Feature Based Sampling: A Fast and Robust Sampling Method for Tasks Using 3D Point Cloud

JUNG-WOO HAN1, DONG-JOO SYNN2, TAE-HYEONG KIM3, HAE-CHUN CHUNG4, JONG-KOOK KIM5,

1School of Electrical Engineering, Korea University, Seoul, 02841, Republic of Korea (e-mail: hanjungwoo1@korea.ac.kr)
2School of Electrical Engineering, Korea University, Seoul, 02841, Republic of Korea (e-mail: alansynn@korea.ac.kr)
3School of Electrical Engineering, Korea University, Seoul, 02841, Republic of Korea (e-mail: magental@korea.ac.kr)
4School of Electrical Engineering, Korea University, Seoul, 02841, Republic of Korea (e-mail: marcomx@korea.ac.kr)
5School of Electrical Engineering, Korea University, Seoul, 02841, Republic of Korea (e-mail: jongkook@korea.ac.kr)

Corresponding author: Jong-Kook Kim (e-mail: jongkook@korea.ac.kr)

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A1B04933156).

ABSTRACT Point cloud data sets are frequently used in machines to sense the real world because sensors such as LIDAR are readily available to be used in many applications including autonomous cars and drones. PointNet and PointNet++ are widely used point-wise embedding methods for interpreting Point clouds. However, even for recent models based on PointNet, real-time inference is still challenging. The solution to a faster inference is sampling, where, sampling is a method to reduce the number of points that is computed in the next module. Furthest Point Sampling (FPS) is widely used, but disadvantage is that it is slow and it is difficult to select critical points. In this paper, we introduce Feature-Based Sampling (FBS), a novel sampling method that applies the attention technique. The results show a significant speedup of the training time and inference time while the accuracy is similar to previous methods. Further experiments demonstrate that the proposed method is better suited to preserve critical points or discard unimportant points.

INDEX TERMS Artificial Intelligence (AI), Point-wise MLP, Layered Architecture, Machine Learning, 3D Point Cloud, Sampling methods

I. INTRODUCTION

N real-world applications, it is essential to receive and use 3D data as input to interpret the real world. For example, Simultaneous Localization and Mapping (SLAM) is used for robot or autonomous vehicle applications to locate the current position and it is necessary to process 3D geometric information in the form of point cloud from sensors such as LIDARs (light detection and ranging). Point cloud data is both unordered and unstructured where, N pieces of data represent an object or surrounding terrain, and each point consists of coordinates and features. To process point cloud data, a method that guarantees permutation invariance and fast inference is required.

There are many point cloud data processing methods that use Deep Learning techniques. Multi-view-based methods using traditional 2D models by projecting point cloud data in multiple directions and turning it into 2D images [1]–[3] and Volumetric-based methods using 3D convolution by voxelizing the point cloud data [4]–[9] incur substantial computational cost for the pre-processing. Researchers began to attempt point-based methods that did not use pre-processing and thus reducing the computational cost [10]–[15]. Among
these point-based method, a scheme that uses the SharedMLP model called the PointNet [10] that can process 2D data and point cloud data was introduced. PointNet became the basis of many pointwise-MLP methods.

An improved version of PointNet, a hierarchical structured version called PointNet++ [11] was proposed. The PointNet++ architecture uses the sampling and grouping module throughout the enhanced 2D convolution. Furthermore, the hierarchical structure gives advantages to the receptive field. PointNet++ uses a sampling method called Furthest Point Sampling (FPS) that reduces the size of the original image while attempting to preserve points that are crucial for recognition tasks. The FPS method incurs a substantial computational cost because the cost increases squarely as the number of points in the image increases. Thus, as complex images consist of more points in a point cloud, the original speed advantage is diminished. Also, in general, the FPS scheme fails to exclude outliers in noisy and complex images, which can be critical to a model’s performance. Many recent approaches that enhance the point-wise methods by adding techniques for robustness [12], [16] still use FPS as the sampling method resulting in slow training and inference time. There are other sampling methods such as Gumbel Subset Sampling [17] that uses high-dimensional spatial analysis by mixing features extracted from each point in the point cloud, adaptive sampling for PointASNL [12] and ASHFNet [18] that shift the neighbor points to the centroid point and then uses FPS, Self-Organizing Map for So-Net [19] that finds points that express objects well using unsupervised competitive learning, and random sampling in RandLANet [20] that has a tremendous advantage in speed.

