Beyond Farthest Point Sampling in Point-Wise Analysis

Yiqun Lin, Lichang Chen, Haibin Huang, Chongyang Ma,
Xiaoguang Han✉, *Member, IEEE,* Shuguang Cui, *Fellow, IEEE*

Abstract—Sampling, grouping, and aggregation are three important components in the multi-scale analysis of point clouds. In this paper, we present a novel data-driven sampler learning strategy for point-wise analysis tasks. Unlike the widely used sampling technique, Farthest Point Sampling (FPS), we propose to learn sampling and downstream applications jointly. Our key insight is that uniform sampling methods like FPS are not always optimal for different tasks: sampling more points around boundary areas can make the point-wise classification easier for segmentation. Towards the end, we propose a novel sampler learning strategy that learns sampling point displacement supervised by task-related ground truth information and can be trained jointly with the underlying tasks. We further demonstrate our methods in various point-wise analysis architectures, including semantic part segmentation, point cloud completion, and keypoint detection. Our experiments show that jointly learning of the sampler and task brings remarkable improvement over previous baseline methods.

Index Terms—3D vision, point cloud, point sampling, point cloud segmentation.

1 Introduction

With the rapid development of 3D sensing devices and the growing number of 3D shape repositories available online, it has become easier to access and process 3D data. One of the remaining challenges is 3D point cloud analysis, a popular 3D representation wildly used in the areas of robotics, autonomous driving, and virtual reality applications that [1], [2], [3], [4]. Unlike 2D images, point clouds are usually irregularly structured. Thus, directly converting them to 3D grids and applying 3DCNN [5], [6], [7] require more computational cost with limited performance.

Point sampling, as a task of selecting a subset of points to represent the original one at a sparse scale, can help to improve the efficiency of 3D data processing and has broad applications. So far, there are various handcrafted sampling strategies, such as random sampling, farthest point sampling [8], and grid (voxel) sampling [9], [10]. Recent works S-NET [11], and SampleNet [12] found that there exist better sampling ways by a learning method. Similar to 2D pooling layers, point sampling is also adopted in point-wise analysis architectures [13] for expanding receptive fields and reducing computational cost. Rather than as an input like [11], [12], the sampled points are used to group neighbor points and aggregate features. However, many researchers are studying feature aggregations [9], [14], [15] and rarely pay attention to point sampling. In this paper, we demonstrate that learning-based methods can perform better than uniform sampling strategies like FPS in point-wise analysis tasks.

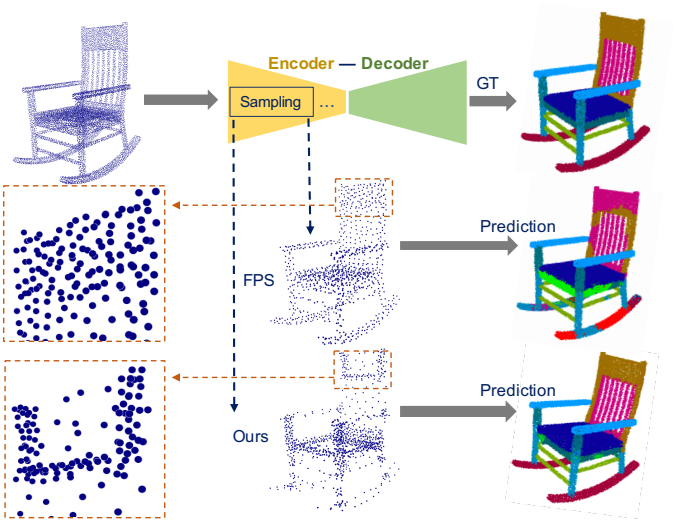

Fig. 1: We explore task-related samplings that can be learned by a novel training strategy in point-wise analysis tasks. For example, sampling points at a higher rate in boundary areas of a point cloud can help to aggregate more detailed features in boundary areas and significantly improve semantic segmentation results.

Point-wise analysis tasks including part segmentation [16], [17], [18], point cloud completion [19], [20], [21], and keypoint detection [22], usually require point-wise classification or regression. Our key observation is that points should have different semantics based on downstream tasks. Take part segmentation as an example. Points around the segmentation boundary are more sensitive than those that are far away. Hence the prior requires a higher sampling rate to group more boundary points for better results.

- *Y. Lin, X. Han and S. Cui are with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China. E-mail: {yiqunlin@link., hanxiaoguang@, shuguangcui@}cuhk.edu.cn*
- *L. Chen is with the Department of Electrical and Computer Engineering, University of Pittsburgh, PA, USA. E-mail: bobchen23@pitt.edu*
- *H. Huang and C. Ma are with Kuaishou Technology. E-mail: {jackiehuanghaibin, chongyangm}@gmail.com*
- *Xiaoguang Han is the corresponding author.*

In general, it is always possible to design a handcrafted sampler with ground truth information related to the task to provide better performance than FPS. Here we call the handcrafted sampler with task-related ground truth information as Task-GT sampler. Since the GT information cannot be observed during inference, our goal is to learn that Task-GT sampler for each specific task automatically.

To this end, we develop a supervised sampler learning strategy. The core idea is to jointly learn sampling and optimize the given task. Specifically, the network learns to minimize the displacement of sampled points w.r.t handcraft GT sampler and the downstream task-related loss. After training, the learned sampler can provide task-related information for better inference by approximating the Task-GT sampling points. To verify the proposed strategy, we integrate it into different point-wise architectures such as PointNet++ [13], PointConv [14], FPConv [15] and evaluate them on various tasks, including segmentation, completion, and keypoint detection. Our experiments show that jointly training the sampler and tasks brings a significant improvement over previous methods with FPS.

To summarize, the main contributions of this work include:

- We explore sampling in point-wise analysis tasks and propose a novel task-related sampler learning strategy.
- The proposed sampler can be integrated into different point-wise architectures such as PointNet++ [13], PointConv [14] and FPConv [15].
- Extensive experiments are conducted on various backbones and tasks, including segmentation, completion, and keypoint detection, to show a significant improvement of our learnable sampler compared to FPS.
- Our work inspires two future research directions: "What is the best sampling in point-wise analysis?" and "How to further learn a better sampler?".

2 Related Works

Deep learning on point clouds. Deep learning has developed rapidly in recent years. In 2D images, convolutional neural networks (CNNs) play a significant role and greatly improve performance on almost every vision task. Nevertheless, due to the irregularity and sparsity of point clouds, applying pooling and convolution layers on point clouds is much more difficult than applying them on 2D images. Comprising approaches are proposed like applying convolution on 3D voxel space. Specifically, earlier works such as [5], [6], [7] voxelize the point cloud into volumetric grids, and apply 3D convolution and pooling layers for feature extraction. However, they suffer from low resolution and high computational cost problems, since the point cloud is extremely sparse and a large number of grids are empty. More recently, [10], [23], [24] propose to apply 3D convolution in a sparse way to reduce the computational cost, but their method will still lose details during voxelization.

To process point clouds efficiently and reduce detail loss, recent works explore new designs of local aggregation operators on point clouds. PointNet [25] proposes a shared MLP followed by a max-pooling layer to approximate continuous set function for unordered data processing. [9], [14], [26], [27], [28], [29] define convolution kernels for points with learnable weights, which are similar to image convolution. When the relationships among points have been established, graph convolution [18], [30], [31], [32], [33] can then be applied for local feature learning with high efficiency. Since 3D sensing devices capture points on object surfaces, [15], [34], [35], [36] attempt to operate feature aggregation on surfaces directly and apply 2D CNN or surface CNN instead of in a volumetric way.

