Received 3 December 2023, accepted 23 January 2024, date of publication 13 February 2024, date of current version 23 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3365519

RESEARCH ARTICLE

Style-Based Tree GAN for Point Cloud Generator

YANG SHEN[1], HAO XU[1], YANXIA BAO[1,2,3], ZHENGNAN XU [1], YUANHU GU[1], JIANG LU[1], ZHEN YING[1], AND KENAN LOU[1], (Member, IEEE)
[1]Department of Computer Science and Technology, Lishui University, Lishui, Zhejiang 323000, China
[2]Zhejiang Juxin Automation Equipment Company Ltd., Lishui, Zhejiang 323000, China
[3]Zhejiang Zhangxin Media Technology Company Ltd., Lishui, Zhejiang 323000, China

Corresponding author: Yanxia Bao (tlsheny@126.com)

This work was supported in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LY21F020004, in part by the National Natural Science Foundation of China under Grant 61572243 and Construction Project of Lishui University Discipline, Zhejiang Province First Class Discipline, Discipline name: intelligent science and technology under Grant XK0430403005, and in part by the Public Welfare Technology Application Research Program Project of Lishui City under Grant 2022GYX12.

ABSTRACT Point cloud generation and representation is important in industry areas. Generating and editing high quality 3D shapes is challenging work in deep learning. Inspired by StyleGAN, a style based generative adversarial networks is proposed to generate high quality 3D point cloud. An improved non-linear mapping network learn distribution of points and is used to generate well distributed point cloud point cloud. We also provide a coarse to fine representation for point cloud. According to the experimental results on the ShapeNet Part data set(including aircraft, chair single category and overall 16 categories), our method can generate more uniform point cloud than other GAN methods with less training epoches. The latent code for point cloud has better linear separation,and is more easy to edit.

INDEX TERMS Point cloud, StyleGAN, TreeGAN, mapping network.

I. INTRODUCTION

The generation method of 3D point cloud is one of the hot issues in the field of computer vision. 3D datasets are being widely used in robot navigation [1] and autonomous vehicles [2], [3], augmented reality [4] health care [5]. Among various datasets, point clouds are becoming popular as an original representation [6], [7] which can capture complex details of objects. 3D point cloud can be considered as a disordered set of irregular points collected from the surface of an object. Each point is represented by cartesian coordinate and other additional information such as curvature, surface normal and RGB color.

Point cloud generation models based on generative adversarial networks are proposed in recent years. Generative adversarial network (GAN) is a generative model proposed by Goodfellow et al. [8]in 2014, and has become a hot research topic in the field of artificial intelligence [9], [10]. It is widely used in image generation and point cloud generation in the image generation model, the pixels generated in the image are arranged in regular grids, while point cloud is represented by discrete points, which are not aligned by regular grids. Traditional GANs always generate non-uniform point clouds without grid structure. The generated points are concentrated in some special regions, such as geometric center of the object and junction of different semantic parts, while other regions are sparse. It leads to inhomogeneous points in point cloud. To get high quality point cloud, there always need large number of iterations in training.

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

Our work is inspired by styleGAN. In styleGAN, latent code is used to represent the probability density of data and make it editable. In this paper, a style tree generative adversarial network is proposed to improve training efficiency and make generated point clouds well distributed. Mapping network in our method is designed as pyramid structure, which give a coarse to fine representation for point cloud, so it can learn the distribution of point clouds more accurately. In each layer of generation network, the distribution of the point cloud is aligned with the style code though AdaIN module, which changes the generated nodes and makes the shape of the generated point cloud more accurate. Our work not only solves the problem of uneven distribution, but also improves the training efficiency.

Style tree GAN also provides latent space which can used in point cloud editing. Output in each level of mapping network make up of W_k space. Like that in image editing, we can perform style mixing operations to mix two different point clouds, and use the latent code to change the attribute of point cloud. Compared with space Z, W_k space of point cloud shows better linear separability in semantic editing. Experiments prove the effectiveness of the method.

The main contributions of this paper are as follows:

(1) We analyze uniform problem in point cloud generation, and show that it highly influences the quality of generated point cloud and training efficiency.

(2) Style based Tree GAN is proposed to improve the uniform of point cloud. An improved mapping network is proposed to improve the expressiveness of latent code. The method is more efficient in training, and experiments show that it can generate more uniform point cloud that other methods.

(3) We give a coarse to fine representation for style editing in point cloud. Experiments show that W_k space has better linear separation and more easy to edit.

II. RELATED WORK

Recently, the issue of point cloud generation based on deep neural networks has attracted wide research interest,such as image-to-point cloud [11], image-to-voxel [12], image-to-grid [13], point-to-voxel [14] and point-to-point cloud [15]. It has been widely used in computer vision applications (such as segmentation [16], [17], classification [18], [19], target detection [20], [21], feature extraction [22], etc.) and has achieved remarkable results.

GAN [23], [24] for image generation has been widely studied for many years. Achlioptas et al. [25] proposed r-GAN for 3D point cloud generation. The generator is made up of many fully connected layers. Since the fully connected layer cannot maintain local structure in point cloud, r-GAN is hard to generate realistic 3D shapes. Valsesia et al. [26] used graph convolution instead of fully connected network in GAN generator. During training of graph convolution in each layer, an adjacency matrix is dynamically constructed using feature vectors from each vertex.The space complexity of training is $O(V^2)$, where V represents the number of vertices, so this method needs large number of memory to train. Unlike the method in Valsesia, Tree-GAN [27] generate point cloud like building tree. It firstly generate the primary point as ancestral nodes, and uses the ancestral nodes to generate child nodes. Tree-GAN only requires a list of tree structures. It is more efficient because it does not construct adjacency matrices. In ShapeInversion [28], a uniform loss is proposed to solve the problem of generating point cloud inhomogeneity. The farthest point sampling (FPS) is used to randomly sample the n seed positions on the surface of the object, and then the K nearest neighbors of each seed are used to form a cell block. The average distance between each seeds and its K nearest neighbors is calculated, and the variance of the average distance of all blocks is penalized. However, due to the lack of appropriate regularization, the points of upon methods tend to form a Gaussian-like distribution. For example, the points gather at the geometric center of the object or the junction of different semantic parts. The nodes generated in the current layers are dependent on the nodes in previous layers, resulting in uneven shape of the generated point cloud.