In this paper, we propose a novel sampling method called feature-based sampling (FBS) that is inspired by Importance Sampling [21] to significantly reduce computational cost while trying to preserve the accuracy for recognition tasks. The targeted accuracy is the accuracy of the PointNet++ and pointASNL that currently shows the best accuracy when using 1k number of points. FBS consists of two steps which are Positional Encoding and Self-Attention Scoring as shown in Fig. 1 and is applied to all input points to reduce the number of points for the next step of a given task. The Positional Encoding phase calculates the relationship information of all points, for each point using its surrounding points and the query ball point method [11]. In the Self-Attention Scoring phase, important features are highlighted using a dot product calculation [22] of the weights of the features.

Our proposed sampling method, FBS, is applied for the classification task [23] and part segmentation task [24]. In the classification task, our proposed method shows a maximum of three times faster inference time than the existing model PointNet++ and about eight times faster than Point-ASNL [12], with small accuracy improvement. In the part segmentation task, our proposed method shows that the accuracy decreases by less than 1%, while it improves the inference time by 25%. An additional experiment applying random noise to each dataset shows that PointNet++ using our FBS method successfully excludes the noise data.

To summarize, our contributions include:

- A novel feature-based sampling based on the concept of importance sampling is proposed that reduces the computation cost of the sampling process that are used in many tasks using point cloud data.
- Our proposed method show comparable accuracy for classification and part segmentation tasks while reducing the computation cost.
- FBS can better preserve important data points while deleting outlier points.

The rest of this paper is organized as follows. In Section II, previous research is reviewed and difference is discussed. The novel sampling method is proposed in Section III. In Section IV, robustness, accuracy and the speed of FBS is compared with previous research. Finally, conclusions are summarized in section V.

II. RELATED WORK

PointNet [10] is a pioneering work that directly uses the unordered and unstructured point cloud data. PointNet uses a point-wise multi-layer perceptrons (MLPs) followed by the max-pooling operation. An improved hierarchical version of PointNet, PointNet++ [11] collects information sequentially through hierarchical structures and calculates the relationship between points and use the grouping scheme for the furthest point sampling method. PointASNL [12] improves the efficiency of feature extraction by applying the attention technique and also achieves robustness through Adaptive Sampling. In addition, Point-ASNL proposed a local-nonlocal (L-NL) module for capturing the neighbor and long-range dependencies of the sampled points. PointWeb introduced Adaptive Feature Adjustment to efficiently select features for local neighborhoods from the given points [16], PointNet++ and PointASNL and PointWeb that improves PointNet++ use Furthest Point Sampling as the sampling method. Our Feature Based Sampling method incur lesser computational cost while keeping the accuracy comparable and show better performance for outlier removal.

III. PROPOSED METHOD

A. OVERVIEW

This Section proposes Feature-based Sampling (FBS) that consists of two modules called Positional Encoding (PE) in subsection III-B and Attention Scoring (AS) in subsection III-C. The general flow diagram of FBS is illustrated in Fig. 2 (b).

B. POSITIONAL ENCODING

Positional Encoding (PE) extracts relevant information of a given point using its surrounding points to determine the importance of that point. PE encodes the surrounding points \( P^k \) for a given point in point cloud \( P \), constructing new information using existing features \( F \) (e.g., raw RGB or learned A. OVERVIEW

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
FIGURE 2. Feature-based Sampling’s Process and Flow Diagram: (a) shows the sampling process of an example point cloud image. (b) shows the step-by-step process of the feature-based sampling module.

features). Encoding based on x-y-z coordinates creates the geometric information of the center point relative to its surrounding points. PE is divided into Grouping Neighbour Points, Feature Extraction, and Feature Augmentation steps.

1) Grouping Neighbour Points
For each i-th point, K neighborhood points are determined. The proposed method utilizes the query ball scheme where it only collects information from neighbors that are only a certain distance away (inside a certain radius).

\[ index_i = Q(p_i, K, P) \]
\[ index_x_i = \{p_1 \cdots p_i \cdots p_K\} \]
\[ index_x_i(F) = \{f_1 \cdots f_i \cdots f_K\} \]  

The Q is the Query ball point algorithm, \( P \) is the x-y-z coordinates of the points, and \( F \) is the feature of the points. \( index_i \) is the index set for the i-th point determined using the query ball point scheme. \( p_i \) denotes the center point (centroid) in \( P \).

