TIGER: Time-Varying Denoising Model for 3D Point Cloud Generation with Diffusion Process

Zhiyuan Ren, Minchul Kim, Feng Liu, Xiaoming Liu
Michigan State University
{renzhiy1, kimminc2, liufeng6, liuxm}@msu.edu

Abstract

Recently, diffusion models have emerged as a new powerful generative method for 3D point cloud generation tasks. However, few works study the effect of the architecture of the diffusion model in the 3D point cloud, resorting to the typical UNet model developed for 2D images. Inspired by the wide adoption of Transformers, we study the complementary role of convolution (from UNet) and attention (from Transformers). We discover that their respective importance change according to the timestep in the diffusion process. At early stage, attention has an outsized influence because Transformers are found to generate the overall shape more quickly, and at later stages when adding fine detail, convolution starts having a larger impact on the generated point cloud's local surface quality. In light of this observation, we propose a time-varying two-stream denoising model combined with convolution layers and transformer blocks. We generate an optimizable mask from each timestep to reweigh global and local features, obtaining time-varying fused features. Experimentally, we demonstrate that our proposed method quantitatively outperforms other state-of-the-art methods regarding visual quality and diversity. Code is avaiable https://github.com/Zhiyuan-R/Tiger-Diffusion.

1. Introduction

The point cloud is an essential 3D shape representation that is easy to obtain (sensed directly from laser scanning), versatile (serving as a building block of mesh), and easy to manipulate (geometrically transformed in a straightforward manner by affine matrices). 3D point cloud generative models benefit extensive applications across robotics [35], medicine [15] and content creation [36], and lay the groundwork for other vision tasks like point cloud upsampling [52], point cloud completion [21, 31] and mesh generation [9]. However, compared to 2D images, the cost and complexity of acquiring 3D point clouds make it crucial to explore and develop efficient and effective model architectures for 3D point cloud generation.

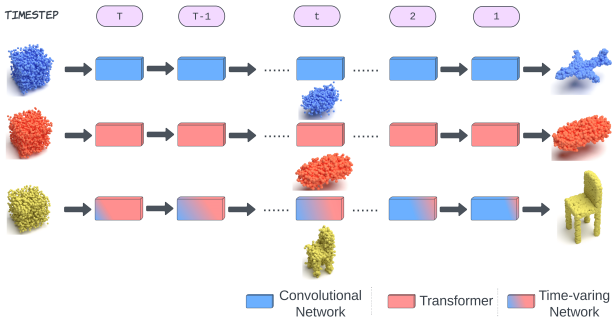


Figure 1. An illustration of different roles played by convolution and attention operations in the denoising model. Convolution is good for learning local relationships, and attention is optimal for modeling global relationships. We propose to merge these two properties across different timesteps in the diffusion process.

Existing point cloud generative models are built on a range of frameworks, including generative adversarial networks (GANs) [1, 5], variational autoencoders (VAEs) [24], normalizing flows [23, 27, 51]. However, these methods neither are stable in training nor produce high-fidelity results. Recently, the denoising diffusion model [18], a novel generative method, has demonstrated exceptional results in 3D point cloud generation. Typically, diffusion models perturb a point cloud using a forward process and denoising algorithm. Once trained, the model can be used to generate new point clouds by iterating the reverse process over a sequence of time steps, with each step adding more noise to the input point cloud.

Several recent works, including DPM [34] and PVD [55], have successfully applied diffusion models to 3D point cloud generation, achieving high levels of naturalness and diversity. Very recently, LION [53] further improves by encoding point clouds into latent space. However, these methods commonly utilize UNet-like convolutional networks that are originally designed for image processing. *There has been limited research on developing suitable de-*

noising model architectures that are specifically tailored for 3D point cloud generation. For example, PVD and LION both utilize Point-Voxel CNN (PVCNN) [30] as their denoising model. However, we observed that these PVCNN-based denoising models require a considerable number of timesteps to establish a rough shape since the limited receptive field cannot capture the global distribution of noise.

In contrast to convolution operations, we observe that the attention mechanism used in Transformers [10, 48] is more effective at capturing long-range dependencies and thus the overall shape of 3D points. This difference is illustrated in the first two rows of Fig. 1. Compared to the convolution-based diffusion model (first row), the attention-based diffusion model (second row) correctly conforms to the overall object shape at earlier denoising time steps (middle column). However, the convolution operation excels in modeling local relationships. Therefore, the details of the final output are better modeled with convolutions (last column) which is also discussed in the ablation analysis of Sec. 4.3. The finding suggests that the two operations, the convolution and the attention are complementary in their roles at different denoising timesteps (the last row of Fig. 1) and we ask the following question.

> *How to leverage the strengths of both CNNs and Transformers and enable them to dynamically capture complementary information in denoising diffusion for 3D point cloud generation?*

To answer this question, we propose a Time-varying denoising model for 3D point cloud generation (TIGER), a two-stream architecture combining a shallow CNN branch and Transformer branch with a timestep-dependent mask that adaptively reassigns weights to the global Transformer feature and the local CNN feature. To enable this feature-level fusion, we design an encoder-decoder to downsample the point cloud for reducing the number of points to a manageable size for Transformer. And we learn a selection mask that finds the optimal configuration at each timestep.

Furthermore, in this two-stream architecture where global and local information is modeled separately, we discover that supplying 3D points position information to the global branch is critical to the performance. To supply the 3D position information more effectively, we design two novel position encoding methods: phase-shifted position encoding and Base-λ position encoding. They create 3D position information in a continuous space and retain the property of linear expression of relative position. The effect of the time-dependent dual branch mask and the proposed position embedding is ablated in Sec. 4.3.

A unique property of TIGER is that the weights of the two branches are dependent on the diffusion timesteps, and they are learned and optimized for the 3D point cloud generation. Interestingly, experiments demonstrate that the weight of the convolution branch is monotonically increasing with the timestep, which is in accordance with our initial observation that convolutions are good for modeling fine details. Empirically we show that TIGER achieves state-of-the-art (SoTA) performance in 3D point cloud generation.