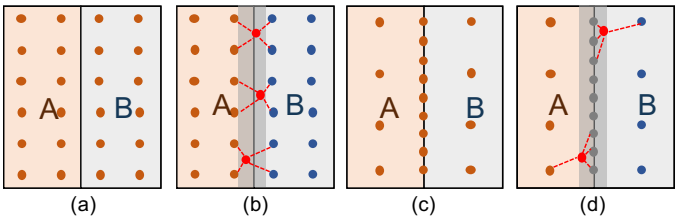

Fig. 2: Comparison of uniform sampling and task-oriented sampling. A plane is split to two parts, A and B. Points sampled uniformly in (a). As shown in (b), in upsampling steps, for points close to boundary area, their neighbors are half in part A and half in part B, which will increase the difficulty of classification. While in (c) and (d), points are sampled densely in boundary area, then the neighbors are only from the boundary area and part A (or B).

Previous aggregation methods are widely used in various high-level point cloud analysis architectures, including segmentation [9], [15], [16], [37], point cloud completion [19], [21], [38], object detection [39], [40], [41], [42] and keypoint detection [22]. However, those works mainly focus on local feature learning and architecture design. Few researchers study the impact of sampling in point-wise analysis, although it is important and indispensable.

Sampling methods for point cloud analysis. Sampling, which aims to represent the original point cloud in a sparse way, plays a key role in point cloud analysis. Widely used sampling strategies incorporate farthest point sampling [13], selecting a subset of points in which points are farthest away from each other, and grid (voxel) sampling [9], [10] to downsample over 3D voxels similar to the 2D pooling layer. These sampling methods aim to sample points in uniformly. Recently, [18] proposed a coverage-aware grid query (CAGQ) module for seeking optimal coverage over the original point cloud.

The samplers mentioned above are non-parametric and sensitive to outlier points, which means they may not be robust enough to handle real-world data. To overcome this drawback, [43] proposes a local adaptive shifting targeting on sampled points to improve the robustness and reduce the sensitivity to noisy points. S-NET [11] and SampleNet [12] develop parametric and task-oriented sampling strategies for a better sparse representation than FPS, demonstrating that FPS is not the best way for sparse encoding in some specific tasks, including classification, registration, and reconstruction. They guide the sampler to learn a similar point distribution under the supervision of the original point cloud and optimize it for downstream tasks.

However, in Section 5, we demonstrate that samplers of similar point distribution have no difference in point-wise analysis tasks. The possible reason is that adopting similar samplers will extract similar features from the previous point cloud. We have attempted to replace the learned sampler supervised with the original point cloud with FPS, while the changes are trivial. Hence, we propose a novel supervised sampler learning strategy to learn point sampling of different point distribution in point-wise architectures.

3 METHOD

We describe our sampler learning strategy in this section. We first revisit the impact of sampling on point-wise analysis networks and demonstrate that different tasks require different sampling strategies. We then present our learning approach for task-oriented sampling, from supervision generation to jointly training.

3.1 Sampling in Point-Wise Analysis Tasks

Sampling is an important component of deep learning-based approaches for multi-scale point cloud processing. By selecting essential points from the input, sampling can significantly reduce computational costs for deep network training. It also works like the pooling layer in 2D CNNs, the sampled points are used to group neighbor points by ball query or KNN query, and then we can adopt 3D convolutions on grouped local point clouds for feature aggregation. Usually, the sampling algorithm is predefined (more like a hyperparameter) for 3D networks and task-independent. Among existing sampling methods, FPS is a uniform sampling strategy and can cover all points in space as much as possible and give a stable performance on different tasks [13], [14], [15]. However, FPS is agnostic to downstream applications, which means the sampled points are selected only based on low-level information without considering object semantic and task-related information. The underlying assumption is that each point has the same importance to task learning.

Following this direction, we study the sampling problem in point-wise analysis. Our observation is that point-wise analysis tasks like point labeling would have significant point-wise semantics, and FPS is not optimal to select representative samples to learn. Taking point labeling as an example, as shown in Fig. 2, points near the segmentation boundary can provide more information than points far away. A plane is split evenly into two parts, A and B. During feature learning, methods like FPS will generate uniformly distributed samples like those in (a). The next step is to propagate and interpolate the learned features, as shown in (b). For points close to the boundary, their neighbor points can be two from part A and two from part B, increasing the difficulty of classifying which part they belong to since they may be confused when gathering two-part features. Hence, if we use a non-uniform sampling method like (c), adding more sampling points around the boundary can significantly improve the learning results (d) for the reason that the neighbors are only from one part and boundary area.

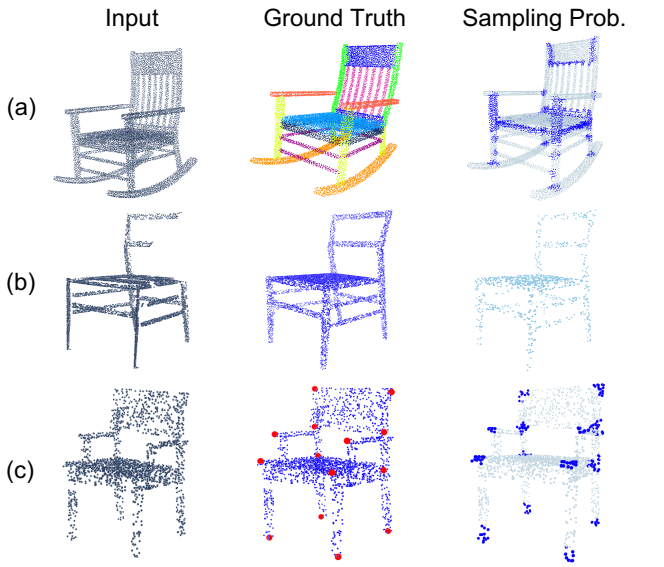

Fig. 3: Task-GT sampling points in different tasks. (a) In semantic segmentation task, points are sampled at a higher sampling rate in boundary areas while at a lower sampling rate in other areas; (b) in point cloud completion task, we directly perform uniform sampling on the complete point cloud (ground truth); (c) in keypoint detection task, points near keypoint have a higher sampling rate than others.

3.2 Learning the Sampling

As demonstrated in Section 3.1, introducing task-related information to the sampling stage can boost performance. However, the challenge is that there is no such information during the inference time to rely on. We propose a novel sampling learning strategy in a data-driven fashion: we first design a sampling method involving task information for each task and use them as an additional constraint during the task learning. Specifically, we refer to the task-related sampling as Task-GT sampling, and jointly train the sampler and the underlying task so that the generated sample points can approximate the Task-GT sampling points and optimize the task loss.

Supervising sampling. We first introduce how to incorporate task information into sampling. It is a process of adaptive sampling where we can change the underlying point density. Specifically, we change the sampling rate of different areas w.r.t the given task. In this paper, we demonstrate three different tasks, including segmentation, completion, and keypoint detection. As shown in Fig. 3, in the segmentation task, since the points in boundary areas are more sensitive, a higher sampling rate in boundary areas should be helpful for predictions on these points. Similarly, in point cloud comapletion, instead of sampling points from the original partial point cloud, we uniformly sample from the complete point cloud (ground truth) for a stronger hint for better surface completion. Sampling more points around keypoints will also be useful for keypoint detection. For more implementation details of the Task-GT sampler design, please refer to Section 4.

