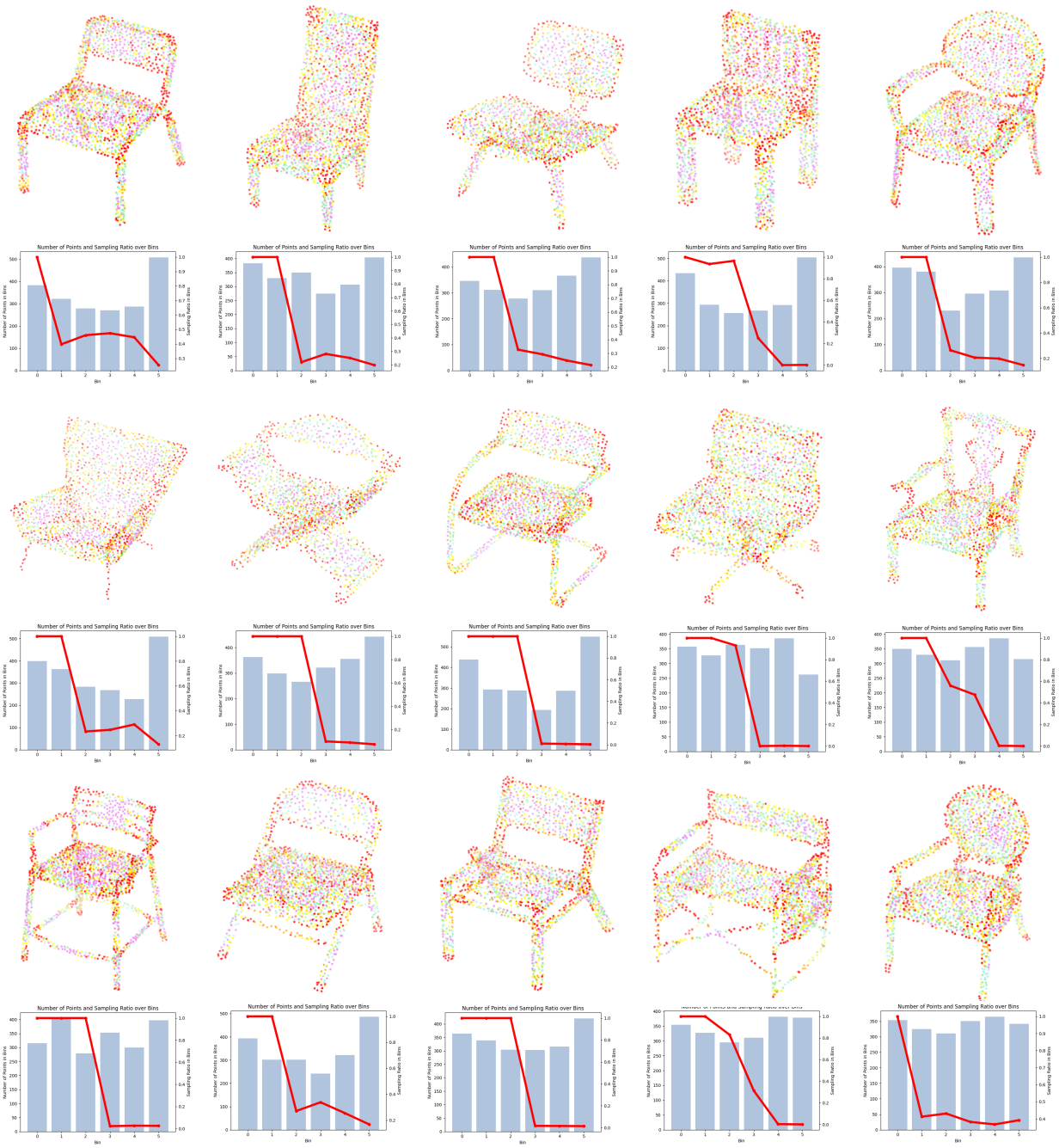

Figure 22. More visualization results of bin partitioning and learned shape-specific sampling strategies on the chair category. Zoom in for optimal visual clarity.

