Conditional GAN for Point Cloud Generation

Zhulun Yang[1,2][0000−0002−2018−5036], Yijun Chen[1,2], Xianwei Zheng[3] ⋆, Yadong Chang[1,2], and Xutao Li[1,2]

[1] Key Lab of Digital Signal and Image Processing of Guangdong Province, Shantou University, Shantou, 515063, China
[2] Department of Electronic Engineering, Shantou University, Shantou, 515063, China
{15zlyang3,21yjchen1,21ydchang,lixt}@stu.edu.cn
[3] School of Mathematics and Big Data, Foshan University, Foshan, 52800, China
alex.w.zheng@hotmail.com

Abstract. Recently, 3D data generation problems have attracted more and more research attention and have been addressed through various approaches. However, most of them fail to generate objects with given desired categories and tend to produce hybrids of multiple types. Thus, this paper proposes a generative model for synthesizing high-quality point clouds with conditional information, which is called Point Cloud conditional Generative Adversarial Network (PC-cGAN). The generative model of the proposed PC-cGAN consists of two main components: a pre-generator to generate rough point clouds and a conditional modifier to refine the last outputs with specific categories. To improve the performance for multi-class conditional generation for point clouds, an improved tree-structured graph convolution network, called BranchGCN, is adopted to aggregate information from both ancestor and neighbor features. Experimental results demonstrate that the proposed PC-cGAN outperforms state-of-the-art GANs in terms of conventional distance metrics and novel latent metric, Frechet Point Distance, and avoids the intra-category hybridization problem and the unbalanced issue in generated sample distribution effectively. The results also show that PC-cGAN enables us to gain explicit control over the object category while maintaining good generation quality and diversity. The implementation of PC-cGAN is available at https://github.com/zlyang3/PC-cGAN.

1 Introduction

In recent years, point clouds, a popular representation for 3D realistic objects data, are adopted in various applications (e.g., object classification [7, 9, 12, 18, 21,23,32,36] , semantic segmentation [12,21,23,32,36], and shape completion [28, 33, 35]) and have become increasingly attractive in computer vision application, such as augmented reality [19, 20, 25] and virtual reality [2, 29]. As each point in raw point clouds consists of a Cartesian coordinate, along with other additional information such as a surface normal estimate and RGB color value, point clouds

⋆ Corresponding author

can represent a 3D object to capture intricate details by an unordered set of irregular points collected from the surface of the objects.

With huge demand for data to train models for the aforementioned applications, generative models, including generative adversarial networks (GANs) [10] and variational autoencoders (VAEs) [15], have draw significant attention to generate point cloud data with high quality and diversity. Some researchers [1] stacked fully connected layers to convert random latent codes into 3D point clouds, which were borrowed from the image generative domain. Additionally, some generative models [26, 31] have utilized graph convolution networks and k-nearest neighbor techniques to generate point clouds. Recently, SP-GAN [17] succeeds in synthesize diverse and high-quality shapes and promote controllability for part-aware generation and manipulation without any part annotations. And a novel GAN was proposed to generate 3D objects conditioned on a continuous parameter in [30].

However, part of these generative models need to train separate models for each categories, leading to poor reusability of them. Worse, most of the generative model for point clouds fail to actively synthesize objects with specific categories, which causes the intra-category hybridization problem. Since these models are trained to simulate the distribution of the training data with all kind of point clouds and generate objects in random category, they are likely to generate point clouds composed of parts of several different classes of objects as Figure 1 shown. In addition, as the datasets are unbalanced with regard to categories, aforementioned generative models also tend to generate point cloud with unbalanced distribution. Therefore, this paper proposes a novel framework, called PC-cGAN that can generate 3D point clouds from random latent codes with categories as auxiliary information to avoid the intra-category hybridization problem and data unbalance problem. Besides, to enhance our performance in terms of high quality and diversity, graph convolution layer, called BranchGCN, to aggregate information from ancestors and neighbors in feature spaces.

The main contributions of this paper are listed as follows.

- We present a BranchGCN, which passes messages from not only ancestors, but also neighbors.
- We propose a generative model, consisting of a pre-generator and a conditional modifier, to generate 3D point clouds based on the given category information in a supervised way.
- Using the conditional generation framework, the intra-category hybridization problem and the data unbalance problem in generated distribution can be solved directly.

The remainder of this paper is organized as follows. We give a summary about related studies in Section 2. In Section 3, we introduce the basic related concepts about GANs and graph convolution networks. And the details of our model are provided in Section 4. Next, We present the experimental setup and results in Section 5. A conclusion of our work is given in Section 6.

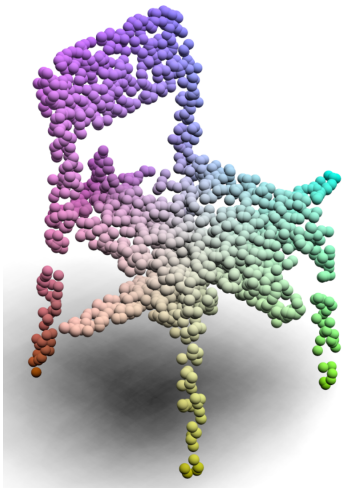

Fig. 1. An example of the intra-category hybridization phenomenon: The point cloud is combination of a chair and an airplane.

2 Related Works

Graph Convolutional Networks. During the past years, many works have focused on the generalization of neural networks for graph structures. [5] proposed graph convolution network for classification tasks. [16] introduced the Chebyshev approximation of spectral graph convolutions for semi-supervised graph learning. Even GCN [34] can be adopted to extract spatial-temporal features for time series data. And dynamic graph convolution network [32] was designed to extract feature for point clouds using the connectivity of pre-defined graphs. Similarly, TreeGCN [27] was introduced to represent the diverse typologies of point clouds.

