Learning Progressive Point Embeddings for 3D Point Cloud Generation

Cheng Wen Baosheng Yu Dacheng Tao
School of Computer Science, Faculty of Engineering,
The University of Sydney, 6 Cleveland St, Darlington, NSW 2008, Australia
{cwen6671@uni., baosheng.yu@, dacheng.tao@}sydney.edu.au

Abstract

Generative models for 3D point clouds are extremely important for scene/object reconstruction applications in autonomous driving and robotics. Despite recent success of deep learning-based representation learning, it remains a great challenge for deep neural networks to synthesize or reconstruct high-fidelity point clouds, because of the difficulties in 1) learning effective pointwise representations; and 2) generating realistic point clouds from complex distributions. In this paper, we devise a dual-generators framework for point cloud generation, which generalizes vanilla generative adversarial learning framework in a progressive manner. Specifically, the first generator aims to learn effective point embeddings in a breadth-first manner, while the second generator is used to refine the generated point cloud based on a depth-first point embedding to generate a robust and uniform point cloud. The proposed dual-generators framework thus is able to progressively learn effective point embeddings for accurate point cloud generation. Experimental results on a variety of object categories from the most popular point cloud generation dataset, ShapeNet, demonstrate the state-of-the-art performance of the proposed method for accurate point cloud generation.

1. Introduction

3D data representation has received increasing attention from the community due to the rapid development of robot perception and scene/object reconstruction technologies, especially in autonomous driving, robotics, and augmented reality applications [29, 24]. As a simple yet effective 3D data format, 3D point clouds can capture much more sophisticated surface geometry of different objects than voxel grids and are suitable for large-scale rendering [13]. However, the surface-centric nature of point clouds also poses challenges to scanning the objects, such as the issues of occlusion and distance. Therefore, high-fidelity 3D point cloud generation is of great importance for a variety of 3D applications. For example, the generated large-scale 3D data can be used in the learning of different 3D tasks, such as segmentation [29, 21], volumetric shape estimation [44], object detection [28], and scene understanding [26].

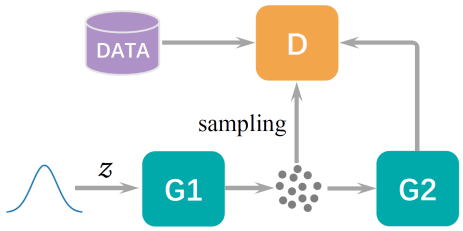

Figure 1. The proposed dual-generators framework for 3D point cloud generation. The overall pipeline mainly consists of two generators accompanying with a shared discriminator to generate point clouds in a progressive manner.

With the great success of deep learning in 2D image data, deep learning-based 3D data generation has attracted increasing attention, and a wide range of generative approaches have been intensively investigated from different perspectives, such as image to point cloud [7, 16], image to voxel [42, 45], image to mesh [39, 41], image to SDF [27], point cloud to voxel [51], and point cloud to point cloud [47, 48, 35, 14]. With a better prior of point clouds, a point cloud generation model can benefit a variety of synthesis tasks such as reconstruction and super-resolution. Recently, several typical deep architectures such as auto-encoders [19] and generative adversarial networks (GANs) [11] have been very successful in learning effective representations and generating realistic samples from complex distributions. For example, GAN-based methods have been explored in transforming random latent codes into 3D point clouds [1, 37]. In this paper, we focus on the learning of point cloud generation models under the generative adversarial learning framework: given a sparse, noisy, and non-uniform latent code, the target is then to generate point clouds that are dense, complete, and uniform, as a faithful representation of the underlying 3D object surface.

To generate high-quality point clouds, we progressively learn effective point embeddings and generate point clouds in a coarse-to-fine manner. Specifically, the idea of pro-

gressive generation/refinement has been widely used in very challenging 2D image and video tasks, such as image generation, object detection, and semantic landmark localization. Motivated by this, we devise a dual-generators framework for point cloud generation. As shown in Fig. 1, a stack of two generators sequentially transform the input latent variables into a suitable 3D representation, such that the discriminator cannot distinguish between the ground truth and the generated point clouds. The overall dual-generators framework for point cloud generation contains two main steps: (1) the first generator generates a dense point cloud to sketch the primitive geometry of the underlying object using an oversampling operation; and (2) the second generator is an encoder-decoder network, which refines the point cloud of the first step to obtain the final high-fidelity point cloud. The main contribution of the proposed method for point cloud generation is threefold:

1. We propose the dual-generators architecture to learn effective point embeddings and generate high-quality point clouds in a progressive manner;

2. The discriminator simultaneously considers two kinds of discriminative cues on point cloud generation: shapewise and pointwise, which facilitates the generator to generate more realistic point clouds;

3. We devise an oversampling conception in regard to point cloud generation. Different from previous methods, the proposed method generates point clouds via a dense-to-sparse process.

Extensive qualitative and quantitative experiments on the ShapeNet dataset demonstrate the effectiveness of the proposed method as well as each individual components, providing valuable insights on how to design effective point cloud generation models.

