SampleNet: Differentiable Point Cloud Sampling

Itai Lang
Tel Aviv University
itailang@mail.tau.ac.il

Asaf Manor
Tel Aviv University
asafmanor@mail.tau.ac.il

Shai Avidan
Tel Aviv University
avidan@eng.tau.ac.il

Abstract

There is a growing number of tasks that work directly on point clouds. As the size of the point cloud grows, so do the computational demands of these tasks. A possible solution is to sample the point cloud first. Classic sampling approaches, such as farthest point sampling (FPS), do not consider the downstream task. A recent work showed that learning a task-specific sampling can improve results significantly. However, the proposed technique did not deal with the non-differentiability of the sampling operation and offered a workaround instead.

We introduce a novel differentiable relaxation for point cloud sampling that approximates sampled points as a mixture of points in the primary input cloud. Our approximation scheme leads to consistently good results on classification and geometry reconstruction applications. We also show that the proposed sampling method can be used as a front to a point cloud registration network. This is a challenging task since sampling must be consistent across two different point clouds for a shared downstream task. In all cases, our approach outperforms existing non-learned and learned sampling alternatives. Our code is publicly available[1].

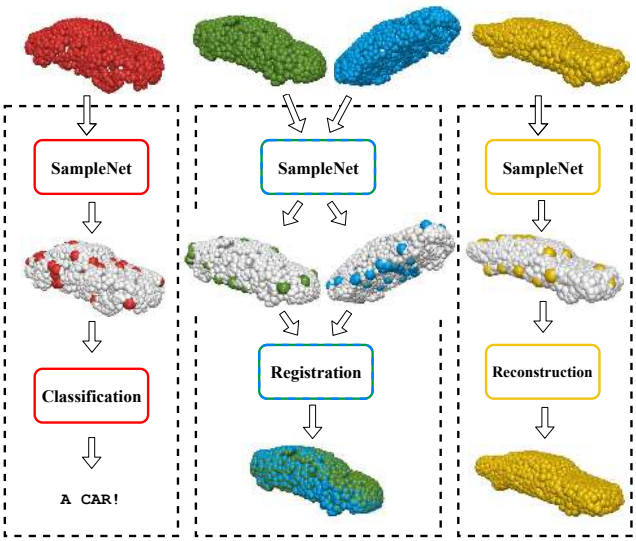

Figure 1. **Applications of SampleNet.** Our method learns to sample a point cloud for a subsequent task. It employs a differentiable relaxation of the selection of points from the input point cloud. SampleNet lets various tasks, such as classification, registration, and reconstruction, to operate on a small fraction of the input points with minimal degradation in performance.

1. Introduction

The popularity of 3D sensing devices increased in recent years. These devices usually capture data in the form of a point cloud - a set of points representing the visual scene. A variety of applications, such as classification, registration and shape reconstruction, consume the raw point cloud data. These applications can digest large point clouds, though it is desirable to reduce the size of the point cloud (Figure 1) to improve computational efficiency and reduce communication costs.

This is often done by sampling the data before running the downstream task [8, 11, 12]. Since sampling preserves the data structure (i.e., both input and output are point clouds), it can be used natively in a process pipeline. Also, sampling preserves data fidelity and retains the data in an interpretable representation.

An emerging question is how to select the data points. A widely used method is farthest point sampling (FPS) [30, 52, 18, 27]. FPS starts from a point in the set, and iteratively selects the farthest point from the points already selected [7, 23]. It aims to achieve a maximal coverage of the input.

FPS is task agnostic. It minimizes a geometric error and does not take into account the subsequent processing of the sampled point cloud. A recent work by Dovrat *et al*. [6] presented a task-specific sampling method. Their key idea was to simplify and then sample the point cloud. In the first step, they used a neural network to produce a small set of simplified points in the ambient space, optimized for the task. This set is not guaranteed to be a subset of the

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

input. Thus, in a post-processing step, they matched each simplified point to its nearest neighbor in the input point cloud, which yielded a subset of the input.

This learned sampling approach improved application performance with sampled point clouds, in comparison to non-learned methods, such as FPS and random sampling. However, the matching step is a non-differentiable operation and can not propagate gradients through a neural network. This substantially compromises the performance with sampled points in comparison to the simplified set, since matching was not introduced at the training phase.

We extend the work of Dovrat *et al*. [6] by introducing a differentiable relaxation to the matching step, i.e., nearest neighbor selection, during training (Figure 2). This operation, which we call *soft projection*, replaces each point in the simplified set with a weighted average of its nearest neighbors from the input. During training, the weights are optimized to approximate the nearest neighbor selection, which is done at inference time.

The soft projection operation makes a change in representation. Instead of absolute coordinates in the free space, the projected points are represented in weight coordinates of their local neighborhood in the initial point cloud. The operation is governed by a temperature parameter, which is minimized during the training process to create an annealing schedule [38]. The representation change renders the optimization goal as multiple localized classification problems, where each simplified point should be assigned to an optimal input point for the subsequent task.

Our method, termed SampleNet, is applied to a variety of tasks, as demonstrated in Figure 1. Extensive experiments show that we outperform the work of Dovrat *et al*. consistently. Additionally, we examine a new application - registration with sampled point clouds and show the advantage of our method for this application as well. Registration introduces a new challenge: the sampling algorithm is required to sample consistent points across two different point clouds for a common downstream task. To summarize, our key contributions are threefold:

- A novel differentiable approximation of point cloud sampling;
- Improved performance with sampled point clouds for classification and reconstruction tasks, in comparison to non-learned and learned sampling alternatives;
- Employment of our method for point cloud registration.

