2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

Representation Learning via Parallel Subset Reconstruction for 3D Point Cloud Generation

Kohei Matsuzaki† and Kazuyuki Tasaka†

***Abstract*— Three-dimensional (3D) point cloud processing has attracted a great deal of attention in computer vision, robotics, and the machine learning community because of significant progress in deep neural networks on 3D data. Another trend in the community is learning of generative models based on generative adversarial networks. In this paper, we propose a framework for 3D point cloud generation based on a combination of auto-encoders and generative adversarial networks. The framework first trains auto-encoders to learn latent representations, and then trains generative adversarial networks in the learned latent space. We focus on improving the training method for auto-encoders in order to generate 3D point clouds with higher fidelity and coverage. We add parallel sub-decoders that reconstruct subsets of the input point cloud. In order to construct these subsets, we introduce a point sampling algorithm that imposes a method to sample spatially localized point sets. These local subsets are utilized to measure local reconstruction losses. We train auto-encoders to learn an effective latent representation for both global and local shape reconstruction based on the multi-task learning approach. Furthermore, we add global and local adversarial losses to generate more plausible point clouds. Quantitative and qualitative evaluations demonstrate that the proposed method outperforms state-of-the-art method on the task of 3D point cloud generation.**

I. INTRODUCTION

With the advancement of 3D sensing technologies, 3D point cloud processing has attracted a great deal of attention in computer vision, robotics, and the machine learning community. A 3D point cloud is typically represented as a set of 3D points, namely (x, y, z) coordinates. Since the point cloud is a spatially irregular data format, it is difficult to apply it to the traditional deep learning framework, in other words, convolutional neural networks (CNNs). Therefore, many studies have been transformed point clouds into regular data formats, such as voxel grids [1], [2], [3], [4] and multi-view images [5], [6], [7], [8], for input into the CNNs framework.

Qi et al. [9] proposed a deep learning framework that takes the 3D point cloud directly into input without transforming the data format, which was referred to as PointNet. This work showed better results than state-of-the-art voxel-based methods and multi-view image-based methods on 3D object classification and segmentation benchmarks. Subsequent studies that used PointNet as a feature extractor have also been successful in various applications, such as 3D object detection [10], local feature description [11], and pose estimation [12].

†Authors are with Media Recognition Laboratory, Media ICT Department, KDDI Research, Inc., Saitama, Japan {ko-matsuzaki, ka-tasaka}@kddi-research.jp

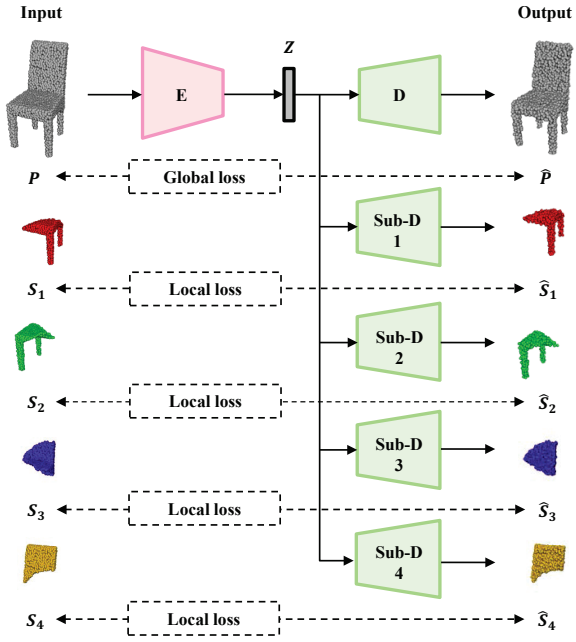
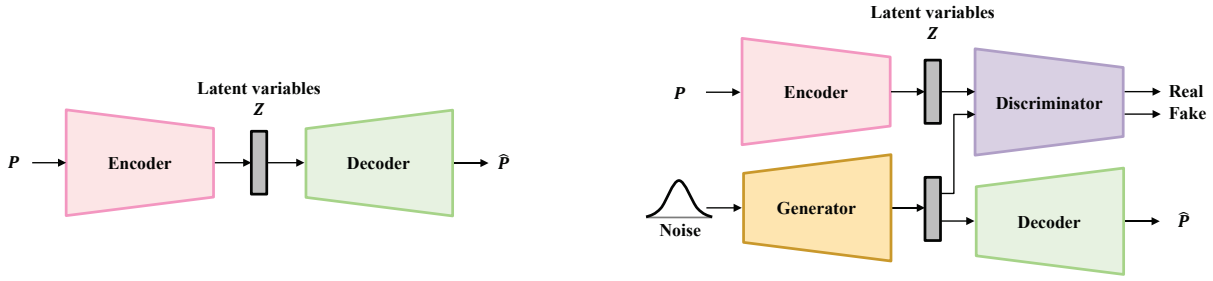


Fig. 1: Illustration of the proposed auto-encoders with parallel sub-decoders. The encoder (E) compresses the input point cloud P to the latent variables Z, and the decoder (D) reconstructs the output point cloud $\widehat{P}$ from the Z. At the same time, M sub-decoders reconstruct point clouds $\widehat{S}_1, ..., \widehat{S}_M$ from the Z. In this figure, we set $M = 4$. The global loss is measured between P and $\widehat{P}$. The local losses are measured between P's local subsets $S_1, ..., S_M$ and the reconstructed subsets $\widehat{S}_1, ..., \widehat{S}_M$.