2. Related Work

Deep learning for 3D point clouds. Deep learning-based point cloud analysis has attracted more and more attention, and the publicly available point cloud datasets, such as ModelNet [44], ShapeNet [3] and ScanNet [6], further boost the research of deep learning on 3D point clouds. Recently, an increasingly number of methods have been proposed to address the problems in point cloud analysis, including 3D point cloud classification and segmentation [29, 22], 3D object detection and tracking [5, 51], point set completion [14, 48]. For example, pointwise MLP networks [29, 30] have been widely used as the building blocks to learn pointwise representations, while convolution-based networks [23, 43, 36] achieve superior performance on irregular 3D point clouds.

Different from images, a point cloud is a set of size-varying and permutation-varying points on irregular domain. When introducing neural network to process point clouds directly, learning-based methods face significant challenges because of the unstructured nature of 3D point clouds. For example, classical convolution and deconvolution operation defined on regular domain can be applied to 1D signal and 2D image directly, but for points scatting in 3D space, it is intricate to define a proper operation to extract high dimension features. Fortunately, there are already several works to explore the convolution operation on points, such as [43], [36], [26], and [23]. Another problem is that efficient evaluation metrics for different point cloud tasks should be proposed. Although some emtrics such as overall accuracy (OA), mean class accuracy (mAcc) and average precision (AP) have been frequently used in 3D shape classification and 3D object detection, for point cloud generation, there is no universal accepted metrics.

In various methods, PointNet [29] is the pioneering work in applying deep neural nets to point sets, which first embeds the inputs into high dimensional feature space pointwisely and then uses a symmetric function to aggregate the features for all points in a permutation-invariant manner. However, relations between the points are not sufficiently captured in this way. Following PointNet, some hierarchical architectures have been developed to aggregate local neighborhood information. For example, PointNet++ [30] introduces a hierarchical structure based on Euclidean-space nearest-neighbor graph, KdNet [21] designs spatial KD-trees for efficient information aggregation, and DGCNN [40] develops a graph neural network (GNN) approach with dynamic graph construction.

Point cloud generation. Substantial progress has been made in point cloud synthesis tasks such as auto-encoding [1, 47], 3D reconstruction [7, 9], and point cloud completion [25, 48, 14]. Further, there are several widely used frameworks for deep generative learning, including generative adversarial networks [11, 2], variational auto-encoders (VAEs) [19], auto-regressive models [38], and flow-based models [46, 31, 20]. For example, [9] and [49] apply variational auto-encoders and adversarial auto-encoders to point cloud generation, respectively. [1] explores generative adversarial networks for point clouds in both raw data space and latent space with a pretrained auto-encoder.

To address the problem of point cloud generation, [7] introduces two symmetric distance metrics, Chamfer distance and Earth Mover's distance, to measure the similarity between two point sets. These metrics are order-invariant, which makes them suitable as loss function operated directly on point clouds. By taking advantage of these metrics, models have been proposed to address point cloud synthesis problems under different settings [9, 24, 8, 16]. Given

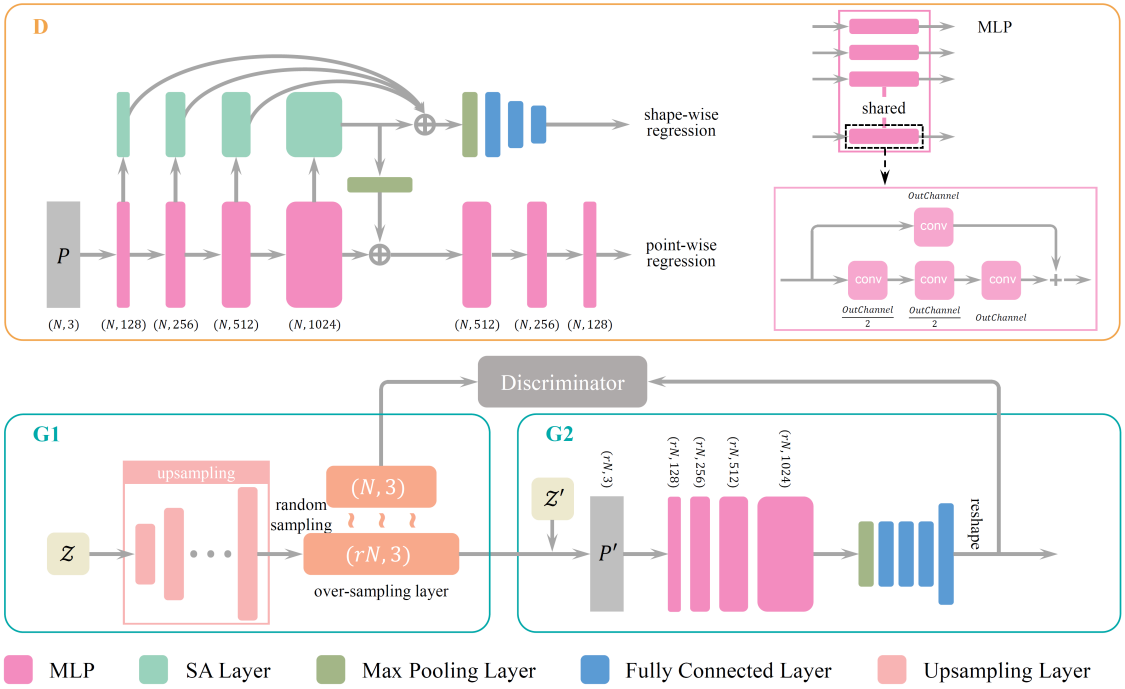

Figure 2. The main dual-generators framework. The proposed method mainly contains three components: one discriminator and two different generators. The discriminator is shared by the two generators and the output of the first generator is fed into the second generator to produce the final results. Consequently, the overall training method is a progressive process, i.e., we first train the discriminator and the first generator, and then jointly train all three components in an end-to-end manner.

