APSNet: Attention Based Point Cloud Sampling

Yang Ye
yye10@student.gsu.edu
Xiulong Yang
xyang22@student.gsu.edu
Shihao Ji
sji@gsu.edu

Department of Computer Science
Georgia State University
Atlanta, US

Abstract

Processing large point clouds is a challenging task. Therefore, the data is often downsampled to a smaller size such that it can be stored, transmitted and processed more efficiently without incurring significant performance degradation. Traditional task-agnostic sampling methods, such as farthest point sampling (FPS), do not consider downstream tasks when sampling point clouds, and thus non-informative points to the tasks are often sampled. This paper explores a task-oriented sampling for 3D point clouds, and aims to sample a subset of points that are tailored specifically to a downstream task of interest. Similar to FPS, we assume that point to be sampled next should depend heavily on the points that have already been sampled. We thus formulate point cloud sampling as a sequential generation process, and develop an attention-based point cloud sampling network (APSNet) to tackle this problem. At each time step, APSNet attends to all the points in a cloud by utilizing the history of previously sampled points, and samples the most informative one. Both supervised learning and knowledge distillation-based self-supervised learning of APSNet are proposed. Moreover, joint training of APSNet over multiple sample sizes is investigated, leading to a single APSNet that can generate arbitrary length of samples with prominent performances. Extensive experiments demonstrate the superior performance of APSNet against state-of-the-arts in various downstream tasks, including 3D point cloud classification, reconstruction, and registration. Our code is available at https://github.com/Yangyeeee/APSNet.

1 Introduction

With the rapid development of 3D sensing devices (e.g., LiDAR and RGB-D camera), huge point cloud data are generated in the areas of robotics, autonomous driving and virtual reality [7, 20, 21]. A 3D point cloud, composed of the raw coordinates of scanned points in a 3D space, is an accurate representation of an object or shape and plays a key role in perception of the surrounding environment. Since point clouds lie in irregular space with variable densities, traditional feature extraction methods, such as convolutional neural networks (CNNs), designed for grid-structured 2D data do not perform well on 3D point clouds. Some methods

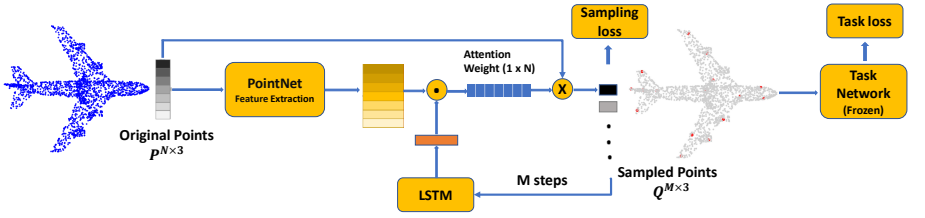

Figure 1: Overview of APSNet. APSNet first extracts features with a simplified PointNet that preserves the geometric information of a point cloud. Then, an LSTM with attention mechanism is used to capture the relationship among points and select the most informative point sequentially. Finally, the sampled point cloud is fed to a task network for prediction. The whole pipeline is optimized by minimizing a task loss and a sampling loss jointly.

attempt to first stiffly transform point clouds into grid-structured data and then take advantage of CNNs for feature extraction, such as projection-based methods [2, 27] and volumetric convolution-based methods [6, 14]. Because placing a point cloud on a regular grid generates an uneven number of points in grid cells, applying the same convolution operation on such grid cells leads to information loss in crowded cells and wasting computation in empty cells. Recently, many methods of directly processing point cloud [16, 24, 25, 36] have been proposed to enable efficient computation and performances in many applications, such as 3D point cloud classification [16, 24, 29, 32], semantic segmentation [15, 18, 28, 30, 31] and reconstruction [1, 8, 35, 38], have been improved significantly. These methods take raw point clouds as input (without quantization) and aggregate local features at the last stage of the network, so the accurate data locations are kept intact but the computation cost grows linearly with the number of points. However, processing large-scale dense 3D point clouds is still challenging due to the high cost of storing, transmitting and processing these data. Point cloud sampling, a task of selecting a subset of points to represent the original point clouds at a sparse scale, can reduce data redundancy and improve the efficiency of 3D data processing. So far, there are a few heuristics-based sampling methods, such as random sampling (RS), farthest point sampling (FPS) [5, 19], and grid (voxel) sampling [22, 33]. However, all of these methods are task-agnostic as they do not take into account the subsequent processing of the sampled points and may select non-informative points to the downstream tasks, leading to suboptimal performance. Recently, S-NET [4] and SampleNet [13] are proposed, which demonstrate that better sampling strategies can be learnt via a task-oriented sampling network. These sampling networks can generate a small number of samples that optimize the performance of a downstream task, and outperform traditional task-agnostic samplers significantly in various applications [4, 13].

