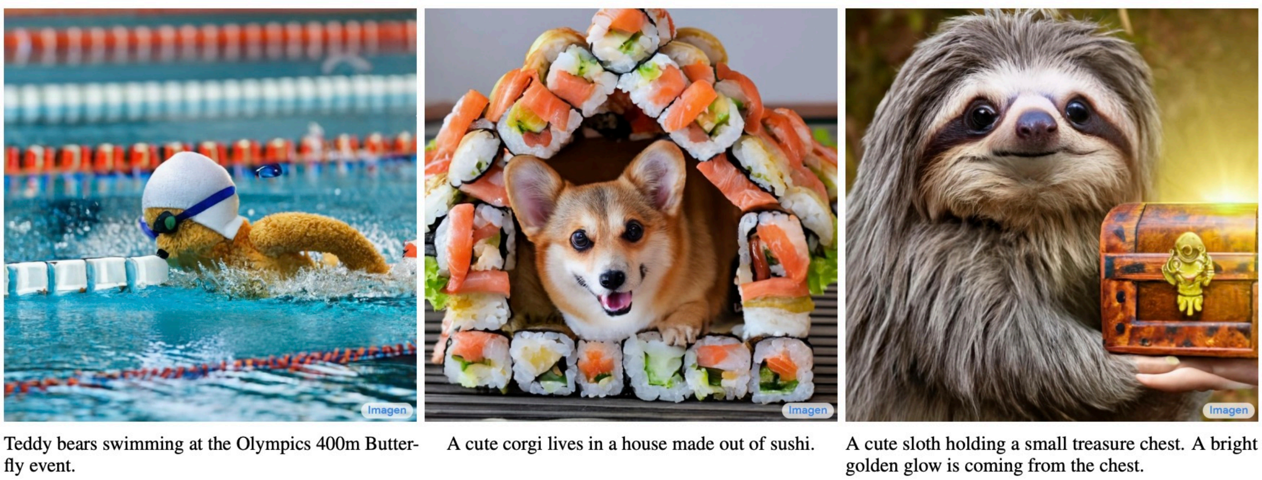
13 Jun

生成扩散模型漫谈（一）：DDPM = 拆楼 + 建楼

By 苏剑林 | 2022-06-13 | 393459位读者 引用

说到生成模型，VAE、GAN可谓是“如雷贯耳”，本站也有过多次分享。此外，还有一些比较小众的选择，如flow模型、VQ-VAE等，也颇有人气，尤其是VQ-VAE及其变体VQ-GAN，近期已经逐渐发展到“图像的Tokenizer”的地位，用来直接调用NLP的各种预训练方法。除了这些之外，还有一个本来更小众的选择——扩散模型（Diffusion Models）——正在生成模型领域“异军突起”，当前最先进的两个文本生成图像——OpenAI的DALL·E 2和Google的Imagen，都是基于扩散模型来完成的。



Imagen“文本-图片”的部分例子

从本文开始，我们开一个新坑，逐渐介绍一下近两年关于生成扩散模型的一些进展。据说生成扩散模型以数学复杂闻名，似乎比VAE、GAN要难理解得多，是否真的如此？扩散模型真的做不到一个“大白话”的理解？让我们拭目以待。

新的起点

其实我们在之前的文章《能量视角下的GAN模型（三）：生成模型=能量模型》、《从去噪自编码器到生成模型》也简单介绍过扩散模型。说到扩散模型，一般的文章都会提到能量模型（Energy-based Models）、得分匹配（Score Matching）、朗之万方程（La ngevin Equation）等等，简单来说，是通过得分匹配等技术来训练能量模型，然后通过郎之万方程来执行从能量模型的采样。

从理论上来讲，这是一套很成熟的方案，原则上可以实现任何连续型对象（语音、图像等）的生成和采样。但从实践角度来看，能量函数的训练是一件很艰难的事情，尤其是数据维度比较大（比如高分辨率图像）时，很难训练出完备能量函数来；另一方面，通过朗之万方程从能量模型的采样也有很大的不确定性，得到的往往是带有噪声的采样结果。所以很长时间以来，这种传统路径的扩散模型只是在比较低分辨率的图像上做实验。

如今生成扩散模型的大火，则是始于2020年所提出的DDPM（Denoising Diffusion Probabilistic Model），虽然也用了“扩散模型”这个名字，但事实上除了采样过程的形式有一定的相似之外，DDPM与传统基于朗之万方程采样的扩散模型可以说完全不一样，这完全是一个新的起点、新的篇章。