2. Related Work

Deep learning on point clouds Early research on deep learning for 3D point sets focused on regular representations of the data, in the form of 2D multi-views [29, 35] or 3D voxels [44, 29]. These representations enabled the natural extension of successful neural processing paradigms

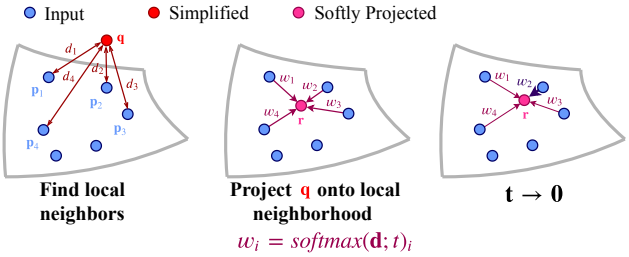

Figure 2. **Illustration of the sampling approximation.** We propose a learned sampling approach for point clouds that employs a differentiable relaxation to nearest neighbor selection. A query point $\mathbf{q}$ (in Red) is projected onto its local neighborhood from the input point cloud (in Blue). A weighted average of the neighbors form a softly projected point $\mathbf{r}$ (in Magenta). During training the weights are optimized to approximated nearest neighbor sampling ($\mathbf{p}_2$ in this example), which occurs at inference time.

from the 2D image domain to 3D data. However, point clouds are irregular and sparse. Regular representations come with the cost of high computational load and quantization errors.

PointNet [28] pioneered the direct processing of raw point clouds. It includes per point multi-layer perceptrons (MLPs) that lift each point from the coordinate space to a high dimensional feature space. A global pooling operation aggregates the information to a representative feature vector, which is mapped by fully connected (FC) layers to the object class of the input point cloud.

The variety of deep learning applications for point clouds expanded substantially in the last few years. Today, applications include point cloud classification [30, 18, 36, 43], part segmentation [15, 34, 21, 42], instance segmentation [40, 19, 41], semantic segmentation [13, 25, 39], and object detection in point clouds [27, 33]. Additional applications include point cloud autoencoders [1, 48, 10, 54], point set completion [53, 5, 31] and registration [2, 22, 32], adversarial point cloud generation [14, 46], and adversarial attacks [20, 45]. Several recent works studied the topic of point cloud consolidation [52, 51, 16, 49]. Nevertheless, little attention was given to sampling strategies for point sets.

Nearest neighbor selection Nearest neighbor (NN) methods have been widely used in the literature for information fusion [9, 30, 26, 42]. A notable drawback of using nearest neighbors, in the context of neural networks, is that the selection rule is non-differentiable. Goldberger *et al*. [9] suggested a stochastic relaxation of the nearest neighbor rule. They defined a categorical distribution over the set of candidate neighbors, where the 1-NN rule is a limit case of the distribution.

Later on, Plötz and Roth [26] generalized the work of Goldberger *et al*., by presenting a deterministic relaxation of the k nearest neighbor (KNN) selection rule. They pro-

posed a neural network layer, dubbed neural nearest neighbors block, that employs their KNN relaxation. In this layer, a weighted average of neighbors in the features space is used for information propagation. The neighbor weights are scaled with a temperature coefficient that controls the uniformity of the weight distribution. In our work, we employ the relaxed nearest neighbor selection as a way to approximate point cloud sampling. While the temperature coefficient is unconstrained in the work of Plötz and Roth, we promote a small temperature value during training, to approximate the nearest neighbor selection.

Sampling methods for points clouds in neural networks Farthest point sampling (FPS) has been widely used as a pooling operation in point cloud neural processing systems [30, 27, 50]. However, FPS does not take into account the further processing of the sampled points and may result in sub-optimal performance. Recently, alternative sub-sampling methods have been proposed [17, 24, 47]. Nezhadarya *et al*. [24] introduced a critical points layer, which passes on points with the most active features to the next network layer. Yang *et al*. [47] used Gumbel subset sampling during the training of a classification network instead of FPS, to improve its accuracy. The settings of our problem are different though. Given an application, we sample the input point cloud and apply the task on the sampled data.

Dovrat *et al*. [6] proposed a learned task-oriented simplification of point clouds, which led to a performance gap between train and inference phases. We mitigate this problem by approximating the sampling operation during training, via a differentiable nearest neighbor approximation.

3. Method

An overview of our sampling method, SampleNet, is depicted in Figure 3. First, a task network is pre-trained on complete point clouds of n points and frozen. Then, SampleNet takes a complete input P and simplifies it via a neural network to a smaller set Q of m points [6]. Q is soft projected onto P by a differentiable relaxation of nearest neighbor selection. Finally, the output of SampleNet, R, is fed to the task.

SampleNet is trained with three loss terms:

$$\mathcal{L}_{total}^{samp} = \mathcal{L}_{task}(R) + \alpha\mathcal{L}_{simplify}(Q,P) + \lambda\mathcal{L}_{project}. \quad (1)$$

The first term, $\mathcal{L}_{task}(R)$, optimizes the approximated sampled set R to the task. It is meant to preserve the task performance with sampled point clouds. $\mathcal{L}_{simplify}(Q,P)$ encourages the simplified set to be close to the input. That is, each point in Q should have a close point in P and vice-versa. The last term, $\mathcal{L}_{project}$ is used to approximate the sampling of points from the input point cloud by the soft projection operation.

