This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3069195, IEEE Transactions on Visualization and Computer Graphics

SG-GAN: Adversarial Self-Attention GCN for Point Cloud Topological Parts Generation

Abstract—Point clouds are fundamental in the representation of 3D objects. However, they can also be highly unstructured and irregular. This makes it difficult to directly extend 2D generative models to three-dimensional space. In this paper, we cast the problem of point cloud generation as a topological representation learning problem. In order to capture the representative features of 3D shapes in the latent space, we propose a hierarchical mixture model that integrates self-attention with an inference tree structure for constructing a point cloud generator. Based on this, we design a novel *Generative Adversarial Network* (GAN) architecture that is capable of generating recognizable point clouds in an unsupervised manner. The proposed adversarial framework (SG-GAN) relies on self-attention mechanism and *Graph Convolution Network* (GCN) to hierarchically infer the latent topology of 3D shapes. Embedding and transferring the global topology information in a tree framework allows our model to capture and enhance the structural connectivity. Furthermore, the proposed architecture endows our model with partially generating 3D structures. Finally, we propose two gradient penalty methods to stabilize the training of SG-GAN and overcome the possible mode collapse of GAN networks. To demonstrate the performance of our model, we present both quantitative and qualitative evaluations and show that SG-GAN is more efficient in training and it exceeds the state-of-the-art in 3D point cloud generation.

Index Terms—Generative Adversarial Network, Graph Convolution Network, binary tree, self-attention, 3D shape generation, point cloud learning, gradient penalty.

1 Introduction

The most natural representation of three-dimensional shapes, point clouds can provide an immense amount of geometric and structural flexibility about their underlying representative objects at any scale. Generally, point clouds with complex structures are crucial samples of surfaces in 3D space and have strong adaptability in modeling 3D shapes in a wide range of applications such as object classification, pattern detection, registration, and semantic segmentation. Therefore, enabling generative machine learning models to infer complex 3D shapes is a fundamental problem that is worthy of further study.

Deep neural networks have achieved a high level of performance when faced with structured image data. However, different from 2D images, the distribution of points obtained from directly scanning 3D objects is non-uniform and highly irregular. In addition, point cloud data usually maintains abundant spatial geometric information and complex regional structures. These properties limit the application of typical deep learning methods in processing dense 3D vision data. For instance, directly employing *Convolutional Neural Network* (CNN) to deal with the voxelized point cloud often demand high memory and computational cost in practice.

In order to overcome the shortages of voxel-based methods, Qi et al. [24] combined *Multi-Layer Perceptron* (MLP) and symmetric function to directly process dense 3D point cloud data. Inspired by this work, many feature learning models [18, 25, 29, 34, 38, 39] have been proposed, aiming at capturing local structure information and aggregating better representative features of 3D shapes in latent space. With these advances, deep learning models have greatly boosted the performance of supervised learning tasks.

In comparison with supervised point cloud feature learning, inferring accurate 3D shapes from random raw code in unsupervised manner is still challenging. Although various methods [1, 22, 30, 35, 41] have been proposed to generate point clouds by leveraging deep generative models, the latent topology of point clouds has not been taken into account. This results in a lack of connection between the global features of a point cloud and its parts during the parts inference process.

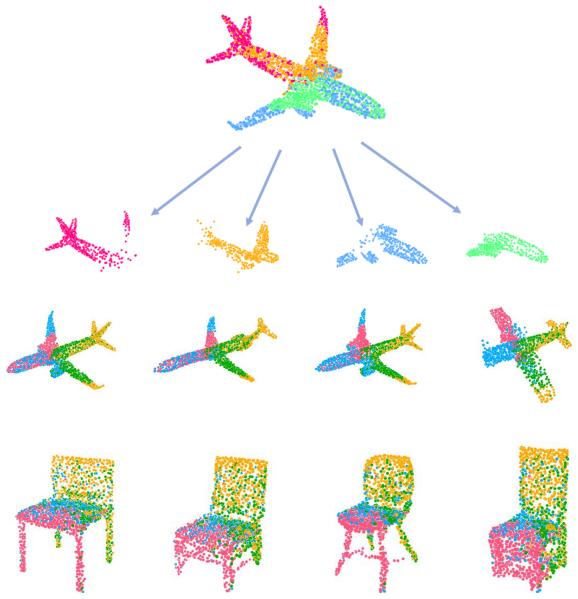


Fig. 1: Partial structures generated by our model. SG-GAN decodes the latent structural information embedded in non-leaf nodes into visible point sets and packs them together in the 3D space. Samples generated by our model are marked as the descendant points of different nodes by varying colors.

The graph convolutional network provides us with an effective tool to generalize latent topology aggregation into

a graph learning problem because the graph is a natural representation of the topological structure. This suggests a novel GAN framework, **S**elf-Attention **G**raph Learning **GAN** (**SG-GAN**). In our generative model, we consider the representative topology of a point cloud in the latent space as a weighted undirected graph.

In the hierarchically learning process, we integrate self-attention and GCN in a tree framework. For a given random code, SG-GAN firstly extends it to several non-leaf nodes embedding partial structure information and then constructs a coarse graph upon them. Also, we propose a graph learning module to aggregate the graphical topology information into further graph inference and dynamically update the latent representation of a point cloud. With self-attention and spectral GCN, our generative model is capable to generate compact and realistic 3D shapes presented by point clouds. Moreover, the tree-based generative architecture facilitates the handling of partially generate point clouds. This property is indicated in Fig. 1, and we mark the 3D points derived from different non-leaf nodes by different colors. The details of our architecture are discussed in Sec. 3.2.

Another challenge we address is the GAN training process. A GAN is an adversarial two-players game. The employed discriminator differentiates real and synthetic data in order to compel the generator to provide more realistic outputs. Early GANs are based on a non-universal assumption that the discriminator is optimally updated at each step, and the objective divergence between real and generated data will converge to a consistent minimum. Despite some specific merits in generating realistic results, training stability and convergence are not guaranteed in these GANs. Taking the original GAN [8] for example, the employed *Jensen–Shannon* (JS) divergence produces non-overlapping training and generative distributions in the latent space, which always results in instability and mode-collapse.

An intuitive improvement [2] replaced JS divergence by Wasserstein divergence to measure the objective loss. This method successfully solved gradient vanishing in GANs training. However, both [1] and its improvement [9] cannot settle non-convergence and unstable training caused by sharp gradients. In practice, these methods do not guarantee the training stability and convergence of our model. To overcome these problems, we propose two different gradient penalty methods in Sec. 3.3. Both endow our model with two advantages: (1) stable training and convergence, (2) avoiding multiple discriminator updates. In this work, we address 3D shape generation as representative topology learning and present a novel adversarial model to generalize latent topology inference into graph learning.

Both quantitative and qualitative results demonstrate that our framework outperforms previous GANs in point cloud generation. We further explore the property of our model in resolving the self-attention score with the local proximity and partial generation. Our contributions are as follows:

- We propose a learning module that combines self-attention and GCN in order to transform latent topology inference into dynamic graph learning;
- We design a novel generative architecture by composing a sequence of the new graph learning units into a tree structure with the goal of generating high-quality 3D point clouds;
- We present two different gradient-based penalization methods to constraint the Wasserstein loss, which benefits our model in training stability and convergence. Furthermore, both of these penalties help SG-GAN to prevent time-consuming multiple discriminator updates;
- We further explore the property of SG-GAN in partial structure generation resulted from the proposed generative architecture.

2 Related Work

2.1 Deep learning on point clouds

Deep learning on point clouds has been effective in a variety of research areas such as 3D classification, object recognition, and scene segmentation. Capturing geometric features from 3D point clouds is especially attracting more and more attention in recent years. In comparison with traditional hand-crafted features, the methods relying on deep learning frameworks are more effective to handle 3D vision data such as point clouds, meshes and volumetric grids.

Because a regular grid is easy to adopt within traditional learning modules like *Convolutional Neural Network* (CNN), early point cloud learning methods usually voxelize a point cloud into sparse volume grids. Motivated by this, Wu et al. [40] proposed a convolutional deep belief network in order to aggregate information from 3D objects. By estimating the probabilistic distribution of variables on 3D voxel grids, this method achieves encouraging performance on 3D object recognition. Similarly, [21] proposed another volume-based architecture that integrates supervised CNN with a grid to learn 3D geometric features. Subsequent point cloud learning research focused on promoting the performance of volumetric processing and increasing the training efficiency. Qi et al. [23] proposed a multi-resolution learning module and demonstrated that introducing multi-orientation pooling into data augmentation is able to significantly improve the performance of volumetric CNNs. Wang et al. [37] utilized a hybrid tree-based algorithm to speed up volume-based point cloud learning. Although these methods improve the performance of voxel-based frameworks to a certain extent, they are unable to scale dense point clouds due to high computation and memory cost. This drawback limits the application of volumetric-based methods in 3D shape learning.