准确来说，DDPM叫“渐变模型”更为准确一些，扩散模型这一名字反而容易造成理解上的误解，传统扩散模型的能量模型、得分匹配、朗之万方程等概念，其实跟DDPM及其后续变体都没什么关系。有意思的是，DDPM的数学框架其实在ICML2015的论文《Deep Unsupervised Learning using Nonequilibrium Thermodynamics》就已经完成了，但DDPM是首次将它在高分辨率图像生成上调试出来了，从而引导出了后面的火热。由此可见，一个模型的诞生和流行，往往还需要时间和机遇，

拆楼建楼

很多文章在介绍DDPM时，上来就引入转移分布，接着就是变分推断，一堆数学记号下来，先吓跑了一群人（当然，从这种介绍我们可以再次看出，DDPM实际上是VAE而不是扩散模型），再加之人们对传统扩散模型的固有印象，所以就形成了“需要很高深的数学知识”的错觉。事实上，DDPM也可以有一种很“大白话”的理解，它并不比有着“造假-鉴别”通俗类比的GAN更难。

首先，我们想要做一个像GAN那样的生成模型，它实际上是将一个随机噪声$\boldsymbol{z}$变换成一个数据样本$\boldsymbol{x}$的过程：

$$\begin{array}{ccc}\text{随机噪声}\boldsymbol{z} & \xrightarrow{\quad\text{变换}\quad} & \text{样本数据}\boldsymbol{x}\\ \scriptstyle{\text{类比}}\Big\downarrow & & \Big\downarrow\scriptstyle{\text{类比}}\\ \text{砖瓦水泥} & \xrightarrow{\quad\text{建设}\quad} & \text{高楼大厦}\end{array}\tag{1}$$

我们可以将这个过程想象为“建设”，其中随机噪声$\boldsymbol{z}$是砖瓦水泥等原材料，样本数据$\boldsymbol{x}$是高楼大厦，所以生成模型就是一支用原材料建设高楼大厦的施工队。

请叫我工程师

这个过程肯定很难的，所以才有了那么多关于生成模型的研究。但俗话说“破坏容易建设难”，建楼你不会，拆楼你总会了吧？我们考虑将高楼大厦一步步地拆为砖瓦水泥的过程：设$\boldsymbol{x}_0$为建好的高楼大厦（数据样本），$\boldsymbol{x}_T$为拆好的砖瓦水泥（随机噪声），假设“拆楼”需要T步，整个过程可以表示为

$$\boldsymbol{x} = \boldsymbol{x}_0 \to \boldsymbol{x}_1 \to \boldsymbol{x}_2 \to \cdots \to \boldsymbol{x}_{T-1} \to \boldsymbol{x}_T = \boldsymbol{z}\tag{2}$$

建高楼大厦的难度在于，从原材料$\boldsymbol{x}_T$到最终高楼大厦$\boldsymbol{x}_0$的跨度过大，普通人很难理解$\boldsymbol{x}_T$是怎么一下子变成$\boldsymbol{x}_0$的。但是，当我们有了“拆楼”的中间过程$\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T$后，我们知道$\boldsymbol{x}_{t-1} \to \boldsymbol{x}_t$代表着拆楼的一步，那么反过来$\boldsymbol{x}_t \to \boldsymbol{x}_{t-1}$不就是建楼的一步？如果我们能学会两者之间的变换关系$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}(\boldsymbol{x}_t)$，那么从$\boldsymbol{x}_T$出发，反复地执行$\boldsymbol{x}_{T-1} = \boldsymbol{\mu}(\boldsymbol{x}_T)$、$\boldsymbol{x}_{T-2} = \boldsymbol{\mu}(\boldsymbol{x}_{T-1})$、...，最终不就能造出高楼大厦$\boldsymbol{x}_0$出来？

该怎么拆

正所谓“饭要一口一口地吃”，楼也要一步一步地建，DDPM做生成模型的过程，其实跟上述“拆楼-建楼”的类比是完全一致的，它也是先反过来构建一个从数据样本渐变到随机噪声的过程，然后再考虑其逆变换，通过反复执行逆变换来完成数据样本的生成，所以本文前面才说DDPM这种做法其实应该更准确地称为“渐变模型”而不是“扩散模型”。

具体来说，DDPM将“拆楼”的过程建模为

$$\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_{t-1} + \beta_t \boldsymbol{\varepsilon}_t,\quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{3}$$