We argue that point cloud sampling can be considered as a sequential generation process, in which points to be sampled next should depend on the points that have already been sampled. However, existing task-oriented sampling methods, such as S-NET [4] and SampleNet [13], do not pay enough attention to the sample dependency and generate all samples in one shot (without parameter reusing or sharing when generating samples of different sizes). In this paper, we propose an attention-based point cloud sampling network (APSNet) for task-oriented sampling, which enables a FPS-like sequential sampling but with a task-oriented objective. Specifically, APSNet employs a novel LSTM-based sequential model to capture the correlation of points with a global attention. The feature of each point is extracted by a simplified PointNet architecture, followed by an LSTM [11] with attention mechanism to capture the relationship of points and select the most informative ones for sampling. Fi-

nally, the sampled point cloud is fed to a (frozen) task network for prediction. The whole pipeline is fully differentiable and the parameters of APSNet can be trained by optimizing a task loss and a sampling loss jointly (See Fig. 1).

Depending on the availability of labeled training data, APSNet can be trained in supervised learning or self-supervised learning via knowledge-distillation [10]. In the latter case, no ground truth label is needed for the training of APSNet. Instead, the soft predictions of task network are leveraged to train APSNet. Interestingly, the self-supervised training of APSNet achieves impressive results that are close to the performance of supervised training. This makes APSNet widely applicable in situations where only a deployed task net is available but the original labeled training dataset of the task net is no longer accessible.

In addition, given the autoregressive model of APSNet, our method can generate arbitrary length of samples from a single model. This entails an effective joint training of APSNet with multiple sample sizes, resulting in a single compact model for point cloud sampling, while S-NET and SampleNet require a growing model size to generate larger sized point samples and the parameter reusing or sharing is not as effective as APSNet [4, 13]. Our main contributions are summarized as follows:

1. We propose APSNet, a novel attention-based point cloud sampling network, which enables a FPS-like sequential sampling with a task-oriented objective.
2. APSNet can be trained in supervised learning or self-supervised learning via knowledge-distillation, while the latter requires no ground truth labels for training and is thus widely applicable in situations where only a deployed task network is available.
3. APSNet can be jointly trained with multiple sample sizes, yielding a single compact model that can generate arbitrary length of samples with prominent performance.
4. Compared with state-of-the-art sampling methods, APSNet demonstrate superior performance on various 3D point cloud applications.

2 Related Work

Deep Learning on Point Clouds Following the breakthrough results of CNNs in 2D image processing tasks [9, 12], there has been a strong interest of adapting such methods to 3D geometric data. Compared to 2D images, point clouds are sparse, unordered and locality-sensitive, making it non-trivial to adapt CNNs to point cloud processing. Early attempts focus on regular representations of the data in the form of 3D voxels [22, 33]. These methods quantize point clouds into regular voxels in 3D space with a predefined resolution, and then apply volumetric convolution. More recently, some works explore new designs of local aggregation operators on point clouds to process point sets efficiently and reduce the loss of details [23, 24, 31]. PointNet [23] is a pioneering deep network architecture that directly processes point clouds for classification and semantic segmentation; it proposes a shared multi-layer perception (MLP), followed by a max-pooling layer, to approximate continuous set functions to deal with unordered point sets. PointNet++ [24] further proposes a hierarchical aggregation of point features to extract global features. In later works, DGCNN [31] proposes an effective EdgeConv operator that encodes the point relationships as edge features to better capture local geometric features of point clouds while still maintaining permutation invariance. In this paper, we leverage a simplified PointNet architecture to extract local features from a point cloud before feeding it to an attention-based LSTM for point cloud sampling.

Point Cloud Sampling Random sampling (RS) selects a set of points randomly from a point cloud and has the smallest computation overhead. But this method is sensitive to density imbalance issue [13]. Farthest point sampling (FPS) [5, 19] has been widely used as a pooling operator in point cloud processing. It starts from a randomly selected point in the set and iteratively selects the next point from the point cloud that is the farthest from the selected points, such that the sampled points can achieve a maximal coverage of the input point cloud. Recently, S-NET [4] and SampleNet [13] have demonstrated that better sampling strategies can be learnt by a sampling network. These methods aim to generate a small set of samples that optimize the performance of a downstream task. Moreover, the generated 3D coordinates can be pushed towards a subset of original points to minimize the training loss if a matched point set is desired. Both methods treat the sampling process as a generation task and produce all the points in one shot, which does not pay sufficient attention to sample dependency, and leads to suboptimal performances. Our APSNet is a combination of FPS and task-oriented sampling in the sense that points are sampled sequentially with a task-oriented objective.