In the pioneering work of PointNet proposed by Qi et al. [24], it was demonstrated that independently modeling each point and aggregating a global feature by integrating MLPs with symmetric function is suitable to deal with unordered point sets. This is regarded as one of the most significant proposals for 3D point cloud learning. Inspired by PointNet, some frameworks, such as [12], borrowed the MLP-based architecture and symmetric pooling function to aggregate descriptive features of 3D point clouds. In spite of this, simply pooling the maximum items of the representation feature ignores the latent information implied within local structural neighborhoods.

2.2 Local geometric features

Despite achieving permutation invariance, PointNet overlooks the local structural information. For better capturing the local geometric information, Qi et al. [25] introduced a hierarchical network that utilizes PointNet recursively on the nested parts of the input point cloud. This iterative process dramatically improves the performance of PointNet++. Nevertheless, repeatedly applying PointNet complicates the learning architecture and results in low training efficiency.

Subsequently, some local-based approaches were proposed to take both accuracy and efficiency into account. These models can be categorized into the following categories: MLP-based, CNN-based, and GNN-based. All of them focus on exploiting local geometric information.

On the basis of PointNet++, Zhao et al. [43] presented an interesting local learning network, called Pointweb, that employs adaptive feature adjustment to take advantage of the context of local geometry. Different from this, Lin et al. [19] integrated a lookup table with PointNet to accelerate the integration of point clouds with the local geometry. Yang et al. [42] enriched the representation of 3D points by concatenating relative spatial locations with respect to neighbors into the absolute point positions.

Another intuitive method to aggregate local features is applying redefined convolutional kernel on point clouds. As a typical CNN-based pipeline, the approach presented by Li et al. [18] captures neighborhood relations of a point cloud using hierarchical convolution kernels. Relying on training a transformation matrix to match the convolution kernel with discrete 3D points, this framework makes typical convolutional operator can be applied on point clouds. Differ from discrete convolution, Wu et al. [39] defined the continuous convolution kernel as a set of nonlinear functions by using Monte Carlo approximation. Similarly, Hermosilla et al. [11] treated convolution as a Monte Carlo estimation process, but this approach is based on Poisson disk sampling. Later, Lei et al. [16] propose a CNN-based model that associates spherical convolution with Octree structure aiming at efficiently aggregate informative regional features. To be resistant to perturbation and rotation, Rao et al. [27] approximated the spherical convolution kernel by regular icosahedral lattices. Liu et al. [20] exploited the contextual information in local geometry by redefining convolution kernel as single-layer perceptron.

Different from MLPs and CNNs, graph-based networks generally convert the neighborhood of each 3D point to a local graph and identify this point as a graph vertex. Leveraging on *graph neural network* (GNN), Qi et al. [26] achieved RGBD-based 3D semantic segmentation by associating the representative features aggregated from 2D depth images with graph nodes. According to the best of our knowledge, Simonovsky and Komodakis [31] designed the first *graph convolution networks* (GCN) architecture to deal with point clouds, which considers the discrete 3D points as graph nodes and connects them via constructing a directed graph. In [28], the local structure of a point cloud is interpreted by defining a kernel correlation measure function. Additionally, in this model, a graph-based pooling method is combined to improve semantic parsing performance. After this, Wang et al. [38] proposed a dynamic graph learning framework with a simple module, named EdgeConv, aiming at dynamically extracting informative features from 3D neighboring points. Chen et al. [6] presented a hierarchical clustering structure and designed a rotation-invariant graph learning unit to aggregate rotation-invariant features from point neighbors.

Unlike spatial GNN models, spectral-based GNNs rely on multiplying graph signals with eigenvectors of the corresponding Laplacian matrix. To accelerate the computation process of GNNs, Defferrard et al. [7] approximated the spectral filters as Chebyshev polynomials. With the motivation of this work, Wang et al. [36] utilized spectral graph convolution to learn local point set features. Te et al. [33] employed spectral graph filter to adaptively captures the structure of dynamic graphs.

2.3 Generative models for point clouds

With the success of deep learning, generative models are becoming an essential research topic in 3D vision. The goal of 3D generation models is to learn the shape information from the training set and reconstruct realistic results. Overall, the generative models are often categorized into the following categories: (1) generative adversarial network, (2) flow-based generative model and (3) *Variational Auto-Encoder* (VAE).

In this section, we mainly focus on reviewing GANs. A GAN framework consists of two modules: a generator and a discriminator, which was first presented by Goodfellow et al. [8]. Generally, optimizing a GAN framework is considered as an adversarial game between generator and discriminator. In other words, these two units contest each other by estimating whether the synthesized results are real or fake. Although the original GAN model provides an interpretable theory for generating arbitrary realistic results, the training of GANs is not stable and often encounters various problems, such as the mode-collapse and non-convergence. To maintain the training stability of GANs, Arjovsky et al. [2] replaced the JS divergence by an approximation of Wasserstein divergence in order to overcome the mode-collapse problem. Based on this work, a penalty improvement of WGAN was introduced by Gulrajani et al. [9].

As a pioneer of point cloud generation, Achlioptas et al. Achlioptas et al. [1] proposed a set of generative models including raw point cloud GAN and latent-space GAN to generate multi-class point clouds. Due to their fully connected layers, these models are not good at maintaining informative structures. Li et al. [17] incorporated a hierarchical Bayesian model into the GAN architecture to reconstruct complex point clouds. The likelihood-based method limits its performance on reconstructing thin and local details. To better make use of local information, Valsesia et al. [35] applied graph convolution in GAN architecture. However, the results generated by this approach are always not compact. On the basis of this work, Shu et al. [30] further integrated graph convolution with a tree structure to generate 3D shapes.

The generative models based on other architectures, such as AE and VAE have also been attempted. In addition, various learning approaches (e.g. ,capsule networks, self-attention and flowing) are introduced in generative models.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3069195, IEEE Transactions on Visualization and Computer Graphics

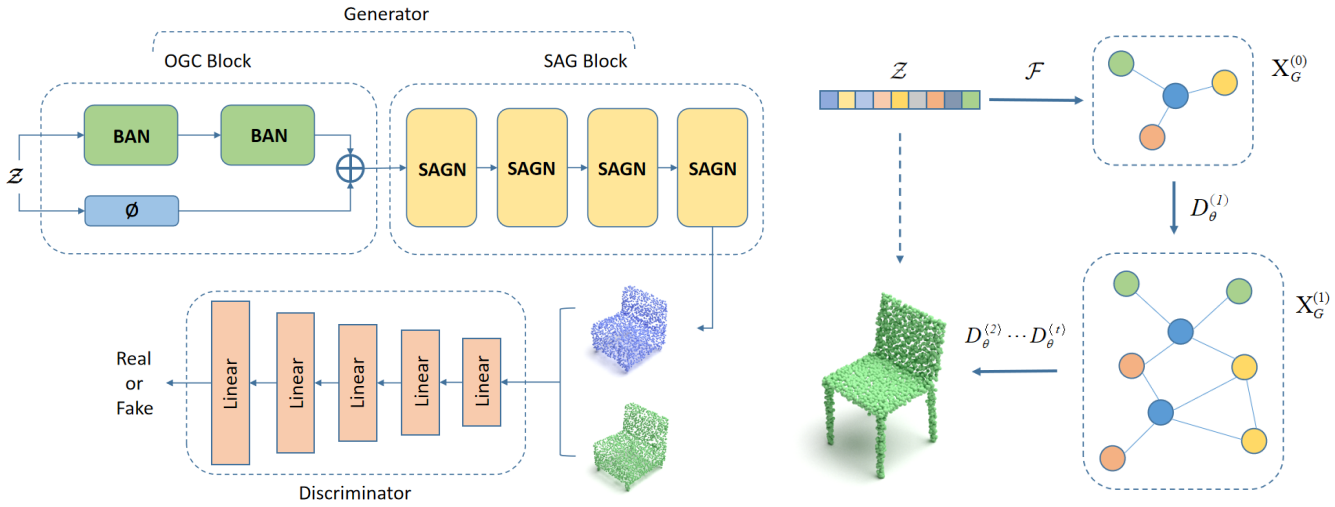


Fig. 2: Architecture and the basic mechanism of SG-GAN. Our generative framework includes two learning blocks: OGC and SAG. In OGC block, to initially frame the graphical topology, we employ two BAN modules to up-sample the random generated $\mathcal{Z}$, and adopt a function ϕ to adjust the representative information inherited by children nodes in the initial graph. After this, we integrate four SAGN modules to constitute the SAG block in order to dynamically extend and update the latent graph. The right subfigure illustrates the basic mechanism of the proposed framework.

Zhao et al. [44] used capsule network to preserve spatial arrangements of sparse point clouds. To complete partial input point clouds, Sun et al. [32] adopted self-attention to capture long-range dependencies in point set. Yang et al. [41] combined continuous normalizing flows in a variation auto-encoder to implement point cloud generation. Recently, Pumarola et al. [22] proposed a framework with two parallel flow branches to achieve multi-modality transfer (2D images and 3D point clouds).