Our main contributions include:

- We propose a novel two-stream denoising model, which uses timestep to optimally reweigh the global feature from Transformer and the local feature from shallow CNN.
- We design two novel 3D continuous position encoding methods and position-aware self-attention to enable our Transformer branch to aggregate global features effectively from the latent point cloud input.
- Experiments show that our method outperforms prior works in 3D point cloud generation tasks qualitatively and quantitatively on the ShapeNetv2 dataset.

2. Related Work

3D Point Cloud Generation. The advances in 3D point cloud generation can be categorized by how the input is represented. Early methods [1, 12, 14, 38, 39, 44] represent the point cloud as a $N \times 3$ matrix, where N is the predefined number of 3D points. The backbone comprises 1D convolutions and max-pooling to allow the points' permutation invariance. So, the shape is represented by a global feature vector.

More recent works [5, 7, 13, 27, 34, 51] have improved upon the input representation by introducing a probabilistic distribution assumption, where each point cloud is considered as one data point sampled from the distribution. For instance, PointFlow [51] models the distribution of points by continuous normalizing flows, allowing the sampling of an arbitrary number of points to represent a point cloud. However, the backbone architecture is still based on 1D convolution-based encoder with the max-pooling operation [38].

Recently, PVD [55] and LION [53] adopt a hybrid point-voxel representation [30] which voxelize the point clouds for 3D convolution. This representation with rich spatial information combines well with the diffusion objective and produces promising results [53]. However, the backbone is still confined to convolution-based networks, lacking the capacity to effectively model the long range-dependencies. We, on the other hand, find the complementary roles of attention and convolution in the point cloud generator and propose a network that combines the benefits of both.

Diffusion Models. Diffusion [18] and score-based methods [22, 46] have enjoyed remarkable success in images synthesis. Diffusion has been extended to many generation

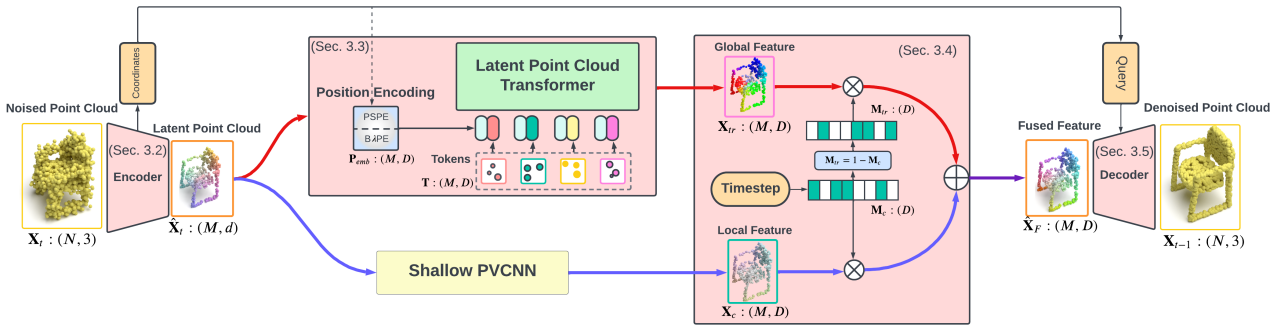


Figure 2. Illustration of our time-varying two-stream architecture (TIGER). The network's input is a noisy point cloud $\mathbf{X}_t$ at timestep t, and the goal is to predict the noise in $\mathbf{X}_t$ to obtain the denoised point cloud $\mathbf{X}_{t-1}$. We first encode $\mathbf{X}_t$ into latent point cloud $\hat{\mathbf{X}}_t$ and export coordinates information for position encoding and decoder. We feed the latent point cloud into both the Latent Point Cloud Transformer and a shallow CNN branch to get global feature $\mathbf{X}_{tr}$ and local feature $\mathbf{X}_c$. Then, we combine two features using timestep dependent mask $\mathbf{M}_c$ to create a fused feature $\hat{\mathbf{X}}_F$. Last, we apply our latent point cloud decoder to predict the noise from $\hat{\mathbf{X}}_F$.

tasks [2, 8, 17, 20, 25, 28, 41–43, 47], and largely improved by innovative techniques, such as fast sampling [33, 45], classifier-free guidance [19, 37] and latent diffusion [4, 42]. Recently 3D point cloud generation with diffusion [34, 53, 55] is also proposed with remarkable diversity and fidelity.

However, previous works do not investigate the behavior of the diffusion backbone with respect to the diffusion time step. We find that the network equipped with both convolution and attention operations learns to utilize the local and global shape representation with different magnitudes based on the timestep. This time adaptive architecture allows the 3D point cloud generator to be trained faster than LION [53] while producing higher-quality outputs.

Transformer for Point Clouds. Transformer architecture [48] is intrinsically a set operator, which is suitable for permutation-invariant point clouds. PCT [16] applies global attention to the point cloud. Point Transformer V1 [54] and V2 [49] convert global attention to local attention to reduce memory consumption and computational complexity.

Contrary to the previous methods, our method decouples the global and local information by using Transformer to aggregate global information effectively while using CNN to model the local information. Also, we find that position embedding is crucial to performance and propose a new position embedding to further improve the performance.

3. Method

In this section, we first briefly formulate the probabilistic model of forward and reverse diffusion processes for 3D point cloud generation. Then, we dive into the details of our time-varying two-stream architecture, including the encoder part, latent point Transformer, time mask generator, and decoder part. The entire architecture is shown in Fig. 2.

