Diffusion Probabilistic Models for 3D Point Cloud Generation

Shitong Luo, Wei Hu *
Wangxuan Institute of Computer Technology
Peking University
{luost, forhuwei}@pku.edu.cn

Abstract

We present a probabilistic model for point cloud generation, which is fundamental for various 3D vision tasks such as shape completion, upsampling, synthesis and data augmentation. Inspired by the diffusion process in non-equilibrium thermodynamics, we view points in point clouds as particles in a thermodynamic system in contact with a heat bath, which diffuse from the original distribution to a noise distribution. Point cloud generation thus amounts to learning the reverse diffusion process that transforms the noise distribution to the distribution of a desired shape. Specifically, we propose to model the reverse diffusion process for point clouds as a Markov chain conditioned on certain shape latent. We derive the variational bound in closed form for training and provide implementations of the model. Experimental results demonstrate that our model achieves competitive performance in point cloud generation and auto-encoding. The code is available at `https://github.com/luost26/diffusion-point-cloud`.

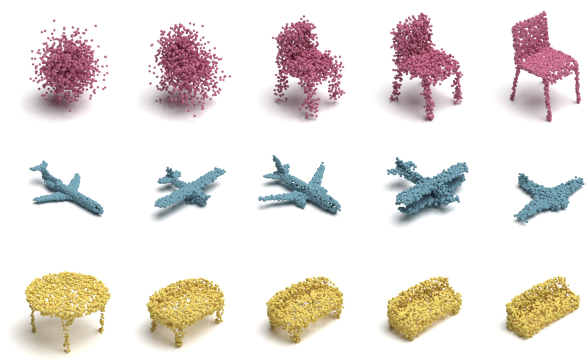

Figure 1. **Top**: The diffusion process that converts noise to some shape (left to right). **Middle**: Generated point clouds from the proposed model. **Bottom**: Latent space interpolation between the two point clouds at both ends.

1. Introduction

With recent advances in depth sensing and laser scanning, point clouds have attracted increasing attention as a popular representation for modeling 3D shapes. Significant progress has been made in developing methods for point cloud analysis, such as classification and segmentation [16, 17, 23]. On the other hand, learning generative models for point clouds is powerful in unsupervised representation learning to characterize the data distribution, which lays the foundation for various tasks such as shape completion, upsampling, synthesis, *etc*.

Generative models such as variational auto-encoders (VAEs), generative adversarial networks (GANs), normalizing flows, *etc*., have shown great success in image generation [13, 8, 5, 6]. However, these powerful tools cannot be directly generalized to point clouds, due to the irregular sampling patterns of points in the 3D space in contrast to regular grid structures underlying images. Hence, learning generative models for point clouds is quite challenging. Prior research has explored point cloud generation via GANs [1, 22, 19], auto-regressive models [21], flow-based models [25] and so on. While remarkable progress has been made, these methods have some inherent limitations for modeling point clouds. For instance, the training procedure could be unstable for GANs due to the adversarial losses. Auto-regressive models assume a generation ordering which is unnatural and might restrict the model's flexibility.

In this paper, we propose a probabilistic generative model for point clouds inspired by non-equilibrium thermodynamics, exploiting the *reverse diffusion process* to learn the point distribution. As a point cloud is composed of discrete points in the 3D space, we regard these points as particles in a non-equilibrium thermodynamic system in contact with a heat bath. Under the effect of the heat bath, the position of particles evolves stochastically in the way that they

*Corresponding author: Wei Hu (forhuwei@pku.edu.cn). This work was supported by the National Key R&D project of China under contract No. 2019YFF0302903 and National Natural Science Foundation of China under contract No. 61972009.

diffuse and eventually spread over the space. This process is termed the *diffusion process* that converts the initial distribution of the particles to a simple noise distribution by adding noise at each time step [12, 20]. Analogously, we connect the point distribution of point clouds to a noise distribution via the diffusion process. Naturally, in order to model the point distribution for point cloud generation, we consider *the reverse diffusion process*, which recovers the target point distribution from the noise distribution.

In particular, we model this reverse diffusion process as a Markov chain that converts the noise distribution into the target distribution. Our goal is to learn its transition kernel such that the Markov chain can reconstruct the desired shape. Further, as the purpose of the Markov chain is modeling the point distribution, the Markov chain alone is incapable to generate point clouds of various shapes. To this end, we introduce a *shape latent* as the condition for the transition kernel. In the setting of generation, the shape latent follows a prior distribution which we parameterize via normalizing flows [5, 6] for strong model expressiveness. In the setting of auto-encoding, the shape latent is learned end-to-end. Finally, we formulate the training objective as maximizing the variational lower bound of the likelihood of the point cloud conditional on the shape latent, which is further formulated into tractable expressions in closed form. We apply our model to point cloud generation, auto-encoding and unsupervised representation learning, and results demonstrate that our model achieves competitive performance on point cloud generation and auto-encoding and comparable results on unsupervised representation learning.

Our main contributions include:

- We propose a novel probabilistic generative model for point clouds, inspired by the diffusion process in non-equilibrium thermodynamics.
- We derive a tractable training objective from the variational lower bound of the likelihood of point clouds conditioned on some shape latent.
- Extensive experiments show that our model achieves competitive performance in point cloud generation and auto-encoding.