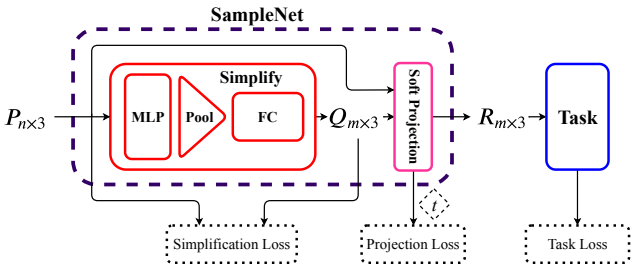

Figure 3. **Training of the proposed sampling method.** The task network trained on complete input point clouds P and kept fixed during the training of our sampling network SampleNet. P is simplified with a neural network to a smaller set Q. Then, Q is softly projected onto P to obtain R, and R is fed to the task network. Subject to the denoted losses, SampleNet is trained to sample points from P that are optimal for the task at hand.

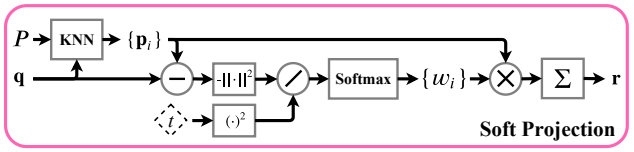

Figure 4. **The soft projection operation.** The operation gets as input the point cloud P and the simplified point cloud Q. Each point $\mathbf{q} \in Q$ is projected onto its k nearest neighbors in P, denoted as $\{\mathbf{p}_i\}$. The neighbors $\{\mathbf{p}_i\}$ are weighted by $\{w_i\}$, according to their distance from $\mathbf{q}$ and a temperature coefficient t, to obtain a point $\mathbf{r}$ in the soft projected point set R.

Our method builds on and extends the sampling approach proposed by Dovrat *et al*. [6]. For clarity, we briefly review their method in section 3.1. Then, we describe our extension in section 3.2.

3.1. Simplify

Given a point cloud of n 3D coordinates $P \in \mathbb{R}^{n\times3}$, the goal is to find a subset of m points $R^* \in \mathbb{R}^{m\times3}$, such that the sampled point cloud R^* is optimized to a task T. Denoting the objective function of T as $\mathcal{F}$, R^* is given by:

$$R^* = \underset{R}{\operatorname{argmin}}\, \mathcal{F}(T(R)), \quad R \subseteq P, \quad |R| = m \leq n. \quad (2)$$

This optimization problem poses a challenge due to the non-differentiability of the sampling operation. Dovrat *et al*. [6] suggested a simplification network that produces Q from P, where Q is optimal for the task and its points are close to those of P. In order to encourage the second property, a simplification loss is utilized. Denoting average nearest neighbor loss as:

$$\mathcal{L}_a(X,Y) = \frac{1}{|X|}\sum_{\mathbf{x}\in X}\min_{\mathbf{y}\in Y}||\mathbf{x}-\mathbf{y}||_2^2, \quad (3)$$

and maximal nearest neighbor loss as:

$$\mathcal{L}_m(X,Y) = \max_{\mathbf{x}\in X}\min_{\mathbf{y}\in Y}||\mathbf{x}-\mathbf{y}||_2^2, \tag{4}$$

the simplification loss is given by:

$$\begin{aligned}\mathcal{L}_{simplify}(Q,P) = \mathcal{L}_a(Q,P) + \beta\mathcal{L}_m(Q,P) \\ +(\gamma+\delta|Q|)\mathcal{L}_a(P,Q).\end{aligned} \tag{5}$$

In order to optimize the point set Q to the task, the task loss is added to the optimization objective. The total loss of the simplification network is:

$$\mathcal{L}_s(Q,P) = \mathcal{L}_{task}(Q) + \alpha\mathcal{L}_{simplify}(Q,P). \tag{6}$$

The simplification network described above is trained for a specific sample size m. Dovrat *et al.* [6] also proposed a progressive sampling network. This network orders the simplified points according to their importance for the task and can output any sample size. It outputs n points and trained with simplification loss on nested subsets of its output:

$$\mathcal{L}_{prog}(Q,P) = \sum_{c\in C_s}\mathcal{L}_s(Q_c,P), \tag{7}$$

where C_s are control sizes.

3.2. Project

Instead of optimizing the simplified point cloud for the task, we add the *soft projection* operation. The operation is depicted in Figure 4. Each point $\mathbf{q}\in Q$ is softly projected onto its neighborhood, defined by its k nearest neighbors in the complete point cloud P, to obtain a projected point $\mathbf{r}\in R$. The point $\mathbf{r}$ is a weighted average of original points form P:

$$\mathbf{r} = \sum_{i\in\mathcal{N}_P(\mathbf{q})} w_i\mathbf{p}_i, \tag{8}$$

where $\mathcal{N}_P(\mathbf{q})$ contains the indices of the k nearest neighbors of $\mathbf{q}$ in P. The weights $\{w_i\}$ are determined according to the distance between $\mathbf{q}$ and its neighbors, scaled by a learnable temperature coefficient t:

$$w_i = \frac{e^{-d_i^2/t^2}}{\sum_{j\in\mathcal{N}_P(\mathbf{q})} e^{-d_j^2/t^2}}, \tag{9}$$

The distance is given by $d_i = ||\mathbf{q}-\mathbf{p}_i||_2$.

The neighborhood size $k = |\mathcal{N}_P(\mathbf{q})|$ plays a role in the choice of sampled points. Through the distance terms, the network can adapt a simplified point's location such that it will approach a different input point in its local region. While a small neighborhood size demotes exploration, choosing an excessive size may result in loss of local context.

The weights $\{w_i\}$ can be viewed as a probability distribution function over the points $\{\mathbf{p}_i\}$, where $\mathbf{r}$ is the expectation value. The temperature coefficient controls the shape of this distribution. In the limit of $t\to 0$, the distribution converges to a Kronecker delta function, located at the nearest neighbor point.