For each centroid \( p_i \) (dashed line in Fig. 2), the \( index_i \) is applied to \( P \) and \( F \). The points surrounding the centroid \( p_i^K \) (yellow-green \( (N, k) \) in Fig. 2), and the features corresponding to the surrounding points are expressed as \( f_i^K \) (orange \( (N, k) \) in Fig. 2).

2) Feature Extraction
For each centroid \( p_i \) (yellow (1, 3) in Fig. 2), geometric information is extracted using the set of K points close to the centroid \( \{p_1 \cdots p_i \cdots p_K\} \).

\[ e_i^K = MLP(p_i \oplus p_i^K \oplus (p_i - p_i^K)) \]  

\( p_i \) and \( p_i^K \) consist of x-y-z coordinates, and results are derived through concatenation marked as \( \oplus \). Then using a multi-layer perceptron (MLP) module, the features \( e_i^K \) (green in Fig. 2) are determined. This method is similar to the method used in randLA-Net [20]. The difference is that the Query Ball Point scheme is used. The resulting features are used to calculate the score in subsection III-C.
3) Feature Augmentation
The feature is augmented using $e_k^i$ determined from equation (2). That is, a feature $f_k^i$ of $p_i$ is concatenated using $e_k^i$. Then, the total dimension of $e_k^i \oplus f_k^i$ is expanded to C, which is the number of channels. There is a trade off between the number of channels and accuracy. Thus, C is determined empirically.

$$F_i^c = MLP(e_k^i \oplus f_k^i)$$ (3)

The encoded information $e_k^i$, $F_i^c$ is created (orange and green in Fig. 2) by concatenating the previously obtained $f_k^i$. $F_i^c$.

C. SELF-ATTENTION SCORING
The self-attention method ([22]) is used to determine the importance of the features generated in the previous positional encoding phase. The dot product equations (2) and (3) is calculated to increase the difference of the importance. Finally, the sum of the calculated importance is used for sampling (selecting or discarding).

From the given set of local features $F_i^c = \{F_1^c, \ldots, F_n^c, \ldots, F_i^c\}$, Feature Attention Score is computed using $S(x)$ where softmax is first applied and then a shared MLP is used. The result $s_i^c$ is shown in equation 4 (blue in Fig. 2).

$$s_i^c = S(F_i^c)$$ (4)

Applying softmax makes it possible to generate a weight that can emphasize the quality of each feature. That is, this weight will identify the essential features and adverse features. Using this information, importance of features can be distinguished and the ranks of the points are determined. This method is suitable for removing outliers, which means that noise is minimized, objects are expressed more clearly, and important points can be selected in the sampling phase.

$$Score_i = \sum_{k} F_i^c \cdot s_i^c$$ (5)

The final score, $Score_i$, (purple in Fig. 2) of a particular point is determined by the dot product calculation of equations (3) and (4) and summing all the C features as shown in equation (5). Summation is instead of typical pooling techniques such as Max/Min/Avg to prevent information loss.

D. FEATURE-BASED SAMPLING
1) Importance Sampling
Feature based sampling (FBS) discovers neighboring points for each centroid, creates a new feature, assigns an attention weight, and a sampling score for each point. The process is used to determine how critical the point is for a given point cloud image. Therefore, our sampling method can be referred to as importance sampling.

2) Computational Cost
Feature-based Sampling consists of a repeated sequence of overlapping Positional Encoding and Attention Scoring. Because all modules are connected in a linear form, the module with the highest time complexity dominates the overall time complexity. In positional encoding, Grouping Neighbor Points is worst case $O(N^2)$ when using the query ball point method, where N is the total number of points. In both feature extraction and feature augmentation, the time complexity is $O(N)$ because of concat and MLP operations N times. Attention scoring consists of N dot-products and sigma, and the time complexity is $O(N)$. Thus, the time complexity of FBS is dominated by the query ball scheme which is worst case $O(N^2)$. Compared to the K nearest neighbor computation whose time complexity is always $O(N^2)$, query ball point method stops when $K$ points are found within the desired radius. The worst case occurs when the query ball method cannot find K points within a certain radius and distance for all points are calculated. The best case time complexity is $O(N \times K)$, where the query ball scheme only searches K times and finds exactly K points within a radius for all N total number of points.

