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

Supplementary Material

A. 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. 1. 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. 1, hence we use carve-based SAM as default for most experiments in our paper.

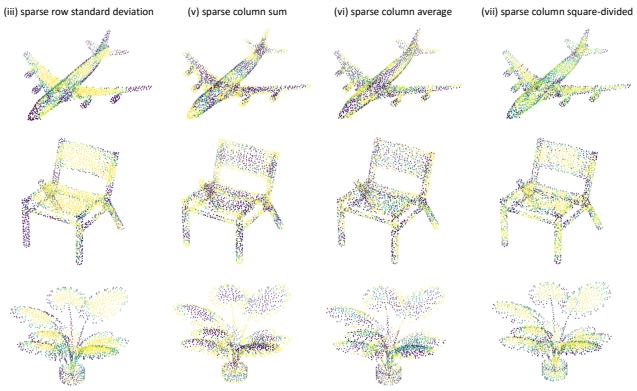

Figure 1. Heatmaps under different indexing modes with insert-based sparse attention map.

SAM	Indexing Mode	Cls. OA (%)	Seg. Cat. mIoU (%)	Seg. 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 1. Classification and segmentation performance of different indexing modes with different SAMs. Top-M sampling is adopted as the sampling strategy.

B. 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. 2 for better comprehensibility.

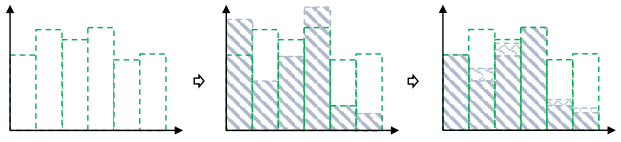

Figure 2. Illustration of redistributing excess points to other bins that have not been fully selected.

C. 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. 3. More detailed histogram results are provided in Sec. K.

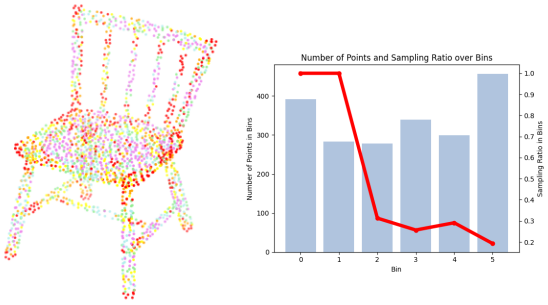

Figure 3. Left: bin partitioning, each color represents the points belonging to this bin. Right: the learned sampling strategy.

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 2. 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. 2, 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.

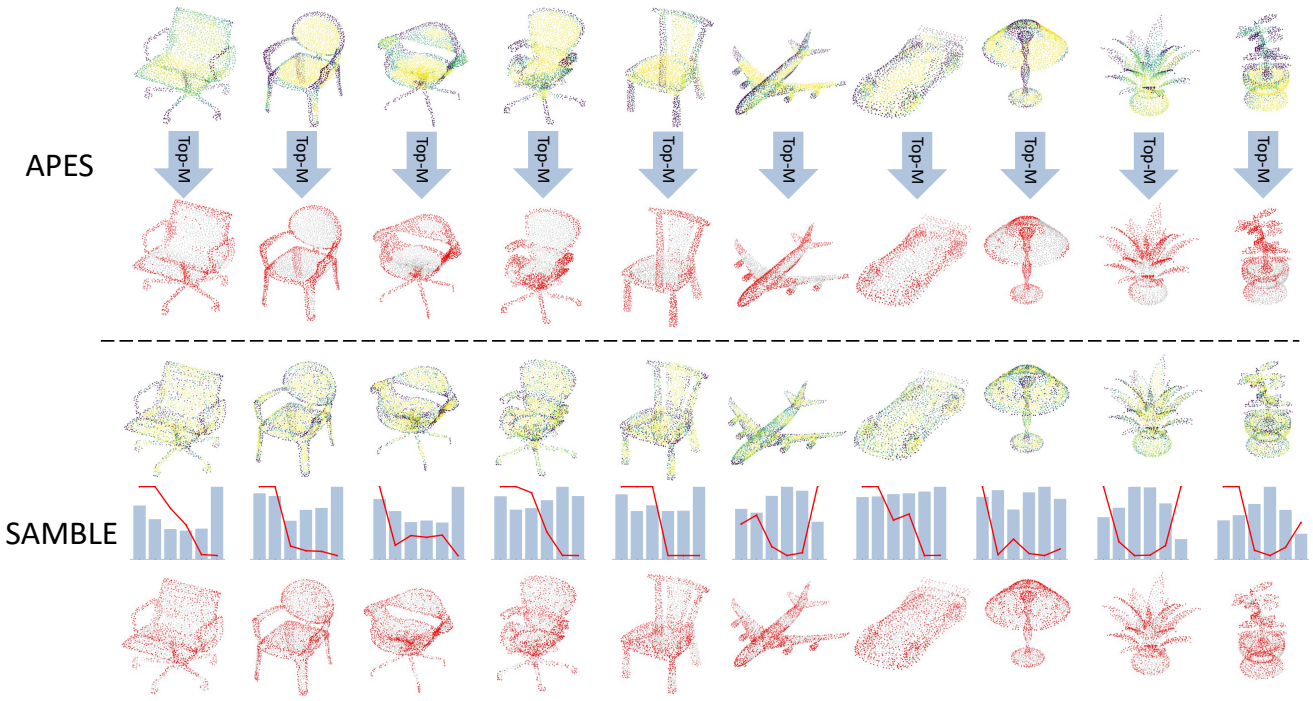

Figure 4. Qualitative results of our proposed SAMBLE, in comparison with APES. In addition to the sampled results, sampling score heatmaps and sampling strategies are also provided. All shapes are from the test set.