Knowledge Distillation As one of the popular model compression techniques, knowledge distillation [10] is inspired by knowledge transfer from teachers to students. Its key strategy is to orientate compact student models to approximate over-parameterized teacher models such that student models can achieve the performances that are close to (sometimes even higher than) those of teachers'. Different from traditional knowledge distillation, which forces student networks to approximate the soft prediction of pre-trained teacher networks, self distillation [37] distills knowledge within a network itself from its own soft predictions. Our APSNet can be trained both in supervised learning and self-supervised learning via knowledge distillation. In the former case, labeled training data are required to train APSNet, while in the latter case the soft predictions of task network can be used to train APSNet such that the sampled point clouds from APSNet can achieve similar predictions as the original point clouds.

3 The Proposed Method

The overview of our proposed APSNet is depicted in Fig. 1, which contains two main components: (a) A simplified PointNet for feature extraction, (b) An LSTM with attention mechanism for sequential point sampling. In this section, we first describe the details of these components and then discuss different approaches to train APSNet.

Notation and Problem Statement Let $\boldsymbol{P} = \{\boldsymbol{p}_i \in \mathbb{R}^3\}_{i=1}^{n}$ denote a point cloud that contains n points, with $\boldsymbol{p}_i = [x_i, y_i, z_i]$ representing the 3D coordinates of point i. We consider two types of point cloud samples: (1) $\boldsymbol{Q}^* = \{\boldsymbol{q}_i^* \in \mathbb{R}^3\}_{i=1}^{m}$ denotes a **sampled** point cloud of m points that is a subset of $\boldsymbol{P}$ with $m < n$, i.e., $\boldsymbol{Q}^* \subset \boldsymbol{P}$. (2) $\boldsymbol{Q} = \{\boldsymbol{q}_i \in \mathbb{R}^3\}_{i=1}^{m}$ denotes a **generated** point cloud of m points that may not be a subset of $\boldsymbol{P}$. Typically, we can convert $\boldsymbol{Q}$ to $\boldsymbol{Q}^*$ by a matching process, i.e., match each point in $\boldsymbol{Q}$ to its nearest point in $\boldsymbol{P}$, and then replace the duplicated points[1] in resulting $\boldsymbol{Q}^*$ by FPS. Without loss of generality, in the following we present our algorithm in terms of $\boldsymbol{Q}$ since $\boldsymbol{Q}$ is more general than $\boldsymbol{Q}^*$ and can be converted to $\boldsymbol{Q}^*$ by matching. Moreover, let $f_{\boldsymbol{\theta}}(\cdot) : \boldsymbol{P} \rightarrow \boldsymbol{Q}$ denote APSNet with the parameters $\boldsymbol{\theta}$.

[1] Multiple points in $\boldsymbol{Q}$ can be mapped to the same point in $\boldsymbol{P}$.

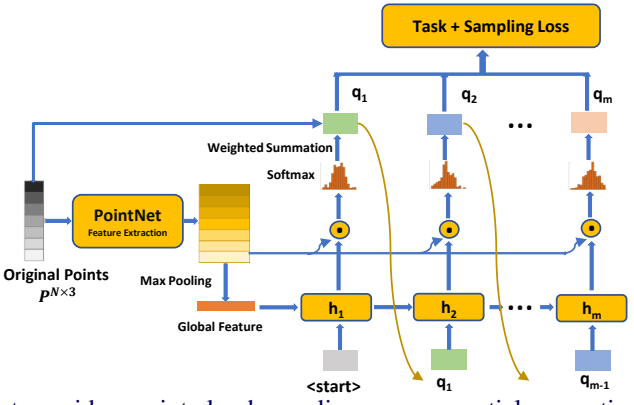

Figure 2: APSNet considers point cloud sampling as a sequential generation process with a task-oriented objective, and uses an LSTM with attention mechanism for sampling.