Another trend in the above community is learning of generative models based on generative adversarial networks (GANs) [13]. Although GANs were originally designed for image generation task, they have been applied to various tasks, such as data augmentation [14], domain translation [15], and anomaly detection [16]. Therefore, GANs have become a fundamental technique for many machine learning tasks. However, there are not many studies on GANs for the 3D point cloud.

Achlioptas et al. [17] proposed the first deep generative models for the 3D point cloud. In the latent-space GANs that they introduced, they first train auto-encoders to learn latent representations, and then GANs are trained in the latent space of the auto-encoders. Its architecture is inspired by PointNet which takes the 3D point cloud directly as input, and outputs the reconstructed the 3D point cloud. However, PointNet cannot capture the local structures of the point cloud. Therefore, there is a possibility that it cannot generate

978-1-7281-4003-2/19/$31.00 ©2019 IEEE



(a) Auto-encoders

(b) Generative adversarial networks

Fig. 2: The framework of latent-space GANs. The training pipeline consists of two stages. The first stage is a training of (a) Auto-encoders: the encoder learns to compress the input point cloud P into the latent variables Z, and the decoder learns to reconstruct the output point cloud $\widehat{P}$ from the latent variables. The second stage is a training of (b) Generative adversarial networks: the generator learns to convert Gaussian noise vectors to latent variables, and the discriminator learns to distinguish between real samples and fake samples on the latent space of the auto-encoders.

point clouds with faithful shapes up to the finer parts such as the legs of a chair.

In this paper, we propose a method to improve fidelity and coverage in deep generative models for the 3D point cloud. As shown in Fig. 1, we reconstruct the local subsets of the input point cloud from the latent variables simultaneously when training the auto-encoders. Then, we optimize the auto-encoders using the global and local reconstruction losses obtained from input point cloud and its subsets. In other words, we train auto-encoders to learn an effective latent representation for both global and local shape reconstruction based on a multi-task learning approach. Although our network is not intended to perform multiple tasks, it introduces multiple auxiliary losses for training the auto-encoders. Furthermore, we add global and local adversarial losses for the training to generate more plausible point clouds.

Our contribution can be summarized as follows:

- We propose a novel training method for 3D point cloud auto-encoders that uses multiple auxiliary losses measured from the local subsets of input point cloud.
- We introduce a point sampling algorithm that imposes a method to sample spatially localized point sets.
- In the experiments, we demonstrate that the proposed method improves the fidelity and coverage in 3D point cloud generation.

II. Related work

In this section, we briefly discuss related work to deep generative models for 3D point cloud and our approach.

A. Deep Neural Networks on 3D Data

With the significant progress of deep neural networks on 2D data, the deep neural networks on 3D data have been researched. These studies are roughly classified into voxel-based approaches [1], [2], [3], [4], multi-view image-based approaches [5], [6], [7], [8], and point-cloud-based approaches [9], [18], [19], [20], [21]. Since voxel-based and multi-view image-based approaches transform input data into 3D grids of values and collections of images, respectively, these approaches introduce quantization artifacts. On the other hand, the point cloud is more suitable for 3D data processing because it is a 3D representation close to raw data measured by 3D sensors (e.g., LiDARs, time-of-flight cameras). Therefore, we focus on a point-cloud-based approach in this paper. The most popular method in a point-cloud-based approach is PointNet [9]. It can take unordered point cloud directly as input and extracts global shape features. However, it has the limitation that it cannot capture the local structure. PointNet++ [18] addresses this problem by grouping subsets of input point cloud and applying PointNet to each subset, but this grouping scheme does not necessarily cover all of the input point cloud.

B. Generative Adversarial Networks

GANs [13] are state-of-the-art generative models that can learn to map random noise vectors to the data distribution of a training set. Basic GAN models consist of a generator (G) and a discriminator (D). The D learns to distinguish between real samples and fake samples. On the other hand, the G learns to generate fake samples from random noise vectors, which are indistinguishable from real samples. In training GANs, the G and D are alternately optimized based on the following two-player minmax game:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z}))], \tag{1}$$

where $\mathbf{x}$ represents a real sample drawn from data distribution $p(\mathbf{x})$ and $\mathbf{z}$ represents a noise vector drawn from prior distribution $p(\mathbf{z})$.

C. 3D Point Cloud Generation

Although many methods have been proposed to apply the GAN framework for image generation [22], [23], [24] and video generation [25], [26], [27], there are not many studies on 3D shape generation. Achlioptas et al. [17] proposed the first deep generative models for the 3D point cloud, which was referred to as latent-space GANs. As shown in Fig. 2, these models have two components—auto-encoders and GANs. They are trained in two stages: auto-encoders are first trained, and then the GANs are trained using a pre-trained encoder, a trainable generator, and a trainable discriminator. In the auto-encoders training stage, the encoder learns to