One of the most important improvement of GAN is StyleGAN [29],which propose a novel style-based generator. Different from traditional GAN, styleGAN use nonlinear mapping network to control the style of each convolutional layer. Different from the three-dimensional point cloud, the pixels of the two-dimensional image are arranged regularly, and the latent code in StyleGAN controls the gray color feature of the image. Inspired by StyleGAN, the paper improves the mapping network and applies it to point cloud generation. Different from ShapeInversion [28], which solved the inhomogeneity problem of by loss function, we solve the problem by learning uniform latent code in Style TreeGAN. Latent code is mapped to point cloud by the AdaIN module and make the points well distributed. Our method need not any additional loss function. Training efficiency is much higher than that of traditional Tree GAN.

In recent years, diffused based methods [30], [31], [32] achieved good results in image generation. Wang et al. [33] proposed roll-out diffusion network to generate 3D avatars. Different from GAN and FLOW methods, latent code in diffused based methods could not reflect semantic information. To solve this problem, Preechakul et al. [34] proposed a diffusion autoencoders for images, the key idea is to use a two-part latent code, the first part use semantically meaningful latent code from VAE, the second part captures stochastic details. In point cloud editing, linear and meaningful representation are also important in point cloud editing. In the paper, we give a coarse to fine representation for style editing in point cloud, it has better linear separation and more easy to edit.

Shen et al. [35] proposed interface GAN to edit the image. Similarly with interface GAN, we use the latent code in point style GAN to perform semantic editing. Experiments show that our method is effective in point cloud editing.

III. LIMITATION OF TRADITIONAL GAN FOR POINT CLOUD GENERATION

The GAN applications for point cloud are similar. In each layers of generator, new nodes are generated from parent nodes in network. Here, we take TreeGAN as example. TreeGAN is a typical tree structure generation network. Points are obtained from the Gaussian distribution as input. At each layer of the generator, branching and graph convolution operations are performed to generate the next layer of points. All nodes generated by the previous layers are stored and attached to the tree of the current layer. The tree starts from the root node, splits into child nodes through branch operations, and modifies the value of the nodes through graph convolution operations. In the generator, different branching degrees are used for each layer, the Branching module is used to increase the total number of nodes in the process of generating point clouds, which is similar to the upsampling in two-dimensional convolution. 2048 points are obtained after the last layer of branching operation. Different from the traditional graph convolution, which updates its value through the value of the adjacent points, the tree graph convolution (TreeGCN) proposed by TreeGAN updates its nodes through ancestors. The formula is as follows:

$$p_i^{l+1} = \sigma(F_K^l(p_i^l) + \sum_{q_j \in N(p_i^l)} U_j^l q_j + b^l) \tag{1}$$

where σ is the activation unit. F_K^l is a fully connected layer containing K nodes in l_{th} layer, p_i^l is the i_{th} node at.

Because TreeGAN updates the value of the current node through ancestor nodes, it uses the ancestor information to improve the representation ability of current node. On the other hand, the newly generated points in each layer of the tree structure is affected by the ancestor nodes. If the distribution of the ancestor nodes is not uniform, the child nodes generated by parent nodes could not construct the shape of point cloud as well as possible, and the final points are easy to concentrate on the semantic center part of the object or the different semantic connections (as shown in Figure 1), and the training efficiency is low.

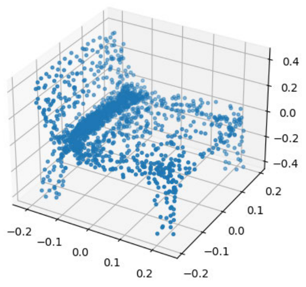

FIGURE 1. Chair point cloud generated by TreeGAN.

A. LOSS FUNCTION

In this GAN, the loss function [36] of the generator is always defined as follows:

$$L_{gen} = -E_{z \in Z}[D(G(z))] \tag{2}$$

D is the discriminator, G is the generator, Z is the potential distribution, and the normal distribution $N(0, I)$ is used. In TreeGAN, the loss function of discriminator is defined as follows:

$$\begin{aligned} L_{disc} = &E_{z \in Z}[D(G(z))] - E_{z \in Z}[D(x)] \\ &+ \lambda_{gp} E_{\hat{x}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2] \end{aligned} \tag{3}$$

where $G(z)$ is the generated point cloud, x is the real point cloud, $\hat{x}$ is sampled from the true and false point cloud, λ_{gp} is a constant coefficient of the gradient penalty term.

The loss function of the discriminator will make the generated points focus on the discriminative features of the point cloud, such as the edges, which will also lead to uneven distribution of the generated point cloud.

Upon analysis show that both training efficiency and accuracy of point cloud have strong relationship with distribution of point cloud. Since distribution of parent nodes affect the distribution of child nodes, coarse to fine representation is helpful to generate accurate point cloud. In this paper, we align the distribution of point cloud to the latent code though mapping network, so as to generate more uniform point cloud.

IV. STYLE BASED TREE GAN

In this section, style based TreeGAN [27] is proposed. Inspired by StyleGAN [29] which generates image by style transfering, we propose style based TreeGAN to generate point cloud. The method use an improved mapping network to learn the probability distribution of the point cloud, and use the style transfer to control the generation of point cloud, so the generated point clouds are well distributed.

A. STYLE BASED GENERATOR

In order to solve the problem of non-uniformity caused by tree network and discriminator loss function, we introduce the method in style transfer [37] into the generator and use the improved mapping network to transfer the latent code to style code, and use style code to control the distribution of the generated point cloud.

Our generator is composed of mapping network and tree structure generation network. The mapping network has 11 fully connected layers to learn the probability distribution of point clouds. The tree structure network consists of seven PT-Block modules (ProbabilityTree-Block). Each module has three parts: adaptive instance normalization (AdaIN), branching, and graph convolution (GraphConv). The latent code is trained through the fully connected layer, and then mapped to each PT-Block module through the AdaIN module. The network structure is shown in Figure 2.

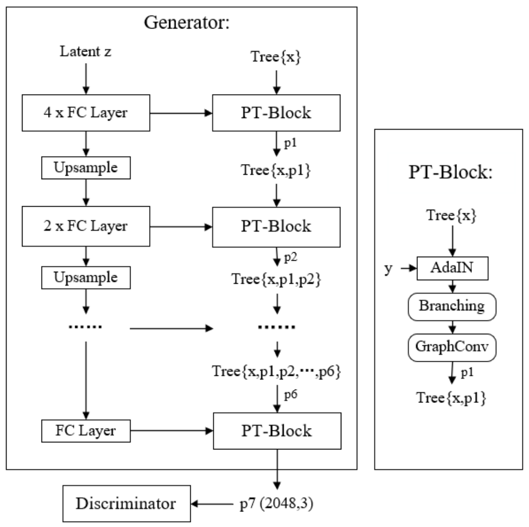

FIGURE 2. Style TreeGAN network structure.