3.1. Problem Formulation

Given a point cloud $\mathbf{X}_0 \in \mathbb{R}^{N\times 3}$, where N is the number of points, the diffusion forward process gradually diffuses the original point cloud into Gaussian noise $\mathbf{X}_T$; the forward process can be formulated with a transition probability $q(\mathbf{X}_t|\mathbf{X}_{t-1})$. By giving a sequence of pre-defined noising scales $\beta_1, \beta_2, ..., \beta_T$, the transition probability can be expressed as :

$$q(\mathbf{X}_t|\mathbf{X}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\mathbf{X}_{t-1}, \beta_t\mathbf{I}). \tag{1}$$

For the reverse process, we use learn the $p_\theta(\mathbf{X}_{t-1}|\mathbf{X}_t)$, a Gaussian distribution which approximates the intractable real distribution $q(\mathbf{X}_{t-1}|\mathbf{X}_t)$. Specifically,

$$p_\theta = \mathcal{N}(\mu_\theta(\mathbf{X}_t, t), \sigma_t^2\mathbf{I}), \tag{2}$$

where we learn the mean $\mu_\theta(\mathbf{X}_t, t)$ and set the variance σ_t^2 to a fixed schedule as in [18]. Conventionally, $\mu_\theta(\mathbf{X}_t, t)$ is modeled using a UNet backbone and it depends on the timestep t to scale and shift the feature. We design a two-stream backbone where the timestep modulates the weight between the two streams. To sample a 3D point cloud, we iteratively denoise $\mathbf{X}_T$, T times to finally generate the high-fidelity samples. In training, we optimize the MSE loss:

$$\begin{aligned}\mathcal{L}_{simple} &= \mathbb{E}_{t\sim[1,T]}\|\mu - \mu_\theta(\mathbf{X}_t, t)\|_2^2 \\ &= \mathbb{E}_{t\sim[1,T]}\|\epsilon - \epsilon_\theta(\mathbf{X}_t, t)\|_2^2,\end{aligned} \tag{3}$$

where ϵ is the ground truth noise and ϵ_θ is the noise prediction by our diffusion model.

3.2. Noisy Point Cloud Encoder

To effectively represent the $N \times 3$ points, we adopt the voxelization encoding scheme of PVCNN [30]. This representation works well with both CNN and Transformer architecture as we can downsample the voxel-grid following [11] to

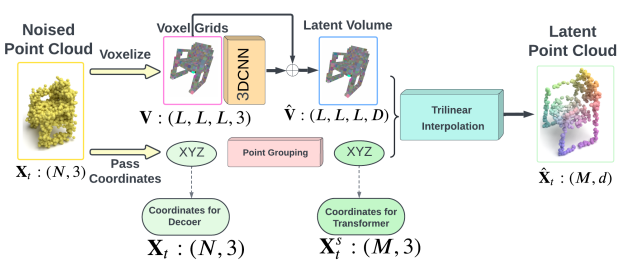
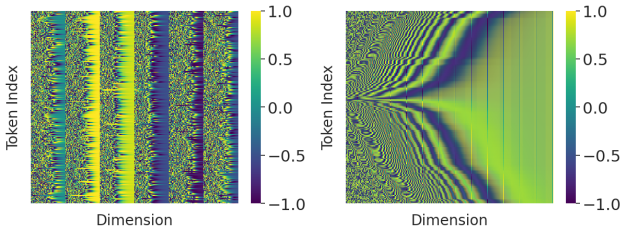


Figure 3. The overview of our encoder. It extracts features in voxel space and downsamples the point cloud by applying trilinear interpolation of voxel with grouped points.



(a) PSPE **(b)** BλPE

Figure 4. Illustration of two examples of the position embedding from PSPE and BλPE respectively. Both methods show distinguished representation for each position.

increase computation efficiency. Specifically, a noisy point cloud encoder $\mathcal{E}: \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}^{M\times d}$ transforms the noisy point cloud $\mathbf{X}_t \in \mathbb{R}^{N\times 3}$ at timestep t into a latent point cloud representation $\hat{\mathbf{X}}_t \in \mathbb{R}^{M\times d}$, where N is the number of original points, M is the number of downsampled points and d is the dimension of each latent point's feature. The overview of our encoder is illustrated in Fig. 3.

To aggregate features in a rigid field, we first voxelize the noisy point cloud $\{x_n, y_n, z_n\} = \mathbf{X}_t \in \mathbb{R}^{N\times 3}$ into the voxel grids $\{\mathbf{V}_{u,v,w}\} = \mathbf{V} \in \mathbb{R}^{L\times L\times L}$, where L is the resolution and n is the point index in N. Specifically,

$$\mathbf{V}_{u,v,w} = \sum_{n=1}^{N} \mathbb{I}(x_n \in \mathbb{N}(u,\delta), y_n \in \mathbb{N}(v,\delta), z_n \in \mathbb{N}(w,\delta)) * \mathbf{f}_n, \tag{4}$$

where $\mathbb{N}(z,\delta) = \{x | x \in [z-\delta, z+\delta)\}$, $\mathbf{f}_n$ denotes the n^{th} extracted latent feature corresponding to the n^{th} point $(\mathbf{X}_t)_n$ and $\mathbb{I}(\cdot)$ is an indicator of whether the n^{th} point falls into the voxel grid (u,v,w). After voxelization, we apply a series of 3D convolution, Swish [40] activation and GroupNorm [50]. After applying a series of these operations, this results in a latent volume $\hat{\mathbf{V}} \in \mathbb{R}^{L\times L\times L\times d}$, where d is the channel size for each voxel.

We use furthest point sampling algorithm [11] to downsample the input noisy point cloud $\mathbf{X}_t \in \mathbb{R}^{N\times 3}$ into a sparser point cloud $\mathbf{X}_t^s \in \mathbb{R}^{M\times 3}$ $(M < N)$. Then, as the position in volume is discrete, we query the latent volume $\hat{\mathbf{V}}$ by the sparse point cloud $\mathbf{X}_t^s$ via trilinear interpolation, eventually transformed into latent point cloud $\hat{\mathbf{X}}_t \in \mathbb{R}^{M\times d}$. It is worth noting we preserve the coordinates of $\mathbf{X}_t$ and $\mathbf{X}_t^s$ to do upsampling with the decoder and provide position embedding for our latent point cloud Transformer.

3.3. Latent Point Cloud Transformer

Previous denoising models such as PVCNN [30] simply apply stacked convolutional layers to create feature representations. Contrarily, we aim to better model the global relationship by introducing a novel Transformer branch named latent point cloud Transformer.