Usually when applying the Attention technique, it is applied in a point-wise form. However, FBS uses a feature-wise from and therefore, the computational cost is lower than other techniques that use Attention.

IV. RESULTS
A. EXPERIMENTAL SETUP
In this paper, ModelNet40 [23] for the classification task and ShapeNet [24] for part segmentation task is used to evaluate the proposed method. Python and PyTorch are used for the implementation of the proposed method. The machine used for all experiments was the Threadripper 3990X 64-Core (128-thread) with 128 GB RAM and the NVIDIA RTX 3090 24 GB as the GPU. The PointNet++, and Point-ASNL are implemented and executed for accurate comparison of the proposed feature based sampling (FBS) method. PointNet++ is used as the backbone and the only difference is the sampling part (FBS). Negative Log-Likelihood (NLL) [25] is adopted for the loss function.

ModelNet40 has 9843 training data and 2468 test data in 40 different classes. For the training phase, 1024 points images are used as the input. The augmentation strategy from PointASNL [12] is used that includes (1) anisotropic scales in the range $[-0.5, 1.25]$, (2) translate in the range $[0.1, 0.1]$, and (3) 20% random dropout points. The same hyper-parameters and settings were used except for the training module for fair comparison. Adam Optimizer is used with an initial learning rate of 0.001 and the decay rate of 0.0001. The batch size was set to 16, and the methods were trained for 200 epochs. When determining the overall accuracy for all test data, the experiments were conducted by applying vote 10. A random scale was applied to the data loader. Overall Accuracy (OA) is used as the metric.

ShapeNet is a manufactured synthetic dataset of 16,881 shapes divided into 16 classes and 50 subdivided parts. Unlike classification, part segmentation is a more difficult task. Because there are several partial classifications of an object. All parameters are set to the same ones used in PointNet++. The batch size was set to 16 and was trained for 251 epochs.
The learning rate was 0.001 and the weight decay rate was 0.001. For the base model, the number of points was 2048, a learning decay of 0.5 was applied for every 20 step size and used Adam as the optimizer. From the dataset, up to 2048 points are randomly selected and used for training and the class is determined through one-hot encoding at the end. Mean units of intersection (mIoU) is used as the metric.

B. CLASSIFICATION
As shown in Table 1, the overall accuracy of the two best methods using 1k points, PointNet++ and Point-ASNL is 92.9%. Compared to these methods, the overall accuracy of the proposed method that uses FBS was slightly increased to 92.95%. All methods are run 100 times and the average is shown. SO-Net is inserted into Table 1 to be a reference of the accuracy performance. SO-Net uses a lot more points to solve the classification problem and therefore it is difficult to compare directly. However, as shown in the table, the accuracy performance is comparable.

Train and Inference Time is determined for different number of input points (e.g., 512, 1024, and 2048). This experiment is aimed to compare the performance when the number of points is increased such that the results can be a guide for similar tasks or applications that use that number of points. The performance counter (perf_counter()) from the Python Time package was used to profile the computational time. While training time was determined per batch, inference time was measured in terms of the end-to-end time of the method. The three methods are run 100 times and the average time was used for comparison.

When compared to PointNet++, the proposed method completed the task faster for 512, 1024, and 2048 input data points as shown in Table 3. For both 512 and 1024 points, our method reduced training time by around 40% and time was reduced by about 23% for the 2048 points case. The most time consuming part is the sampling process and the proposed method in this paper effectively reduces the time to sample while also determining the points that are important. Furthest Point Sampling idea, which is used by previous research, approximates the importance of points using the distance and thus it is not a sampling method that requires learning. On the other hand, FBS has learning parameters because it proceeds in the form of calculating and learning the relationship between points. Even though FBS has more learning parameters than FPS, our method is still faster because of the reason discussed in Section III. The inference speed is about three times faster for 512 and 1024 points cases. For the 2048 points case, the inference time was reduced by about 21%.