B. AdaIN MODULE

Adaptive Instance Normalization (AdaIN) [37] in style transfer aligns the style of latent code to that of image features. In point cloud generation, the AdaIN module is used to transfer the mean and variance of latent code to the nodes of point cloud. The formula is as follows:

$$AdaIN(x, y) = \sigma(y)(\frac{x - \mu(x)}{\sigma(x)}) + \mu(y) \qquad (4)$$

where $\sigma(x)$ and $\sigma(y)$ are the variance of vectors and the latent code respectively, and $\mu(x)$ and $\mu(y)$ are the mean of the basis vector and the latent code respectively.

In point generation network, AdaIN modules transfer the latent code to distribution of nodes in different resolutions, so it aligns point cloud with different resolution, and make the generated points well distributed.

C. IMPROVED MAPPING NETWORK

Inspired by the idea that the latent code of mapping network in StyleGAN is used to control the generation of image, this paper apply it to point cloud generation. Different from images which represent color in fixed grid, point cloud are made up of 3D points distributed in irregular positions. Here we use a coarse to fine mapping network to represent the distribution of 3D points. It is different from that of styleGAN for image.

Figure 3 (a) shows the mapping network in StyleGAN for image. The latent code is mapped to first layer after 8 fully connected layers, and is upsampled to other levels. Our mapping network is shown in Figure 3 (b), the first two layers of the mapping network is used to learn the coarse-grained features of the point cloud (such as shape, size, etc.), then it is expand to high resolution in next layers by upsampling operation. After upsampling, two full connection layers are used to learn a better distribution for high resolution in latent code space. With growing of the depth of network, the mapping network can give a coarse to fine representation for point cloud.

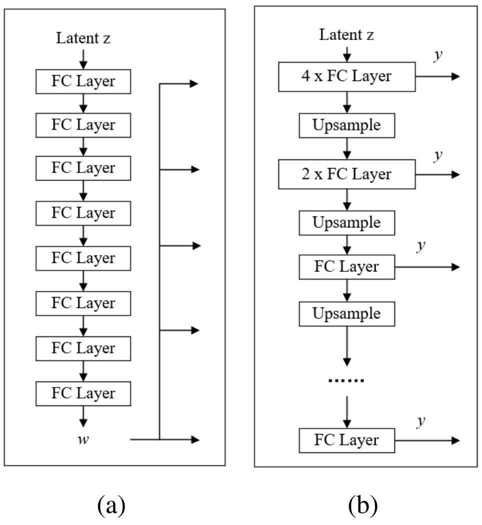

FIGURE 3. The mapping network structure before and after improvement.

Our improved mapping network seems very simple, however, it is very effective. The advantage of mapping network are reflected in three aspects. Firstly, it provides coarse to fine representation for point cloud generation, many experiments show that it has better linear separations in semantic editing. Secondly, this mapping network greatly increase training efficiency, to achieve same JSD value, we only need one fifth time of original tree GAN, and the training procedure is more stable. Finally, the generated point cloud is more uniform compared with origin tree GAN.

The key technique is that we use style code to control the generation of point cloud. Since the upsample operations generate the latent code uniformly in new layers, so the new latent code also aligns the new generated nodes uniformly by AdaIN module, and can generate more uniform point cloud. Because the generated nodes are aligned in each layers of network, the training is stable and efficient.

V. SEMANTIC EDITING IN STYLE TREEGAN

The coarse to fine representation can not only learn more accurate distribution of point cloud, but also provide better linear separations. Similar with method in [35], the generator can be viewed as a deterministic function g: $Z \rightarrow X$. Where $Z \in R^d$ represents the d-dimensional latent space; X represents the semantic space, such as the shape and size of the point cloud.

Suppose there is a semantic scoring function f_S: $X \rightarrow S$, where S represents a semantic space with m semantics. We can bridge latent space Z and semantic space S by $s = f_S(g(z))$, where s and z denote the semantic score and sampled latent code, respectively.

Given a hyperplane with a normal vector $n \in R^d$, define the distance of a sample z from this hyperplane as:

$$d(n, z) = n^T z \qquad (5)$$

The scoring function f for semantics is defined as:

$$f(g(z)) = \lambda d(n, z) \qquad (6)$$

where $\lambda > 0$ is a scalar measuring how quickly semantics change with distance. The corresponding semantics can be expressed by the normal vector n. To change the semantics of a generated point cloud, we edit the original latent z using Equation 7.

$$z_{edit} = z + \alpha n \qquad (7)$$

where α controls the direction of latent code, and the edited semantic score becomes as shown in Equation 8.

$$f(g(z_{edit})) = f(g(z)) + \lambda\alpha \qquad (8)$$

Taking $\alpha > 0$ will make the generated point cloud tend to have positive semantics, and similarly $\alpha < 0$ will make the generated point cloud tend to have negative semantics. Semantic editing in formula 8 requires different attributes can be linearly separated from the space Z. The output of mapping network in Style TreeGAN make up of W_k space. Different from space Z, mapping network provides a better disentanglement in latent space, so space W_k has better linear separation.

VI. EXPERIMENTAL RESULTS AND ANALYSIS

A. IMPLEMENTATION DETAILS

Our experimental platform is the LINUX server with GeForce RTX 3090 GPU, configured in Python 3.8 and Tensorflow2.4-gpu. Adam optimizer was used for both the generator and discriminator networks, learning rate of $\alpha = 10^{-4}$, $\beta_1 = 0$, $\beta_2 = 0.99$. LeakyReLU($\alpha = 0.2$) is used as a nonlinearity function without batch normalization. To make the training more stable, while generator is updated for one iteration, the discriminator is updated for five iterations.The gradient penalty coefficient in the loss function is set to 10. K value in formula 1 is set to 10.

1) PARAMETER SETTING

For generator, the latent vector $z \in \mathbb{R}^{96}$ is sampled from the normal distribution $N(0, I)$ as input. The total number of points in the last layer is set to n = 2048.

For generation network, the network structure and parameters are shown in following table. In table 1, the generation network is made up of 7 PT-blocks, which upsample the nodes to 2048 3D points. Table 1 shows the parameters of mapping network, in first 4 full convolution layers, mapping network learns the coarse distribution of point cloud, with following 7 full convolution layers, mapping network learns the details of point cloud.

TABLE 1. The parameters of generation network.

Network Layer	Input	Output	batchsize	degrees
0	1×96	1×256	32	1
1	1×256	2×256	32	2
2	2×256	4×256	32	2
3	4×256	8×128	32	2
4	8×128	16×128	32	2
5	16×128	32×128	32	2
6	32×128	2048×3	32	64

TABLE 2. The parameters of mapping network.

