On-the-Fly Object-aware Representative Point Selection in Point Cloud

Xiaoyu Zhang
RMIT University
s3500791@student.rmit.edu.au

Ziwei Wang
CSIRO
ziwei.wang@data61.csiro.au

Hai Dong
RMIT University
hai.dong@rmit.edu.au

Zhifeng Bao
RMIT University
zhifeng.bao@rmit.edu.au

Jiajun Liu
CSIRO
jiajun.liu@csiro.au

Abstract

Point clouds are essential for object modeling and play a critical role in assisting driving tasks for autonomous vehicles (AVs). However, the significant volume of data generated by AVs creates challenges for storage, bandwidth, and processing cost. To tackle these challenges, we propose a representative point selection framework for point cloud downsampling, which preserves critical object-related information while effectively filtering out irrelevant background points. Our method involves two steps: (1) Object Presence Detection, where we introduce an unsupervised density peak-based classifier and a supervised Naïve Bayes classifier to handle diverse scenarios, and (2) Sampling Budget Allocation, where we propose a strategy that selects object-relevant points while maintaining a high retention rate of object information. Extensive experiments on the KITTI and nuScenes datasets demonstrate that our method consistently outperforms state-of-the-art baselines in both efficiency and effectiveness across varying sampling rates. As a model-agnostic solution, our approach integrates seamlessly with diverse downstream models, making it a valuable and scalable addition to the 3D point cloud downsampling toolkit for AV applications.

1. Introduction

In the domain of autonomous vehicle (AV) applications, point clouds play a crucial role in modeling the environments. Real-time analysis of these point clouds enables vehicles to perceive their surroundings and supports safe driving by facilitating key perception tasks such as object detection, semantic segmentation, and motion prediction [5]. However, processing the point cloud data collected by AVs pose a significant challenge due to two primary reason: the size of individual point clouds is large, and the high frequency at which they are generated. Specifically, a typical point cloud representing a single scene comprises 20,000 to 30,000 points [16], and these are captured at high frequencies, such as 10Hz for datasets like KITTI [2] and 2Hz for nuScenes [3].

Motivation. The substantial data volume generated by AVs would cause burdens on infrastructures, including the storage, network, and computation resources: **1)** Storage and Network Constraints. The AV data sometimes would be collected and uploaded to a remote cloud for further process, e.g., to fine-tune the existing models. AVs generate approximately 5 TB of point cloud data per hour, a volume that tests the limits of current 5G network capabilities, which can optimally handle up to 4.5 TB per hour [13]. This unmatching necessitate a reduction in data size before transmission to remote servers for further processing. **2)** Hardware Limitations in Data Processing. The scale of point cloud data imposes high demands on computational resources, particularly during the training of machine learning models. High-performance GPUs, although effective, represent a significant expense [12]. Thus, we need to find an alternative approach that reduces the data size to alleviate the need for hardware upgrades, thereby enabling more cost-efficient data processing.

Limitations and challenges. Existing point cloud downsampling strategies main have three limitations: **1)** Scalability: Deep learning-based methods, such as those proposed in [6, 15], are primarily designed for single-object point clouds and face significant challenges when extended to large-scale outdoor point clouds. This limitation arises from the inherent complexity and uneven point distribution characteristic of outdoor environments. **2)** Generalization capability. There are studies that integrates the point cloud subsampling as a module of the object detection model. Such a module is trained end-to-end [10, 32]. The sampling result is downstream model specific and cannot be generalize to others models. **3)** Processing efficiency. Deep learning-based downsampling methods rely on feature extraction,

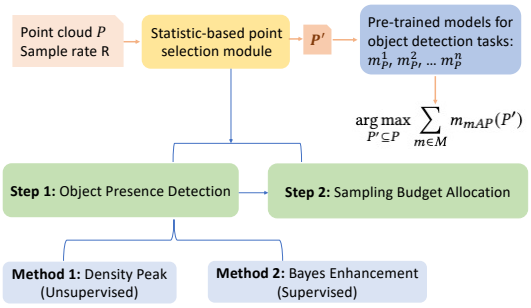


Figure 1. Overview

which often becomes a bottleneck for computational efficiency. On the other hand, traditional non-learning-based solution, such as Farthest Point Sampling (FPS), suffer from high computational complexity of $\theta(n^2)$, making it unsuitable for real-time applications.

Problem Formulation. We propose an object-aware representative point selection approach, which aims to best describe the main content of the point cloud by selecting a subset of points. The subset of points is expected to generalize to various detection models. As demonstrated in Figure 1, we achieve the goal by subsampling. First, we classify points in a point cloud into two categories: important object points and insignificant background points. To achieve this, we propose an unsupervised method that utilizes density peaks as classification features. Furthermore, we propose a supervised Naïve Bayes enhanced classifier to improve the classification performance. Second, we propose an imbalanced sample budget allocation with different sampling granularity for the two categories of points, which guarantees both effectiveness and efficiency.

