1. Detailed Derivations

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

\[
\mathbb{E}_{q_{\text{data}}} \left[ \log p_{\theta}(X^{(0)}) \right] = \int q_{\text{data}}(X^{(0)}) \left[ \log \int p_{\theta}(X^{(0):T}, z) \, dX^{(1:T)} \, dz \right] \, dX^{(0)} \\
\geq \log \int q_{\text{data}}(X^{(0)}) p_{\theta}(X^{(0):T}, z) \, dX^{(1:T)} \, dz \, dX^{(0)} \quad \text{(Jensen’s inequality)} \\
\geq \int q_{\text{data}}(X^{(0)}) q(X^{(1:T)}, z|X^{(0)}) \log \frac{p_{\theta}(X^{(0):T}, z)}{q(X^{(1:T)}, z|X^{(0)})} \, dX^{(1:T)} \, dz \, dX^{(0)} \quad \text{(ELBO)}
\]

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

\[
= \int q_{\text{data}}(X^{(0)}) q(X^{(1:T)}|X^{(0)}) q_{\phi}(z|X^{(0)}) \left[ \log p(X^{(T)}) + \log p(z) \right] \\
+ \sum_{t=1}^{T} \log p_{\theta}(X^{(t-1)}|X^{(t)}, z) - \log q_{\phi}(z|X^{(0)}) \\
- \sum_{t} \log q(X^{(t)}|X^{(t-1)}) \, dX^{(0):T} \, dz \\
= \int q_{\text{data}}(X^{(0)}) q(X^{(1:T)}|X^{(0)}) q_{\phi}(z|X^{(0)}) \left[ \log p(X^{(T)}) + \log p(z) \right] \\
+ \sum_{t=1}^{T} \log \frac{p_{\theta}(X^{(t-1)}|X^{(t)}, z)}{q(X^{(t)}|X^{(t-1)})} - \log q_{\phi}(z|X^{(0)}) \, dX^{(0):T} \, dz
\]
Since $q(X^{(t)}|X^{(t-1)})$ is intractable, we rewrite it using Bayes’ rule,

$$
= \int q_{\text{data}}(X^{(0)}) q(X^{(1:T)}|X^{(0)}) q_\phi(z|X^{(0)}) \left[ \log p(X^{(T)}) + \log p(z) \\
+ \sum_{t=2}^{T} \log \frac{p_\theta(X^{(t-1)}|X^{(t)}, z)}{q(X^{(t-1)}|X^{(t)}, X^{(0)})} q(X^{(t-1)}|X^{(0)}) \\
+ \log \frac{p(X^{(0)}|X^{(1)}) z}{q(X^{(1)}|X^{(0)})} - \log q_\phi(z|X^{(0)}) \right] dX^{(0:T)} d z
$$

$$
= \int q_{\text{data}}(X^{(0)}) q(X^{(1:T)}|X^{(0)}) q_\phi(z|X^{(0)}) \left[ \log \frac{p(X^{(T)})}{q(X^{(T)}|X^{(0)})} \\
+ \sum_{t=2}^{T} \log \frac{p_\theta(X^{(t-1)}|X^{(t)}, z)}{q(X^{(t-1)}|X^{(t)}, X^{(0)})} + \log p_\theta(X^{(0)}) | X^{(1)} z \\
+ \log p(z) - \log q_\phi(z|X^{(0)}) \right] dX^{(0:T)} d z
$$

$$
= \int q_\phi(X^{(0:T)}, z) \left[ \log \frac{p(X^{(T)})}{q(X^{(T)}|X^{(0)})} + \sum_{t=2}^{T} \log \frac{p_\theta(X^{(t-1)}|X^{(t)}, z)}{q(X^{(t-1)}|X^{(t)}, X^{(0)})} q_\phi(X^{(t)}, z) \\
+ \log p_\theta(X^{(0)} | X^{(1)} z) + \log \frac{p(z)}{q_\phi(z|X^{(0)})} \right] dX^{(0:T)} d z
$$

On the right hand side, all the terms except $\log p_\theta(X^{(0)} | X^{(1)} z)$ 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.

