Diffusion Probabilistic Models for 3D Point Cloud Generation Supplementary Material

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

1. Detailed Derivations

We present the detailed derivations of the training objective in Eq. (9).

$$\begin{aligned}
\mathbb{E}_{q_{\text{data}}}\Big[\log p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(0)})\Big] &= \int q_{\text{data}}(\boldsymbol{X}^{(0)}) \left[\log \int p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(0:T)}, \boldsymbol{z}) \,\mathrm{d}\, \boldsymbol{X}^{(1:T)} \,\mathrm{d}\, \boldsymbol{z}\right] \mathrm{d}\, \boldsymbol{X}^{(0)} \\
&= \int q_{\text{data}}(\boldsymbol{X}^{(0)}) \left[\log \int q(\boldsymbol{X}^{(1:T)}, \boldsymbol{z}|\boldsymbol{X}^{(0)}) \frac{p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(0:T)}, \boldsymbol{z})}{q(\boldsymbol{X}^{(1:T)}, \boldsymbol{z}|\boldsymbol{X}^{(0)})} \,\mathrm{d}\, \boldsymbol{X}^{(1:T)} \,\mathrm{d}\, \boldsymbol{z}\right] \mathrm{d}\, \boldsymbol{X}^{(0)} \\
&\geq \int q_{\text{data}}(\boldsymbol{X}^{(0)}) q(\boldsymbol{X}^{(1:T)}, \boldsymbol{z}|\boldsymbol{X}^{(0)}) \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(0:T)}, \boldsymbol{z})}{q(\boldsymbol{X}^{(1:T)}, \boldsymbol{z}|\boldsymbol{X}^{(0)})} \,\mathrm{d}\, \boldsymbol{X}^{(1:T)} \,\mathrm{d}\, \boldsymbol{z} \,\mathrm{d}\, \boldsymbol{X}^{(0)} \quad \text{(ELBO)}
\end{aligned}$$

Note that $\boldsymbol{X}^{(1:T)}$ and $\boldsymbol{z}$ are conditionally independent on $\boldsymbol{X}^{(0)}$,

$$\begin{aligned}
&= \int q_{\text{data}}(\boldsymbol{X}^{(0)}) q(\boldsymbol{X}^{(1:T)}|\boldsymbol{X}^{(0)}) q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)}) \Big[\log p(\boldsymbol{X}^{(T)}) + \log p(\boldsymbol{z}) \\
&\quad + \sum_{t=1}^{T} \log p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{z}) - \log q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)}) \\
&\quad - \sum_{t}^{T} \log q(\boldsymbol{X}^{(t)}|\boldsymbol{X}^{(t-1)})\Big] \,\mathrm{d}\, \boldsymbol{X}^{(0:T)} \,\mathrm{d}\, \boldsymbol{z} \\
&= \int q_{\text{data}}(\boldsymbol{X}^{(0)}) q(\boldsymbol{X}^{(1:T)}|\boldsymbol{X}^{(0)}) q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)}) \Big[\log p(\boldsymbol{X}^{(T)}) + \log p(\boldsymbol{z}) \\
&\quad + \sum_{t=1}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{z})}{q(\boldsymbol{X}^{(t)}|\boldsymbol{X}^{(t-1)})} - \log q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)})\Big] \,\mathrm{d}\, \boldsymbol{X}^{(0:T)} \,\mathrm{d}\, \boldsymbol{z}
\end{aligned}$$

Since $q(\boldsymbol{X}^{(t)}|\boldsymbol{X}^{(t-1)})$ is intractable, we rewrite it using Bayes' rule,