the difficulties in modeling details of the generated samples, several recent works, such as StackGAN [50] (for 2D data), progressively growing GAN [17] (for 2D data) and PDGN [15] (for 3D data), have been developed to improve the generation quality, in which the difficult problem of generating high-fidelity samples is divided into multiple steps and significantly improves the performance of data generation. Motivated by this, we uses a stack of two generators in the proposed 3D point cloud generation method, accompanying with single discriminator, to generate robust and accurate 3D point clouds. Also named progressive generation, our work is different from PDGN [15] in two aspects. The two generators in our network are supervised by single discriminator, reducing the network parameters. What's more, our network produces high quality results in a dense-to-sparse fashion, instead of the multi-step strategy in PDGN [15].

3. Method

We introduce the proposed dual-generators point cloud generation method as follows. Given a vector sampled from the latent space, $z \sim \boldsymbol{p_z}$, we aim to generate a point cloud or a set of 3D points, $P = \{p_i\}_i^N$, where N is the number of points. Specifically, the generated points should fulfill the following two requirements: 1) the generated points indicate the underlying geometry structure of the target object, i.e., all points in P should lie on and cover the surface of the target object; and 2) the generated points in P should be uniformly-distributed on the surface of the target object. We demonstrate the main point cloud generation framework in Fig. 2, where the proposed method contains three main components (a shared discriminator and two generators) and is optimized using an end-to-end training strategy.

3.1. Discriminator

In adversarial learning, the discriminator distinguishes whether the input (e.g., a point cloud) is produced by the generator or sampled from the ground truth, while the generator requires to fool the discriminator by generating a high-fidelity point cloud. Recently, [1] has investigated GAN-based architectures for point cloud generation: the discriminator first embeds the input points into high dimension in a pointwise manner, and then aggregates the feature vectors of previous step to obtain the global features, e.g., using the symmetric operation [29]. However, the above architecture usually fails to capture the local information implied in geometry structures. To overcome this, we simultaneously extract global and local point features to learn effective point representations.

As shown in Fig.2, to extract effective feature representations from point clouds, we utilize multiple MLP blocks with residual connections to learn both low- and high-level features in a pointwise manner. The input of each pointwise

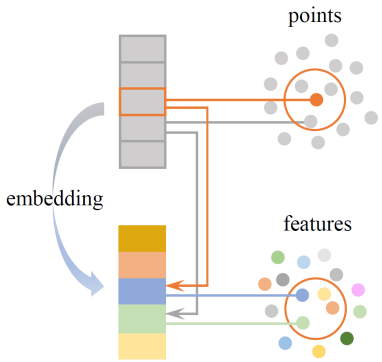

Figure 3. The extraction of local point features. Specifically, points are first sampled (the orange point) and then grouped (the gray points in the circle) together in 3D space. Point features then can easily be retrieved through the grouped point.

MLP block is a set of point features with the size $N \times C_{in}$ and the output size $N \times C_{out}$, where C_{in} and C_{out} denote the numbers of channels in the input and output feature attributes, respectively. After each MLP block, we also attach a point convolution layer or the set abstraction (SA) layer [30] to extract local features by aggregating the information of neighbor points. Specifically, each point convolution layer consists of a sampling operation and a grouping operation, as shown in Fig. 3. The input size of this layer is $N \times C'_{in}$ and the output size is $N_s \times C'_{out}$, where N_s indicates the number of sampled points.

Unlike 2D image generation, 3D point cloud generation involves a more immense spatial domain. Considering the nature of point clouds, we utilize two different kinds of supervision, shapewise and pointwise, to enhance the training process of the discriminator. Specifically, for the shapewise regression, we concatenate the output of each SA layer to obtain hierarchical features with the size $N_s \times \sum_i C_i$. We then apply the max pooling operation to aggregate the information from all points to obtain the final permutation-invariant features for shapewise regression. For the pointwise regression, it is used to synthesize globally and locally coherent point clouds. We first employ the max pooling operation after the last SA layer. We then concatenate N copies of the output with the per-point features from the last residual block as the enhanced pointwise features with the size $N \times (C_{rb} + C_{sg})$, where C_{rb} and C_{sg} indicate the output channels of the last residual block and the last SA layer, respectively. We then embed the per-point features to a high dimensional space for pointwise regression. Therefore, the discriminator not only maintains the global geometry coherence of synthesized point clouds but also provides detailed per-point representation to the generator, thus contributing to produce high-fidelity point clouds. Lastly, the discriminator can be jointly optimized by the shapewise regression loss L_s and the pointwise regression loss L_p as follows:

$$L_{dis} = L_s(\boldsymbol{p}, \tilde{\boldsymbol{p}}) + L_p(\mathbb{E}[\boldsymbol{p}_i], \mathbb{E}[\tilde{\boldsymbol{p}}_i]), \tag{1}$$