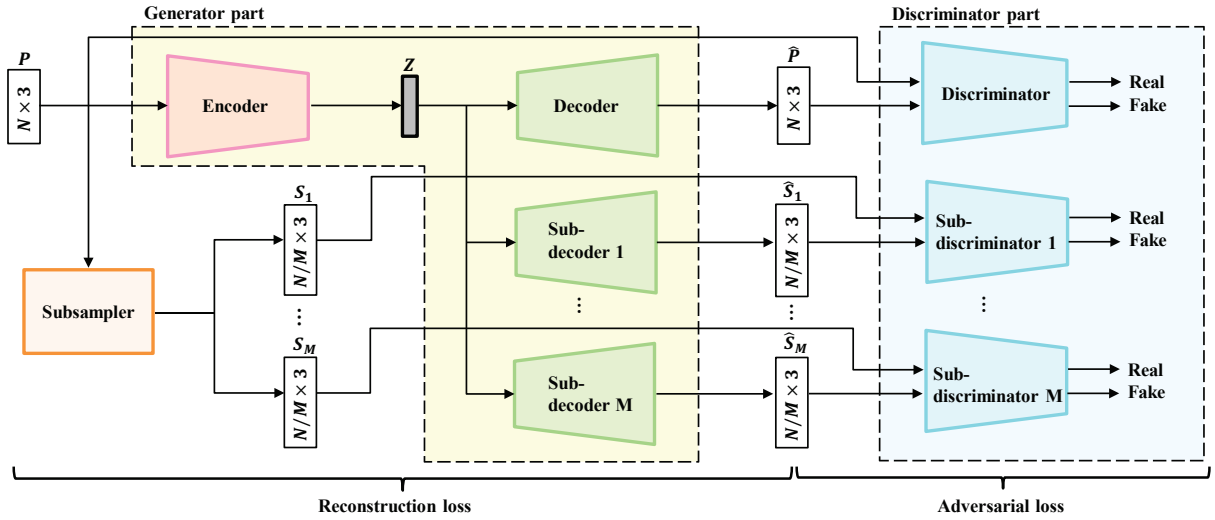


Fig. 3: Network architecture of the proposed auto-encoders. The network consists of a generator part and a discriminator part for measuring reconstruction loss and adversarial loss respectively. The generator part includes an encoder, a decoder, and M sub-decoders. The discriminator part includes a discriminator and M sub-discriminators corresponding to the decoder and sub-decoders. The input to this network is a point cloud $P \in \mathbb{R}^{N\times 3}$. The subsampler constructs M subset point clouds $S_1, ..., S_M \in \mathbb{R}^{N/M\times 3}$ from P. The decoder and sub-decoders reconstruct point clouds $\widehat{P} \in \mathbb{R}^{N\times 3}$ and $\widehat{S}_1, ..., \widehat{S}_M \in \mathbb{R}^{N/M\times 3}$ respectively, with common latent variables as input. The discriminator and the sub-discriminator judge whether the input point cloud is a real sample or a fake sample.

compress input point cloud P into latent variables Z, and the decoder learns to reconstruct output point cloud $\widehat{P}$ from Z. The reconstruction loss metric is the Chamfer distance (CD) or Earth Mover's distance (EMD) between the input and the output. In the GAN training stage, Gaussian noise vectors are input to the generator, and latent variables are generated as fake samples. On the other hand, point clouds from the training set are input to the pre-trained encoder, and latent variables are obtained as real samples. The discriminator is trained to distinguish between real samples and fake samples on the latent space of the auto-encoders. After this training, 3D point cloud generation is achieved by decoding latent variables generated from Gaussian noise vectors using a pre-trained decoder.

D. Multi-task Learning

Multi-task learning [28] is an approach to improve learning for one task by learning multiple-tasks in parallel while using a shared representation. This approach has been widely used for machine learning applications such as scene recognition and object detection [29]; prediction of surface normal, depth, and instance contour [30]; robot manipulation [31]; and domain adaptation [32]. In this paper, we propose to apply a multi-task learning approach to 3D point cloud reconstruction with auto-encoders. The proposed method reconstructs the input point cloud and its subsets in parallel from a shared representation. We regard the reconstruction tasks of the input point cloud and its subsets as related but different tasks. However, our purpose is only to achieve the task of input point cloud reconstruction. The subset reconstruction is performed just to improve learning for input point cloud reconstruction.

III. Proposed Method

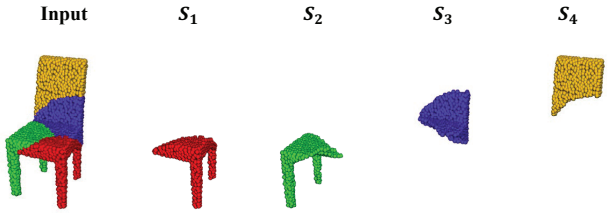
This section describes our approach to improve fidelity and coverage in deep generative models for the 3D point cloud. As a baseline, we employ the latent-space GANs [17], which are state-of-the-art deep generative models. As explained in section II-C, it first learns auto-encoders and learns GANs on the latent space of auto-encoders. Since EMD is known to capture the shape better than CD [33], we select EMD as the reconstruction loss metric of auto-encoders.

The network architecture is shown in Fig. 3. The encoder and decoder are components of standard auto-encoders, whose input is a point cloud $P \in \mathbb{R}^{N\times 3}$ and output is a reconstructed point cloud $\widehat{P} \in \mathbb{R}^{N\times 3}$. In addition, our network has several additional components: subsampler, sub-decoders, discriminator, and sub-discriminators. We first construct M local subsets from P by the subsampler (Sec. III-A). After that, the encoder compresses P into latent variables Z, and the multiple sub-decoders reconstruct the point clouds from the shared latent variables Z. Here, the decoder reconstructs the point cloud with N points while the M sub-decoders additionally reconstruct the point cloud with N/M points. These reconstructed point clouds are used to measure reconstruction loss. Furthermore, we measure adversarial losses with point-cloud-based discriminators inspired by [34]. We add a discriminator and M sub-discriminators that consider the input point clouds as real samples and the reconstructed point clouds as fake samples (Sec. III-B). Finally, we calculate the joint loss combining the reconstruction loss and the adversarial loss. In the training of a network, the generator part and discriminator part in Fig. 3 are alternately optimized based on the two-player minmax game (Sec. III-C). After training of the network, we