$$
\begin{aligned}
&= \int q_{\text{data}}(\boldsymbol{X}^{(0)}) q(\boldsymbol{X}^{(1:T)}|\boldsymbol{X}^{(0)}) q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)}) \Big[\log p(\boldsymbol{X}^{(T)}) + \log p(\boldsymbol{z}) \\
&\quad + \sum_{t=2}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{z})}{q(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{X}^{(0)})} \cdot \frac{q(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(0)})}{q(\boldsymbol{X}^{(t)}|\boldsymbol{X}^{(0)})} \\
&\quad + \log \frac{p(\boldsymbol{X}^{(0)}|\boldsymbol{X}^{(1)}, \boldsymbol{z})}{q(\boldsymbol{X}^{(1)}|\boldsymbol{X}^{(0)})} - \log q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)}) \Big] \, \mathrm{d}\, \boldsymbol{X}^{(0:T)} \, \mathrm{d}\, \boldsymbol{z} \\
&= \int q_{\text{data}}(\boldsymbol{X}^{(0)}) q(\boldsymbol{X}^{(1:T)}|\boldsymbol{X}^{(0)}) q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)}) \Big[\log \frac{p(\boldsymbol{X}^{(T)})}{q(\boldsymbol{X}^{(T)}|\boldsymbol{X}^{(0)})} \\
&\quad + \sum_{t=2}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{z})}{q(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{X}^{(0)})} + \log p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(0)}|\boldsymbol{X}^{(1)}, \boldsymbol{z}) \\
&\quad + \log p(\boldsymbol{z}) - \log q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)}) \Big] \, \mathrm{d}\, \boldsymbol{X}^{(0:T)} \, \mathrm{d}\, \boldsymbol{z} \\
&= \int q_{\boldsymbol{\varphi}}(\boldsymbol{X}^{(0:T)}, \boldsymbol{z}) \Big[\log \frac{p(\boldsymbol{X}^{(T)})}{q(\boldsymbol{X}^{(T)}|\boldsymbol{X}^{(0)})} + \sum_{t=2}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{z})}{q(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{X}^{(0)})} \\
&\quad + \log p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(0)}|\boldsymbol{X}^{(1)}, \boldsymbol{z}) + \log \frac{p(\boldsymbol{z})}{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)})} \Big] \, \mathrm{d}\, \boldsymbol{X}^{(0:T)} \, \mathrm{d}\, \boldsymbol{z}
\end{aligned}
$$

On the right hand side, all the terms except $\log p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(0)}|\boldsymbol{X}^{(1)}, \boldsymbol{z})$ can be rewritten into the form of the KL divergence. We show how to do it on one of the terms. For other terms, it is similar.

$$
\begin{aligned}
&\int q_{\boldsymbol{\varphi}}(\boldsymbol{X}^{(0:T)}, \boldsymbol{z}) \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{z})}{q(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{X}^{(0)})} \, \mathrm{d}\, \boldsymbol{X}^{(0:T)} \, \mathrm{d}\, \boldsymbol{z} \\
&\quad = \int q_{\boldsymbol{\varphi}}(\boldsymbol{X}^{(0)}, \boldsymbol{X}^{(t-1)}, \boldsymbol{X}^{(t)}, \boldsymbol{z}) \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{z})}{q(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{X}^{(0)})} \, \mathrm{d}\, \boldsymbol{X}^{(0,t-1,t)} \, \mathrm{d}\, \boldsymbol{z} \\
&\quad = \int q(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(0)}, \boldsymbol{X}^{(t)}) q_{\boldsymbol{\varphi}}(\boldsymbol{X}^{(0)}, \boldsymbol{X}^{(t)}, \boldsymbol{z}) \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{z})}{q(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{X}^{(0)})} \, \mathrm{d}\, \boldsymbol{X}^{(0,t-1,t)} \, \mathrm{d}\, \boldsymbol{z} \\
&\quad = - \int q_{\boldsymbol{\varphi}}(\boldsymbol{X}^{(0)}, \boldsymbol{X}^{(t)}, \boldsymbol{z}) D_{\mathrm{KL}} \left(q(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{X}^{(0)}) \middle\| p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{z}) \right) \mathrm{d}\, \boldsymbol{X}^{(0,t)} \, \mathrm{d}\, \boldsymbol{z} \\
&\quad = -\mathbb{E}_{\boldsymbol{X}^{(0)}, \boldsymbol{X}^{(t)}, \boldsymbol{z} \sim q_{\boldsymbol{\varphi}}} \left[D_{\mathrm{KL}} \left(q(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{X}^{(0)}) \middle\| p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{z}) \right) \right].
\end{aligned}
$$

Next, notice that $\log \frac{p(\boldsymbol{X}^{(T)})}{q(\boldsymbol{X}^{(T)}|\boldsymbol{X}^{(0)})}$ has no trainable parameters, so we can ignore it in the training objective. Finally, by negating the variational bound and decomposing the distributions, we obtain the training objective in Eq. (9) as follows:

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

2. Simplified Training Algorithm

The original training and sampling algorithm is formulated according to the generator's training objective in Eq. (15),.