Given these observations, we would like the point $\mathbf{r}$ to approximate nearest neighbor sampling from the local neighborhood in P. To achieve this we add a projection loss, given by:

$$\mathcal{L}_{project} = t^2. \tag{10}$$

This loss promotes a small temperature value.

In our sampling approach, the task network is fed with the projected point set R rather than simplified set Q. Since each point in R estimates the selection of a point from P, our network is trained to *sample* the input point cloud rather than simplify it.

Our sampling method can be easily extended to the progressive sampling settings (Equation 7). In this case, the loss function takes the form:

$$\begin{aligned}\mathcal{L}^{prog}_{total} = \sum_{c\in C_s}\left(\mathcal{L}_{task}(R_c) + \alpha\mathcal{L}_{simplify}(Q_c,P)\right) \\ +\lambda\mathcal{L}_{project},\end{aligned} \tag{11}$$

where R_c is the point set obtained by applying the soft projection operation on Q_c (Equation 8).

At inference time we replace the soft projection with sampling, to obtain a sampled point cloud R^*. Like in a classification problem, for each point $\mathbf{r}^*\in R^*$, we select the point $\mathbf{p}_i$ with the highest projection weight:

$$\mathbf{r}^* = \mathbf{p}_{i^*}, \quad i^* = \operatorname*{argmax}_{i\in\mathcal{N}_P(\mathbf{q})} w_i. \tag{12}$$

Similar to Dovrat *et al.* [6], if more than one point $\mathbf{r}^*$ corresponds the same point $\mathbf{p}_{i^*}$, we take the unique set of sampled points, complete it using FPS up to m points and evaluate the task performance.

Soft projection as an idempotent operation Strictly speaking, the soft projection operation (Equation 8) is not idempotent [37] and thus does not constitute a mathematical projection. However, when the temperature coefficient in Equation 9 goes to zero, the idempotent sampling operation is obtained (Equation 12). Furthermore, the nearest neighbor selection can be viewed as a variation of projection under the Bregman divergence [4]. The derivation is given in the supplementary.

4. Results

In this section, we present the results of our sampling approach for various applications: point cloud classification, registration, and reconstruction. The performance with

point clouds sampled by our method is contrasted with the commonly used FPS and the learned sampling method, S-NET, proposed by Dovrat *et al*. [6].

Classification and registration are benchmarked on ModelNet40 [44]. We use point clouds of 1024 points that were uniformly sampled from the dataset models. The official train-test split [28] is used for training and evaluation.

The reconstruction task is evaluated with point sets of 2048 points, sampled from ShapeNet Core55 database [3]. We use four shape classes with the largest number of examples: Table, Car, Chair, and Airplane. Each class is split to 85%/5%/10% for train/validation/test sets.

Our network SampleNet is based on PointNet architecture. It operates directly on point clouds and is invariant to permutations of the points. SampleNet applies MLPs to the input points, followed by a global max pooling. Then, a simplified point cloud is computed from the pooled feature vector and projected onto the input point cloud. The complete experimental settings are detailed in the supplemental.

4.1. Classification

Following the experiment of Dovrat *et al*. [6], we use PointNet [28] as the task network for classification. PointNet is trained on point clouds of 1024 points. Then, instance classification accuracy is evaluated on sampled point clouds from the official test split. The sampling ratio is defined as $1024/m$, where m is the number of sampled points.

SampleNet Figure 5 compares the classification performance for several sampling methods. FPS is agnostic to the task, thus leads to substantial accuracy degradation as the sampling ratio increases. S-NET improves over FPS. However, S-NET is trained to simplify the point cloud, while at inference time, sampled points are used. Our SampleNet is trained directly to sample the point cloud, thus, outperforms the competing approaches by a large margin.

For example, at sampling ratio 32 (approximately 3% of the original points), it achieves 80.1% accuracy, which is 20% improvement over S-NET's result and only 9% below the accuracy when using the complete input point set. SampleNet also achieves performance gains with respect to FPS and S-NET in progressive sampling settings (Equation 7). Results are given in the supplementary material.

Simplified, softly projected and sampled points We evaluated the classification accuracy with simplified, softly projected, and sampled points of SampleNet for progressive sampling (denoted as SampleNet-Progressive). Results are reported in Figure 6. For sampling ratios up to 16, the accuracy with simplified points is considerably lower than that of the sampled points. For higher ratios, it is the other way around. On the other hand, the accuracy with softly projected points is very close to that of the sampled ones. This indicates that our network learned to select optimal points

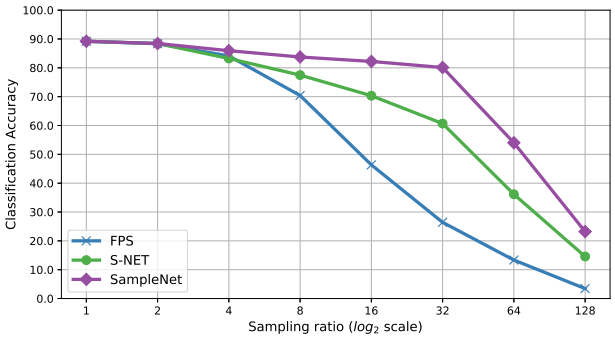

Figure 5. **Classification accuracy with SampleNet.** PointNet is used as the task network and was pre-trained on complete point clouds with 1024 points. The instance classification accuracy is evaluated on sampled point clouds from the test split of ModelNet40. Our sampling method SampleNet outperforms the other sampling alternatives with a large gap.