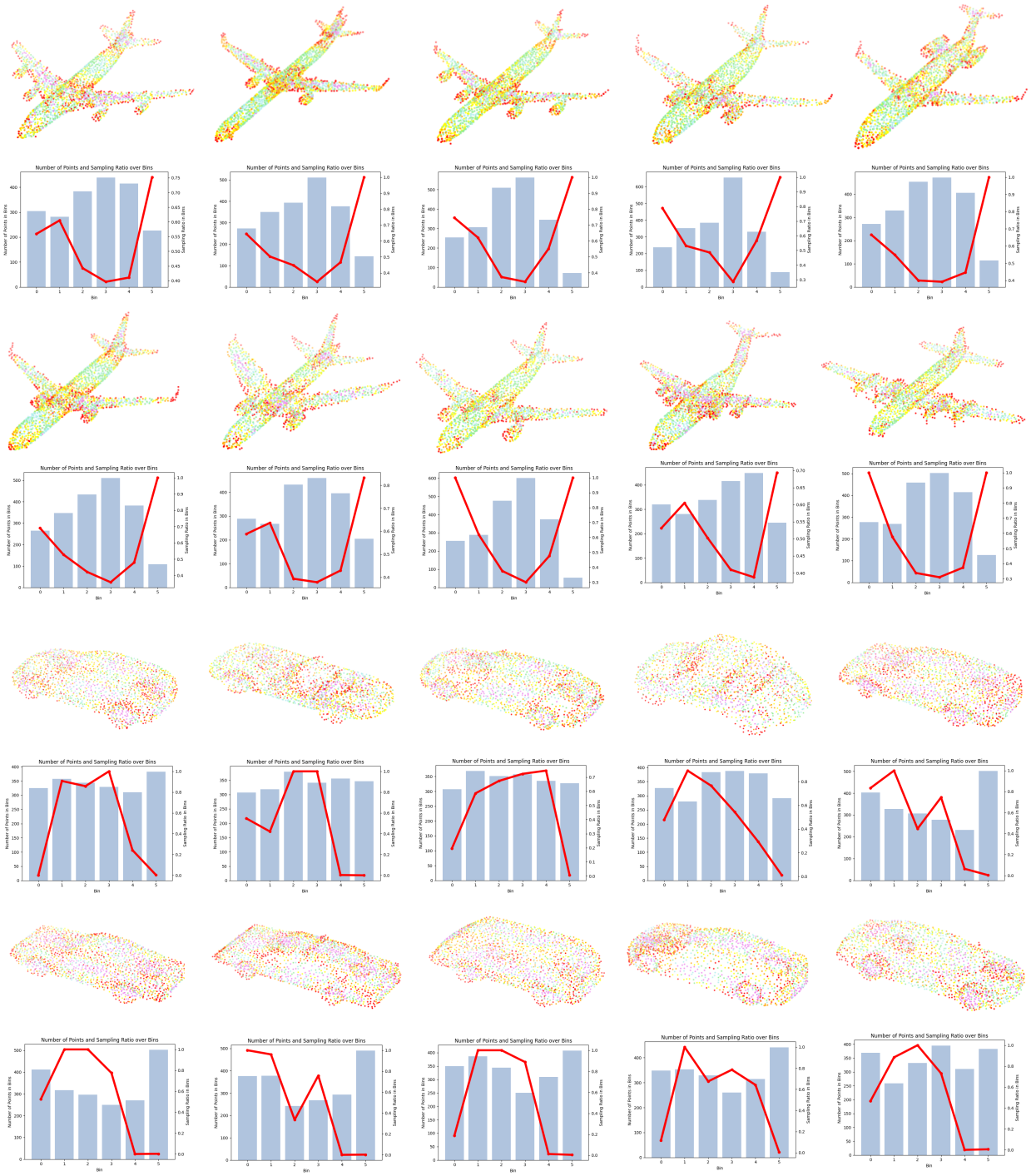

Figure 23. More visualization results of bin partitioning and learned shape-specific sampling strategies on the airplane and car categories. Zoom in for optimal visual clarity.