D. Network Architecture

For a fair comparison, the same basic network architectures from APES are used in our experiments, as illustrated in Fig. 5. The downsampling layers are replaced with our proposed ones, and the upsampling layers are replaced with the classical interpolation-based ones.

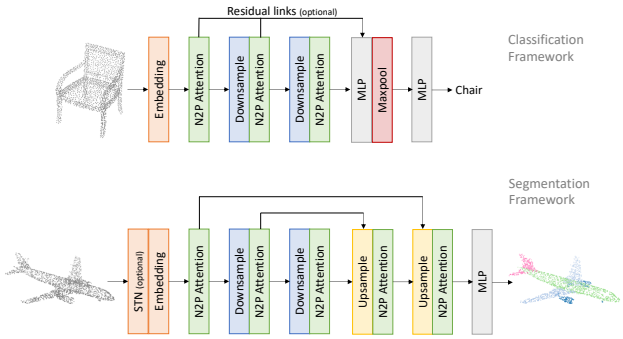

Figure 5. Network architectures for the classification task and the segmentation task.

E. 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.

F. Sampling Results in Comparison with APES

Additional qualitative results in comparison with APES are provided in Fig. 4 and Fig. 6. 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.

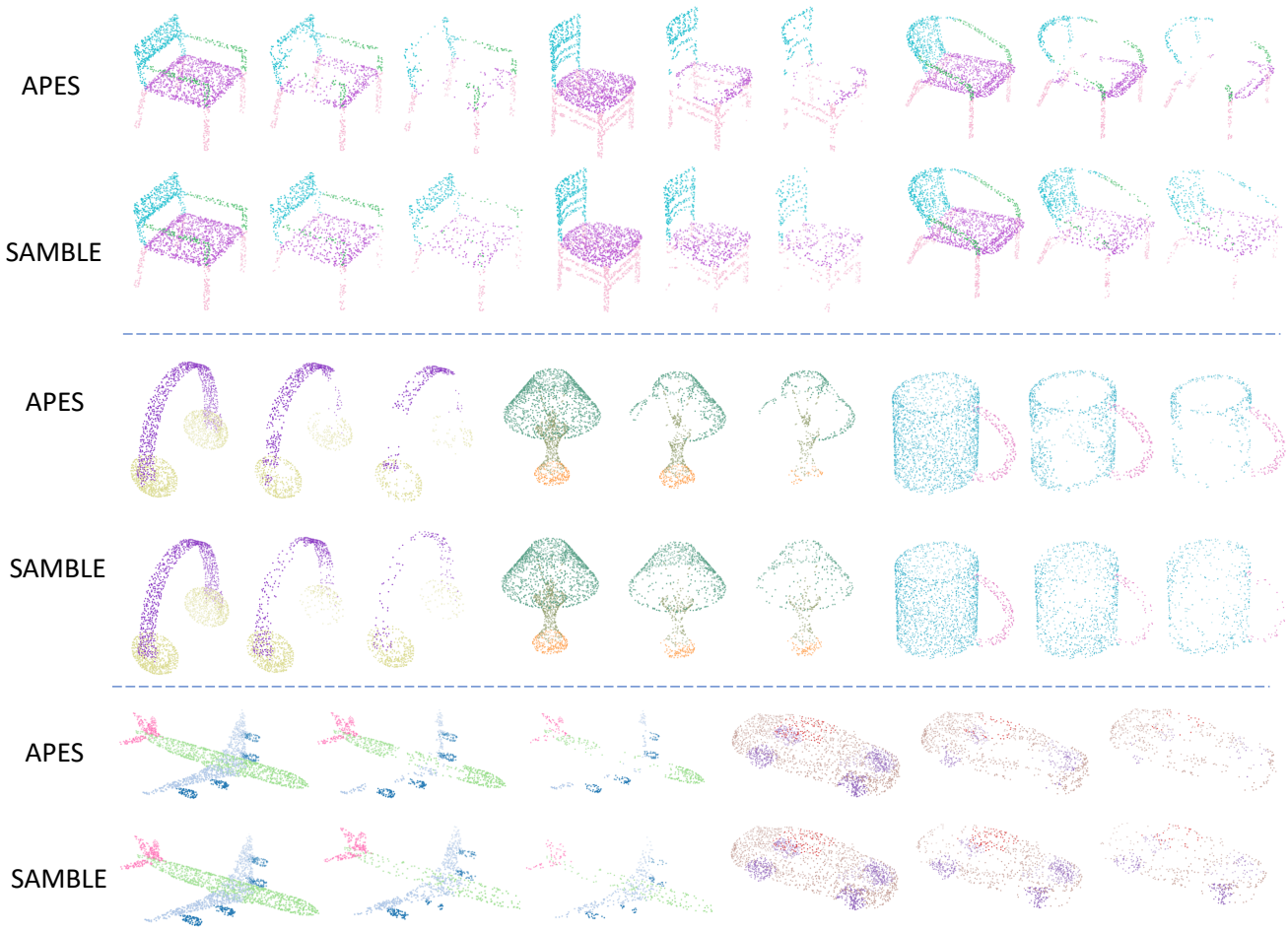

Figure 6. Segmentation results of our proposed SAMBLE, in comparison with APES. All shapes are from the test set.