To summarise, we make the following contributions:

1) By identifying the challenge posed by the significant data size of point cloud from autonomous vehicles, we define our point selection problem, which aims at finding a representative point set in a point cloud and thereby decrease the overall data size (Section 2.1).

2) We propose a two-step sampling method, including classifying object points and background points by detecting the presence of the object and allocating different sampling budgets to object points and background points (Section 3).

3) To detect the presence of an object, we propose an unsupervised density peak classifier (in Section 3.2), and an supervised method with Naïve Bayes enhancement (in Section 3.3) to further improve the effectiveness. The two selection methods also provide options to suit supervised and unsupervised application scenarios.

4) The experimental results demonstrate that our methods exhibit significant advantages in both effectiveness and efficiency, achieving a tenfold increase in speed. Also, our approach is compatible with various downstream models.

2. Problem Definition and Related Work

In this section, we provide a formal problem definition and review its related work.

2.1. Problem Definition

Given a point cloud P, where each point in P is formed as a d-dimension feature vector $p_i \in \mathbb{R}^d$, a set of pretrained machine learning models M that perform object detection as the downstream task, and a sample rate r, we aim to find a method $F(P, r)$ to obtain a subset $P^* \subseteq P$ (where $|P^*| = |P| \times r$) as the representative point set. The objective is to maximize the evaluation result of the object detection task, measured using mean Average Precision (mAP), by selecting P^* for $m \in M$:

$$P^* = \arg\max_{P' \subseteq P} \sum_{m \in M} m_{mAP}(P') \tag{1}$$

2.2. Related Work

Important Points Selection. To achieve the goal of selecting the representative points from a point cloud, deep learning-based methods [6, 15] have been proposed. However, both methods have the limitation that they are designed for single-object point clouds and may struggle to extend to large-scale point clouds for outdoor scenes. Those methods use PointNet [18] or PointNet++ [19] as the backbone, which can hardly be extended to the outdoor point cloud feature extraction due to significant computational and memory requirements as mentioned in [35]. It is limited by both the complexity and scale, and the training efficiency for the outdoor point clouds. Another stream of studies integrates the point cloud subsampling as part of the deep learning module, which is trained end-to-end [10, 32]. Although such methods can handle large-scale outdoor point clouds, they have limited capability to achieve our desired feature of being compatible with different downstream models. The reason is that the conventional point selection methods inevitably have a bias towards selecting points based on the design of downstream model [9]. For example, in [32], foreground points are deemed more important for the downstream model, leading the point selection module to strategically pick those points, which has no guarantee of performance when the downstream model changes. In [27], a regional-based point selection using active learning is proposed. Although this study focuses on point cloud labeling, it conveys an essential idea that different objects in a scene may not require uniform point density for perception. In their experiments on point clouds for indoor scenes, it is obvious that removing the point of the floor has a very limited effect on the performance of the trained model.

Point Cloud Feature Extraction. Learning from point clouds involves extracting richer information from the

(a) Original (b) Random Sampling (c) Object-aware Sampling

Figure 2. Demo of different sampling strategy with 50% sample rate

points, thereby facilitating downstream tasks. Deep learning has been widely used in extracting features from point clouds, with methods broadly categorized into point-based and voxel-based approaches. With pioneer works including [18, 19, 26] utilize the convolutional network to points and break the ice for point-based solutions, the later studies provide more elegant solutions by incorporating more point features [25, 28, 33]. Compared to the point-based methods, the voxel-based methods are more efficient and computationally friendly [17, 24, 36], which can be easily extended to the large-scale autonomous driving-related dataset [2, 3, 23]. Recently, richer resources, such as camera images and radar data, have been introduced to facilitate feature learning. The most recent state-of-the-art studies are utilizing multiple resources to enhance point cloud perceptions [29–31].

Compression. In the context of our goal to reduce data size, compression is a way to achieve it. From the literature, the compression can be divided into two categories: Octree-based methods [11, 20, 21] and Range Image-based methods [34]. However, resorting to compressed data inevitably leads to the necessity for decompression. The uncompressed data maintain the original size and does not reduce the hardware demands for model training. Therefore, compression techniques do not align directly with our research focus, presenting a divergent path rather than a directly comparable methodology.

3. Methodology

In this section, we first demonstrate a case study to explain the effectiveness of our Object Presence Detection based solution design (Section 3.1). Then, we formally introduce our solution. As shown in Figure 1, our solution has two major steps. The first step is Object Presence Detection, which aims to develop a binary classifier to discern object points from background points. We propose two alternative solutions: the supervised density peak-based method (Section 3.2) and the supervised Bayes enhanced method (Section 3.3). The second step is Sample Budget Allocation, with which we allocate sampling budgets to prioritize the object point set over the background point set (Section 3.4).