GANs for Point Cloud. Although GANs for image generation takes have been comprehensively studies with success [3,8,10,11,14,24,37], but GANs for 3D point clouds have seldom been studies in computer vision domain. Recently, [1] proposed a GAN for 3D point clouds called r-GAN, the generator of which is based on fully connected layers. Since these layers failed to utilize the structural information of point clouds, the r-GAN met difficulty to synthesize realistic objects with diversity. In order to utilize the structural information in point clouds, [31] used graph convolutions in the generator for the first time. However, the computational cost for the construction of adjacency matrices is $O(V^2)$, which leads to lengthy training period. Therefore, tree-GAN [27] saved computational cost and time without construction of adjacency matrices. Instead, its generator used ancestor information from the tree to exploit the connectivity of a graph, in which only a list of tree structure is needed. But tree-GAN lacked

attention on neighbor information of each point clouds, which leaded to slow convergence in its training process.

Conditional Generation. In contrast to the aforementioned generative model that learned to directly generate point clouds without regard to category, several generative models in image processing received additional conditional parameters to construct specific objects or styles. Conditional GAN [22] used explicit condition to generate hand-written digit images by an additional auxiliary classifier, which formed the basis of many other conditional generative models. StyleGAN [14] and others [8, 37] investigated how to enhance desirable attributes of generated images selectively. However, in 3D computer vision, conditional generative models are rarely adopted to synthesize point cloud data. Recently, the generator of [4] introduced a progressive generative network that created both geometry and color for point clouds based on the given class labels. Their work focus on generating dense and colored point clouds and struggled with generating objects that have fewer samples in the training data. And SP-GAN [17] is able to promote controllability for part-aware shape generation and manipulation without any part annotations, while SP-GAN requires huge amount of training epochs. Moreover, the work in [30] proposed conditioning point cloud generation using continuous physical parameters, but not category information.

3 Preliminaries

3.1 GAN

In general, a generative adversarial network (GAN) is designed to train a generative sub-network that transfers Gaussian random vectors $z \in \mathbf{Z}$ ($\mathbf{Z}$ is a Gaussian distribution by default) into data in a real sample space $x \in \mathbf{X}$ ($\mathbf{X}$ is the set of training samples) to foolish the discriminator, and the discriminator that learns to judge samples from the given dataset and generator as real or fake. Therefore, the task of GAN could be formulated as the minimax objective:

$$\min_G \max_D \mathbb{E}_{x\sim\mathbb{P}_r}[\log(D((x))] + \mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[\log(1 - D(\tilde{x}))], \tag{1}$$

where $\mathbb{P}_r$ is the data distribution from the given dataset and $\mathbb{P}_g$ is the generative distribution defined by $\tilde{x} = G(z), z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Additionally, in order to increase diversity and stability of GAN, a gradient penalty that penalize the norm of gradient of the critic with respect to its input is utilized in improved WGAN, so that it can satisfy the 1-Lipschitz condition.

3.2 Graph Convolution Network

The classical graph convolution network can be defined as a network with multiple message passing layers of the form

$$\mathbf{x}_i^{l+1} = \sigma\left(W^l \mathbf{x}_i^l + \sum_{j\in\mathcal{N}(i)} U^l \mathbf{x}_j^l + b^l\right), \tag{2}$$

where $\sigma(\cdot)$ is the activation unit, $\mathbf{x}_i^l$ denotes the feature (e.g., 3D coordinate of a point cloud at the first layer) of i-th vertex in the graph at l-th layer, while $\mathcal{N}(i)$ denotes the set of all neighbor vertices of the i-th vertex. And W_l, U_l is learnable weights at l-th layer, with b_l as learnable bias at l-th layer. The first and second term in (2) are called the self-loop term and the neighbors term, respectively.

In tree-GAN [27], a novel GCN, called TreeGCN, is defined with tree structure, passing information from ancestors to descendants of points. TreeGCN can be formulated as

$$p_i^{l+1} = \sigma\left(W^l p_i^l + \sum_{q_j^l \in \mathcal{A}(p_i^l)} U^l q_j^l + b^l\right), \tag{3}$$

where p_i^l is the i-th point in the point cloud at the l-th layer, q_j^l is the j-th neighbor of p_i^l, and $\mathcal{A}(p_i^l)$ denotes the set of all ancestors of p_i^l. That means the second term in Equation 3 updates a current point based on all its ancestors but not its neighbors.

4 Methodology

We propose an approach inspired by cGAN [22] in image generation and tree-structured graph convolution operations in [27]. Instead of synthesizing samples without any instruction like previous GAN models for point clouds, we design a supervised generative model by restricting the output samples to a specific category and train a discriminator to judge whether the given samples are real or fake, and which categories the samples belong to.

Unlike previous conditional GANs in image processing, our generative model does not take a random vector and the given condition information as the direct input. The model generates an artificial sample without any restriction in the first step, which accelerates the convergence speed. Then the unconditioned sample is modified by the rest of the model with the given category condition. After modification, the conditional input forces the model to transform the generated sample to the givn category. We present the overview of our GAN framework in Figure 2. BrachGCN, the basic module of PC-cGAN, is introduced in Section 4.1, while in Section 4.2, we discuss the auxiliary supervised module to help GAN to generate category-special objects. And the loss functions for training procedure is explained in Section 4.3.

