CPCGAN: A Controllable 3D Point Cloud Generative Adversarial Network with Semantic Label Generating

Ximing Yang[1], Yuan Wu[1,2], Kaiyi Zhang[1] and Cheng Jin[1,2]
[1]School of Computer Science, Fudan University, Shanghai, China
[2]Peng Cheng Laboratory, Shenzhen, China
{xmyang19,wuyuan,zhangky20,jc}@fudan.edu.cn

Abstract

Generative Adversarial Networks (GAN) are good at generating variant samples of complex data distributions. Generating a sample with certain properties is one of the major tasks in the real-world application of GANs. In this paper, we propose a novel generative adversarial network to generate 3D point clouds from random latent codes, named Controllable Point Cloud Generative Adversarial Network(CPCGAN). A two-stage GAN framework is utilized in CPCGAN and a sparse point cloud containing major structural information is extracted as the middle-level information between the two stages. With their help, CPCGAN has the ability to control the generated structure and generate 3D point clouds with semantic labels for points. Experimental results demonstrate that the proposed CPCGAN outperforms state-of-the-art point cloud GANs.

Introduction

Generative Adversarial Networks (GANs) have attracted significant research interest because of their successes in many applications. Those applications on 2D images include single image super-resolution(Dong et al. 2017; Wang et al. 2018), interactive image generation(Rott Shaham, Dekel, and Michaeli 2019), image editing(Perarnau et al. 2016), and image-to-image transformation(Karras, Laine, and Aila 2019; Rott Shaham, Dekel, and Michaeli 2019). Captured 3D information has been garnering attention over the past few years, and the requirements of expanding or generating 3D data is also growing. Various approaches have been proposed on 3D data generation. Such as point cloud upsampling(Li et al. 2019; Yu et al. 2018a,b), point cloud completion(Yang et al. 2018; Yuan et al. 2018), and 3D data form transformation(Dai et al. 2017; Yang et al. 2018; Zhou and Tuzel 2018). To use GAN in real-world applications, one of GANs' fundamental targets is to construct and learn a generative model that can be controlled to generate samples with a certain distribution. Controllable GANs using random latent codes as input have been well researched on 2D images(Chen et al. 2016; Karras, Laine, and Aila 2019; Mirza and Osindero 2014; Wang and Gupta 2016).

Owing to the sparse representation of point clouds and the lack of computational power, GANs for generating 3D point clouds from random latent codes have been rarely researched until recent years. (Achlioptas et al. 2017) implements 3D point cloud GAN using simple fully connected layers. (Valsesia, Fracastoro, and Magli 2018) uses dynamic graph convolutions network(GCN) to deal with local point cloud features. (Shu, Park, and Kwon 2019) uses tree-structured GCN, reducing the computational cost to some extent. (Hui et al. 2020) proposes a learning-based bilateral interpolation method to generate more precise point clouds. Validation metrics aiming to measure the point cloud generation performances are proposed in (Achlioptas et al. 2017; Shu, Park, and Kwon 2019). Although the methods are getting better on those metrics, the controllability of generation has not been well concerned. Besides, those works(Shu, Park, and Kwon 2019; Valsesia, Fracastoro, and Magli 2018) with GCN are somewhat inefficient due to the complexity of their network structures.

In this paper, we present a novel method called Controllable Point Cloud Generative Adversarial Network (CPCGAN), which can generate 3D point clouds from random latent codes. CPCGAN can also generate semantic labels for points and it is controllable for generated shape. Our approach takes the inspiration of S^2-GAN(Wang and Gupta 2016) and constructs a two-stage GAN, which suits well on the 3D point cloud generation task. The first-stage GAN takes a random latent code as input and generates a sparse point cloud with semantic labels, which we call structure point cloud. The second-stage GAN takes the output of first-stage as input and generates the complete point cloud by breeding a certain number of points from every structure point. The semantic label of the structure point will be inherited by the bred points. Some simple fully connected layers and a self-attention layer are used to construct the generators.

The main contributions of this paper are:

1. A novel CPCGAN method is proposed, which achieves state-of-the-art performance on both generating results and computational effectiveness.
2. A two-stage GAN framework is constructed to make the CPCGAN able to control the generated shapes.
3. A concept of structure point clouds as well as its extraction method are introduced to help the CPCGAN generate not only point clouds but also semantic labels.

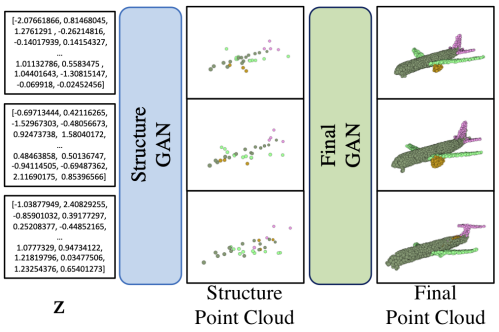

Figure 1: Overview of CPCGAN. Two stages are constructed in CPCGAN. A sparse point cloud containing major structural information, which we call structure point cloud, is introduced as the middle-level representation between two stages. Semantic information is generated by Structure GAN. The controllability is achieved by modifying the structure point cloud.