where $\boldsymbol{p}$ and $\tilde{\boldsymbol{p}}$ indicate the real samples and the generated samples, respectively, $\boldsymbol{p}_i$ is the i-th point of the point cloud $\boldsymbol{p}$, $\mathbb{E}[\boldsymbol{p}_i]$ indicates the mean decision of discriminator over all points, and $\mathbb{E}[\tilde{\boldsymbol{p}}_i]$ can be defined in a similar way. For $L_s(\cdot)$ and $L_p(\cdot)$, we use the similar loss function $L(\cdot)$ defined in [2, 12],

$$\begin{aligned} L(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = &\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim P_g}[\mathcal{D}(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_r}[\mathcal{D}(\boldsymbol{x})] \\ &+ \lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim P_{\hat{x}}}[(\|\nabla_{\hat{\boldsymbol{x}}} \mathcal{D}(\hat{\boldsymbol{x}})\|_2 - 1)^2], \end{aligned} \tag{2}$$

where P_r indicates the data distribution, P_g indicates the generator distribution, $P_{\hat{x}}$ is defined as the uniform sampling along the straight lines between pairs of points from P_r and P_g, and $\mathcal{D}$ is the set of 1-Lipschitz functions.

3.2. Dual-Generators

Progressive image generation pipelines have been intensively explored to generate high-resolution images [50, 17]. Inspired by this, we devise the dual-generators method to generate high-quality point clouds as follows: the first generator learns a dense point cloud to sketch the underlying structure of the object, while the second generator downsamples and refines the dense point cloud generated by the first generator.

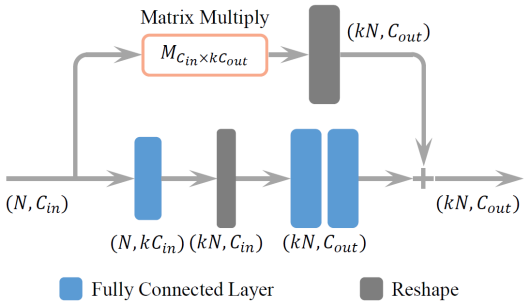

Figure 4. The detailed structure of the upsampling layer used in our method. The input features are embedded by two branch, i.e., the fully-connected branch and the matrix multiplication branch. The matrix multiplication branch aims to learn an affine transformation matrix to align the input features. k indicates the upsampling rate.

As shown in Fig. 2, in the first generator, the latent vector z is transformed by several upsampling layers with upsampling rate k. We demonstrate the structure of the upsampling layer in Fig. 4. At the end of the first generator, there is an upsampling layer to generate a dense point cloud with rN points. We refer to this upsampling layer as the oversampling layer since our target is to generate a set of N points. After that, we randomly sample N points from the rN dense point cloud, and then feed it into the discriminator. Intuitively, the above random sample operation performs like a dropout layer [34]. Here, we adopt the idea of dropout for point cloud generation, and the motivation is

that if the downsampled point cloud is deemed real by the discriminator, the dense point cloud is also real. This simple yet effective step makes the generation process more robust and steady. The loss function of the first generator can be defined as

$$L_{gen1} = G_s(\tilde{\boldsymbol{p}}) + G_p(\mathbb{E}[\tilde{\boldsymbol{p}}_i]), \tag{3}$$

where $G_s(\cdot)$ and $G_p(\cdot)$ share the same form with $G(\cdot)$, i.e.,

$$G(\tilde{\boldsymbol{x}}) = -\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim P_{g1}}[\mathcal{D}(\tilde{\boldsymbol{x}})], \tag{4}$$

where P_{g1} indicates the distribution of the first generator.

For the second generator, we embed the rN input points into a high dimensional space using a way similar to the discriminator. The loss function of the second generator, L_{gen2}, keeps the same form with the first generator. In addition, we introduce the reconstruction loss L_{rec} as a regularizer to enhance the quality of generated point clouds. The reconstruction loss is then defined by the Chamfer distance [7] as follows:

$$L_{rec} = \sum_{p \in P} \min_{q \in Q} \|p - q\|_2^2 + \sum_{q \in Q} \min_{p \in P} \|p - q\|_2^2, \tag{5}$$

where $P, Q \in \mathbb{R}^3$ indicate the source and target point sets, respectively. The overall loss function for the second generator can be defined as:

$$\hat{L}_{gen2} = \lambda_1 L_{rec} + \lambda_2 L_{gen2}. \tag{6}$$

Furthermore, the second generator also can be explained as a filter or a denoising auto-encoder, for it is natural to regard the input as a point cloud with noise and outliers.

4. Experiments

In this section, we first introduce the implementation details of the proposed method, including datasets and hyperparameters. We then compare the proposed method with recent state-of-the-art point cloud generation methods. Lastly, we perform ablation studies on several important components.

4.1. Implementation Details

Point clouds are usually obtained by uniformly sampling from mesh objects, e.g., ShapeNet [3]. For fair comparison, we use three widely-used object categories, chair, car and airplane, in our experiments. Following the evaluate metrics used in previous works [1, 46], we use 2,048 points for each category in both training and testing stages.