As discussed in the introduction, we are interested in task-oriented sampling that yields a small set of points $\boldsymbol{Q}$ to optimize a downstream task represented by a well-trained deployed task network T, where T can be a model for 3D point cloud classification, reconstruction or registration, etc. Given $\boldsymbol{P}$, the goal of APSNet is to generate a point cloud $\boldsymbol{Q} = f_{\boldsymbol{\theta}}(\boldsymbol{P})$ that maximizes the predictive performance of task network T. Specifically, the parameters of APSNet, $\boldsymbol{\theta}$, is optimized by minimizing a task loss and a sampling loss jointly as

$$\min_{\boldsymbol{\theta}} \ell_{task}(T(\boldsymbol{Q}), y) + \lambda L_{sample}(\boldsymbol{Q}, \boldsymbol{P}), \tag{1}$$

whose details are to be discussed in Sec. 3.2.

3.1 Attention-based Point Cloud Sampling

Existing task-oriented sampling methods, such as S-NET [4] and SampleNet [13], formulate the sampling process as a point cloud generation problem from a global feature vector, and generate all m points in one shot. We argue that the sampling process is naturally a sequential generation process, in which points to be sampled next should depend on the points that have already been sampled. We therefore propose APSNet, an attention-based LSTM for sequential point sampling in order to capture the relationship among points.

The overall architecture of APSNet is depicted in Fig. 2. First, APSNet takes the original point cloud $\boldsymbol{P}$ as input and samples from $\boldsymbol{P}$ via an LSTM with attention mechanism to produce a small point cloud $\boldsymbol{Q}$ of m points, each point of which is a soft point generated by projecting $\boldsymbol{P}$ on a set of attention coefficients from the LSTM. Finally, the output of APSNet, $\boldsymbol{Q}$, is fed to a well-trained deployed task network T for prediction and task loss evaluation[2].

The first step is to extract a feature representation for each point in $\boldsymbol{P}$. APSNet follows the architecture of PointNet [23], a basic feature extraction backbone on 3D point clouds, to extract point-wise local features. Specifically, a set of 1×1 convolution layers are applied to the original point cloud $\boldsymbol{P}$ and produce a set of d-dimensional point-wise feature vectors, denoted by $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n]^\mathsf{T} \in \mathbb{R}^{n \times d}$. Then, a symmetric feature-wise max pooling operator is applied to $\boldsymbol{X}$ and produce a global feature vector $\boldsymbol{g} \in \mathbb{R}^d$, which is then fed to an LSTM as the initial state for sequential point generation.

[2]The parameters of T is frozen during the training of APSNet.

The sequential point generation process is similar to the attention-based sequential model for image captioning [34]. Given initial state $\boldsymbol{g}$ and <start> as inputs, the LSTM updates its hidden state to $\boldsymbol{h}_t \in \mathbb{R}^d$ at each time step $t = 1, 2, \cdots, m$. The hidden state $\boldsymbol{h}_t$ encodes the history of all the sampled points and is indicative for APSNet to identify the next most informative point of $\boldsymbol{P}$ to sample. To achieve this, a set of attention scores is calculated as the dot product of the hidden state $\boldsymbol{h}_t$ and point-wise feature vector $\boldsymbol{x}_i$ for $i = 1, 2, \cdots, n$, followed by a softmax for normalization:

$$s_{it} = \boldsymbol{x}_i \cdot \boldsymbol{h}_t, \qquad a_{it} = \frac{\exp(s_{it})}{\sum_i \exp(s_{it})}. \tag{2}$$

The attention coefficients a_{it} indicates the importance of point i at the sampling step t, from which sampled point $\boldsymbol{q}_t \in \mathbb{R}^3$ can be generated as a weighted sum of all the points in $\boldsymbol{P}$:

$$\boldsymbol{q}_t = \sum_i a_{it} \cdot \boldsymbol{p}_i. \tag{3}$$

The generated point $\boldsymbol{q}_t$ is then fed to the LSTM as input for the next time step to generate the next point until all m points are generated. During sequential generation process, the attention mechanism enables the model to attend to all the points in $\boldsymbol{P}$ and identify the most informative "point" to sample.

3.2 Training of APSNet

APSNet is a task-oriented sampling network, which can be trained to optimize its performance on the downstream tasks of interest. In this section, we discuss the objective functions that can be used to train APSNet. Depending on the availability of labeled training data and deployment requirements, we consider three different approaches to train APSNet: (1) supervised training, (2) self-supervised training, and (3) joint training.

3.2.1 Training with or without Ground Truth Labels

We consider two training scenarios: (a) both task network T and a labeled training set $\{\boldsymbol{P}_j, y_j\}_{j=1}^N$ are available; (b) only task network T and some input point clouds $\{\boldsymbol{P}_j\}_{j=1}^N$ are available, but no labels is provided. The latter case corresponds to the situation where original labeled training data of T is no longer available for the development of APSNet.