3.1. Case Study

In this case study, we use an example to illustrate the motivation of our solution design and its effectiveness. Intuitively, in a complete point cloud, the points of the objects (e.g., pedestrians, vehicles) hold more significance than the points of the background (e.g., ground and buildings).

In our case study, we use two different sample strategies: (1) Random Sampling. (2) Object-aware Sampling, which is prone to retain more object points than the background points. We demonstrate the result of 50% sample rate.

Visualization Result. Figure 2 visualizes the result of different sample strategies. According to the visualization, compared with the original point cloud (in Figure 2a), it is evident that Object-aware Sampling (in Figure 2c) provides a better depiction of the objects, whereas random sampling (in Figure 2b) tends to blur the entire point cloud.

Experimental Results. We validate the effectiveness of the "Object-aware Sampling" strategy through simple experimental results. Table 1 illustrated the inference results with the full point cloud and the two aforementioned sampling strategies (conducted with Part-A2 [22] as the inference model). The results indicate that the mAP significantly drops when employing the random sampling strategy compared to the original full point cloud. However, with the "Object-awre Sampling", although there is a drop in mAP, it remains within an acceptable range, especially considering the halving of the number of points.

Table 1. Eval mAP of various sample strategies on KITTI

Strategy	Easy	Med	Hard
Original	77.84	64.45	60.71
Random	70.79	56.57	53.09
Keep Object	76.07	63.00	59.27

3.2. Object Presence Detection with Density Peak

In this section, we introduce an efficient statistics-based object presence detection approach, to achieve our goal of classifying points into two categories: object points and background points. Essentially, this method provides the swift identification of object presence while eliminating the need for training.

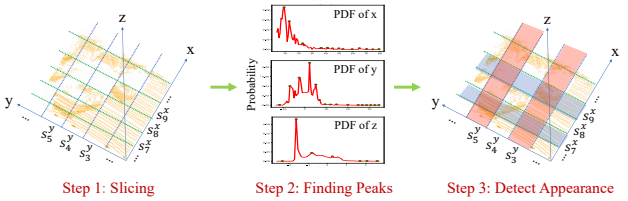


Figure 3. Demonstration of Density Peaks based classification

With the 3D features of the point cloud, we can obtain information from statistics of x, y, and z axes. We utilize the density peak as the statistical feature from the three axes for detecting the object presence, as demonstrated in Figure 3. To elaborate, our method mainly comprises three steps:

Step 1: Slicing. The purpose of slicing is to get initial statistics from the point cloud for x, y, and z axes respectively. Given a point cloud P that contains a set of points $\{p_1, p_2, .., p_{|P|}\} \in P$. Each point has three coordinates, i.e., $(p_i.x, p_i.y, p_i.z)$, to represent its location in the point cloud. We evenly slice P along the X-axis and Y-axis into m and n slices respectively. We use s_i^x to denote the i_{th} slice in the x direction. Thus, after slicing, we will have $s_1^x, s_2^x, ..., s_m^x$ by slicing the X-axis, and we will get $s_1^y, s_2^y, ..., s_n^y$ by slicing the Y-axis (as shown in Figure 3 Step 1).

It is worth noting that we do not perform the slicing along the Z-axis because it lacks sufficient features that contribute to Step 2, and it may negatively impact the completeness of the object. For details on how to determine the number of slices expected for each axis, please refer to "Data-driven Parameter Study" in our technical report [1].

Step 2: Finding Local Peaks. Since the point cloud uses points to formulate objects, the appearance of objects inevitably results in a surge in the number of points along both the X-axis and Y-axis. By leveraging this property, we can identify the slices that are more likely to contain objects.

Within each slice s, we can calculate the number of points it contains, thereby determining the point density of the corresponding slice. This point density is a crucial feature we use to detect the presence of objects. Considering the points tend to be sparse in distance places, instead of identifying the overall peak, we focus on detecting the local peaks of point density (as shown in Figure 3 Step 2). By specifying a stride, we iterate through the data and find the peak within the stride as the local peak. The detailed algorithm can be found in "Local Peak Search Algorithm" of our technical report [1].

Step 3: Detecting Object Presence. From the previous step, we identify the slices that possibly contain objects along both the X-axis and Y-axis. To pinpoint the presence of the object, we examine the intersections of the slices along the X-axis and Y-axis. We designate the points within these intersections as significant, while considering the remaining points as less significant background points.