Robustness The robustness experiment on the missing data ratio was conducted for the three sampling methods, Random, FPS, and FBS. The dataset consisting of 1024 points was used and out of 1024 points, 512, 128, 64, and 32 points are randomly picked for the robustness test (i.e., corruption ratio of 50%, 88.5%, 93.7%, and 96.8%).

| Method          | input | #points | M40  |
|-----------------|-------|---------|------|
| SO-Net [19]     | pnt, nor | 2k | 90.9 |
| EllipsoidNet [26] | pnt | 2k | 92.7 |
| SO-Net [19]     | pnt, nor | 5k | 93.4 |
| pointwise CNN [27] | pnt | 1k | 86.1 |
| ECC [28]        | graphs | 1k | 87.4 |
| PointNet [10]   | pnt | 1k | 89.2 |
| SGMNet [29]     | pnt | 1k | 90.0 |
| PAT [17]        | pnt, nor | 1k | 91.7 |
| Spec-GCN [30]   | pnt | 1k | 91.8 |
| PointGrid [5]   | pnt | 1k | 92.0 |
| PointCNN [31]   | pnt | 1k | 92.2 |
| DGCNN [32]      | pnt | 1k | 92.2 |
| PCNN [33]       | pnt | 1k | 92.3 |
| PointConv [34]  | pnt, nor | 1k | 92.5 |
| A-CNN [35]      | pnt, nor | 1k | 92.6 |
| Point2Sequence [36] | pnt | 1k | 92.6 |
| PointNet++ [11] | pnt, nor | 1k | 92.9 |
| PointASNL [12]  | pnt, nor | 1k | 92.9 |
| Ours            | pnt, nor | 1k | 93.0 |

TABLE 1. Overall accuracy results of the classification task using the ModelNet40 dataset, where “pnt” stands for coordinate of points, “nor” stands for feature of points. #points stands for input number of points. Our result of 93.0 is the rounded up value of 92.96.
Although the results are similar, FBS always showed better results than FPS and random generally showed worse results than the other two.

![Figure 4. Robustness to data Corruption.](image)

**PointASNL** When compared to Point-ASNL that uses Furthest Point Sampling, our method showed significant time reduction. Because the experimental environment of PointASNL was originally Tensorflow, it had to be implemented using the PyTorch for fair comparison. The Table 4 shows the experimental results of Training Time and Inference Time for 1024 input Points. PointASNL presented an L-NL module that selects features well and an adaptive sampling method that removes outliers. Although these techniques enhanced the PointNet++, PointASNL still uses Furthest Point Sampling and thus a significant time is used for training time and inference time. When the proposed sampling method (FBS) is used the speed up was six times for training time and was four times for inference time.

| TABLE 2. | Part Segmentation performance with part-averaged IoU on ShapeNet. |
|---|---|
| **Method** | **mIoU** | **accco** | **bag** | **cap** | **car** | **chair** | **desk** | **phone** | **guitar** | **knife** | **lamp** | **laptop** | **motor** | **mug** | **pistol** | **rocket** | **skateboard** |
| **#shapes** | **2060** | **76** | **55** | **898** | **3758** | **69** | **787** | **392** | **1547** | **451** | **202** | **184** | **286** | **66** | **154** | **5271** |
| PointNet++ [11] | 85.7 | 83.4 | 78.7 | 82.5 | 78.9 | 89.6 | 73.0 | 91.5 | 85.9 | 80.8 | 93.3 | 66.2 | 93.0 | 81.2 | 97.9 | 72.8 | 80.9 |
| SO-Net [19] | 84.9 | 82.8 | 77.8 | 88.0 | 73.0 | 90.6 | 73.5 | 90.7 | 83.9 | 82.9 | 94.8 | 69.1 | 94.2 | 80.9 | 53.1 | 72.9 | 83.0 |
| DGCNN [32] | 85.1 | 84.2 | 83.7 | 84.4 | 77.1 | 90.9 | 78.5 | 91.5 | 87.3 | 82.9 | 96.0 | 86.3 | 93.3 | 82.6 | 59.7 | 75.5 | 82.0 |
| Point2Sequence [36] | 85.2 | 82.6 | 81.8 | 87.5 | 77.3 | 90.8 | 77.1 | 91.1 | 86.9 | 83.9 | 95.7 | 70.8 | 94.6 | 79.3 | 58.1 | 75.2 | 82.8 |
| PointCNN [31] | 86.1 | 84.1 | 86.5 | 86.0 | 80.8 | 90.6 | 79.7 | 92.3 | 88.4 | 85.3 | 96.1 | 77.2 | 95.2 | 84.2 | 64.2 | 80.0 | 83.0 |
| PointASNL [12] | 86.1 | **84.1** | **84.7** | **87.9** | 79.7 | **92.2** | 73.7 | 91.0 | 87.2 | 84.2 | 95.8 | 74.4 | **95.2** | 81.0 | 63.0 | 76.3 | **83.2** |