None of the current generative models consider point cloud generation as a global topology inference. This makes them cannot recover the global and connectivity information in generative learning. In this paper, we present a novel GAN framework, SG-GAN, which leverages on topological graph learning and tree structure to generate 3D shapes with accurate and compact geometric structures. The details of our approach are discussed in Sec. 3. Meanwhile, the experimental results are illustrated in Sec. 4.

3 Methodology

3.1 Overview

In the methodology section, we begin with a brief introduction to the formulation and the overall implementation of SG-GAN. A topological graph maintains the intrinsic structure of a 3D shape presented by a point cloud. To define topology-aware representations for point clouds, we assume that the topology of a 3D point cloud can be approximated by a sequence of representative graphs in a tree structure. With this assumption, we transform point cloud generation into latent graph unrolling.

Assuming a representative graph $G = (\mathbf{V}, E)$ at the non-leaf level where $\mathbf{V}$ and E denote the vertices and edges respectively. Each vertex of the graph contains the structural information of a 3D shape, while the graph topology maintains the partial connection and global structure information. We define a decoder $D_\theta : \mathbf{X}_G \rightarrow \mathcal{P}$ that maps latent graph information $\mathbf{X}_G$ into point set $\mathcal{P}$ in $R^{N\times 3}$. Theoretically, D_θ can be any non-linear function parameterized by θ.

We construct a binary tree to design the basic architecture of SG-GAN, because the tree-based framework provides a natural way to transfer information from parent nodes to children. However, relying only on a binary tree makes the generative model less effective in embedding the global topological information into shape formation. Simply spreading the initial latent information to leaves would result in unwanted deformations and fragmentation in final shape generation. Moreover, descending along the tree, the structural representation capability of each non-leaf node is decreasing.

Therefore, we combine self-attention and GCN with binary tree aiming at overcoming these problems. Specifically, we first define a graph initialization function $\mathcal{F} : \mathcal{Z} \rightarrow \mathbf{X}_G^{(0)}$ mapping the input latent code $\mathcal{Z} \in R^{\mathcal{Z}}$ into an initial graph with n nodes. Then, we apply D_θ to decode the initialized graph. At the decoding stage of D_θ, we gradually extend the vertex set of the latent graph by branching and dynamically aggregate graph topology features to correct graph information in latent space.

To take advantage of graph topology in 3D point cloud generation, we propose a graph learning module integrated with self-attention, named as *self-attention graph learning network* (SAGN), to construct the tree-based generative model. In our learning module, the graph topology information g_G is aggregated by combining spectral GCN and self-attention masking. The main motivation for employing spectral GCN in our module is that the spectral representations of graph signals depend on the graph topology. This property makes spectral GCN capable to extract intrinsic graph topology information in latent space. Furthermore, we present a self-attention based mechanism to weightedly aggregate the global topology information learned by spectral GCN. Since GCN has the potential to over smooth graph nodes, we additionally feed $\mathbf{X}$, which is the node information of the latent graph, into the learning process.

Fig. 2 shows an illustration of our architecture. In the proposed framework, we design two learning blocks to play the roles of $\mathcal{F}$ and D_θ. Particularly, we first map the randomly generated information into an initial graph

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3069195, IEEE Transactions on Visualization and Computer Graphics

using *original graph construction* (OGC) block constituted by two *branching and adjustment network* (BAN). Afterward, we apply *self-attention graph learning* (SAG) block (with four SAGN modules) to decode the graph into a point cloud. Finally, same as other GANs, the generated result is input into the discriminator to judge whether it is real or fake.

Another challenge we address in this paper is stable GAN training. Although WGAN [2] solved the mode-collapse problem resulting from gradient vanishing, it cannot guarantee the convergence and the stability of GAN training, especially when sharp gradients occur. Thus, we propose two simple and effective penalties to constraint WGAN loss for maintaining the training stability and convergence of SG-GAN. Furthermore, both of these two strategies improve the performance of our model on generation diversity.

3.2 Model

3.2.1 Graph initialization

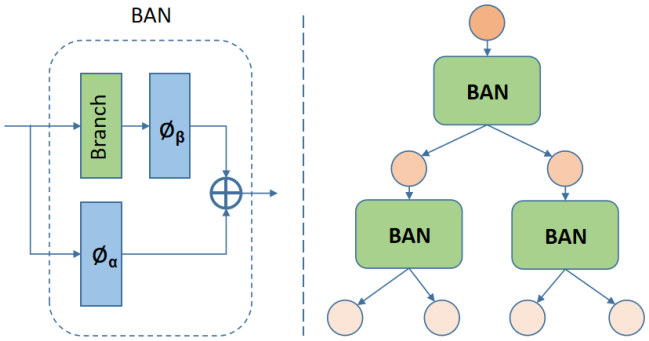


Fig. 3: BAN architecture. In BAN, we combine a branching module with two ϕ units in order to generate informative children nodes. The right figure indicates how we use BAN modules to construct the feature vertices of the initial graph.

Given a high-dimensional code $\mathcal{Z}$, we construct an initial graph $G = (\mathbf{V}, E)$ from it, where all the graphical nodes $\mathbf{V}$ inherit information from $\mathcal{Z}$. Instead of directly mapping the latent input into a graph, we hierarchically construct the initial graph. In other words, we design OGC block that consists multiple BAN modules to achieve the mapping function defined as $\mathcal{F}$.

The architecture of BAN is shown in Fig. 3, and the operation of this learning module is defined as:

$$\mathbf{X}^{(l+1)} = \phi_\beta(\mathbf{W}_b^{(l+1)} \cdot \mathbf{X}^{(l)}) \oplus \phi_\alpha(\mathbf{X}^{(l)}) \tag{1}$$

where both ϕ_α and ϕ_β are the weighting function applied to adjust the information inherited from the input, while $\mathbf{W}_b$ is a learnable matrix used to branch the input code. In our architecture, all ϕ functions are realized by applying MLP. Notably, $\mathbf{X}^{(l)}$ is the features learned at level l and $\mathbf{X}^{(0)}$ is $\mathcal{Z}$. The right figure shown in Fig. 3 demonstrates how we combine BAN units to generate the nodes demanded by the initial graph. After node generation, we further embed the modified information of $\mathcal{Z}$ into these nodes and assemble them to construct the initial graph.

3.2.2 Learning graph topology

A. Spectral graph convolution

Typically, the spectral representation of graph signals encodes important graph topological features. Inspired by this, we employ spectral GCN to aggregate intrinsic topology information from the graph. As a *Graph Signal Processing* (GSP) based method, spectral graph convolution first projects graph signals onto the space determined by the eigenvectors $\{\mathbf{u}_i\}_{i=1}^{|\mathbf{V}|}$ decomposed from the graph Laplacian matrix. Since G is undirected, its Laplacian matrix is defined as: $\mathbf{L}$=$\mathbf{D}$-$\mathbf{A}$, where $\mathbf{D}$ is the degree matrix and $\mathbf{A}$ denotes the adjacency matrix. Then, this method convolves the vertex embedding in the spectral domain. Finally, it transforms the convolved results back to the spatial domain.

Relying on *Graph Fourier Transformation* (GFT), we can simply extend convolution to the graphs with irregular and unordered vertices. However, the eigen-decomposition $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\mathbf{T}$ of this naive way is time-consuming. Therefore, we need to avoid spectral decomposition of $\mathbf{L}$. Following the method proposed in [4], we first estimate the normalized graph Laplacian $\mathbf{L}$ as:

$$\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \tag{2}$$

Then, we define spectral convolution as filtering a signal $\mathbf{x} \in R^N$ in the Fourier domain:

$$f_\theta * \mathbf{x} = \mathbf{U}\left[f_\theta(\Lambda)\right]\mathbf{U}^\mathbf{T}\mathbf{x} \tag{3}$$

where $\mathbf{U}$ and Λ denote the eigenvector matrix of the normalized graph Laplacian $\mathbf{L}$ and the diagonal eigenvalues matrix respectively. In this equation, f_θ presents the filter with hyper-parameter $\theta \in R^N$.

This approach provides a computationally systematic way to process the graph in the Fourier domain. Nevertheless, it is still difficult to apply spectral convolution in practice, because eigen-decomposition is not evaded, and the spectral filter is not spatially localized.

For solving these problems, [10] proposed a method to approximate the spectral filtering by Chebyshev polynomials as:

$$f_\theta * \mathbf{x} = \sum_{k=0}^{K} \theta_k P_k\left(\frac{2}{\lambda_{max}}\mathbf{L} - \mathbf{I}_N\right)\mathbf{x} \tag{4}$$

where λ_{max} denotes the maximum eigenvalue of $\mathbf{L}$, and $\mathbf{P}_k$ represents the kth term of the K-order Chebyshev polynomial. This method successfully bypasses the computation of the Laplacian eigenvectors.