We use an example to illustrate object presence detection, as shown in Figure 3. In Step 2, peaks were identified in slices s_7^x, s_9^x along the X-axis and s_3^y, s_5^y along the Y-axis. Subsequently, in step 3, we categorize the points falling within the intersections of (s_7^x, s_3^y), (s_7^x, s_5^y), (s_9^x, s_3^y) and (s_9^x, s_5^y) as object points, while the remaining points are considered background points.

Special Filter: Z-axis. The Z-axis filter plays a unique role within the filtering process. While the Z-axis is relatively less sensitive to object appearances, it has a distinctive ability to effectively identify a critical subset of background points—the ground. Despite the points for the ground has limited contribution to object detection, it often constitutes a substantial portion of a point cloud. By analyzing the Z-axis values, we can identify a global peak, which can be confidently classified as background points.

3.3. Object Presence Detection with Naïve Bayes

In Section 3.2, we introduce the unsupervised density peak-based detection approach. This method is superior in efficiency but at the cost of effectiveness. Therefore, in this section, we propose an effective supervised solution by leveraging the Naïve Bayes to improve the learning of point density while improve in performance. Using the training data, we build the Bayes model for each cross-region to predict the probability of objects appearing in that specific region.

3.3.1 Naïve Bayes Based Binary Classification

The Naïve Bayes-based binary classification is extended from the previous density peak-based method in Section 3.2 by incorporating training data to develop a Naïve Bayes model. It enables a more precise assessment of object presence. The Naïve Bayes-based method comprises two fundamental steps as follows:

Step 1. Slicing and Crossing. We use the same slicing strategy as introduced in Section 3.2, which slices a point cloud Pinto m slices along X-axis and n slices along Y-axis. However, unlike dynamically estimating m and n for each point cloud in the peak-based method, with the Naïve Bayes-based method, we fix m and n for all point clouds $P \in \mathcal{P}$. By crossing slices, we get $m \times n$ regions.

Step 2. Building Naïve Bayes Models for classification. With the aforementioned steps, the point cloud P is divided into $m \times n$ regions. Each region results from the intersection of slice s_i^x and slice s_j^y, which is denoted as $r_{(s_i^x, s_j^y)}$. For each region, we use Naïve Bayes to train a model over the training set $\mathcal{P}_{train}$. For example, in region $r_{(s_i^x, s_j^y)}$, given its point density in s_i^x, s_j^y (which is denoted by $r.d_x$ and $r.d_y$), we use equation 2 to train the corresponding Bayesian model. In total, we would have $m \times n$ Bayesian models. It is

worth noting that, this is a binary classification task, intending to categorise points into object point set or background point set. To prepare the training data and the detailed explanation of the equation, please refer to " Prepare Training Data for Naïve Bayes" and "Naïve Bayes Explaination" of our technical report [1].

$$P(y_r^{obj}|r.d_x, r.d_y) = \frac{P(y_r^{obj}) \times P(r.d_x, r.d_y|y_r^{obj})}{P(r.d_x, r.d_y)} \quad (2)$$

3.3.2 Naïve Bayes Models as Filter

To achieve the goal of differentiating object points from background points, we utilize our trained $m \times n$ Naïve Bayes model to filter out background points with higher precision. Compared with the density peak-based method which is very efficient, the Naïve Bayes enhanced detector is slower, because using the model for prediction is time-consuming. To ensure both effectiveness and efficiency, we create a staged filter:

a) Coarse-grained Filtering Stage. As objects in the point cloud for a scene are sparsely distributed, there is a high probability that many regions would never have objects present. Such regions lack positive examples for the density of objects appearing. Therefore, we can confidently predict a high likelihood of these regions being environmental regions, even without invoking the model for predictions.

b) Fine-grained Filtering Stage. Built upon the efficiency gains achieved through the coarse-grained filter, this step focuses on the remaining regions. We leverage the previously mentioned Naïve Bayes models to predict the categorization outcomes of the remaining regions.

3.4. Sampling

To ensure both efficiency and effectiveness, we perform the sampling with strategies including imbalanced sample budget allocation and samples with different granularity. Essentially, the object points hold more significance compared to the background points. Thus, we retain more object points and sampled with finer granularity, while allocating less budget to background points and sampled with coarse granularity to accelerate the efficiency. The detailed implementation is as follows:

a) Sampling Budget Allocation. Given an overall sampling rate, we can calculate the anticipated amount of points to be sampled. With this budget, our strategy involves a deliberate allocation of more budget to object points, while reserving a smaller portion to select background points.

b) Sampling with Different Granularity. Given the relatively lower significance of the environmental point set, we use the efficient random sampling strategy to select points from this set. For the object point set, we will perform region-based random sampling. Specifically, we execute random sampling within each distinct region containing objects instead of an overall random sampling over the whole set. Although this may sacrifice some efficiency, the object points are preserved to the utmost extent possible.