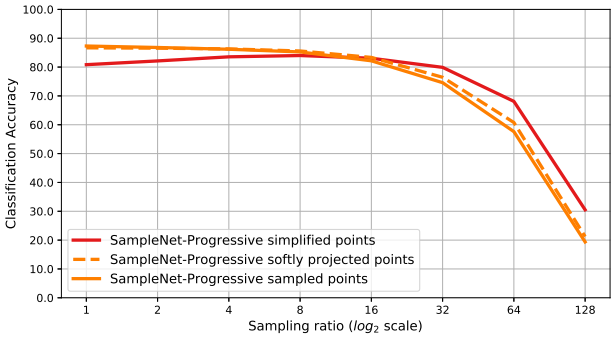

Figure 6. **Classification accuracy with simplified, softly projected, and sampled points.** The instance classification accuracy over the test set of ModelNet40 is measured with simplified, softly projected, and sampled points of SampleNet-Progressive. The accuracy with simplified points is either lower (up to ratio 16) or higher (from ratio 16) than that of the sampled points. On the contrary, the softly projected points closely approximate the accuracy achieved by the sampled points.

for the task from the input point cloud, by approximating sampling with the differentiable soft projection operation.

Weight evolution We examine the evolution of projection weights over time to gain insight into the behavior of the soft projection operation. We train SampleNet for $N_e \in \{1, 10, 100, 150, 200, \ldots, 500\}$ epochs and apply it each time on the test set of ModelNet40. The projection weights are computed for each point and averaged over all the point clouds of the test set.

Figure 7 shows the average projection weights for SampleNet trained to sample 64 points. At the first epoch, the weights are close to a uniform distribution, with a maximal and minimal weight of 0.19 and 0.11, respectively. During training, the first nearest neighbor's weight increases, while the weights of the third to the seventh neighbor de-

crease. The weight of the first and last neighbor converges to 0.43 and 0.03, respectively. Thus, the approximation of the nearest neighbor point by the soft projection operation is improved during training.

Interestingly, the weight distribution does not converge to a delta function at the first nearest neighbor. We recall that the goal of our learned sampling is to seek optimal points for a subsequent task. As depicted in Figure 6, similar performance is achieved with the softly projected and the sampled points. Thus, the approximation of the nearest neighbor, as done by our method, suffices.

To further investigate this subject, we trained SampleNet with additional loss term: a cross-entropy loss between the projection weight vector and a 1-hot vector, representing the nearest neighbor index. We also tried an entropy loss on the projection weights. In these cases, the weights do converge to a delta function. However, we found out that this is an over constraint, which hinders the exploration capability of SampleNet. Details are reported in the supplemental.

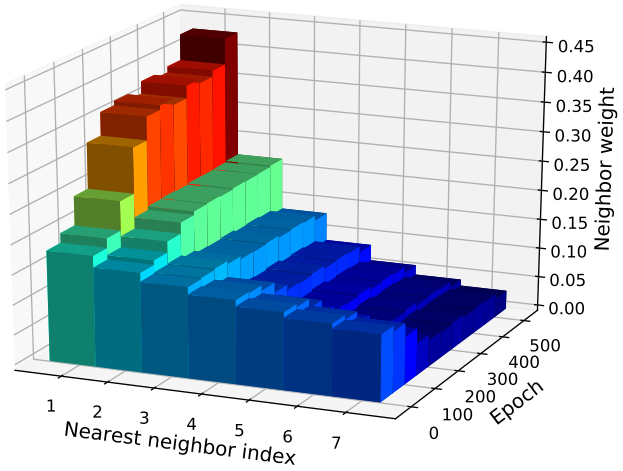

Figure 7. **Evolution of the soft projection weights.** SampleNet is trained to sample 64 points. During training, it is applied to the test split of ModelNet40. The soft projection weights are computed with $k = 7$ neighbors (Equation 9) and averaged over all the examples of the test set. Higher bar with warmer color represents higher weight. As the training progresses, the weight distribution becomes more centered at the close neighbors.

Temperature profile The behavior of the squared temperature coefficient (t^2 in Equation 9) during training is regarded as temperature profile. We study the influence of the temperature profile on the inference classification accuracy. Instead of using a learned profile via the projection loss in Equation 11, we set $\lambda = 0$ and use a pre-determined profile.

Several profiles are examined: linear rectified, exponential, and constant. The first one represents slow convergence; the exponential one simulates convergence to a lower value than that of the learned profile; the constant profile is set to 1, as the initial temperature.

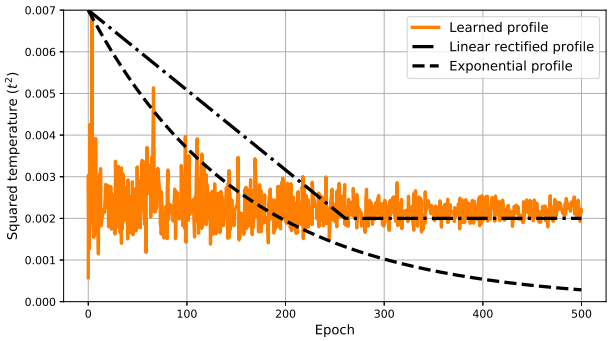

Figure 8. **Temperature profile.** Several temperature profiles are used for the training of SampleNet-Progressive: a learned profile; a linear rectified profile, representing slow convergence; and an exponential profile, converging to a lower value than the learned one. The classification accuracy for SampleNet-Progressive, trained with different profiles, is reported in Table 1.

SR	2	4	8	16	32	64	128
FPS	85.6	81.2	68.1	49.4	29.7	16.3	8.6
Con	85.5	75.8	49.6	32.7	17.1	7.0	4.7
Lin	86.7	86.0	85.0	**83.1**	73.7	50.9	20.5
Exp	86.6	85.9	**85.6**	82.0	74.2	55.6	**21.4**
Lrn	**86.8**	**86.2**	85.3	82.2	**74.6**	**57.6**	19.4

Table 1. **Classification accuracy with different temperature profiles.** SR stands for sampling ratio. Third to last rows correspond to SampleNet-Progressive trained with constant (Con), linear rectified (Lin), exponential (Exp), and learned (Lrn) temperature profile, respectively. SampleNet-Progressive is robust to the decay behavior of the profile. However, if the temperature remains constant, the classification accuracy degrades substantially.

