Progressive Point Cloud Deconvolution Generation Network

Le Hui, Rui Xu, Jin Xie(✉), Jianjun Qian, and Jian Yang(✉)

Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education
Jiangsu Key Lab of Image and Video Understanding for Social Security
PCA Lab, School of Computer Science and Engineering
Nanjing University of Science and Technology
{le.hui, xu_ray, csjxie, csjqian, csjyang}@njust.edu.cn

Abstract. In this paper, we propose an effective point cloud generation method, which can generate multi-resolution point clouds of the same shape from a latent vector. Specifically, we develop a novel progressive deconvolution network with the learning-based bilateral interpolation. The learning-based bilateral interpolation is performed in the spatial and feature spaces of point clouds so that local geometric structure information of point clouds can be exploited. Starting from the low-resolution point clouds, with the bilateral interpolation and max-pooling operations, the deconvolution network can progressively output high-resolution local and global feature maps. By concatenating different resolutions of local and global feature maps, we employ the multi-layer perceptron as the generation network to generate multi-resolution point clouds. In order to keep the shapes of different resolutions of point clouds consistent, we propose a shape-preserving adversarial loss to train the point cloud deconvolution generation network. Experimental results on ShpaeNet and ModelNet datasets demonstrate that our proposed method can yield good performance. Our code is available at https://github.com/fpthink/PDGN.

Keywords: Point cloud generation, GAN, deconvolution network, bilateral interpolation

1 Introduction

With the development of 3D sensors such as LiDAR and Kinect, 3D geometric data are widely used in various kinds of computer vision tasks. Due to the great success of generative adversarial network (GAN) [10] in the 2D image domain, 3D data generation [38,5,7,16,36,11,46,45,47] has been receiving more and more attention. Point clouds, as an important 3D data type, can compactly and flexibly characterize geometric structures of 3D models. Different from 2D image data, point clouds are unordered and irregular. 2D generative models cannot be directly extended to point clouds. Therefore, how to generate realistic point clouds in an unsupervised way is still a challenging and open problem.

Recent research efforts have been dedicated to 3D model generation. Based on the voxel representation of 3D models, 3D convolutional neural networks (3D CNNs) can be applied to form 3D GAN [40] for 3D model generation. Nonetheless, since 3D CNNs on the voxel representation require heavy computational and memory burdens, 3D GANs are limited to generate low-resolution 3D models. Different from the regular voxel representation, point clouds are spatially irregular. Therefore, CNNs cannot be directly applied on point clouds to form 3D generative models. Inspired by PointNet [26] that can learn compact representation of point clouds, Achlioptas *et al.* [1] proposed an auto-encoder based point cloud generation network in a supervised manner. Nonetheless, the generation model is not an end-to-end learning framework. Yang *et al.* [44] proposed the PointFlow generation model, which can learn a two-level hierarchical distribution with a continuous normalized flow. Based on graph convolution, Valsesia *et al.* [37] proposed a localized point cloud generation model. Shu *et al.* [31] developed a tree structured graph convolution network for point cloud generation. Due to the high computational complexity of the graph convolution operation, training the graph convolution based generation models is very time-consuming.

In this paper, we propose a simple yet efficient end-to-end generation model for point clouds. We develop a progressive deconvolution network to map the latent vector to the high-dimensional feature space. In the deconvolution network, the learning-based bilateral interpolation is adopted to enlarge the feature map, where the weights are learned from the spatial and feature spaces of point clouds simultaneously. It is desirable that the bilateral interpolation can capture the local geometric structures of point clouds well with the increase of the resolution of generated point clouds. Following the deconvolution network, we employ the multi-layer perceptron (MLP) to generate spatial coordinates of point clouds. By stacking multiple deconvolution networks with different resolutions of point clouds as the inputs, we can form a progressive deconvolution generation network to generate multi-resolution point clouds. Since the shapes of multi-resolution point clouds generated from the same latent vector should be consistent, we formulate a shape-preserving adversarial loss to train the point cloud deconvolution generation network. Extensive experiments are conducted on the ShapeNet [3] and ModelNet [41] datasets to demonstrate the effectiveness of our proposed method. The main contributions of our work are summarized as follows:

- We present a novel progressive point cloud generation framework in an end-to-end manner.
- We develop a new deconvolution network with the learning-based bilateral interpolation to generate high-resolution feature maps.
- We formulate a shape-preserving loss to train the progressive point cloud network so that the shapes of generated multi-resolution point clouds from the same latent vector are consistent.

The rest of the paper is organized as follows: Section 2 introduces related work. In Section 3, we present the progressive end-to-end point cloud generation model. Section 4 presents experimental results and Section 5 concludes the paper.

2 Related Work

2.1 Deep Learning on 3D Data

Existing 3D deep learning methods can be roughly divided into two classes. One class of 3D deep learning methods [33,41,22,27] convert the geometric data to the regular-structured data and apply existing deep learning algorithms to them. The other class of methods [24,32,17,35,26,28] mainly focus on constructing special operations that are suitable to the unstructured geometric data for 3D deep learning.

In the first class of 3D deep learning methods, view-based methods represent the 3D object as a collection of 2D views so that the standard CNN can be directly applied. Specifically, the max-pooling operation across views is used to obtain a compact 3D object descriptor [33]. Voxelization [41,22] is another way to represent the 3D geometric data with regular 3D grids. Based on the voxelization representation, the standard 3D convolution can be easily used to form the 3D CNNs. Nonetheless, the voxelization representation usually leads to the heavy burden of memory and high computational complexity because of the computation of the 3D convolution. Qi *et al.* [27] proposed to combine the view-based and voxelization-based deep learning methods for 3D shape classification.