2. Related Works

Point Cloud Generation Early point cloud generation methods [1, 7] treat point clouds as $N \times 3$ matrices, where N is the fixed number of points, converting the point cloud generation problem to a matrix generation problem, so that existing generative models are readily applicable. For example, [7] apply variational auto-encoders [13] to point cloud generation. [1] employ generative adversarial networks [8] based on a pre-trained auto-encoder. The main defect of these methods is that they are restricted to generating point clouds with a fixed number of points, and lack the property of permutation invariance. FoldingNet and AtlasNet [26, 10] mitigate this issue by learning a mapping from 2D patches to the 3D space, which deforms the 2D patches into the shape of point clouds. These two methods allow generating arbitrary number of points by first sampling some points on the patches and then applying the mapping on them. In addition, the points on the patches are inherently invariant to permutation.

The above methods rely on heuristic set distances such as the Chamfer distance (CD) and the Earth Mover's distance (EMD). As pointed out in [25], CD has been shown to incorrectly favor point clouds that are overly concentrated in the mode of the marginal point distribution, and EMD is slow to compute while approximations could lead to biased gradients.

Alternatively, point clouds can be regarded as samples from a point distribution. This viewpoint inspires exploration on applying likelihood-based methods to point cloud modeling and generation. PointFlow [25] employs continous normalizing flows [4, 9] to model the distribution of points. DPF-Net [14] uses affine coupling layers as the normalizing flow to model the distribution. PointGrow [21] is an auto-regressive model with exact likelihoods. More recently, [2] proposed a score-matching energy-based model ShapeGF to model the distribution of points.

Our method also regards point clouds as samples from a distribution, but differs in the probabilistic model compared to prior works. We leverage the reverse diffusion Markov chain to model the distribution of points, achieving both simplicity and flexibility. Specifically, the training process of our model involves learning the Markov transition kernel, whose training objective has a simple function form. By contrast, GAN-based models involve complex adversarial losses, continuous-flow-based methods involve expensive ODE integration. In addition, our model is flexible, because it does not require invertibility in contrast to flow-based models, and does not assume ordering compared to auto-regressive models.

Diffusion Probabilistic Models The diffusion process considered in this work is related to the diffusion probabilistic model [20, 11]. Diffusion probabilistic models are a class of latent variable models, which also use a Markov chain to convert the noise distribution to the data distribution. Prior research on diffusion probabilistic models focuses on the unconditional generation problem for toy data and images. In this work, we focus on point cloud generation, which is a *conditional* generation problem, because the Markov chain considered in our work generates points of a point cloud conditioned on some shape latent. This conditional adaptation leads to significantly different training and sampling schemes compared to prior research on diffusion probabilistic models.

3. Diffusion Probabilistic Models for Point Clouds

In this section, we first formulate the probabilistic model of both the forward and the reverse diffusion processes for point clouds. Then, we formulate the objective for training the model. The implementation of the model will be provided in the next section.

3.1. Formulation

We regard a point cloud $\boldsymbol{X}^{(0)} = \{\boldsymbol{x}_i^{(0)}\}_{i=1}^N$ consisting of N points as a set of particles in an evolving thermodynamic system. As discussed in Section 1, each point $\boldsymbol{x}_i$ in the point cloud can be treated as being sampled independently from a point distribution, which we denote as $q(\boldsymbol{x}_i^{(0)}|\boldsymbol{z})$. Here, $\boldsymbol{z}$ is the shape latent that determines the distribution of points.

Physically, as time goes by, the points gradually diffuse into a chaotic set of points. This process is termed the *diffusion process*, which converts the original meaningful point distribution into a noise distribution. The forward diffusion process is modeled as a Markov chain [12]:

$$q(\boldsymbol{x}_i^{(1:T)}|\boldsymbol{x}_i^{(0)}) = \prod_{t=1}^{T} q(\boldsymbol{x}_i^{(t)}|\boldsymbol{x}_i^{(t-1)}), \tag{1}$$

where $q(\boldsymbol{x}_i^{(t)}|\boldsymbol{x}_i^{(t-1)})$ is the Markov diffusion kernel. The kernel adds noise to points at the previous time step and models the distribution of points at the next time step. Following the convention of [20], we define the diffusion kernel as:

$$q(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(t-1)}) = \mathcal{N}(\boldsymbol{x}^{(t)}|\sqrt{1-\beta_t}\boldsymbol{x}^{(t-1)}, \beta_t \boldsymbol{I}), t = 1, ..., T, \tag{2}$$

where $\beta_1 \dots \beta_T$ are variance schedule hyper-parameters that control the diffusion rate of the process.