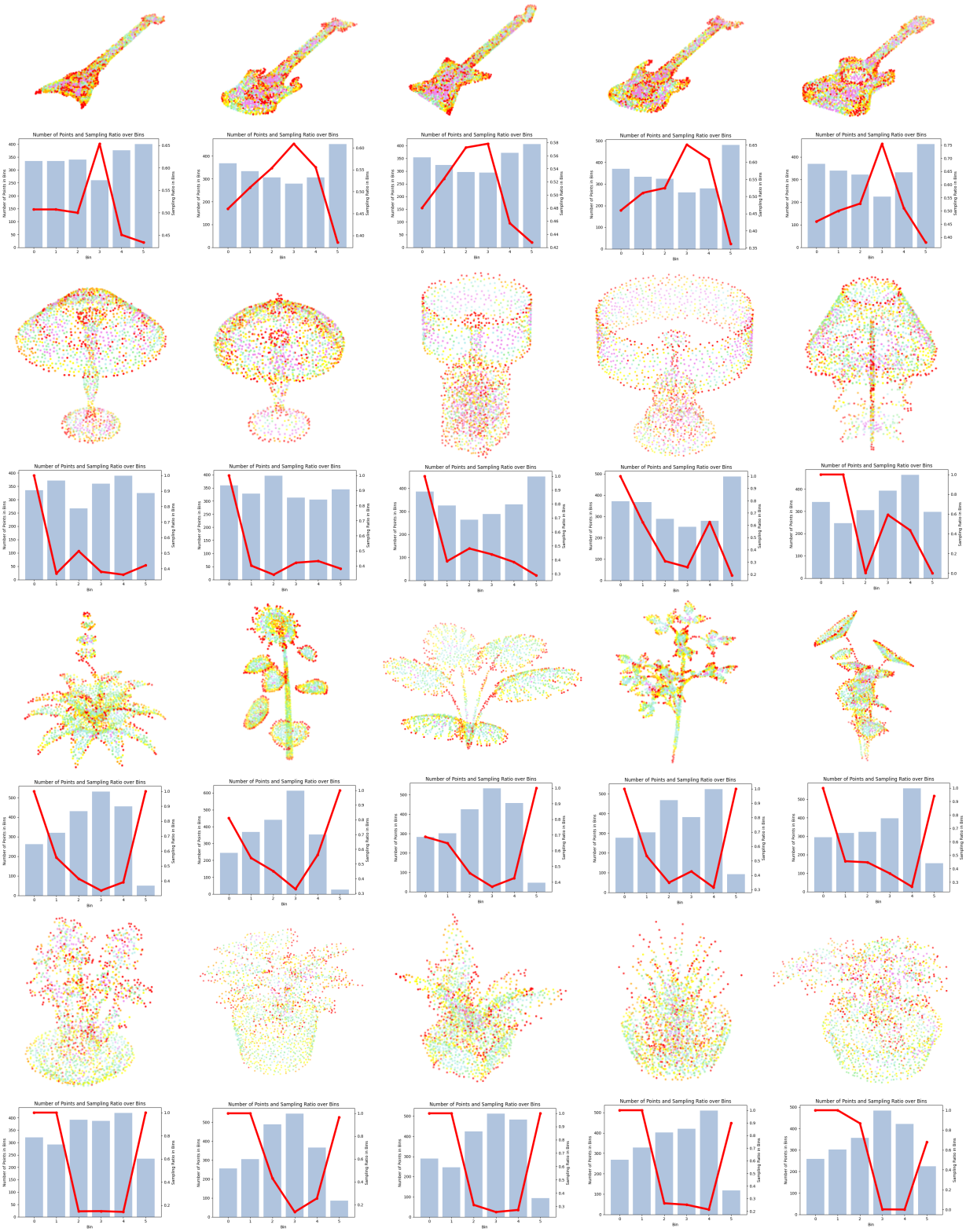

Figure 24. More visualization results of bin partitioning and learned shape-specific sampling strategies on the guitar, lamp, plant, and flower pot categories. Zoom in for optimal visual clarity.