In 3D deep learning, variants of deep neural networks are also developed to characterize the geometric structures of 3D data. [32,35] formulated the unstructured point clouds as the graph-structured data and employed the graph convolution to form the 3D deep learning representation. Qi *et al.* [26] proposed PointNet that treats each point individually and aggregates point features through several MLPs followed by the max-pooling operation. Since PointNet cannot capture the local geometric structures of point clouds well, Qi *et al.* [28] proposed PointNet++ to learn the hierarchical feature representation of point clouds. By constructing the k-nearest neighbor graph, Wang *et al.* [39] proposed an edge convolution operation to form the dynamic graph CNN for point clouds. Li *et al.* [19] proposed PointCNN for feature learning from point clouds, where the χ-transform is learned to form the χ-convolution operation.

2.2 3D Point Cloud Generation

Variational auto-encoder (VAE) is an important type of generative model. Recently, VAE has been applied to point cloud generation. Gadelha *et al.* [8] proposed MRTNet to generate point clouds from a single image. Specifically, using a VAE framework, a 1D ordered list of points is fed to the multi-resolution encoder and decoder to perform point cloud generation in unsupervised learning. Zamorski *et al.* [46] applied the VAE and adversarial auto-encoder (AAE) to point cloud generation. Since the VAE model requires the particular prior distribution to make KL divergence tractable, the AAE is introduced to learn the prior distribution by utilizing adversarial training. Lately, Yang *et al.* [44] proposed a probabilistic framework (PointFlow) to generate point clouds by modeling them as a two-level hierarchical distribution. Nonetheless, as mentioned

in PointFlow [44], it converges slowly and fails for the cases with many thin structures (like chairs).

Generative adversarial network (GAN) has also achieved great success in the field of image generation [2,6,21,29,23]. Recently, a series of attractive works [7,5,12,43,31] ignite a renewed interest in the 3D object generation task by adopting CNNs. Wu *et al.* [40] first proposed 3D-GAN, which can generate 3D objects from a probabilistic space by using the volumetric convolutional network and GAN. Zhu *et al.* [48] proposed a GAN-based neural network that can leverage information extracted from 2D images to improve the quality of generated 3D models. However, due to the sparsely occupied 3D grids of the 3D object, the volumetric representation approach usually faces a heavy memory burden, resulting in the high computational complexity of the volumetric convolutional network. To alleviate the memory burden, Achlioptas *et al.* [1] proposed a two-stage deep generative model with an auto-encoder for point clouds. It first maps data points into the latent representation and then trains a minimal GAN in the learned latent space to generate point clouds. However, the two-stage point cloud generation model cannot be trained in an end-to-end manner. Based on graph convolution, Valsesia *et al.* [37] focused on designing a graph-based generator that can learn the localized features of point clouds. Shu *et al.* [31] developed a tree structured graph convolution network for 3D point cloud generation.

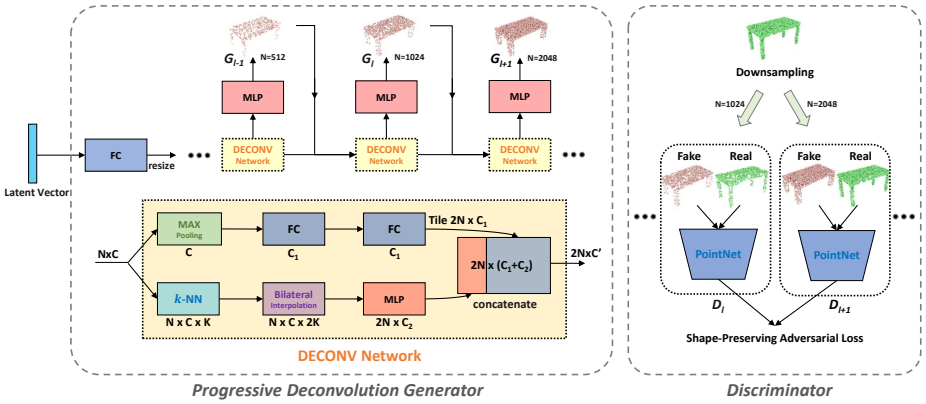

Fig. 1. The architecture of our progressive point cloud framework. The progressive deconvolution generator aims to generate point clouds, while the discriminator distinguishes it from the real point clouds.

3 Our Approach

In this section, we present our progressive generation model for 3D point clouds. The framework of our proposed generation model is illustrated in Fig. 1. In

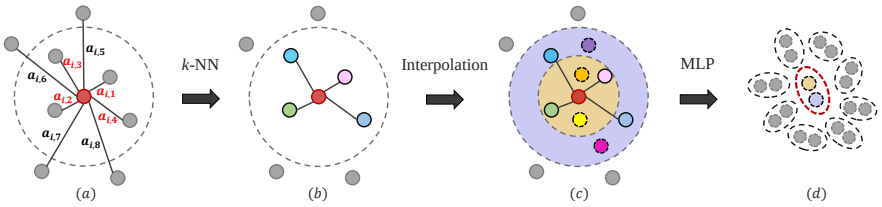

Fig. 2. The constructed deconvolution operation. First, we define the similarity between point pairs in the feature space (a). We choose the k nearest neighbor points (k-NN) in the feature space with the defined similarity in (b). Then we interpolate in the neighborhood to form an enlarged feature map in (c). Finally, we apply the MLP to generate new high-dimensional feature maps in (d). Note that we can obtain double numbers of points through the deconvolution operation.