Our goal is to generate point clouds with a meaningful shape, encoded by the latent representation $\boldsymbol{z}$. We treat the generation process as the reverse of the diffusion process, where points sampled from a simple noise distribution $p(\boldsymbol{x}_i^{(T)})$ that approximates $q(\boldsymbol{x}_i^{(T)})$ are given as the input. Then, the points are passed through the reverse Markov chain and finally form the desired shape. Unlike the forward diffusion process that simply adds noise to the points, the reverse process aims to recover the desired shape from the input noise, which requires training from data. We formulate the reverse diffusion process for generation as:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(0:T)}|\boldsymbol{z}) = p(\boldsymbol{x}^{(T)}) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(t-1)}|\boldsymbol{x}^{(t)}, \boldsymbol{z}), \tag{3}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(t-1)}|\boldsymbol{x}^{(t)}, \boldsymbol{z}) = \mathcal{N}\big(\boldsymbol{x}^{(t-1)}\big|\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}^{(t)}, t, \boldsymbol{z}), \beta_t \boldsymbol{I}\big), \tag{4}$$

where $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ is the estimated mean implemented by a neural network parameterized by $\boldsymbol{\theta}$. $\boldsymbol{z}$ is the latent encoding the target shape of the point cloud. The starting distribution $p(\boldsymbol{x}_i^{(T)})$ is set to a standard normal distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Given a shape latent $\boldsymbol{z}$, we obtain the point cloud with the target shape by passing a set of points sampled from $p(\boldsymbol{x}_i^{(T)})$ through the reverse Markov chain.

For the sake of brevity, in the following sections, we use the distribution with respect to the entire point cloud $\boldsymbol{X}^{(0)}$. Since the points in a point cloud are independently sampled from a distribution, the probability of the whole point cloud is simply the product of the probability of each point:

$$q(\boldsymbol{X}^{(1:T)}|\boldsymbol{X}^0) = \prod_{i=1}^{N} q(\boldsymbol{x}_i^{(1:T)}|\boldsymbol{x}_i^{(0)}), \tag{5}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(0:T)}|\boldsymbol{z}) = \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(0:T)}|\boldsymbol{z}). \tag{6}$$

Having formulated the forward and reverse diffusion processes for point clouds, we will formalize the training objective of the reverse diffusion process for point cloud generation as follows.

3.2. Training Objective

The goal of training the reverse diffusion process is to maximize the log-likelihood of the point cloud: $\mathbb{E}[\log p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(0)})]$. However, since directly optimizing the exact log-likelihood is intractable, we instead *maximize* its variational lower bound:

$$\begin{aligned}\mathbb{E}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(0)})\big] &\geq \mathbb{E}_q\Big[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(0:T)}, \boldsymbol{z})}{q(\boldsymbol{X}^{(1:T)}, \boldsymbol{z}|\boldsymbol{X}^{(0)})}\Big] \\ &= \mathbb{E}_q\Big[\log p(\boldsymbol{X}^{(T)}) \\ &\qquad + \sum_{t=1}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{z})}{q(\boldsymbol{X}^{(t)}|\boldsymbol{X}^{(t-1)})} \\ &\qquad - \log \frac{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)})}{p(\boldsymbol{z})}\Big].\end{aligned} \tag{7}$$

The above variational bound can be adapted into the training objective L to be *minimized* (the detailed derivation is provided in the supplementary material):

$$\begin{aligned}L(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \mathbb{E}_q\Big[&\sum_{t=2}^{T} D_{\mathrm{KL}}\big(q(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{X}^{(0)}) \| \\ &\qquad p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{z})\big) \\ &- \log p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(0)}|\boldsymbol{X}^{(1)}, \boldsymbol{z}) \\ &+ D_{\mathrm{KL}}\big(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)}) \| p(\boldsymbol{z})\big)\Big].\end{aligned} \tag{8}$$

Since the distributions of points are independent of each other as described in Eq. (5), we further expand the training

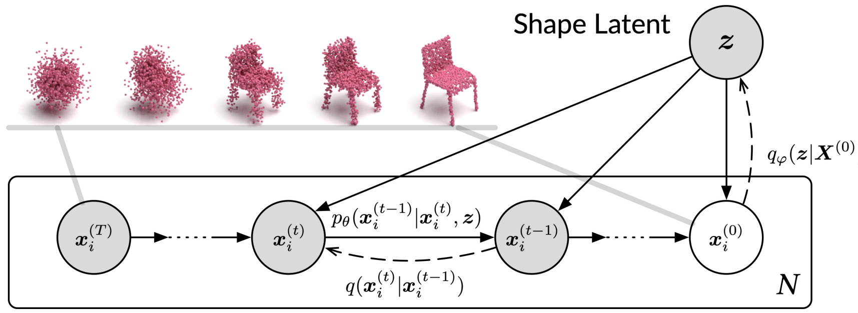

Figure 2. The directed graphical model of the diffusion process for point clouds. N is the number of points in the point cloud $\boldsymbol{X}^{(0)}$.

objective:

$$
\begin{aligned}
L(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \mathbb{E}_q \Big[\sum_{t=2}^{T} \sum_{i=1}^{N} D_{\mathrm{KL}} \big(& \underbrace{q(\boldsymbol{x}_i^{(t-1)}|\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_i^{(0)})}_{①} \, \| \\
& \underbrace{p_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(t-1)}|\boldsymbol{x}_i^{(t)}, \boldsymbol{z}))}_{②} \\
- \sum_{i=1}^{N} & \underbrace{\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(0)}|\boldsymbol{x}_i^{(1)}, \boldsymbol{z})}_{③} \\
+ D_{\mathrm{KL}} \big(& \underbrace{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)})}_{④} \, \| \, \underbrace{p(\boldsymbol{z})}_{⑤} \big) \Big].
\end{aligned}
\tag{9}
$$

The training objective can be optimized efficiently since each of the terms on the right hand side is tractable and q is easy to sample from the forward diffusion process. Next, we elaborate on the terms to reveal how to compute the objective.