The discriminator takes a point cloud containing 2,048 points as the input, which is then embedded into high dimension pointwisely by four MLP and SA layers. We demonstrate the configurations of each layer in Fig. 2. Specifically, the max pooling is used in the network to reduce the impact of different point permutations. Note that, we use WGAN [2] as our basic framework, and thus there is no batch normalization layer in the discriminator. The first generator transforms the latent vector to a fixed-size feature using five upsampling layers whose upsampling rates are (1, 2, 8, 16, 8) and following the oversampling layer embed these features to size $rN \times 3$. If not mentioned, the oversampling layer in our experiment is another upsampling layer and we set $r = k = 4$ in our experiments. The second generator mainly refine the result from the first step. For the backbone network in the second generator, we adopt four MLP blocks as in the discriminator, shown in Fig. 2. We implement the proposed method for point cloud generation using TensorFlow. All our models are trained on one NVIDIA GeForce RTX 2080Ti GPU. The training process consists of two steps: 1) we first train the discriminator and the first generator; and then 2) we simultaneously train all three components in our network. We use the Adam optimizer with $\beta = 0.5$ and the initial learning rate is 0.0001.

4.2. Comparison with Recent State-of-the-Arts

In this subsection, we compare the proposed method with recent state-of-the-art point cloud generation methods. We demonstrate the point cloud generation performance using different object categories and provide quantitative comparisons with the following six methods, raw-GAN [1], latent-GAN [1], GraphCNN-GAN [37], PointFlow [46], tree-GAN [33], and PDGN [15], using their official implementations with default training parameters. For fair comparison, we adopt five popular evaluation metrics used by Achlioptas et al. [1], (i.e., JSD, MMD-CD, MMD-EMD, COV-CD, and COV-EMD) and 1-NNA recently proposed by [46]. The experimental results on three different categories, chair, car and airplane, are shown in Table 1. Specifically, our model outperforms raw-GAN across all three categories with a large margin and obtains either comparable or the best score under all evaluation metrics.

We also demonstrate some samples generated by our model for all three categories in Fig. 5, where our proposed method are able to generate uniformly distributed and diverse typologies of point clouds.

4.3. Ablation Studies

To better understand the proposed method, we perform ablation studies on several important components used in our method, including the architecture of the discriminator, the oversampling refinement operation, and the dual-generators structure.

Discriminator Architecture. For the discriminator in the raw-GAN [1], it only uses MLP blocks to embed the input points into high dimensional space and then utilize the max pooling operation to extract global features. Differ-

Class	Model	JSD(↓)	MMD(↓)		COV(↑, %)		1-NNA(↓, %)	
			CD	EM	CD	EM	CD	EM
Chair	raw-GAN	0.1266	0.2417	11.12	46.25	32.38	73.20	97.36
	latent-GAN	0.0728	0.2397	8.72	50.49	34.21	61.52	84.31
	GraphCNN-GAN	0.1542	0.3584	11.98	23.30	13.58	-	-
	PointFlow	0.0957	0.8096	11.92	46.15	43.14	62.58	63.24
	tree-GAN	0.1438	1.2690	10.78	37.35	33.50	-	-
	PDGN	0.0827	0.2069	**7.51**	**53.09**	47.57	52.38	57.14
	Ours	**0.0716**	**0.2036**	8.04	51.30	**49.22**	**49.90**	**55.31**
Car	raw-GAN	0.0820	0.0907	7.42	35.18	19.43	93.68	96.98
	latent-GAN	0.0632	0.0781	6.31	40.49	21.72	65.64	85.10
	GraphCNN-GAN	0.0787	0.1235	7.46	15.27	7.36	-	-
	PointFlow	0.0971	0.2203	6.35	20.33	20.15	61.53	61.22
	tree-GAN	0.1136	0.1555	11.18	25.50	18.64	-	-
	PDGN	0.0607	0.0721	6.03	**41.33**	**31.82**	53.34	56.06
	Ours	**0.0531**	**0.0713**	**5.91**	39.21	30.59	**52.10**	**55.12**
Airplane	raw-GAN	0.1047	0.0785	7.54	35.80	16.44	90.46	96.71
	latent-GAN	0.0841	0.0690	7.22	36.41	30.02	85.32	93.25
	GraphCNN-GAN	0.1952	0.1124	8.59	18.38	7.85	-	-
	PointFlow	0.0946	0.1294	7.44	30.22	28.10	77.39	76.16
	tree-GAN	0.1423	0.1660	10.50	15.73	17.89	-	-
	PDGN	0.0732	0.1211	**6.91**	40.54	38.34	63.15	60.52
	Ours	**0.0719**	**0.0646**	7.16	**42.03**	**39.32**	**61.10**	**60.34**

Table 1. Quantitative comparisons with recent state-of-the-art point cloud generation methods. ↑: the higher the better, ↓: the lower the better. The best scores are highlighted in bold. MMD-CD and MMD-EMD scores are multiplied by 10^3 and 10^2 respectively.