Section 3.1, we describe how to construct the proposed progressive deconvolution generation network. In Section 3.2, we present the details of the shape-preserving adversarial loss to train the progressive deconvolution generation network.

3.1 Progressive deconvolution generation network

Given a latent vector, our goal is to generate high-quality 3D point clouds. One key problem in point cloud generation is how to utilize a one-dimensional vector to generate a set of 3D points consistent with the 3D object in geometry. To this end, we develop a special deconvolution network for 3D point clouds, where we first obtain the high-resolution feature map with the learning-based bilateral interpolation and then apply MLPs to generate the local and global feature maps. It is desirable that the fusion of the generated local and global feature maps can characterize the geometric structures of point clouds in the high-dimensional feature space.

Learning-based bilateral interpolation. Due to the disordered and irregular structure of point clouds, we cannot directly perform the interpolation operation on the feature map. Therefore, we need to build a neighborhood for each point on the feature map to implement the interpolation operation. In this work, we simply employ the k-nearest neighbor (k-NN) to construct the neighborhood of each point in the feature space. Specifically, given an input with N feature vectors $\boldsymbol{x}_i \in \mathbb{R}^d$, the similarity between points i and j is defined as:

$$a_{i,j} = \exp\left(-\beta \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2^2\right) \tag{1}$$

where β is empirically set as $\beta = 1$ in our experiments. As shown in Figs. 2 (a) and (b), we can choose k nearest neighbor points in the feature space with the defined similarity. And the parameter k is set as $k = 20$ in this paper.

Once we obtain the neighborhood of each point, we can perform the interpolation in it. As shown in Fig. 2 (c), with the interpolation, k points in the

neighborhood can be generated to $2k$ points in the feature space. Classical interpolation methods such as linear and bilinear interpolations are non-learning interpolation methods, which cannot be adaptive to different classes of 3D models during the point cloud generation process. Moreover, the classical interpolation methods does not exploit neighborhood information of each point in the spatial and feature space simultaneously.

To this end, we propose a learning-based bilateral interpolation method that utilizes the spatial coordinates and features of the neighborhood of each point to generate the high-resolution feature map. Given the point $\boldsymbol{p}_i \in \mathbb{R}^3$ and k points in its neighborhood, we can formulate the bilateral interpolation as:

$$\tilde{\boldsymbol{x}}_{i,l} = \frac{\sum_{j=1}^{k} \varphi_l\left(\boldsymbol{p}_i, \boldsymbol{p}_j\right) \phi_l\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \boldsymbol{x}_{j,l}}{\sum_{j=1}^{k} \varphi_l\left(\boldsymbol{p}_i, \boldsymbol{p}_j\right) \phi_l\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)} \tag{2}$$

where $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ are the 3D spatial coordinates, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are the d-dimensional feature vectors, $\boldsymbol{\varphi}\left(\boldsymbol{p}_i, \boldsymbol{p}_j\right) \in \mathbb{R}^d$ and $\boldsymbol{\phi}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \in \mathbb{R}^d$ are two embeddings in the spatial and feature spaces, $\tilde{\boldsymbol{x}}_{i,l}$ is the l-th element of the interpolated feature $\tilde{\boldsymbol{x}}_i$, $l = 1, 2, \cdots, d$. The embeddings $\boldsymbol{\varphi}\left(\boldsymbol{p}_i, \boldsymbol{p}_j\right)$ and $\boldsymbol{\phi}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ can be defined as:

$$\boldsymbol{\varphi}\left(\boldsymbol{p}_i, \boldsymbol{p}_j\right) = \mathrm{ReLU}(\boldsymbol{W}_{\theta,j}^{\top}\left(\boldsymbol{p}_i - \boldsymbol{p}_j\right)), \quad \boldsymbol{\phi}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \mathrm{ReLU}(\boldsymbol{W}_{\psi,j}^{\top}\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)) \tag{3}$$

where ReLU is the activation function, $\boldsymbol{W}_{\theta,j} \in \mathbb{R}^{3\times d}$ and $\boldsymbol{W}_{\psi,j} \in \mathbb{R}^{d\times d}$ are the weights to be learned. Based on the differences between the points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$, $\boldsymbol{p}_i - \boldsymbol{p}_j$ and $\boldsymbol{x}_i - \boldsymbol{x}_j$, the embeddings $\boldsymbol{\varphi}\left(\boldsymbol{p}_i, \boldsymbol{p}_j\right)$ and $\boldsymbol{\phi}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ can encode local structure information of the point $\boldsymbol{p}_i$ in the spatial and feature spaces, respectively. It is noted that in Eq. 2 the channel-wise bilateral interpolation is adopted. As shown in Fig. 3, the new interpolated feature $\tilde{\boldsymbol{x}}_i$ can be obtained from the neighborhood of $\boldsymbol{x}_i$ with the bilateral weight. For each point, we perform the bilateral interpolation in the k-neighborhood to generate new k points. Therefore, we can obtain a high-resolution feature map, where the neighborhood of each point contains $2k$ points.

After the interpolation, we then apply the convolution on the enlarged feature maps. For each point, we divide the neighborhood of $2k$ points into two regions according to the distance. As shown in Fig. 2 (c), the closest k points belong to the first region and the rest as the second region. Similar to PointNet [26], we first use the MLP to generate high-dimensional feature maps and then use the max-pooling operation to obtain the local features of the two interpolated points from two regions. As shown in Fig. 2 (d), we can double the number of points from the inputs through the deconvolution network to generate a high-resolution local feature map $\boldsymbol{X}_{local}$. We also use the max-pooling operation to extract the global feature of point clouds. By replicating the global feature for N times, where N is the number of points, we can obtain the high-resolution global feature map $\boldsymbol{X}_{global}$. Then we concatenate the local feature map $\boldsymbol{X}_{local}$ and the global feature map $\boldsymbol{X}_{global}$ to obtain the output of the deconvolution network $\boldsymbol{X}_c = [\boldsymbol{X}_{local}; \boldsymbol{X}_{global}]$. Thus, the output $\boldsymbol{X}_c$ can not only characterize the local geometric structures of point clouds, but also capture the global shape of point clouds during the point cloud generation process.