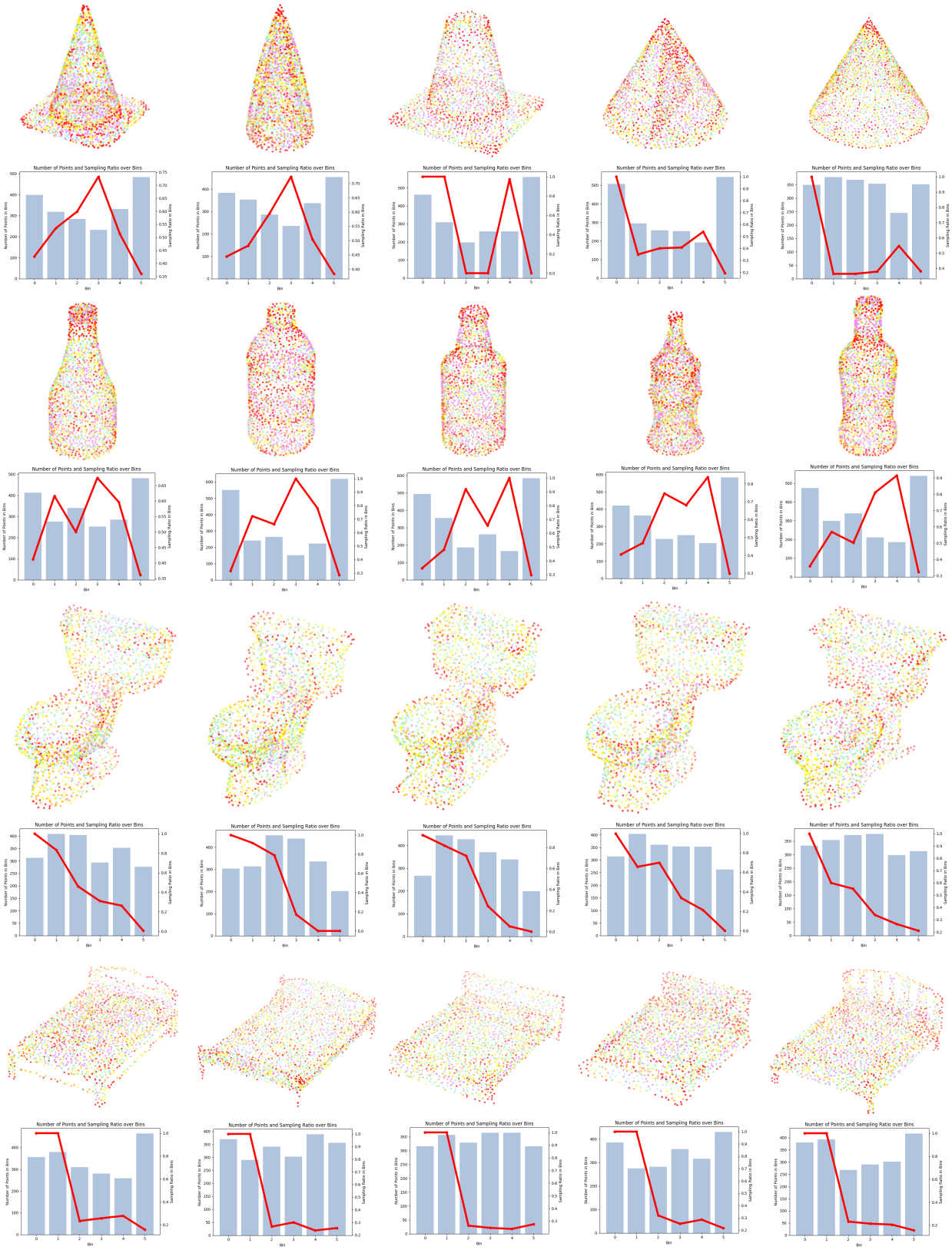

Figure 25. More visualization results of bin partitioning and learned shape-specific sampling strategies on the cone, bottle, toilet, and bed categories. Zoom in for optimal visual clarity.