Tokenization. After encoding, we get a latent point cloud representation $\hat{\mathbf{X}}_t \in \mathbb{R}^{M\times d}$ and its coordinates $\mathbf{X}_t^s \in \mathbb{R}^{M\times 3}$. Following [29], we use dual PatchNorm to project the latent point cloud into tokens, which place LayerNorm before and after an MLP layer for more stable training:

$$\mathbf{T} = \mathrm{LN}(\mathrm{MLP}(\mathrm{LN}(\hat{\mathbf{X}}_\mathbf{t}))), \tag{5}$$

where $\mathbf{T} \in \mathbb{R}^{M\times D}$ and D is the dimension in the token space (in our paper, $D = 2d$).

Position Encoding. Since the order of the latent points is arbitrary in a point cloud, it's necessary to create a 3D position embedding and fuse it into the feature, so that the model knows the position of each point. Contrary to the position embedding in image domains [10, 26], the design of 3D position embedding plays a vital role in the performance and our position embedding produces the best result. We propose two novel 3D space continuous position encoding methods: Phase Shift Position Encoding (PSPE) and Base-λ Position Encoding (BλPE). For PSPE, we use Sine and Cosine functions of different frequencies and phases:

$$\mathbf{P}_{emb}(pos_j, 6i+2*(j-1)) = \sin(\frac{pos_j}{10000^{\frac{2i}{D}}} + (j-1)\frac{2\pi}{3}) \tag{6}$$

$$\mathbf{P}_{emb}(pos_j, 6i+1+2*(j-1)) = \cos(\frac{pos_j}{10000^{\frac{2i}{D}}} + (j-1)\frac{2\pi}{3})), \tag{7}$$

where pos_j is j-th element in each point's position, specifically $pos_1 = x$, $pos_2 = y$, $pos_3 = z$, and i is the index of the dimension. We maximize the phase shift difference $(\frac{4\pi}{3})$ of different axes to distinguish them and combine Sine and Cosine to guarantee the linear representation of relative position, which will be proved mathematically in Supp. An example is shown in Fig. 4a and $\mathbf{P}_{emb}$ is shaped $\mathbb{R}^{M\times D}$.

However, PSPE requires 6 channels to represent one frequency feature, which is quite inefficient. To improve channel utilization, we apply BλPE to compress 3D position into one scalar:

$$pos = \lambda^2 * z + \lambda * y + x \tag{8}$$

$$\mathbf{P}_{emb}(pos, 2i) = \sin(\frac{pos}{10000^{\frac{2i}{D}}}) \tag{9}$$

$$\mathbf{P}_{emb}(pos, 2i+1) = \cos(\frac{pos}{10000^{\frac{2i}{D}}}), \tag{10}$$

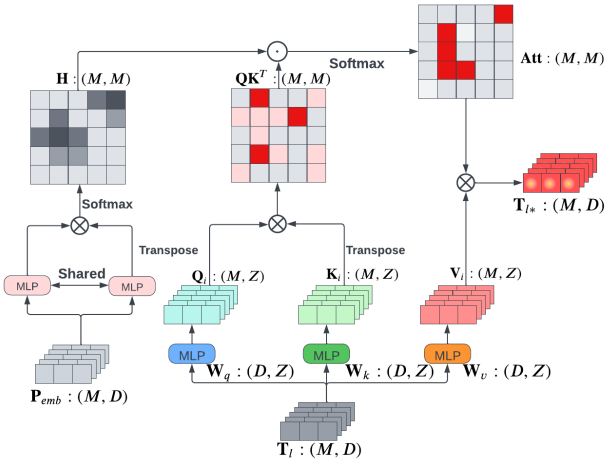


Figure 5. The overview of the position-aware self-attention. We compute position relationship map $\mathbf{H}$ by previous position embedding to reweigh the $\mathbf{QK}^T$ matrix.

where pos is a polynomial expression of λ. Specifically, we set λ to 1000, which means this preserves three decimal places of precision and one example is shown in Fig. 4b. The choice between these two position encoding methods is essentially a trade-off between granularity and expressiveness. We ablate them and other non-position encoding and learnable position encoding in Sec. 4.3.

Transformer and Position-Aware Attention We stack L layers of Transformer block [48] which comprises of LayerNorm(LN) [3] and Multihead Self Attention (MHA) and MultiLayer Perceptrons. We let the token $\mathbf{T}_0 = \mathbf{T}$ and let

$$\mathbf{T}_{l*} = \text{MHA}\left(\text{LN}(\mathbf{T}_{l-1})\right) + \mathbf{T}_{l-1}, \tag{11}$$

$$\mathbf{T}_{l} = \text{MLP}\left(\text{LN}(\mathbf{T}_{l*})\right) + \mathbf{T}_{l*}, \tag{12}$$

$$\mathbf{X}_{tr} := \mathbf{T}_L. \tag{13}$$

To add position embedding into the model, one can simply add it to $\mathbf{T}$. But as the Transformer blocks are gradually stacked, the continuous 3D position encoding is difficult to retain in the token representation. Inspired by [54], we explicitly add the position encoding to each MHA in order to make our attention map position-aware, as shown in Fig. 5. Specifically, our position-aware MHA modified as

$$\mathbf{H} = \text{Softmax}((\mathbf{P}_{emb}\mathbf{W}_p)(\mathbf{P}_{emb}\mathbf{W}_p)^T) \tag{14}$$

$$\mathbf{T}_{l*} = \text{Softmax}(\frac{(\mathbf{T}_l\mathbf{W}_q)(\mathbf{T}_l\mathbf{W}_k)^T \odot \mathbf{H}}{\sqrt{D}})(\mathbf{T}_l\mathbf{W}_v) \tag{15}$$

where $\mathbf{W}_q$,$\mathbf{W}_k$,$\mathbf{W}_v$,$\mathbf{W}_p \in \mathbb{R}^{D\times Z}$ are projection matrices and $\mathbf{H} \in \mathbb{R}^{M\times M}$ is the position relationship map. Modified MHA could incorporate position information explicitly and sufficiently during training. $\mathbf{X}_{tr}$ is the final output.