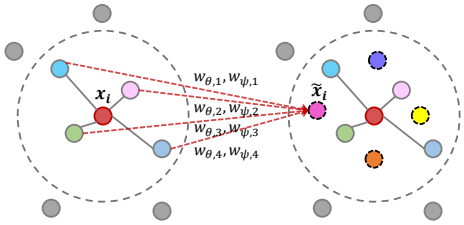

Fig. 3. The illustration of the learning-based bilateral interpolation method. The points in the neighborhood of the center point $\boldsymbol{x}_i$ are colored. We interpolate new points by considering the local geometric features of the points in the neighborhood. $\boldsymbol{W}_{\theta,j}$ and $\boldsymbol{W}_{\psi,j}$, $j = 1, 2, 3, 4$, are the weights in the spatial and feature spaces to be learned.

3D point cloud generation. Our goal is to progressively generate 3D point clouds from the low resolution to the high resolution. Stacked deconvolution networks can progressively double the number of points and generate their high-dimensional feature maps. We use the MLP after each deconvolution network to generate the 3D coordinates of point clouds at each resolution. Note that two outputs of the DECONV block are the same, one for generating 3D coordinates of point clouds and the other as the features of the point clouds. We concatenate the generated 3D coordinates with the corresponding features as the input to the next DECONV block.

3.2 Shape-preserving adversarial loss

Shape-consistent constraint. During the training process, different resolutions of 3D point clouds are generated. With the increase of the resolution of the output of the progressive deconvolution network, generated point clouds become denser. It is expected that the local geometric structures of the generated point clouds are as consistent as possible between different resolutions. Since our progressive deconvolution generation network is an unsupervised generation model, it is difficult to distinguish different shapes from the same class of 3D objects for the discriminator. Thus, for the specific class of 3D objects, the deconvolution generation networks at different resolutions might generate 3D point clouds with different shapes. Therefore, we encourage that the means and covariances of the neighborhoods of the corresponding points between different resolutions are as close as possible so that the corresponding parts of different resolutions of generated point clouds are consistent.

Shape-preserving adversarial loss. We employ the mean and covariance of the neighborhoods of the corresponding points to characterize the consistency of the generated point clouds between different resolutions. We use the farthest point sampling (FPS) to choose centroid points from each resolution and find the k-neighborhoods for centroid points. The mean and covariance of the neigh-

borhood of the i-th centroid point are represented as:

$$\boldsymbol{\mu}_i = \frac{\sum_{j \in \mathcal{N}_i} \boldsymbol{p}_j}{k}, \quad \boldsymbol{\sigma}_i = \frac{\sum_{j \in \mathcal{N}_i} (\boldsymbol{p}_j - \boldsymbol{\mu}_i)^\top (\boldsymbol{p}_j - \boldsymbol{\mu}_i)}{k-1} \tag{4}$$

where $\mathcal{N}_i$ is the neighborhood of the centroid point, $\boldsymbol{p}_j \in \mathbb{R}^3$ is the coordinates of the point cloud, $\boldsymbol{\mu}_i \in \mathbb{R}^3$ and $\boldsymbol{\sigma}_i \in \mathbb{R}^{3\times3}$ are the mean and covariance of the neighborhood, respectively.

Since the sampled centroid points are not completely matched between adjacent resolutions, we employ the Chamfer distances of the means and covariances to formulate the shape-preserving loss. We denote the centroid point sets at the resolutions l and $l+1$ by S_l and S_{l+1}, respectively. The Chamfer distance $d_1(S_l, S_{l+1})$ between the means of the neighborhoods from the adjacent resolutions is defined as:

$$d_1(S_l, S_{l+1}) = \max \left\{ \frac{1}{|S_l|} \sum_{i \in S_l} \min_{j \in S_{l+1}} \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|_2 \ , \quad \frac{1}{|S_{l+1}|} \sum_{j \in S_{l+1}} \min_{i \in S_l} \|\boldsymbol{\mu}_j - \boldsymbol{\mu}_i\|_2 \right\} \tag{5}$$

Similarly, the Chamfer distance $d_2(S_l, S_{l+1})$ between the covariances of the neighborhoods is defined as:

$$d_2(S_l, S_{l+1}) = \max \left\{ \frac{1}{|S_l|} \sum_{i \in S_l} \min_{j \in S_{l+1}} \|\boldsymbol{\sigma}_i - \boldsymbol{\sigma}_j\|_F \ , \quad \frac{1}{|S_{l+1}|} \sum_{j \in S_{l+1}} \min_{i \in S_l} \|\boldsymbol{\sigma}_j - \boldsymbol{\sigma}_i\|_F \right\} \tag{6}$$

The shape-preserving loss (SPL) for multi-resolution point clouds is defined as:

$$SPL(G_l, G_{l+1}) = \sum_{l=1}^{M-1} d_1(S_l, S_{l+1}) + d_2(S_l, S_{l+1}) \tag{7}$$

where M is the number of resolutions, G_l and G_{l+1} represents the l-th and $(l+1)$-th point cloud generators, respectively.