Latent z	Input	Output
4×FC Layer	1×96	1×256
2×FC Layer	1×256	2×256
FC Layer	2×256	4×256
FC Layer	4×256	8×128
FC Layer	8×128	16×128
FC Layer	16×128	32×128
FC Layer	32×128	2048×3

As shown in Table3, our discriminator network is 10 layers convolution network, which include 6 point convolution layers, the number of channels increases from 3 to 1024. The network gives the output after the max pooling and three full convolution layers.

TABLE 3. The parameters of discriminator network.

Network Layer	Output
Input point cloud	2048×3
1D-Conv, stride=1, channel=3	2048×3
1D-Conv, stride=1, channel=64	2048×64
1D-Conv, stride=1, channel=128	2048×128
1D-Conv, stride=1, channel=256	2048×256
1D-Conv, stride=1, channel=512	2048×512
1D-Conv, stride=1, channel=1024	2048×1024
GlobalMaxPooling1D	1×1024
FC Layer, unit=128	1×128
FC Layer, unit=64	1×64
FC Layer, unit=1	1×1

B. EVALUATING DATASETS

We use two dataset to evaluate our point generation method, one is Shapenet, the other is ModelNet. ShapeNet Part has 16 categories, 12137 training samples, 1870 validation samples, 2874 test samples, a total of 16881 samples. ModelNet10 has 10 categories,including 4899 samples.

C. EVALUATING INDICATOR

In this paper, JS divergence (Jensen-Shannon Divergence) is used to evaluate the quality of point cloud generated by GAN.JS divergence is defined as the edge distribution of Euclidean three-dimensional space. The assumption is that the axis-aligned point cloud data can measure the degree to which point cloud A and point cloud B occupy similar positions. By counting the number of points in each voxel in point cloud A and point cloud B respectively, the JSD between empirical distributions (P_A,P_B) is obtained:

$$JSD(P_A\|P_B) = \frac{1}{2}KL(P_A\|M) + \frac{1}{2}KL(P_B\|M) \tag{9}$$

where $M = (P_A + P_B)/2$, $D(\cdot\|\cdot)$ is the Kullback-Leibler divergence between two distributions. The KL divergence is a metric used to measure the similarity between two probability distributions. The formula is as follows:

$$KL(P\|Q) = \sum_{x\in X}[P(x)log(\frac{P(x)}{Q(x)})] \tag{10}$$

D. COMPARISON OF TRAINING

In order to prove the effectiveness of Style treeGAN in training, we compare it with original TreeGAN. The aircraft category was trained for 2000 cycles. From figure 4, it can be seen that the JSD of original TreeGAN fluctuates greatly, while the JSD of style TreeGAN is smooth. This is because that latent code make the generated point cloud more regular and stable. Compared with rGAN [25], our method can get better JSD. It proves that our mapping network can learn a good representation for the point cloud probability distribution.

E. COMPARATIVE EXPERIMENTS OF ACCURACY

We compare our model with other GAN models for point cloud generation. 5000 epoches were trained on the ShapeNet

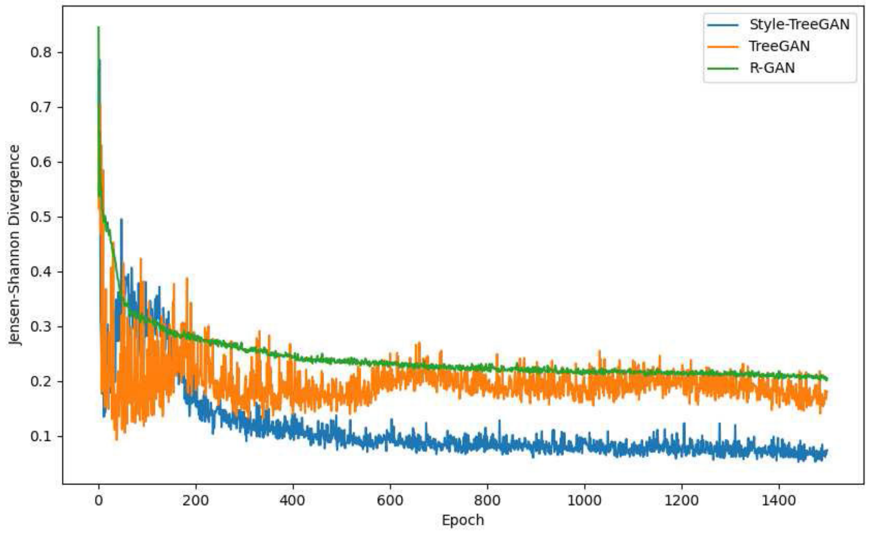

FIGURE 4. Comparison between Style TreeGAN and original TreeGAN and rGAN.

TABLE 4. Quantitative comparison in JSD(Shapenet).

Class	Model	JSD↓
Chair	r-GAN(dense)	0.238
	r-GAN(conv)	0.517
	Valsesia et al. (no up.)	0.119
	Valsesia et al. (up.)	0.100
	TreeGAN	0.119
	Style TreeGAN(ours)	**0.093**
Airplane	r-GAN(dense)	0.182
	r-GAN(conv)	0.350
	Valsesia et al. (no up.)	0.164
	Valsesia et al. (up.)	0.083
	TreeGAN	0.097
	Style TreeGAN(ours)	**0.074**
All(16classes)	r-GAN(dense)	0.171
	TreeGAN	0.105
	Style TreeGAN(ours)	**0.091**

TABLE 5. Quantitative comparison in JSD(Modelnet10).

Class	Model	JSD↓
Chair	r-GAN	0.117
	TreeGAN	0.195
	Style TreeGAN(ours)	**0.067**
bed	r-GAN	0.196
	TreeGAN	0.114
	Style TreeGAN(ours)	**0.099**
desk	r-GAN	0.150
	TreeGAN	0.204
	Style TreeGAN(ours)	**0.080**
sofa	r-GAN	0.073
	TreeGAN	0.150
	Style TreeGAN(ours)	**0.055**
table	r-GAN	0.098
	TreeGAN	0.105
	Style TreeGAN(ours)	**0.071**
toilet	r-GAN	0.095
	TreeGAN	0.140
	Style TreeGAN(ours)	**0.052**

Part dataset and Modelnet dataset. On Shapenet Part dataset, the JSD of our model reaches 0.093, 0.074 and 0.091 on chairs, airplanes and 16 classes respectively. It is better than other GAN models for point cloud generation. The results based on aircraft and chairs are compared and all 16 categories are evaluated in table 4. On Modelnet10 dataset, the JSD of our model also achieves best results in all important classes, which is shown in table 5.