4. A series of metrics are proposed to evaluate the performance of the semantic labels generation.

Related Work

3D Point Clouds Feature Extracting

In 3D point cloud GAN, a comparison between real and fake point clouds is indispensable. It is hard to compare point clouds using the raw data. Therefore we need a comparable representation of point clouds. 3D point cloud features extracted by deep neural networks are one of the most popular choices. The extraction methods have been wildly researched, and are already used on discriminators of 3D point cloud GANs.

Over the past few years, plenty of works focus on 3D point cloud feature extraction with the purpose of point cloud classification and semantic segmentation(Atzmon, Maron, and Lipman 2018; Li et al. 2018; Mao, Wang, and Li 2019; Qi et al. 2017a,b; Su et al. 2018; Wang et al. 2019; Wu, Qi, and Fuxin 2019; Zhang, Hua, and Yeung 2019). Qi et al.(Qi et al. 2017a) proposed PointNet, which uses multi-layer perceptrons with shared weights and point-wise max-pooling layer to avoid the disorderliness of point cloud. Because the max-pooling layer is applied across all the points, it is difficult to capture local structures. There are many other works improving PointNet through various approaches. Some use grid-based methods like SPLATNet(Su et al. 2018), InterpCNN(Mao, Wang, and Li 2019), etc. and others adopt continuous methods on 3D space to extract local features, like PointCNN(Li et al. 2018), PointConv(Wu, Qi, and Fuxin 2019), ShellNet(Zhang, Hua, and Yeung 2019), etc.

In this paper, we adopt wildly used PointNet based models, which are simple and yet efficient, as the discriminators.

GANs for 3D Point Clouds Generation

GANs have achieved great success on 2D image generation tasks(Isola et al. 2017; Lin et al. 2018; Radford, Metz, and Chintala 2015; Rott Shaham, Dekel, and Michaeli 2019), but are rarely studied in the 3D point cloud generation field. Achlioptas et al.(Achlioptas et al. 2017) presented a methodology to apply autoencoder models on point cloud learning, and introduced the r-GAN, which is based on multiple fully connected layers. As structure information cannot be embedded into fully connected layers, the r-GAN has difficulty in generating diverse shapes. Valsesia et al.(Valsesia, Fracastoro, and Magli 2018) used multi-layer dynamic graph convolutions network(GCN) as a generator in GAN. Adjacency matrixes of the graphs are dynamically constructed, which causes the weakness of time complexity. Hui et al.(Hui et al. 2020) introduced a learning-based bilateral interpolation method to upsample point clouds, and proposed a new network with several discriminators in different scales. Shu et al.(Shu, Park, and Kwon 2019) proposed tree-GAN, which uses tree-structured graph convolutions to obtain a hierarchical structure in feature space. Compared to the method in (Valsesia, Fracastoro, and Magli 2018), tree-GAN requires no prior knowledge regarding the connectivity of the graph, which is more computationally efficient. Tree-GAN also claimed that it can generate point clouds for different semantic parts without prior knowledge, but no semantic label will be generated for points and the accuracy of semantic part generation is not so satisfactory.

Furthermore, the controllability of generation hasn't been considered in the above works. CPCGAN uses semantic data from the training dataset and information extracted by a K-means clustering algorithm to achieve the controllability of the generated structure. Besides, our method is able to generate semantic labels for points.

GANs for Controllable Generation

Controllable GANs have been studied deeply in 2D image generations. Conditional GAN(CGAN)(Mirza and Osindero 2014) added a one-hot vector to embed classification information into GAN network, thus has the ability to generate samples in certain classes. Chen et al.(Chen et al. 2016) improved CGAN to infoGAN, in which property information is embedded into latent code. Besides, infoGAN proposed a regularization term using variational mutual information, which makes the model learn a controllable feature explanation. StyleGAN(Karras, Laine, and Aila 2019) has the ability to control high-level attributes in generation with the help of new architecture and a latent space mapping function.

Wang et al.(Wang and Gupta 2016) used another way to control the generation. A two-stage GAN was presented by (Wang and Gupta 2016). The first stage generates a middle-level representation of the final result, which is a surface normal map of a scene, and the second stage generates an image with the guidance of the middle-level representation. The controllability is ensured by changing the middle-level representation.

Hierarchical structures of semantic information are maintained more completely in constructed 3D point clouds than in 2D images. Therefore, useful information like structures and semantics can be embedded into an appropriate middle-level representation more precisely than images. We take the inspiration of (Wang and Gupta 2016) and construct a

two-stage GAN, which are named Structure GAN and Final GAN. With the help of the structure point clouds extracted by a K-means clustering algorithm, CPCGAN has the ability to control the generated structure.

Proposed Method

Overview

Fig. 1 shows the overview structure of our method. Given a random latent code $z \in \mathbb{R}^{96} \sim N(0, I)$, we aim to generate a complete point cloud $\mathcal{CP}_{\mathcal{GE}} = \{g_i\}_{i=1}^{2048}$, where $g_i \in \mathbb{R}^3$. Corresponding semantic labels $\mathcal{CS}_{\mathcal{GE}} = \{s_i\}_{i=1}^{2048}$ are also generated, where $s_i \in [1, c] \cap \mathbb{Z}$ and c denotes the total number of semantic types. To generate $\mathcal{CP}_{\mathcal{GE}}$ and $\mathcal{CS}_{\mathcal{GE}}$, we construct a two-stage GAN network called CPCGAN. The structure point cloud $\mathcal{SP}_{\mathcal{GT}} = \{sp_i\}_{i=1}^{32}$, and its semantic labels $\mathcal{SS}_{\mathcal{GT}} = \{ss_i\}_{i=1}^{32}$ are introduced as the middle-level representation between two stages. The first stage, which we call Structure GAN, is aiming to learn the distribution of structure point clouds $\mathcal{SP}_{\mathcal{GT}}$ and $\mathcal{SS}_{\mathcal{GT}}$. The outputs of this stage are a generated structure point cloud $\mathcal{SP}_{\mathcal{GE}} = \{gsp_i\}_{i=1}^{32}$ and semantic labels $\mathcal{SS}_{\mathcal{GE}} = \{gss_i\}_{i=1}^{32}$. The distribution of the complete point clouds $\mathcal{CP}_{\mathcal{GT}} = \{cp_i\}_{i=1}^{2048}$ is learned by the second-stage GAN, which we call Final GAN. Fig. 2 shows the structure of CPCGAN.