Algorithm 1 Training

repeat
  Sample $\boldsymbol{X}^{(0)} \sim q_{\text{data}}(\boldsymbol{X}^{(0)})$
  Sample $\boldsymbol{z} \sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)})$
  for $t = 1 \dots T$ **do**
    Sample $\boldsymbol{X}^{(t)} \sim q(\boldsymbol{X}^{(t)}|\boldsymbol{X}^{(t-1)})$
  end for
  Compute $\nabla L_G(\boldsymbol{\theta}, \boldsymbol{\varphi}, \boldsymbol{\alpha})$ using samples $\boldsymbol{X}^{(0:T)}$ and $\boldsymbol{z}$; Then perform gradient descent.
until converged

Algorithm 2 Sampling

Sample $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
$\boldsymbol{z} \leftarrow F_{\boldsymbol{\alpha}}(\boldsymbol{w})$
$\boldsymbol{X}^{(T)} \leftarrow \{\boldsymbol{x}_i^{(T)}\} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
for $t = T \dots 1$ **do**
  Sample $\boldsymbol{X}^{(t-1)} \sim p_{\boldsymbol{\theta}}(\boldsymbol{X}^{(t-1)}|\boldsymbol{X}^{(t)}, \boldsymbol{z})$
end for
return $\boldsymbol{X}^{(0)}$

[5] proposed a simplified training algorithm. We also adapt the simplified algorithm for our model. Before formulating the simplified algorithm, we should further analyse $L_i^{(t-1)}$. Since both $q(\boldsymbol{x}_i^{(t-1)}|\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_i^{(0)})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(t-1)}|\boldsymbol{x}_i^{(t)}, \boldsymbol{z})$ are Gaussians (Eq. (10), Eq. (4)), the term $L_i^{(t-1)}$ can be expanded as:

$$L_i^{(t-1)} = \mathbb{E}_{\boldsymbol{x}_i^{(0)}\boldsymbol{x}_i^{(t)}\boldsymbol{z}}\left[\frac{1}{2\beta_t}\left\|\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x}_i^{(0)} + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_i^{(t)} - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(t)}, t, \boldsymbol{z})\right\|^2\right] + C.$$

Evaluating $L_i^{(t-1)}$ requires sampling $\boldsymbol{x}_i^{(t)}$ from $q(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(0)})$. In principle, it can be done by sampling iteratively through the Markov chain. However, [5] showed $q(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(0)})$ is a Gaussian, thus allowing us to sample $\boldsymbol{x}^{(t)}$ efficiently without iterative sampling:

$$q(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(0)}) = \mathcal{N}(\boldsymbol{x}^{(t)}|\sqrt{\bar{\alpha}_t}\boldsymbol{x}^{(0)}, (1-\bar{\alpha}_t)\boldsymbol{I}). \tag{14}$$

Using the Gaussian above, we can parameterize $\boldsymbol{x}_i^{(t)}$ as $\boldsymbol{x}_i^{(t)}(\boldsymbol{x}_i^{(0)}, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_i^{(0)} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$:

$$L_i^{(t-1)} = \mathbb{E}_{\boldsymbol{x}_i^{(0)}, \boldsymbol{\epsilon}, \boldsymbol{z}}\left[\frac{1}{2\beta_t}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_i^{(t)} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right) - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(t)}, t, \boldsymbol{z})\right\|^2\right] + C. \tag{14}$$

The above equation reveals that $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(t)}, t)$ must predict $\frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_i^{(t)} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right)$ given $\boldsymbol{x}_i^{(t)}$. Thus, $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(t)}, t)$ can be parameterized as:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(t)}, t) = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_i^{(t)} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(t)}, t, \boldsymbol{z})\right), \tag{14}$$

where $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(t)}, t, \boldsymbol{z})$ is a function approximator (*i.e.*, neural network) intended to predict $\boldsymbol{\epsilon}$ from $\boldsymbol{x}_i^{(t)}$. Finally, $L_i^{(t-1)}$ can be simplified as

$$L_i^{(t-1)} = \mathbb{E}_{\boldsymbol{x}_i^{(0)}, \boldsymbol{\epsilon}, \boldsymbol{z}}\left[\frac{\beta_t^2}{2\beta_t\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_i^{(0)} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t, \boldsymbol{z})\right\|^2\right] + C. \tag{14}$$

To minimize $L_i^{(t-1)}$, we can only minimize $\mathbb{E}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\|^2\right]$ because the coefficient $\frac{\beta_t^2}{2\beta_t\alpha_t(1-\bar{\alpha}_t)}$ is constant.