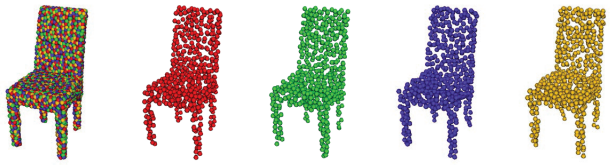
Algorithm 1 Region growing sampling

Input: point cloud $P = \{x_1, ..., x_N \in \mathbb{R}^3\}$, number of subsets M, threshold for nearest neighbor search k
Output: subsets of point cloud $S_i = \{x_1, ..., x_{N/M} \in \mathbb{R}^3\}$, where $i = 1, ..., M$

for $i = 1 \rightarrow M$ **do**
  $S_i \leftarrow \emptyset$
  $\mathcal{N}(x) \leftarrow k$-nearest neighbor points of $x \in P$
  $g \leftarrow$ gravity point of P
  $s_i \leftarrow \arg\max_{x \in P} \|x - g\|_2$
  $S_i \leftarrow S_i \cup \{s_i\}$, $P \leftarrow P \setminus \{s_i\}$
  for $j = 1 \rightarrow N/M - 1$ **do**
    $C \leftarrow \bigcup_{x \in S_i} \mathcal{N}(x) \cap P$
    if $C = \emptyset$ **then**
      $y \leftarrow \arg\min_{x \in P} \|x - s_i\|_2$
      $C \leftarrow C \cup \{y\}$
    end if
    $y \leftarrow \arg\min_{x \in C} \|x - s_i\|_2$
    $S_i \leftarrow S_i \cup \{y\}$, $P \leftarrow P \setminus \{y\}$
  end for
end for



(a) Region growing sampling



(b) Farthest point sampling

Fig. 4: Visualization of point cloud subsampling. The (a) and (b) show the results based on region growing sampling and farthest point sampling, respectively. The left image shows the input point cloud, otherwise its subsets. The color of the input point cloud corresponds to color of the resulting subsets. Region growing sampling constructs spatially localized subsets, while farthest point sampling constructs spatially uniform subsets.

only need the encoder and decoder. Therefore, the proposed method does not require more capacity to the model than standard auto-encoders in downstream tasks.

A. Point Cloud Subsampling

Our subsampler constructs multiple local subsets by subsampling spatially localized points from the input points. For this purpose, the sphere cover algorithm is often used to subsample all points within the radius from a query point [18], [35]. However, this algorithm does not guarantee coverage of all points of the input point cloud even if a sufficiently large number of spheres are used [35]. In order to guarantee a subsample of all the points, we introduce a novel sampling algorithm referred to as region growing sampling. It samples neighboring points from a seed point based on the region growing method.

Details of the algorithm are shown in Algorithm 1. This algorithm inputs a point cloud P with N points, the number of required subsets M, and a threshold for the nearest neighbor search k. For i-th subset S_i, we consider the farthest point from the center of gravity of P as seed point s_i and store it in S_i. We delete the point stored in the subset from P. In order to impose a method to sample the spatially localized points, we construct candidate set C that is the k-nearest neighbors of each point in S_i. Here, the points already stored in other subsets are not included in C. If C is empty, we put the point closest to seed point s_i in P into the C as an exception process. Then we select the point closest to seed point s_i in C and store it in S_i. For each subset, we repeat this sampling until N/M number of points are stored. As a result, we obtain M subsets consisting of N/M points.

Fig. 4(a) shows an example in the case of $M = 4$. For comparison, an example of subsampling results based on the farthest point sampling (FPS) algorithm [36] is also shown in Fig. 4(b). In contrast to FPS, which is spatially uniform subsampling, we can see that our algorithm achieves spatially localized subsampling.

B. Network Architecture

The input of the encoder is a point cloud with N points, and the output is a b-dimensional vector referred to as latent variables Z. The encoder consists of 1D-convolutional layers with kernel size 1, which follows the design of PointNet. ReLU activation [37] and batch normalization [38] are applied to each layer.

The input of the decoder is latent variables Z, and the output is a reconstructed point cloud with N points. The decoder consists of fully-connected layers that apply ReLU activation, except the last layer. The sub-decoders have the common network architecture with the decoder except for the output dimension. Although the input to them is the latent variables Z, the output is a reconstructed point cloud with N/M points. That is, these sub-decoders reconstruct the local subsets described in section III-A in parallel.

The input of the discriminator is a point cloud with N points and the output is a scalar value in the range [0, 1]. This point cloud is either an input point cloud or a reconstructed point cloud corresponding to a real sample or a fake sample, respectively. The discriminator consists of 1D-convolutional layers applying leaky ReLU activation [39] that extracts features from the point cloud and fully-connected layers that reduce the feature dimension to 1. This feature is fed into the sigmoid neuron, and the discriminator

outputs a confidence probability on the fact that the point cloud is a real sample. The sub-discriminators have the common network architecture with the discriminator except for the input dimension. The input of the sub-discriminators is either a local subset or its reconstructed point cloud.

C. Joint Loss Function

Inspired by [34], [40], we design a joint loss function combining the reconstruction loss and the adversarial loss. Furthermore, our loss function includes multiple auxiliary loss terms that are measured using multiple subsets of the input point cloud.