Based on Eq. 7, for the generator G_l and discriminator D_l, we define the following shape-preserving adversarial loss:

$$\begin{aligned} L(D_l) &= E_{\boldsymbol{s} \sim p_{real}(\boldsymbol{s})}(\log D_l(\boldsymbol{s}) + \log(1 - D_l(G_l(\boldsymbol{z})))) \\ L(G_l) &= E_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}(\log(1 - D_l(G_l(\boldsymbol{z})))) + \lambda SPL(G_l(\boldsymbol{z}), G_{l+1}(\boldsymbol{z})) \end{aligned} \tag{8}$$

where $\boldsymbol{s}$ is the real point cloud sample, $\boldsymbol{z}$ is the randomly sampled latent vector from the distribution $p(\boldsymbol{z})$ and λ is the regularization parameter. Note that we ignore the SPL in $L(G_l)$ for $l = M$. Thus, multiple generators G and discriminators D can be trained with the following equation:

$$max_D \sum_{l=1}^{M} L(D_l), min_G \sum_{l=1}^{M} L(G_l) \tag{9}$$

where $D = \{D_1, D_2, \cdots, D_M\}$ and $G = \{G_1, G_2, \cdots, G_M\}$. During the training process, multiple generators G and discriminators D are alternatively optimized till convergence.

4 Experiments

4.1 Experimental Settings

We evaluate our proposed generation network on three popular datasets including ShapeNet [3], ModelNet10 and ModelNet40 [41]. ShapeNet is a richly annotated large-scale point cloud dataset containing 55 common object categories and 513,000 unique 3D models. In our experiments, we only use 16 categories of 3D objects. ModelNet10 and ModelNet40 are subsets of ModelNet, which contain 10 categories and 40 categories of CAD models, respectively.

Our proposed framework mainly consists of progressive deconvolution generator and shape-preserving discriminator. In this paper, we generate four resolutions of point clouds from a 128-dimensional latent vector. In the generator, the output size of 4 deconvolution networks are 256×32, 512×64, 1024×128 and 2048×256. We use MLPs to generate coordinates of point clouds. Note that MLPs are not shared for 4 resolutions. After the MLP, we adopt the $Tanh$ activation function. In the discriminator, we use 4 PointNet-like structures. For different resolutions, the network parameters of the discriminators are different. We use Leaky ReLU [42] and batch normalization [13] after every layer. The more detailed structure of our framework is shown in the supplementary material. In addition, we use the $k = 20$ nearest points as the neighborhood for the bilateral interpolation. During the training process, we adopt Adam [15] with the learning rate 10^{-4} for both generator and discriminator. We employ an alternative training strategy in [10] to train the generator and discriminator. Specifically, the discriminator is optimized for each generator step.

4.2 Evaluation of point cloud generation

Visual results. As shown in Fig. 4, on the ShapeNet [3] dataset, we visualize the synthesized point clouds containing 4 categories, which are "Airplane", "Table", "Chair", and "Lamp", respectively. Due to our progressive generator, each category contains four resolutions of point clouds generated from the same latent vector. It can be observed that the geometric structures of different resolutions of generated point clouds are consistent. Note that the generated point clouds contain detailed structures, which are consistent with those of real 3D objects. More visualizations are shown in the supplementary material.

Quantitative evaluation. To conduct a quantitative evaluation of the generated point clouds, we adopt the evaluation metric proposed in [1,20], including Jensen-Shannon Divergence (JSD), Minimum Matching Distance (MMD), and Coverage (COV), the earth mover's distance (EMD), the chamfer distance (CD) and the 1-nearest neighbor accuracy (1-NNA). JSD measures the marginal distributions between the generated samples and real samples. MMD is the distance between one point in the real sample set and its nearest neighbors in the generation set. COV measures the fraction of point clouds in the real sample set that can be matched at least one point in the generation set. 1-NNA is used as a metric to evaluate whether two distributions are identical for two-sample tests.

Fig. 4. Generated point clouds including "Airplane", "Table", "Chair" and "Lamp". Each category has four resolutions of point clouds (256, 512, 1024 and 2048).

Table. 1 lists our results with different criteria on the "Airplane" and "Chair" categories in the ShapeNet dataset. In Table. 1, except for PointFlow [44] (VAE-based generation method), the others are GAN-based generation methods. For these evaluation metrics, in most cases, our point cloud deconvolution generation network (PDGN) outperforms other methods, demonstrating the effectiveness of the proposed method. Moreover, the metric results on the "Car" category and the mean result of all 16 categories are shown in the supplementary material.

Table 1. The results on the "Airplane" and "Chair" categories. Note that JSD scores and MMD-EMD scores are multiplied by 10^2, while MMD-CD scores are multiplied by 10^3.

Category	Model	JSD (↓)	MMD (↓)		COV (%, ↑)		1-NNA (%, ↓)	
			CD	EMD	CD	EMD	CD	EMD
Airplane	r-GAN [1]	7.44	0.261	5.47	42.72	18.02	93.50	99.51
	l-GAN (CD) [1]	4.62	0.239	4.27	43.21	21.23	86.30	97.28
	l-GAN (EMD) [1]	3.61	0.269	3.29	47.90	50.62	87.65	85.68
	PC-GAN [18]	4.63	0.287	3.57	36.46	40.94	94.35	92.32
	GCN-GAN [37]	8.30	0.800	7.10	31.00	14.00	-	-
	tree-GAN [31]	9.70	0.400	6.80	61.00	20.00	-	-
	PointFlow [44]	4.92	**0.217**	3.24	46.91	48.40	75.68	75.06
	PDGN (ours)	**3.32**	0.281	**2.91**	**64.98**	**53.34**	**63.15**	**60.52**
Chair	r-GAN [1]	11.5	2.57	12.8	33.99	9.97	71.75	99.47
	l-GAN (CD) [1]	4.59	2.46	8.91	41.39	25.68	64.43	85.27
	l-GAN (EMD) [1]	2.27	2.61	7.85	40.79	41.69	64.73	65.56
	PC-GAN [18]	3.90	2.75	8.20	36.50	38.98	76.03	78.37
	GCN-GAN [37]	10.0	2.90	9.70	30.00	26.00	-	-
	tree-GAN [31]	11.9	**1.60**	10.1	58.00	30.00	-	-
	PointFlow [44]	1.74	2.24	7.87	46.83	46.98	60.88	59.89
	PDGN (ours)	**1.71**	1.93	**6.37**	**61.90**	**57.89**	**52.38**	**57.14**