In the training period, CPCGAN is optimized by a combined loss function. Besides, a stage-wise training strategy is designed in order to provide a more stable training process. A controllable generation method will also be introduced in this section.

Data Preprocessing

We use the ShapeNet-Partseg(Chang et al. 2015) dataset for training, which contains more than 13,000 samples within 16 classes. Each sample includes a point cloud and the semantic labels for points. Samples with more than 2048 points are selected and randomly downsampled to 2048 points.

In order to extract the 32-point structure point cloud $\mathcal{SP}_{\mathcal{GT}}$ and their semantic labels $\mathcal{SS}_{\mathcal{GT}}$, we use K-means clustering to find the centroids of the points with the same semantic labels. The value of K for each semantic label is determined by the percentage of the points with that label in the whole point cloud. The semantic labels are automatically inherited in the K allocating procedure.

Structure GAN

Structure GAN is constructed by a simple generator and a modified PointNet-based discriminator with a global average-pooling layer. The generator receives z as input and outputs the generated point cloud $\mathcal{SP}_{\mathcal{GE}}$ with $\mathcal{SS}_{\mathcal{GE}}$. A multi-layer perceptron(MLP) is used to transform z into a 256-dim vector. A fully connected layer is succeeded, aiming to convert the vector into 32 positions in space, namely $\mathcal{SP}_{\mathcal{GE}}$. Another fully connected layer and a softmax layer convert the 256-dim vector into $32 \times c$ scores represent the probability of semantic labels, namely $\mathcal{SS}_{\mathcal{GE}}$.

Both $\mathcal{SP}_{\mathcal{GE}}$ and $\mathcal{SS}_{\mathcal{GE}}$ are fed into the first discriminator, which we call structure discriminator. Note that $\mathcal{SP}_{\mathcal{GE}}$ and $\mathcal{SS}_{\mathcal{GE}}$ are both treated as the points' feature and are balanced by two fully connected layers. A PointNet-based network is applied in the discriminator. Since the ShapeNet-Partseg is a normalized dataset, we remove the T-Net of PointNet.

In experiments, we found that some points in $\mathcal{SP}_{\mathcal{GE}}$ are clustered within a very small distance. This phenomenon also causes the distortion of the semantic types' distribution. In some generated samples in the class of airplane, half of the points are generated as the tail wings. This might be caused by the max-pooling layer in PointNet. By using the max-pooling layer among all points, PointNet is not sensitive to the spatial distribution of points. So a global average-pooling layer is used instead.

Final GAN

Final GAN takes the $\mathcal{SP}_{\mathcal{GE}}$, $\mathcal{SS}_{\mathcal{GE}}$, and z as input, aiming to fit the distribution of the complete point cloud. A PointNet-like network is used to breed the points. A novel method, which is called Delta, and a self-attention layer are used to improve the performance.

Three fully connected layers are implemented first to balance the influences caused by different dimensions of $\mathcal{SP}_{\mathcal{GE}}$, $\mathcal{SS}_{\mathcal{GE}}$, and z, generating a feature for every point. The inputs of Final GAN only contain the coordinate and semantic information for every point. As the succeeded network is independent between points, the data transformation between points is indispensable to generate better complete point clouds. For this purpose, a self-attention layer is used to further transform the features.

With the features generated above, Final GAN outputs a complete point cloud by generating 64 points around every generated structure point. Ancestor relationships are established between the structure point cloud and the complete point cloud. Follow the inspiration of PointNet in dealing with the disorderliness of point cloud, we propose a novel method which is called Delta to breed the structure point cloud. Thirty-two MLPs with shared weights are applied to generate structure points. Each MLP transforms the features into 64×3 delta values that represent the relative positions between final points and their ancestor. By adding the coordinates of structure points, the Final GAN outputs the coordinates of the generated complete point cloud.

Most earlier methods directly use the models' output as the generated point cloud(Achlioptas et al. 2017; Valsesia, Fracastoro, and Magli 2018), or generate a transformation matrix and branch points by multiple the matrix with the ancestor's coordinates(Shu, Park, and Kwon 2019). To control the generated shape by changing the structure point cloud, the points are expected to be closer to their ancestor. Earlier methods lead to a mixture of all child points around the origin after initialization. The PointNet-based discriminator only cares about the overall shape, and doesn't have the ability to re-cluster the points belong to the same ancestor. In our method, points will be distributed around their ancestors after initialization, which meets our expectation.