To smoothly convolve the graph, GCN will force $\frac{2}{\lambda_{max}}\mathbf{L}$ to be closed to $\mathbf{I}_N$. Following [13], we approximate λ_{max} to 2 since the neural network parameters will adapt to this change during training. Therefore, Equation. 4 is simplified as:

$$f_\theta * \mathbf{x} = \theta_0\mathbf{x} - \theta_1\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{x} \tag{5}$$

Thereafter, we obtain a generalized expression of spectral convolution defined as:

$$\mathbf{Z}_G = \mathbf{D}_g^{-\frac{1}{2}}\mathbf{A}_g\mathbf{D}_g^{-\frac{1}{2}}\mathbf{X}\Theta \tag{6}$$

with $\mathbf{A}_g = \mathbf{A} + \mathbf{I}_N$ and $\mathbf{D}_g$ is the degree matrix of $\mathbf{A}_g$. In addition, $\mathbf{X} \in R^{N\times C}$ is the graph signal (C is the number of input channels), and $\mathbf{Z}_G$ presents the result estimated by spectral GCN.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3069195, IEEE Transactions on Visualization and Computer Graphics

B. Self-attention masking

Different from traditional convolution methods, the self-attention mechanism captures the weighted responses from all feature locations, while the weights are estimated with a small computation cost. This mechanism makes learning modules focus more on important features.

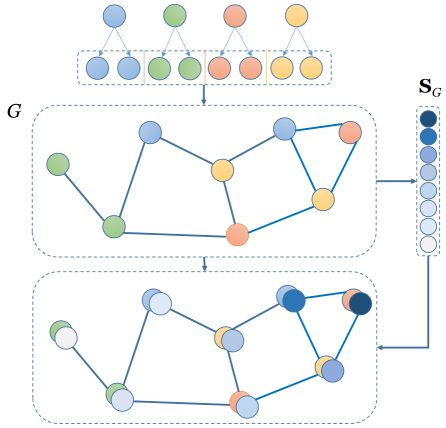


Fig. 4: Self-attention mechanism. This figure visually illustrates the scoring and masking operations of our model. At first, we learn the self-attention scores $\mathbf{S}_G$ that determines the importance of each node. Then we mask the corresponding graph with $\mathbf{S}_G$.

Inspired by this, we employ a scoring method to measure the weighted response from each node on the constructed graph. The key point of this scoring operation is applying a GCN layer to estimate the self-attention score of each graph node. Following Equation. 6, we define the self-attention scores as:

$$\mathbf{S}_G = \sigma(\mathrm{GCN}(\mathbf{X}_G)) = \sigma(\mathrm{GCN}(\mathbf{X}, \mathbf{A})) = \sigma(\mathbf{Z}_G) \quad (7)$$

where σ presents the activation function $tanh$. Similar to the method proposed in [15], the estimated scores are permuted from largest to smallest, which represents the weighting of each node in the global topology information. After this step, we mask the input graph node features with the permuted self-attention scores as following:

$$\widetilde{\mathbf{X}} = \mathbf{X}_r \odot \mathrm{rank}(\mathbf{S}_G),\ \mathbf{X}_r = \mathrm{rank}(\mathbf{X}) \quad (8)$$

By this masking operation, we enhance the weights of the dominant graph features that usually contain the latent information reflecting representative geometry structures.

C. Dynamic graph learning

Based on GCN and self-attention scoring, we design a module SAGN to aggregate topology information from the graph and employ it to update the extended node set for constructing a new graph. To appropriately merge the global topology features with the structural information maintained in non-leaf nodes, we further adopt MLPs to dynamically adjust the weights of global information and the discrete message inherited from parent nodes. With this module, our model gradually learns and updates the graph for approximating the representation of a point cloud in latent space.

In the proposed learning module, we estimate the global topology information by a mixed aggregation step which combines mean pooling and max pooling:

$$g_G^{(l)} = [\mathrm{MAX}(\phi(\widetilde{\mathbf{X}}^{(l)})) \parallel \mathrm{MEAN}(\phi(\widetilde{\mathbf{X}}^{(l)}))] \quad (9)$$

To generate children nodes and adaptively transfer useful messages to them, we integrate branching operation with the MLP layer. The branching scheme employed by us is similar to the method applied in generating 2D images. Differently, we incorporate two MLP units in tandem to strengthen the branching ability of our model. The branched node features $\mathbf{X}_b^{(l+1)}$ is defined as:

$$\mathbf{X}_b^{(l+1)} = \phi_1^{(l+1)}(\mathbf{W_b}^{(l+1)} \cdot \phi_2^{(l+1)}(\mathbf{X}^{(l)})) \quad (10)$$

Notably, we re-permute $\mathbf{X}^{(l)}$ following the sorted order of self-attention scores before inputting it into the following MLP layer. Then, we form the new node features $\mathbf{X}^{(l+1)}$ as:

$$\mathbf{X}^{(l+1)} = \sigma(\mathbf{X}_b^{(l+1)} \oplus \phi_3(g_G)^{(l)} \oplus \phi_4(\widetilde{\mathbf{X}}^{(l)})) \quad (11)$$

In the above equation, the activation function σ is $\mathrm{LeakyReLU}$. The architecture of the proposed graph learning module is depicted in Fig. 5. The proposed self-attention mechanism makes SAGN able to effectively aggregate global topology information within a graph. Our model embeds the global topology information into the partial representation message inherited from parents. In addition, we employ two MLP units ϕ_3 and ϕ_4 for weightedly merging the global and partial information. By combining BAN and SAGN in a tree architecture, our model is capable to generate realistic point clouds with compact structures.

3.3 Training objective

As a powerful generative model, the goal of GANs is to reach Nash-equilibrium where the synthesized results are closed to real distributions. Generally, GANs are defined as a min-max game between two players. The discriminator tries to distinguish real data and fake data, while the generator tries to produce realistic results. Nevertheless, there are some persistent problems of GANs that need to be solved.

Although GANs have the potential to generate realistic distributions, the alternating gradient updates procedure is highly unstable and often causes mode-collapse or non-convergence in training. More seriously, normal gradient descent optimizations cannot guarantee its convergence. An important reason for these problems is the assumption that the discriminator is ideally optimized at each training step and the divergence between real and generated distributions is consistently minimized. In practice, this assumption is not true for most cases of GANs. In addition, GANs often converge to an unexpected local equilibrium, despite decreasing the learning rate would improve this.

To alleviate the instability problem of GANs, some approaches were proposed. WGAN [2] replaces the Jensen-Shannon divergence with the Wasserstein divergence in order to improve the continuity of loss function. WGAN-GP [9] further improves this approach by adding a penalty regularization. Different from these, DRGAN [14] proposes that local equilibria is often accompanied by sharp discriminator gradients. However, these approaches still cannot

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3069195, IEEE Transactions on Visualization and Computer Graphics

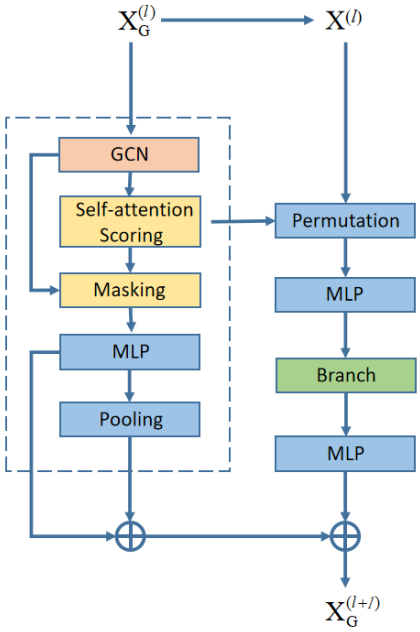


Fig. 5: SAGN architecture. In this module, we integrate GCN with self-attention masking to aggregate global graph information. Then, we employ a mixed pooling and MLPs to fuse it with the features inherited from parent nodes in order to enhance the structural representation in the generated results.

stabilize the training process of GANs when the data distribution is concentrated along with some low-dimensional manifolds.

According to our observation, the instability of our model is mainly caused by sharp gradients of the discriminator: the sharp gradients always enforce the discriminator to deviate from the equilibrium. Therefore, we utilize regularization methods to penalize the discriminator gradients in order to eliminate unexpected sharp cases. In our model, the loss of discriminator is based on the WGAN metric [2]:

$$\mathbf{E}_{x\sim P_g}[D(G(x))] - \mathbf{E}_{x\sim P_r}[D(x)] \tag{12}$$

With this, we propose an effective way to penalize the discriminator gradient on the real data. This penalization method performs well in training, especially when the discriminator tends to create sharp gradient. This penalty term is defined as:

$$\lambda_p[\mathbf{E}_{x\sim P_r}[\|\nabla D(x)\|^2] \tag{13}$$

where λ_p is the penalty parameter. We also present another similar regularization term defined as:

$$\lambda_p[\mathbf{E}_{x\sim P_g}[\|\nabla D(x)\|^2] \tag{14}$$

where this term penalizes the discriminator gradient on the generated distributions rather than the real samples. The regularization methods proposed in this paper prevent the discriminator from equilibrium deviation. They all bring our framework another benefit that our model does not need to multiply update the discriminator in each training iteration. Additionally, the generator loss adopted in our model is the same as the one presented in [2].