Different from the existing GAN-based generation methods, we develop a progressive generation network to generate multi-resolution point clouds. In order to

generate the high-resolution point clouds, we employ the bilateral interpolation in the spatial and feature spaces of the low-resolution point clouds to produce the geometric structures of the high-resolution point clouds. Thus, with the increase of resolutions, the structures of generated point clouds are more and more clear. Therefore, our PDGN can yield better performance in terms of these evaluation criteria. In addition, compared to PointFlow, our method can perform better on point clouds with thin structures. As shown in Fig. 5, it can be seen that our method can generate more complete point clouds. Since in PointFlow the VAE aims to minimize the lower bound of the log-likelihood of the latent vector, it may fail for point clouds with thin structures. Nonetheless, due to the bilateral deconvolution and progressive generation from the low resolution to the high resolution, our PDGN can still achieve good performance for point cloud generation with thin structures. For more visualization comparisons to PointFlow please refer to the supplementary material.

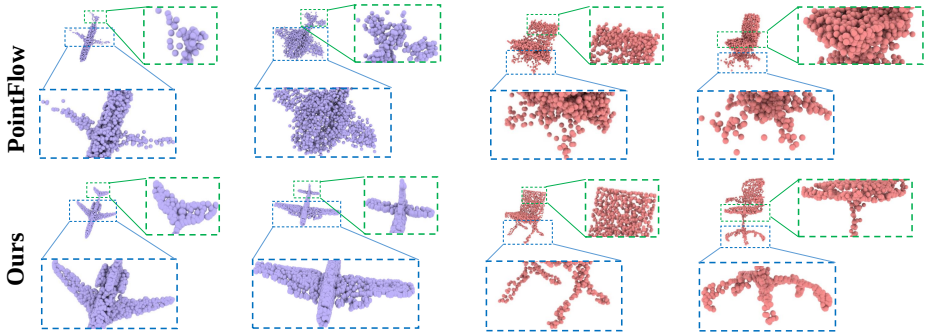

Fig. 5. Visualization results on the "Airplane" and "Chair" categories.

Classification results. Following [40,44], we also conduct the classification experiments on ModelNet10 and ModelNet40 to evaluate our generated point clouds. We first use all samples from ModelNet40 to train our network with the iteration of 300 epochs. Then we feed all samples from ModelNet40 to the trained discriminator (PointNet) for feature extraction. With these features, we simply train a linear SVM to classify the generated point clouds. The settings of ModelNet10 are consistent with ModelNet40. The classification results are listed in Table. 2. Note that for a fair comparison we only compare the point cloud generation methods in the classification experiment. It can be found that our PDGN outperforms the state-of-the-art point cloud generation methods on the ModelNet10 and ModelNet40 datasets. The results indicate that the generator in our framework can extract discriminative features. Thus, our generator can produce high-quality 3D point clouds.

Computational cost. We compare our proposed method to PointFlow and tree-GAN in terms of the training time and GPU memory. We conduct

Table 2. Classification results on ModelNet10 (MN10) and ModelNet40 (MN40).

Model	MN10 (%)	MN40 (%)
SPH [14]	79.8	68.2
LFD [4]	79.9	75.5
T-L Network [9]	-	74.4
VConv-DAE [30]	80.5	75.5
3D-GAN [40]	91.0	83.3
PointGrow [34]	-	85.7
MRTNet [8]	91.7	86.4
PointFlow [44]	93.7	86.8
PDGN (ours)	**94.2**	**87.3**

point cloud generation experiments on the "Airplane" category in the ShapeNet dataset. For a fair comparison, both codes are run on a single Tesla P40 GPU using the PyTorch [25] framework. For training 1000 iterators with 2416 samples of the "Airplane" category, our proposed method costs about 1.9 days and 15G GPU memory, while PointFlow costs about 4.5 days and 7.9G GPU memory, and tree-GAN costs about 2.5 days and 9.2G GPU memory. Our GPU memory is larger than others due to the four discriminators.

4.3 Ablation study and analysis

Bilateral interpolation. We conduct the experiments with different ways to generate the high-resolution feature maps, including the conventional reshape operation, bilinear interpolation and learning-based bilateral interpolation. In the conventional reshape operation, we resize the feature maps to generate new points. As shown in Fig. 6, we visualize the generated point clouds from different categories. One can see that the learning-based bilateral interpolation can generate more realistic objects than the other methods. For example, for the "Table" category, with the learning-based bilateral interpolation, the table legs are clearly generated. On the contrary, with the bilinear interpolation and reshape operation, the generated table legs are not complete. Besides, we also conduct a quantitative evaluation of generated point clouds. As shown in Table. 3, on the "Chair" category, PDGN with the bilateral interpolation can obtain the best metric results. In contrast to the bilinear interpolation and reshape operation, the learning-based bilateral interpolation exploits the spatial coordinates and high-dimensional features of the neighboring points to adaptively learn weights for different classes of 3D objects. Thus, the learned weights in the spatial and feature spaces can characterize the geometric structures of point clouds better. Therefore, the bilateral interpolation can yield good performance.