Reconstruction Loss. As a metric of reconstruction loss between the input point cloud and the reconstructed point cloud, we use the EMD proposed in [33]. If two point clouds with equal size $X_1 \subseteq \mathbb{R}^3$ and $X_2 \subseteq \mathbb{R}^3$ are given, their EMD is defined by

$$EMD(X_1, X_2) = \min_{\phi: X_1 \to X_2} \sum_{x \in X_1} \|x - \phi(x)\|_2, \quad (2)$$

where $\phi : X_1 \to X_2$ is a bijection.

We independently measure the EMD for point clouds of size N and point clouds of size N/M. To distinguish between these, we conveniently call the former a global reconstruction loss and the latter a local reconstruction loss. The global reconstruction loss is defined by

$$\mathcal{L}^g_{rec} = EMD(P, \widehat{P}). \quad (3)$$

We take the sum of local reconstruction losses for all subsets. The total local reconstruction loss is defined by

$$\mathcal{L}^l_{rec} = \sum_{i=1}^{M} EMD(S_i, \widehat{S}_i). \quad (4)$$

Adversarial Loss. To generate a more plausible point cloud, we add adversarial loss. Since it is known that standard GANs often cause a mode collapse, we adopt Wasserstein GANs with a gradient penalty (WGANs-gp) [41], [42].

Here, we denote encoder/decoder as generator G and discriminator as D. A point cloud $X \subseteq \mathbb{R}^3$ is given, the G first encodes the point cloud X into latent variables and then decodes a point cloud $\widehat{X} \subseteq \mathbb{R}^3$ from the latent variables, namely $\widehat{X} = G(X)$. The generator loss is defined by

$$\mathcal{L}_{gen}(X) = -\mathbb{E}_{\widehat{X} \sim p(\widehat{X})}[D(\widehat{X})], \quad (5)$$

where $p(\widehat{X})$ is a generator distribution. The discriminator loss is defined by

$$\begin{aligned} \mathcal{L}_{dis}(X) = \mathbb{E}_{X \sim p(X)}[D(X)] - \mathbb{E}_{\widehat{X} \sim p(\widehat{X})}[D(\widehat{X})] \\ + \lambda_{gp} \mathbb{E}_{X' \sim p(X')}[(\|\nabla_{X'} D(X')\|_2 - 1)^2], \end{aligned} \quad (6)$$

where $p(X')$ is a distribution obtained by sampling uniformly along a straight line between a pair of points sampled from data distribution $p(X)$ and generator distribution $p(\widehat{X})$. The λ_{gp} is a hyper-parameter for gradient penalty term.

As with the reconstruction loss, we distinguish between global and local adversarial loss. The global adversarial loss is defined by

$$\mathcal{L}^g_{gen} = \mathcal{L}_{gen}(P), \ \mathcal{L}^g_{dis} = \mathcal{L}_{dis}(P). \quad (7)$$

The total local reconstruction loss is defined by

$$\mathcal{L}^l_{gen} = \sum_{i=1}^{M} \mathcal{L}_{gen}(S_i), \ \mathcal{L}^l_{dis} = \sum_{i=1}^{M} \mathcal{L}_{dis}(S_i). \quad (8)$$

Joint Loss. Finally, we optimize G and D alternately using the following joint loss functions $\mathcal{L}_G$ and $\mathcal{L}_D$, respectively:

$$\mathcal{L}_G = \mathcal{L}^g_{gen} + \lambda^l_{adv} \mathcal{L}^l_{gen} + \lambda^g_{rec} \mathcal{L}^g_{rec} + \lambda^l_{rec} \mathcal{L}^l_{rec}, \quad (9)$$

$$\mathcal{L}_D = \mathcal{L}^g_{dis} + \lambda^l_{adv} \mathcal{L}^l_{dis}, \quad (10)$$

where λ^l_{adv}, λ^g_{rec}, and λ^l_{rec} are hyper-parameters that control the balance between reconstruction loss and adversarial loss.

IV. Experiments

In this section, we describe experimental evaluations that verify the effectiveness of our approach. After explaining the experimental setup, we report the quantitative and qualitative evaluation results.

A. Experimental Setup

Dataset. We use the ModelNet10 dataset [43] for evaluation. It consists of a large collection of 3D CAD models from 10 object categories. We uniformly sample 2,048 points from each CAD model using the Poisson disk sampling method [44] and normalize them to be zero-mean inside a unit cube.

Implementation Details. For training the auto-encoders, we use the Adam optimizer [45] with an initial learning rate of 0.0005 and momentum 0.9. The input point cloud size N is 2,048. We empirically set the number of subsets to $M = 4$, resulting in the size of each subset to $N/M = 512$. The dimension of latent variables is set to b =128, which yielded the best reconstruction accuracy in the research of Achlioptas et al. [17]. For region growing sampling, we set the threshold for nearest neighbor search $k = 5$. As the coefficient of joint loss described in Sec. III-C, we use $\lambda^l_{adv} = 0.1$, $\lambda^g_{rec} = 10^4$, and $\lambda^l_{rec} = 10^3$. As hyper-parameters with respect to WGANs-gp, we use $\lambda_{gp} = 10$ and $n_{critic} = 5$.