**TABLE 3.** Experimental result of train and inference times using ModelNet40 dataset compared to PointNet++.

| Number of Points | Training Time(ms/batch) | Inference Time(sec/epoch) |
|---|---|---|
| **#1024 points** | PointNet++ | our Method | PointNet++ | our Method |
| 512 | 225.54 | **154.52** | 25.01 | **8.36** |
| 1024 | 234.56 | 140.76 | 26.12 | 9.35 |
| 2048 | 255.55 | 198.47 | 32.52 | 25.68 |

**TABLE 4.** Experimental result of train and inference times using ModelNet40 dataset compared to PointASNL.

**C. OUTLIER REMOVAL**

Experiments are executed to compare the outlier removal between FPS and FBS. For a point cloud image, noise $(x, y, z)\pm random$ was randomly injected. The experiment was carried out using the visually well-represented Airplane and Chair (average 8.4 noise points) point cloud image. A total of 40 objects were selected for the experiment, 20 for each class. After sampling from 1024 points images, FPS resulted in an average of 7.1 outlier points for 512 samplings and 5.6 for 128 samplings. The FBS method resulted in an average of 5.1 outlier points in 512 samplings and 2.6 in 128 samplings. The reason is because the Furthest Point Sampling selects points that are furthest apart by calculating the distance and has no information about the importance of the points and therefore, outliers are more prone to be selected for the next stage of the classification process. Because outliers are most likely to be further away from other points. Adaptive sampling used in PointASNL proceeds with the combination of adaptive shift and FPS. However, as it uses FPS, the same problem arises. The example shape shown in Figure 5 may seem like a degraded image for the human perception, but it has sufficient information for a learning network to complete the classification task.

**D. TENSORFLOW CUDA KERNEL**

Experiments are conducted for FPS in two different settings; the Tensorflow CUDA kernel implementation and PyTorch implementation. The reason is that the version using Tensorflow has a time discrepancy by creating and using a C++ code that processes the FPS algorithm directly using the CUDA kernel. The machine used for the experiment was Intel Xeon 1.6GHz 6-Core (x2) (12-Thread) with 98 GB RAM and Titan XP (12GB) as the GPU. Pytorch uses only the GPU memory it needs for the sampling process because it calculates during the learning phase. Therefore, the number of points affects
the inference time. In the Tensorflow implementation, the maximum gpu memory is used when the sampling operation is performed during the preprocessing phase. Results show that the both implements are similar in time but the pytorch implementation is faster. Thus, the comparison of FBS versus FPS using the pytorch implementation is a valid comparison.

**E. PART SEGMENTATION**

The part segmentation experiment resulted in our method achieving a mIoU of 84.3% as shown in Table 2, which is slightly lower than other methods. Compared to PointNet++, the difference is less than 1%.

**Train and Inference Time** is reduced but in smaller amounts than the classification problem, because the part

| Sampling points | Pytorch Time(s) | Pytorch GPU(MB) | Tensorflow CUDA Time(s) | Tensorflow CUDA GPU(MB) |
|-----------------|----------------|-----------------|-------------------------|-------------------------|
| 512 points      | 0.0808         | 557             | 0.2330                  | 11,627                  |
| 1024 points     | 0.2845         | 557             | 0.2343                  | 11,627                  |
| total time      | **0.3653**     |                 | **0.4673**              |                         |

**TABLE 5.** Efficiency experiment with different frameworks of sampling.
segmentation task is a more complex task than the classification task, additional modules are included to better identify the parts from the part segmentation data. Therefore, the sampling portion of the overall model is reduced and thus the effect of a better sampling is also reduced. As shown in Table 6, for both 512 and 1024 points data, our method reduced the training time by 35% and 25% for 2048 points. This result shows that the volume of tensor increases as the number of points increases. The inference time is reduced by approximately 60% for both 512 and 1024 points data and around 20% for 2048 points. The inference time also increases as the number of points increases.