Shape-preserving adversarial loss. To demonstrate the effectiveness of our shape-preserving adversarial loss, we train our generation model with the classical adversarial loss, EMD loss, CD loss and shape-preserving loss. It is noted that in the EMD loss and CD loss we replace the shape-preserving constraint

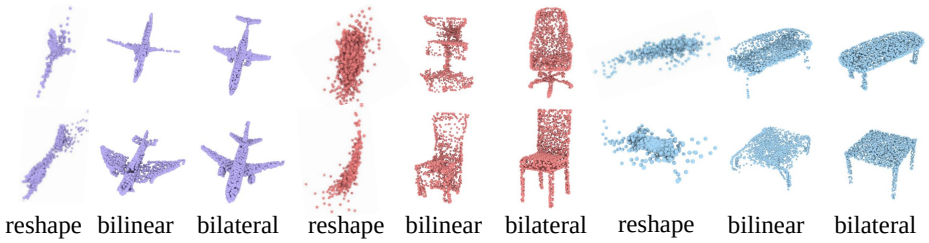

Fig. 6. Visualization results with different operations in the deconvolution network.

Table 3. The ablation study results on the "Chair" category.

Model	JSD (↓)	MMD (↓)		COV (%, ↑)		1-NNA (%, ↓)	
		CD	EMD	CD	EMD	CD	EMD
PDGN (reshape)	8.69	3.38	9.30	55.01	44.49	82.60	80.43
PDGN (bilinear interpolation)	5.02	3.31	8.83	53.84	48.35	69.23	68.18
PDGN (bilateral interpolation)	**1.71**	**1.93**	**6.37**	**61.90**	**57.89**	**52.38**	**57.14**
PDGN (adversarial loss)	3.28	3.00	8.82	56.15	53.84	57.14	66.07
PDGN (EMD loss)	3.35	3.03	8.80	53.84	53.34	60.89	68.18
PDGN (CD loss)	3.34	3.38	9.53	55.88	52.63	59.52	67.65
PDGN (shape-preserving loss)	**1.71**	**1.93**	**6.37**	**61.90**	**57.89**	**52.38**	**57.14**
PDGN (256 points)	5.57	5.12	9.69	39.47	42.85	67.56	70.27
PDGN (512 points)	4.67	4.89	9.67	47.82	51.17	71.42	67.86
PDGN (1024 points)	2.18	4.53	11.0	56.45	55.46	64.71	70.58
PDGN (2048 points)	**1.71**	**1.93**	**6.37**	**61.90**	**57.89**	**52.38**	**57.14**

(Eq. 7) with the Earth mover's distance and Chamfer distance of point clouds between the adjacent resolutions, respectively. We visualize the generated points with different loss functions in Fig. 7. One can see that the geometric structures of different resolutions of generated point clouds are consistent with the shape-preserving adversarial loss. Without the shape-preserving constraint on the multiple generators, the classical adversarial loss cannot guarantee the consistency of generated points between different resolutions. Although the EMD and CD losses impose the constraint on different resolutions of point clouds, the loss can only make the global structures of point clouds consistent. On the contrary, the shape-preserving loss can keep the consistency of the local geometric structures of multi-resolution point clouds with the mean and covariance of the neighborhoods. Thus, our method with the shape-preserving loss can generate high-quality point clouds. Furthermore, we also conduct a quantitative evaluation of generated point clouds. As shown in Table. 3, metric results show that the shape-preserving loss can obtain better results than the other three losses.

Point cloud generation with different resolutions. To verify the effectiveness of our progressive generation framework, we evaluate the metric results of generated point clouds in the cases of different resolutions. As shown in Ta-

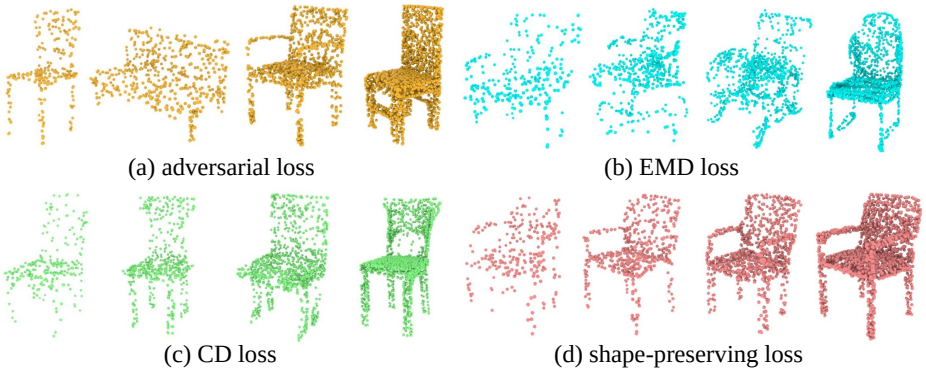

Fig. 7. Visualization results of generated point clouds with different loss functions. For each loss, four resolutions of point clouds (256, 512, 1024 and 2048) are visualized.

ble. 3, for the "Chair" category, we report the results in the cases of four resolutions. One can see that as the resolution increases, the quality of the generated point clouds is gradually improved in terms of the evaluation criteria. As shown in Fig. 4, with the increase of resolutions, the local structures of point clouds become clearer. This is because our progressive generation framework can exploit the bilateral interpolation based deconvolution to generate the coarse-to-fine geometric structures of point clouds.