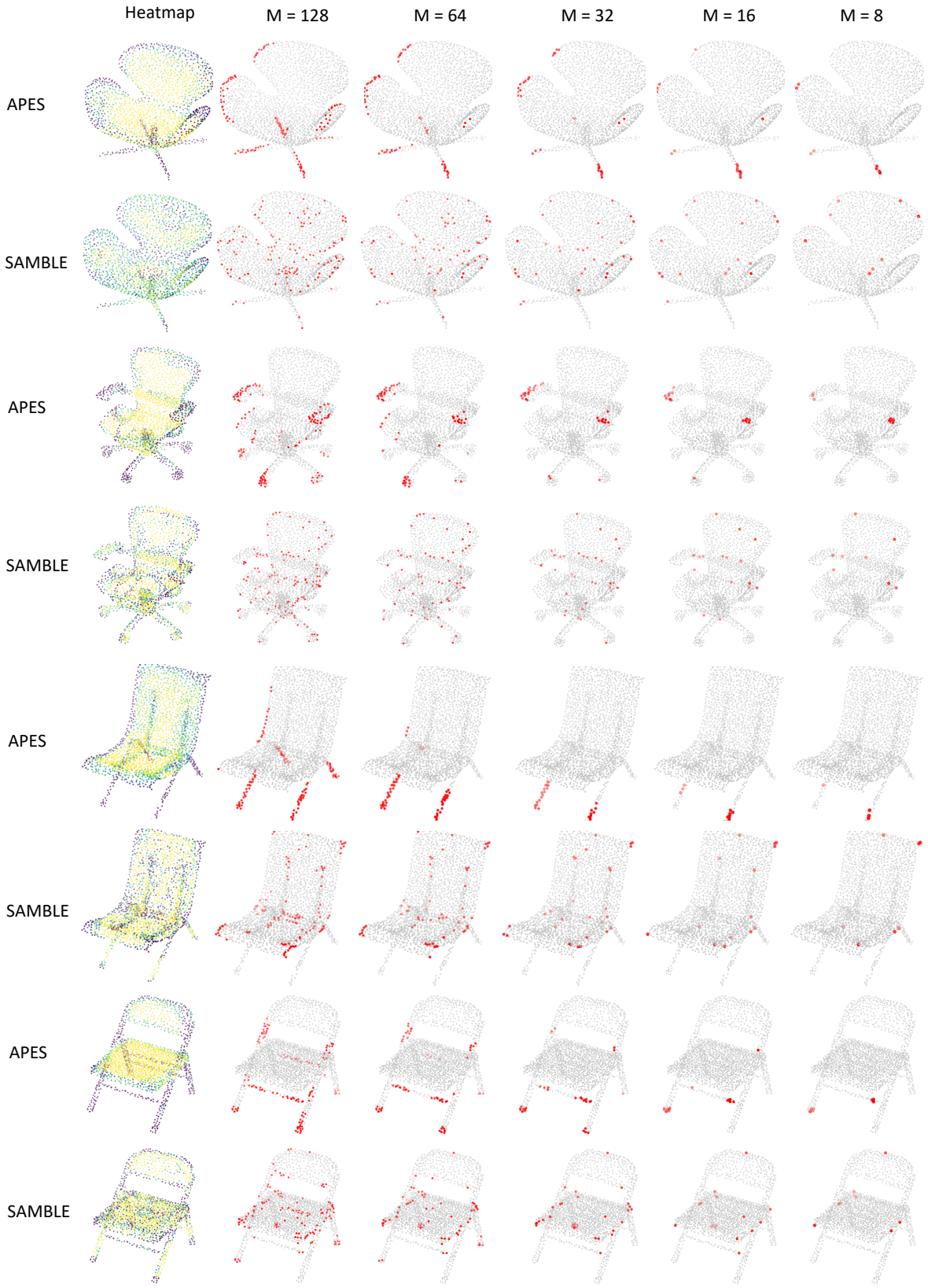

Figure 26. Sampled results of few-point sampling on the chair shapes. No pre-processing with FPS into $2M$ points was performed. Zoom in for optimal visual clarity.