G. 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. 7, 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. 7, 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

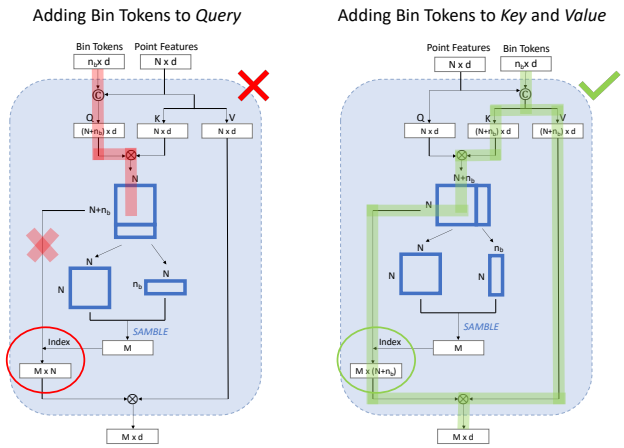

Figure 7. Adding bin tokens to Query leads to no gradient being backpropagated to the tokens, while adding bin tokens to Key and Value enables the gradient backpropagation.

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:

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

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. 8 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. 3, 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 3. 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:

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

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

$$m'_{ij} = \frac{e^{m_{ij}}}{\sum_{j=1}^{N+n_b} e^{m_{ij}}} \approx \frac{1 + m_{ij}}{N + n_b + \sum_{j=1}^{N+n_b} m_{ij}} \,. \tag{3}$$

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:

$$\ln(m'_{ij}) = \ln(\frac{e^{m_{ij}}}{\sum_{j=1}^{N+n_b} e^{m_{ij}}}) = m_{ij} - \ln(\sum_{j=1}^{N+n_b} e^{m_{ij}}) \tag{4}$$

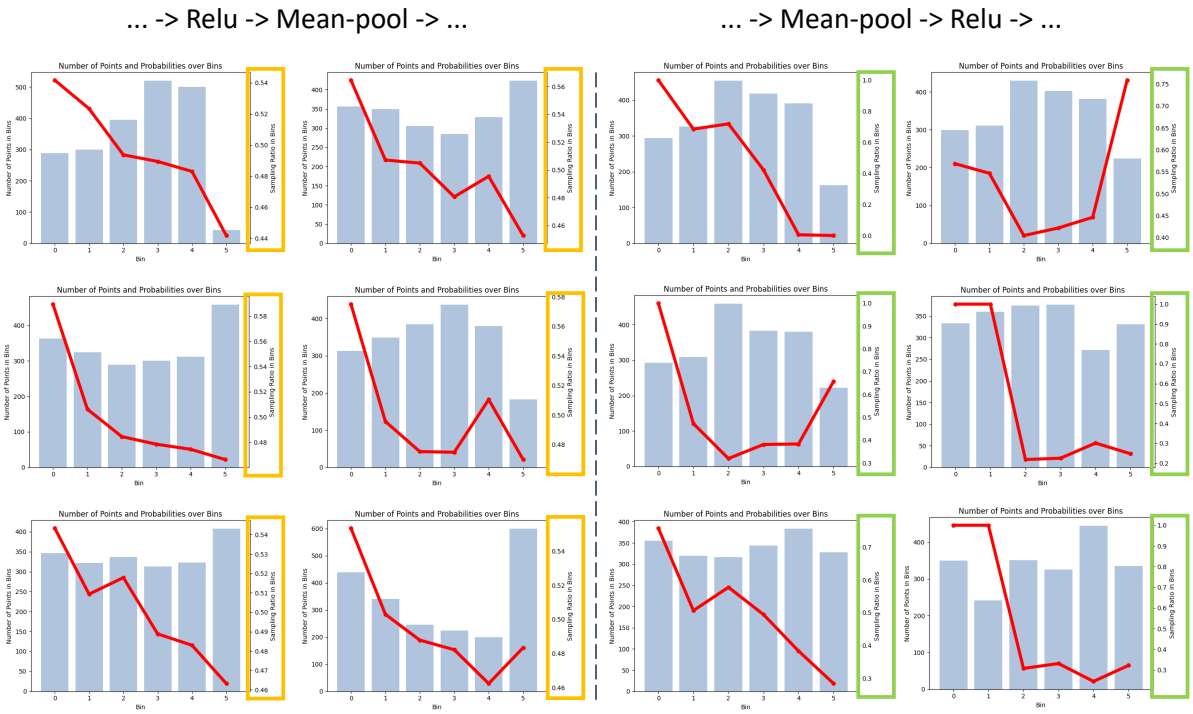

Figure 8. Learned sampling strategies with the mean-pooling and ReLU operations applied in different orders.

After the logarithmic operation, every value in the sub-attention map is negative. Therefore, a normalization operation is necessary. However, as shown in Fig. 9, 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. (4) 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.

H. 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

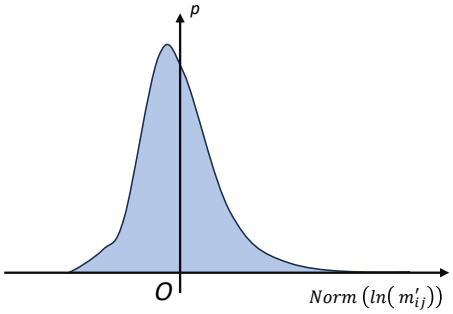

Figure 9. Illustrative figure of the distribution of the element values in the post-softmax attention map, after normalization.

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. 4. 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. 10. 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 4. Classification performance with different values of the momentum update factor γ.