4.1 BranchGCN

To improve TreeGCN adopted in [27], we propose a new GCN combining TreeGCN with Dynamic Graph Convolution [32] and define it as BranchGCN. Since a set of point clouds is unable to be converted into a fixed graph before being processed, and those traditional GCNs require information about connectivity of a graph, tree-GAN [27] introduced TreeGCN to avoid use prior information about

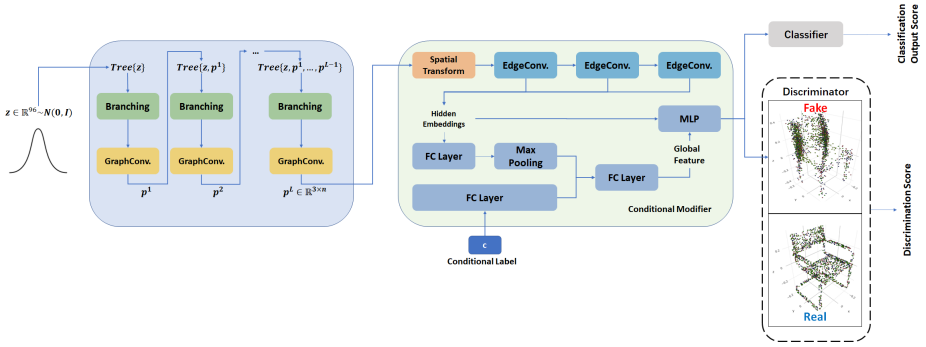

Fig. 2. Overview of model architecture: Our model consists of four parts: 1) Pre-Generator, 2) Conditional Modifier, 3) Discriminator, and 4) Classifier. The Pre-Generator generates the rough point clouds without any auxiliary information. The Conditional Modifier modifies the output point clouds of the Pre-Generator based on the given category labels. The Discriminator, meanwhile, tries to discriminate between the real point clouds and the generated point clouds. The classifier learns to classify the given point clouds, no matter whether they are real or artificial.

connectivity among vertices. However, TreeGCN only considers information from ancestors, but does not take horizontal connection into account. Therefore, we remain the ancestor term in TreeGCN, but add an extract term: neighbor term. The additional neighbor term is able to aggregate the information of vertices from their nearest neighbors. The neighbor term is beneficial to recompute the graph structure by using nearest neighbors in the embedded feature spaces produced by each layer. By adding the neighbor term, the receptive field can be expanded to the diameter of the point cloud. Specifically, we compute a pairwise Euclidean distance matrix in the feature spaces and then take closest k vertices for each single vertex in our experimental implementation. Figure 3 illustrates the differences between TreeGCN and BranchGCN.

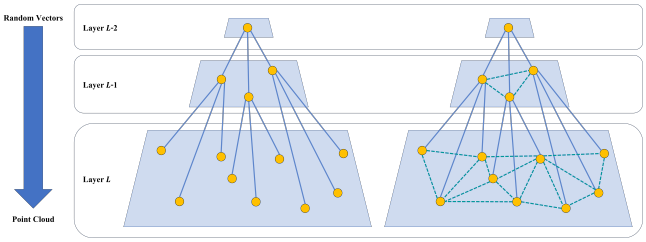

Fig. 3. TreeGCN (left) and BranchGCN (right).

To assist the neighbor term, we construct $\mathcal{G}^l$ as the k-nearest neighbor (k-NN) graph of points at l-th layer based on the pairwise distance between their

features, so that the connectivity $\mathcal{E}^l \subseteq \mathcal{V} \times \mathcal{V}$ between each point could be obtained. Accordingly, the output of this term could be formulated as

$$\sum_{k:(i,k)\in\mathcal{E}} H^l p_k^l, \tag{4}$$

where H^l is also learnable weights at l-th layer. It is noticeable that the graph structure is recomputed using nearest neighbors in the feature space built in each layer. By adding this neighbors term, the Branching plays an effective role to aggregate local features during generative procedure.

4.2 Conditional Generative Model

Since there is a lack of control on traditional GANs to generate artificial samples of specific categories, these models tend to learn one-to-one mappings from a random distribution to the distribution of the real dataset, which suffer from the risk to produce mixtures of different kinds of samples and tend to produce a unbalanced distribution just like the real dataset. We attribute these two issues to the lack of explicit control on generated samples' category. On the contrary, the conditional version of GAN takes the combination of a random vector and a specific label about the desired class at the input layer to generate samples with the specific shape. Therefore, to address these issues, the proposed GAN adopts another strategy to utilize conditional information in our work. We divide the generative model into two part: a pre-generator similar to the existing GANs, and a conditional modifier that enables the generative model to employ category information to generated point clouds. As for the pre-generator, we take the generator of tree-GAN as our backbone but replace the TreeGCN module with our BranchGCN, as Figure 2 shown. Meanwhile, we keep the Branching module in [27] to upsample the generated points from low-dimension random vectors. On the another hand, we extend the architecture of DGCNN to receive the output point clouds from the pre-generator, and an additional input that is represented as a one-hot code about category information. These codes and the outputs from the pre-generator are fed into the sub-network in our modifier together to synthesize point clouds based on the given labels.

With regard to our discriminator, since we add auxiliary information into generator, the corresponding discriminator should have the ability to distinguish which category the given point clouds belong to, and whether they are real or artificial. Consequently, an extra classifier is employed to recognize them. And our discriminator has two outputs: one for the adversarial feedback $D(x)$, and another to determine the classification results $\hat{y} = C(x)$. To adapt the new strategy, we train the overall model by Algorithm 1. Different from traditional GANs, we need to train the generator, the discriminator, and classifier simultaneously in each iteration.