① $q(\boldsymbol{x}_i^{(t-1)}|\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_i^{(0)})$ can be computed with the following closed-form Gaussian according to [11]:

$$
q(\boldsymbol{x}_i^{(t-1)}|\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_i^{(0)}) = \mathcal{N}(\boldsymbol{x}_i^{(t-1)}|\boldsymbol{\mu}_t(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(0)}), \gamma_t \boldsymbol{I}),
\tag{10}
$$

where, using the notation $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$:

$$
\begin{aligned}
\boldsymbol{\mu}_t(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(0)}) &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x}^{(0)} + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}^{(t)}, \\
\gamma_t &= \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t.
\end{aligned}
\tag{11}
$$

②, ③ $p_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(t-1)}|\boldsymbol{x}_i^{(t)}, \boldsymbol{z})(t = 1, \ldots, T)$ are trainable Gaussians as described in Eq. (4).

④ $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)})$ is the approximate posterior distribution. Using the language of variational auto-encoders, $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)})$ is an encoder that encodes the input point cloud $\boldsymbol{X}^{(0)}$ into the distribution of the latent code $\boldsymbol{z}$. We assume it as a Gaussian following the convention:

$$
q(\boldsymbol{z}|\boldsymbol{X}^{(0)}) = \mathcal{N}\big(\boldsymbol{z}|\boldsymbol{\mu}_{\boldsymbol{\varphi}}(\boldsymbol{X}^{(0)}), \boldsymbol{\Sigma}_{\boldsymbol{\varphi}}(\boldsymbol{X}^{(0)})\big).
\tag{12}
$$

⑤ The last term $p(\boldsymbol{z})$ is the prior distribution. The most common choice of $p(\boldsymbol{z})$ is the isotropic Gaussian $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. In addition to a fixed distribution, the prior can be a trainable parametric distribution, which is more flexible. For example, normalizing flows [5, 6] can be employed to parameterize the prior distribution.

In the following section, we show how to optimize the objective in Eq. (9) in order to train the model.

3.3. Training Algorithm

In principle, training the model amounts to minimizing the objective in Eq. (9). However, evaluating Eq. (9) requires summing the expectation of the KL terms over all the time steps, which involves sampling a full trajectory $\boldsymbol{x}_i^{(1)}, \ldots, \boldsymbol{x}_i^{(T)}$ from the forward diffusion process in order to compute the expectation.

To make the training simpler and more efficient, following [11], instead of evaluating the expectation of the whole summation over all the time steps in Eq. (9), we randomly choose one term from the summation to optimize at each training step.

Specifically, this simplified training algorithm is as follows:

Algorithm 1 Training (Simplified)

repeat
  Sample $\boldsymbol{X}^{(0)} \sim q_{\mathrm{data}}(\boldsymbol{X}^{(0)})$
  Sample $\boldsymbol{z} \sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)})$
  Sample $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
  Sample $\boldsymbol{x}_1^{(t)}, \ldots, \boldsymbol{x}_N^{(t)} \sim q(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(0)})$
  $L_t \leftarrow \sum_{i=1}^{N} D_{\mathrm{KL}}\left(q(\boldsymbol{x}_i^{(t-1)}|\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_i^{(0)}) \middle\| p_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(t-1)}|\boldsymbol{x}_i^{(t)}, \boldsymbol{z})\right)$
  $L_{\boldsymbol{z}} \leftarrow D_{\mathrm{KL}}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)}) \| p(\boldsymbol{z}))$
  Compute $\nabla_{\boldsymbol{\theta}}(L_t + \frac{1}{T}L_{\boldsymbol{z}})$. Then perform gradient descent.
until converged

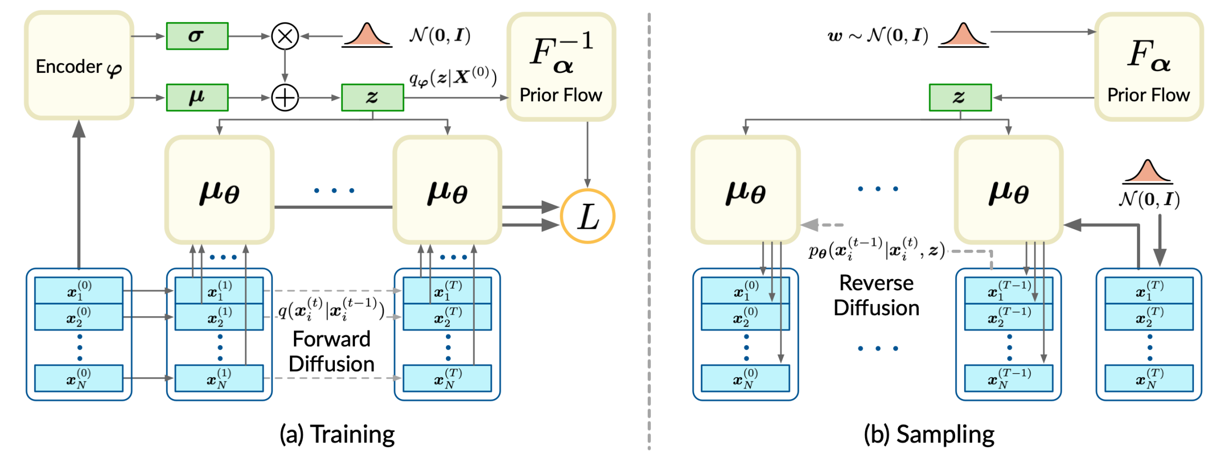

Figure 3. The illustration of the proposed model. (a) illustrates how the objective is computed during the training process. (b) illustrates the generation process.