F. COMPARISON OF UNIFORMITY

We compare the uniformity of point cloud with original tree GAN. Two models (2048 points) were trained for 2000 epoches for comparison. The results are shown in figure 5.

Figure 5 shows the results generated by TreeGAN and Style TreeGAN. In the first row, points of seat are gathered in the joint area. In second and third row, the points of plane and desk are distributed in the center of object, while points in other parts are sparse. For Style TreeGAN, points are uniform and well distributed in whole object.The experimental results show that the point clouds generated by our model are more uniform.

G. COMPARISON OF TRAINING EFFICIENCY

To prove the training efficiency and convergence speed of our model, the change of the JSD in first 150 epoches are drawn by the line chart, as shown in Figure 6.

It can be seen from the line chart that Style TreeGAN is faster than TreeGAN in JSD convergence speed. In order to see the quality of the point cloud generated during model training, the point cloud generated during each cycle of two models are compared in figure 7.

From figure 7, when training 100 epoches, the chair generated by our model has begun to take shape, while the chairs generated by TreeGAN and rGAN are still clustered

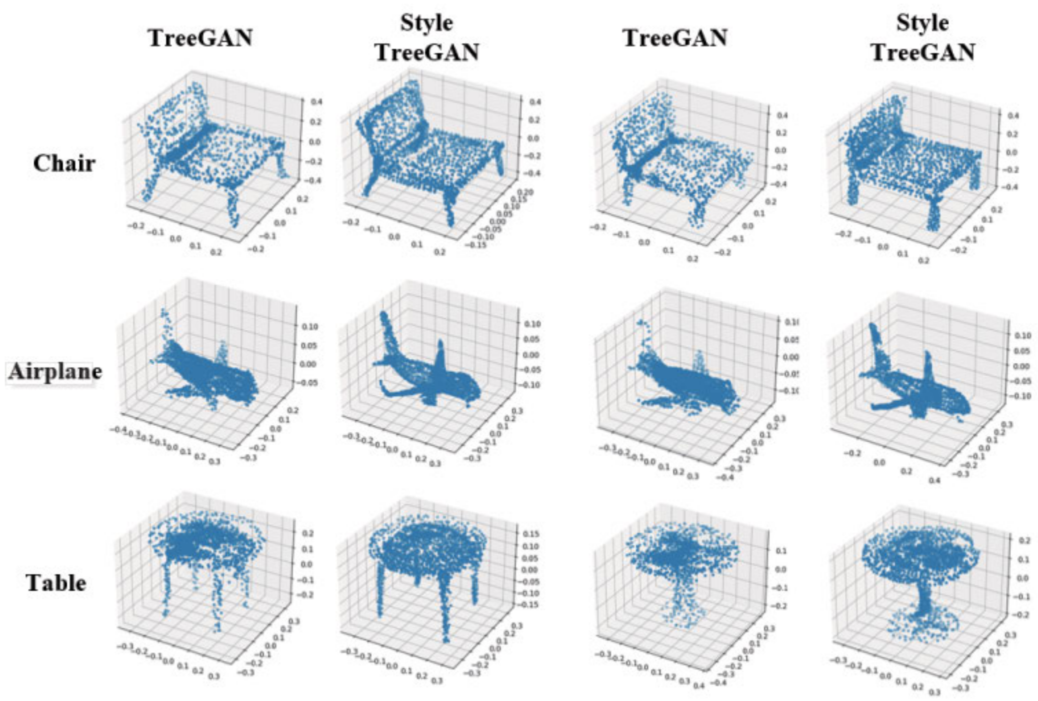

FIGURE 5. Uniformity comparison of point cloud generation.

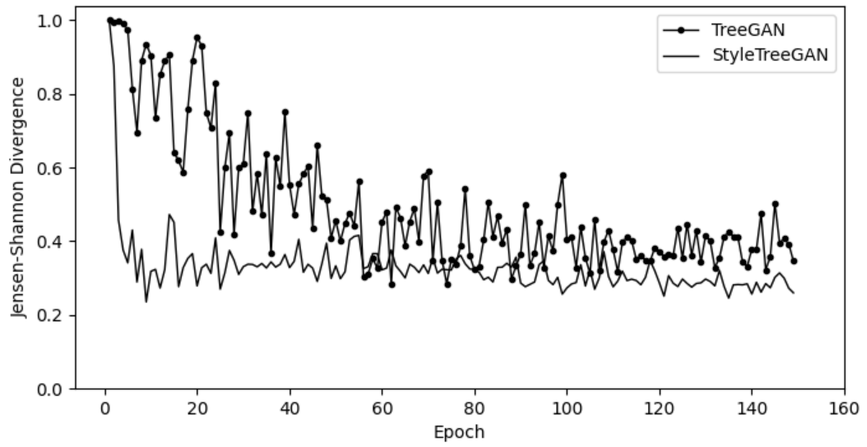

FIGURE 6. Model training efficiency comparison by line chart.

together. After training 500 epoches, the chair generated by TreeGAN and rGAN form the contour, while the chair generated by the model in this paper has concrete shapes. From the visualization results, the Style-TreeGAN is better than TreeGAN. In additional, for a more intuitive comparison of training speed, we counted the training cycles required to different JSD, as shown in table 6.

As can be seen from table 6, to achieve the same JSD, the training epoches of the proposed model is much less than TreeGAN and rGAN. When JSD reaches 0.3, rGAN need 680 epoches, TreeGAN needs 600 epoches, while Style TreeGAN only needs 100 epoches, which is 6 times faster than TreeGAN. When JSD reaches 0.15, Style TreeGAN requires 1000 epoches less training than TreeGAN. The above experiments prove the efficiency of the proposed model in training efficiency.

TABLE 6. The epoches required for the specified JSD.