Supervised Training When a labeled training set $\{\boldsymbol{P}_j, y_j\}_{j=1}^N$ is available, we can train APSNet in a supervised learning paradigm. Similar to S-NET [4] and SampleNet [13], two types of losses are exploited to train APSNet, i.e., the task loss ℓ_{task} and the sampling loss L_{sample}. Specifically, the task loss measures the quality of predictions based on the sampled point cloud $\boldsymbol{Q}$:

$$\ell_{task}(T(\boldsymbol{Q}), y), \tag{4}$$

where y is the ground-truth label of $\boldsymbol{P}$. For different downstream tasks, y can be the class label or the original point cloud $\boldsymbol{P}$ when the task is for classification or reconstruction, respectively. Accordingly, the corresponding task loss $\ell_{task}()$ is defined differently, e.g., the cross-entropy loss for classification or the Chamfer distance for reconstruction [1].

The sampling loss L_{sample} encourages the sampled points in $\boldsymbol{Q}$ to be close to those of $\boldsymbol{P}$ and also have a maximal coverage w.r.t. the original point cloud $\boldsymbol{P}$. We found that this sampling loss provides an important prior knowledge for sampling, and is critical for APSNet

to achieve a good performance. Specifically, given two point sets $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, denoting average nearest neighbor loss as:

$$L_a(\boldsymbol{S}_1,\boldsymbol{S}_2)=\frac{1}{|\boldsymbol{S}_1|}\sum_{s_1\in\boldsymbol{S}_1}\min_{s_2\in\boldsymbol{S}_2}||s_1-s_2||_2^2, \tag{5}$$

and maximal nearest neighbor loss as:

$$L_m(\boldsymbol{S}_1,\boldsymbol{S}_2)=\max_{s_1\in\boldsymbol{S}_1}\min_{s_2\in\boldsymbol{S}_2}||s_1-s_2||_2^2, \tag{6}$$

the sampling loss is then given by:

$$L_{sample}(\boldsymbol{Q},\boldsymbol{P})=L_a(\boldsymbol{Q},\boldsymbol{P})+\beta L_m(\boldsymbol{Q},\boldsymbol{P})+(\gamma+\delta|\boldsymbol{Q}|)L_a(\boldsymbol{P},\boldsymbol{Q}), \tag{7}$$

where β, γ and δ are the hyperparameters that balance the contributions from different loss components.

Putting all the components together, the total loss of APSNet is given by:

$$L_{total}=\ell_{task}(T(\boldsymbol{Q}),y)+\lambda L_{sample}(\boldsymbol{Q},\boldsymbol{P}), \tag{8}$$

where λ is a hyperparameter that balances the contribution between the task loss and the sampling loss.

Self-supervised Training with Knowledge Distillation In some practical scenarios, we may only have task network T and some input point clouds $\{\boldsymbol{P}_j\}_{j=1}^N$ at our disposal. This is the situation where a deployed task network T is available, but the original labeled training data of T is no longer available for the development of APSNet. In this case, we propose to train APSNet via self-supervised learning based on the idea of knowledge distillation [10, 37]. In knowledge distillation, we can transfer the knowledge from a teacher network to a student network such that the student network can yield a similar prediction as the teacher network while being much more efficient. Inspired by knowledge distillation, we treat the task network T as the teacher model, and APSNet as the student model and use the soft predictions of T as the targets to train APSNet. Specifically, the task loss for self-supervised training of APSNet is updated to

$$\ell_{task}(T(\boldsymbol{Q}),\tilde{y}),\ \text{with}\ \tilde{y}=T(\boldsymbol{P}), \tag{9}$$

where $\tilde{y}$ is the soft prediction of T given the original point cloud $\boldsymbol{P}$, and the loss function ℓ_{task} is defined differently for different downstream tasks. The goal of the new loss function is to generate a sampled point cloud $\boldsymbol{Q}$ that can yield a similar prediction as the original $\boldsymbol{P}$.

Similar idea is also explored in [3], where mutual information between the predictions of backbone network from sparsified input and original input are maximized for model interpretation, while here we sparsify point clouds.

3.2.2 Joint Training

APSNet described above is trained for a specified sample size m. Given the autoregressive model of our method, APSNet can generate arbitrary length of samples from a single model. This entails an effective joint training of APSNet with multiple different sample sizes, resulting in a single compact model to generate arbitrary sized point clouds with prominent performance. Specifically, we can train one APSNet with different sample sizes by:

$$L_{joint}=\sum_{c\in C_s}\Big(\ell_{task}(T(\boldsymbol{Q}_c),y)+\lambda L_{sample}(\boldsymbol{Q}_c,\boldsymbol{P})\Big), \tag{10}$$

where C_s is a set of sample sizes of interest. In our experiments, we set $C_s = \{2^l\}_{l=3}^{7}$.