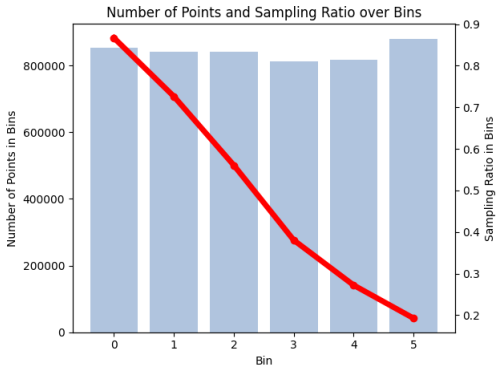

Figure 10. Partitioning the distribution of point sampling scores of all shapes and points in the test dataset into bins with the learned boundary values.

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:

$$\rho_{\mathbf{p}_i} = \frac{e^{a_{\mathbf{p}_i}/\tau}}{\sum_{i=1}^{N} e^{a_{\mathbf{p}_i}/\tau}}, \tag{5}$$

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. 11. 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. 11 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. 5, where $\tau = 0.1$ achieves the best performance.

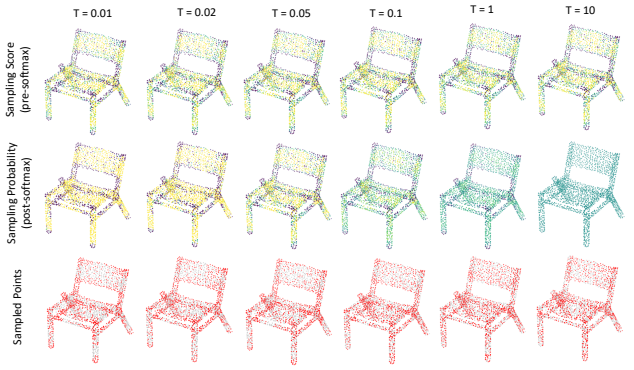

Figure 11. Different sampling results using different τ in the softmax with temperature during the sampling process. The indexing mode is the sparse column square-divided.

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

I. Sampling Policy Comparison

Three different sampling policies are illustrated in Fig. 12, 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.

J. 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. 6. 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

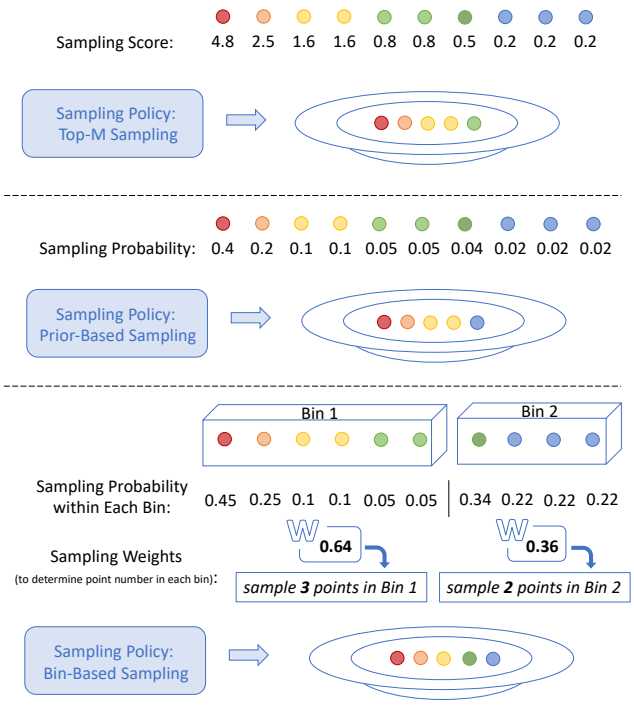

Figure 12. An illustration of different sampling policies. Note for bin-based sampling, either top-M sampling or prior-based sampling may be used within each bin.

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. 6, 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. 12). 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 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.

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 6. 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.

K. More Visualization Results

Learned Shape-Specific Sampling Strategies. We present additional extensive results in Fig. 13, Fig. 14, Fig. 15, and Fig. 16 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. 17 and Fig. 18. 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.