Loss Function

Four networks are proposed in CPCGAN, which we represent as G_s, G_f, D_s, D_f. We adopt the loss function intro-

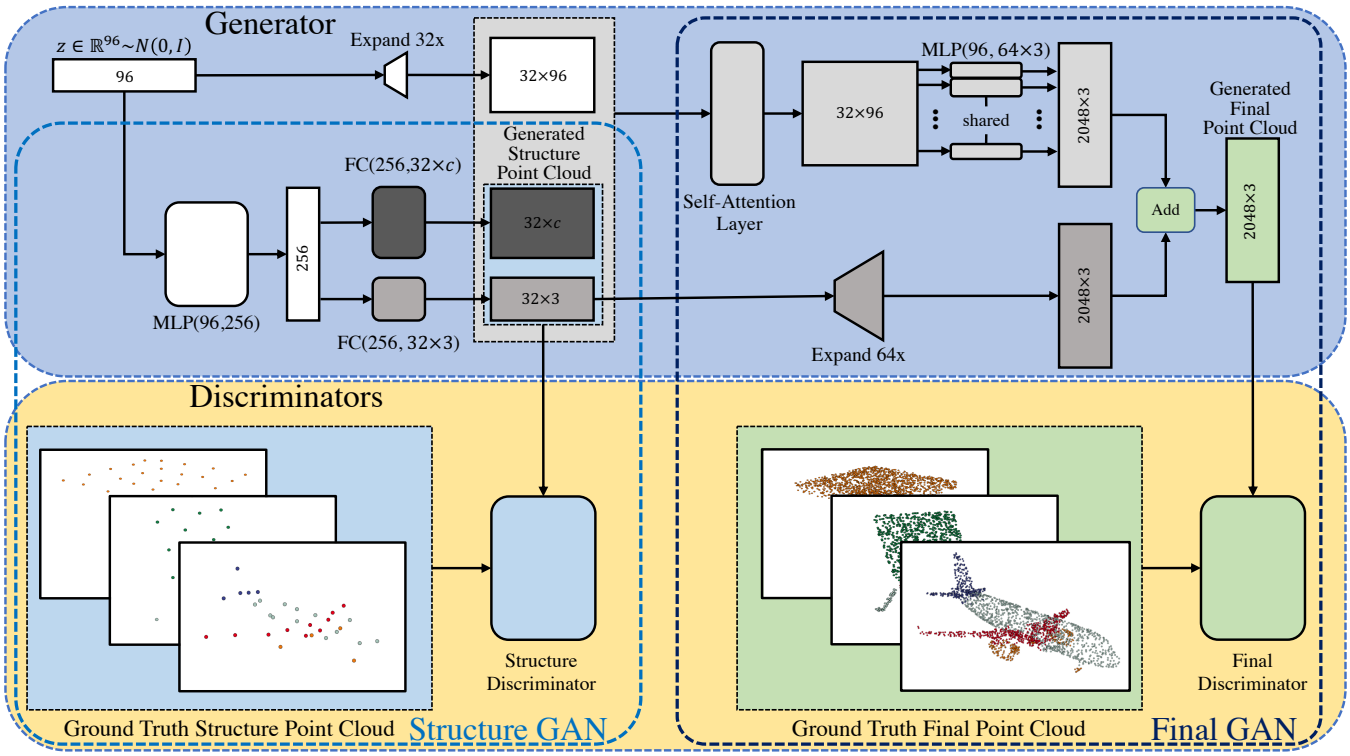

Figure 2: Network Structure of CPCGAN. Rectangles represent vectors or matrixes, rounded rectangles represent networks. Note that c is the number of semantic types, expand means to copy the data several times and concatenate them.

duced in Wasserstein GAN(Arjovsky, Chintala, and Bottou 2017) with gradient penalty(Gulrajani et al. 2017). Four loss functions for four networks are defined as

$$L_{G_s} = -\mathbb{E}_{z\sim Z}[D_s(G_s(z))]$$

$$L_{D_s} = \mathbb{E}_{z\sim Z}[D_s(G_s(z))] - \mathbb{E}_{x_s\sim R_s}[D_s(x_s)] + \lambda_{gp}\mathbb{E}_{\hat{x_s}}[(||\nabla_{\hat{x_s}} D_s(\hat{x_s})||_2 - 1)^2]$$

$$L_{G_f} = -\mathbb{E}_{z\sim Z}[D_f(G_f(z))]$$

$$L_{D_f} = \mathbb{E}_{z\sim Z}[D_f(G_f(z))] - \mathbb{E}_{x_f\sim R_f}[D_f(x_f)] + \lambda_{gp}\mathbb{E}_{\hat{x_f}}[(||\nabla_{\hat{x_f}} D_s(\hat{x_f})||_2 - 1)^2]$$

where Z represents a latent code distribution, which we designed as $Z = N(0, I)$. x_s denotes the structure point cloud, x_f denotes the complete point cloud, R_s and R_f represent the real distribution of structure and complete point clouds, the formulas after λ_{gp} are the gradient penalty terms, which were proposed in (Gulrajani et al. 2017).

Final loss functions L_G and L_D are defined as

$$L_G = L_{G_f} + \lambda_s L_{G_s} \quad (1)$$

$$L_D = L_{D_f} + \lambda_s L_{D_s} \quad (2)$$

The hyperparameters, λ_{gp} and λ_s, are set to 10 and 0.5 respectively.