4 Experiment and Evaluation

4.1 Evaluation metrics

Same as r-GAN [1], Localized-GAN [35] and Tree-GAN [30], we quantitatively evaluate the performance of our model on point cloud generation in terms of *Jensen-Shannon divergence* (JSD), *coverage* (COV) and *minimum matching distance* (MMD). To estimate the similarity between ground truth and generated results, we utilize two different distance measurements: *Chamfer distance* (CD) and the *earth mover's distance* (EMD). Both of them are commonly applied in 3D evaluation metrics. The goals of applying JSD, COV and MMD are summarized as follows:

- *Jensen-Shannon Divergence (JSD)*: JSD is employed to evaluate the similarity between the reconstructed 3D shapes and their corresponding benchmarks. This metric is based on estimating the marginal distributions of generated results and benchmarks.
- *Coverage (COV)*: COV measures authenticity and the diversity of the generated results by estimating the matched fraction between ground truth and generated point clouds.
- *Minimum Matching Distance (MMD)*: MMD is commonly applied to measure the fidelity of the generated point clouds with respect to the ground truth by evaluating the average matching distance between real and generated data.

4.2 Implementation details

Layer	BAN1	BAN2	SAGN1	SAGN2	SAGN3	SAGN4
Branch degree	2	2	2	2	2	64
SA	-	-	✓	✓	✓	✓

TABLE 1: Implementation details

The datasets used in our experiments are ShapeNet [5] and D-FAUST [3]. We use 16 shape categories of ShapeNet to implement the comparison and evaluation of rigid object generation. The 3D models of ShapeNet are constructed by uniformly sampling 3D surfaces and both of them are represented as point clouds. To further evaluate the performance of SG-GAN on learning non-rigid bodies, we additionally employ D-FAUST dataset in our experiment. The data of D-FAUST is 4D scan of human motion, and we sample the mesh data using FPS algorithm in order to obtain the point clouds of non-rigid human bodies.

In our model, Adam optimizer is chosen to train the generator and discriminator, and the learning rate is 0.5^{-4}. Meanwhile, the betas coefficients are set to 0 and 0.99. We set the batch size and the dimension of the initial latent code to 10 and 96 respectively. The input code is sampled from the normal distribution. Although our GAN framework is updated in an alternative process, the generator and discriminator only update one time in each iteration. This is different to other Wasserstein-based GANs [2, 9, 30]. The implementation details of SG-GAN are illustrated in Table. 1, where "SA" denotes self-attention based module.

Similar to [1], our discriminator consists of five MLP layers $[3, 64, 128, 256, 512, 1024]$ and three fully connected layers $[1024, 256, 64]$. The point cloud generated by our model consists 2048 3D points. The GPU applied in our model training is one NVIDIA 2080TI.

4.3 Qualitative and quantitative evaluation

4.3.1 Rigid object generation

In this section, we qualitatively evaluate the performance of our adversarial framework on point cloud generation by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3069195, IEEE Transactions on Visualization and Computer Graphics

Fig. 6: Comparison of generated results and ground truth. We visually compare the generated results (blue) with their corresponding benchmarks (pink) to demonstrate the capability of our model in generating realistic 3D shapes.

visually painting the generated results. In Fig. 6, we provide the ground truth samples corresponding to the generated results as comparison reference. In this experiment, we inspect the generated point clouds of four categories: airplane, pistol, chair and table. It is clear that our model successfully reconstructs the 3D shapes, even the geometric structures are relatively complex, non-convex, and multi–part objects. These visual results demonstrate that our model is able to reconstruct most essential geometric details. To indicate the generation diversity of SG-GAN, we present the generated results of all 16 categories in Fig. 7.

To further illustrate the effectiveness of the proposed model, we conduct metric evaluation with respect to five commonly applied performance measures: JSD, MMD-CD, MMD-EMD, COV-CD and COV-EMD, and compare the quantitative results provided by our model with other state-of-the-art approaches. In comparison with r-GAN [1], Localized-GAN [35] and Tree-GAN [30], SG-GAN is very competitive in the task of multi-class point cloud generation. Notably, we directly use the official experimental results published by the authors in our quantitative comparison. As shown in Table. 2, our model outperforms all the compared frameworks in both evaluation metrics, and this is especially significant with respect to MMD-EMD.

Category	Model	JSD (↓)	MMD (↓)		COV (↑)	
			CD	EMD	CD	EMD
Chair	r-GAN(dense)⋆	0.238	0.0029	0.136	33	13
	r-GAN(conv)⋆	0.517	0.0030	0.223	23	4
	V-GAN (no up)⋆	0.119	0.0033	0.104	26	20
	V-GAN (up)⋆	0.100	0.0029	0.097	30	26
	Tree-GAN⋆	0.119	0.0016	0.101	58	30
	Ours (rp)	**0.041**	**0.0011**	**0.018**	**78**	**33**
	Ours (gp)	0.067	0.0012	0.019	70	19
Airplane	r-GAN(dense)⋆	0.182	0.0009	0.094	31	9
	r-GAN(conv)⋆	0.350	0.0008	0.101	26	7
	V-GAN (no up)⋆	0.164	0.0010	0.102	24	13
	V-GAN (up)⋆	0.083	0.0008	0.071	31	14
	Tree-GAN⋆	0.097	0.0004	0.068	61	20
	Ours (rp)	0.113	**0.0004**	0.011	69	15
	Ours (gp)	**0.062**	**0.0004**	**0.008**	**72**	**41**
Table	r-GAN(dense)⋆	0.217	0.0031	0.139	33	15
	r-GAN(conv)⋆	0.359	0.0031	0.247	29	4
	V-GAN (no up)⋆	0.171	0.0045	0.123	24	18
	V-GAN (up)⋆	0.148	0.0035	0.131	36	29
	Ours (rp)	**0.045**	**0.0013**	0.022	**61**	22
	Ours (gp)	0.061	0.0016	**0.018**	49	26
All (16)	r-GAN(dense)⋆	0.171	0.0021	0.155	58	29
	Tree-GAN⋆	**0.105**	0.0018	0.107	66	**39**
	Ours (rp)	0.117	**0.0008**	0.022	**77**	34
	Ours (gp)	0.131	0.0009	**0.021**	71	36

TABLE 2: Metric evaluation. In this table, we compare the performance of different generative models in terms of JSD, MMD-CD, MMD-EMD, COV-CD, and COV-EMD. We directly use the results presented in [35] [30] for models with ⋆. Same as [30], we employ red and blue to highlight the best and the second-best results.

In accordance with the metric results, we find out that penalizing the real data promotes the model capability in generating the 3D shapes with thin or hollow structures such as chairs. Meanwhile, penalizing the generated data is more appropriate in reconstructing the 3D shapes with smooth transitions and connections such as airplanes. We infer that the smoothness and coherence of manifolds would influence the performance of different penalizing methods. Considering the sharp gradients encountered in practical training, the discriminator is more potential to engender drastic changes when assessing the shapes with slender structures.

4.3.2 *Non-rigid body generation*

Besides qualitative and quantitative comparison of rigid generation, we experiment SG-GAN with non-rigid body data of D-FAUST [3]. Same as [1], we integrate our GAN model with an encoder and retrain the auto-encoding model. With the additional encoder, the input point clouds presenting human motions (with 2048 points) are aggregated into the feature vectors with 96 dimensions. Then, we apply SG-GAN to reconstruct the human motions presented by 2048 points.