The simplified algorithm proposed in [5] suggests choosing a random term from $\{\sum_{i=1}^N L_i^{(t-1)}\}_{t=1}^T$ to optimize at each training step. In addition to that, the prior loss term $L_{\boldsymbol{z}}$ in our objective function should also be considered. Since only one term in $\{\sum_{i=1}^N L_i^{(t-1)}\}_{t=1}^T$ is optimized at each step, we re-weight $L_{\boldsymbol{z}}$ with $\frac{1}{T}$. The adapted simplified training algorithm is as follows:

Algorithm 3 Training (Simplified)

repeat
  Sample $\boldsymbol{X}^{(0)} \sim q_{\text{data}}(\boldsymbol{X}^{(0)})$
  Sample $\boldsymbol{z} \sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)})$
  Sample $t \sim \text{Uniform}(\{1, \dots, T\})$
  Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
  Compute $\nabla\left[\sum_{i=1}^N \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_i^{(0)} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t, \boldsymbol{z})\|^2 + \frac{1}{T}D_{\text{KL}}\left[q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)})\|p(\boldsymbol{z})\right]\right.$; Then perform gradient descent.
until converged

Note that the KL divergence in $L_{\boldsymbol{z}}$ is evaluated stochastically by:

$$D_{\mathrm{KL}}\big[q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)})\|p(\boldsymbol{z})\big] = -\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)})}\big[p(\boldsymbol{z})\big] - H\big[q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X}^{(0)})\big]. \tag{14}$$

3. Implementation Details

PointNet Encoder The architecture of our encoder follows that of PC-GAN, PointFlow and ShapeGF [8, 1, 9, 2]. Specifically, we feed point clouds into a 3-128-256-512 MLP with the ReLU nonlinearity followed by a max-pooling to obtain a global 512-dimension feature. Then, the feature is fed into a 512-256-128-256 MLP with the ReLU nonlinearity and we obtain the latent code of 256-dimension.

Prior Flow We stack 14 affine coupling layers to construct the prior flow. The dimension of hidden states is 256, identical to the dimension of latent codes. Following each of the layers, we apply moving batch normalization [6, 4]. Both the scaling and translation networks $F(\cdot)$ and $G(\cdot)$ are 128-256-256-128 MLPs with the ReLU nonlinearity.

Diffusion Process The number of steps T in the diffusion process is 200. We set the variance schedules to be $\beta_1 = 0.0001$ and $\beta_T = 0.05$, and β_t's $(1 < t < T)$ are linearly interpolated.

Intuitively speaking, the *reverse* diffusion process is analogous to MCMC (Langevin dynamics) sampling procedures where β_t is the step size of the $(T - t + 1)$-th step (Eq. 4). Since we normalize point clouds to unit variance and the coordinates of points roughly range from -2 to 2, we set the initial step size β_T to 0.05, slightly larger than $\frac{2-(-2)}{T} = \frac{4}{200}$, in order to ensure that the points can walk through possible regions of different shapes in early steps. To make the points concentrate in desired regions, $\beta_T, \ldots, \beta_1$ should be decaying and the last "step size" β_1 should be sufficiently small, so we set β_1 to 0.0001.

Reverse Diffusion Kernel The reverse diffusion kernel in Eq. (6) is paramtereized by $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(t)}, t, \boldsymbol{z})$, as derived in Appendix 2. We implement it using a variant of MLP, which consists of a series of `concatsquash` layers [4] defined as:

$$\boldsymbol{h}^{\ell+1} = \texttt{CS}(\boldsymbol{h}^{\ell}, t, \boldsymbol{z}) = (\boldsymbol{W}_1\boldsymbol{h}^{\ell} + \boldsymbol{b}_1) \odot \sigma(\boldsymbol{W}_2\boldsymbol{c} + \boldsymbol{b}_2) + \boldsymbol{W}_3\boldsymbol{c}, \tag{14}$$

where $\boldsymbol{h}^{\ell}$ is the input to the layer and $\boldsymbol{h}^{\ell+1}$ is the output. The input to the first layer is the 3D positions of points $\boldsymbol{x}_i^{(t)}$. $\boldsymbol{c} = [t, \sin(t), \cos(t), \boldsymbol{z}]$ is the context vector, and σ denotes the sigmoid function. $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{W}_3$, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are all trainable parameters. The dimension of the `concatsquash`-MLP used in our model is 3-128-256-512-256-128-3, and we use the LeakyReLU nonlinearity between the layers.

Dataset Split We split the ShapeNet [3] into training, testing and validation sets by the ratio 80%, 15%, 5% respectively. We don't use the dataset split in recent works because as reported in [7], their split is somewhat unbalanced (*e.g.*, the training and validation sets of the airplane category mostly contain passenger airplanes while the test sets contains mostly fighter jets and spaceships.), and those works actually use the *validation* set rather than *testing* set to test models.

Open Source The code of this project is available at https://github.com/luost26/diffusion-point-cloud.

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. 4

[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. 4

[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

[4] 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. 4

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020. 3

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 4

[7] Roman Klokov, Edmond Boyer, and Jakob Verbeek. Discrete point flow networks for efficient point cloud generation. *arXiv preprint arXiv:2007.10170*, 2020. 4

[8] 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. 4

[9] 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. 4