To efficiently sample from $q(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(0)})$ (5th statement) and avoid iterative sampling starting from $t = 1$, we leverage on the result in [11], which shows that $q(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(0)})$ is a Gaussian:

$$q(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(0)}) = \mathcal{N}(\boldsymbol{x}^{(t)}|\sqrt{\bar{\alpha}_t}\boldsymbol{x}^{(0)}, (1-\bar{\alpha}_t)\boldsymbol{I}). \tag{13}$$

The gaussianity of $q(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(0)})$ makes further simplification on L_t (6th statement) possible by using the reparameterization trick [13]. We put the detail of this simplification to the supplementary material. Last, note that the KL divergence in $L_{\boldsymbol{z}}$ is evaluated stochastically by $-\mathbb{E}_{\boldsymbol{z}\sim q_{\varphi}(\boldsymbol{z}|\boldsymbol{X}^{(0)})}\big[p(\boldsymbol{z})\big] - H\big[q_{\varphi}(\boldsymbol{z}|\boldsymbol{X}^{(0)})\big]$.

4. Model Implementations

The general training objective and algorithm in the previous section lay the foundation for the formulation of specific point cloud tasks. Next, we adapt the training objective to point cloud generation and point cloud auto-encoding respectively.

4.1. Point Cloud Generator

Leveraging on the model in Section 3, we propose a probabilistic model for point cloud generation by employing normalizing flows to parameterize the prior distribution $p(\boldsymbol{z})$, which makes the model more flexible [18, 5].

Specifically, we use a stack of affine coupling layers [6] as the normalizing flow. In essence, the affine coupling layers provide a trainable bijector $F_{\boldsymbol{\alpha}}$ that maps an isotropic Gaussian to a complex distribution. Since the mapping is bijective, the exact probability of the target distribution can be computed by the change-of-variable formula:

$$p(\boldsymbol{z}) = p_{\boldsymbol{w}}(\boldsymbol{w}) \cdot \left| \det \frac{\partial F_{\boldsymbol{\alpha}}}{\partial \boldsymbol{w}} \right|^{-1} \quad \text{where} \quad \boldsymbol{w} = F_{\boldsymbol{\alpha}}^{-1}(\boldsymbol{z}). \tag{14}$$

Here, $p(\boldsymbol{z})$ is the prior distribution in the model, $F_{\boldsymbol{\alpha}}$ is the trainable bijector implemented by the affine coupling layers, and $p_{\boldsymbol{w}}(\boldsymbol{w})$ is the isotropic Gaussian $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$.

As for the encoder $q_{\varphi}(\boldsymbol{z}|\boldsymbol{X}^{(0)})$, we adopt PointNet [16] as the architecture for $\boldsymbol{\mu}_{\varphi}$ and $\boldsymbol{\Sigma}_{\varphi}$ of the encoder $q_{\varphi}(\boldsymbol{z}|\boldsymbol{X}^{(0)})$.

Substituting Eq. (14) into Eq. (9), the training objective for the generative model is:

$$\begin{aligned} L_G(\boldsymbol{\theta}, \varphi, \boldsymbol{\alpha}) = \mathbb{E}_q \Big[& \sum_{t=2}^{T} \sum_{i=1}^{N} D_{\mathrm{KL}}\big(q(\boldsymbol{x}_i^{(t-1)}|\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_i^{(0)}) \| \\ & \qquad p_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(t-1)}|\boldsymbol{x}_i^{(t)}, \boldsymbol{z})\big) \\ & - \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(0)}|\boldsymbol{x}_i^{(1)}, \boldsymbol{z}) \\ & + D_{\mathrm{KL}}\Big(q_{\varphi}(\boldsymbol{z}|\boldsymbol{X}^{(0)}) \Big\| p_{\boldsymbol{w}}(\boldsymbol{w}) \cdot \left| \det \frac{\partial F_{\boldsymbol{\alpha}}}{\partial \boldsymbol{w}} \right|^{-1}\Big)\Big]. \end{aligned} \tag{15}$$

The algorithm for optimizing the above objective can be naturally derived from Algorithm 1.

To sample a point cloud, we first draw $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and pass it through $F_{\boldsymbol{\alpha}}$ to acquire the shape latent $\boldsymbol{z}$. Then, with the shape latent $\boldsymbol{z}$, we sample some points $\{\boldsymbol{x}_i^{(T)}\}$ from the noise distribution $p(\boldsymbol{x}^{(T)})$ and pass the points through the reverse Markov chain $p_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(0:T)}|\boldsymbol{z})$ defined in Eq. (3) to generate the point cloud $\boldsymbol{X}^{(0)} = \{\boldsymbol{x}_i^{(0)}\}_{i=1}^{N}$.