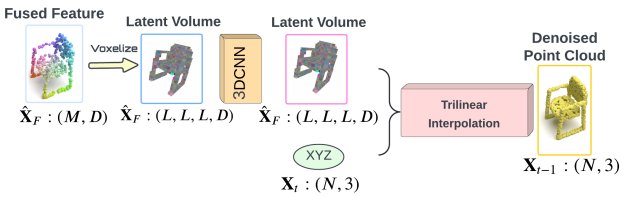


Figure 6. The overview of our decoder. By querying the latent volume $\hat{\mathbf{X}}_F$ with previous coordinates information $\mathbf{X}_t$, we can upsample the latent point cloud into 3D space.

3.4. Time Mask Generator

We apply a depth-2 PVCNN [30] to get local feature $\mathbf{X}_c \in \mathbb{R}^{M\times D}$ and our latent point cloud Transformer to get global feature $\mathbf{X}_{tr} \in \mathbb{R}^{M\times D}$. Time mask generator module scales the local and global features by the time variable. We force the sum of two learnable time masks to be 1. With this constraint, the trade-off of local and global features can be learned by the network.

We first align the dimension and scale of $\mathbf{X}_c$ and $\mathbf{X}_{tr}$ by one layer MLP and Sigmoid function σ.

$$\mathbf{X}_c^* = \sigma(\text{MLP}(\mathbf{X}_c)), \quad \mathbf{X}_{tr}^* = \sigma(\text{MLP}(\mathbf{X}_{tr})). \tag{16}$$

Then we encode a timestep t to a sinusoidal position embedding [18, 48] to $\mathbf{t}_{emb} \in \mathbb{R}^c$. We use two stacked MLP to map $\mathbf{t}_{emb}$ to a higher dimension space and apply Sigmoid to keep one of the masks between 0 to 1. Specifically,

$$\mathbf{M}_c = \sigma(\text{MLP}(\text{LeakyReLU}(\text{MLP}(\mathbf{t}_{emb})))), \tag{17}$$

$$\mathbf{M}_{tr} = \mathbf{1} - \mathbf{M}_c. \tag{18}$$

where $\mathbf{M}_c$, $\mathbf{M}_{tr} \in \mathbb{R}^D$ are the time masks for CNN and Transformer feature respectively. We ablate the dimension of masks in Sec. 4.3 to be $\mathbb{R}^D$ or $\mathbb{R}^1$. We multiply these two masks by the respective features and adjust the fused feature by MLP.

$$\hat{\mathbf{X}}_F = \text{MLP}(\mathbf{X}_c^* \odot \mathbf{M}_c + \mathbf{X}_{tr}^* \odot \mathbf{M}_c), \tag{19}$$

where $\hat{\mathbf{X}}_F \in \mathbb{R}^{M\times D}$ is the final feature adaptively aggregating local and global features based on the timestep.

3.5. Latent Point Cloud Decoder

In order to upsample the latent point cloud and predict the corresponding noise, we design a latent point cloud decoder $\mathcal{D}$, which converts the latent point cloud $\hat{\mathbf{X}}_F \in \mathbb{R}^{M\times D}$ to $\mathbf{X}_t \in \mathbb{R}^{N\times 3}$. Similar to Sec. 3.2, we can voxelize $\hat{\mathbf{X}}_F$ into a latent volume $\hat{\mathbf{V}}_F \in \mathbb{R}^{L\times L\times L\times D}$, and take $\mathbf{X}_t$ as the query for trilinear interpolation, eventually resulting in the final noise prediction $\epsilon_\theta \in \mathbb{R}^{N\times 3}$. This formulation can be expressed as:

$$\epsilon_\theta = \text{Trilinear}(\text{Voxelize}(\hat{\mathbf{X}}_F), \mathbf{X}_t)\mathbf{W}^{D\times 3}, \tag{20}$$

Method	Generative Model	Airplane		Chair		Car	
		CD→ 50%	EMD→ 50%	CD→ 50%	EMD→ 50%	CD→ 50%	EMD→ 50%
r-GAN [1]	GAN	98.40	96.79	83.69	99.70	94.46	99.01
l-GAN(CD) [1]	GAN	87.30	93.95	68.58	83.84	66.49	88.78
l-GAN(EMD) [1]	GAN	89.49	76.91	71.90	64.65	71.16	66.19
PointFlow [51]	Normalizing Flow	75.68	70.74	62.84	60.57	58.10	56.52
DPF-Net [27]	Normalizing Flow	75.18	65.55	62.00	58.53	62.35	54.48
ShapeGF [5]	GAN	80.00	76.17	68.96	65.48	63.20	56.53
SoftFlow [23]	Normalizing Flow	76.05	65.80	59.21	60.05	64.77	60.09
SetVAE [24]	VAE	76.54	67.65	58.84	60.57	59.95	59.94
DPM [34]	Diffusion	76.42	86.91	60.05	74.77	68.89	79.97
PVD [55]	Diffusion	73.82	64.81	56.26	53.32	54.55	53.83
TIGER	Diffusion	71.85	55.82	54.61	52.71	54.31	52.24
LION [53]	Diffusion	67.41	61.23	53.70	52.34	53.41	51.14
TIGER	Diffusion	67.21	56.26	54.32	51.71	54.12	50.24

Table 1. Quantitative comparison with baselines using 1-NN. Both CD and EMD are considered, where CD is multiplied by 10^3 and EMD is multiplied by 10^2. The closer the score is to 50%, the better the quality and diversity of generated samples. [Key: Best, Second Best]