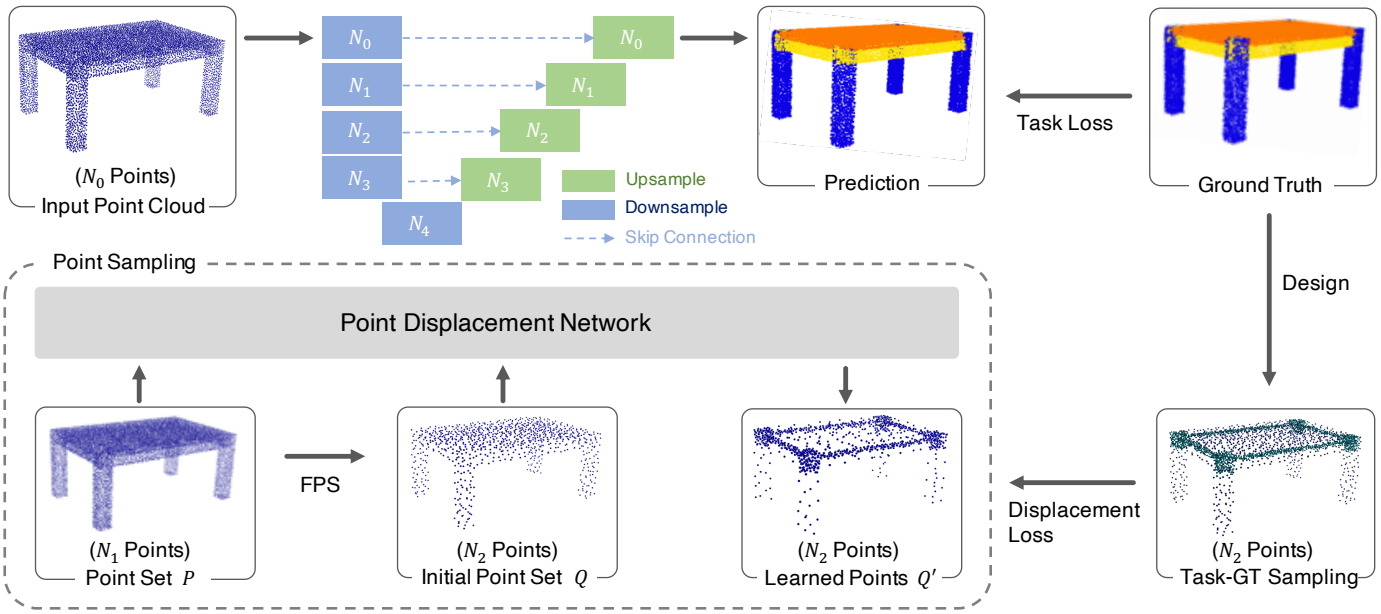

Fig. 4: Network architecture for point-wise analysis. We use a U-Net like architecture that consists of four downsampling and four upsampling layers. The final output is supervised by ground truth with task loss. In the point sampling step, we first select a subset of points as the initial point set and then predict the point-wise offset (or coordinates) to obtain the learned sampling points. The downsampling module uses the learned sampling points to replace FPS points (baseline). The displacement network is supervised by Task-GT sampling with CD (or EMD) loss, and also jointly supervised with task loss since it is one part of the whole task network.

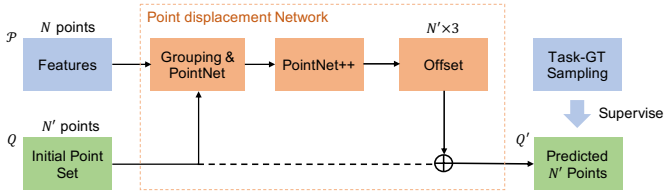

Fig. 5: Learn point displacement. Given a point set $\mathcal{P}$ with features and an initial subset Q, a PointNet++ is used for point-wise feature learning. We predict point-wise offsets to Q or coordinates directly as the final sampled points.

Learning points displacement. Although we have adaptive sampling methods to generate samples related to task information, it is still hard to directly apply to inference time without ground truth information. A naive approach is to train a network to learn such sampling in an end-to-end manner by generating sufficient paired data of point clouds and sample sets. However, such training can not recover the underlying task-related semantic information and may lead to overfitting of geometry-related features. Instead, we propose to train the sampler and task jointly by adding positional constrain of samples during training. To be specific, we additionally learn point displacement to approximate the distribution of the Task-GT sampling. Let $\mathcal{P}$ be the original point cloud with N points and features defined over each point. As shown in Fig. 5, we firstly select N' points as the initial point set Q, and group point features of Q from $\mathcal{P}$. Then a successive PointNet++ layer is used for point-wise offsets learning or directly regressing coordinates. For the selection of learning ways: offset learning or coordinate learning, we conduct ablative experiments in Section 5.

Network architecture. For point-wise analysis, our backbone network architecture is shown in Fig. 4, including four downsampling layers, four upsampling layers, and a fully connected layer at the end for point-wise prediction. There are three modules, sampling, grouping, and aggregation in the downsampling layer, similar to the set abstract module in PointNet++ [13]. Under the same architecture, we design various baseline models with different local aggregation operators to compare our supervised sampler learning strategy with FPS and Task-GT sampler. In our experiments, we take the FPS points as the initial point set for displacement learning. For the selection of the initial point set, we conduct ablative experiments in Section 5.

Loss functions. To put the predicted points closer to the sampled points of Task-GT sampler, we propose to optimize the displacement loss $\mathcal{L}_{\text{disp}}$ using Chamfer distance (CD) [44] or the earth mover's distance (EMD) [45], which are given by

$$\begin{aligned}\mathcal{L}_{\text{CD}}(S_p, S_{gt}) = &\frac{1}{2}\Big(\sum_{p\in S_p}\min_{q\in S_{gt}}||p-q||_2 \\ &+\sum_{q\in S_{gt}}\min_{p\in S_p}||q-p||_2\Big),\end{aligned} \tag{1}$$

$$\mathcal{L}_{\text{EMD}}(S_p, S_{gt}) = \min_{\phi:S_p\to S_{gt}}\sum_{p\in S_p}||p-\phi(p)||_2, \tag{2}$$

where S_p is the predicted point set, S_{gt} is the point set sampled by Task-GT sampler and ϕ indicates the bijection between S_p and S_{gt}. We recommend to compute bijection matching using the initial point set instead of predicted

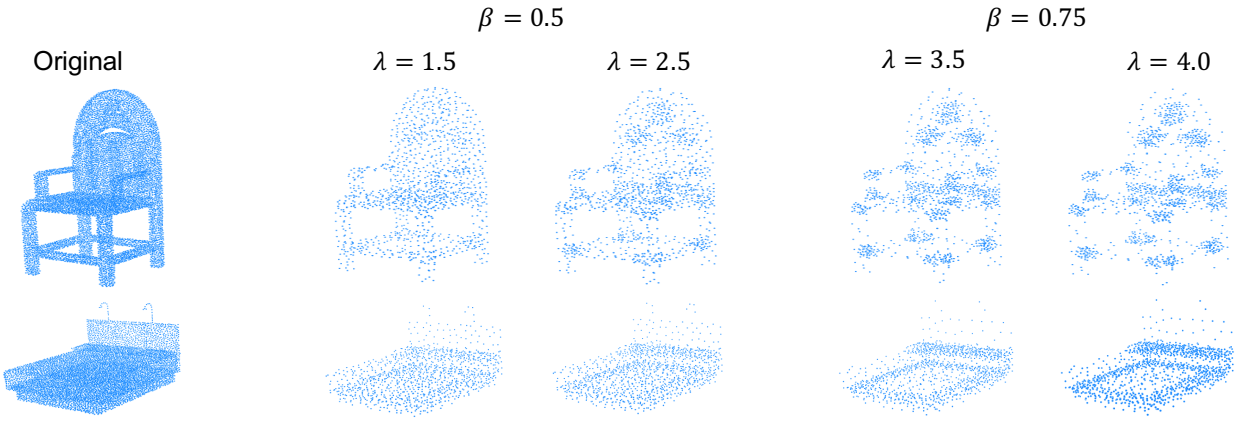

Fig. 6: Visualization results of *Edge FPS* with different hyperparameters (λ and β). Points are sampled more densely in boundary areas than others. Larger λ means to sample more points from boundary areas.

points for a stable shape supervision. Let S_h be the initial point set and p is the predicted point from h. The formulation of EMD loss with initial points is shown as in Equations 1 and 2. We compare the impact of different displacement loss function in Section 5.