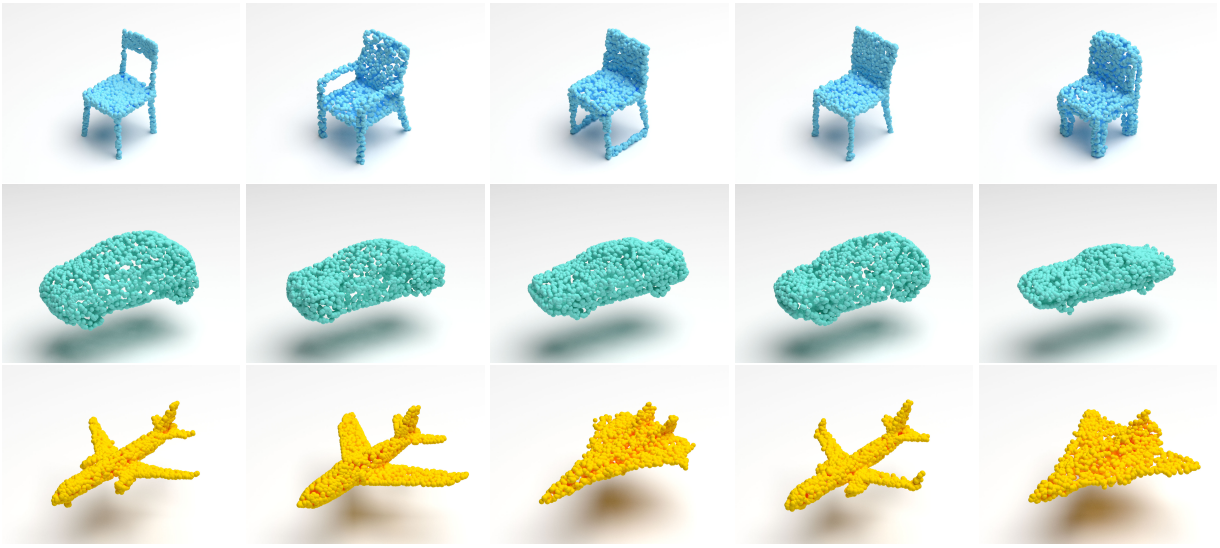

Figure 5. Examples of point cloud generated by our model. From top to bottom: chair, car and airplane.

ent from that, our method further attaches a “SA" layer after each residual MLP block, and concatenate their outputs as the global features. The structure of “MLP + SA" then works as the basic component of feature extraction process for our discriminator. Therefore, we compare the original raw-GAN and the modified raw-GAN (built on the “MLP + SA" structure) in Fig. 6. It is clear that the results of raw-GAN tend to be clustering at some parts of the output, resulting in an uneven distribution, while the modified one can generate more uniform point clouds. An intuitive explanation might be: when embedding the input point cloud into high dimensional space pointwisely, the raw-GAN fails

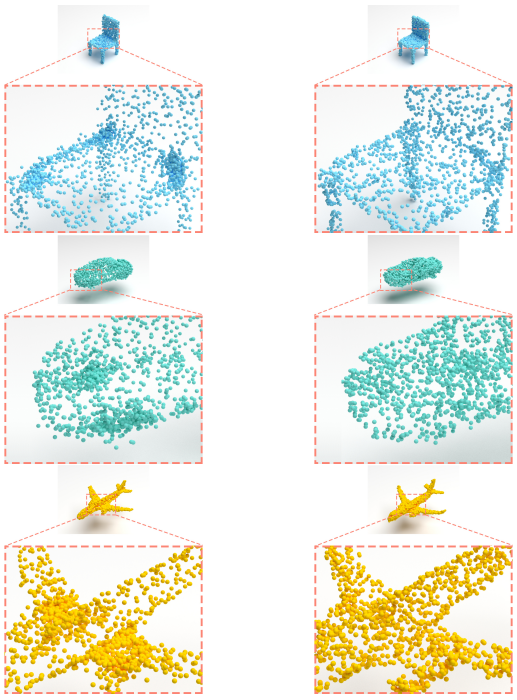

Figure 6. Comparisons between the original raw-GAN (left) and the modified raw-GAN (right). It is clearly shown that there are always some clusters in the raw-GAN's output, while the modified raw-GAN can generate more uniformly distributed point clouds because of local feature extraction.

to extract the local feature information. With the global feature aggregated by the max pooling layer, the results can only represent the underlying object, but it does not imply any distribution of the output points. Different from that, our discriminator is designed to extract global and local features simultaneously and thus facilitates the discriminator to help the generator to produce uniformly distributed point clouds. The same phenomenon also appears in GraphCNN-GAN [37] and tree-GAN [33] as shown in Fig. 7, in which both of them use similar discriminator architectures.

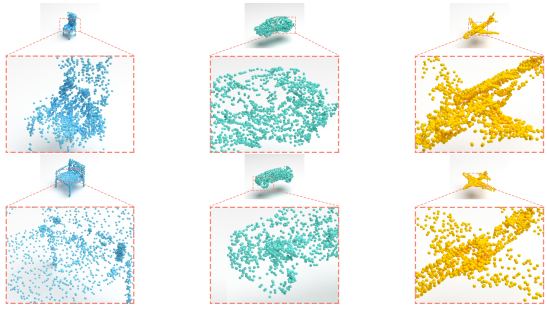

Figure 7. The clustering points also occur in GraphCNN-GAN (top) and tree-GAN(bottom). They both use the similar discriminators as raw-GAN does, and thus fail to extract local feature information which can be used for uniform point cloud generation.