4.2. Point Cloud Auto-Encoder

We implement a point cloud auto-encoder based on the probabilistic model in Section 3. We employ the PointNet as the representation encoder, denoted as $E_{\varphi}(\boldsymbol{X}^{(0)})$ with parameters φ, and leverage the reverse diffusion process

Table 1. Comparison of point cloud generation performance. CD is multiplied by 10^3, EMD is multiplied by 10^1, and JSD is multiplied by 10^3.

		MMD (↓)		COV (%, ↑)		1-NNA (%, ↓)		JSD (↓)
Shape	Model	CD	EMD	CD	EMD	CD	EMD	-
Airplane	PC-GAN [1]	3.819	1.810	42.17	13.84	77.59	98.52	6.188
	GCN-GAN [22]	4.713	1.650	39.04	18.62	89.13	98.60	6.669
	TreeGAN [19]	4.323	1.953	39.37	8.40	83.86	99.67	15.646
	PointFlow [25]	3.688	1.090	44.98	44.65	66.39	**69.36**	1.536
	ShapeGF [2]	3.306	**1.027**	**50.41**	**47.12**	**61.94**	70.51	**1.059**
	Ours	**3.276**	1.061	48.71	45.47	64.83	75.12	1.067
	Train	3.917	1.003	51.73	54.04	48.85	50.82	0.809
Chair	PC-GAN [1]	13.436	3.104	46.23	22.14	69.67	100.00	6.649
	GCN-GAN [22]	15.354	2.213	39.84	35.09	77.86	95.80	21.708
	TreeGAN [19]	14.936	3.613	38.02	6.77	74.92	100.00	13.282
	PointFlow [25]	13.631	1.856	41.86	43.38	66.13	68.40	12.474
	ShapeGF [2]	13.175	1.785	48.53	46.71	**56.17**	**62.69**	**5.996**
	Ours	**12.276**	**1.784**	**48.94**	**47.52**	60.11	69.06	7.797
	Train	13.954	1.756	53.29	54.90	49.14	48.28	3.602

presented in Section 3.1 for decoding, conditioned on the latent code produced by the encoder.

Leveraging on Eq. (9), we train the auto-encoder by minimizing the following adapted objective:

$$L(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \mathbb{E}_q \Big[\sum_{t=2}^{T} \sum_{i=1}^{N} D_{\mathrm{KL}} \big(q(\boldsymbol{x}_i^{(t-1)} | \boldsymbol{x}_i^{(t)}, \boldsymbol{x}_i^{(0)}) \| \\ p_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(t-1)} | \boldsymbol{x}_i^{(t)}, E_{\boldsymbol{\varphi}}(\boldsymbol{X}^{(0)})) \big) \\ - \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(0)} | \boldsymbol{x}_i^{(1)}, E_{\boldsymbol{\varphi}}(\boldsymbol{X}^{(0)})) \Big]. \tag{16}$$

To decode a point cloud that is encoded as the latent code $\boldsymbol{z}$, we sample some points $\{\boldsymbol{x}_i^{(T)}\}$ from the noise distribution $p(\boldsymbol{x}_i^{(T)})$ and pass the points through the reverse Markov chain $p_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(0:T)} | \boldsymbol{z})$ defined in Eq. (3) to acquire the reconstructed point cloud $\boldsymbol{X}^{(0)} = \{\boldsymbol{x}_i^{(0)}\}_{i=1}^{N}$.

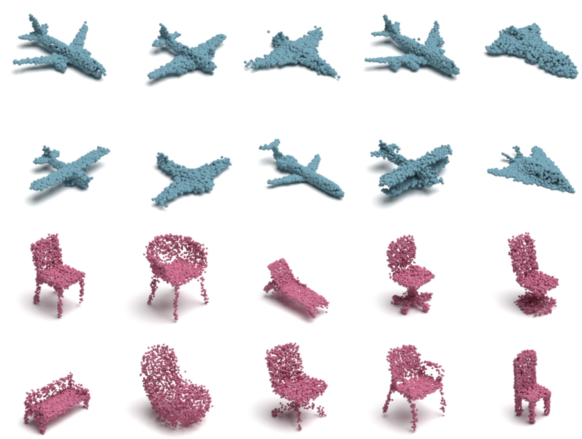

Figure 4. Examples of point clouds generated by our model.

5. Experiments

In this section, we evaluate our model's performance on three tasks: point cloud generation, auto-encoding, and unsupervised representation learning.

5.1. Experimental Setup

Datasets For generation and auto-encoding tasks, we employ the ShapeNet dataset [3] containing 51,127 shapes from 55 categories. The dataset is randomly split into training, testing and validation sets by the ratio 80%, 15%, 5% respectively. For the representation learning task, we use the training split of ShapeNet to train an encoder. Then we adopt ModelNet10 and ModelNet40 [24] to evaluate the representations learned by the encoder. We sample 2048 points from each of the shape to acquire the point clouds and normalize each of them to zero mean and unit variance.