4. Experiments

To evaluate the performance of the proposed approach, we conducted experiments on both KITTI and nuScenes datasets with different inference models. The effectiveness and efficiency of our sampling strategy are evaluated, with an additional ablation study conducted.

4.1. Experiment Setup

All experiments were conducted on a server running Ubuntu 22.04.1, with Intel(R) Xeon(R) CPU E5-2697A v4 @ 2.60GHz and RTX2070. The code was implemented with the MMDetection3D Framework [4], and can be found in the git repository [1]. We conducted experiments to investigate the following research questions:

Q1. How is the effectiveness of our proposed two methods compared with baselines? (Section 4.2.1).

Q2. What advantage does sta_bayes have compared to sta_peak? (Section 4.2.2 (a)) How is the generalization ability of our methods? (Section 4.2.2 (b)).

Q3. What efficiency advantage does our proposed method have compared with the baselines? (Section 4.2.3)

Q4. What benefits do we obtain from different sampling budget allocations? How much does the Z-axis filter contribute to the sampling results? (Section 4.2.4)

Q5. How do our methods compare to the deep learning approach? (See "Comparison with Trained MLP (Q5)" in the technical report [1])

4.1.1 Datasets and Inference Models

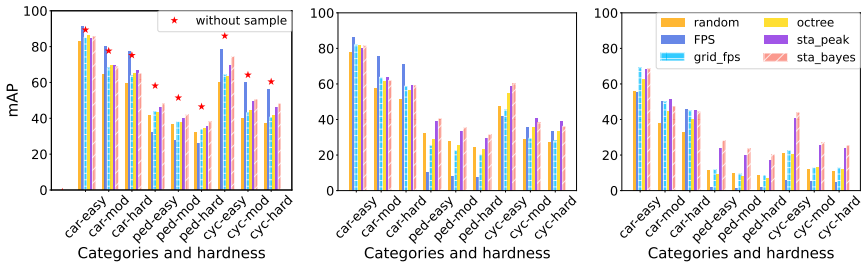
The experiment was conducted with KITTI dataset [2], and we selected Part-A2 [22] and SECOND [30] as our target model for the downstream object detection task. We also conducted experiments on nuScenes [3] and used PointPillars [14] as the inference model.

4.1.2 Methods for Comparison

1) Random sampling. This method involves randomly selecting points from the provided point cloud.

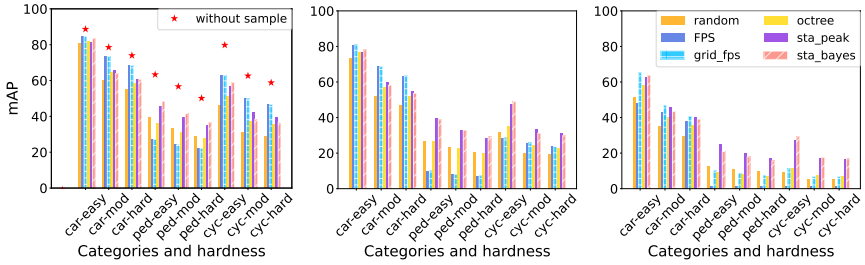
2) Farthest Point Sampling (FPS). FPS is implemented by selecting a subset of points that are maximally distant from one another.

3) FPS with Regular grid sampling (grid_fps). We developed this baseline to address FPS limitations. It involves dividing the point cloud into grids and then applying FPS to select points from each grid cell.



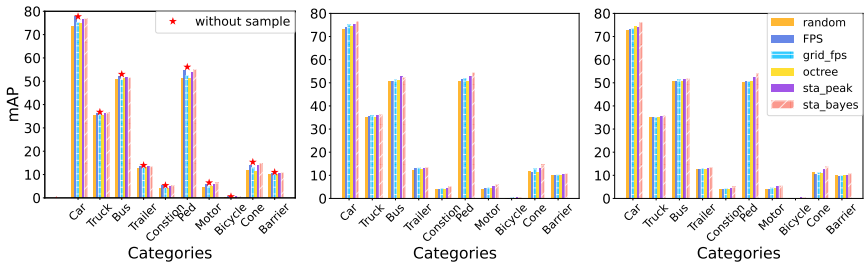
(a) Sample Rate 30% (b) Sample Rate 20% (c) Sample Rate 10%

Figure 4. Categorical mAP with Part-A2 on KITTI



(a) Sample Rate 30% (b) Sample Rate 20% (c) Sample Rate 10%

Figure 5. Categorical mAP with SECOND on KITTI



(a) Sample Rate 10% (b) Sample Rate 8% (c) Sample Rate 6%

Figure 6. Categorical mAP with PointPillars on nuScenes

4) Octree sampling. Octree sampling employs an Octree structure to partition space. The sampling technique is based on the approach in [7].