S-NET [4] and SampleNet [13] propose a progressive training to train a sampling network to generate different sized point clouds. However, their model sizes grow linearly as the sample size m increases. In contrast, due to the autoregressive model of APSNet, we can train one single compact model (with a fixed number of parameters) to generate arbitrary sized point clouds without incurring a linear increase of model parameters. This entails a better parameter reusing or sharing for APSNet, and leads to improved sample efficiency as compared to S-NET and SampleNet.

4 Experiments

We demonstrate the performance of APSNet on three different applications of 3D point clouds for classification, reconstruction, and registration. For the purpose of comparison, random sampling (RS), farthest point sampling (FPS) and SampleNet [13] are used as baselines, where SampleNet is the state-of-the-art task-oriented sampling method. We consider two variants of APSNet: (1) *APSNet*, and (2) *APSNet-KD*, while the former refers to the supervised training of APSNet and the latter refers to the self-supervised training of APSNet with knowledge distillation. A trained APSNet generates point cloud $\boldsymbol{Q}$ that isn't a subset of original input point cloud $\boldsymbol{P}$, but the generated $\boldsymbol{Q}$ can be converted to $\boldsymbol{Q}^*$ by a matching process as discussed in Sec. 3.1. Therefore, we further distinguish them as *APSNet-G* and *APSNet-M*, respectively. SampleNet [13] also generates point cloud $\boldsymbol{Q}$, which is converted to $\boldsymbol{Q}^*$ by the matching process. Similarly, we denote them as *SampleNet-G* and *SampleNet-M*, respectively. In our experiments, we compare the performances of all these variants. However, we would like to emphasize that the default SampleNet is *SampleNet-M*, while the default APSNet is *APSNet-G* since APSNet-G yields the best predictive performance without an expensive matching process as we will demonstrate in the experiments.

Since our APSNet is implemented in PyTorch, we convert the official TensorFlow implementation of SampleNet[3] to PyTorch for a fair comparison. We found that our PyTorch implementation achieves better performances than the official TensorFlow version in most of our experiments. Details of experimental settings and implementation are relegated to supplementary material. All our experiments are performed on Nvidia RTX GPUs. Our source code can be found at https://github.com/Yangyeeee/APSNet.

4.1 3D Point Cloud Classification

We use the point clouds of 1024 points that were uniformly sampled from the ModelNet40 dataset [33] to train PointNet [23] (the task network T). The official train-test split is used for the training and evaluation, and the instance-wise accuracy is used as the evaluation metric for performance comparison. The vanilla task network achieves an accuracy of 90.1% with all the 1024 points. We execute different sampling methods with a variety of sample sizes and report their performances for comparison.

Table 1 reports the classification accuracies of all the five sampling methods. To validate our PyTorch implementation of SampleNet, we also include the official SampleNet results as reported in [13]. It can be observed that our PyTorch implementation outperforms the official TensorFlow code consistently; in particular, when sample size $m = 8$, our implementation

[3] https://github.com/itailang/SampleNet

Table 1: Classification accuracies of five sampling methods with different sample sizes m on ModelNet40. M* denotes the official results from SampleNet [13].

	RS	FPS	DaNet	MOPS-Net [26]		SampleNet			APSNet		APSNet-KD	
m			[17]	G	M	G	M	M*	G	M	G	M
8	8.26	23.29	-	-	-	78.36	73.31	28.7	**81.42**	74.12	80.22	73.81
16	25.11	54.19	-	**84.7**	51.2	80.60	79.68	55.5	83.89	82.25	83.82	82.02
32	55.19	77.32	85.1	86.1	77.6	80.32	82.97	74.4	88.15	86.97	**88.76**	84.95
64	78.26	87.22	86.8	87.1	81.0	79.36	84.01	79.0	88.38	87.58	**88.66**	87.54
128	85.95	88.76	86.8	87.2	85.0	85.52	87.17	79.7	89.22	**89.38**	87.83	88.01
256	88.80	89.30	87.2	87.4	86.7	87.43	89.58	83.4	89.54	**89.86**	88.02	88.21
512	89.66	89.87	-	88.3	88.3	88.01	90.18	88.2	89.78	**90.18**	88.69	88.56

has a gain of nearly 45% over the official code. Therefore, for a fair comparison, we compare APSNet mainly with our improved SampleNet.