Algorithm 1: Training the PC-cGAN model

$\theta_{(Gen,Mod)}, \theta_{Disc}, \theta_{Cls} \leftarrow$ initialize parameters
repeat
  $(\mathbf{X}, c) \leftarrow$ random point clouds and their category labels from dataset;
  $\mathbf{Z} \leftarrow$ samples from prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$;
  $\widetilde{\mathbf{X}} \leftarrow Mod(Gen(\mathbf{X}, c))$;
  $\hat{\mathbf{X}} \leftarrow$ samples from line segments between $\mathbf{X}$ and $\widetilde{\mathbf{X}}$;
  $\mathcal{L}_{(Gen,Mod)} \leftarrow -Disc(\widetilde{\mathbf{X}})$;
  $\mathcal{L}_{Disc} \leftarrow Disc(\widetilde{\mathbf{X}}) - Disc(\mathbf{X}) + \lambda_{gp}(\|\nabla Disc(\hat{\mathbf{X}})\|_2 - 1)^2$;
  $\mathcal{L}_{Cls} \leftarrow CrossEntropy(Cls(\mathbf{X}), c) + CrossEntropy(Cls(\widetilde{\mathbf{X}}), c)$;
  `// Update parameters according to gradients`
  $\theta_{(Gen,Mod)} \overset{+}{\leftarrow} -\nabla_{\theta_{(Gen,Mod)}}(\mathcal{L}_{(Gen,Mod)})$;
  $\theta_{Disc} \overset{+}{\leftarrow} -\nabla_{\theta_{Disc}}(\mathcal{L}_{Disc})$;
  $\theta_{Cls} \overset{+}{\leftarrow} -\nabla_{\theta_{Cls}}(\mathcal{L}_{Cls})$;
until *deadline*;

4.3 Loss Functions

The optimization objective for general GANs is composed of two components, the generative loss, adversarial loss with a gradient penalty, just like the improved WGAN [11]. However, since the category information is added into the proposed GAN, there are some differences inside the loss functions. In PC-cGAN, the loss function of the generator, $\mathcal{L}_{gen}$, is defined as

$$\mathcal{L}_{gen} = -\mathbb{E}_{z\in\mathcal{Z}, c\in\mathcal{C}}[D(G(z,c))] + \mathcal{L}_c(G(z,c),c), \tag{5}$$

where G and D represent the generator and discriminator, respectively. $\mathcal{Z}$ and $\mathcal{C}$ denote a latent Gaussian distribution, and the set of all the class labels. Besides, the last term in Equation 5 represents the loss for multi-class classification of generated point clouds, defined as $\mathcal{L}_c(G(z,c),c) = CrossEntropy(C(G(z,c)),c)$, where C denotes the multi-class classifier. On the another hand, the loss of the discriminator $\mathcal{L}_{disc}$ is formulated as

$$\mathcal{L}_{disc} = \mathbb{E}_{z\in\mathcal{Z}, c\in\mathcal{C}}[D(G(z,c))] - \mathbb{E}_{x\in\mathcal{R}}[D(x)] + \lambda_{gp}\mathcal{L}_{gp} + \mathcal{L}_c(x,c), \tag{6}$$

where x denotes real point clouds belonging to a real data distribution $\mathcal{R}$. Accordingly, in our model, we adopt a gradient penalty from WGAN, $\mathcal{L}_{gp} = \mathbb{E}_{\hat{x}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]$, where $\hat{x}$ is sampled uniformly along straight lines between pairs of points samples from the data distribution $\mathcal{R}$ and the generated distribution $G(z)$ ($z \in \mathcal{Z}$), and λ_{gp} is a weighting parameter that we set to 10.

For better convergence of our model, we add another discriminator to distinguish the generated point clouds from the pre-generator, which is illustrated in Figure 4. Therefore, the loss function of the generative model is refined as

$$\mathcal{L}'_{gen} = \mathcal{L}_{gen} - \mathbb{E}_{z\in\mathcal{Z}}[D'(G'(z))], \tag{7}$$

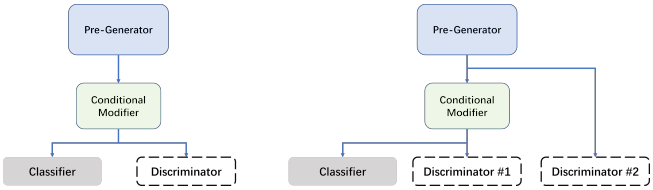

Fig. 4. The difference between the frameworks of the original PC-cGAN (left) and the improved PC-cGAN (right).

where G' and D' denote the pre-generator and the extra discriminator. Since the output point clouds from the pre-generator are lack of category constraint, the additional loss for multi-class classification $\mathcal{L}'_c$ is unnecessary. Similarly, the loss function of discriminator part is modified as

$$\mathcal{L}'_{disc} = \mathcal{L}_{disc} + \mathbb{E}_{z\in\mathcal{Z}}[D'(G'(z))] - \mathbb{E}_{x\in\mathcal{R}}[D'(x)] + \lambda_{gp}\mathcal{L}'_{gp}, \tag{8}$$

where $\mathcal{L}'_{gp}$ is the gradient penalty term for the extra discriminator D'.

5 Experimental Results

In this section, we demonstrate the point cloud generation using PC-cGAN on ShapeNetBenchmark [6] in two tasks: point cloud generation, and point cloud modification.

5.1 Generation Point Cloud Assessment

As for experimental details, an Adam optimizer is adopted to train both our generator and discriminator sub-networks with a learning rate of $\alpha = 10^{-4}$, $\beta_1 = 0$ and $\beta_2 = 0.99$. The discriminator with a classifier was updated ten times per iteration, while the generator was updated one time per iteration. After setting these hyper-parameters, a latent vector $z \in \mathbb{R}^{96}$ sampled from a Gaussian distribution $\mathcal{N}(0, \mathbf{I})$, and the total number of generated point clouds was set to $n = 2048$.