Specifically, we split the selected subsets of D-FAUST into train (80%), validation (10%) and test (10%). In addition, we use a simplified PointNet as the encoder and directly combine it with SG-GAN to construct the AE pipeline. All the results plotted in Fig 11 are generated using the test

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3069195, IEEE Transactions on Visualization and Computer Graphics

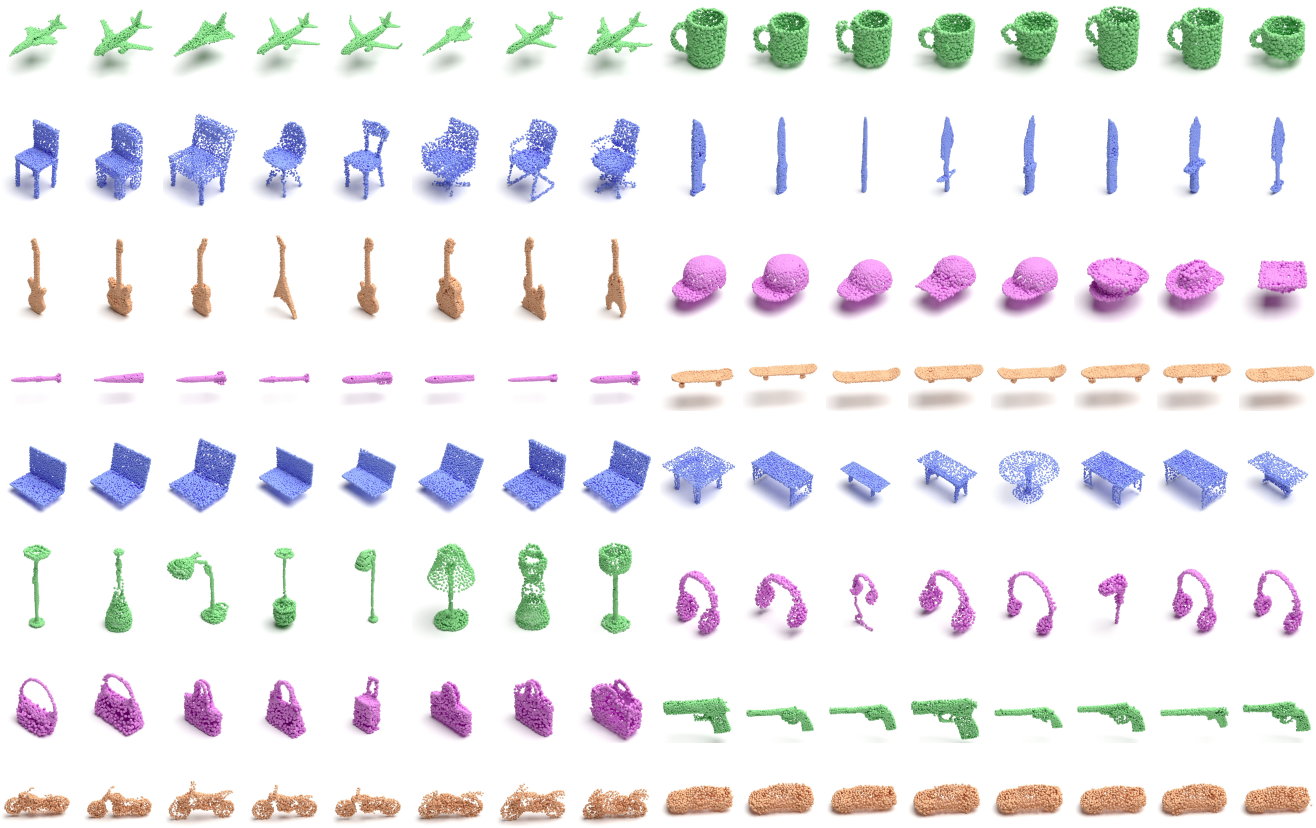


Fig. 7: Generated point clouds. These synthesized 3D point clouds demonstrate the performance of our model in generating 3D shapes. We testify the proposed framework using 16 different shape categories in ShapeNet.

data. These visual results demonstrate the effectiveness of our framework in diverse non-rigid generation.

4.3.3 *Training efficiency*

We provide the training efficiency evaluation of our model in Table. 3. The results prove that the proposed penalization methods give SG-GAN an obvious advantage in training efficiency by avoiding multiple discriminator updates.

Model	Update (g)	Update (d)	Per batch
our model (WGAN loss)	1	5	0.55s
our model (penalty loss)	1	1	**0.19**s
Tree-GAN	1	5	0.41s

TABLE 3: Training efficiency. This evaluation demonstrates that the proposed gradient penalties significantly improve the training efficiency of SG-GAN. In this experiment, we use the same batch size for all models to obtain a fair comparison.

4.4 Ablation study

In this section, we provide the ablation studies of SG-GAN including interpolation, self-attention evaluation and gradient penalty assessment. We inspect the effects of each module of SG-GAN in improving point cloud generation.

4.4.1 *Interpolation*

Interpolation is a common way to testify the latent space learning capability of a framework. The latent representations learned by an effective inference model should be able to provide meaningful interpolation between different

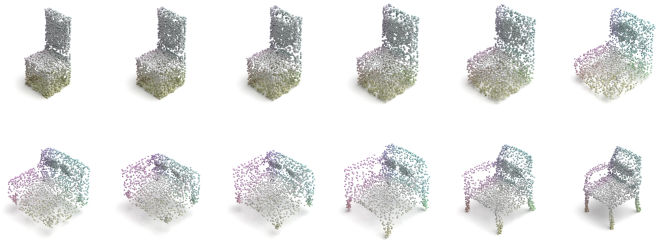


Fig. 8: Interpolation. The realistic and gradually changed interpolations verify the capability of our model in learning latent information. While the left-most and the right-most chairs are generated from z_1 and z_2.

3D structures. The gradually changed point clouds shown in Fig. 8 demonstrate the effectiveness of the proposed model in latent space interpolation learning. It is obvious that the intra-class interpolation presented in this figure is smooth, which further illustrates that our model is good at portraying varied details.

Particularly, we interpolate two shapes in the same category (chair) to test whether our model is able to generate the point clouds with gradual changes. The interpolation parameter is linearly set to weigh the contributions from two latent codes. We uniformly select 12 interpolation parameters: $\omega = [\omega_1, ..., \omega_{12}]$. Meanwhile, the point clouds presenting two initial 3D shapes are generated from the latent vectors z_1 and z_2 respectively.

4.4.2 *SAGN evaluation*

In Fig. 9, we evaluate the impact of the proposed self-attention mechanism in 3D shape synthesis. We quantitatively compare the metric results of the models with and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3069195, IEEE Transactions on Visualization and Computer Graphics

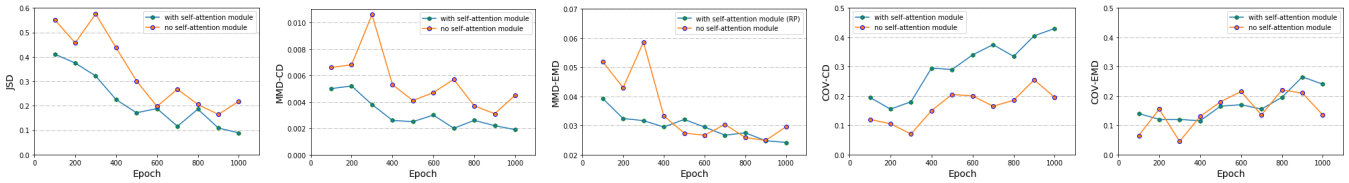


Fig. 9: Ablation study of our self-attention mechanism. SG-GAN outperforms the model without self-attention mechanism in all evaluation metrics, and the comparative advantages are obviously expanded in 1000 epochs.

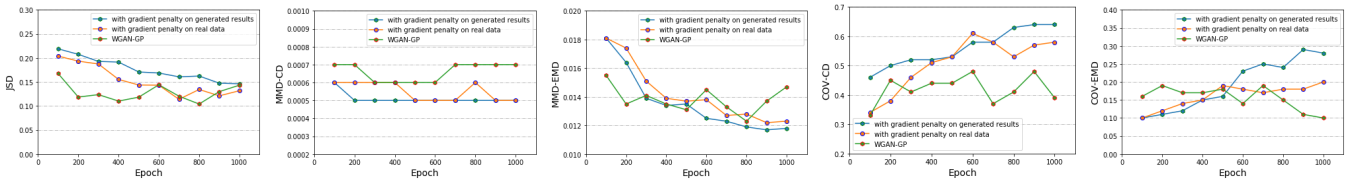


Fig. 10: Ablation study of different loss functions. The JSD and COV curves of the model with WGAN loss fluctuate over time, while the counterparts provided by SG-GAN with different gradient penalties are decreased or increased stably. This illustrates that penalizing sharp gradients promote generation diversity of SG-GAN and help our model overcome potential mode-collapse.