Controllable Generation Method

Note that Structure GAN is designed to learn the distribution of structure point clouds $\mathcal{SP}_{\mathcal{GT}}$ and their semantic labels $\mathcal{SS}_{\mathcal{GT}}$, and to generate $\mathcal{SP}_{\mathcal{GE}}$ and $\mathcal{SS}_{\mathcal{GE}}$ with the same matrix shape as $\mathcal{SP}_{\mathcal{GT}}$ and $\mathcal{SS}_{\mathcal{GT}}$ accordingly. It is obvious that we can replace the $\mathcal{SP}_{\mathcal{GE}}$ and $\mathcal{SS}_{\mathcal{GE}}$ to handcrafted $\mathcal{SP}_{\mathcal{GT}}$ and $\mathcal{SS}_{\mathcal{GT}}$, and feed them into Final GAN to control the generation of complete point clouds. Fig. 3 (b) shows the data flow when the CPCGAN model switches to controllable generation mode. With the help of the self-attention layer and MLPs based on PointNet, Final GAN can receive handcrafted $\mathcal{SP}_{\mathcal{GT}}$ and $\mathcal{SS}_{\mathcal{GT}}$ without caring the order of the points. Latent code z is still fed into Final GAN in order to generate various samples when the handcrafted $\mathcal{SP}_{\mathcal{GT}}$ and $\mathcal{SS}_{\mathcal{GT}}$ are unchanged.

Training Strategies

To accelerate the training speed and improve the performance of CPCGAN, a two-step training strategy is used in the training period. In the first step, we train Structure GAN and Final GAN separately, which is achieved by feeding $\mathcal{SP}_{\mathcal{GT}}$ and $\mathcal{SS}_{\mathcal{GT}}$ instead of $\mathcal{SP}_{\mathcal{GE}}$ and $\mathcal{SS}_{\mathcal{GE}}$ into Final GAN. Fig. 3 (c) shows the data flow for the first step. In the second step, we train structure GAN and final GAN together. Fig. 3 (a) shows the data flow for the second step. In experiments, we train CPCGAN in the first step for 500 epochs and then in the second step for 1500 epochs.

Class	Model	FPD↓	JSD↓	MMD-CD↓	MMD-EMD↓	COV-CD↑	COV-EMD↑
Chair	r-GAN*	1.860	0.238	0.0029	0.136	33	13
	Valsesia et al.*	-	0.100	0.0029	0.097	30	26
	tree-GAN	1.114	0.0725	0.00191	0.0900	60.21	33.92
	CPCGAN(ours)	**0.877**	**0.0433**	**0.00186**	**0.0753**	**62.33**	**50.41**
Airplane	r-GAN*	1.016	0.182	0.0009	0.094	31	9
	Valsesia et al.*	-	0.083	0.0008	0.071	31	14
	tree-GAN	0.549	0.0854	0.00039	0.0584	58.40	23.13
	CPCGAN(ours)	**0.522**	**0.0296**	**0.00038**	**0.0417**	**59.63**	**45.36**

Table 1: Quantitative comparison in terms of FPD and the metrics proposed by Achlioptas et al.(Achlioptas et al. 2017). Bold values denote the best results. The * indicates that the results reported in (Shu, Park, and Kwon 2019) and (Valsesia, Fracastoro, and Magli 2018) are cited for those models. Results of tree-GAN are based on the model we trained using original code and default settings. All metrics we have tested are calculated by averaging the results of 100 times of evaluation.

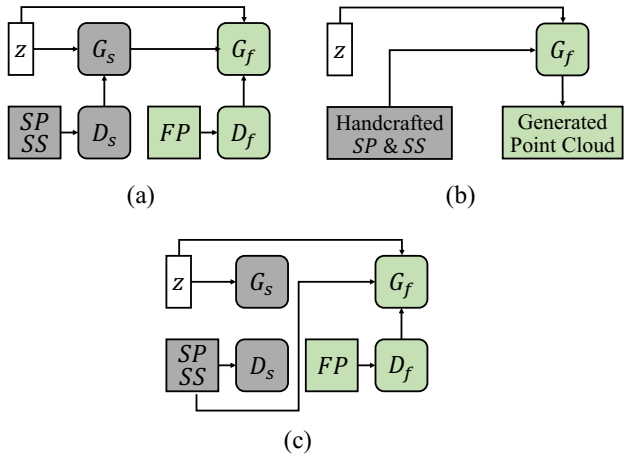

Figure 3: (a)data flow for second step training period, also an abstract of Fig. 2. (b) data flow for controallable generation mode. (c) data flow for first step training period.

Experiments

Datasets and Implementation Details

We train and test our CPCGAN on the ShapeNet-Partseg dataset. The dataset is designed for the 3D shape part segmentation task and has semantic label for every point. ShapeNet is the most popular object-level synthetic dataset with semantic information. Lots of datasets are sampled from ShapeNet including ShapeNet-Partseg, which means they have similar shapes and little differences will be made when changing between those datasets. Thus we use ShapeNet-Partseg as the only data source. Two classes that previous works used are chosen to compare quantitatively.

On implementation of CPCGAN, Adam optimizers are used for every sub-networks with a learning rate of $\alpha = 0.01$, coefficients $\beta_1 = 0$ and $\beta_2 = 0.99$. In loss function, the hyperparameters λ_{gp} and λ_s are set to 10 and 0.5 respectively. In the training period, the batch size is set to 128 and both discriminators in Structure GAN and Final GAN are updated three times per iteration. The size of the structure point cloud is set to 32. The size of the complete point cloud is set to 2048, which is the same with tree-GAN(Shu, Park, and Kwon 2019) and r-GAN(Achlioptas et al. 2017).