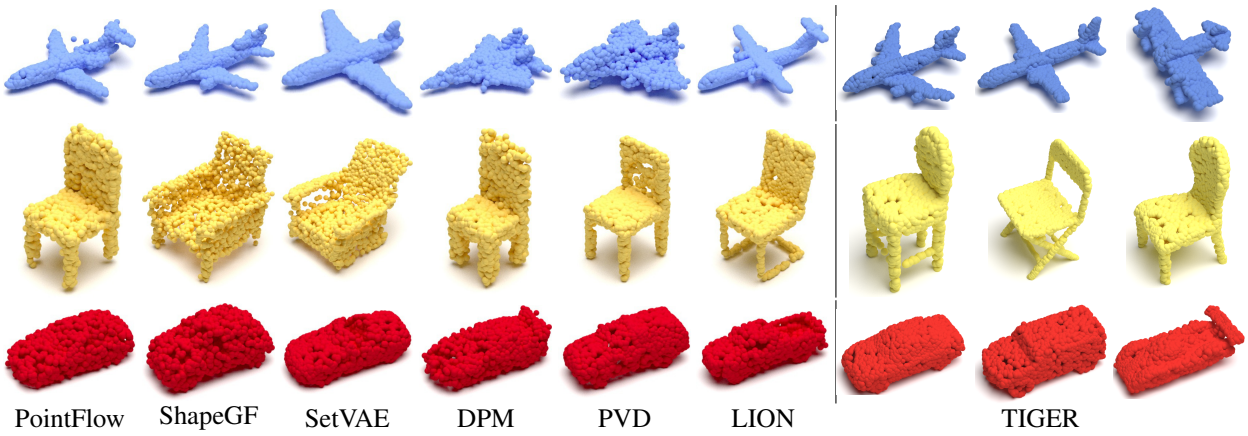


Figure 7. Our generation results (right) compared to baseline models (left). TIGER generates high-quality and diverse 3D point clouds.

where $\mathbf{W}^{D\times 3}$ is the projection matrix to map the noise into 3D space and $\mathbf{X}_t$ is the original point cloud coordinates. The overview of our decoder is shown in Fig. 6.

4. Experiments

4.1. Experimental Setup

Datasets. We follow previous works and choose ShapeNetv2 [6] as our main dataset, which is pre-processed by PointFlow [51]. To fairly compare with other different baseline methods, we train and evaluate on three categories: *airplane*, *chair* and *car*. For each shape, we sample $2,048$ points and normalize them globally across the entire dataset. The training set consists of $2,832$, $4,612$, and $2,458$ shapes, while the evaluation set is composed of 405, 662, and 352 shapes for airplanes, chairs, and cars, respectively.

Metrics. Following the baselines PVD [55] and LION [53], we use 1-NN (1-nearest neighbor) accuracy [32] as our evaluation metric. This metric has been shown to effectively measure both the quality and diversity of generated point clouds and a score closer to 50% indicates superior performance [51]. To calculate the 1-NN distance matrix, we consider two commonly used metrics for measuring the distance between point clouds: CD (Chamfer Distance) and EMD (Earth Movers' Distance).

4.2. Comparison with SoTA methods

Results. To compare our time-varying denoising model with SoTA methods, we evaluate its performance against competitive models such as SetVAE [24], DPM [34], PVD [55] and LION [53]. It is noteworthy that in order to compare with LION, which uses a different dataset splitting strategy (sampling from the first $10,000$ points instead of the latter $5,000$ points), we evaluate our model under both LION's strategy and the standard strategy used by other methods. As shown in Tab. 1, we outperform LION in four out of six metrics. Compared to other methods, our performance is significantly better. Notably, our EMD-based

Method	Training Time (GPU hours)	Inference Time (s)
PVD [55]	142	8.46
LION [53]	550	27.12
Tiger	164	9.73

Table 2. Training and inference time comparison. Training time is counted on average for three categories: airplane, chair and car.

Method	W_1	W_2	Mean CD	Mean EMD
Convolution	1	0	61.34	56.31
Transformer	0	1	62.41	55.85
Conv+Tran	0.5	0.5	64.45	57.92
Ours	learnable (scalar)		60.79	54.12
Ours	learnable (channel-wise)		59.43	53.51

Table 3. Ablation of different architecture designs. W_1 and W_2 are the weights for the ConvNet branch and Transformer branch. [Key: Best, Second Best]

1-NN accuracy shows a significant improvement of 13.45% relative to PVD and 8.12% relative to LION for airplanes. This is attributed to the efficient aggregation of global features by our Transformer branch, which is especially beneficial for EMD as it is primarily influenced by global distribution. Fig. 7 shows that our generated samples are of high quality and diversity compared with other methods.

Efficiency. To compare the training and inference speed, we report the GPU hours with NVidia V100 GPUs as in LION [53]. As shown in Tab. 2, our model trains much faster than LION, with only a quarter of its training time and a third of its inference time. Our training and inference speed is comparable to that of PVD while surpassing LION and PVD in the generation quality as shown in Tab. 1. More computation analysis can be found in Supp.

4.3. Ablation and Analysis

Time-varying two-stream architecture. We ablate our time-varying two-stream architecture by comparing it with single ConvNet, single Transformer, and two-stream networks with equal weight. In this ablation, we also compare the performance of our time masking with scalar value setting and channel-wise value setting. The results, presented in the first two rows of Tab. 3, show that Transformer performs better on the EMD metric, while ConvNet performs better on the CD metric. Using a fixed equal weight for the two-stream network leads to a significant performance drop on both metrics. However, the learnable time mask improves the performance on both metrics, indicating the importance of the time-varying design. Further, channel-wise time masking improves performance slightly more than scale value setting due to its greater expressivity.

Transformer backbones, position encoding, and self-attention strategies. Here we carried out three ablations: (1) We compare the performance of our Transformer backbone with two classic Transformer backbones, PCT [16] and Point Transformer (PointTran) [54]. (2) To assess our proposed position encoding methods, PSPE and BλPE, we compare them with no position encoding and learnable position encoding. (3) We investigate our position-aware self-attention module by comparing it with standard self-attention and self-attention with learnable position encoding as proposed in [54]. As shown in Tab. 4, our latent point cloud Transformer outperforms both PCT and Point Transformer under various settings. Furthermore, our proposed position encoding methods, PSPE and BλPE, significantly improve performance compared to no position encoding or learnable position encoding. Specifically, PSPE improves the CD metric, while BλPE improves the EMD metric. Our position-aware self-attention module, which aggregates corresponding position encoding, achieves better results than standard self-attention and self-attention with learnable position encoding. To illustrate this, Fig. 8 visualizes the attention map in 3D point space for both standard self-attention and our position-aware self-attention. This visualization demonstrates the ability of our method to capture