$$
\int q_\phi(X^{(0:T)}, z) \log \frac{p_\theta(X^{t-1}|X^{(t)}, z)}{q(X^{(t-1)}|X^{(t)}, X^{(0)})} dX^{(0:T)} d z
$$

$$
= \int q_\phi(X^{(t-1)}, X^{(t)}, z) \log \frac{p_\theta(X^{(t-1)}|X^{(t)}, z)}{q(X^{(t-1)}|X^{(t)}, X^{(0)})} dX^{(0,t-1,t)} d z
$$

$$
= \int q(X^{(t-1)}|X^{(0)}, X^{(t)}, z) q_\phi(X^{(0)}, X^{(t)}, z) \log \frac{p_\theta(X^{(t-1)}|X^{(t)}, z)}{q(X^{(t-1)}|X^{(t)}, X^{(0)})} dX^{(0,t-1,t)} d z
$$

$$
= - \int q_\phi(X^{(0)}, X^{(t)}, z) D_{KL} \left( q(X^{(t-1)}|X^{(t)}, X^{(0)}) \| p_\theta(X^{(t-1)}|X^{(t)}, z) \right) dX^{(0,t)} d z
$$

$$
= - \mathbb{E}_{X^{(t)}, X^{(0)}, z \sim q_\phi} \left[ D_{KL} \left( q(X^{(t-1)}|X^{(t)}, X^{(0)}) \| p_\theta(X^{(t-1)}|X^{(t)}, z) \right) \right]
$$

Next, notice that $\log \frac{p(X^{(T)})}{q(X^{(T)}|X^{(0)})}$ 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:

$$
L(\theta, \phi) = \mathbb{E}_{q} \left[ \sum_{t=2}^{T} \sum_{i=1}^{N} D_{KL} \left( q(x^{(t-1)}_i | x^{(t)}_i, x^{(0)}_i) \| p_\theta(x^{(t-1)}_i | x^{(t)}_i, z) \right) \\
- \sum_{i=1}^{N} \log p_\theta(x^{(0)}_i | x^{(1)}_i, z) + D_{KL} \left( q_\phi(z|X^{(0)}) \| p(z) \right) \right].
$$

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

1: repeat
2: Sample $X^{(0)} \sim q_{data}(X^{(0)})$
3: Sample $z \sim q_{x}(z|X^{(0)})$
4: for $t = 1 \ldots T$ do
5: Sample $X^{(t)} \sim q(X^{(t)}|X^{(t-1)})$
6: end for
7: Compute $\nabla L_C(\theta, \varphi, \alpha)$ using samples $X^{(0:T)}$ and $z$; Then perform gradient descent.
8: until converged

Algorithm 2 Sampling

1: Sample $w \sim \mathcal{N}(0, I)$
2: $z \leftarrow F_\alpha(w)$
3: $X^{(T)} \leftarrow \{x_i^{(T)}\} \sim \mathcal{N}(0, I)$
4: for $t = T \ldots 1$ do
5: Sample $X^{(t-1)} \sim p_\theta(X^{(t-1)}|X^{(t)}, z)$
6: end for
7: return $X^{(0)}$

[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 $q(x^{(t-1)}|x_i^{(t-1)}, x_i^{(0)})$ and $p_\theta(x^{(t-1)}|x_i^{(t)}, z)$ are Gaussians (Eq. (10), Eq. (4)), the term $L_i^{(t-1)}$ can be expanded as:

$$L_i^{(t-1)} = \mathbb{E}_{x_i^{(0)}, x_i^{(t)}} \left[ \frac{1}{2\beta_t} \left\| \frac{1}{\sqrt{\alpha_t}} \left( x_i^{(t)} - \frac{\beta_t}{\sqrt{1-\alpha_t}} \epsilon \right) - \mu_\theta(x_i^{(t)}, t, z) \right\|^2 \right] + C.$$ \hfill (14)

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

$$q(x^{(t)}|x_i^{(0)}) = \mathcal{N}(x^{(t)}|\sqrt{\alpha_t}x_i^{(0)}, (1-\alpha_t)I).$$ \hfill (14)