5) Statistics with Density Peak (sta_peak). The proposed method described in Section 3.2.

6) Statistics with Naive Bayes (sta_bayes). The proposed method described in Section 3.3.

4.1.3 Evaluation Metrics

Effectiveness study. We use the Mean Average Precision (mAP) to evaluate the extent our sampling method impacts the result of object detection [8].

Efficiency study. We measure the running time required to sample one point cloud.

4.1.4 Parameter Settings

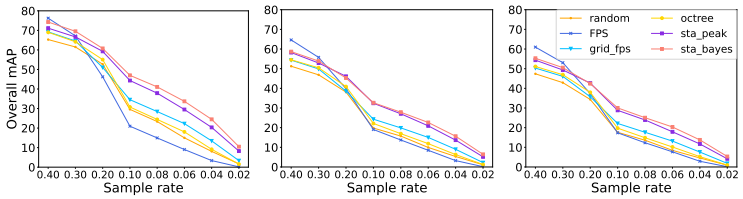
Determine the lower bound of the sample rate. The resolution of point clouds varies across datasets. To ensure method accuracy, determining an appropriate sample rate becomes crucial. This value was derived through the statistics from the dataset, with 10% for KITTI and 7% for nuScenes. Please refer to "Lower Bound of Sample Rate" in our technical report [1] for details.

Default Configurations. For all our demonstrated experiments, we use KITTI with Part-A2 as the inference model at sample rate 30%.

4.2. Experiment Results

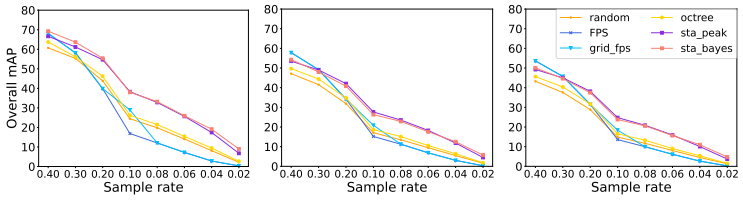
4.2.1 Effectiveness Study (Q1)

To evaluate the effectiveness of the proposed statistical sampling method, we conducted experiments across a range of



(a) Easy (b) Mod (c) Hard

Figure 7. Macro mAP with KITTI (Part-A2)



(a) Easy (SECOND) (b) Mod (SECOND) (c) Hard (SECOND)

Figure 8. Macro mAP with KITTI (SECOND)

sampling rates, ranging from 30% to 10%. To have a comprehensive view of the methods, we analyzed categorical mAP (along with instance recall rate), and macro mAP. As introduced in Section 4.1.4, the sampling rate lower bound for KITTI and nuScenes was 10% and 7%, respectively. We demonstrated our experiments from 30% to 10% for KITTI, and from 10% to 6% for nuScenes. The more experiment results with a wider range of sample rates and the corresponding evaluation result on Instance Recall Rate can be found in "Results on More Sample Rates" and "Study of Instance Recall Rate" of our technical report [1].

a) Comparison with Baselines under Categorical mAP. For better demonstration, our analysis was conducted using Part-A2 on KITTI (Figure 4). Here are our observations:

First, our proposed two methods, statistic_peak (denoted as sta_peak) and statistic_bayes (denoted as sta_bayes), both distinctively outperformed other baselines, except for FPS with a high sampling rate 30%, as shown in Figure 4a. The advantage became more obvious in lower sampling rates (in Figure 4b-4c).

Second, at a high sampling rate, i.e., 30%, FPS showed exceptional performance, especially in categories 'car' and 'cyclist', as shown in Figure 4a. FPS effectively preserved point diversity, rendering an excellent performance on a high sampling rate and on larger objects. However, the performance experienced a steep decline when the sampling rate decreased ($\leq$20%) or with small objects, e.g. pedestrian (ped), as shown in Figure 4b-4c.

Third, grid_fps, as a baseline we proposed, was designed to mitigate the drawback of FPS in both efficiency (which can be found in Section 4.2.3) and effectiveness at low sampling rates. With grid_fps, we divided the point cloud into small grids and performed FPS in each grid. From Figure 4, we can observe that grid_fps was a strong baseline and surpassed almost all others except sta_peak and sta_bayes.

Overall, our methods, sta_peak and sta_bayes had the advantage in both effectiveness and robustness at various sample rates compared to baselines.