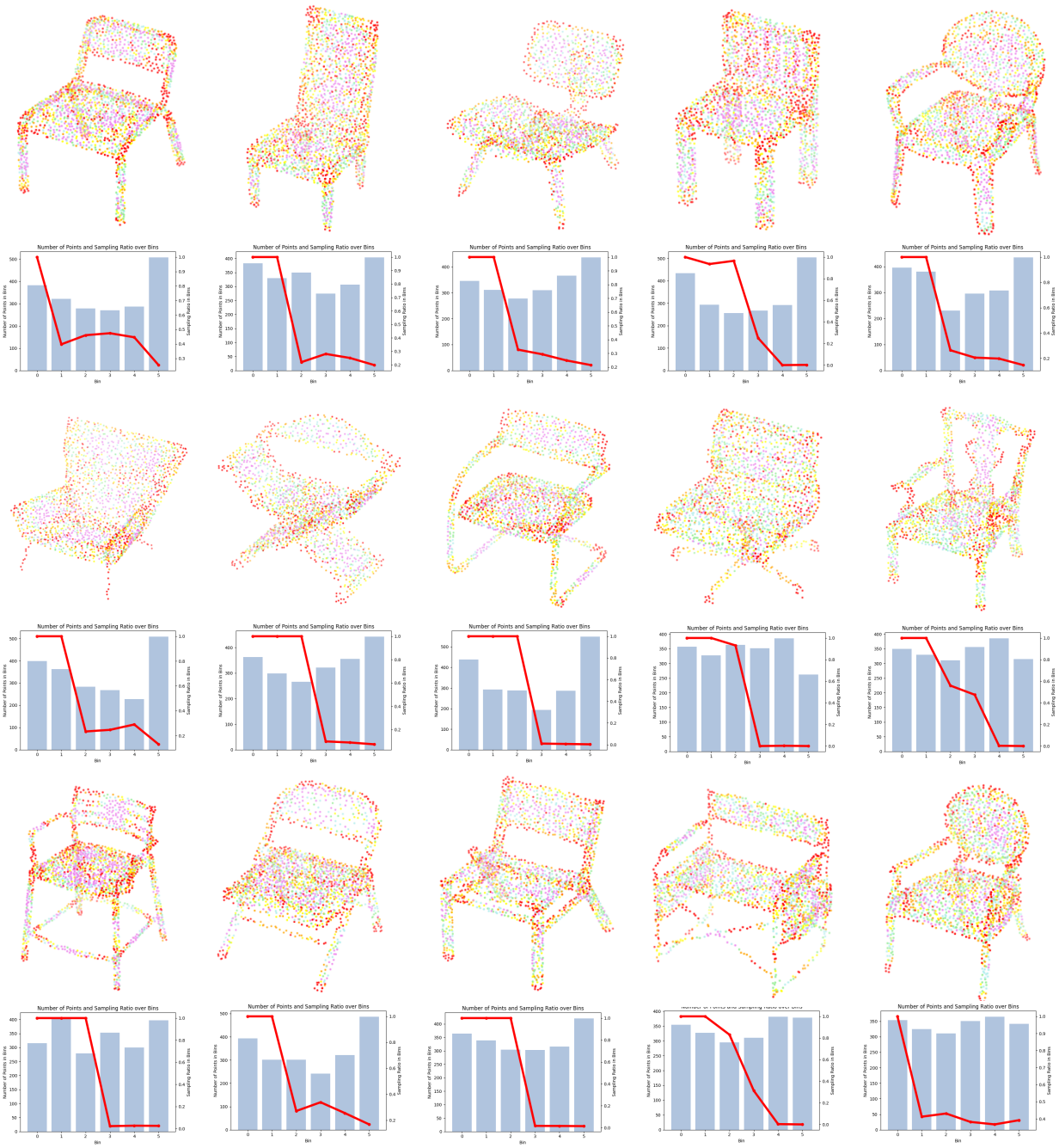

Figure 13. More visualization results of bin partitioning and learned shape-specific sampling strategies on the chair category. Zoom in for optimal visual clarity.

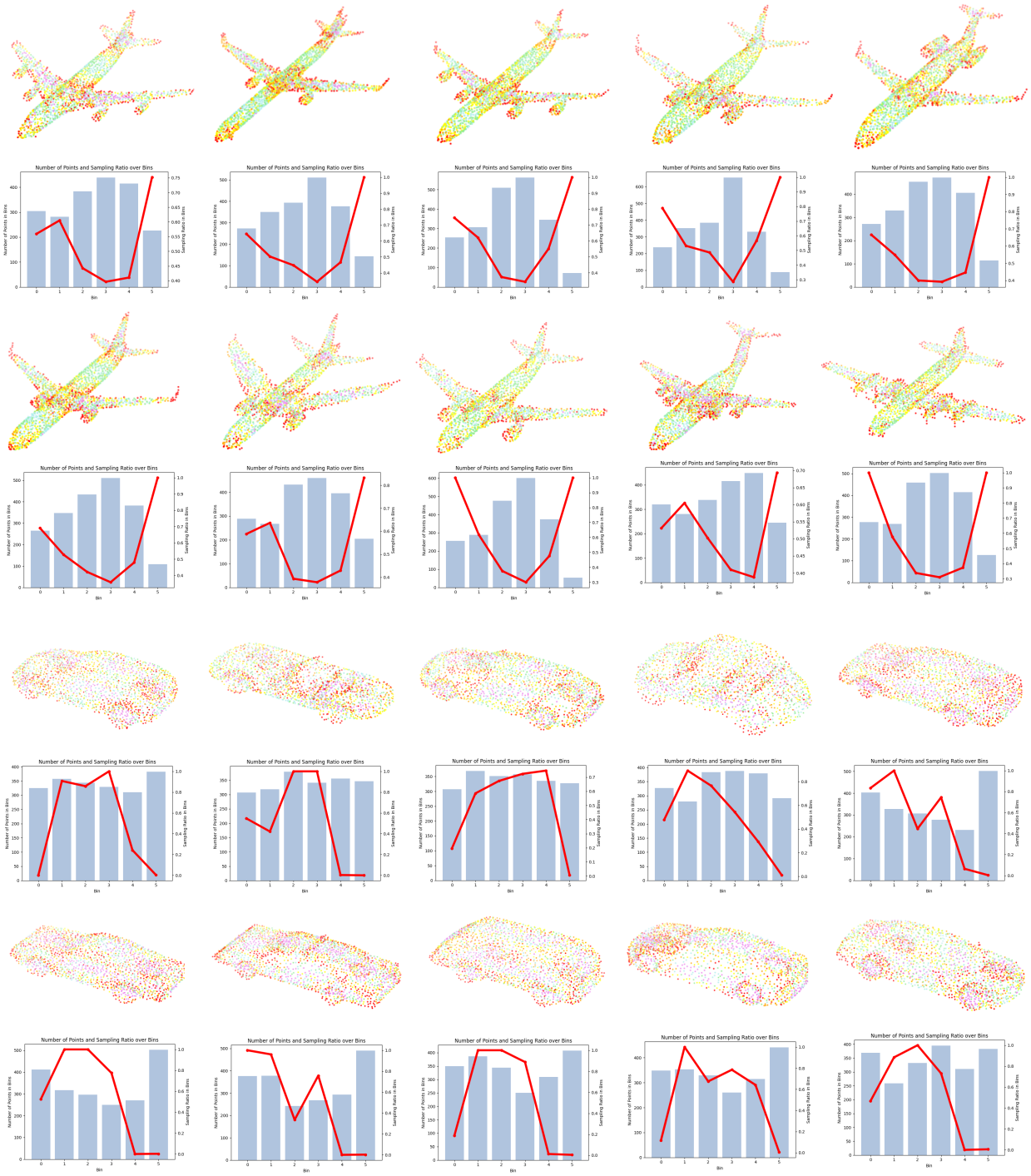

Figure 14. More visualization results of bin partitioning and learned shape-specific sampling strategies on the airplane and car categories. Zoom in for optimal visual clarity.