Model	JSD=0.3(epoch)	JSD=0.2	JSD=0.15
rGAN	680	900	1200
TreeGAN	600	1200	2000
Style TreeGAN	100	200	1000

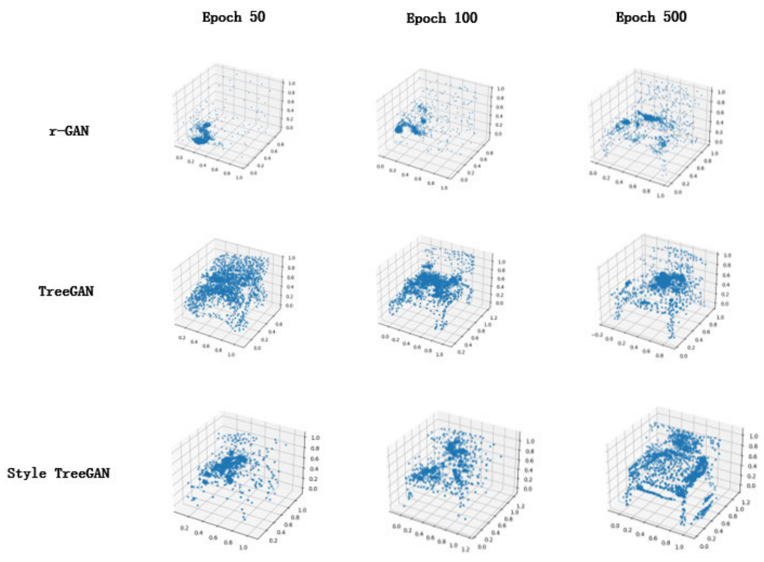

FIGURE 7. Comparison of the point clouds in different epoches.

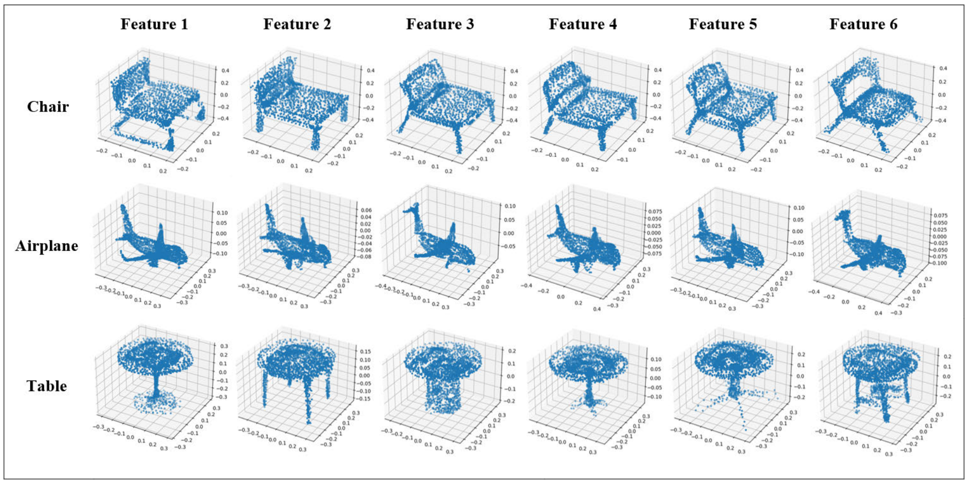

FIGURE 8. Generate point cloud renderings under different features.

H. CHANGE THE LOCAL CHARACTERISTICS OF THE GENERATED POINT CLOUD

The trained model can change the local features of the generated point cloud by changing the initial latent code in the mapping network. Six different latent codes were taken to change the local features of the generated point cloud. The results are shown in figure 8.

I. STYLE MIXING

Similar with styleGAN for image, we perform style mixing on point cloud. In the experiment, two vector $z_1, z_2 \in \mathbb{R}^{96}$ is randomly sampled from Gauss distribution, and the corresponding latent code w_1, w_2 are generated in W_k by different z. Then the style mixing is realized by applying w_1 before crossover point and w_2 after it. Some experimental results are shown in Figure 9.

J. SEMANTIC EDITING

In order to edit the binary semantic attributes of the generated point cloud, we use the method in InterfaceGAN to find a hyperplane in W_k space as the separation boundary, and change the semantics of the point cloud by editing the latent code near the boundary. Taking the table as an example, a round table and a square table are selected as the binary semantics of the point cloud.

Firstly, we use this model to train 2000 epoches to generate point cloud data of round table and square table, select 500 point clouds from the generated results as data samples for training SVM, save and generate the latent code corresponding to each point cloud, and calculate the attribute value. The selected latent code and the corresponding attribute score are input into the SVM for training, and a decision boundary with a normal vector n is obtained. In order

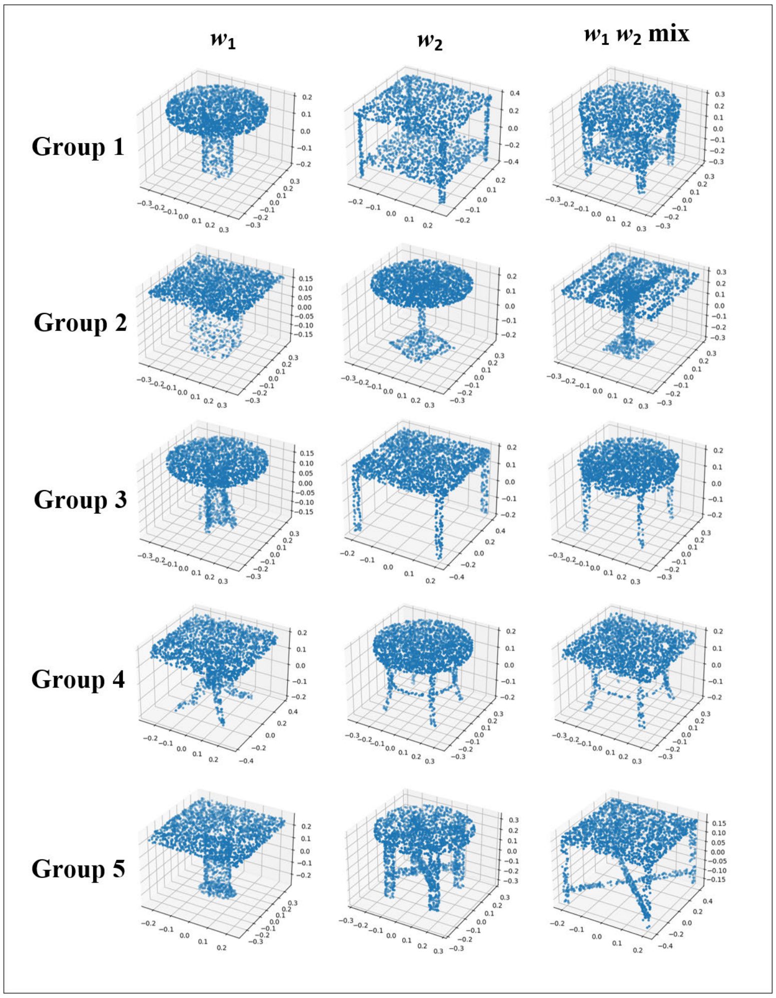

FIGURE 9. Partial style mixing experimental results.

to verify the ability of semantic edit in the latent space, the latent code z is taken on the boundary, and its corresponding hyperplane normal vector n is obtained through the SVM classifier. Operate $z_{edit} = z + \alpha n$ on the latent code z, and change the semantics of the point cloud by changing the value of α. Input the edited latent code into this model to generate a point cloud. We select latent codes in different spaces (W_k, Z) to edit the attributes, and the results are shown in Figure 10.