Evaluation Metrics

In order to evaluate the performance of the complete point cloud generation, metrics in (Achlioptas et al. 2017; Shu, Park, and Kwon 2019) are applied. CPCGAN and tree-GAN(Shu, Park, and Kwon 2019) are trained in the experiments. In the training period, the FPD metrics proposed by (Shu, Park, and Kwon 2019) are the major validation metrics. Because the discrepancy on FPD score among epochs are large enough, the FPD can indicate the performance differences between epochs. We choose the epochs with the best FPD score on both CPCGAN and tree-GAN to compare. However, the FPD is not stable enough for comparison between two well-trained models, because its value may vary by ± 0.2 in our experiments. Therefore, in the evaluation period, we apply FPD on both models for 100 times and take the average, which is different from the original code of tree-GAN(Shu, Park, and Kwon 2019).

In the generated semantic label evaluation, it is hard to give the ground truth semantic labels for a generated point cloud. We use ShellNet(Zhang, Hua, and Yeung 2019) to generate semantic labels as the ground truth, which may contain incorrect labels. A ShellNet model pre-trained on the ShapeNet-Partseg dataset is used. Average IoU, accuracy, and frequency weighted IoU of semantic label generation are calculated and named as $mIoU$, $mAcc$ and $fwIoU$ respectively. Tree-GAN also claimed the ability to generate point cloud with semantic information which is reflected by the leaf layer of the tree structure network. In treeGAN , each ancestor has 64 leaf nodes, who share the same semantic type. However, the semantic label is not explicit defined. This causes the difficulty in evaluating the semantic generation performance. As the generated ground truth have a label for every point, A voting algorithm, which is applied on the 64 points with the same ancestor, is applied to produce labels for the points generated by tree-GAN. With these produced labels, the Average IoU and accuracy are reported as $mIoU_{NL}$ and $mAcc_{NL}$ respectively, which "NL" stands for "No Label". The same NL metrics are also applied to CPCGAN for comparison.

Model	#G params	#D params	batch size	memory	time/100 samples
tree-GAN	$40.690M$	$2.536M$	20	$\sim 8821M$	$\sim 4.15s$
CPCGAN	$1.904M$	$2.693M$	128	$\sim 16347M$	$\sim 2.63s$

Table 2: Comparisons with tree-GAN on computational efficiency. #G params denotes the number of parameters in generator and #D params denotes the number of parameters in discriminator(s). Time efficiency results are tested on a single RTX Titan GPU. The settings of tree-GAN are the default settings in the open source code provided by the authors.

Class	Delta	Attention	Avg-pool	FPD↓	$mIoU\% \uparrow$	$fwIoU\% \uparrow$	$mAcc\% \uparrow$
Chair				1.240	24.781	18.961	30.621
	√			1.188	69.730	68.740	79.176
	√	√		1.045	**75.332**	73.609	82.924
	√	√	√	**0.877**	59.003	**76.269**	**84.954**
Airplane				0.537	13.700	17.817	29.346
	√			0.617	59.407	60.336	74.157
	√	√		0.696	59.747	72.220	81.681
	√	√	√	**0.522**	**69.677**	**76.436**	**85.865**

Table 3: Results for ablation studies. Bold values denote the best results. Delta represents the last component of Final GAN, which generates the delta values from structure points and produces the complete point clouds. Attention represents the self-attention layer. Avg-pool represents the replacement of the max-pooling layer in the discriminator of Structure GAN.

Quantitative Comparisons

Quantitative comparisons on point cloud generation. In the comparison of point cloud generation, metrics in (Achlioptas et al. 2017) and (Shu, Park, and Kwon 2019) are used. Table. 1 shows the comparison results. The proposed CPCGAN significantly outperforms other GANs in terms of all metrics. Compared with the chamfer distance based metrics (indicated with the suffix "-CD"), our CPCGAN has more significant improvements on the earth move distance based metrics (indicated with the suffix "-EMD"). This proves the point clouds generated by CPCGAN are more evenly distributed in space, and also closer to the distribution of point clouds in dataset. This is attributed to the structure point clouds, which restrict the distribution of complete point clouds in 3D space.

Quantitative comparisons on semantic labels generation. In the comparison of semantic labels generation, the metrics we proposed are applied to tree-GAN and our CPCGAN. Table. 4 shows the comparison results. Some other results we generated are shown in Fig. 4. Tree-GAN generates point clouds without label, thus we use the "NL" metrics on tree-GAN's generation. It's obvious that our CPCGAN outperforms in these metrics.

In some classes, the $mIoU$ score is notably lower than other IoU scores. Some semantic types with few points cause this phenomenon. The $mIoU$ score is calculated by directly averaging the $IoUs$ of each semantic type. A semantic type with few points may cause a large influence on $mIoU$ once it was not predicted correctly.

Quantitative comparison on computational efficiency. Table. 2 shows the comparison with tree-GAN on both time efficiency and space efficiency. With the help of two-stage GAN framework and structure point clouds, CPCGAN can use less computational resources and larger batch size to achieve better performance. It takes about 2 days on an RTX Titan GPU when we train the CPCGAN in the class of airplanes which has nearly 2000 samples.

