

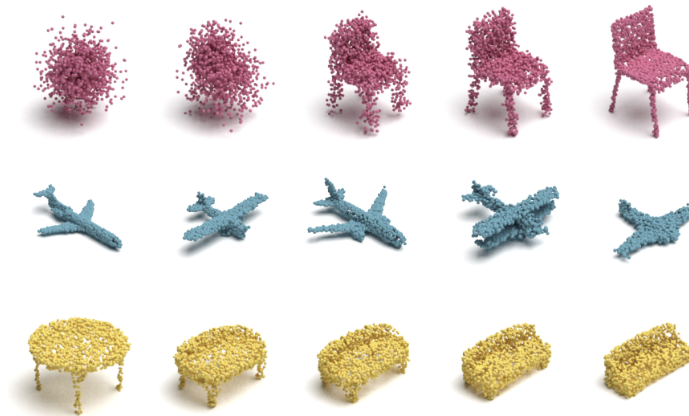
本周工作：

1. 打标签，完成；
2. 研读Diffusion Probabilistic Models；
3. 调研了微软为unity提供的Hololens库和Ar Foundation；

Diffusion Probabilistic Models for 3D Point Cloud Generation

1. Introduction

该文受非平衡热力学启发，基于DiffusionModel提出了点云生成方法。其中，他们认为3D点云中的点可视为非平衡热力学系统中的粒子，在扩散作用下粒子会从某形状扩散到整个空间。这个工作将点云的点分布和噪声分布建立关联。通过学习寻找逆分布，从而从噪声中恢复原始点分布。



本文的创新点：

- 提出了一个基于Diffusion的点云生成；
- 提出了一个易于训练的Diffusion Loss；
- 使用自动编码器获得点云总体特征（就是认为它潜在应该是什么模型，让扩散不那么随机）
- 提供了从预估分布中采样点云的方法；

2. Diffusion Probabilistic Models for Point Clouds

这部分定义了模型训练的前向和后向扩散的概率模型，最后定义了训练Loss。

2.1. Formulation

遵循热力学和点云定义，定义点云 $X^{(0)} = \{x_i^{(0)}\}_{i=1}^N$ 为一组热力学系统中的粒子，每一个粒子 x_i 都可被独立采样于点分布 $q(x_i^{(0)}|z)$ ，其中 z 为 Shape Latent（决定点分布）。

遵循Diffusion，点到随机噪声这一扩散过程可描述为以下蒙特卡洛链：

$$q(x_i^{(1:T)}|x_i^{(0)}) = \prod_{t=1}^T q(x_i^{(t)}|x_i^{(t-1)})$$

$$\text{where } q(x^{(t)}|x^{(t-1)}) = \mathcal{N}(x^{(t)}|\sqrt{1-\beta_t}x^{(t-1)}, \beta_t\mathbf{I}), t = 1, \dots, T$$

由于目标是根据潜在编码 z 生成点云，因此定义逆向扩散过程。

1. 从近似于 $q(x_i^{(t)})$ 的分布 $q(x_i^{(t)})$ 采样一组点作为输入；
2. 通过蒙特卡洛链逆向回期望模型；

相比于前向的简单加噪声，逆向的模型是未知的，需要学习的。逆向过程可描述为：

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

$$p_\theta(x^{(t-1)}|x^{(t)}, z) = \mathcal{N}(x^{(t-1)}|\mu_\theta(x^{(t)}, t, z), \beta_t\mathbf{I}), \text{ where } p(x^{(T)}) \sim \mathcal{N}(0, \mathbf{I})$$