The conventional quantitative metrics proposed by [1] are used to evaluated the quality of generated point clouds by measuring matching distances between real and artificial point clouds. Besides, we also adopt Fréchet dynamic distance (FPD) [27], a nontrivial extension of Fréchet inception distance (FID) [13], to quantitatively evaluate generated point clouds. We generated 1000 random samples for each class and performed an evaluation using the aforementioned matrices. Table 1 presents the results along with quantitative comparison to previous studies [1] on conventional metrics, and Table 2 presents the FPD score. Note that separated models were trained in all the comparative methods to generated point clouds for different classes except [4] used the same model to generate point clouds for five classes, while the proposed model is able to generate point clouds of any category (16 classes) simultaneously on ShapeNetBenchmark dataset.

Table 1. A quantitative evaluation of the Jensen-Shannon divergence (JSD), the minimum matching distance (MMD), coverage (COV) with the Earth mover's distance (EMD), and the pseudo-chamfer distance (CD). Please refer to [1] for details regarding the metrics. For the GANs with $\star$, we adopted the results from [4]. The red and blue values indicate the best and the second best results for each metric, respectively. The resolution of the evaluated point clouds was 2048×3.

Class	Model	JSD ↓	MMD ↓		COV ↑	
			CD	EMD	CD	EMD
Airplane	r-GAN (dense)*	0.182	0.0009	0.094	31	9
	r-GAN (conv)*	0.350	0.0008	0.101	26	7
	Valsesia et al. (no up.)	0.164	0.0010	0.102	24	13
	Valsesia et al. (up.)	**0.083**	0.0008	0.071	31	14
	tree-GAN*	0.097	**0.0004**	**0.068**	**61**	**20**
	PC-cGAN (ours)	**0.086**	**0.0006**	**0.061**	**53**	**23**
Chair	r-GAN (dense)*	0.235	0.0029	0.136	33	13
	r-GAN (conv)*	0.517	0.0030	0.223	23	4
	Valsesia et al. (no up.)	0.119	0.0033	0.104	26	20
	Valsesia et al. (up.)	**0.100**	0.0029	**0.097**	30	26
	tree-GAN*	0.119	**0.0016**	0.101	**58**	**30**
	PCGAN*	**0.089**	0.0027	**0.093**	30	**33**
	PC-cGAN (ours)	0.119	**0.0026**	0.109	**38**	24
Table	PCGAN*	0.250	**0.0016**	0.097	10	9
	tree-GAN	0.074	0.0032	0.115	46	35
	PC-cGAN (ours)	**0.048**	0.0030	**0.096**	**50**	**47**
Motorbike	PCGAN*	0.093	0.0035	0.089	**45**	**43**
	tree-GAN	0.116	0.0015	**0.056**	18	35
	PC-cGAN (ours)	**0.062**	**0.0013**	0.069	25	38
Car	tree-GAN	0.080	0.0014	0.089	35	18
	PC-cGAN (ours)	**0.070**	0.0014	**0.073**	**38**	**19**
Guitar	tree-GAN	**0.046**	0.0008	**0.051**	**40**	18
	PC-cGAN (ours)	0.083	**0.0006**	0.061	32	**23**
All (16 classes)	r-GAN (dense)	0.171	0.0021	0.155	58	29
	tree-GAN	0.105	**0.0018**	0.107	**66**	39
	PC-cGAN (ours)	**0.034**	0.0034	**0.106**	47	**44**

Table 2. The FPD score for point cloud samples generated by generative models. Notice that the score for real point clouds are almost zero. The point clouds were evaluated at a resolution of 2048×3. The bold values denote the best results.

Class	r-GAN	tree-GAN	PC-cGAN (ours)
Airplane	1.860	**0.439**	0.747
Chair	1.016	**0.809**	1.948
All (16 classes)	4.726	3.600	**2.120**

According to the quantitative results from the given tables, we have achieved comparable results in synthesizing high-quality point clouds for each category. Even though our model fails to achieve the best in terms of some metrics for some classes, the model outperforms other GANs on the most metrics (*i.e.*,JSD, MMD-EMD, COV-EMD, and FPD), which demonstrates the effectiveness of the proposed method for multi-class generation. Again, we point out that all the methods train a separate network for each single class except that PCGAN [4] trains the single model for only five classes because of their lack of ability for conditional generation. By contrast, the proposed model uses the same network to generate samples of every single category (16 classes in total) to avoid the hybridization problem and data unbalance issue. Therefore, the quantitative results demonstrate that the proposed method can achieve the comparable performance alongside conditional generation.

In addition to the quantitative results, Figure 5 and Figure 6 show point clouds from the real dataset, and ones generated by the baseline in [27], and our PC-cGAN after 1000 epochs of training. In Figure 5, the generated samples are more realistic due to their large sample size. As for Figure 6, even though the generative results are not as realistic as Figure 5 owing to lack of samples, PC-cGAN can still generate point clouds whose shapes look like the given categories.

5.2 Point Cloud Modification

Since the conditional modifier in the proposed generative model could modify the unclassified point clouds from the pre-generator based on the given classes, it also can generate point clouds of the specified categories from any outputs of pre-generator. Figure 7 shows that during the former generative process, the pre-generator produce point clouds randomly, without specified category information, while the latter part modify the random point clouds into ones of the specified categories. And even the point clouds from the pre-generator seem to fall into the specified categories, the modifier also can enhance their realness by decreasing the distance between the distribution of generated point clouds and real point clouds further.