Furthermore, in our discriminator, the network simultaneously gives both shape and point decision of the output belonging to either the real or the fake class. This architecture empowers a stronger discriminator and makes it more challenging for the generator's to fool the discriminator, thus improving the quality of generated samples. It contributes to maintain a more powerful data representation and encourages to produce global and local coherent results. In Fig. 8, we visualize the per-point confidence during training phase and in Fig. 9, the pointwise feedback can be used to guide the generator to refine the per-point generation.

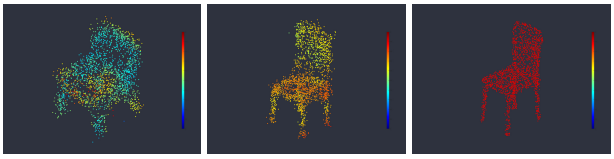

Figure 8. With training, the generator output high per-point confidence results gradually, best viewed in color. Low confidence points are in blue and high ones in red.

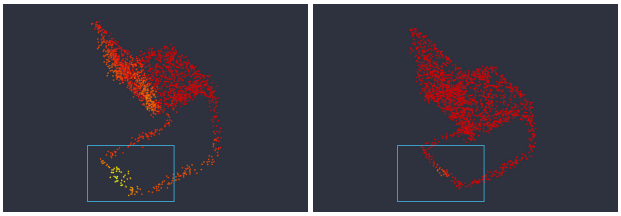

Figure 9. The discriminator gives unrealistic points low confidence. With pointwise feedback, the generator can output more detail confident results.

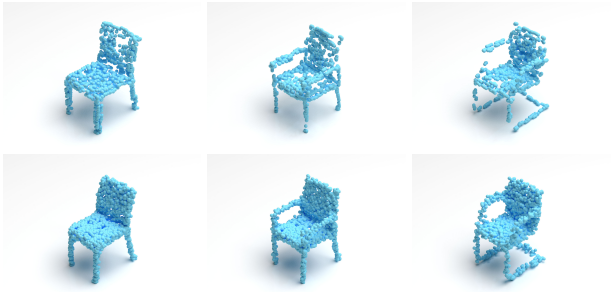

Figure 10. Comparison between the outputs (2048 points) of first generator (top) and second generator (bottom). It is clearly shown that the second generator produce more uniformly distributed point clouds than the first generator.

Dual-Refinement. Different from images, point clouds always scatter in 3D space irregularly and the output of the generator is inevitable with noise and outliers. To further enhance the output, we apply the two-generator structure to refine the generated point clouds. In the first generator, the over-sampling layer works in a similar way to the dropout operation, which mainly contributes to robust point cloud generation and network training. The second generator performs like the auto-encoder that removes the noise and out-

Class	Model	JSD(↓)	MMD(↓)		COV(↑, %)		1-NNA(↓, %)	
			CD	EM	CD	EM	CD	EM
Chair	Gen_One	0.0812	0.2903	13.85	46.21	43.64	53.78	58.20
	Gen_One_OS	0.0776	0.2558	11.05	46.49	44.51	52.91	56.19
	Gen_Two	**0.0716**	**0.2169**	**9.34**	**48.30**	**46.22**	**49.90**	**55.31**
Car	Gen_One	0.0621	0.0801	8.43	35.72	28.37	54.06	60.61
	Gen_One_OS	0.0592	0.0752	7.20	37.16	28.86	53.77	58.32
	Gen_Two	**0.0531**	**0.0713**	**5.91**	**38.21**	**30.59**	**52.10**	**55.12**
Airplane	Gen_One	0.0845	0.0764	8.63	38.41	27.91	64.38	63.78
	Gen_One_OS	0.0817	0.0691	7.97	40.64	29.26	62.73	63.55
	Gen_Two	**0.0719**	**0.0646**	**7.16**	**42.03**	**30.32**	**61.10**	**60.34**

Table 2. Quantitative comparison of three generators. The evaluation metrics are the same as that in Table 1. It is shown that each operation in our dual-generator structure plays its role in the whole pipeline.

Model	MN10(%)	MN40(%)
SPH [18]	79.8	68.2
LFD [4]	79.9	75.5
T-L Network [10]	-	74.4
VConv-DAE [32]	80.5	75.5
3D-GAN [42]	91.0	83.3
MRTNet [9]	91.7	86.4
PointFlow [46]	93.7	86.8
PDGN [15]	**94.2**	87.3
Ours	94.1	**87.6**

Table 3. Classification results on ModelNet10 (MN10) and ModelNet40 (MN40). Models are first trained on ShapeNet to learn point cloud representations, and then evaluated on MN10 and MN40 by object classification accuracy using linear SVM classifier.

liers from the input to refine the final results. In Table 2, we quantitatively compare the outputs (2048 points) of first generator without oversampling layer (Gen_One), first generator with oversampling layer (Gen_One_OS) and second generator using evaluation metrics mentioned before. From the table, we can conclude that each component in our dual-generator structure works closely with the other two and progressively contributes to producing high quality point clouds. In Fig. 10, we show the visual results of the first generator and the second generator. Because the second generator learns to refine the outputs from the first generator, its output are more uniform than the previous step.