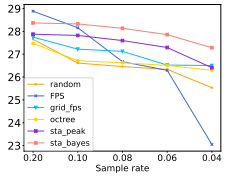


Figure 9. Macro mAP with PointPillars on nuScenes

b) Comparison with Baselines under Macro mAP. Figure 7 illustrated the macro mAP with PartA2 (KITTI). First, for all examined methods, a decline in sampling rate correlates with a decrease in mAP, and FPS demonstrated the most rapid decline. The mAP falls below that of random sampling when the sampling rate falls below 20%. Second, sta_peak and sta_bayes showed a more significant advantage when the sampling rate decreased, indicating that they were capable of effectively downsampling the point cloud by maintaining a high mAP. Third, we assessed the macro mAP in accordance with different levels of hardness, which is defined based on the occlusion, and object size [2].

We found that sta_peak and sta_bayes consistently held a distinct advantage across all hardness levels, surpassing all other baselines. By contrast, FPS and grid_FPS gradually lost their advantage at moderate and hard levels.

4.2.2 Density Peak vs. Naïve Bayes (Q2)

a) Density Peak Method vs. Naïve Bayes Method. Upon examining the categorical mAP, e.g., Figure 4 for mAP with Part-A2 (KITTI), at a sampling rate of 30% and 20%, sta_bayes could hardly outperform sta_peak overwhelmingly, especially in the category 'car'. However, the advantage of sta_bayes gradually became apparent when the sampling rate decreased, especially for small objects (e.g., pedestrians). For the macro mAP, as shown in Figure 7, sta_bayes consistently outperformed sta_peak at all hardness levels. This indicates that sta_bayes excelled in detecting objects with diverse truncations and visibilities, demonstrating enhanced effectiveness.

b) Generalization Ability. In all the aforementioned comparisons, experiments were conducted across different datasets using various inference models, yielding consistent conclusions. Specifically, for the KITTI dataset, categorical mAP results were depicted in Figure 4 and Figure 5, with inference models Part-A2 and SECOND, respectively. Additionally, Figures 7 and 8 present the macro mAP for these two models. Similarly, experiments were conducted on the nuScenes dataset using PointPillars as the inference model, with results illustrated in Figure 6 and Figure 9.

4.2.3 Efficiency Study (Q3)

Table 2. Processing time per point cloud in seconds

Dataset	rand	FPS	g_fps	octree	**sta_p**	**sta_b**
KITTI	0.0005	2.44	0.65	0.68	0.05	0.09
nuScenes	0.001	8.13	2.47	1.36	0.13	0.32

Table 2 provided the time required by various methods for sampling a single point cloud. First, for our proposed two methods, sta_peak showed an advantage compared with baselines, while sta_bayes exhibited an obvious time increase compared with sta_peak. According to our analysis in Section 4.2.2, sta_bayes traded the processing time for better sample results. Second, the KITTI dataset typically consists of approximately 18,000 points per point cloud, whereas nuScenes exhibited approximately 30,000 points per point cloud. Specifically, the LiDAR scanning rate for KITTI operated at 10Hz, resulting in a point cloud generation rate of 0.1 seconds per point cloud. In contrast, nuScenes adopted a data collection rate of 2Hz, equal to 0.5 seconds per point cloud. It is noteworthy that, among the methods we evaluated, only sta_peak, sta_bayes, along with the random sampling, had the capability to meet the generation rates for both datasets.

4.2.4 Ablation Studies (Q4)

Effect of Different Background Ratio. We conducted experiments on different object-to-background ratios for sampled points using both sta_peak and sta_bayes, as depicted in Figure 10. We experimented with the ratios of 6:4, 7:3, and 8:2, respectively. The results showed that, the more object points we sampled, the higher mAP we obtained. However, there was a substantial mAP increase from the 6:4 to the 7:3, while the increase was less pronounced from the 7:3 to the 8:2, which suggested that increasing the ratio of object points can indeed enhance overall accuracy.

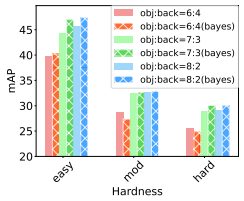


Figure 10. different object and background ratio

Effect of Z-axis Filtering. Table 3 presented the experiment results of performing the point classification with the Z-axis filter and without the Z-axis filter. The results indicated that the Z-axis filter provided more benefits at sample rates higher than 10%, but the advantage gradually disappeared as the sample rate decreased. The experiment results indicated that the Z-axis filter could effectively filter out the ground, which significantly contributed to retaining object points. However, this method inevitably led to the issue of classifying the bottom part of objects on the ground as background points. Consequently, at low sample rates, using the Z-axis filter resulted in a lower mAP.