In Figure 10, the five point clouds from left to right are the results with different α values (-1, -0.5, 0, 0.5, 1), and the generated point clouds gradually change from a square table to a round table. The first 2 lines are the effect of semantic editing in Z space, and the last 2 lines are the renderings of semantic editing in W_k space. Comparing the first line of Figure 10, the quality of point cloud editing in Z space is lower than that in W_k space. When α changes from -1 to 1, the point cloud in Z space changes from a square table to a round table, and turned into a square table again. Experiments indicate that the latent code on the Z space cannot solve the linear entanglement problem very well, while the linear semantic editing on the W_k space work well, and the quality of the generated point cloud is higher.

K. LINEAR CLASSIFICATION

In order to verify that the latent codes in the W_k space have better linear separability, SVM classification experiments are carried out on the latent codes. Taking the table and chair point cloud as an example, the latent codes in the Z space W space W_k space are classified by SVM.

Firstly, 2000 epoches are trained for the latent codes in the Z, W, and W_k spaces respectively. 600 samples are taken from each of the three spaces. We used 6-fold cross-validation, divided 600 data into 6 parts, and used 5 parts for training and 1 part for verification in turn. The average of the results was used as an evaluation of the classification accuracy. The classification accuracy and average accuracy of 6-fold cross-validation are shown in Table 7.

According to the data in Table 7, the average precision of the latent code in the W_k space reaches 73.67%, which is higher than 67.17% and 70.83% in the Z and W space,

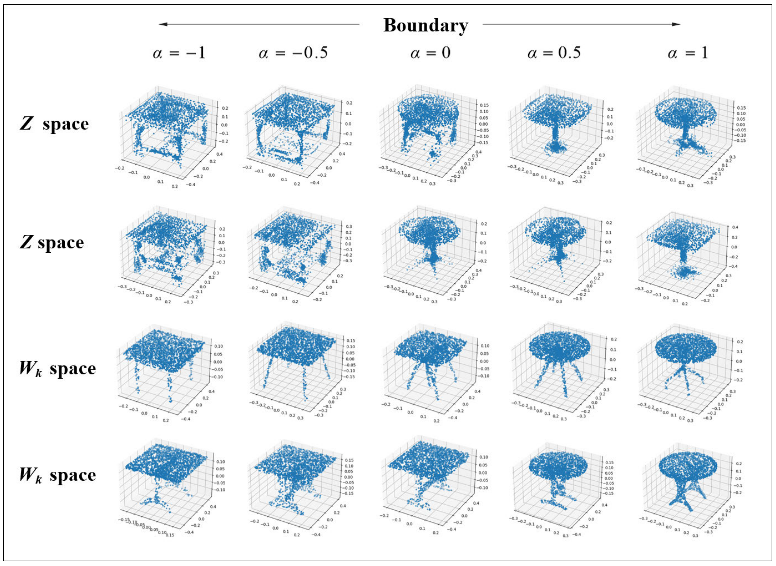

FIGURE 10. Semantic point cloud editing.

TABLE 7. SVM classification accuracy.

Space	Group1	Group2	Group3	Group4	Group5	Group6	Average
Z	66%	71%	68%	66%	63%	69%	67.17%
W	69%	71%	74%	72%	69%	70%	70.83%
W_k	72%	75%	71%	74%	78%	72%	73.67%

indicating that the latent code in the W_k space has better linear separability than the Z and W space.

VII. CONCLUSION

This paper proposes a style based tree generation adversarial network. The mapping network is added to the generator to train and learn the probability distribution of the point cloud. In the point cloud generation, the distribution of the point cloud is aligned with that of the latent code, and the shape of the generated point cloud is more structural. Our method not only improve the training efficiency, but also make the generated point cloud distribution more uniform. The point cloud generated by our model is better than other GAN models in the evaluation of JSD indicators. The latent code of the model has better linear representation.

Our method shows the advantage of style mapping network in point cloud generation. In fact, it does more than that in image generation. In images, the pixels are distributed in fixed cell, but each cell has different color, so in styleGAN, mapping network is used to learn the distribution of color information. In point cloud, all points are distributed in 3D space. The shape of point cloud is decided by the positions of points, points are not distributed in fixed cells, so the function of mapping network in style TreeGAN can learn the distribution of points and control the shape of point cloud. By learning the distribution of points, compared with traditional Tree GAN, our method can generate more accurate point cloud.

Our method also show the importance of coarse to fine representation. Coarse to fine representation can not only provide better ability for semantic editing, but also raise training efficiency. Coarse to fine representations help to generate regular nodes in different layers. Compared with irregular nodes, the regular nodes are more easy to form shape.

However, our methods also need some improvements, though our W_k space is disentangle representation of the point cloud, it could not implement the complicate semantic editing in point cloud. GAN model is not reversible, for given point cloud, we need special methods to get the latent code. Futher improvement like styleflow [38]are consider to be apply to our applications.

ACKNOWLEDGMENT

(Shen Yang and Xu Hao contributed equally to this work.)

REFERENCES

[1] X. Wang, Y. Mizukami, M. Tada, and F. Matsuno, "Navigation of a mobile robot in a dynamic environment using a point cloud map," *Artif. Life Robot.*, vol. 26, no. 1, pp. 10–20, Feb. 2021.

[2] A. Pfrunder, P. V. K. Borges, A. R. Romero, G. Catt, and A. Elfes, "Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3D LiDAR," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 2601–2608.

[3] J. Park, C. Kim, S. Kim, and K. Jo, "PCSCNet: Fast 3D semantic segmentation of LiDAR point cloud for autonomous car using point convolution and sparse convolution network," *Expert Syst. Appl.*, vol. 212, Feb. 2023, Art. no. 118815.

[4] J. Zhou, H. Xu, Z. Ma, Y. Meng, and D. Hui, "Sparse point cloud generation based on turntable 2D LiDAR and point cloud assembly in augmented reality environment," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, May 2021, pp. 1–6.

[5] B. Liu, M. Guo, E. Chou, R. Mehra, S. Yeung, N. L. Downing, F. Salipur, J. Jopling, B. Campbell, and K. Deru, "3D point cloud-based visual prediction of icu mobility care activities," in *Proc. Mach. Learn. Healthcare Conf.*, 2018, pp. 17–29.