The first two profiles and the learned profile are presented in Figure 8. Table 1 shows the classification accuracy with sampled points of SampleNet-Progressive, which was trained with different profiles. Both linear rectified and exponential profiles result in similar performance of the learned profile, with a slight advantage to the latter. However, a constant temperature causes substantial performance degradation, which is even worse than that of FPS. It indicates that a decaying profile is required for the success of SampleNet. Yet, is it robust to the decay behavior.

Time, space, and performance SampleNet offers a trade-off between time, space, and performance. For example, employing SampleNet for sampling 32 points before PointNet saves about 90% of the inference time, with respect to applying PointNet on the original point clouds. It requires only an additional 6% memory space and results in less than 10% drop in the classification accuracy. The computation is detailed in the supplementary.

4.2. Registration

We follow the work of Sarode *et al*. [32] and their proposed PCRNet to construct a point cloud registration network. Point sets with 1024 points of the Car category in ModelNet40 are used. For training, we generate 4925 pairs of source and template point clouds from examples of the train set. The template is rotated by three random Euler angles in the range of $[-45°, 45°]$ to obtain the source. An additional 100 source-template pairs are generated from the test split for performance evaluation. Experiments with other shape categories appear in the supplemental.

PCRNet is trained on complete point clouds with two supervision signals: the ground truth rotation and the Chamfer distance [1] between the registered source and template point clouds. To train SampleNet, we freeze PCRNet and apply the same sampler to both the source and template. The registration performance is measured in mean rotation error (MRE) between the estimated and the ground truth rotation in angle-axis representation. More details regarding the loss terms and the evaluation metric are given in the supplementary material.

The sampling method of Dovrat *et al*. [6] was not applied for the registration task, and much work is needed for its adaption. Thus, for this application, we utilize FPS and random sampling as baselines. Figure 9 presents the MRE for different sampling methods. The MRE with our proposed sampling remains low, while for the other methods, it is increased with the sampling ratio. For example, for a ratio of 32, the MRE with SampleNet is $5.94°$, while FPS results in a MRE of $13.46°$, more than twice than SampleNet.

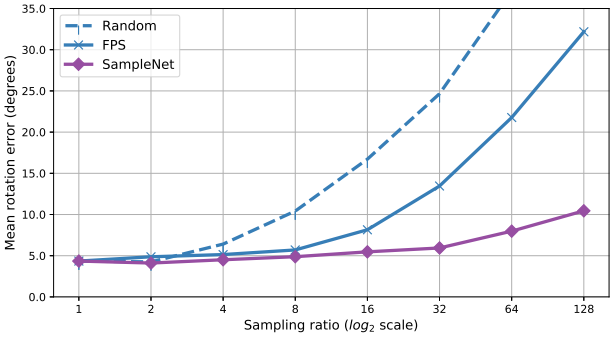

Figure 9. **Rotation error with SampleNet.** PCRNet is used as the task network for registration. It was trained on complete point clouds of 1024 points from the Car category in ModelNet40. Mean rotation error (MRE) between registered source and template point cloud pairs is measured on the test split for different sampling methods. Our SampleNet achieves the lowest MRE for all ratios.

A registration example is visualized in Figure 10. FPS points are taken uniformly, while SampleNet points are located at semantic features of the shape. Using FPS does not enable to align the sampled points, as they are sampled at different parts of the original point cloud. In contrast, SampleNet learns to sample similar points from different source and template clouds. Thus, registration with its sampled sets is possible. Quantitative measure of this sampling consistency is presented in the supplementary.

In conclusion, SampleNet proves to be an efficient sampling method for the registration task, overcoming the challenge of sampling two different point clouds. We attribute this success to the permutation invariance of SampleNet, as opposed to FPS and random sampling. That, together with the task-specific optimization, gives SampleNet the ability to achieve low registration error.

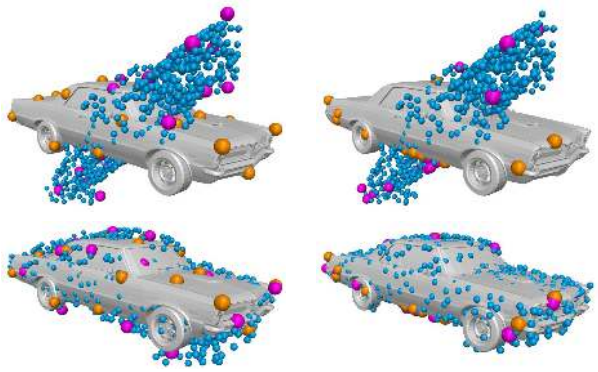

Figure 10. **Registration with sampled points.** Top row: unregistered source with 1024 points in Blue overlaid on the mesh model. Sampled sets of 32 points from the template and source are illustrated in Orange and Magenta, respectively. Bottom row: the registered source cloud is overlaid on the mesh. SampleNet enables us to perform registration of point clouds from their samples.

4.3. Reconstruction

SampleNet is applied to the reconstruction of points clouds from sampled points. The task network, in this case, is the autoencoder of Achlioptas *et al*. [1] that was trained on point clouds with 2048 points. The sampling ratio is defined as $2048/m$, where m is the sample size.