4.4. Other Experiments

Similar to [42], we evaluate the unsupervised representation learning ability of our proposed method. A general way to evaluate the learned representations is to conduct the 3D object classification experiments. Following the common practice, we train our network on ShapeNet [3] and test it on two popular subsets of ModelNet [44], ModelNet10 and ModelNet40. Specifically, we first feed our network with the full ShapeNet dataset and then extract the embedded features of the trained discriminator to learn a linear SVM for classification on ModelNet10 and ModelNet40. We compare our network with recent state-of-the-art point cloud generation methods and demonstrate per-class classification results in Table 3. It can be seen that our method achieves close performance with PDGN [15] and outperforms most other generation methods on both ModelNet10 and ModelNet40 datasets. The results demonstrate that our discriminator architecture can extract discriminative features and thus supervise the generators to produce high-quality 3D point clouds.

5. Conclusion and Future Work

In this paper, we devise a new point cloud generation framework with three main components: a shared discriminator and two generators working in a progressive manner. To generate realistic and robust point clouds, we embed the points in two different fashions. Extensive experiments show that the proposed method enables the generative adversarial learning pipeline to produce high-fidelity point sets, and outperforms most recent point cloud generation methods under various evaluation metrics. In addition, most existing evaluation metrics for point cloud generation have inherent limitations, which will also be the subject of feature study.

Acknowledgements

This work was supported in part by Australian Research Council Projects FL-170100117, DP-180103424, IH-180100002, and IC-190100031.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International Conference on Machine Learning (ICML)*, pages 40–49. PMLR, 2018. 1, 2, 3, 5

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning (ICML)*, pages 214–223. PMLR, 2017. 2, 4, 5

[3] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, et al. Shapenet: An information-rich 3d model repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University, Princeton University, Toyota Technological Institute at Chicago, 2015. 2, 5, 8

[4] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3d model retrieval. In *Computer Graphics Forum*, volume 22, pages 223–232. Wiley Online Library, 2003. 8

[5] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1907–1915, 2017. 2

[6] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5828–5839, 2017. 2

[7] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 605–613, 2017. 1, 2, 5

[8] Matheus Gadelha, Subhransu Maji, and Rui Wang. Shape generation using spatially partitioned point clouds. In *British Machine Vision Conference 2017*, 2017. 2

[9] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution tree networks for 3d point cloud processing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 103–118, 2018. 2, 8

[10] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 484–499, 2016. 8

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2672–2680, 2014. 1, 2

[12] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5767–5777, 2017. 4

[13] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 1

[14] Zitian Huang, Yikuan Yu, Jiawen Xu, Feng Ni, and Xinyi Le. Pf-net: Point fractal network for 3d point cloud completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7662–7670, 2020. 1, 2

[15] Le Hui, Rui Xu, Jin Xie, Jianjun Qian, and Jian Yang. Progressive point cloud deconvolution generation network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 397–413, 2020. 3, 5, 8

[16] Li Jiang, Shaoshuai Shi, Xiaojuan Qi, and Jiaya Jia. Gal: Geometric adversarial loss for single-view 3d-object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 802–816, 2018. 1, 2

[17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018. 3, 4

[18] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3 d shape descriptors. In *Symposium on Geometry Processing*, volume 6, pages 156–164, 2003. 8

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014. 1, 2

[20] Roman Klokov, Edmond Boyer, and Jakob Verbeek. Discrete point flow networks for efficient point cloud generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 694–710, 2020. 2

[21] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 863–872, 2017. 1, 2

[22] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9397–9406, 2018. 2

[23] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 820–830, 2018. 2

[24] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 1, 2

[25] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. Morphing and sampling network for dense point cloud completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2020. 2

[26] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1578–1587, 2019. 1, 2

[27] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. 1

[28] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 9277–9286, 2019. 1

[29] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 1, 2, 3

[30] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5099–5108, 2017. 2, 4

[31] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, pages 1530–1538. PMLR, 2015. 2

[32] Abhishek Sharma, Oliver Grau, and Mario Fritz. Vconv-dae: Deep volumetric shape learning without object labels. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 236–250, 2016. 8

[33] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3859–3868, 2019. 5, 7

[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 4

[35] Lyne P Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. Topnet: Structural point cloud decoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 383–392, 2019. 1

[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. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 6411–6420, 2019. 2

[37] Diego Valsesia, Giulia Fracastoro, and Enrico Magli. Learning localized generative models for 3d point clouds via graph convolution. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 5, 7

[38] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning (ICML)*, pages 1747–1756. PMLR, 2016. 2

[39] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018. 1

[40] 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)*, 38(5):1–12, 2019. 2

[41] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2mesh++: Multi-view 3d mesh generation via deformation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1042–1051, 2019. 1

[42] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 82–90, 2016. 1, 8

[43] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9621–9630, 2019. 2

[44] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 1, 2, 8

[45] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2690–2698, 2019. 1

[46] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4541–4550, 2019. 2, 5, 8

[47] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 206–215, 2018. 1, 2

[48] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *International Conference on 3D Vision (3DV)*, pages 728–737. IEEE, 2018. 1, 2

[49] Maciej Zamorski, Maciej Zięba, Piotr Klukowski, Rafał Nowak, Karol Kurach, Wojciech Stokowiec, and Tomasz Trzciński. Adversarial autoencoders for compact representations of 3d point clouds. *Computer Vision and Image Understanding*, 193:102921, 2020. 2

[50] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5907–5915, 2017. 3, 4

[51] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4490–4499, 2018. 1, 2