$$\mathcal{L}_{\text{EMD}^\star}(S_p, S_h, S_{gt}) = \min_{\phi: S_h \to S_{gt}} \sum_{p \in S_p, h \in S_h} ||p - \phi(h)||_2. \quad (3)$$

To optimize the point set S_p to the task, we jointly train the sampler with task loss. The overall loss function is given in Equation 4, where λ indicates the scale ratio and varies among different tasks.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda \cdot \mathcal{L}_{\text{disp}}. \quad (4)$$

Method		Shape mIoU	Part mIoU	oA
Baseline		49.8	40.4	81.0
$\beta = 0.5$	$\lambda = 1.5$	52.0	44.6	84.2
	$\lambda = 2$	53.4	43.4	84.7
	$\lambda = 2.5$	54.0	44.6	85.2
	$\lambda = 3$	54.2	45.3	85.6
$\beta = 0.75$	$\lambda = 3.5$	**54.7**	**45.6**	**86.0**
	$\lambda = 4$	54.6	44.5	86.0

TABLE 1: Comparison experiments on hyper-parameters (λ and β) selection of *Edge FPS*. A larger λ means that there are more points sampled in boundary area. β is a clip ratio to prevent sampling points from exceeding a certain percentage.

4 Experiments

To demonstrate the efficacy of our proposed sampler learning strategy, we conduct experiments with three different local aggregation operators including PointNet++ [13], PointConv [14] and FPConv [15] under the same architecture shown in Fig. 4. Comparison experiments are conducted in three point-wise analysis tasks, including semantic part segmentation on PartNet [16], point cloud completion on ShapeNet [46] and keypoint detection on KeyPointNet [22]. In the following experiments, we compare different sampling strategies only in the second layer. Ablation study on layer selection is shown in Section 5.

4.1 Semantic Segmentation

Dataset and evaluation metrics. We conduct experiments on PartNet [16] dataset, which is annotated with fine-grained, instance-level, and hierarchical 3D part information. It consists of 573,585 part instances over 26,671 3D models and covers 24 object categories. For some categories with a small amount of data (less than 2,000), we repeat the training data until it exceeds 2,000 for a stable convergence. Each object's point cloud has 10,000 points. In the experiments, we downsample 4,096, 1,024, 256, and 64 points in the four downsampling layers, respectively.

We use the mean Intersection-over-Union (mIoU) scores as the evaluation metric. Following [16], we first remove the unlabeled ground truth points and calculate the IoU between the predicted points and ground truth points for each part category. Then, we obtain part-category mIoU by averaging the per-part-category IoU to evaluate the performance of every semantic part. We further calculate another evaluation metric, shape mIoU by averaging per-shape IoU to evaluate the performance on every object sample.

Design of supervising sampling. We formally introduce Edge FPS. Firstly, we define a boundary point that has at least one neighbor point (i.e., within a radius equals to ϵ) labeled with a different semantic label [47]. Let $\mathcal{P}$ be the original point set with N points and $N^{'}$ be the number of points to sample. Supposing that there are N_b boundary points, the total number of boundary points to sample is given by

$$N_b^{'} = \max(\lambda \frac{N_b}{N} N', N_b, \beta N'), \quad (5)$$

where λ is the scale ratio ($\lambda \geq 1$) and β is the clip ratio ($0 \leq \beta \leq 1$). The number of sampled non-boundary points is

$$N_n^{'} = N' - N_b^{'}. \quad (6)$$

Thus, we propose *Edge FPS* to sample $N_b^{'}$ and $N_n^{'}$ points from boundary points and non-boundary points by FPS, respectively. As shown in TABLE 1, we compare different λ and β, then we choose the best setting of $\lambda = 3.5$ and

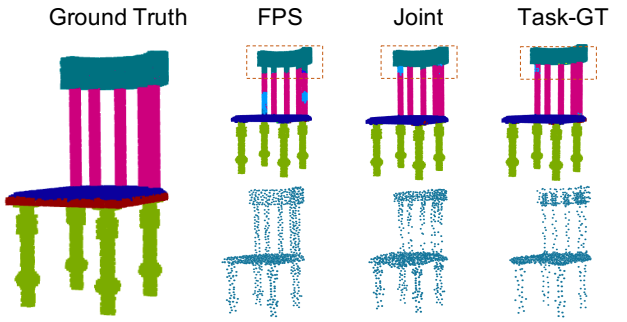

Fig. 7: Comparison of different sampling strategies (*FPS*, *Joint* and *Task-GT* sampling) on semantic segmentation task. *Joint* represents adopting learnable sampler and jointly supervising it with task and displacement loss. The results show that predictions are more consistent near the boundary in Task-GT sampling, and the jointly learned sampling can approximate it well.

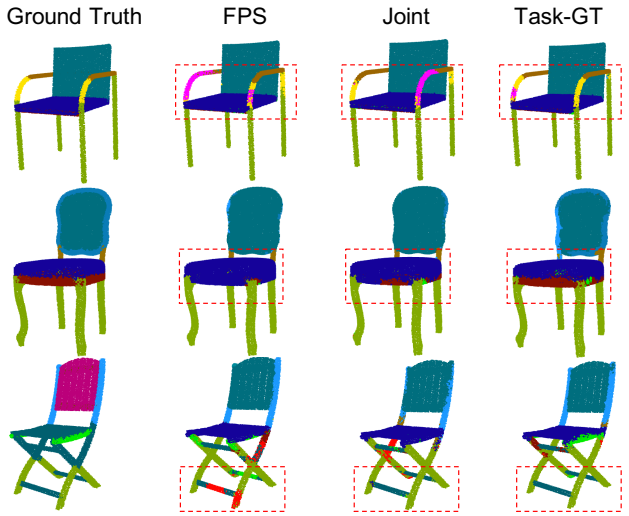

Fig. 8: Qualitative comparsion on PartNet [16] semantic segmentation. Segmentation results near boundary areas are more accurate in *Joint* and *Task-GT* than in *FPS*.

$\beta = 0.75$ in our experiments as the Task-GT sampling. Some examples are shown in Fig. 6. In addition to *Edge FPS*, an alternative sampling way is to sample points from each category point set respectively and more details can be referred to the comparison experiments in Section 5.

Implementation and results. We use the cross-entropy loss as the task loss and train for 150 epochs for every category with a batch size of 12, optimized by momentum gradient descent optimizer with a momentum of 0.98 and an initial learning rate of 0.01. EMD loss with initial points in Equation 3 is used for jointly training. In addition, λ in Equation 4 is set to 50 initially and decayed by 0.95 for every 20 epochs. As shown in TABLE 2, in PointNet++ [13] backbone, our supervised sampler learning strategy outperforms the baseline model 0.7, 1.1, 1.6 shape mIoU %, and 1.4, 1.1, 1.4 part-category mIoU % in three segmentation levels respectively. In some categories such as Lamp-3 and Dishwasher-2, joint training performs better than Task-GT sampling. From our point of view, the guidance from the task loss may be stronger than Task-GT sampling points in such cases. While in rare cases such as Hat-1 and Microwave-1, Task-GT sampler performs even worse than FPS. It is possible that the information gain at boundary areas cannot compensate for the information loss caused by nonuniform sampling, and we will make an in-depth analysis in our future works. In addition, we found that the conclusions with other backbones including PointConv [14] and FPConv [15] are consistent, as shown in TABLE 3. Visualization results are shown in Fig. 7 and Fig. 8.