### Table 6. Experimental result over train time and inference time with ShapeNet dataset.

| Number of points | Training Time(ms/batch) | Inference Time(sec/epoch) |
|------------------|--------------------------|---------------------------|
|                  | PointNet++ ours          | PointNet++ ours           |
| 512              | 246.60                   | 29.77                     | 10.82 |
| 1024             | 251.82                   | 31.30                     | 12.22 |
| 2048             | 280.95                   | 39.12                     | 31.62 |

**TABLE 6.** Experimental result over train time and inference time with ShapeNet dataset.

![Graph showing training and inference time](graph.png)

**FIGURE 6.** Experimental results of training and inference time according to the number of points in ShapeNet.

### F. IMPACT OF FEATURE BASED SAMPLING

We showed that the proposed method achieves acceptable throughput aside from the previous approaches. While the condensed 3D sensing data, including LIDAR, is highly integrated, previous approaches perform sampling to extract representative values and downsize such data. Conducting appropriate sampling at this time dramatically affects both the overall training and inference process. Until recently, most models used FPS [11] or Adaptive Sampling [12] to process the point cloud data. We showed that KNN is an operation with too much overhead, and random sampling is not appropriate because it can eliminate important data points, as shown in outlier removal. However, this paper presents that optimizing appropriate sampling techniques is crucial for real-world point cloud data processing. After all, the engineers in the real world need to increase the throughput for data processing under 10ms for real-time processing. Therefore, the approach presented by this paper points out that optimization of the sampling part and the model itself must be performed to increase the FPS.

### V. CONCLUSION

This paper proposed the Feature-based Sampling (FBS), a sampling method using the importance sampling idea. FBS is a fast and robust sampling method that uses Positional Encoding and Attention Scoring. Results show that FBS significantly reduces the training time and inference time compared to the PointASNL and PointNet++ methods while achieving comparable accuracy for the classification task (minimal increase) and part segmentation task (slight decrease). Outlier removal experiment shows that FBS is better suited to discard outliers (i.e., noise in images) than the Furthest Point Sampling method that is used by PointNet++ and PointASNL. The next step is to apply our sampling method to various other tasks.

### REFERENCES AND FOOTNOTES

#### A. REFERENCES

**REFERENCES**

[1] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, “Multi-view convolutional neural networks for 3d shape recognition,” in Proceedings of the IEEE International Conference on Computer Vision (ICCV), December 2015.

[2] Z. Yang and L. Wang, “Learning relationships for multi-view 3d object recognition,” in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2019.

[3] T. Yu, J. Meng, and J. Yuan, “Multi-view harmonized bilinear network for 3d object recognition,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

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

[5] T. Le and Y. Duan, “Pointgrid: A deep network for 3d shape understanding,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[6] H.-Y. Meng, L. Gao, Y.-K. Lai, and D. Manocha, “V-Net: Voxel VAE net with group convolutions for point cloud segmentation,” in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2019.
[7] Y. Chen, S. Liu, Y. Shen, and J. Jia, “Fast point r-cnn,” in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2019.

[8] L. Zhou, Y. Du, and J. Wu, “3d shape generation and completion through point-voxel diffusion,” in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 5826–5835, October 2021.

[9] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun, and S. Pu, “Rvnnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation,” in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 16024–16033, October 2021.

[10] 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 (CVPR), July 2017.

[11] 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 (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[12] 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/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[13] Y. Liu, B. Fan, S. Xiang, and C. Pan, “Relation-shape convolutional neural network for point cloud analysis,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[14] T. Suzuki, K. Ozawa, and Y. Sekikawa, “Rethinking pointnet embedding for faster and compact model,” in 2020 International Conference on 3D Vision (3DV), pp. 791–800, 2020.

[15] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, “Point transformer,” in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 16259–16268, October 2021.

[16] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, “Pointweb: Enhancing local neighborhood features for point cloud processing,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[17] J. Yang, Q. Zhang, B. Ni, L. Li, J. Liu, M. Zhou, and Q. Tian, “Modeling point clouds with self-attention and gumbel subset sampling,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[18] D. Zong, S. Sun, and J. Zhao, “Ashf-net: Adaptive sampling and hierarchical folding network for robust point cloud completion,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 3625–3632, 2021.