Settings of Our Method. To demonstrate the effectiveness of our joint loss function, we compare the following settings:

- BL: We select auto-encoders consisting of encoders and decoders as the baseline (BL). In this method, we use only the global reconstruction loss as a loss function.
- BL+SD: This is a method of adding sub-decoders (SD) to the baseline. The loss function is a combination of global and local reconstruction loss.
- BL+SD (FPS): This method is a variant of BL+SD which uses the farthest point sampling (FPS) to construct subsets, instead of the region growing sampling.
- Ours: This is a method of adding a discriminator and sub-discriminators to the BL+SD. The loss function is a combination of reconstruction loss and adversarial loss. This is the fully proposed method.

TABLE I: Quantitative evaluations of auto-encoders on the ModelNet10 dataset. The evaluation metric is the Earth Mover's distance (EMD). The symbol "↓" represents lower is better. BL, BL+SD, BL+SD (FPS), and Ours are different settings of auto-encoders: see Sec. IV-A. All auto-encoders are trained for 2,000 epochs. All numbers are scaled by 100.

Metric	Method	Category										
		bathtub	bed	chair	desk	dresser	monitor	stand	sofa	table	toilet	**mean**
EMD ↓	BL [17]	0.567	1.070	1.412	0.592	0.890	0.857	0.889	1.246	0.857	0.957	0.934
	BL+SD	0.547	0.899	1.279	0.573	0.845	0.831	0.669	1.162	0.645	0.700	0.815
	BL+SD (FPS)	0.647	1.294	1.469	0.590	1.052	1.060	0.768	1.318	0.759	0.801	0.976
	Ours	0.502	0.879	1.201	0.562	0.612	0.696	0.622	1.137	0.634	0.649	**0.749**

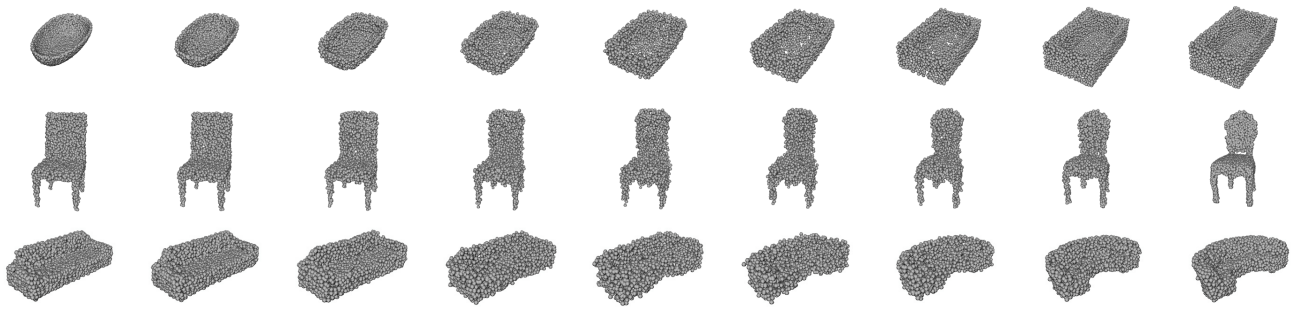


Fig. 5: Shape interpolation results of three categories: bathtub, chair, and sofa. We first obtain latent variables from the point clouds at both ends with encoder. Then we computed their intermediate latent variables using linear interpolation. All point clouds are reconstructed from those latent variables with decoder.

B. Latent Representation Learning

We first train auto-encoders to learn latent representations of the 3D point clouds. Table I shows the quantitative evaluation results of 3D reconstruction. As an indicator of the performance, we use the EMD. All auto-encoders are trained for 2,000 epochs. As can be seen, our approach significantly outperforms the BL on a variety of object classes. Comparing BL and BL+SD, it can be seen that adding a local reconstruction loss improves the performance of point cloud reconstruction. This is because the reconstruction losses of the input point cloud and its local subsets have encouraged the capture of both global and local structure of the point cloud. In contrast to BL+SD, BL+SD (FPS) hardly improves the performance compared to BL. This is due to the fact that there is little shape difference between the input point cloud and its subsets constructed by FPS. In other words, it seems that the effect of our multiple auxiliary losses is reduced because the information obtained from the input point cloud and its subsets are similar. By comparing BL+SD and Ours, we can confirm the effect of adding an adversarial loss. That is, the reconstruction loss and the adversarial loss complementarily improve the fidelity of the point cloud.

We also perform shape interpolation to evaluate latent representations in Ours. Fig. 5 shows the qualitative results. We first obtain a pair of latent variables by encoding the two 3D point clouds shown at both ends of each row in the figure. Then, we linearly interpolate and decode the pair of latent variables. Smooth morphing between the two shapes suggests that latent space is well learned.

C. 3D Point Cloud Generation

We evaluate the performance of our generative models for the 3D point cloud. After training the auto-encoders, we trained the GANs as shown in Fig. 2(b). In this experiment, auto-encoders trained with different settings are independently used for training of the GANs. For stable training of GANs, we also adopt WGANs-gp [41], [42]. All GANs are trained for 2,000 epochs. The 3D point clouds are generated using random Gaussian noise vectors as input to the generator.

Quantitative Evaluation. We quantitatively compared the baseline method and various settings of the proposed method. As indicators of the generation performance, we use Jensen-Shannon Divergence (JSD), Minimum Matching Distance (MMD), and Coverage (COV) introduced in [17]. The MMD and COV measure the fidelity and coverage of a set of point clouds with respect to another one. The JSD evaluates the similarity between two sets of point clouds in a coarser way. Although CD and EMD can be selected for calculating MMD and COV, we always select EMD in this experiment.