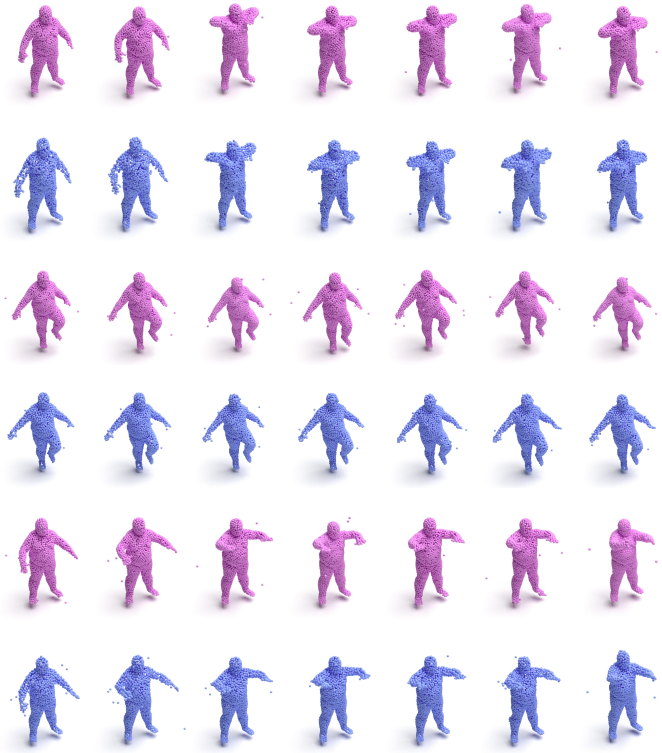


Fig. 11: Non-rigid body generation. In this figure, we present the non-rigid human motion results (blue) generated by our AE pipeline and visually compare them with the ground truth (pink). These results demonstrate the effectiveness of our model in learning non-rigid structures.

without SAGN modules. To obtain the comparing targets, we remove SAGN modules from SG-GAN framework and replace them with a sequence of simplified units. In these units, we only maintain the branching operation. Notably, the branching degrees of the replaced units are the same as SAGN modules.

Same as the metric evaluation applied in qualitative and quantitative comparison, we employ JSD, MMD and COV to study the effect of SAGN in 3D point cloud generation. The experimental results shown in Fig. 9 indicate that our self-attention based module significantly improves the performance of SG-GAN on generating diverse and realistic

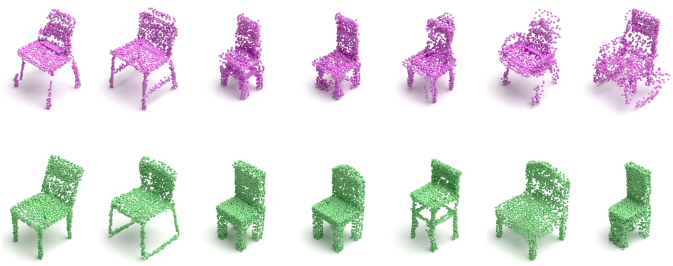


Fig. 12: Point clouds generated by the models with and without self-attention module. In comparison with the results generated by SG-GAN, the 3D shapes synthesized by the model removing our self-attention modules are seriously deformed.

point clouds. Moreover, we provide some visual results in Fig. 12 in order to further demonstrate the improvements brought by the self-attention mechanism. Comparing with the point clouds generated by the model without our self-attention based modules, the results of SG-GAN recover most descriptive details and remain compact structures.

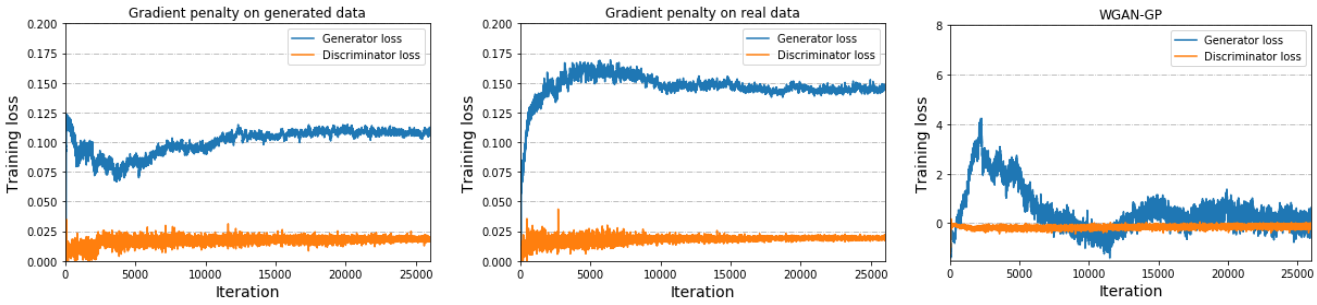
4.4.3 *Gradient penalty*

Different from WGAN [2] and WGAN-GP [9], we propose two penalization methods to improve the convergence and training stability of our model. In accordance with [14], model collapse is always accompanied by the sharp gradients generated in the discriminator. Theoretically, the sharp gradients enforce the discriminator to deviate from the Nash-equilibrium which plays an essential role in the convergence and stable training of GANs, and our observation gives weight to this.

Specifically, we compare our models using two different gradient penalties with the model trained by WGAN-GP loss in terms of JSD, MMD and COV. As shown in Fig. 10, the models with our gradient penalties perform much better in both metric terms. Furthermore, the curves with respect to coverage demonstrate that our loss functions significantly promote generation diversity. In contrast, the model with WGAN-GP loss potentially fails into mode-collapse.

In Fig. 13, we compare the training curves of the models with different loss functions. These curves prove the effectiveness of our penalization methods in maintaining convergence and training stability. It is obvious that both gra-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3069195, IEEE Transactions on Visualization and Computer Graphics



(a) (b) (c)

Fig. 13: Training curves of using different loss functions. These curves demonstrate the effectiveness of our penalty methods in maintaining training stability and convergence.

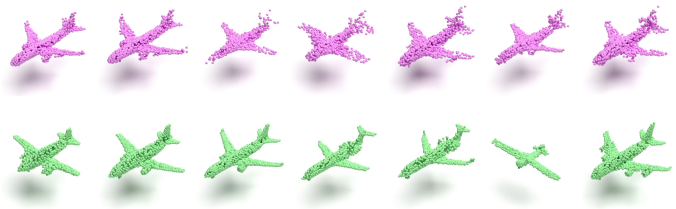


Fig. 14: Point clouds generated by the models with different loss functions. Comparing with the green point clouds provided by the model using our gradient penalty, the pink results synthesized by the model with WGAN-GP loss are similar and lack representative details.

dient penalty methods bring about convergence and stable training. Meanwhile, WGAN-GP loss results in significant fluctuation and non-convergence. The point clouds shown in Fig. 14 further illustrate that the proposed regularization methods improve both generation diversity and fidelity.

4.5 Partial set generation

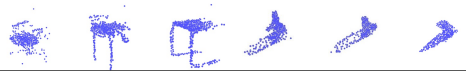
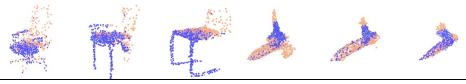
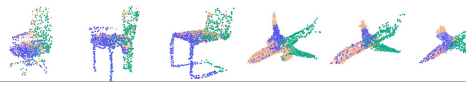
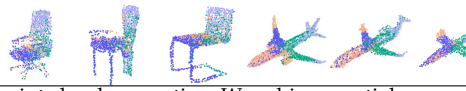
Latent feature	Category	
	Chair	Airplane
$\{s_1\}$		
$\{s_1, s_2\}$		
$\{s_1, ..., s_3\}$		
$\{s_1, ..., s_4\}$		

TABLE 4: Partial point cloud generation. We achieve partial generation by separately grouping the 3D points stem from different non-leaf nodes $\{s_1, ..., s_4\}$.

In this section, we explore how the proposed learning architecture impacts the generation of 3D shapes. Particularly, we examine the correlation between the embedded graph nodes and their corresponding 3D points. Due to the fact that the latent information of different 3D partial structures is embodied in each non-leaf node, our model is capable to create partial point sets and adjust them for accurately composing a 3D shape under unsupervised condition. To be specific, we define a point cloud as $\mathbf{S}$, and the graph at l level as $G^{(l)} = (\mathbf{V}^{(l)}, E^{(l)})$. Each non-leaf node $v_i^{(l)} \subseteq \mathbf{V}^{(l)} (i = 1, ..., |\mathbf{V}|)$ implies the latent information in correspondence with a subset $s \subseteq \mathbf{S}$. However, this latent vector lacks the connective information among different segments. To overcome potential deformations, we embed the topology information $g_G^{(l-1)}$ extracted from graph into the branched non-leaf nodes. This operation makes our model appropriately link up the partial sets derived from different parent nodes in 3D space.

In our experiments, the partial components are not connected correctly without adding the graph features in the learning process. Lacking the meaningful connective information of graph nodes always results in serious deformation in the synthesized point clouds. Integrating a rooted tree architecture with self-attention based graph learning facilitates our framework accurately assembling the partial sets. We can get the partial set s of $\mathbf{S}$ by collecting the 3D points corresponding to the specific non-leaf node $v^{(l)}$. Table. 4 illustrates how SG-GAN partially generates point clouds, and we mark various partial point sets using different colors. Taking the partial set derived from the node with the highest self-attention score as a reference, other subsets present gradual transfiguration in geometric structures. This is because SG-GAN enhances the latent information of the nodes with higher self-attention scores and employs it to adjust the rest nodes. Even though the input latent code is different, SG-GAN results in similar correlation between the self-attention score and local proximity.