由于输入点云是从分布 $p(x_i^t)$ 中采样的，因此整个点云的概率就是所有样本点的乘积：

$$q(X^{(1:T)}|X^0) = \prod_{i=1}^N q(x_i^{(1:T)}|x_i^{(0)})$$

$$p_\theta(X^{(1:T)}|z) = \prod_{i=1}^N p_\theta(x_i^{(1:T)}|z)$$

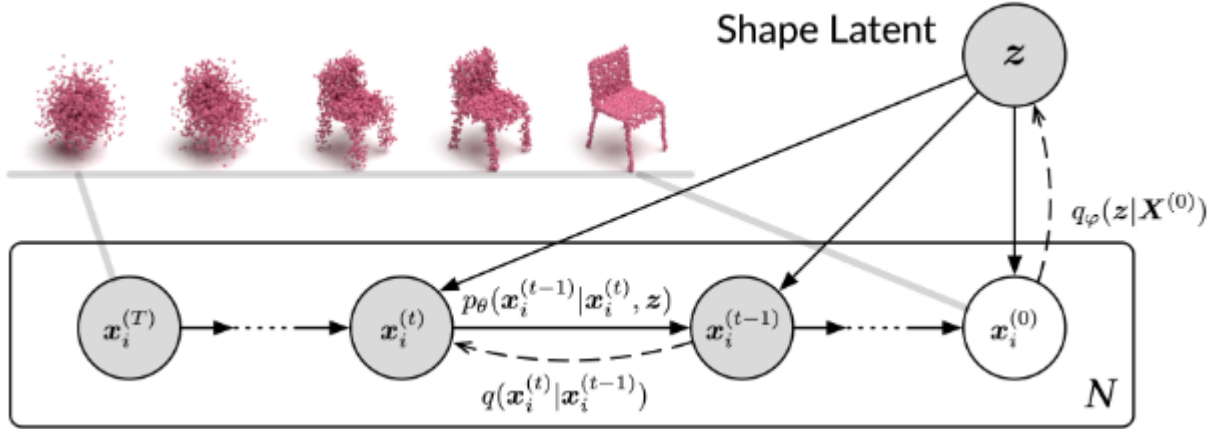
2.2. Training Objective

训练反向扩散的目的是使点云的似然估计 $\mathbb{E}[\log p_\theta(X^{(0)})]$ 最大化。但直接算不行（见EBM），因此使用最大化变分下界化简：

$$\begin{aligned}
\mathbb{E}[\log p_{\theta}(X^{(0)})] &\geq \mathbb{E}_q \left[\log \frac{p_{\theta}(X^{(0:T)}, z)}{q(X^{(1:T)}, z | X^{(0)})} \right] \\
&= E_q \left[\log p(X^{(T)}) \right. \\
&\quad + \sum_{t=1}^T \log \frac{p_{\theta}(X^{(t-1)} | X^{(t)}, z)}{q(X^{(T)} | X^{(t-1)})} \\
&\quad \left. - \log \frac{q_{\varphi}(z | X^{(0)})}{p(z)} \right]
\end{aligned}$$

通过推导，由此得到本文用于训练的Loss：

$$\begin{aligned}
\mathcal{L}(\theta, \varphi) &= \mathbb{E}_q \left[\sum_{t=2}^T D_{KL}(q(X^{(t-1)} | X^{(t)}, X^{(0)}) || p_{\theta}(X^{(t-1)} | X^{(t)}, z)) \right. \\
&\quad - \log p_{\theta}(X^{(0)} | X^{(1)}, z) \\
&\quad \left. + D_{KL}(q_{\varphi}(z | X^{(0)}) || p(z)) \right]
\end{aligned}$$



由于样本的采样的独立性，可继续细化Loss：

$$\begin{aligned}
\mathcal{L}(\theta, \varphi) &= \mathbb{E}_q \left[\sum_{t=2}^T \sum_{i=1}^N \underbrace{D_{KL}(q(x_i^{(t-1)} | x_i^{(t)}, x_i^{(0)}) || p_{\theta}(x_i^{(t-1)} | x_i^{(t)}, z))}_{(1)} \right. \\
&\quad - \sum_{i=1}^N \underbrace{\log p_{\theta}(x_i^{(0)} | x_i^{(1)}, z)}_{(3)} \\
&\quad \left. + \underbrace{D_{KL}(q_{\varphi}(z | X^{(0)}) || p(z))}_{(4) \text{ (5)}} \right]
\end{aligned}$$

1. 该项可通过Closed-form高斯计算[paper](#);
2. $p_{\theta}(x_i^{(t-1)}|x_i^{(t)}, z)$ 可由[前面](#)计算;
3. 同上;
4. (important) $q_{\varphi}(z|X^{(0)})$ 是一个近似后验分布, 通俗理解就是一个变分编码器, 通过输入原始点云得到一个点云总体的潜在编码 z , 不同的是这里用了分布描述, 实际使用PointNet预估;
5. 最后一项是一个先验分布, 本文选择使用参数化先验分布 (normalizing flows) 实现;