We evaluate the reconstruction performance by normalized reconstruction error (NRE) [6]. The reconstruction error is the Chamfer distance [1] between a reconstructed point cloud and the complete input set. The NRE is the error when reconstructing from a sampled set divided by the error of reconstruction from the complete input.

Figure 11 reports the average NRE for the test split of the shape classes we use from ShapeNet database. Up to sampling ratio of 8, all the methods result in similar reconstruction performance. However, for higher ratios, SampleNet outperforms the other alternatives, with an increasing margin. For example, for a sampling ratio of 32, the NRE for S-NET is 1.57 versus 1.33 for SampleNet - a reduction of

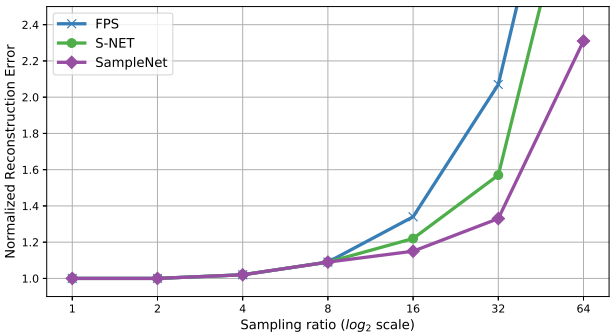

Figure 11. **SampleNet for reconstruction.** The input point cloud is reconstructed from its sampled points. The reconstruction error is normalized by the error when using the complete input point set. Starting from ratio 8, SampleNet achieves lower error, with an increasing gap in the sampling ratio.

24%. We conclude that SampleNet learns to sample useful points for reconstructing point sets unseen during training.

Reconstruction from samples is visualized in Figure 12. FPS points are spread over the shape uniformly, as opposed to the non-uniform pattern of SampleNet and S-NET. Interestingly, some points of the learned sampling methods are sampled in similar locations, for example, at the legs of the chair. Nevertheless, reconstructing using S-NET or FPS points results in artifacts or loss of details. On the contrary, utilizing SampleNet reconstructs the input shape better.

A failure case When computing the NRE per shape class, SampleNet achieves lower NRE for Chair, Car, and Table classes. However, the NRE of FPS is better than that of SampleNet for airplanes. For example, for a sample size of 64 points, the NRE of FPS is 1.31, while the NREs of SampleNet and S-NET are 1.39 and 1.41, respectively. Figure 13 shows an example of reconstructing an airplane from 64 points. FPS samples more points on the wings than SampleNet. These points are important for the reconstruction of the input, thus leading to an improved result.

5. Conclusions

We presented a learned sampling approach for point clouds. Our network, SampleNet, takes an input point cloud and produces a smaller point cloud that is optimized to some downstream task. The key challenge was to deal with the non-differentiability of the sampling operation. To solve this problem, we proposed a differentiable relaxation, termed soft projection, that represents output points as a weighted average of points in the input. During training, the projection weights were optimized to approximate nearest neighbor sampling, which occurs at the inference phase. The soft projection operation replaced the regression of optimal points in the ambient space with multiple classification problems in local neighborhoods of the input.

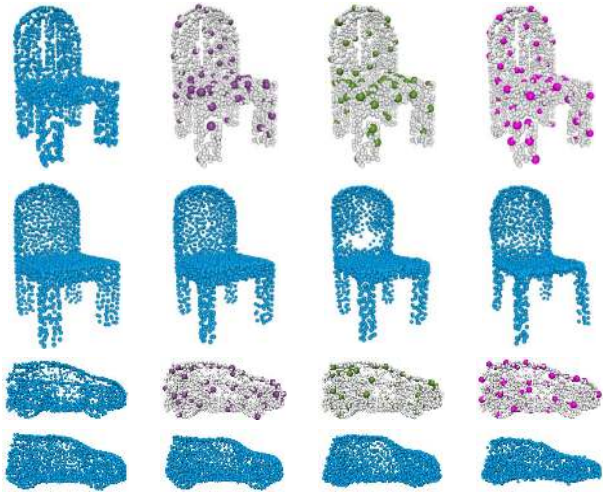

Figure 12. **Reconstruction from sampled points.** Top and third rows: complete input point set of 2048 points, input with 64 SampleNet points (in Purple), input with 64 S-NET points (in Green), input with 64 FPS points (in Magenta). Second and bottom rows: reconstructed point cloud from the input and the corresponding sample. Using SampleNet better preserves the input shape and results in similar reconstruction to one from the complete input.

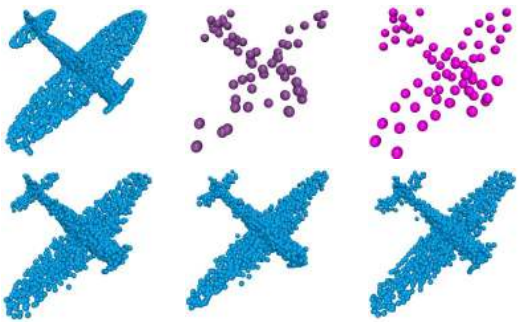

Figure 13. **A failure example.** Top row: complete input with 2048 points, 64 SampleNet points (in Purple), 64 FPS points (in Magenta). Bottom row: reconstruction from complete input and from corresponding sampled points. In this case, uniform sampling by FPS is preferred.

We applied our technique to point cloud classification and reconstruction. We also evaluated our method on the task of point cloud registration. The latter is more challenging than previous tasks because it requires the sampling to be consistent across two different point clouds. We found that our method consistently outperforms the competing non-learned as well as learned sampling alternatives by a large margin.

Acknowledgment This work was partly funded by ISF grant number 1549/19.