Evaluation Metrics Following prior works, we use the Chamfer Distance (CD) and the Earth Mover's Distance

Table 2. Comparison of point cloud auto-encoding performance. Atlas (S1) and Atlas (P25) denote 1-sphere and 25-square variants of AtlasNet respectively. CD is multiplied by 10^3 and EMD is multiplied by 10^2.

Dataset	Metric	Atlas (S1)	Altas (P25)	PointFlow	ShapeGF	Ours	Oracle
Airplane	CD	2.000	**1.795**	2.420	2.102	2.118	1.016
	EMD	4.311	4.366	3.311	3.508	**2.876**	2.141
Car	CD	6.906	6.503	5.828	**5.468**	5.493	3.917
	EMD	5.617	5.408	4.390	4.489	**3.937**	3.246
Chair	CD	5.479	**4.980**	6.795	5.146	5.677	3.221
	EMD	5.550	5.282	5.008	4.784	**4.153**	3.281
ShapeNet	CD	5.873	5.420	7.550	5.725	**5.252**	3.074
	EMD	5.457	5.599	5.172	5.049	**3.783**	3.112

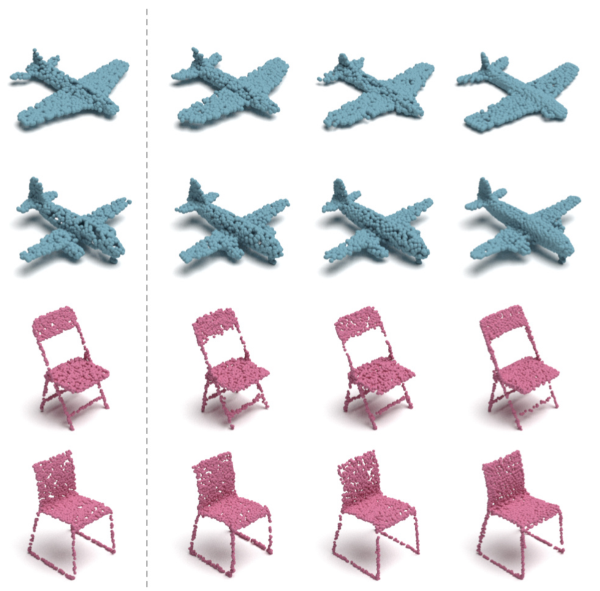

Figure 5. Examples of reconstructed point clouds from different auto-encoders.

(EMD) to evaluate the reconstruction quality of the point clouds [1]. To evaluate the generation quality, we employ the minimum matching distance (MMD), the coverage score (COV), 1-NN classifier accuracy (1-NNA) and the Jenson-Shannon divergence (JSD) [25]. The MMD score measures the fidelity of the generated samples and the COV score detects mode-collapse. The 1-NNA score is computed by testing the generated samples and the reference samples by a 1-NN classifier. If the performance of the classifier is close to random guess, *i.e.*, the accuracy is close to 50%, the quality of generated samples can be considered better. The JSD score measures the similarity between the point distributions of the generated set and the reference set.

Table 3. Comparison of representation learning in linear SVM classification accuracy.

Model	ModelNet10	ModelNet40
AtlasNet [10]	91.9	86.6
PC-GAN (CD) [1]	**95.4**	84.5
PC-GAN (EMD) [1]	**95.4**	84.0
PointFlow [25]	93.7	86.8
ShapeGF [2]	90.2	84.6
Ours	94.2	**87.6**

5.2. Point Cloud Generation

We quantitatively compare our method with the following state-of-the-art generative models: PC-GAN [1], GCN-GAN [22], TreeGAN [19], PointFlow [25] and ShapeGF [2] using point clouds from two categories in ShapeNet: *airplane* and *chair*. Following ShapeGF [2], when evaluating each of the model, we normalize both generated point clouds and reference point clouds into a bounding box of $[-1, 1]^3$, so that the metrics focus on the shape of point clouds but not the scale. We evaluate the point clouds generated by the models using the metrics in Section 5.1 and summarize the results in Table 1. We also visualize some generated point cloud samples from our method in Figure 4.

5.3. Point Cloud Auto-Encoding

We evaluate the reconstruction quality of the proposed auto-encoder, with comparisons against state-of-the-art point cloud auto-encoders: AtlasNet [10], PointFlow [25] and ShapeGF [2]. Four datasets are used in the evaluation, which include three categories in ShapeNet: *airplane*, *car*, *chair* and the whole ShapeNet. We also report the lower bound "oracle" of the reconstruction errors. This bound is obtained by computing the distance between