4.2 Point Cloud Completion

Dataset and evaluation metrics. ShapeNetCore [46] is used for our training. We adopt train/validation/test split from [48] with eight categories (i.e., airplane, bench, car, chair, lamp, rifle, table, and watercraft) and 20k+ models in total. In data preparation, each object model is normalized into a unit cube and 8 partial point clouds are obtained by back-projecting the rendered depth maps from different viewpoints. We uniformly sample 8,192 points from the surface of the partial point cloud and 8,192 points from the complete point cloud for task supervision. The number of downsampling points are the same as in segmentation task in each layer. We repeat the training data until it exceeds 15,000. In the evaluation, we adopt Chamfer distance to evaluate the completion results on surface points.

Design of supervising sampling. We downsample 1,024 points from the complete point cloud for sampling supervision. Another alternative sampling way is to sample from skeleton points [49]. In our experiments, for faster training, we adopt random downsampling instead of FPS. Comparison of two supervision samplings are shown in Section 5 and we recommend directly sampling points from the complete point cloud for supervision since it is easy to obtain and has better performance. More visualization results are shown in Fig. 9.

Implementation and results. We train the networks for 60 epochs for every category with a batch size of 12 and the same optimizer as in semantic segmentation task. Differently, we use CD loss given in Equation 1 for shape supervision. λ is set to 0.5 and the joint loss is scaled by 10^2 for better convergence. As the experiments shown in TABLE 4, our method outperforms the baseline but still has a great gap with Task-GT sampler. We visualize the predicted sampling points in Fig. 11, since the distribution of full scan and partial scan is much different, it is more difficult for displacement learning than in segmentation task. Visualization results are shown in Fig. 10.

4.3 Keypoint Detection

Dataset and evaluation metrics. We conduct experiments on KeypointNet [22], a large-scale and diverse 3D keypoint dataset that contains 103,450 keypoints and 8,234 3D models from 16 categories. We compare different sampling strategies on four categories (i.e., bottle, chair, car, and table) with 2,048 input points. Following the data preparation of [22],

	Avg.	Bag	Bed	Bottle	Bowl	Chair	Clock	Dish.	Disp.	Door	Earp.	Fauc.	Hat
B^1	70.9/60.5	66.3/57.0	49.8/55.7	82.5/**45.4**	68.3/54.6	84.6/69.8	**64.6**/43.0	64.2/69.0	90.8/91.1	70.0/64.5	71.9/**65.4**	72.3/70.1	**62.6**/**68.5**
J^1	**71.6**/**61.9**	**67.5**/**57.5**	**56.1**/**56.9**	**83.4**/44.0	**69.0**/**55.9**	**85.1**/**70.0**	64.5/**43.1**	**67.8**/**72.9**	**91.7**/**92.1**	**71.9**/**65.8**	**73.2**/64.7	**74.3**/**70.7**	61.7/67.9
G^1	73.2/64.4	69.8/60.8	60.1/58.3	83.7/47.2	71.2/58.6	89.1/81.0	65.4/45.9	69.8/75.2	95.4/94.9	76.9/71.1	75.2/65.3	75.4/70.8	59.5/67.4
B^2	49.5/44.8	-	30.5/40.7	-	-	55.2/**45.6**	-	53.4/55.8	-	**57.9**/53.1	-	-	-
J^2	**50.6**/**45.7**	-	**33.6**/**42.7**	-	-	**55.4**/43.2	-	**55.2**/**58.8**	-	55.7/**53.6**	-	-	-
G^2	51.7/46.6	-	34.8/45.5	-	-	60.6/50.1	-	54.4/54.2	-	59.1/54.6	-	-	-
B^3	50.0/44.2	-	23.1/**39.4**	62.1/42.3	-	49.8/40.4	40.9/31.6	48.0/47.6	81.3/81.7	47.5/41.0	**51.0**/**43.6**	59.3/61.7	-
J^3	**51.9**/**45.6**	-	**24.4**/38.6	**63.5**/**43.4**	-	**51.0**/**41.1**	**41.0**/**32.1**	**51.7**/**54.3**	**81.9**/**82.2**	**48.3**/**41.5**	50.6/41.9	**62.6**/**62.9**	-
G^3	52.5/45.8	-	25.5/42.0	67.7/44.5	-	54.2/44.0	42.5/33.4	47.4/51.1	85.5/85.7	46.7/40.6	50.7/39.7	63.2/60.9	-

	Keyb.	Knife	Lamp	Laptop	Micro.	Mug	Refri.	Scis.	Stora.	Table	Trash.	Vase
B^1	67.8/67.5	63.0/53.4	37.5/28.6	**95.8**/95.2	**81.5**/**53.6**	84.3/79.0	49.5/**43.6**	**82.1**/79.4	60.8/**62.6**	85.8/23.4	60.4/65.5	77.7/63.9
J^1	**68.4**/**68.4**	**64.1**/**55.7**	**39.8**/**30.5**	95.7/**95.2**	79.5/53.5	**85.4**/**79.3**	**51.8**/43.0	81.7/**79.4**	**61.8**/62.2	**86.1**/**25.2**	**60.4**/**65.5**	**77.9**/**66.2**
G^1	70.1/70.1	67.6/58.6	37.2/29.3	96.3/95.5	79.4/53.8	83.0/79.7	49.4/43.6	85.3/83.6	67.4/73.6	90.2/25.7	59.0/66.3	81.1/68.2
B^2	-	-	33.7/**29.0**	-	60.0/**52.2**	-	54.8/44.4	-	52.6./56.3	47.5/25.7	-	-
J^2	-	-	**36.4**/28.9	-	**62.0**/51.5	-	**55.9**/**47.3**	-	**53.5**/**56.4**	**47.9**/**28.7**	-	-
G^2	-	-	35.1/28.2	-	62.3/54.1	-	56.0/48.2	-	53.9/57.8	49.0/27.0	-	-
B^3	-	41.6/32.5	29.3/21.1	-	52.7/48.3	-	46.5/40.5	-	51.2/**47.6**	42.1/25.4	46.7/**53.5**	76.5/53.7
J^3	-	**44.8**/**35.6**	**32.6**/**21.3**	-	**55.3**/**51.8**	-	**50.5**/**47.3**	-	**52.2**/47.3	**43.5**/**25.4**	**49.8**/53.3	**78.7**/**55.3**
G^3	-	48.3/41.7	30.2/21.3	-	54.1/48.0	-	51.4/43.8	-	52.2/48.7	44.1/25.4	50.5/53.4	78.6/55.0

TABLE 2: Semantic segmentation results (shape mIoU / part-category mIoU %) on PartNet. B, J, and G represent the baseline model, jointly trained sampler, and Task-GT sampling respectively. The superscripts (^{1,2,3}) indicate different segmentation levels. We do not show the results on category-levels that are not annotated in PartNet.

		Avg.	Chair-1	Chair-2	Chair-3	Bag-1	Door-1	Bottle-3
PN	Base.	64.7	84.6	55.2	49.8	66.3	70.0	62.1
	Joint	**65.7**	**85.1**	**55.4**	**51.0**	**67.5**	**71.9**	**63.5**
	T.G.	69.7	89.1	60.6	54.2	69.8	76.9	67.7
PC	Base.	64.3	84.2	53.4	49.2	67.0	70.3	61.7
	Joint	**66.5**	**84.8**	**56.2**	**51.1**	**68.7**	**74.3**	**64.1**
	T.G.	71.1	90.1	61.2	54.2	72.0	80.6	68.5
FP	Base.	65.3	84.7	54.4	49.8	70.2	74.0	58.6
	Joint	**67.8**	**85.6**	**57.0**	**52.0**	**71.8**	**77.9**	**62.2**
	T.G.	73.7	90.9	65.0	57.6	72.3	87.5	68.6

TABLE 3: Shape mIoU % on PartNet [16] semantic segmentation. We compare FPS (Base.), learnable sampling (Joint), and Task-GT sampling (T.G.) in different backbone networks, including PointNet++ (P++) [13], PointConv (PC) [14] and FPConv (FP) [15]. The numbers (1,2,3) indicate different segmentation levels.