Using the Gaussian above, we can parameterize $x_i^{(t)}$ as $x_i^{(t)}(x_i^{(0)}, \epsilon) = \sqrt{\alpha_t}x_i^{(0)} + \sqrt{1-\alpha_t}\epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$:

$$L_i^{(t-1)} = \mathbb{E}_{x_i^{(0)}, \epsilon, z} \left[ \frac{1}{2\beta_t} \left\| \frac{1}{\sqrt{\alpha_t}} \left( x_i^{(t)} - \frac{\beta_t}{\sqrt{1-\alpha_t}} \epsilon \right) - \mu_\theta(x_i^{(t)}, t, z) \right\|^2 \right] + C. \hfill (14)$$

The above equation reveals that $\mu_\theta(x_i^{(t)}, t)$ must predict $\frac{1}{\sqrt{\alpha_t}} \left( x_i^{(t)} - \frac{\beta_t}{\sqrt{1-\alpha_t}} \epsilon \right)$ given $x_i^{(t)}$. Thus, $\mu_\theta(x_i^{(t)}, t)$ can be parameterized as:

$$\mu_\theta(x_i^{(t)}, t) = \frac{1}{\sqrt{\alpha_t}} \left( x_i^{(t)} - \frac{\beta_t}{\sqrt{1-\alpha_t}} \epsilon \theta(x_i^{(t)}, t, z) \right),$$ \hfill (14)

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

$$L_i^{(t-1)} = \mathbb{E}_{x_i^{(0)}, \epsilon, z} \left[ \frac{\beta_t^2}{2\beta_t\alpha_t(1-\alpha_t)} \left\| \epsilon - \epsilon \theta(\sqrt{\alpha_t}x_i^{(0)} + \sqrt{1-\alpha_t}\epsilon, t, z) \right\|^2 \right] + C. \hfill (14)$$

To minimize $L_i^{(t-1)}$, we can only minimize $\mathbb{E} [\| \epsilon - \epsilon \theta \|^2]$ because the coefficient $\frac{\beta_t^2}{2\beta_t\alpha_t(1-\alpha_t)}$ 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 $L_z$ 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 $L_z$ with $\frac{1}{T}$. The adapted simplified training algorithm is as follows:

Algorithm 3 Training (Simplified)

1: repeat
2: Sample $X^{(0)} \sim q_{data}(X^{(0)})$
3: Sample $z \sim q_{x}(z|X^{(0)})$
4: Sample $t \sim \text{Uniform}\{1, \ldots, T\}$
5: Sample $\epsilon \sim \mathcal{N}(0, I)$
6: Compute $\nabla \left[ \sum_{i=1}^N \| \epsilon - \epsilon \theta(\sqrt{\alpha_t}x_i^{(0)} + \sqrt{1-\alpha_t}\epsilon, t, z)\|^2 + \frac{1}{T}D_{KL}[q_{\phi}(z|X^{(0)}) || p(z)] \right]$; Then perform gradient descent.
7: until converged
Note that the KL divergence in $L_2$ is evaluated stochastically by:

$$D_{KL}[q_{\phi}(z|X^{(0)})||p(z)] = -\mathbb{E}_{z \sim q_{\phi}(z|X^{(0)})}[p(z)] - H[q_{\phi}(z|X^{(0)})].$$

(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 128-256-128 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 $\beta_i$’s ($1 < t < T$) are linearly interpolated.

Intuitively speaking, the reverse diffusion process is analogous to MCMC (Langevin dynamics) sampling procedures where $\beta_t$ 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 $\beta_T$ to 0.05, slightly larger than $\frac{2(-2)}{200} = \frac{1}{100}$, 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, $\beta_T, \ldots, \beta_1$ be should be decaying and the last “step size” $\beta_1$ should be sufficiently small, so we set $\beta_1$ to 0.0001.

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

$$h^{t+1} = \text{cs}(h^t, t, z) = (W_1h^t + b_1) \circ \sigma(W_2c + b_2) + W_3c,$$

where $h^t$ is the input to the layer and $h^{t+1}$ is the output. The input to the first layer is the 3D positions of points $x^{(t)}_i$. $c = [t, \sin(t), \cos(t), z]$ is the context vector, and $\sigma$ denotes the sigmoid function. $W_1, W_2, W_3, b_1$ and $b_2$ are all trainable parameters. The dimension of the $\text{concatsquash}$-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