Class	**Model**	**mIoU%**	**fwIoU%**	**mAcc%**
Chair	CPCGAN	59.003	76.269	84.954
Airplane	CPCGAN	69.677	76.436	85.865
Class	**Model**	**mIoU_{NL}%**	**mAcc_{NL}%**	
Chair	tree-GAN	85.342	91.605	
	CPCGAN	**89.814**	**94.925**	
Airplane	tree-GAN	54.263	79.072	
	CPCGAN	**80.216**	**92.355**	

Table 4: Semantics generation performance comparison with tree-GAN. Larger value is better. Bold values denote the better results. Statistics are the means of 10^4 samples generated by those models.

Ablation Study

We perform ablation studies to investigate the components of our CPCGAN. Three components of CPCGAN, which abbreviated as **Delta**, **Attention** and **Avg-pool**, are significantly different from those of other 3D point clouds GANs. Table. 3 shows the contributions of these components. The **Delta** is the basis of the semantic label generation ability. Without this component, points won't be close to their ancestors. This causes the semantic labels of points, which are inherited from their ancestors, become meaningless. The fluctuation in the $mIoU$ is caused by the calculation method of $mIoU$, which has been explained in Section 4.3. The improvements in most metrics prove the positive influences of those components.

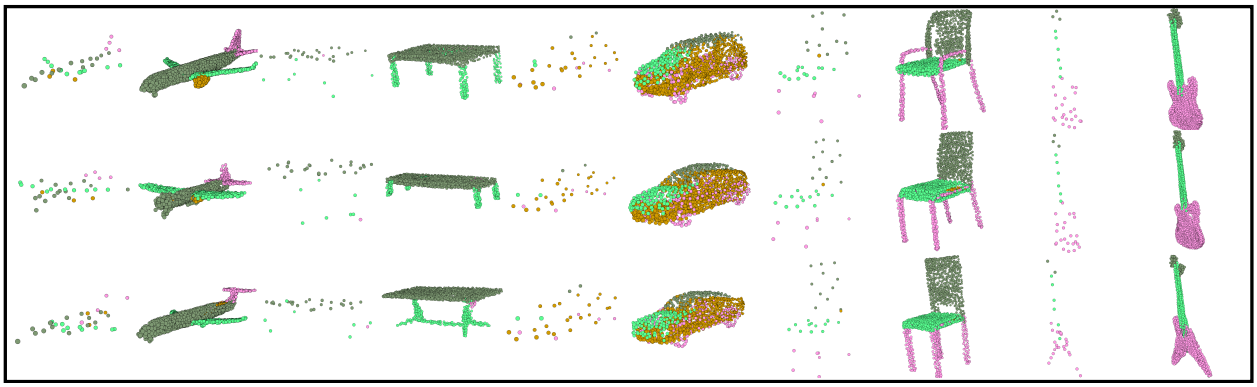

Figure 4: Examples of generated point clouds. The same color in the same class denotes the same semantic types.

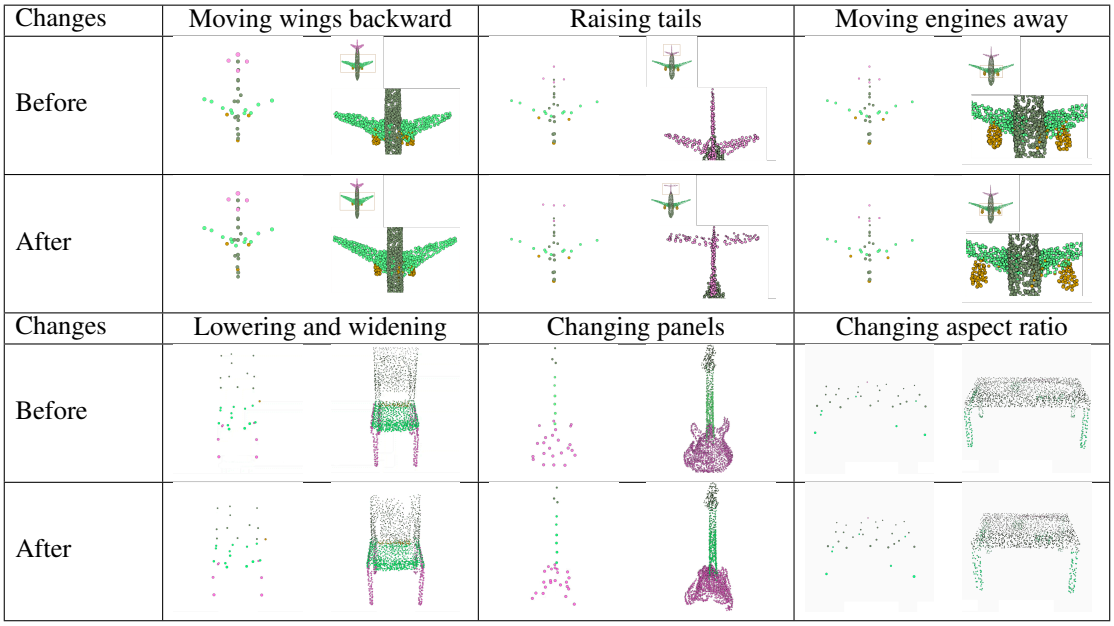

Figure 5: Appearance changes when structure point cloud changes.