(test/val) ↓	Baseline	Joint	Task-GT
Airplane*	0.586/0.337	**0.571**/**0.335**	0.455/0.332
Bench	0.144/0.136	**0.136**/**0.130**	0.085/0.084
Car	**0.132**/**0.121**	0.137/0.125	0.117/0.110
Chair	0.192/0.177	**0.184**/**0.171**	0.123/0.116
Lamp	**0.178**/0.148	0.186/**0.145**	0.103/0.085
Rifle*	0.351/0.372	**0.313**/**0.329**	0.337/0.347
Table	0.250/0.212	**0.229**/**0.201**	0.132/0.123
Watercraft	**0.113**/**0.105**	0.115/0.106	0.079/0.075
Avg.	0.138/0.121	**0.134**/**0.118**	0.090/0.083

TABLE 4: Quantitative comparisons on completion task. We compare charmfer distance ($\times 10^3$) between predicted points and complete point cloud on test/val dataset splits. * indicates $\times 10^4$. *Joint* indicates jointly training sampler with task and displacement loss, and *Task-GT* present the Task-GT sampling.

each object model is normalized into a unit sphere and we downsample 1,024, 256, 64, and 16 points for multi-scale learning. The training data is repeated until it exceeds 2,000. A keypoint is considered as being detected when there is at least one predicted keypoint within a given distance. We evaluate the average precision (AP) of all object models within a distance threshold of 0.05.

Design of supervising sampling. Firstly, since the number of labeled keypoints is only 10 to 20 in an object model, we regard these points that locate in 20-NN (Nearest Neighbors) of labeled keypoints as *pseudo keypoints*. Then similar to *Edge FPS*, we sample more points from *pseudo keypoints* and less from other areas. We set $\lambda = 7.5$ and $\beta = 0.8$ in our experiments. Examples are shown in Fig. 12.

Implementation and results. Models are trained for 100 epochs with a batch size of 6 and optimized by Adam optimizer with a learning rate of 0.01. We use BCE for binary classification and CD loss for displacement supervision. Since there are only 10∼20 keypoints in each object model, we scale the loss weight of positive labels by 10. λ is set to 0.5 in this task. Quantitative results are shown in TABLE 5, and our supervised sampler outperforms the baseline in all categories. Visualization results are shown in Fig. 13.

5 Ablation Study

In this section, we show several ablation studies on the sampler learning. Experiments are mainly conducted on the segmentation task of the Chair category in level 3 (PartNet [16]), since Chair-3 has enough training data (3k+) and part categories.

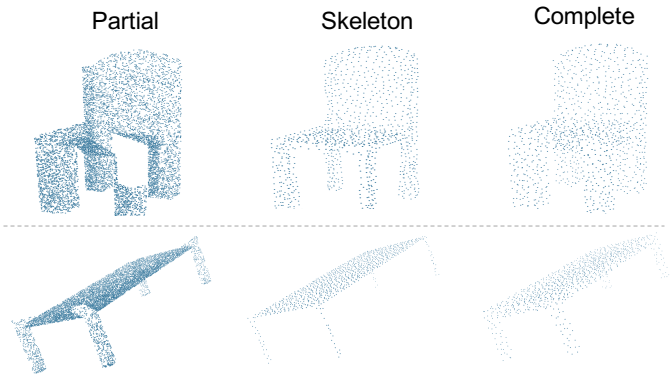

Fig. 9: Comparison of different Task-GT samplings on point cloud completion. We compare sampling points from skeleton points (*Skeleton*) [49] and the complete point cloud (*Complete*) for supervision.

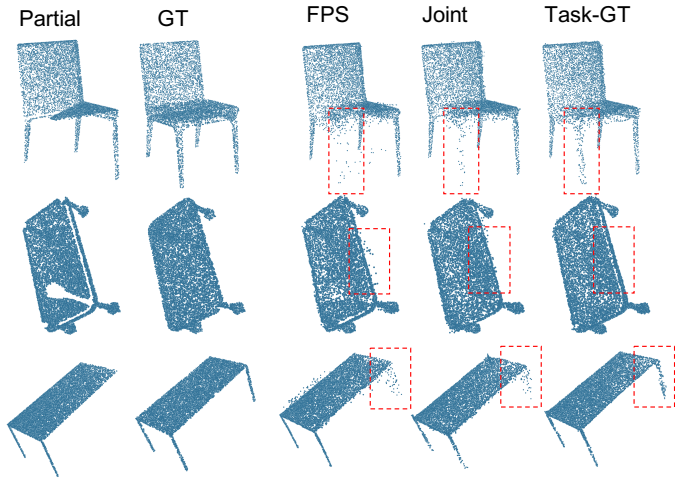

Fig. 10: Qualitative comparison on ShapeNet [46] point cloud completion. Introducing ground truth information to the sampling phase makes the completion more accurate and reduces outlier points.

	Avg.	Bottle	Chair	Car	Table
Baseline	67.1	68.5	57.8	66.2	75.8
Joint	**67.8**	**69.0**	**58.9**	**66.6**	**76.8**
Task-GT	70.4	74.8	59.3	68.9	78.5

TABLE 5: Quantitative comparisons on keypoint detection. We conduct experiments on four categories and compare average precision % with a distance threshold of 0.05 ($AP^{0.05}$).

Layer selection. We conduct ablative experiments on studying the effect of sampling in different downsampling layers. In TABLE 6, although adopting Task-GT sampling in more layers is better, cumulative sampler learning brings more random noise in the initial training stage, making it difficult for point displacement networks in the later layers to converge. On the other hand, the sampled points are extremely dense in the first layer and sparse in the last two layers, which results in receptive fields being either too small or too large. Thus, we conduct all other comparison experiments only performing supervision in the second layer.

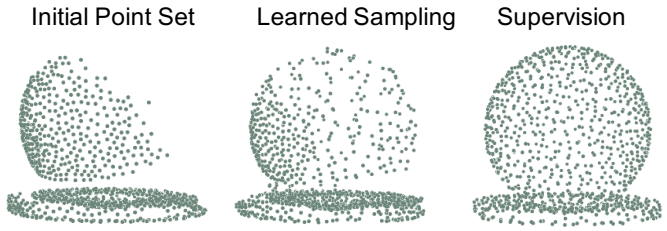

Fig. 11: Comparison of learned sampling with supervising sampling on point cloud completion task. Initial point set is sampled from the partial point cloud (input) by FPS and supervised by the downsampled complete point cloud (ground truth). we show that it is difficult to learn a point distribution that is too far away from the initial one.

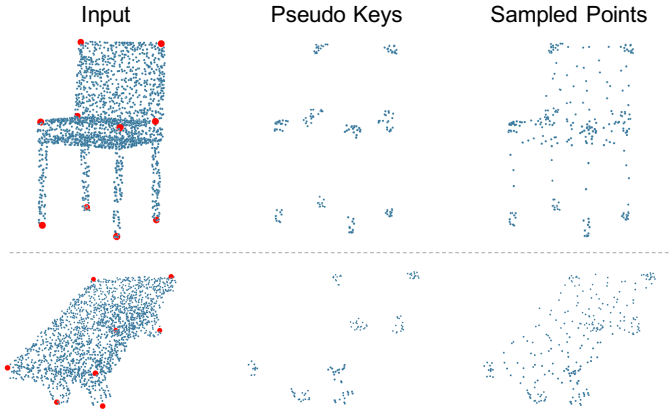

Fig. 12: Visual examples of Task-GT sampling on keypoint detection. Pseudo keys are the points near the labeled keypoints. Similar to *Edge FPSP*, points are sampled more densely from pseudo keys and sparsely from other areas.