Table 3. Ablation study on z-axis for density peak method

Rate	without z-axis filter			**with z-axis filter**		
	Easy	Med	Hard	Easy	Med	Hard
50%	74.26	60.49	56.96	75.39	61.01	57.35
30%	66.65	52.89	49.34	67.50	52.13	48.84
10%	44.33	32.41	28.81	42.19	29.89	26.65
6%	29.50	20.92	17.89	29.75	19.91	17.23
2%	8.23	5.15	4.10	6.05	3.72	3.16

5. Conclusion

The excessive amount of 3D points used in autonomous driving a great challenge to processing efficiency and hardware cost. To tackle the problem, we introduce the Representative Point Selection approach, aiming at downsampling point clouds with awareness to object presence. We propose a two-step sampling approach to efficiently downsample point clouds, focusing on retaining object points. Through experiments on KITTI and nuScenes, the method proves more effective and efficient than existing baselines.

References

[1] Technical report and codes. https://github.com/PetalZh/pointSelection, 2023. 4, 5, 6, 7

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision*, 2019. 1, 3, 5, 7

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 1, 3, 5

[4] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 5

[5] Yaodong Cui, Ren Chen, Wenbo Chu, Long Chen, Daxin Tian, Ying Li, and Dongpu Cao. Deep learning for image and point cloud fusion in autonomous driving: A review. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):722–739, 2021. 1

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

[7] Emad El-Sayed, Rehab F. Abdel-Kader, Heba Nashaat, and Mahmoud Marei. Plane detection in 3d point cloud using octree-balanced density down-sampling and iterative adaptive plane extraction. *IET Image Process.*, 12(9):1595–1605, 2018. 6

[8] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010. 6

[9] Tobias Glasmachers. Limits of end-to-end learning. In *Proceedings of The 9th Asian Conference on Machine Learning*, pages 17–32. The Proceedings of Machine Learning Research, 2017. 2

[10] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11108–11117, 2020. 1, 2

[11] Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. Octsqueeze: Octree-structured entropy model for lidar compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1313–1323, 2020. 3

[12] Nisarg S Joshi, Raghav Raghuwanshi, Yash M Agarwal, B Annappa, and DN Sachin. Arima-pid: container auto scaling based on predictive analysis and control theory. *Multimedia Tools and Applications*, pages 1–18, 2023. 1

[13] Fiodar Kazhamiaka, Matei Zaharia, and Peter Bailis. Challenges and opportunities for autonomous vehicle query systems. In *11th Conference on Innovative Data Systems Research*, 2021. 1

[14] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019. 5

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

[16] Ying Li, Lingfei Ma, Zilong Zhong, Fei Liu, Michael A Chapman, Dongpu Cao, and Jonathan Li. Deep learning for lidar point clouds in autonomous driving: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3412–3432, 2020. 1

[17] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[18] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 2, 3

[19] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2, 3

[20] Zizheng Que, Guo Lu, and Dong Xu. Voxelcontext-net: An octree based framework for point cloud compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6042–6051, 2021. 3

[21] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. In *3rd Symposium on Point Based Graphics, PBG@SIGGRAPH 2006*, pages 111–120. Eurographics Association, 2006. 3

[22] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(8):2647–2664, 2021. 3, 5

[23] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3

[24] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *Computer Vision - ECCV 2020 - 16th European Conference, Proceedings, Part XXVIII*, volume 12373 of *Lecture Notes in Computer Science*, pages 685–702. Springer, 2020. 3

[25] XiaoHui Wang, HuaWei Chen, and LuShen Wu. Feature extraction of point clouds based on region clustering segmentation. *Multimedia tools and applications*, 79:11861–11889, 2020. 3

[26] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 3

[27] Tsung-Han Wu, Yueh-Cheng Liu, Yu-Kai Huang, Hsin-Ying Lee, Hung-Ting Su, Ping-Chia Huang, and Winston H. Hsu. Redal: Region-based and diversity-aware active learning for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15510–15519, 2021. 2

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

[29] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *Computer Vision - ECCV 2022 - 17th European Conference, Proceedings, Part XXVIII*, pages 677–695. Springer, 2022. 3

[30] Yan Yan, Yuxing Mao, and Bo Li. SECOND: sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 3, 5

[31] Renrui Zhang, Liuhui Wang, Yu Qiao, Peng Gao, and Hongsheng Li. Learning 3d representations from 2d pre-trained models via image-to-point masked autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21769–21780, 2023. 3

[32] Yifan Zhang, Qingyong Hu, Guoquan Xu, Yanxin Ma, Jianwei Wan, and Yulan Guo. Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18953–18962, 2022. 1, 2

[33] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021. 3

[34] Xuanyu Zhou, Charles Ruizhongtai Qi, Yin Zhou, and Dragomir Anguelov. Riddle: Lidar data compression with range image deep delta encoding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17212–17221, 2022. 3

[35] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018. 2

[36] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9939–9948, 2021. 3