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, 7

[2] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. PointNetLK: Robust & Efficient Point Cloud Registration using PointNet. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7163–7172, 2019. 2

[3] Angel X. Chang, Thomas Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*, 2015. 5

[4] Pengwen Chen, Yunmei Chen, and Murali Rao. Metrics Defined by Bregman Divergences. *Communications in Mathematical Sciences*, 6, 2008. 4

[5] Xuelin Chen, Baoquan Chen, and Niloy J. Mitra. Unpaired Point Cloud Completion on Real Scans using Adversarial Training. *arXiv preprint arXiv:1904.00069*, 2019. 2

[6] Oren Dovrat, Itai Lang, and Shai Avidan. Learning to Sample. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2760–2769, 2019. 1, 2, 3, 4, 5, 7

[7] 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. 1

[8] Natasha Gelfand, Leslie Ikemoto, Szymon Rusinkiewicz, and Marc Levoy. Geometrically Stable Sampling for the ICP Algorithm. *Proceedings of the IEEE International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pages 260–267, 2003. 1

[9] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood Components Analysis. *Proceedings of Advances in Neural Information Processing Systems (NeuralIPS)*, pages 513–520, 2005. 2

[10] 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. 2

[11] Michael Himmelsbach, Thorsten Luettel, and H-J Wuensche. Real-time Object Classification in 3D Point Clouds using Point Feature Histograms. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 994–1000, 2009. 1

[12] Roman Klokov and Victor Lempitsky. Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models. *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017. 1

[13] Loic Landrieu and Martin Simonovsky. Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4558–4567, 2018. 2

[14] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point Cloud GAN. *arXiv preprint arXiv:1810.05795*, 2018. 2

[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. 2

[16] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-GAN: a Point Cloud Upsampling Adversarial Network. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[17] Xianzhi Li, Lequan Yu, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Unsupervised Detection of Distinctive Regions on 3D Shapes. *arXiv preprint arXiv:1905.01684*, 2019. 3

[18] 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. 1, 2

[19] Yi Li, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J. Guibas. GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in Point Cloud. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3947–3956, 2019. 2

[20] Daniel Liu, Roland Yu, and Hao Su. Extending Adversarial Attacks and Defenses to Deep 3D PointCloud Classifiers. *arXiv preprint arXiv:1901.03006*, 2019. 2

[21] 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. 2

[22] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. DeepICP: An End-to-End Deep Neural Network for 3D Point Cloud Registration. *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019. 2

[23] Carsten Moenning and Neil A. Dodgson. Fast Marching farthest point sampling. *Eurographics Poster Presentation*, 2003. 1

[24] Ehsan Nezhadarya, Taghavi Ehsan, Bingbing Liu, and Jun Luo. Adaptive Hierarchical Down-Sampling for Point Cloud Classification. *arXiv preprint arXiv:1904.08506*, 2019. 3

[25] Quang-Hieu Pham, Duc Thanh Nguyen, Binh-Son Hua, Gemma Roig, and Sai-Kit Yeung. JSIS3D: Joint Semantic-Instance Segmentation of 3D Point Clouds with Multi-Task Pointwise Networks and Multi-Value Conditional Random Fields. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8827–8836, 2019. 2

[26] Tobias Plötz and Stefan Roth. Neural Nearest Neighbors Networks. *Proceedings of Advances in Neural Information Processing Systems (NeuralIPS)*, 2018. 2

[27] 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. 1, 2, 3

[28] 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. 2, 5

[29] 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. 2

[30] 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. 1, 2, 3

[31] Muhammad Sarmad, Hyunjoo Jenny Lee, and Young Min Kim. RL-GAN-Net: A Reinforcement Learning Agent Controlled GAN Network forReal-Time Point Cloud Shape Completion. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5898–5907, 2019. 2

[32] Vinit Sarode, Xueqian Li, Hunter Goforth, Rangaprasad Arun Aoki, Yasuhiro Srivatsan, Simon Lucey, and Howie Choset. PCRNet: Point Cloud Registration Network using PointNet Encoding. *arXiv preprint arXiv:1908.07906*, 2019. 2, 7

[33] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–779, 2019. 2

[34] 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. 2

[35] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view Convolutional Neural Networks for 3D Shape Recognition. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 945–953, 2017. 2

[36] 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. 2

[37] Robert J Valenza. *Linear Algebra: An Introduction to Abstract Mathematics*. Springer Science & Business Media, 2012. 4

[38] P. J. M. Van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, 1987. 2

[39] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph Attention Convolution for Point Cloud Semantic Segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10296–10305, 2019. 2

[40] 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. 2

[41] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. Associatively Segmenting Instances and Semantics in Point Clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4096–4105, 2019. 2

[42] 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. 2

[43] 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. 2

[44] 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. 2, 5

[45] Chong Xiang, Charles R. Qi, and Bo Li. Generating 3D Adversarial Point Clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9136–9144, 2019. 2

[46] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. PointFlow: 3D Point Cloud Generation with Continuous Normalizing Flows. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[47] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling Point Clouds with Self-Attention and Gumbel Subset Sampling. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3323–3332, 2019. 3

[48] 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. 2

[49] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. Patch-based Progressive 3D Point Set Upsampling. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5958–5967, 2019. 2

[50] Kangxue Yin, Zhiqin Chen, Hui Huang, Daniel Cohen-Or, and Zhang Hao. LOGAN: Unpaired Shape Transform in Latent Overcomplete Space. *arXiv preprint arXiv:1904.10170*, 2019. 3

[51] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen Or, and Pheng Ann Heng. EC-Net: an Edge-aware Point set Consolidation Network. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2

[52] 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. 1, 2

[53] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. PCN: Point Completion Network. *Proceedings of the International Conference on 3D Vision (3DV)*, 2018. 2

[54] 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. 2