To further testify our analysis, we estimate the contiguous relationships among the partial sets in correspondence to the graph nodes at the same level. Specifically, we permute the nodes in descendant order following the self-attention scores. Then, we compute Hausdorff distances between the rest partial sets and the reference set. For providing rational evaluation, the experimental data is averagely estimated from 500 random samples. The curves shown in Fig. 15 (a)-(c) present the same situation as Table. 4. The Hausdorff distances from the reference target to other partial sets are increased as the self-attention sorting. However, the curves shown in (d)-(f) are much more gentle, especially the curves with respect to chair. This difference is caused by the penalization methods employed in our model. In the light of our analysis, penalizing the gradient of generated data results in gradual shape optimization. Meanwhile, adopting the gradient penalty of real data makes the model tend to uniformly enhance the representative information of each node.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3069195, IEEE Transactions on Visualization and Computer Graphics

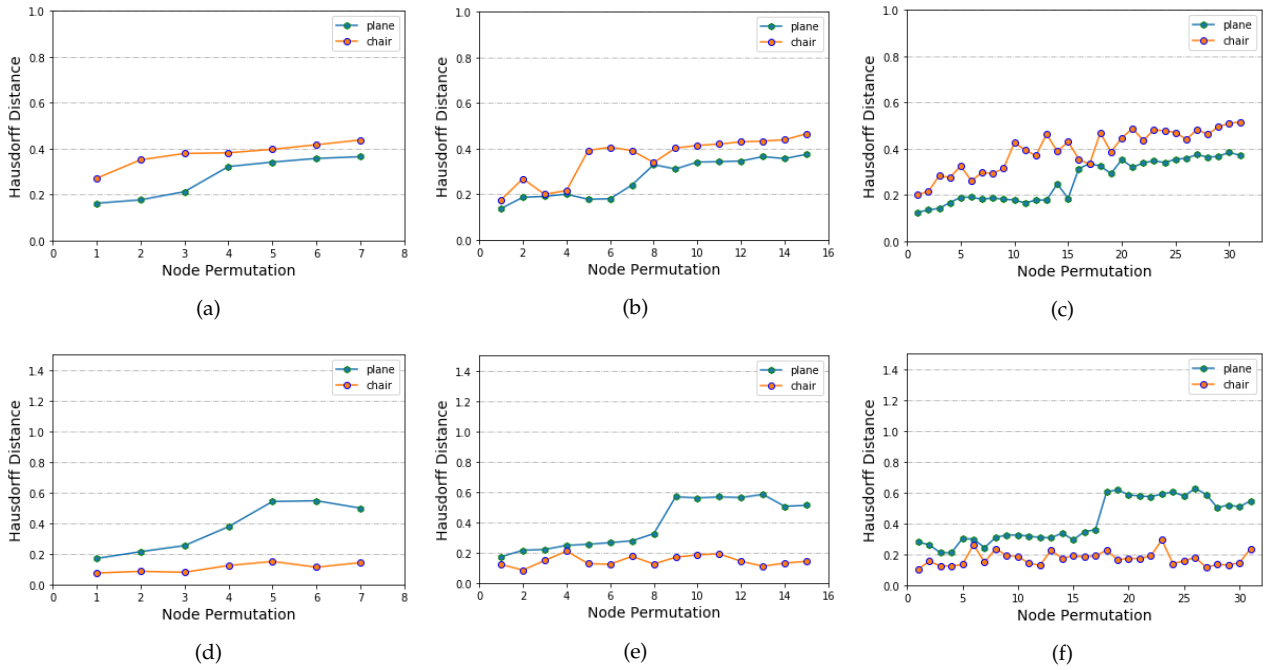


Fig. 15: Hausdorff distance evaluation. To further analyze the influence of SAGN layers on partial structure generation, we permute the nodes of different layers in accordance with the assigned self-attention scores. After this, we take the point sets derived from the node with the highest score as a reference and estimate the Hausdorff distances between it and other nodes. Notably, (a)-(c) show the results of the model with real data penalization, while (d)-(f) present the results of the model with generated data penalization.

4.6 Future improvement

In this paper, SG-GAN model focuses on recovering and transferring topology information from dynamically updated graphs, which makes this approach capable to generate compact 3D shapes. However, the binary tree structure limits the capability of our generator in learning different latent topologies and reconstructing meticulous details. What is more, aggregating the global graph information rather than the local region information can potentially alleviate the structural information related to some specific details. The results shown in Fig. 16 demonstrate that our model is still limited in generating the 3D shapes with complicated and exiguous structures, especially when these details are scarce in the training set. According to our knowledge, the possible improving paths are: (1) replacing global graph learning with local graph learning and (2) employing the frameworks with stronger inference capability, such as Bayesian, to replace the binary tree architecture.

Another strong advantage of SG-GAN is the unsupervised generation of topological parts. Both qualitative and quantitative results provided in this paper demonstrate the effectiveness of SG-GAN in diverse point cloud generation.

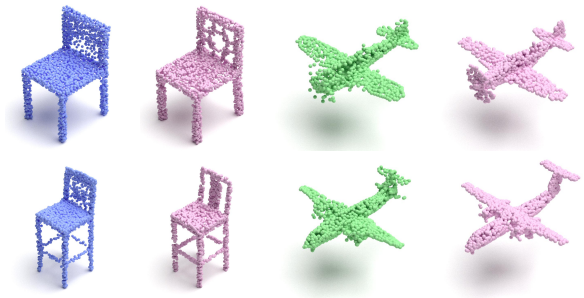


Fig. 16: Difficult cases. Some shapes with detailed structures are difficult for our model to generate. The pink samples are the ground truth, while the green (airplane) and blue (chair) point clouds are the generated results of our model.

5 Conclusion

Modified generative adversarial networks are highly effective in self-attention based graph learning, especially for high-dimensional feature data sets. In this paper, we present a novel generative framework that considers 3D shape generation as the topological graph learning. By employing self-attention to aggregate latent information encoded in a topological graph, our model reliably transfers useful information along a binary tree. Furthermore, we present two methods to penalize the Wasserstein loss in order to overcome the problems in GAN-based models such as non-convergence and mode-collapse. The proposed training objective not only solves these problems but also helps our model to avoid multiple discriminator updates in training.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[3] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[6] Chao Chen, Guanbin Li, Ruijia Xu, Tianshui Chen, Meng Wang, and Liang Lin. Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4994–5002, 2019.

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[8] 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*, pages 2672–2680, 2014.

[9] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[10] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[11] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Transactions on Graphics (TOG)*, 37(6):1–12, 2018.

[12] Mor Joseph-Rivlin, Alon Zvirin, and Ron Kimmel. Momen (e) t: Flavor the moments in learning to classify shapes. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[13] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[14] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, 2017.

[15] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. *arXiv preprint arXiv:1904.08082*, 2019.

[16] Huan Lei, Naveed Akhtar, and Ajmal Mian. Octree guided cnn with spherical kernels for 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9631–9640, 2019.

[17] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point cloud gan. *arXiv preprint arXiv:1810.05795*, 2018.

[18] 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*, pages 820–830, 2018.

[19] Hongxin Lin, Zelin Xiao, Yang Tan, Hongyang Chao, and Shengyong Ding. Justlookup: One millisecond deep feature extraction for point clouds by lookup tables. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 326–331. IEEE, 2019.

[20] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5239–5248, 2019.

[21] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[22] Albert Pumarola, Stefan Popov, Francesc Moreno-Noguer, and Vittorio Ferrari. C-flow: Conditional generative flow models for images and 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7949–7958, 2020.

[23] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016.

[24] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[25] 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*, pages 5099–5108, 2017.

[26] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 3d graph neural networks for rgbd semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5199–5208, 2017.

[27] Yongming Rao, Jiwen Lu, and Jie Zhou. Spherical fractal convolutional neural networks for point cloud recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 452–460, 2019.

[28] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Neighbors do help: Deeply exploiting local structures of point clouds. *arXiv preprint arXiv:1712.06760*, 1(2), 2017.

[29] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4548–4557, 2018.

[30] 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*, pages 3859–3868, 2019.

[31] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017.

[32] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. Pointgrow: Autoregressively learned point cloud generation with self-attention. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 61–70, 2020.

[33] Gusi Te, Wei Hu, Amin Zheng, and Zongming Guo. Rgcnn: Regularized graph cnn for point cloud segmentation. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 746–754, 2018.

[34] 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*, pages 6411–6420, 2019.

[35] Diego Valsesia, Giulia Fracastoro, and Enrico Magli. Learning localized generative models for 3d point clouds via graph convolution. 2018.

[36] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–66, 2018.

[37] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36(4):1–11, 2017.

[38] 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.

[39] 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*, pages 9621–9630, 2019.

[40] 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*, pages 1912–1920, 2015.

[41] 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*, pages 4541–4550, 2019.

[42] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3323–3332, 2019.

[43] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5565–5573, 2019.

[44] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 3d point capsule networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1009–

1018, 2019.