TB	PE	AP	Mean CD	Mean EMD
PCT [16]	Learnable	-	65.82	60.21
PointTran [54]	Learnable	Learnable	66.42	62.39
Ours	-	-	63.51	57.98
Ours	Learnable	Learnable	62.17	56.13
Ours	PSPE	-	60.11	55.74
Ours	PSPE	Learnable	61.48	55.97
Ours	PSPE	PSPE	59.02	54.49
Ours	BλPE	-	61.35	54.27
Ours	BλPE	Learnable	61.89	56.02
Ours	BλPE	BλPE	59.43	53.51

Table 4. Ablation of Transformer backbones, position encoding, and self-attention strategies. [Key: Best, Second Best; TB=Transformer backbone; PE=position encoding; AP=additional position information]

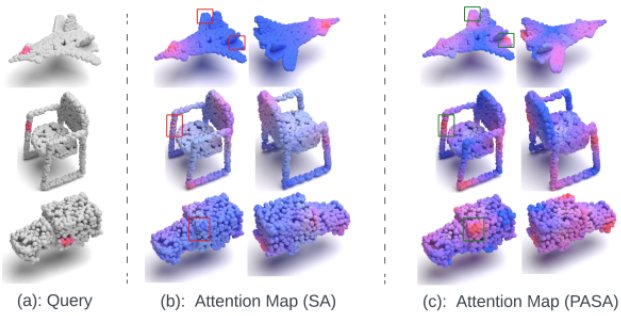
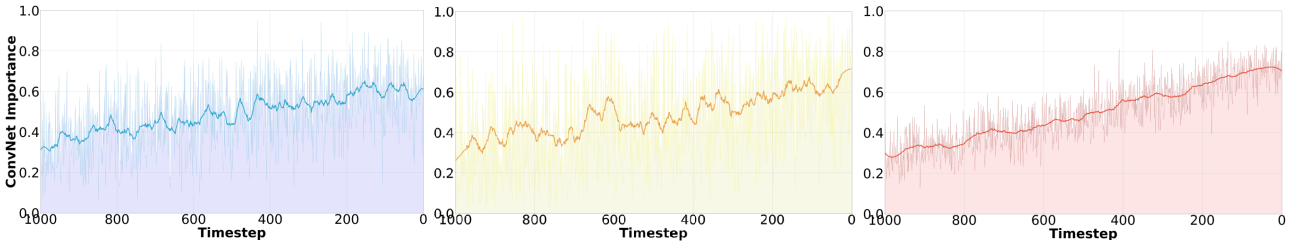


Figure 8. Visualization of attention maps in 3D point clouds. Column (c) is the result of column (b) multiplied by position relationship map **H**. Taking the first row as an example, queried by the airplane head's position, our method can aggregate the position information from the airplane's right wing and left elevator compared with standard self-attention. This comparison highlights the effectiveness of our position-aware self-attention module in capturing long-range relationships in 3D point clouds. [SA: Self-attention, PASA: Position-aware Self-attention]



(a) Airplane (b) Chair (c) Car

Figure 9. The changes of ConvNet importance for each category over timesteps with the channel-wise time mask setting. The thick curve is the result of Gaussian smoothing.

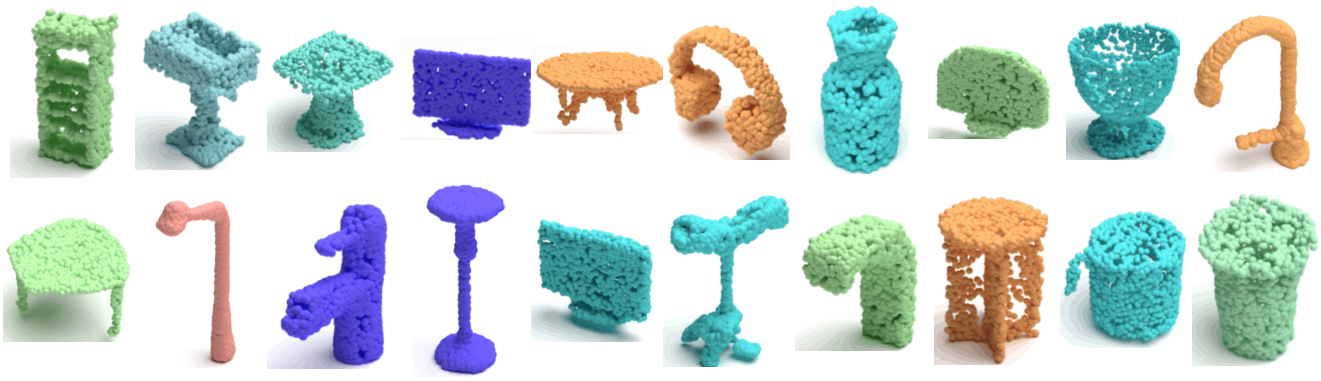


Figure 10. High quality and diverse 3D point clouds generated from our TIGER model trained on 55 ShapeNetv2 categories.

long-range relationships in the data.

Weight changing over timesteps. To validate our motivation of reweighing the local and global features from ConvNet and Transformer at different time steps, we visualize the changes in weights. For channel-wise time mask setting, we use the ratio of the weight of the local feature over the weight of the global feature to measure the importance of ConvNet and Transformer at each timestep. Fig. 9 indicates that the importance of ConvNet for each category increases nearly monotonically, implying that the diffusion process of 3D point cloud generation benefits from global features at the early stages of diffusion and requires the aggregation of more local features at later stages. Moreover, we observe that the curves of airplanes and chairs change more drastically than that of cars, which we hypothesize is due to the complicated shapes of airplanes and chairs, resulting in a more complex diffusion process.

Unconditional generation of multiple classes. To further demonstrate the generalizability of our model, we train a universal TIGER model on all 55 categories of ShapeNetv2 [6]. Fig. 10 shows that our universal model can produce shapes with high quality and diversity.