Controllable Generation

By changing the structure point clouds, which are fed into Final GAN, CPCGAN has the ability to control the appearance of the complete point clouds. Fig. 5 show examples of this control ability. Note that the shape of tail wings are changing with the raising. That proves the change of structure is not only the translation of parts. The information of structure points' coordinates and relative positions between points are also guiding the shape generation.

Conclusions

In this paper, we proposed a generative adversarial network called CPCGAN, which aims to generate 3D point clouds and their semantic labels. A two-stage GAN framework is applied to ensure CPCGAN has the controllability on the appearance of the final results. A sparse point cloud, which we call structure point cloud, is introduced to make CPCGAN able to generate meaningful semantic labels. Through various experiments, we demonstrate that CPCGAN can generate semantic labels for points and control the generated structure. Our framework outperforms other 3D point cloud generation GANs in terms of generation performance and computational efficiency.

Acknowledgements

This work was supported by National Natural Science Fund of China (61672165) and Zhejiang Lab (2019KD0AB06). Cheng Jin is the corresponding author.

References

Achlioptas, P.; Diamanti, O.; Mitliagkas, I.; and Guibas, L. 2017. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392* .

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* .

Atzmon, M.; Maron, H.; and Lipman, Y. 2018. Point convolutional neural networks by extension operators. *ACM Transactions on Graphics (TOG)* 37(4): 1–12.

Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; Xiao, J.; Yi, L.; and Yu, F. 2015. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago.

Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; and Abbeel, P. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, 2172–2180.

Dai, A.; Chang, A. X.; Savva, M.; Halber, M.; Funkhouser, T.; and Nießner, M. 2017. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5828–5839.

Dong, H.; Supratak, A.; Mai, L.; Liu, F.; Oehmichen, A.; Yu, S.; and Guo, Y. 2017. TensorLayer: A Versatile Library for Efficient Deep Learning Development. *ACM Multimedia* URL http://tensorlayer.org.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved training of wasserstein gans. In *Advances in neural information processing systems*, 5767–5777.

Hui, L.; Xu, R.; Xie, J.; Qian, J.; and Yang, J. 2020. Progressive Point Cloud Deconvolution Generation Network. In *ECCV*.

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1125–1134.

Karras, T.; Laine, S.; and Aila, T. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4401–4410.

Li, R.; Li, X.; Fu, C.-W.; Cohen-Or, D.; and Heng, P.-A. 2019. Pu-gan: a point cloud upsampling adversarial network. In *Proceedings of the IEEE International Conference on Computer Vision*, 7203–7212.

Li, Y.; Bu, R.; Sun, M.; and Chen, B. 2018. PointCNN. *arXiv preprint arXiv:1801.07791* .

Lin, J.; Xia, Y.; Qin, T.; Chen, Z.; and Liu, T.-Y. 2018. Conditional image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5524–5532.

Mao, J.; Wang, X.; and Li, H. 2019. Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, 1578–1587.

Mirza, M.; and Osindero, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* .

Perarnau, G.; Van De Weijer, J.; Raducanu, B.; and Álvarez, J. M. 2016. Invertible conditional gans for image editing. *arXiv preprint arXiv:1611.06355* .

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, 5099–5108.

Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* .

Rott Shaham, T.; Dekel, T.; and Michaeli, T. 2019. SinGAN: Learning a Generative Model from a Single Natural Image. In *Computer Vision (ICCV), IEEE International Conference on*.

Shu, D. W.; Park, S. W.; and Kwon, J. 2019. 3d point cloud generative adversarial network based on tree structured graph convolutions. In *Proceedings of the IEEE International Conference on Computer Vision*, 3859–3868.

Su, H.; Jampani, V.; Sun, D.; Maji, S.; Kalogerakis, E.; Yang, M.-H.; and Kautz, J. 2018. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2530–2539.

Valsesia, D.; Fracastoro, G.; and Magli, E. 2018. Learning localized generative models for 3d point clouds via graph convolution. In *International conference on learning representations*.

Wang, X.; and Gupta, A. 2016. Generative Image Modeling using Style and Structure Adversarial Networks. In *ECCV*.

Wang, X.; Yu, K.; Wu, S.; Gu, J.; Liu, Y.; Dong, C.; Qiao, Y.; and Loy, C. C. 2018. ESRGAN: Enhanced super-resolution generative adversarial networks. In *The European Conference on Computer Vision Workshops (ECCVW)*.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)* 38(5): 1–12.

Wu, W.; Qi, Z.; and Fuxin, L. 2019. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9621–9630.

Yang, Y.; Feng, C.; Shen, Y.; and Tian, D. 2018. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 206–215.

Yu, L.; Li, X.; Fu, C.-W.; Cohen-Or, D.; and Heng, P.-A. 2018a. EC-Net: an Edge-aware Point set Consolidation Network. In *ECCV*.

Yu, L.; Li, X.; Fu, C.-W.; Cohen-Or, D.; and Heng, P.-A. 2018b. PU-Net: Point Cloud Upsampling Network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yuan, W.; Khot, T.; Held, D.; Mertz, C.; and Hebert, M. 2018. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, 728–737. IEEE.

Zhang, Z.; Hua, B.-S.; and Yeung, S.-K. 2019. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, 1607–1616.

Zhou, Y.; and Tuzel, O. 2018. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4490–4499.