[19] J. Li, B. M. Chen, and G. H. Lee, “So-net: Self-organizing network for point cloud analysis,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[20] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, “Randla-net: Efficient semantic segmentation of large-scale point clouds,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[21] T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák, “Neural importance sampling,” CoRR, vol. abs/1808.03856, 2018.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017.

[23] 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), June 2015.

[24] 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.

[25] L. L. Miranda, “Understanding softmax and the negative log-likelihood,” ljvmiranda921.github.io, 2017.

[26] Y. Lyu, X. Huang, and Z. Zhang, “Ellipsoidnet: Ellipsoid representation for point cloud classification and segmentation,” in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pp. 854–864, January 2022.

[27] 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 (CVPR), June 2018.

[28] M. Simonovsky and N. Komodakis, “Dynamic edge-conditioned filters in convolutional neural networks on graphs,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[29] J. Xu, X. Tang, Y. Zhu, J. Sun, and S. Pu, “Sgmnet: Learning rotation-invariant point cloud representations via sorted gram matrix,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10468–10477, 2021.

[30] C. Wang, B. Samari, and K. Siddiqi, “Local spectral graph convolution for point set feature learning,” in Proceedings of the European Conference on Computer Vision (ECCV), September 2018.

[31] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, “Pointcnn: Convolution on x-transformed points,” in Advances in Neural Information Processing Systems (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[32] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph cnn for learning on point clouds,” 2018.

[33] M. Artoni, M. Maron, and Y. Lipman, “Point convolutional neural networks by extension operators,” 2018.

[34] W. Wu, Z. Qi, and L. Fuxin, “Pointconv: Deep convolutional networks on 3d point clouds,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[35] A. Komarichev, Z. Zhong, and J. Hua, “A-cnn: Annularly convolutional neural networks on point clouds,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[36] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, “Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network,” 2018.
JUNGWOO HAN was born in Seoul, South Korea in 1994. He received the B.S. degree in electrical engineering from the Sahmyook University, Seoul, Republic of Korea in 2020. He is currently pursuing the M.S. degree in electrical and computer engineering at Korea University, Seoul, Republic of Korea. His research interests include image segmentation, 3D point cloud task and fast robust model.

DOANGJOO SYNN was born in Seoul, South Korea in 1992. He received the B.S. degree in electrical engineering from the Korea University, Seoul, Republic of Korea in 2020. He is currently pursuing the M.S. degree in electrical and computer engineering at Korea University, Seoul, Republic of Korea. From 2012 to 2014, he was a software engineer with the KAIST Engineering School inc. From 2017 to 2020, he was a Software Engineer with the OS and Production Cells, AKA Intelligence inc. and Flysher inc. From 2020 to 2021, he was a AWS Cloud Student Ambassador with the Amazon Web Services. Since 2020, he is a Research Assistant with the High Performance and Intelligent Computing Lab. in Korea University. Since 2021, he is a Research Engineer with the Propia AI inc. Austin, Texas. His research interests include distributed computing, machine learning system, cloud computing and hyperscalar computing.

TAEHYEONG KIM was born in Suyeong District, Busan, Republic of Korea in 1995. He received the B.S. degrees in Electronical Engineering, Computer Science and Engineering from Korea University in 2021. He is currently pursuing the M.S. degree in Electronic Engineering at Korea University, Seoul, Republic of Korea. His current research interests include eXplainable Artificial Intelligence (XAI) in computer vision domain and adversarial attack.

HAECHUN CHUNG was born in Nowon, Seoul, Republic of Korea in 1994. He received the B.S. degrees in computer engineering from the Hongik University in 2020. He is currently pursuing the M.S. degree in electronic engineering at Korea University, Seoul, Republic of Korea. His research interests include self-supervised learning and image generation.

JONG-KOOK KIM received the B.S. degree in electronic engineering from Korea University, Seoul, South Korea, in 1998, and the M.S. and Ph.D. degrees from the School of Electrical and Computer Engineering, Purdue University, West Lafayette, USA in 2000 and 2004, respectively. He is currently a Professor at the School of Electrical Engineering, Korea University, where he joined, in 2007. He was at the Samsung SDS’s IT Research and Development Center, from 2005 to 2007. His research interests include heterogeneous distributed computing, energy-aware computing, resource management, evolutionary heuristics, distributed mobile computing, artificial neural networks, efficient deep learning, and distributed robot systems. He is a Senior Member of the IEEE and ACM.