A few notable observations can be made from Table 1. (1) As sample size m increases, all the sampling methods have improved accuracies. The performances of task-oriented samplers, e.g., SampleNet and APSNet, outperform those of task-agnostic samplers, e.g., random sampling and FPS, consistently. However, the gains are getting smaller as sample size m increases; when $m = 512$, all sampling methods achieves a comparable accuracy that is close to the best accuracy (90.1%) achieved with all the 1024 points. (2) In general, SampleNet-M achieves a better performance than SampleNet-G. When sample size m increases, the gain is more pronounced. (3) In contrast, APSNet-G achieves a better performance than APSNet-M. When sample size m is small, the gain is large, while as sample size m increases, both variants of APSNet have very similar performances. This is likely because the downstream task networks are trained with original points P, and the matched Q^* from APSNet-M can fit better to the downstream task networks. The gains are getting smaller because when sampling ratio becomes larger the performance is approaching to the upper bound which uses all the points. (4) Comparing APSNet-G with SampleNet-M (the best defaults for both algorithms), APSNet outperforms SampleNet consistently; especially when $m \leq 128$, we observe a 2% to 8% accuracy gain, demonstrating the effectiveness of APSNet. (5) APSNet-KD achieves a very impressive result without utilizing labeled point clouds for training; its performance is almost on-par with APSNet that is trained with labeled point clouds.

Discussion The above experiments show that SampleNet-M outperforms SampleNet-G consistently, while the opposite is observed for APSNet. This can be explained by the limitations of SampleNet as we indicated in the introduction. As sample size m increases, SampleNet has a higher probability of generating similar (redundant) points due to the one-shot generation of m samples. Since these redundant points cannot improve the classification accuracy effectively, the matching process (which replaces the redundant points with the FPS samples) becomes critical for SampleNet-M to improve its performance, leading to improved performances over SampleNet-G. On the other hand, APSNet-G generates the next sample depending heavily on previously sampled points, and therefore is able to capture the relationship among points and generate more informative samples, yielding a better performance without an expensive matching process.

Joint Training Next, we investigate the joint training of APSNet, and compare it with separated training of ASPNet and SampleNet for each sample size m. For the joint training of APSNet, we train a single compact model of APSNet with $C_s = \{8, 16, 32, 64, 128\}$ by optimizing the joint loss (10). In contrast, in separated training of SampleNet or APSNet, a

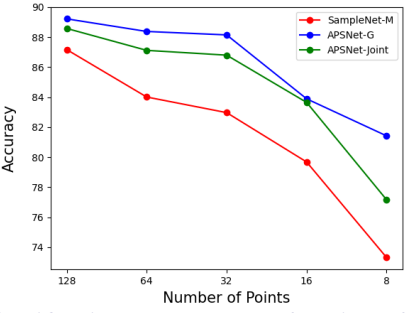

Figure 3: Evolution of classification accuracy as a function of sample size m for different sampling methods. APSNet-Joint uses a single model to generate different number of samples, while SampleNet and APSNet use separately trained models to generate each specific number of samples.

separated model is trained for each sample size $m \in C_s$ and its performance is evaluated for the specific m it was trained with.

Fig. 3 shows the performance comparison between joint training of APSNet and separated training. It can be observed a single model trained by APSNet-Joint can generate arbitrary length of samples with competitive performances. Indeed, the performance of APSNet-Joint is lower than the separately trained APSNet, but it still consistently outperforms separately trained SampleNet.

4.2 Additional Experiments

Due to page constraints, additional experiments of (1) point cloud sampling for reconstruction, (2) registration, (3) ablation study of the sampling loss (7), (4) inference speed comparison, and (5) visualization of attention coefficients are relegated to supplementary material.

5 Conclusion

This paper introduces APSNet, an attention-based sampling network for point cloud sampling. Compared to S-Net and SampleNet, which formulate the sampling process as an one-shot generation task with MLPs, APSNet employs a sequential autoregressive generation with a novel LSTM-based sequential model for sampling. Depending on the availability of labeled training data, APSNet can be trained in supervised learning or self-supervised learning via knowledge distillation. We also present a joint training of APSNet, yielding a single compact model that can generate arbitrary length of samples with prominent performances. Extensive experiments demonstrate the superior performance of APSNet over the state-of-the-arts both in terms of sample quality and inference speed, which make APSNet widely applicable in many practical application scenarios.

6 Acknowledgements

We would like to thank the anonymous reviewers for their comments and suggestions, which helped improve the quality of this paper. We would also gratefully acknowledge the support of Cisco Systems, Inc. for its university research fund to this research.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning Representations and Generative Models For 3D Point Clouds. Proceedings of the 35th International Conference on Machine Learning (ICML), pages 40–49, 2018.