Table II shows the quantitative results for 10 categories. The proposed method achieves the best results in all metrics. It indicates that the proposed method can generate the most faithful point clouds with respect to the set of ground-truth point clouds. As can be seen, improving the training methods of auto-encoders boosts the performance of 3D point cloud generation with GANs. This is due to our multi-task learning approach that learned better latent representations compared with single-task learning.

TABLE II: Quantitative evaluations of generative adversarial networks (GANs) on the ModelNet10 dataset. The evaluation metrics are the Jensen-Shannon Divergence (JSD), Minimum Matching Distance (MMD), and Coverage (COV). The symbols "↓" and "↑" represent lower is better and higher is better, respectively. BL, BL+SD, BL+SD (FPS), and Ours are different settings of the auto-encoders: see Sec. IV-A. We use GANs with common architectures, except for auto-encoder's settings. All GANs are trained for 2,000 epochs. All numbers in the rows of JSD and MMD are scaled by 10.

Metric	Method	Category										
		bathtub	bed	chair	desk	dresser	monitor	stand	sofa	table	toilet	**mean**
JSD ↓	BL [17]	0.535	0.170	0.337	0.486	0.525	0.390	0.620	0.112	0.330	0.251	0.375
	BL+SD	0.428	0.135	0.158	0.358	0.313	0.214	0.429	0.104	0.249	0.207	0.259
	BL+SD (FPS)	0.451	0.239	0.248	0.755	0.413	0.392	0.432	0.110	0.436	0.319	0.379
	Ours	0.411	0.127	0.142	0.353	0.303	0.187	0.402	0.101	0.225	0.164	**0.241**
MMD ↓	BL [17]	0.557	0.506	0.660	0.807	0.577	0.561	0.723	0.432	0.589	0.626	0.604
	BL+SD	0.515	0.485	0.634	0.775	0.523	0.520	0.702	0.416	0.565	0.592	0.573
	BL+SD (FPS)	0.522	0.511	0.662	0.799	0.534	0.560	0.710	0.432	0.599	0.630	0.596
	Ours	0.510	0.482	0.629	0.768	0.514	0.514	0.692	0.413	0.560	0.587	**0.567**
COV ↑	BL [17]	0.339	0.421	0.388	0.400	0.425	0.320	0.390	0.392	0.474	0.404	0.395
	BL+SD	0.462	0.473	0.448	0.410	0.470	0.443	0.435	0.447	0.500	0.488	0.457
	BL+SD (FPS)	0.396	0.396	0.388	0.355	0.451	0.339	0.425	0.427	0.431	0.372	0.398
	Ours	0.464	0.487	0.472	0.450	0.475	0.447	0.435	0.451	0.507	0.517	**0.470**

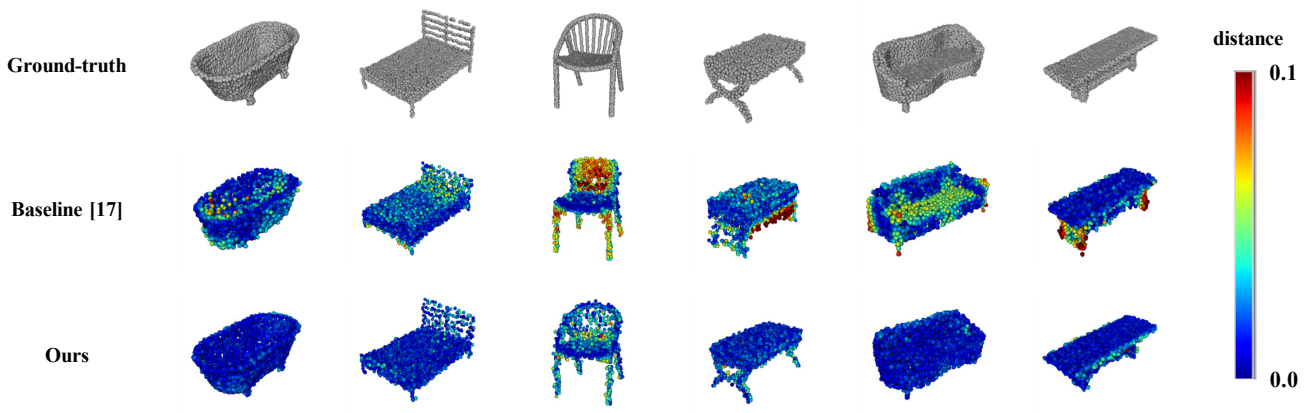


Fig. 6: Qualitative evaluations of generative adversarial networks on the ModelNet10 dataset. The upper row shows ground-truth, and the middle and lower row show 3D point clouds generated by the baseline method and proposed method, respectively. For each ground-truth, we retrieve the nearest neighbor point cloud with respect to the EMD from the generated point clouds. Color represents the one-sided Hausdorff distance between the ground-truth points and the generated points. Best viewed in color.

Qualitative Evaluation. Fig. 6 shows the 3D point clouds generated by the baseline method and the proposed method. For each ground-truth point cloud, we search the nearest neighbor from the set of generated point clouds. EMD is used for the distance measure in this search. The color of the point represents the one-sided Hausdorff distance between the ground-truth point and the generated point. The proposed method provides 3D point clouds with much higher quality compared to the baseline method. These results demonstrate that the proposed method significantly outperforms the baseline method with regard to generate a similar one of the training sets. Since the proposed method captures the local structures of a 3D point cloud, it can generate point clouds with more faithful shapes up to fine parts, such as the frame of the bed. However, the proposed method shows low fidelity to a more complex structure like a chair backrest. Also, it often generates noisy points for finer detailed structures. These limitations are in common with the baseline method.