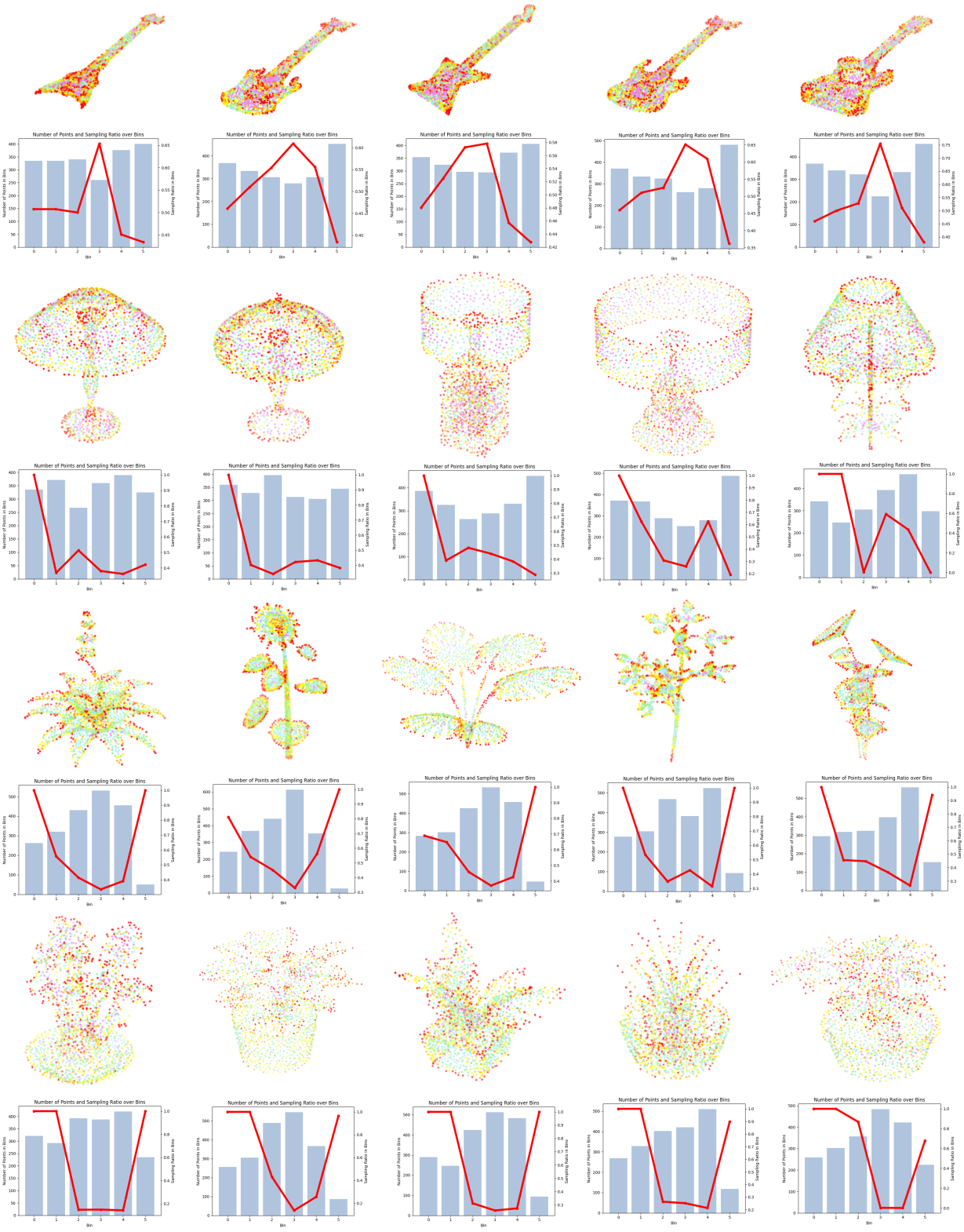

Figure 15. More visualization results of bin partitioning and learned shape-specific sampling strategies on the guitar, lamp, plant, and flower pot categories. Zoom in for optimal visual clarity.