但是计算每一个节点的期望同样耗时, 所以提出一个优化: 随机计算一个 t 的期望, 增加一小段训练时间, 提高总体效率。

- 蓝色部分为Diffusion模型的Loss;
- 绿色部分是Flow模型需要计算的Loss, 对于AutoEncoder这个模块, 不需要计算这个交叉熵。

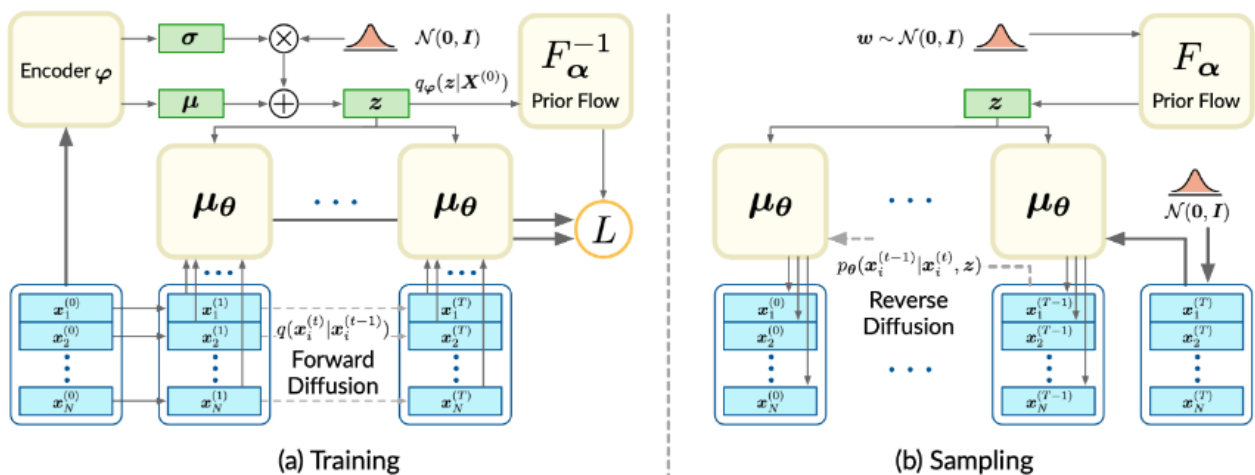
Algorithm 1 Training (Simplified)

```

1: repeat
2:   Sample  $X^{(0)} \sim q_{\text{data}}(X^{(0)})$ 
3:   Sample  $z \sim q_{\varphi}(z|X^{(0)})$ 
4:   Sample  $t \sim \text{Uniform}(\{1, \dots, T\})$ 
5:   Sample  $x_1^{(t)}, \dots, x_N^{(t)} \sim q(x^{(t)}|x^{(0)})$ 
6:    $L_t \leftarrow \sum_{i=1}^N D_{\text{KL}}(q(x_i^{(t-1)}|x_i^{(t)}, x_i^{(0)}) \| p_{\theta}(x_i^{(t-1)}|x_i^{(t)}, z))$ 
7:    $L_z \leftarrow D_{\text{KL}}(q_{\varphi}(z|X^{(0)}) \| p(z))$ 
8:   Compute  $\nabla_{\theta}(L_t + \frac{1}{T}L_z)$ . Then perform gradient descent.
9: until converged

```

2.3. Model Implementations



如上, 模型训练Loss分两个部分, DiffusionLoss和自编码Loss。

在这块的Work:

- 找到一个双向分布映射 $F_\alpha: w \rightarrow z$, 可以将一个各向同性分布映射为一个参数化的复分布, 且能直接计算复分布样本的概率 $p(z)$;
- 对于编码器 $q_\varphi(z|X^{(0)})$, 使用PointNet实现对均值和方差的预估;
- 提出了一个AutoEncoder, 对应Loss部分蓝绿Loss相关内容。

3. How 2 use Diffusion for denoising only?

首先, 之前点云降噪任务和这个工作:

- 降噪过程: 这篇是从 $\mathcal{N}(0, \mathbf{I})$ 将点云**重采样**为目标形状; 之前降噪往往对点云Patch进行估计某个属性 (e.g. 偏移量、梯度方向...);
- 泛化: 这篇提出的点云生成结果依赖于训练用的数据集, 它无法生成它不认识的模型; 大部分降噪方法往往不受模型本身影响, 少数 (e.g. ShapeGF) 除外;
- ...

如何使用Diffusion实现点云降噪, 思路分析:

1. 噪声点云, 若它可被一个分布表示, 例如高斯分布 $q(X^T|X_0)$, 那它是不是可以被视为Sampling过程中 t 较小的比较靠前的链中的某个 X^t ? 那么这个问题就可以被约化为寻找一个预测 t 的方法, 根据输入噪声点云预测 t , 而PointNet能做预测属性。但难点目前存在以下几个方面:
 - PointNet能否预测 t ?
 - 需要一个用于控制DiffusionSampling的 z , 它的输入是噪声点云, 目的是让最终生成的模型是噪声点云潜在描述的那个样子。有如下难点:
 - 这个分布的输入一个各向异性分布, 输出一个潜在特征;
 - 如何计算Loss? 或者, 不用算? Flow能做吗? AutoEncoder能做吗?
2. 从之前的Score-Based方法中, 它们通过猜Patch中平面的分布从而实现降噪。那么对于Diffusion来说, 每一Step都是通过输入猜噪声分布, 然后再通过重采样采样下一个Step, 从而实现从噪声到实体的生成过程。那么, 能否使用Diffusion对点云Patch进行降噪? 因为对于Diffusion的每Step可以理解为通过重采样采样一个ScoreFunction, 从而实现降噪。