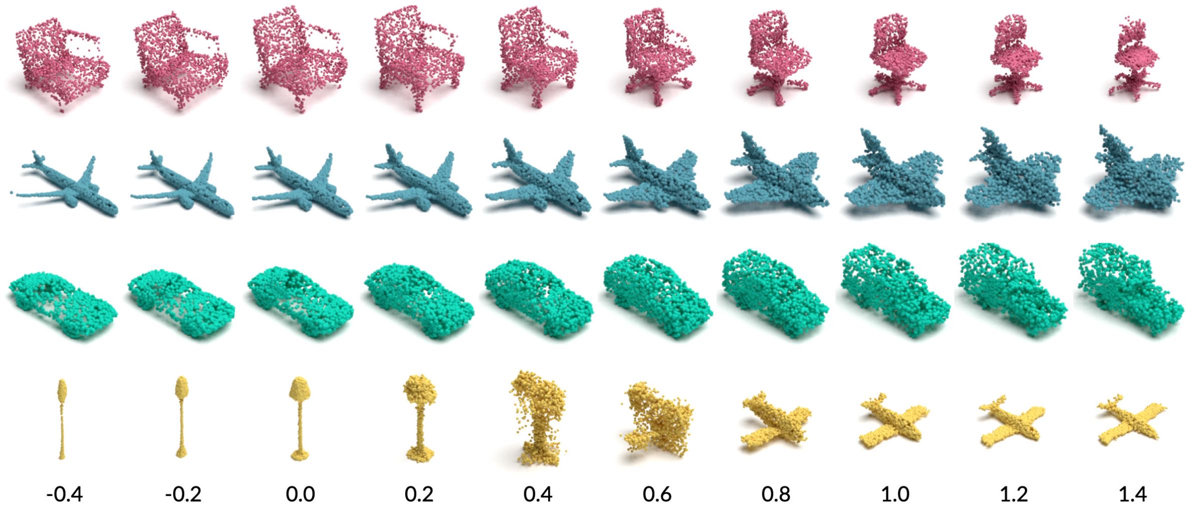

Figure 6. Latent space interpolation and extrapolation.

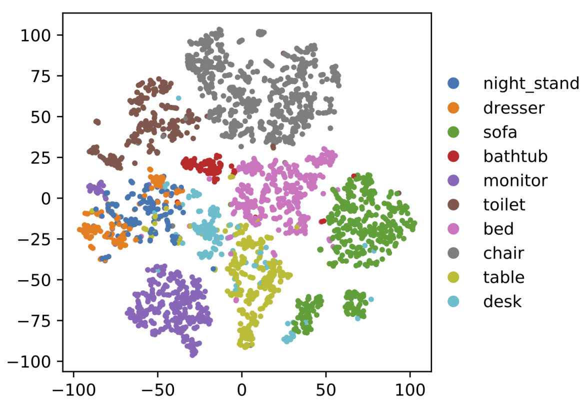

Figure 7. The t-SNE clustering visualization of latent codes obtained from the encoder.

two different point clouds with the same number of points and the identical shape. As shown in Table 2, our method outperforms other methods when measured by EMD, and pushes closer towards the lower bounded "oracle" performance. The CD score of our method is comparable to others. Notably, when trained and tested on the whole ShapeNet dataset, our model outperforms others in both CD and EMD, which suggests that our model has higher capacity to encode different shapes. Also, the visualization of reconstructed point clouds in Figure 5 validates the effectiveness of our model.

5.4. Unsupervised Representation Learning

Further, we evaluate the representation learned by our auto-encoder. Firstly, we train an auto-encoder with the whole ShapeNet dataset. During the training, we augment point clouds by applying random rotations along the gravity axis, which follows previous works. Then, we learn the feature representations of point clouds in ModelNet-10 and ModelNet-40 using the trained encoder, and train a linear SVM using the codes of point clouds in the training split and their categories. Finally, we test the SVM using the testing split and report the accuracy in Table 3. We run the official code of AtlasNet and ShapeGF to obtain the results, since the results are not provided in their papers. For PC-GAN and PointFlow, we use the results reported in the papers. The performance of our encoder is comparable to related state-of-the-art generative models.

In addition, we project the latent codes of ModelNet-10 point clouds produced by the encoder into the 2D plane using t-SNE [15], and present it in Figure 7. It can be observed that there are significant margins between most categories, which indicates that our model manages to learn informative representations. Further, we visualize the interpolation and extrapolation between latent codes in Figure 6.

6. Conclusions

We propose a novel probabilistic generative model for point clouds, taking inspiration from the diffusion process in non-equilibrium thermodynamics. We model the reverse diffusion process for point cloud generation as a Markov chain conditioned on certain shape latent, and derive a tractable training objective from the variational bound of the likelihood of point clouds. Experimental results demonstrate that the proposed model achieves the state-of-the-art performance in point cloud generation and auto-encoding.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018.

[2] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. *arXiv preprint arXiv:2008.06520*, 2020.

[3] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[4] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

[5] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

[6] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[7] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution tree networks for 3d point cloud processing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 103–118, 2018.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[9] 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.

[10] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018.

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

[12] Christopher Jarzynski. Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach. *Physical Review E*, 56(5):5018, 1997.

[13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[14] Roman Klokov, Edmond Boyer, and Jakob Verbeek. Discrete point flow networks for efficient point cloud generation. *arXiv preprint arXiv:2007.10170*, 2020.

[15] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[16] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[18] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

[19] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3859–3868, 2019.

[20] Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv preprint arXiv:1503.03585*, 2015.

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

[22] Diego Valsesia, Giulia Fracastoro, and Enrico Magli. Learning localized generative models for 3d point clouds via graph convolution. In *International conference on learning representations*, 2018.

[23] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

[24] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[25] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4541–4550, 2019.

[26] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018.