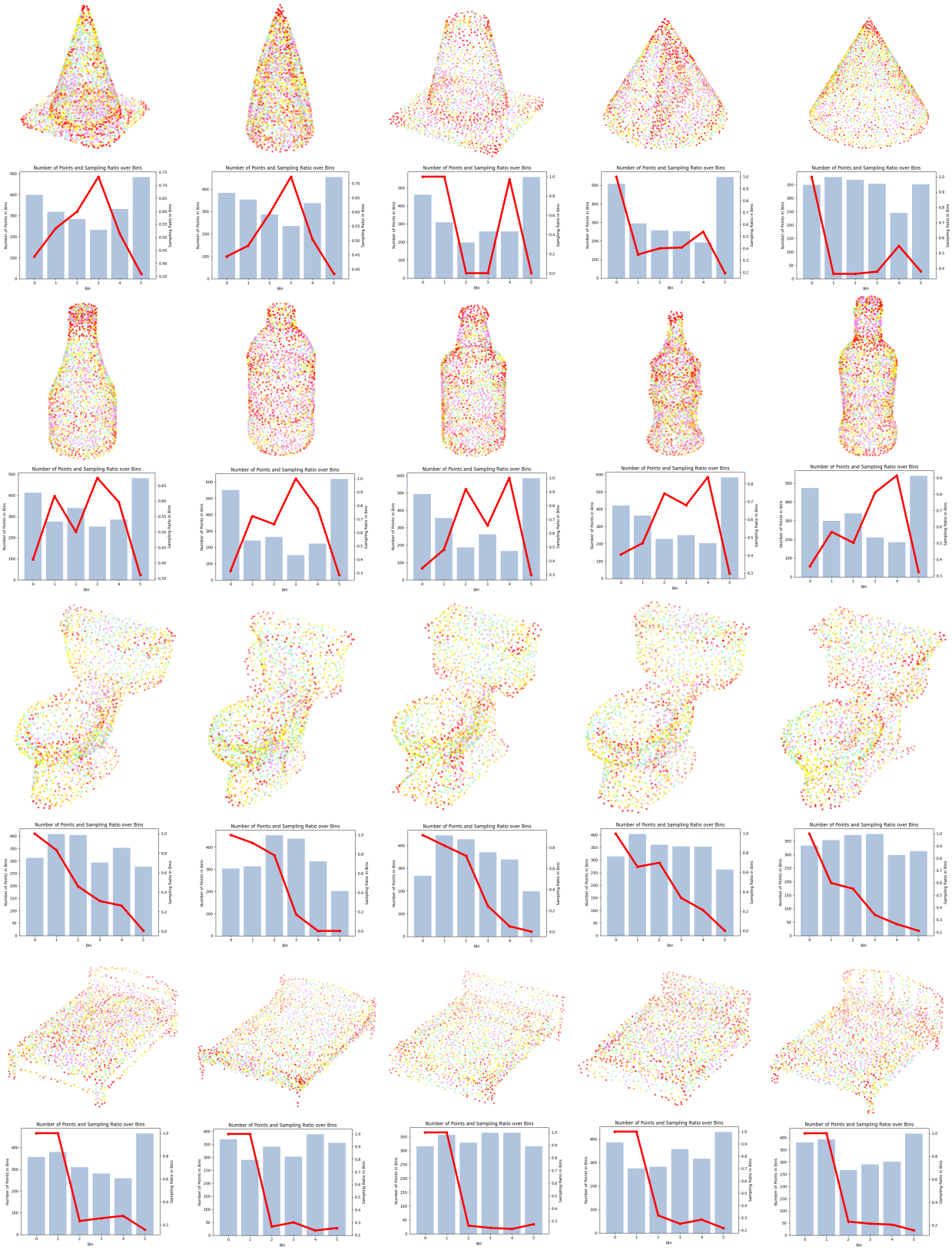

Figure 16. More visualization results of bin partitioning and learned shape-specific sampling strategies on the cone, bottle, toilet, and bed categories. Zoom in for optimal visual clarity.