V. Conclusions

In this paper, we presented an approach to improve the performance of 3D point cloud generative models based on a combination of auto-encoders and generative adversarial networks. In addition, we introduced a sampling algorithm to construct spatially localized subsets. We focused on the training of auto-encoders. Our training method reconstructs input point cloud and its local subsets simultaneously according to a multi-task learning approach. We also defined a joint loss function of reconstruction losses and adversarial losses that measured between input point clouds and reconstructed point clouds. In the experiments, we demonstrated the effectiveness of the proposed method. Compared with the conventional method, the proposed method can generate 3D point clouds with higher fidelity and coverage. In future work, we will apply the proposed method to improve the performance of other tasks such as data augmentation and domain translation.

REFERENCES

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

[2] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox, "Orientation-boosted voxel nets for 3d object recognition," in *IEEE British Machine Vision Conference (BMVC)*, 2017.

[3] G. Riegler, A. O. Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3577–3586.

[4] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner, "Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4578–4587.

[5] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 945–953.

[6] S. Tulsiani, A. A. Efros, and J. Malik, "Multi-view consistency as supervisory signal for learning shape and pose prediction," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2897–2905.

[7] P. Zanuttigh and L. Minto, "Deep learning for 3d shape classification from multiple depth maps," in *IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 3615–3619.

[8] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1907–1915.

[9] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5099–5108.

[10] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 918–927.

[11] H. Deng, T. Birdal, and S. Ilic, "Ppfnet: Global context aware local features for robust 3d point matching," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 195–205.

[12] L. Ge, Y. Cai, J. Weng, and J. Yuan, "Hand pointnet: 3d hand pose estimation using point sets," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8417–8426.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014, pp. 2672–2680.

[14] S.-W. Huang, C.-T. Lin, S.-P. Chen, Y.-Y. Wu, P.-H. Hsu, and S.-H. Lai, "Auggan: Cross domain adaptation with gan-based data augmentation," in *IEEE European Conference on Computer Vision (ECCV)*, 2018, pp. 718–731.

[15] Y. Latif, R. Garg, M. Milford, and I. Reid, "Addressing challenging place recognition tasks using generative adversarial networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2349–2355.

[16] N. Patel, A. N. Saridena, A. Choromanska, P. Krishnamurthy, and F. Khorrami, "Adversarial learning-based on-line anomaly monitoring for assured autonomy," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 6149–6154.

[17] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," in *International Conference on Machine Learning (ICML)*, 2018, pp. 40–49.

[18] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 2672–2680.

[19] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 206–215.

[20] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3d surface generation," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 216–224.

[21] R. Klokov and V. Lempitsky, "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 863–872.

[22] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[23] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *International Conference on Learning Representations (ICLR)*, 2018.

[24] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for gans do actually converge?" in *International Conference on Machine Learning (ICML)*, 2018, pp. 3478–3487.

[25] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016, pp. 613–621.

[26] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, "Mocogan: Decomposing motion and content for video generation," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1526–1535.

[27] Z. Hao, X. Huang, and S. Belongie, "Controllable video generation with sparse trajectories," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7854–7863.

[28] R. Caruana, "Multitask learning," *Machine Learning,*, vol. 28, no. 1, pp. 41–75, 1997.

[29] H. Sun, Z. Meng, P. Y. Tao, and M. H. Ang, "Scene recognition and object detection in a unified convolutional neural network on a mobile manipulator," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5875–5881.

[30] Z. Ren and Y. J. Lee, "Cross-domain self-supervised multi-task feature learning using synthetic imagery," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 762–771.

[31] L. Pinto and A. Gupta, "Learning to push by grasping: Using multiple tasks for effective learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2161–2168.

[32] K. Fang, Y. Bai, S. Hinterstoisser, S. Savarese, and M. Kalakrishnan, "Multi-task domain adaptation for deep learning of instance grasping from simulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3516–3523.

[33] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 605–613.

[34] L. Jiang, S. Shi, X. Qi, and J. Jia, "Gal: Geometric adversarial loss for single-view 3d-object reconstruction," in *IEEE European Conference on Computer Vision (ECCV)*, 2018, pp. 820–834.

[35] G. Elbaz, T. Avraham, and A. Fischer, "3d point cloud registration for localization using a deep neural network auto-encoder," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2472–2481.

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

[37] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.

[38] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*, 2015, pp. 448–456.

[39] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning Workshop (ICMLW)*, 2013.

[40] B. Yang, H. Wen, S. Wang, R. Clark, A. Markham, and N. Trigoni, "3d object reconstruction from a single depth view with adversarial learning," in *IEEE International Conference on Computer Vision Workshop (ICCVW)*, 2017, pp. 679–688.

[41] M. Arjovsky, S. Chintala, , and L. Bottou, "Wasserstein gan," in *arXiv:1701.07875*, 2017.

[42] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5767–5777.

[43] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.

[44] G. Ranzuglia, M. Callieri, M. Dellepiane, P. Cignoni, and R. Scopigno, "Efficient and flexible sampling with blue noise properties of triangular meshes," *Transactions on Visualization and Computer Graphics*, vol. 18, no. 6, pp. 914–924, 2012.

[45] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv:1412.6980*, 2014.