[6] P. Tian, X. Hua, K. Yu, and W. Tao, "Robust segmentation of building planar features from unorganized point cloud," *IEEE Access*, vol. 8, pp. 30873–30884, 2020.

[7] H. Xu, S. Zhou, Y. Shen, K. Lou, R. Zhang, Z. Ye, X. Li, and S. Wang, "Kernel product neural networks," *IEEE Access*, vol. 9, pp. 167076–167083, 2021.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, vol. 2, Dec. 2014, pp. 2672–2680.

[9] B. Liu, J. Zhang, and J. Zhu, "Boosting 3D adversarial attacks with attacking on frequency," *IEEE Access*, vol. 10, pp. 50974–50984, 2022.

[10] J. Zhang, Y. Dong, M. Kuang, B. Liu, B. Ouyang, J. Zhu, H. Wang, and Y. Meng, "The art of defense: Letting networks fool the attacker," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 3267–3276, 2023.

[11] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 605–613.

[12] F. Liu and X. Liu, "Voxel-based 3D detection and reconstruction of multiple objects from a single image," 2021, *arXiv:2111.03098*.

[13] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2Mesh: Generating 3D mesh models from single RGB images," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 52–67.

[14] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4490–4499.

[15] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point cloud auto-encoder via deep grid deformation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 206–215.

[16] L. Yongwei, L. Jiazong, S. Yuliang, and W. Xiangyang, "Point cloud completion of indoor scenes based on category-instance segmentation," *Chin. J. Comput.*, vol. 44, no. 11, pp. 2189–2202, 2021.

[17] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, Oct. 2019.

[18] M. A. Jinghui, P. Wei, and W. Ru, "3D point cloud classification based on k-means clustering," *Comput. Eng. Appl.*, vol. 56, no. 17, pp. 181–186, 2020.

[19] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

[20] H. Xuan, X. Fei, and C. Tao, "UAV target detection on quantum multi-pattern recognition optimization algorithm," *Comput. Eng. Appl.*, vol. 57, no. 7, pp. 228–236, 2021.

[21] Z. Yu, Z. Jing, Z. Hua, and X. Xianpeng, "Hand-eye calibration method based on object detection for robots," *Comput. Eng.*, vol. 48, no. 3, pp. 100–106, 2022.

[22] J. Dai, M. Wei, Q. Xie, and J. Wang, "Aircraft seam feature extraction from 3D raw point cloud via hierarchical multi-structure fitting," *Comput.-Aided Des.*, vol. 130, Jan. 2021, Art. no. 102945.

[23] K. Ehsani, R. Mottaghi, and A. Farhadi, "SeGAN: Segmenting and generating the invisible," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6144–6153.

[24] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. H. Lau, and M. H. Yang, "VITAL: Visual tracking via adversarial learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8990–8999.

[25] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3D point clouds," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 40–49.

[26] D. Valsesia, G. Fracastoro, and E. Magli, "Learning localized generative models for 3D point clouds via graph convolution," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–15.

[27] D. Shu, S. W. Park, and J. Kwon, "3D point cloud generative adversarial network based on tree structured graph convolutions," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3858–3867.

[28] J. Zhang, X. Chen, Z. Cai, L. Pan, H. Zhao, S. Yi, C. K. Yeo, B. Dai, and C. C. Loy, "Unsupervised 3D shape completion through GAN inversion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 1768–1777.

[29] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 4401–4410.

[30] J. A. Ho and P. Abbeel, "Denoising diffusion probabilistic models," 2020, *arXiv:2006.11239*.

[31] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," 2021, *arXiv:2102.12092*.

[32] S. Luo and W. Hu, "Diffusion probabilistic models for 3D point cloud generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 2836–2844.

[33] T. Wang, B. Zhang, T. Zhang, S. Gu, J. Bao, T. Baltrusaitis, J. Shen, D. Chen, F. Wen, Q. Chen, and B. Guo, "Rodin: A generative model for sculpting 3D digital avatars using diffusion," 2022, *arXiv:2212.06135*.

[34] K. Preechakul, N. Chatthee, S. Wizadwongsa, and S. Suwajanakorn, "Diffusion autoencoders: Toward a meaningful and decodable representation," 2021, *arXiv:2111.15640*.

[35] Y. Shen, C. Yang, X. Tang, and B. Zhou, "InterFaceGAN: Interpreting the disentangled face representation learned by GANs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 4, pp. 2004–2018, Apr. 2022.

[36] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein gans," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5769–5779.

[37] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1501–1510.

[38] R. Abdal, P. Zhu, N. J. Mitra, and P. Wonka, "StyleFlow: Attribute-conditioned exploration of StyleGAN-generated images using conditional continuous normalizing flows," *ACM Trans. Graph.*, vol. 40, no. 3, pp. 1–21, Jun. 2021.

YANG SHEN received the Ph.D. degree from Shanghai Jiaotong University, in 2011. From 2012 to 2014, he was a Lecturer with Lishui University, where he has been an Associate Professor with the Department of Computer Science, since 2015. He is the author of three books, more than 20 articles, and two inventions. His research interests include image processing and computer graphics.

HAO XU received the B.S. degree in information and communication engineering, the M.S. degree in information and communication engineering, and the Ph.D. degree in computer engineering from Honam University, South Korea, in 2012, 2014, and 2017, respectively. He is currently a Lecturer with the College of Engineering, Lishui University, Zhejiang, China. His current research interests include the IoT and deep learning.

YANXIA BAO received the Ph.D. degree from Chosun University, South Korea, in 2022. She is currently a Lecturer in computer science with Lishui University, China. Her current research interests include image processing and deep learning.

ZHENGNAN XU received the M.S. degree from Zhejiang Sci-Tech University. His current research interest includes machine learning.

YUANHU GU received the Ph.D. degree from the University of the Cordilleras, Philippines. He is currently with Lishui University, China, where he teaches computer-related courses focusing on software engineering and artificial intelligence.

JIANG LU received the M.S. degree from Yunnan University. He is currently a Lecturer with Lishui University, Zhejiang, China. His current research interest includes machine learning.

ZHEN YING received the M.S. degree from the Zhejiang University of Technology. He is currently an Associate Professor with Lishui University, Zhejiang, China. His current research interest includes machine learning.

KENAN LOU (Member, IEEE) received the M.S. degree in tourism studies and the Ph.D. degree in hotel tour from Honam University, South Korea, in 2013 and 2018, respectively. She is currently a Lecturer with the College of Business, Lishui University, Zhejiang, China. Her current research interests include wisdom tourism and machine learning. She is a Reviewer of IEEE Access and *Artificial Intelligence Review*.