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`.

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.

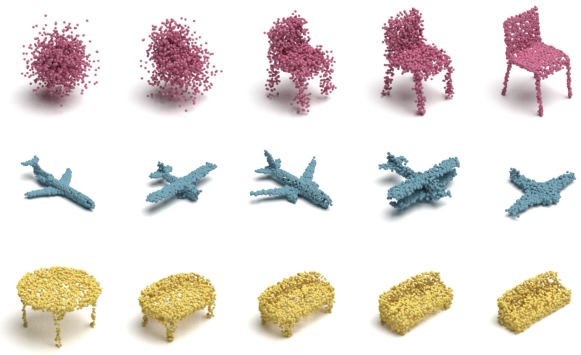

Figure 1. **Top**: The diffusion process that converts noise to some shape (left to right). **Middle**: Generated point clouds from the proposed model. **Bottom**: Latent space interpolation between the two point clouds at both ends.

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):

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

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

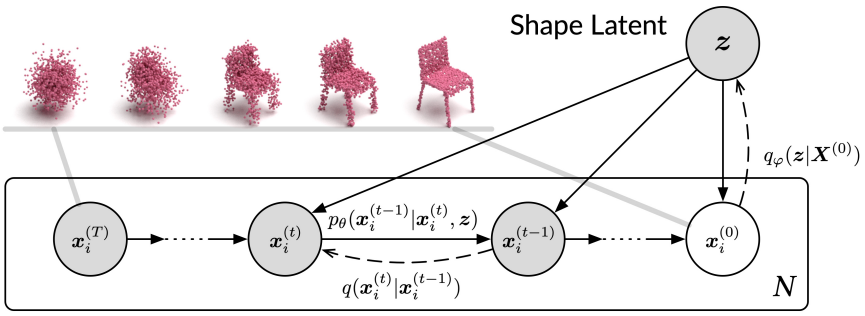

Figure 2. The directed graphical model of the diffusion process for point clouds. N is the number of points in the point cloud $\boldsymbol{X}^{(0)}$.

objective:

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

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

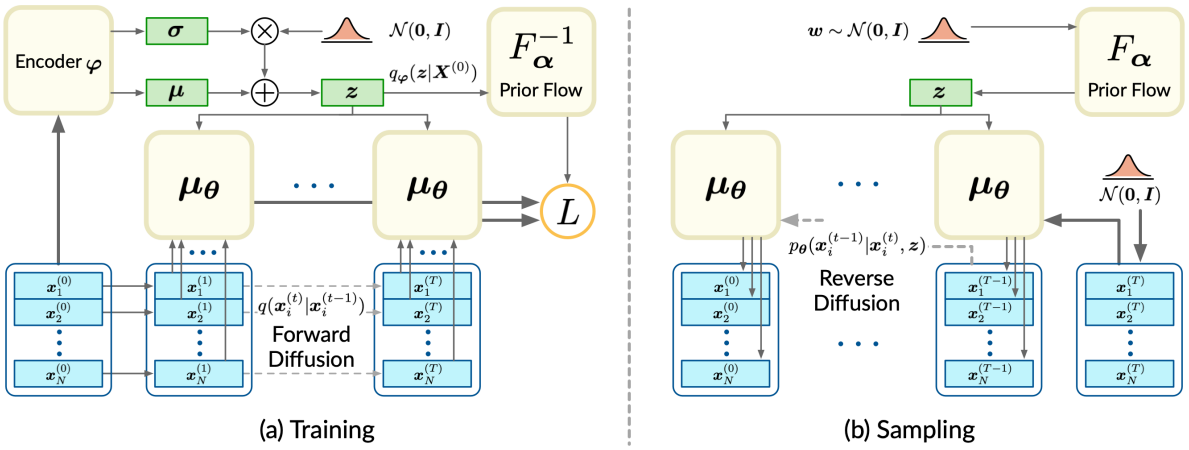

Figure 3. The illustration of the proposed model. (a) illustrates how the objective is computed during the training process. (b) illustrates the generation process.

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_{α} 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:

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

Here, $p(\boldsymbol{z})$ is the prior distribution in the model, F_{α} 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:

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

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_{α} 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