5 Conclusions

In this paper, we proposed a novel end-to-end generation model for point clouds. Specifically, we developed a progressive deconvolution network to generate multi-resolution point clouds from the latent vector. In the deconvolution network, we employed the learning-based bilateral interpolation to generate high-resolution feature maps so that the local structures of point clouds can be captured during the generation process. In order to keep the geometric structure of the generated point clouds at different resolutions consistent, we formulated the shape-preserving adversarial loss to train the point cloud deconvolution network. Experimental results on ShapeNet and ModelNet datasets verify the effectiveness of our proposed progressive point cloud deconvolution network.

Acknowledgments

This work was supported by the National Science Fund of China (Grant Nos. U1713208, 61876084, 61876083), Program for Changjiang Scholars.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3d point clouds. In: ICML (2018)
2. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan. arXiv preprint arXiv:1701.07875 (2017)
3. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
4. Chen, D.Y., Tian, X.P., Shen, Y.T., Ouhyoung, M.: On visual similarity based 3d model retrieval. CGF (2003)
5. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: ECCV (2016)
6. Denton, E.L., Chintala, S., Fergus, R., et al.: Deep generative image models using alaplacian pyramid of adversarial networks. In: NeurIPS (2015)
7. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: CVPR (2017)
8. Gadelha, M., Wang, R., Maji, S.: Multiresolution tree networks for 3d point cloud processing. In: ECCV (2018)
9. Girdhar, R., Fouhey, D.F., Rodriguez, M., Gupta, A.: Learning a predictable and generative vector representation for objects. In: ECCV (2016)
10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NeurIPS (2014)
11. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: A papier-mâché approach to learning 3d surface generation. In: CVPR (2018)
12. Gwak, J., Choy, C.B., Chandraker, M., Garg, A., Savarese, S.: Weakly supervised 3d reconstruction with adversarial constraint. In: 3DV (2017)
13. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
14. Kazhdan, M., Funkhouser, T., Rusinkiewicz, S.: Rotation invariant spherical harmonic representation of 3d shape descriptors. In: SGP (2003)
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
16. Kulkarni, N., Misra, I., Tulsiani, S., Gupta, A.: 3d-relnet: Joint object and relational network for 3d prediction. In: ICCV (2019)
17. Landrieu, L., Simonovsky, M.: Large-scale point cloud semantic segmentation with superpoint graphs. In: CVPR (2018)
18. Li, C.L., Zaheer, M., Zhang, Y., Póczos, B., Salakhutdinov, R.: Point cloud gan. arXiv preprint arXiv:1810.05795 (2018)
19. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. In: NeurIPS (2018)
20. Lopez-Paz, D., Oquab, M.: Revisiting classifier two-sample tests. In: ICLR (2016)
21. Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Paul Smolley, S.: Least squares generative adversarial networks. In: ICCV (2017)
22. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: IROS (2015)
23. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
24. Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In: CVPR (2017)

25. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS (2019)
26. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR (2017)
27. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view cnns for object classification on 3d data. In: CVPR (2016)
28. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS (2017)
29. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
30. Sharma, A., Grau, O., Fritz, M.: Vconv-dae: Deep volumetric shape learning without object labels. In: ECCV (2016)
31. Shu, D.W., Park, S.W., Kwon, J.: 3d point cloud generative adversarial network based on tree structured graph convolutions. In: ICCV (2019)
32. Simonovsky, M., Komodakis, N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: CVPR (2017)
33. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: ICCV (2015)
34. Sun, Y., Wang, Y., Liu, Z., Siegel, J.E., Sarma, S.E.: Pointgrow: Autoregressively learned point cloud generation with self-attention. arXiv preprint arXiv:1810.05591 (2018)
35. Te, G., Hu, W., Guo, Z., Zheng, A.: Rgcnn: Regularized graph cnn for point cloud segmentation. In: ACM MM (2018)
36. Tulsiani, S., Gupta, S., Fouhey, D.F., Efros, A.A., Malik, J.: Factoring shape, pose, and layout from the 2d image of a 3d scene. In: CVPR (2018)
37. Valsesia, D., Fracastoro, G., Magli, E.: Learning localized generative models for 3d point clouds via graph convolution. In: ICLR (2018)
38. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2mesh: Generating 3d mesh models from single rgb images. In: ECCV (2018)
39. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. arXiv preprint arXiv:1801.07829 (2018)
40. Wu, J., Zhang, C., Xue, T., Freeman, B., Tenenbaum, J.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: NeurIPS (2016)
41. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: CVPR (2015)
42. Xu, B., Wang, N., Chen, T., Li, M.: Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853 (2015)
43. Yang, B., Wen, H., Wang, S., Clark, R., Markham, A., Trigoni, N.: 3d object reconstruction from a single depth view with adversarial learning. In: ICCV (2017)
44. Yang, G., Huang, X., Hao, Z., Liu, M.Y., Belongie, S., Hariharan, B.: Pointflow: 3d point cloud generation with continuous normalizing flows. In: ICCV (2019)
45. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: CVPR (2018)
46. Zamorski, M., Zikeba, M., Nowak, R., Stokowiec, W., Trzcinski, T.: Adversarial autoencoders for compact representations of 3d point clouds. arXiv preprint arXiv:1811.07605 (2018)
47. Zhao, Y., Birdal, T., Deng, H., Tombari, F.: 3d point capsule networks. In: CVPR (2019)

48. Zhu, J., Xie, J., Fang, Y.: Learning adversarial 3d model generation with 2d image enhancer. In: AAAI (2018)