5.3 Ablation Study

To verify the effectiveness of the modules proposed in the previous section, including BranchGCN in section 4.1 and the improved loss functions in section 4.3, we further perform several ablation studies on PC-cGAN. For all experiments in this section, the models are trained in the same experimental configuration as section 5.1 on the table class. Note that although the table class is chosen as an example, all the models in this section are trained on complete ShapeNetBenchmark dataset (16 classes at all) to keep consistency with section 5.1. The results of our ablation studies are presented in Table 3. Applying BranchGCN module to aggregate not only ancestors' information but also features from neighbors brings considerable performance improvement in point cloud generation. The results also demonstrate that the effectiveness of the improved loss functions. Although

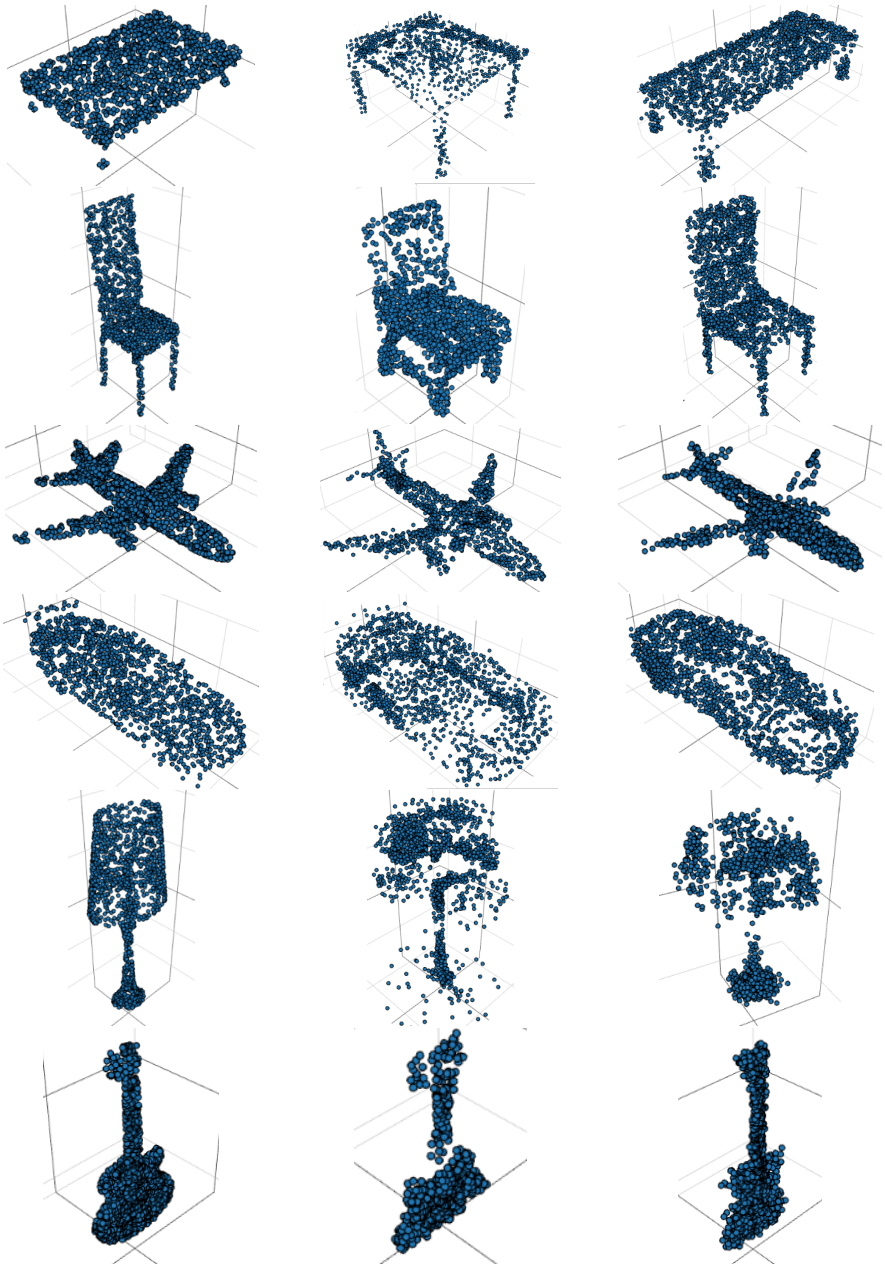

Fig. 5. Real point clouds from ShapeNetBenchmark [6] (left), and 3D point clouds generated by the baseline (middle), *i.e.*, tree-GAN [27], and our PC-cGAN (right).