5. Conclusions

In this work, we propose a novel time-varying denoising model with a latent point transformer backbone. Experimentally, we show how the time mask optimally aggregates local and global features at different timesteps, which quantitatively reaches SoTA performance. In accordance with our hypothesis, Transformer is more important at early diffusing stages and ConvNet has a larger weight at later timesteps in the 3D point cloud generation task. Future work will be focused on exploring how our time-varying denoising model would behave in other diffusion-based tasks.

Limitations. Although we generate high-quality and natural samples, we cannot control the category of the generated shape. One potential solution is to encode the category into latent space and train the diffusion model conditioned on the latent code. We also acknowledge that our model needs to get the local feature and global feature by two subnetworks, which is not quite efficient. In this work, we gain efficiency with the model converging with fewer epochs. But future works can increase the backbone efficiency by proposing time-varying properties with only one network.

Potential Negative Impacts. While it has the potential to speed up progress in both areas, it also has the risk of eliminating the jobs of game and furniture designers due to automation. However, our model should have a mostly positive impact, because the most likely use of our method would be to assist designers not completely replace their jobs. It is still necessary to select suitable shapes among many generated samples.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018. 1, 2, 6

[2] Vishal Asnani, John Collomosse, Tu Bui, Xiaoming Liu, and Shruti Agarwal. ProMark: Proactive diffusion watermarking for causal attribution. In *CVPR*, 2024. 3

[3] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. 5

[4] Fan Bao, Kun Xu, Chongxuan Li, Lanqing Hong, Jun Zhu, and Bo Zhang. Variational (gradient) estimate of the score function in energy-based latent variable models. In *ICML*, 2021. 3

[5] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *ECCV*, 2020. 1, 2, 6

[6] 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, 8

[7] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *NeurIPS*, 31, 2018. 2

[8] Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. Diffusiondet: Diffusion model for object detection. *arXiv preprint arXiv:2211.09788*, 2022. 3

[9] Tamal K Dey and James Hudson. Pmr: Point to mesh rendering, a feature-based approach. In *VIS*, 2002. 1

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2, 4

[11] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y Zeevi. The farthest point strategy for progressive image sampling. In *ICPR*, 1994. 3, 4

[12] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution tree networks for 3d point cloud processing. In *ECCV*, 2018. 2

[13] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018. 2

[14] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3D surface generation. In *CVPR*, 2018. 2

[15] Jia Guo, Xuanxia Yao, Mengyu Shen, Jiafei Wang, and Wanyou Liao. A deep learning network for point cloud of medicine structure. In *ITME*, 2018. 1

[16] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7:187–199, 2021. 3, 7

[17] Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. *arXiv preprint arXiv:2210.01549*, 2022. 3

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 1, 2, 3, 5

[19] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 3

[20] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022. 3

[21] Zitian Huang, Yikuan Yu, Jiawen Xu, Feng Ni, and Xinyi Le. Pf-net: Point fractal network for 3d point cloud completion. In *CVPR*, 2020. 1

[22] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 2005. 2

[23] Hyeongju Kim, Hyeonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim. Softflow: Probabilistic framework for normalizing flow on manifolds. In *NeurIPS*, 2020. 1, 6

[24] Jinwoo Kim, Jaehoon Yoo, Juho Lee, and Seunghoon Hong. Setvae: Learning hierarchical composition for generative modeling of set-structured data. In *CVPR*, 2021. 1, 6

[25] Minchul Kim, Feng Liu, Anil Jain, and Xiaoming Liu. DCFace: Synthetic face generation with dual condition diffusion model. In *CVPR*, 2023. 3

[26] Minchul Kim, Yiyang Su, Feng Liu, Anil Jain, and Xiaoming Liu. Keypoint relative position encoding for face recognition. In *CVPR*, 2024. 4

[27] Roman Klokov, Edmond Boyer, and Jakob Verbeek. Discrete point flow networks for efficient point cloud generation. In *ECCV*, 2020. 1, 2, 6

[28] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020. 3

[29] Manoj Kumar, Mostafa Dehghani, and Neil Houlsby. Dual patchnorm. *arXiv preprint arXiv:2302.01327*, 2023. 4

[30] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel CNN for efficient 3d deep learning. In *NeurIPS*, 2019. 2, 3, 4, 5

[31] Yunfei Long, Daniel Morris, Xiaoming Liu, Marcos Paul Gerardo Castro, Punarjay Chakravarty, and Praveen Narayanan. Radar-camera pixel depth association for depth completion. In *CVPR*, 2021. 1

[32] David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545*, 2016. 6

[33] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *arXiv preprint arXiv:2206.00927*, 2022. 3

[34] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *CVPR*, 2021. 1, 2, 3, 6

[35] Peiyuan Ni, Wenguang Zhang, Xiaoxiao Zhu, and Qixin Cao. Pointnet++ grasping: Learning an end-to-end spatial grasp generation algorithm from sparse point clouds. In *ICRA*, 2020. 1

[36] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. 1

[37] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh,

Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: towards photorealistic image generation and editing with text-guided diffusion models. In *ICML*, 2022. 3

[38] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2

[39] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 2

[40] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. In *ICLRW*, 2018. 4

[41] Zhiyuan Ren, Zhihong Pan, Xin Zhou, and Le Kang. Diffusion motion: Generate text-guided 3d human motion by diffusion model. *arXiv preprint arXiv:2210.12315*, 2022. 3

[42] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 3

[43] Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. Step-unrolled denoising autoencoders for text generation. *arXiv preprint arXiv:2112.06749*, 2021. 3

[44] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3D object proposal generation and detection from point cloud. In *CVPR*, 2019. 2

[45] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 3

[46] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *NeurIPS*, 2019. 2

[47] Haoru Tan, Sitong Wu, and Jimin Pi. Semantic diffusion network for semantic segmentation. *arXiv preprint arXiv:2302.02057*, 2023. 3

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2, 3, 5

[49] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. In *NeurIPS*, 2022. 3

[50] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 4

[51] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *ICCV*, 2019. 1, 2, 6

[52] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *CVPR*, 2018. 1

[53] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. LION: Latent point diffusion models for 3d shape generation. In *NeurIPS*, 2022. 1, 2, 3, 6, 7

[54] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 3, 5, 7

[55] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *ICCV*, 2021. 1, 2, 3, 6, 7