[2] Jorge Beltrán, Carlos Guindel, Francisco Miguel Moreno, Daniel Cruzado, Fernando García, and Arturo De La Escalera. Birdnet: A 3d object detection framework from lidar information. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pages 3517–3523, 2018. doi: 10.1109/ITSC.2018.8569311.

[3] Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. Learning to Explain: An Information-Theoretic Perspective on Model Interpretation. Proceedings of the International Conference on Machine Learning (ICML), 2018.

[4] Oren Dovrat, Itai Lang, and Shai Avidan. Learning to Sample. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 2760–2769, 2019.

[5] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Y. Yehoshua Zeevi. The Farthest Point Strategy for Progressive Image Sampling. IEEE Transactions on Image Processing, 6:1305–1315, 1997.

[6] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 1355–1361, 2017.

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In 2012 IEEE Conference on Computer Vision and Pattern Recognition, pages 3354–3361, 2012.

[8] Zhizhong Han, Xiyang Wang, Yu-Shen Liu, and Matthias Zwicker. Multi-Angle Point Cloud-VAE: Unsupervised Feature Learning for 3D Point Clouds from Multiple Angles by Joint Self-Reconstruction and Half-to-Half Prediction. Proceedings of the International Conference on Computer Vision (ICCV), 2019.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2016.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. Proceedings of Advances in Neural Information Processing Systems (NIPS), 2014.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems (NeurIPS), 2012.

[13] 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 (CVPR), pages 7578–7588, 2020.

[14] Bo Li. 3d fully convolutional network for vehicle detection in point cloud. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1513–1518, 2017.

[15] Jiaxin Li, Ben M Chen, and Gim Hee Lee. SO-Net: Self-Organizing Network for Point Cloud Analysis. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 9397–9406, 2018.

[16] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution On X-Transformed Points. Proceedings of Advances in Neural Information Processing Systems (NeuralIPS), 2018.

[17] Yanan Lin, Yan Huang, Shihao Zhou, Mengxi Jiang, Tianlong Wang, and Yunqi Lei. Da-net: Density-adaptive downsampling network for point cloud classification via end-to-end learning. In 2021 4th International Conference on Pattern Recognition and Artificial Intelligence (PRAI), pages 13–18, 2021. doi: 10.1109/PRAI53619.2021.9551070.

[18] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-Shape Convolutional Neural Network for Point Cloud Analysis. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 8895–8904, 2019.

[19] Carsten Moenning and Neil A. Dodgson. Fast Marching farthest point sampling. Eurographics Poster Presentation, 2003.

[20] Andreas Nüchter and Joachim Hertzberg. Towards semantic maps for mobile robots. Robotics Auton. Syst., 56:915–926, 2008.

[21] Youngmin Park, Vincent Lepetit, and Woontack Woo. Multiple 3d object tracking for augmented reality. In 2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, pages 117–120, 2008.

[22] Charles R. Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and Multi-View CNNs for Object Classification on 3D Data. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5648–5656, 2016.

[23] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 652–660, 2017.

[24] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. Proceedings of Advances in Neural Information Processing Systems (NeuralIPS), 2017.

[25] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep Hough Voting for 3D Object Detection in Point Clouds. Proceedings of the International Conference on Computer Vision (ICCV), 2019.

[26] Yue Qian, Junhui Hou, Qijian Zhang, Yiming Zeng, Sam Kwong, and Ying He. Mops-net: A matrix optimization-driven network fortask-oriented 3d point cloud downsampling. arXiv preprint arXiv:2005.00383, 2020.

[27] Martin Simony, Stefan Milzy, Karl Amendey, and Horst-Michael Gross. Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, September 2018.

[28] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2530–2539, 2018.

[29] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2019.

[30] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. SGPN: Similarity Group Proposal Network for 3D Point Cloud InstanceSegmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2569–2578, 2018.

[31] 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), 2019.

[32] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep Convolutional Networks on 3D Point Clouds. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 9622–9630, 2019.

[33] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A Deep Representation for Volumetric Shapes. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1912–1920, 2015.

[34] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In Proceedings of the 32nd International Conference on Machine Learning, 2015.

[35] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. FoldingNet: Point Cloud Auto-encoder via Deep Grid Deformation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 206–215, 2018.

[36] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng Ann Heng. PU-Net: Point Cloud Upsampling Network. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2790–2799, 2018.

[37] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. Proceedings of the International Conference on Computer Vision (ICCV), 2019.

[38] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 3D Point-Capsule Networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1009–1018, 2019.