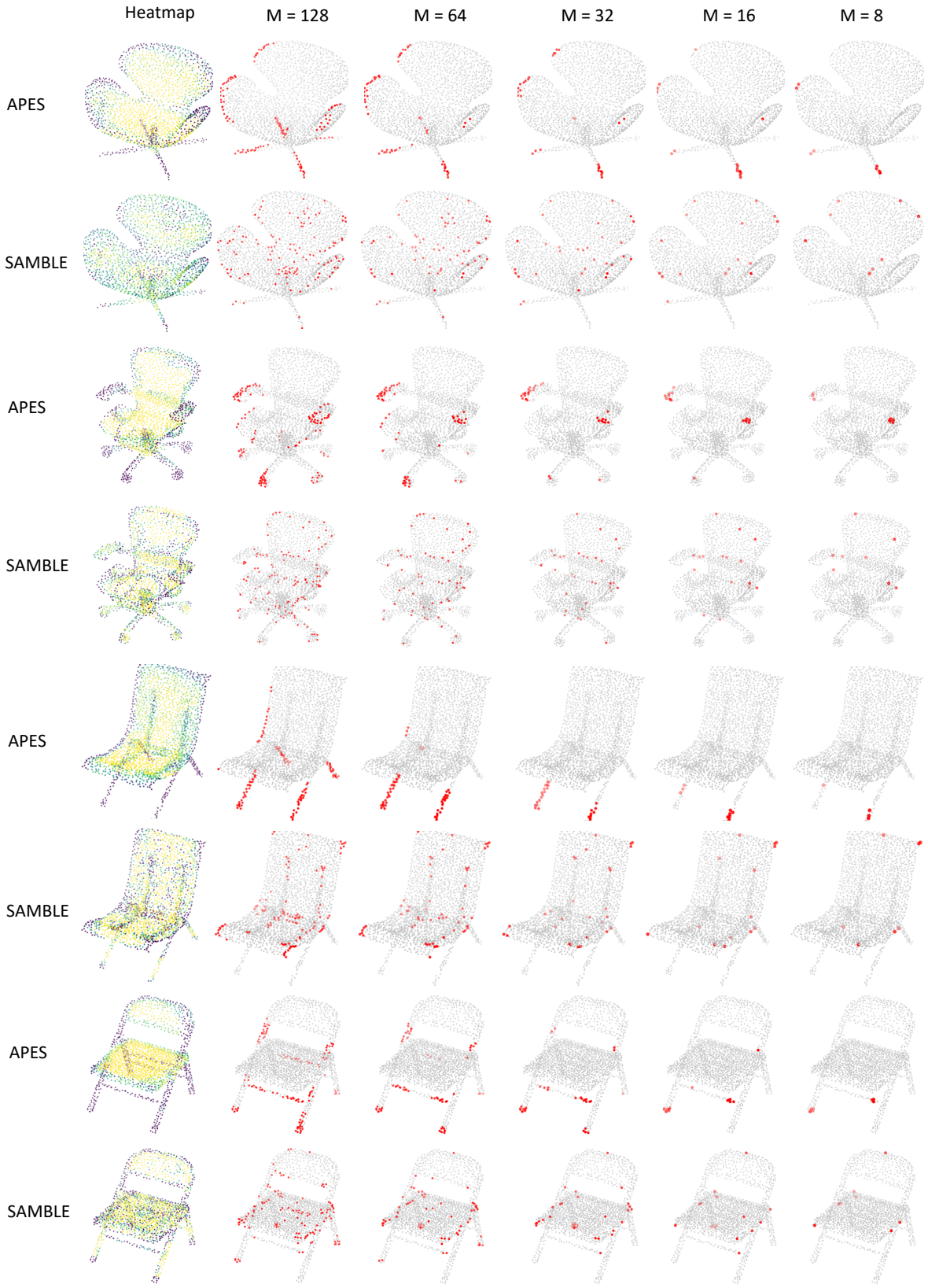

Figure 17. Sampled results of few-point sampling on the chair shapes. No pre-processing with FPS into $2M$ points was performed. Zoom in for optimal visual clarity.

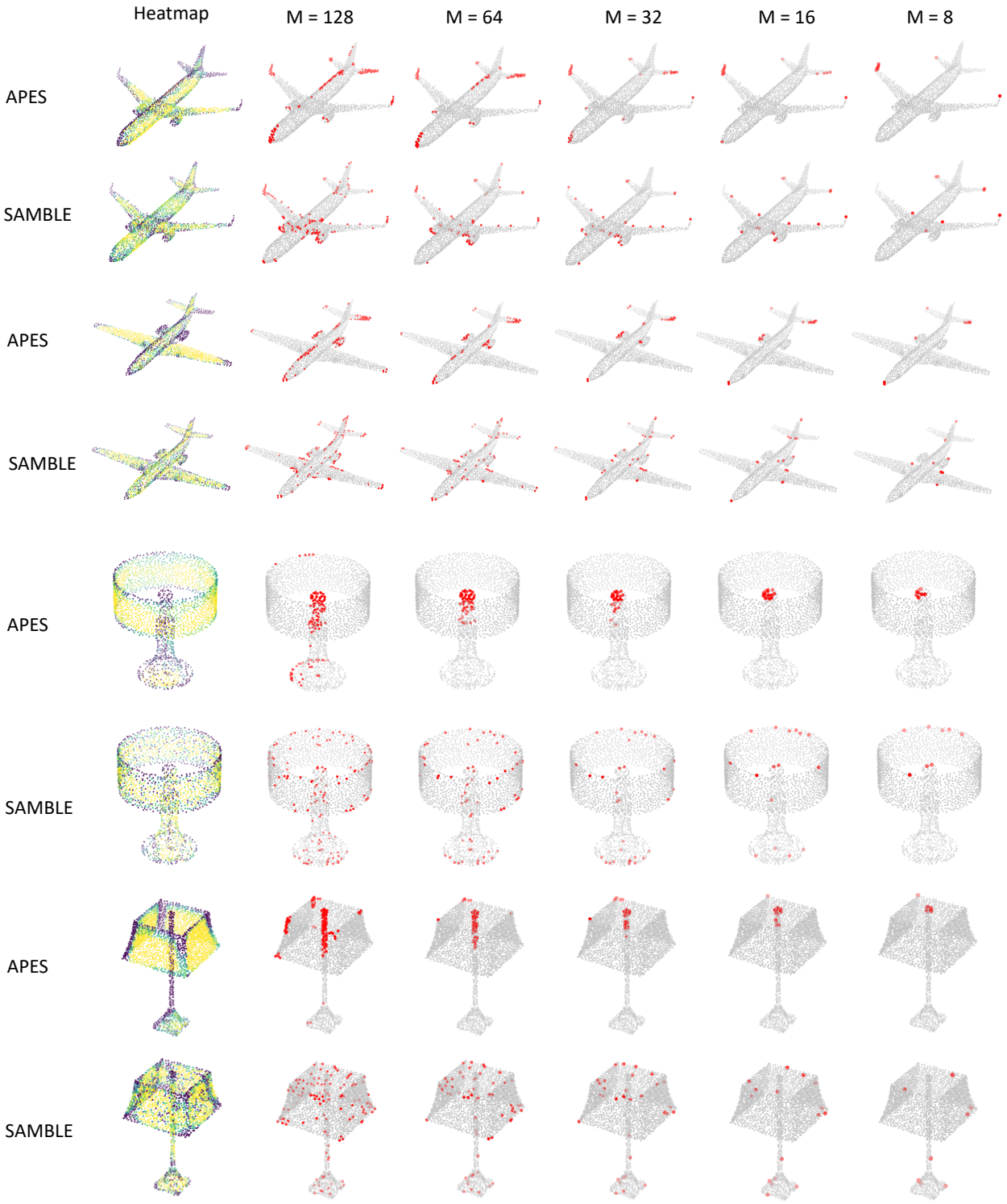

Figure 18. Sampled results of few-point sampling on the airplane and lamp shapes. No pre-processing with FPS into $2M$ points was performed. Zoom in for optimal visual clarity.