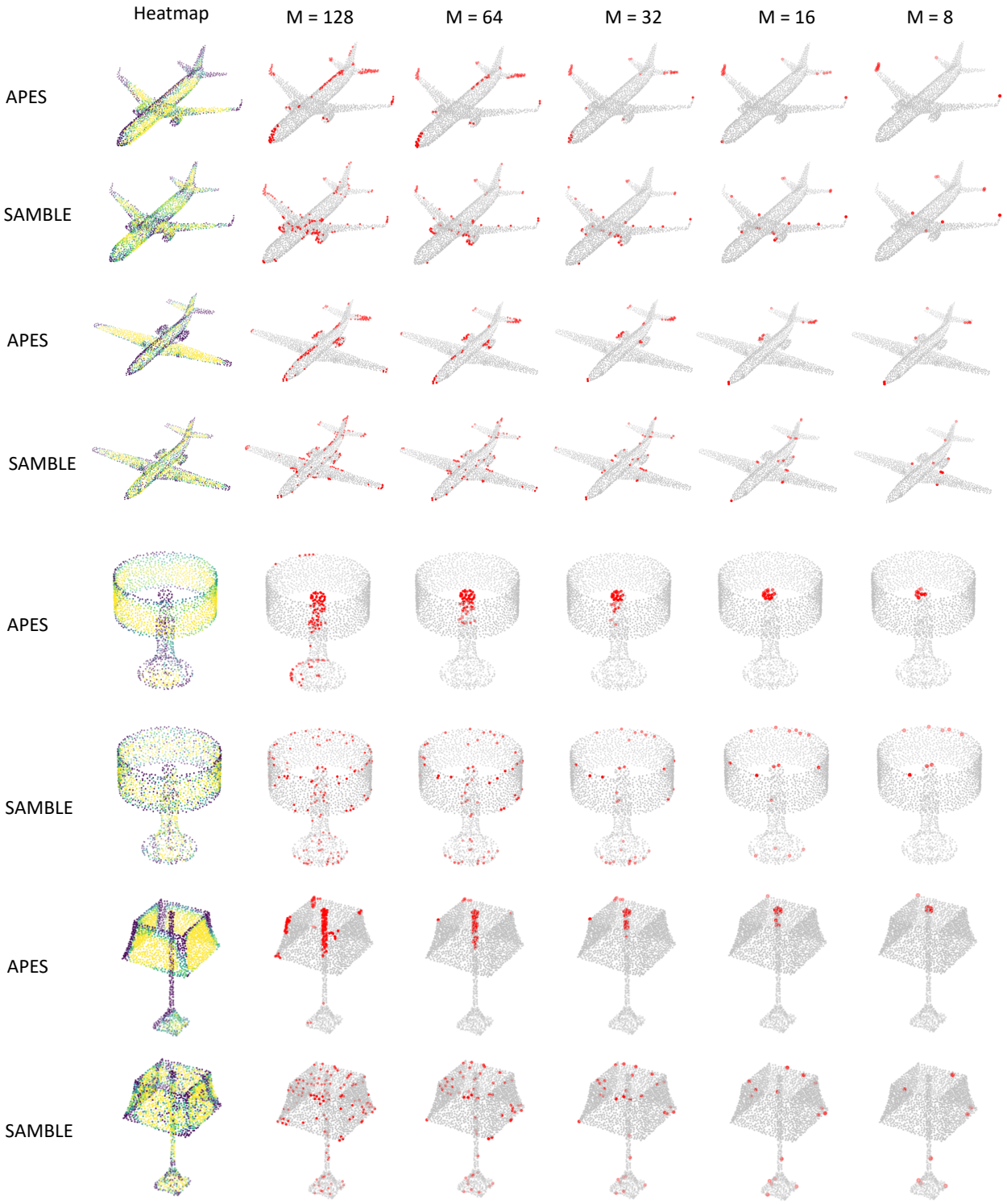

Figure 27. Sampled results of few-point sampling on the airplane and lamp shapes. No pre-processing with FPS into $2M$ points was performed. Zoom in for optimal visual clarity.