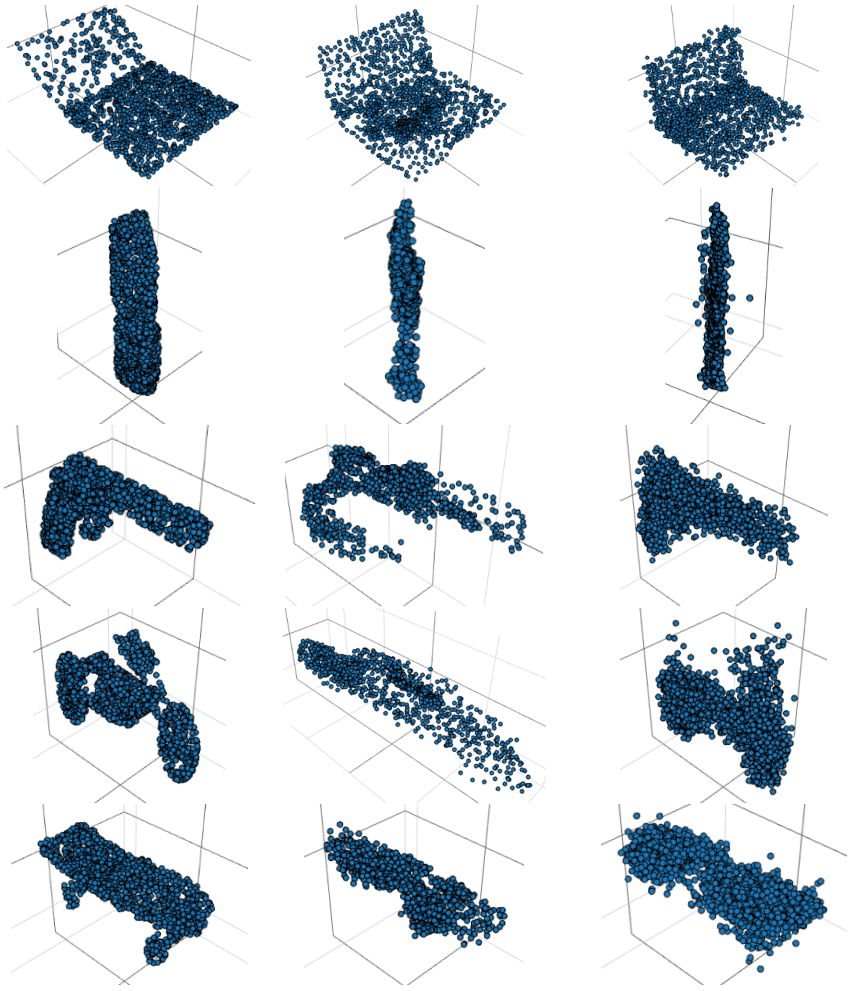

Fig. 6. A supplement for point clouds generation of other categories lacking sample data. The same order is adopted as Figure 5

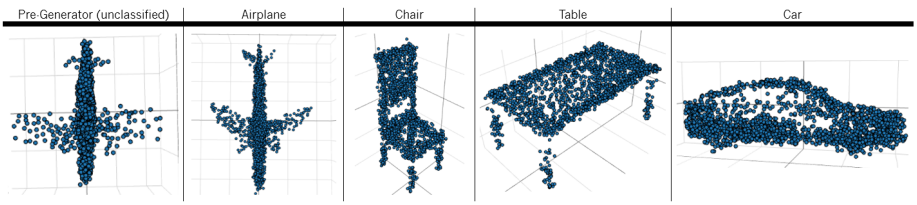

Fig. 7. Point cloud modification. The leftmost point cloud is fixed as a random object generated by the pre-generator, whose shape is closed to a coarse airplane in the implementation. The following four objects are generated by modifying the leftmost point cloud according to the given classes.

PC-cGAN with the losses requires more computational resources due to the extra discriminator, the additional discriminator can guarantee the pre-generator produces more realistic samples without category constraint, and enhance the quality of final outputs as mentioned in section 4.3.

Table 3. Supplemented ablation study on the table class. Improved loss used the framework of improved PC-cGAN. We try replacing BranchGCN module with previous TreeGCN in [27] [**BranchGCN**], and removing the additional discriminator with improved loss function [**Improved Loss**].

BranchGCN	**Improved Loss**	JSD↓	MMD↓		COV↑	
			CD	EMD	CD	EMD
✘	✘	0.584	0.0215	0.178	20	15
✘	✔	0.174	0.0164	0.156	22	20
✔	✘	0.080	0.0150	0.113	46	35
✔	✔	**0.048**	**0.0030**	**0.096**	**50**	**47**

6 Conclusions

In this work, a improved tree-structured graph convolution network is proposed to aggregate information from ancestors and neighbors in feature space. Based on that, we propose a conditional GAN for point clouds, called PC-cGAN, to generate 3D objects of specific categories. To introduce the given information about categories into the generative model, we propose a two-stage generator, which consist of a pre-generator and a conditional modifier to solve the intra-category hybridization problem and the data unbalance issue. Based on the two-stage structure, we build the corresponding loss functions and the training algorithm to assist our model to converge effectively. By comparisons with recent generation approaches, we evaluate the generated point clouds by PC-cGAN on the conventional metrics, including FPD score. The quantitative and visual results show that our model is capable of simulating high-quality and diverse samples for multi-class point cloud generation.

Acknowledgement This work was supported by the National Natural Science Foundation of China (No. 61471229 and No. 61901116), the Natural Science Foundation of Guangdong Province (No. 2019A1515011950), the Guangdong Basic and Applied Basic Research Foundation (No. 2019A1515010789 and No. 2021A1515012289), and in part by the Key Field Projects of Colleges and Universities of Guangdong Province (No. 2020ZDZX3065), and in part by Shantou University Scientific Research Foundation for Talents under Grant NTF19031.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3D point clouds. In: Dy, J., Krause, A. (eds.) Pro-

ceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 40–49. PMLR (10–15 Jul 2018), https://proceedings.mlr.press/v80/achlioptas18a.html
2. Alexiou, E., Yang, N., Ebrahimi, T.: PointXR: A toolbox for visualization and subjective evaluation of point clouds in virtual reality. In: 2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX). IEEE (may 2020). https://doi.org/10.1109/qomex48832.2020.9123121
3. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein GAN. arXiv e-prints arXiv:1701.07875 (Jan 2017)
4. Arshad, M.S., Beksi, W.J.: A progressive conditional generative adversarial network for generating dense and colored 3d point clouds (nov 2020). https://doi.org/10.1109/3dv50981.2020.00081
5. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014), http://arxiv.org/abs/1312.6203
6. 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., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015)
7. Charles, R.Q., Su, H., Kaichun, M., Guibas, L.J.: PointNet: Deep learning on point sets for 3d classification and segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (jul 2017). https://doi.org/10.1109/cvpr.2017.16
8. Choi, Y., Choi, M., Kim, M., Ha, J., Kim, S., Choo, J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 8789–8797. Computer Vision Foundation / IEEE Computer Society (2018). https://doi.org/10.1109/CVPR.2018.00916, http://openaccess.thecvf.com/content_cvpr_2018/html/Choi_StarGAN_Unified_Generative_CVPR_2018_paper.html
9. Feng, M., Zhang, L., Lin, X., Gilani, S.Z., Mian, A.: Point attention network for semantic segmentation of 3d point clouds. Pattern Recognit. **107**, 107446 (2020). https://doi.org/10.1016/j.patcog.2020.107446, https://doi.org/10.1016/j.patcog.2020.107446
10. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial networks. CoRR **abs/1406.2661** (2014), http://arxiv.org/abs/1406.2661
11. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf
12. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: PCT: Point cloud transformer. Computational Visual Media **7**(2), 187–199 (apr 2021). https://doi.org/10.1007/s41095-021-0229-5

13. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. pp. 6626–6637 (2017), https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html
14. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4396–4405 (2019). https://doi.org/10.1109/CVPR.2019.00453
15. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014), http://arxiv.org/abs/1312.6114
16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings (2017), https://openreview.net/forum?id=SJU4ayYgl
17. Li, R.; Li, X.; Hui, K.-H. ; Fu, C.-W.: SP-GAN:Sphere-Guided 3D Shape Generation and Manipulation. In: ACM Transactions on Graphics (Proc. SIGGRAPH), ACM, 2021, **40**
18. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada. pp. 828–838 (2018), https://proceedings.neurips.cc/paper/2018/hash/f5f8590cd58a54e94377e6ae2eded4d9-Abstract.html
19. Lim, S., Shin, M., Paik, J.: Point cloud generation using deep local features for augmented and mixed reality contents. In: 2020 IEEE International Conference on Consumer Electronics (ICCE). IEEE (jan 2020). https://doi.org/10.1109/icce46568.2020.9043081
20. Ma, K., Lu, F., Chen, X.: Robust planar surface extraction from noisy and semi-dense 3d point cloud for augmented reality. In: 2016 International Conference on Virtual Reality and Visualization (ICVRV). IEEE (sep 2016). https://doi.org/10.1109/icvrv.2016.83
21. Ma, X., Qin, C., You, H., Ran, H., Fu, Y.: Rethinking network design and local geometry in point cloud: A simple residual MLP framework. CoRR **abs/2202.07123** (2022), https://arxiv.org/abs/2202.07123
22. Mirza, M., Osindero, S.: Conditional generative adversarial nets. CoRR **abs/1411.1784** (2014), http://arxiv.org/abs/1411.1784
23. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 5105–5114. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)
24. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: Bengio, Y., LeCun, Y. (eds.) 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016), http://arxiv.org/abs/1511.06434

25. Sagawa, H., Nagayoshi, H., Kiyomizu, H., Kurihara, T.: [POSTER] hands-free AR work support system monitoring work progress with point-cloud data processing. In: 2015 IEEE International Symposium on Mixed and Augmented Reality. IEEE (sep 2015). https://doi.org/10.1109/ismar.2015.50
26. Sarmad, M., Lee, H.J., Kim, Y.M.: RL-GAN-net: A reinforcement learning agent controlled GAN network for real-time point cloud shape completion. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (jun 2019). https://doi.org/10.1109/cvpr.2019.00605
27. Shu, D., Park, S.W., Kwon, J.: 3d point cloud generative adversarial network based on tree structured graph convolutions. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE (oct 2019). https://doi.org/10.1109/iccv.2019.00396
28. Singh, P., Sadekar, K., Raman, S.: Treegcn-ed: Encoding point cloud using a tree-structured graph network. CoRR **abs/2110.03170** (2021)
29. Tredinnick, R., Broecker, M., Ponto, K.: Progressive feedback point cloud rendering for virtual reality display. In: 2016 IEEE Virtual Reality (VR). IEEE (mar 2016). https://doi.org/10.1109/vr.2016.7504773
30. Triess, L.T., Bühler, A., Peter, D., Flohr, F.B., Zöllner, M.: Point cloud generation with continuous conditioning. In: International Conference on Artificial Intelligence and Statistics, AISTATS 2022, 28-30 March 2022, Virtual Event. pp. 4462–4481 (2022), https://proceedings.mlr.press/v151/triess22a.html
31. Valsesia, D., Fracastoro, G., Magli, E.: Learning localized generative models for 3d point clouds via graph convolution. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019 (2019), https://openreview.net/forum?id=SJeXSo09FQ
32. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. ACM Transactions on Graphics **38**(5), 1–12 (nov 2019). https://doi.org/10.1145/3326362
33. Yu, X., Rao, Y., Wang, Z., Liu, Z., Lu, J., Zhou, J.: Pointr: Diverse point cloud completion with geometry-aware transformers. In: 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021. pp. 12478–12487 (2021). https://doi.org/10.1109/ICCV48922.2021.01227
34. Yu, Z., Zheng, X., Yang, Z., Lu, B., Li, X., Fu, M.: Interaction-temporal GCN: A hybrid deep framework for covid-19 pandemic analysis. IEEE Open Journal of Engineering in Medicine and Biology **2**, 97–103 (2021). https://doi.org/10.1109/ojemb.2021.3063890
35. Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M.: PCN: point completion network. In: 2018 International Conference on 3D Vision, 3DV 2018, Verona, Italy, September 5-8, 2018. pp. 728–737. IEEE Computer Society (2018). https://doi.org/10.1109/3DV.2018.00088
36. Zhao, H., Jiang, L., Jia, J., Torr, P.H.S., Koltun, V.: Point transformer. In: 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021. pp. 16239–16248. IEEE (2021). https://doi.org/10.1109/ICCV48922.2021.01595
37. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 2242–2251 (2017). https://doi.org/10.1109/ICCV.2017.244