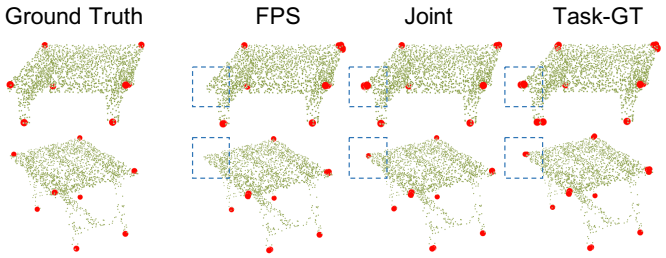

Fig. 13: Qualitative comparision on KeypointNet [22] keypoint detection. Points in red are labeled or predicted (with a classification threshold of 0.5) keypoints. Sampling more points near keypoints (*Joint* and *Task-GT*) helps the task network identify keypoints more correctly.

Training strategy. We compare different sampler training strategies in TABLE 7. Sampler trained only with task loss performs much worse than the baseline model. Because without the constraint of displacement loss, the sampled

	1	2	3	4	Shape	Part	oA
Task-GT		√			**54.2**	44.0	86.0
			√		51.1	42.3	83.8
		√	√		53.4	**45.0**	85.7
	√	√	√	√	53.3	44.2	**86.5**
Joint		√			**51.0**	**41.1**	**81.4**
		√	√	√	50.7	38.9	81.3

TABLE 6: Layer selection. We atop Task-GT sampling or jointly learned sampling in different downsampling layers. The evaluation results (shape, part-category mIoU and overall accuracy %) show that adopting such samplings in layer 2 only is the best.

	Shape mIoU	Part mIoU	oA
Baseline	49.8	40.4	81.0
Joint	**51.0**	**41.1**	**81.4**
Task Loss Only	39.7	32.1	78.6
w/o Task Loss	49.5	40.3	81.0
+ finetune	50.7	40.8	81.3

TABLE 7: Sampler training strategies. We compare the impact of two components of joint loss function, i.e., task loss and displacement loss. The last row indicates finetuning the sampler (trained without task loss) with joint loss.

	Loss	Shape mIoU	Part mIoU	oA
Baseline	-	49.8	40.4	81.0
FPS	CD	49.2	40.3	80.9
	EMD	50.3	40.6	81.2
Edge FPS	EMD	**51.0**	**41.1**	**81.4**
Part FPS	EMD	50.9	40.4	81.3

TABLE 8: Selection of Task-GT samplers for supervision in semantic segmentation. The results show that good supervision is important for sampler learning, and *Edge FPS* is not the only choice.

	Chair	**Airplane***	**Car**	**Bench**	**Lamp**
Baseline	0.177	0.337	0.121	0.136	0.148
Complete	**0.116**	0.332	**0.110**	**0.084**	**0.085**
Skeleton	0.127	**0.327**	0.118	0.089	0.108

TABLE 9: Selection of Task-GT samplers in point cloud completion. We randomly downsample 1,024 points from complete point cloud and skeleton point cloud for supervision. Chamfer Distance ($\times 10^3$) is used as the evaluation metric. $\star$ indicates $\times 10^4$.

points will distribute very far away from the original and cannot capture good shape details. On the other hand, training the sampler only with the displacement loss (i.e., without task loss) cannot bring improvement over the baseline model. However, if we further finetune the sampler with task loss, it can still achieve considerable improvement. In our view, Task-GT sampling is not easy to learn, while task loss can bring semantic hints to promote better sampler learning.

Sampling supervision. We refer to sampling more points in boundary areas as *Edge FPS*. Besides, we design two other supervisions, sampling points with FPS in each part separately (named *Part FPS*, with GT information) and uniformly sampling points in the original point cloud (named FPS, without GT information). As shown in TABLE 8, both *Edge FPS* and *Part FPS* bring considerable improvements over the baseline model, while FPS cannot. The possible reason is that supervised with FPS will bring similar sampling distribution, and the extracted features have trivial differences from directly using FPS. To prove that, we replaced the learned sampler (supervised by FPS points) with FPS, and only suffered 0.1% shape mIoU loss. Thus, we can conclude that effective supervision is important for sampler learning, and *Edge FPS* is not the only choice. We can also find the same conclusion in point cloud completion task from TABLE 9.

Initial point set. We compare different initial point sets used in displacement learning. We first introduce *predicted Edge FPS* by sampling more points in pseudo-boundary areas predicted by a boundary detection network and less in others. Then we adopt sampled points as the initial point set for displacement learning. As shown in TABLE 10, *predicted Edge FPS* as the initial point set outperforms the baseline model but still worse than using FPS.

Displacement learning. As shown in Fig. 5, there are two ways to obtain the sampled points by predicting coordinates directly or point-wise offsets to the initial point set. The experiment results are shown in TABLE 11. Directly regressing coordinates is more difficult than offset learning since the points may move too far away from the initial positions. Inspired by [12], we adopt soft projection on the predicted points and project onto the original point cloud to prevent too much distribution change. In most cases, we recommend using offset learning or coordinate learning with soft projection. Therefore, we adopt offset learning in other segmentation comparison experiments. However, for the case that the target distribution is far away from the original (full scan and partial scan in completion task), regressing coordinates can also be considered.

Displacement loss. In Section 3.2, we introduce two displacement loss functions for shape supervision, i.e., Chamfer distance (CD) loss, and earth mover's distance (EMD) loss. As shown in TABLE 12, training with EMD loss outperforms CD loss in semantic segmentation. The reason may be that we still have points sampled in other areas, though it is sparse, limiting the movement range of the initial points. That is, in bidirectional nearest point matching, each point can only be matched to a nearby point. Therefore, the learned distribution will not be far away from the initial. In other tasks, we also conduct comparison experiments and choose the best loss function.

Number of sampling points. We explore the impact of the number of sampling points on the performance of the second downsampling layer. As shown in TABLE 13, adopting the jointly learned sampler with fewer (896 and 768) sampling points could still achieve better results than sampling 1024 points by FPS.

	Shape mIoU	Part mIoU	oA
Baseline	49.8	40.4	81.0
FPS	**51.0**	**41.1**	**81.4**
Pred. Edge FPS	50.8	41.1	81.2

TABLE 10: Selection of initial point sets: FPS and predicted Edge FPS. We first train a network for boundary points prediction, then generate predicted Edge FPS by sampling more points in predicted boundary areas and less in others.

	Shape mIoU	Part mIoU	oA
Baseline	49.8	40.4	81.0
Offsets	**51.0**	**41.1**	**81.4**
Coords.	49.8	39.8	81.1
Coords. + Soft Proj.	50.8	41.0	81.3

TABLE 11: Displacement learning strategies: point-wise offset learning, coordinate learning and learning with soft projection [12].

	Loss	Shape	Part	oA
Baseline	-	49.8	40.4	81.0
Joint	CD	50.2	40.2	81.1
	EMD	**51.0**	**41.1**	**81.4**
	CD + EMD	50.8	40.5	81.3

TABLE 12: Selection of displacement loss in sampler learning: CD and EMD. CD + EMD indicates that the loss function is composed of CD and EMD loss. The proportions are 0.3 and 0.7, respectively.

	Num.	Shape mIoU	Part mIoU	oA
Baseline	1,024	49.8	40.4	81.0
Joint	1,024	**51.0**	**41.1**	**81.4**
	896	50.6	39.8	81.1
	768	50.1	39.3	80.9

TABLE 13: Number of sampling points. We train the model with a learnable sampler that samples less than 1,024 points (i.e., 896 and 768) in the second downsampling layer.

6 Discussion

6.1 Comparison with SampleNet [12]

Although both SampleNet [12] and this work are studying point sampling, they are designed for different application areas and have different learning and supervising ways. We elaborate on their differences in detail from three aspects.

(a) They are designed based on different perspectives of sampling. We would first emphasize there are fundamental differences between the purposes of our algorithm and SampleNet [12]. The goal of sampling in [12] is to sample a sparse subset that can be processed more easily, to represent the original point cloud, and maintain the similar performance of downstream tasks. While ours is to sample an intermediate point set for neighbor point grouping and feature aggregation to boost the overall task performance.

