Mitigating Geometric Degradation in Fast DownSampling via FastAdapter for Point Cloud Segmentation

Supplementary Material

A. Geometric Information Degradation in Fast Downsampling

In this section, we visualize and analyze the geometric information degradation problem present in fast downsampling, with the results shown in Fig. A. We visualized the results of FPS, RandomSample, and GridSample on ScanNet. For FPS and RandomSample, we chose a 32x downsampling, while for GridSample, we selected a voxel size of 0.12. This choice ensures that the number of points in GridSample is roughly aligned with that of RandomSample and FPS for a fair comparison. First, we can observe that FPS, due to its strict constraints, maintains relatively complete geometric information in its downsampling results. On the other hand, RandomSample has a significant drawback: due to its randomness, certain local regions may be completely discarded during the downsampling process, leading to the destruction of their geometric structures, which greatly affects the model's semantic judgment. As for GridSample, its voxel-based downsampling method ensures that no local region is lost during the process. However, one issue is that it samples each local region equally, ignoring the impact of density. Thus, in densely populated areas, more points should be downsampled to capture additional object information, a capability that can be achieved by both FPS and RandomSample. However, GridSample fails to achieve this effect, leading to geometric information degradation compared to FPS.

In summary, the two commonly used fast downsampling methods (RandomSample and GridSample) exhibit varying degrees of geometric information degradation compared to FPS. Our FastAdapter is designed to address this issue.

B. Finetuning Costs

Here, we provide a brief overview of the overhead involved when embedding FastAdapter for fine-tuning. First, regarding the experimental setup, we maintained the same default settings for the optimizer, learning rate scheduler, and data augmentation as those used in the baseline with FastAdapter.

For the training epochs, taking PointMetaBase-L/XL as an example, we set the fine-tuning epochs to only 15, while training the baseline from scratch requires 100 epochs. Furthermore, in the Table A, we report the parameter comparisons of FastAdapter embedded in different models. As we can see, the additional parameter we introduced is generally around 0.3¡ to 3M, which indicates that the trainable parameters during fine-tuning are approximately 3M. This demonstrates that the overhead for fine-tuning is quite minimal.

Table A. Additional Parameters

	PointNet++	PointMetaBase-XL	PTV3	RandLA
Baseline	1.0M	15.3M	46M	1.3M
+FastAdapter	1.3M	18.0M	49M	1.6M

Table B. Comparison with Other Fine-Tuning Methods

SetUps	mIoU(%)	mACC(%)	OA(%)
Baseline	69.47	75.81	89.87
DAPT	70.31	75.97	90.72
PointGST	70.65	76.49	90.80
FastAdapter	71.22	77.34	90.86

C. Comparison with Other Fine-Tuning Methods

The performance decline resulting from replacing FPS with random sample can be regarded as a distribution shift problem. Therefore, some state-of-the-art adapter-based fine-tuning methods can also be applied to address the issues associated with random sampling. In this section, we primarily compare FastAdapter with other advanced adapter-based fine-tuning methods (PointGST [17], DAPT [45]) to demonstrate the effectiveness of our approach. We used PointMetaBase-XL and conducted tests on S3DIS Area 5, with the results shown in Table B. Specifically, DAPT and PointGST are designed as transformer-based point cloud methods; however, they can also be applied to most hierarchical models. We treat the output of each encoder layer as the input for DAPT and PointGST, which is similar to the approach taken by FastAdapter.

First, we can observe that these advanced adapter-based fine-tuning methods effectively improve the performance decline of the model by adapting to the data distribution resulting from random sampling. Furthermore, DAPT focuses solely on the features of each input token by modulating their feature channels to adapt to the new distribution. However, it lacks attention to local regions, which makes it difficult to address the issue of losing local regions during random downsampling, resulting in limited performance improvements on this task. In contrast, PointGST constructs

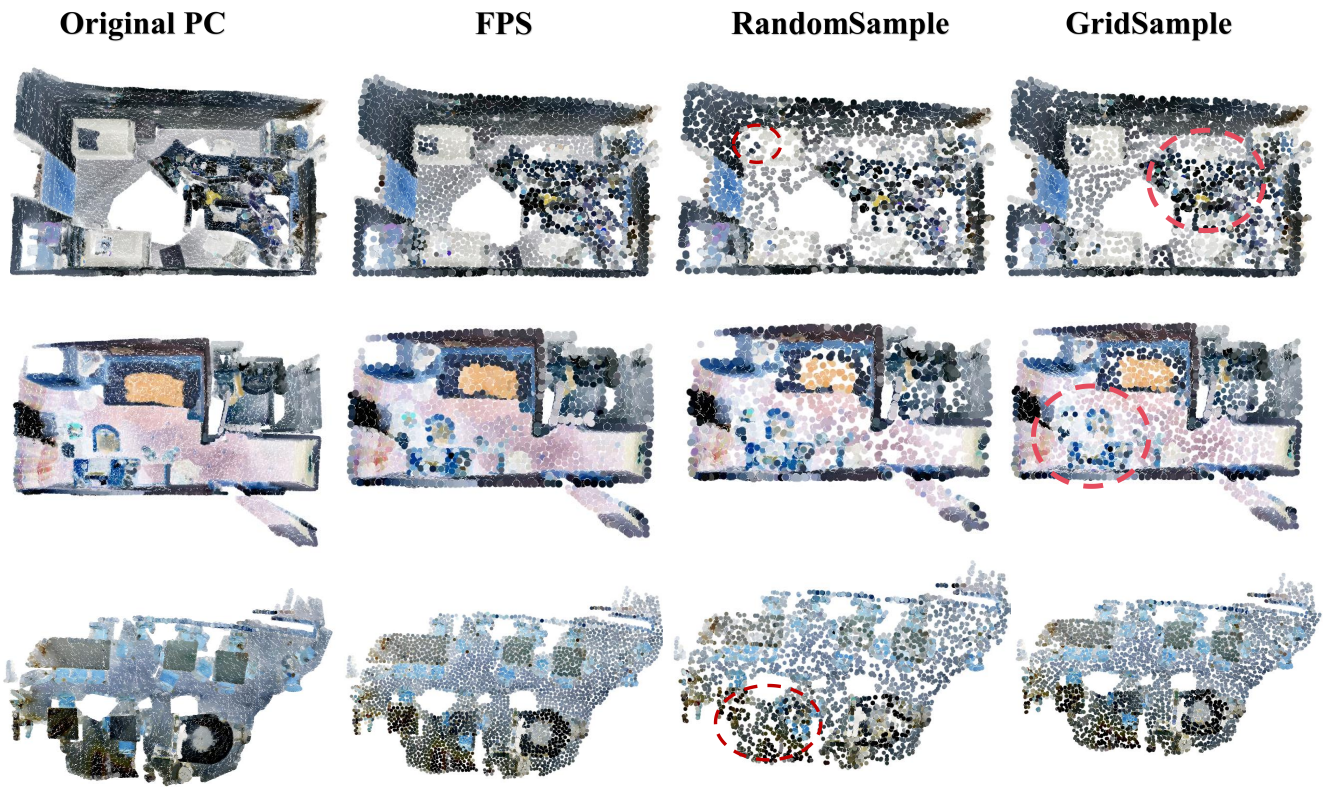

Figure A. Visualization Results of Different Sampling Methods.

local graphs for feature correction, allowing it to achieve better results than DAPT. Our FastAdapter enhances this by incorporating large receptive fields for local information and cross-layer interactions, thereby providing richer local features than PointGST, which enables it to achieve the best performance.