(b) Direct application of SampleNet [12] does not work well. We have tried to directly adopt the sampler learning strategy and module of SampleNet [12] on point-wise analysis networks (i.e., supervised by the original point cloud) while it brings no improvements. This is because we found that the learned sampled points are still distributed uniformly and almost have no difference from FPS. Thus we propose to supervise the sampler with a task-related sampling to learn a different local point distribution.

(c) The design of the point displacement module is different. [12] generates a set of sampled points using the global feature which is extracted from the original point cloud. In our method, we select an initial point set and predict offsets or coordinates with point-wise features for each initial point.

6.2 Sampling Supervision

We find that the improvement of jointly learned sampling is limited in some categories of segmentation task. We have experimented with several methods to learn sampled points, but sometimes the learned distribution cannot capture all local details perfectly. There are two possible reasons, (1) we think a deeper network can help improve the sampler learning with sufficient GPU memory; (2) Chamfer Distance (CD) and Earth Mover's Distance (EMD) loss are not good measurement for local point distribution. These would be left as directions of our future work.

Our method indeed requires a pre-design of sampling supervision. However, for the tasks of semantic segmentation, keypoint detection and point cloud completion as reported above, we actually did not put much efforts into the design of the Task-GT samplers. Most importantly, our experiments show that sampling is not trivial and better samplers bring considerable improvements in point-wise analysis tasks. Although the design of Task-GT sampler is heuristic, we hope that the design and learning of samplers can become one of the future research directions of 3D point clouds.

7 Conclusion

In this work, we explore the impact of sampling in point cloud analysis based on deep neural networks. Our observation is that uniform sampling like FPS is not always the optimal choice. We conduct analytical experiments and demonstrate that by introducing task-related information into the sampling process, the network can be trained more effectively for point-wise feature learning. We further present a novel pipeline for supervised sampler learning and achieve considerable performance than FPS in various tasks with different backbones. Moreover, Task-GT samplers are manually designed and might not be the best. These bring two potential future research problems in point-wise analysis: "What is the best sampling?" and "How to further learn a better sampler?".

Acknowledgments

This work was supported in part by the Key Area R&D Program of Guangdong Province with grant No. 2018B030338001, by the National Key R&D Program of

China with grant No. 2018YFB1800800, by Shenzhen Outstanding Talents Training Fund, and by Guangdong Research Project No. 2017ZT07X152.

REFERENCES

[1] A. Pomares, J. L. Martínez, A. Mandow, M. A. Martínez, M. Morán, and J. Morales, "Ground extraction from 3d lidar point clouds with the classification learner app," in *2018 26th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2018, pp. 1–9.

[2] X. Yue, B. Wu, S. A. Seshia, K. Keutzer, and A. L. Sangiovanni-Vincentelli, "A lidar point cloud generator: from a virtual world to autonomous driving," in *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*. ACM, 2018, pp. 458–464.

[3] A. Behl, O. Hosseini Jafari, S. Karthik Mustikovela, H. Abu Alhaija, C. Rother, and A. Geiger, "Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios?" in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2574–2583.

[4] J. Rambach, A. Pagani, and D. Stricker, "[poster] augmented things: Enhancing ar applications leveraging the internet of things and universal 3d object tracking," in *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*. IEEE, 2017, pp. 103–108.

[5] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," 2015.

[6] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 922–928.

[7] C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," 2016.

[8] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1305–1315, 1997.

[9] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6411–6420.

[10] B. Graham, M. Engelcke, and L. van der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," *CVPR*, 2018.

[11] O. Dovrat, I. Lang, and S. Avidan, "Learning to sample," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2760–2769, 2019.

[12] I. Lang, A. Manor, and S. Avidan, "Samplenet: Differentiable point cloud sampling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7578–7588.

[13] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108.

[14] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9621–9630.

[15] Y. Lin, Z. Yan, H. Huang, D. Du, L. Liu, S. Cui, and X. Han, "Fpconv: Learning local flattening for point convolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4293–4302.

[16] F. Yu, K. Liu, Y. Zhang, C. Zhu, and K. Xu, "Partnet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9491–9500.

[17] Z. Liu, H. Hu, Y. Cao, Z. Zhang, and X. Tong, "A closer look at local aggregation operators in point cloud analysis," *ECCV*, 2020.

[18] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (TOG)*, 2019.

[19] Y. Nie, Y. Lin, X. Han, S. Guo, J. Chang, S. Cui, J. Zhang *et al.*, "Skeleton-bridged point completion: From global inference to local adjustment," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[20] J. Chibane, T. Alldieck, and G. Pons-Moll, "Implicit functions in feature space for 3d shape reconstruction and completion," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[21] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[22] Y. You, Y. Lou, C. Li, Z. Cheng, L. Li, L. Ma, C. Lu, and W. Wang, "Keypointnet: A large-scale 3d keypoint dataset aggregated from numerous human annotations," *arXiv preprint arXiv:2002.12687*, 2020.

[23] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," *arXiv preprint arXiv:1706.01307*, 2017.

[24] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3075–3084.

[25] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[26] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 984–993.

[27] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "Spidercnn: Deep learning on point sets with parameterized convolutional filters," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 87–102.

[28] F. Groh, P. Wieschollek, and H. P. Lensch, "Flex-convolution," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 105–122.

[29] M. Atzmon, H. Maron, and Y. Lipman, "Point convolutional neural networks by extension operators," *arXiv preprint arXiv:1803.10091*, 2018.

[30] Q. Xu, X. Sun, C.-Y. Wu, P. Wang, and U. Neumann, "Grid-gcn for fast and scalable point cloud learning," 2020.

[31] C. Wang, B. Samari, and K. Siddiqi, "Local spectral graph convolution for point set feature learning," *arXiv preprint arXiv:1803.05827*, 2018.

[32] G. Te, W. Hu, Z. Guo, and A. Zheng, "Rgcnn: Regularized graph cnn for point cloud segmentation," 2018.

[33] G. Li, M. Müller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[34] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, "Tangent convolutions for dense prediction in 3d," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3887–3896.

[35] J. Huang, H. Zhang, L. Yi, T. Funkhouser, M. Nießner, and L. J. Guibas, "Texturenet: Consistent local parametrizations for learning from high-resolution signals on meshes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4440–4449.

[36] Y. Yang, S. Liu, H. Pan, Y. Liu, and X. Tong, "Pfcnn: Convolutional neural networks on 3d surfaces using parallel frames," 2020.

[37] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[38] K. Yin, H. Huang, D. Cohen-Or, and H. Zhang, "P2p-net: Bidirectional point displacement net for shape transform," *ACM Transactions on Graphics(Special Issue of SIGGRAPH)*, vol. 37, no. 4, pp. 152:1–152:13, 2018.

[39] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," *arXiv preprint arXiv:1711.08488*, 2017.

[40] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[41] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[42] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[43] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, "Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[44] Z. Huang, Y. Yu, J. Xu, F. Ni, and X. Le, "Pf-net: Point fractal network for 3d point cloud completion," 2020.
[45] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *2018 International Conference on 3D Vision (3DV)*, 2018, pp. 728–737.
[46] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
[47] M. Loizou, M. Averkiou, and E. Kalogerakis, "Learning part boundaries from 3d point clouds," in *Computer Graphics Forum*, vol. 39, no. 5. Wiley Online Library, 2020, pp. 183–195.
[48] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, "A scalable active framework for region annotation in 3d shape collections," *SIGGRAPH Asia*, 2016.
[49] J. Tang, X. Han, J. Pan, K. Jia, and X. Tong, "A skeleton-bridged deep learning approach for generating meshes of complex topologies from single rgb images," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.