其中有$\alpha_t, \beta_t > 0$且$\alpha_t^2 + \beta_t^2 = 1$，$\beta_t$通常很接近于0，代表着单步“拆楼”中对原来楼体的破坏程度，噪声$\boldsymbol{\varepsilon}_t$的引入代表着对原始信号的一种破坏，我们也可以将它理解为“原材料”，即每一步“拆楼”中我们都将$\boldsymbol{x}_{t-1}$拆解为“$\alpha_t \boldsymbol{x}_{t-1}$的楼体 + $\beta_t \boldsymbol{\varepsilon}_t$的原料”。（**提示：**本文$\alpha_t, \beta_t$的定义跟原论文不一样。）

反复执行这个拆楼的步骤，我们可以得到：

$$\begin{aligned}
\boldsymbol{x}_t &= \alpha_t \boldsymbol{x}_{t-1} + \beta_t \boldsymbol{\varepsilon}_t \\
&= \alpha_t \big(\alpha_{t-1} \boldsymbol{x}_{t-2} + \beta_{t-1} \boldsymbol{\varepsilon}_{t-1}\big) + \beta_t \boldsymbol{\varepsilon}_t \\
&= \cdots \\
&= (\alpha_t \cdots \alpha_1) \boldsymbol{x}_0 + \underbrace{(\alpha_t \cdots \alpha_2)\beta_1 \boldsymbol{\varepsilon}_1 + (\alpha_t \cdots \alpha_3)\beta_2 \boldsymbol{\varepsilon}_2 + \cdots + \alpha_t \beta_{t-1} \boldsymbol{\varepsilon}_{t-1} + \beta_t \boldsymbol{\varepsilon}}_{\text{多个相互独立的正态噪声之和}}
\end{aligned}$$

可能刚才读者就想问为什么叠加的系数要满足$\alpha_t^2 + \beta_t^2 = 1$了，现在我们就可以回答这个问题。首先，式中花括号所指出的部分，正好是多个独立的正态噪声之和，其均值为0，方差则分别为$(\alpha_t \cdots \alpha_2)^2 \beta_1^2$、$(\alpha_t \cdots \alpha_3)^2 \beta_2^2$、…、$\alpha_t^2 \beta_{t-1}^2$、$\beta_t^2$；然后，我们利用一个概率论的知识——正态分布的叠加性，即上述多个独立的正态噪声之和的分布，实际上是均值为0、方差为$(\alpha_t \cdots \alpha_2)^2 \beta_1^2 + (\alpha_t \cdots \alpha_3)^2 \beta_2^2 + \cdots + \alpha_t^2 \beta_{t-1}^2 + \beta_t^2$的正态分布；最后，在$\alpha_t^2 + \beta_t^2 = 1$恒成立之下，我们可以得到式(4)的各项系数平方和依旧为1，即

$$(\alpha_t \cdots \alpha_1)^2 + (\alpha_t \cdots \alpha_2)^2 \beta_1^2 + (\alpha_t \cdots \alpha_3)^2 \beta_2^2 + \cdots + \alpha_t^2 \beta_{t-1}^2 + \beta_t^2 = 1 \tag{5}$$

所以实际上相当于有

$$\boldsymbol{x}_t = \underbrace{(\alpha_t \cdots \alpha_1)}_{\text{记为}\bar{\alpha}_t} \boldsymbol{x}_0 + \underbrace{\sqrt{1 - (\alpha_t \cdots \alpha_1)^2}}_{\text{记为}\bar{\beta}_t} \bar{\boldsymbol{\varepsilon}}_t,\quad \bar{\boldsymbol{\varepsilon}}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{6}$$

这就为计算$\boldsymbol{x}_t$提供了极大的便利。另一方面，DDPM会选择适当的α_t形式，使得有

$\bar{\alpha}_T \approx 0$，这意味着经过T步的拆楼后，所剩的楼体几乎可以忽略了，已经全部转化为原材料$\boldsymbol{\varepsilon}$。（**提示：**本文$\bar{\alpha}_t$的定义跟原论文不一样。）

又如何建

“拆楼”是$\boldsymbol{x}_{t-1} \to \boldsymbol{x}_t$的过程，这个过程我们得到很多的数据对$(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$，那么“建楼”自然就是从这些数据对中学习一个$\boldsymbol{x}_t \to \boldsymbol{x}_{t-1}$的模型。设该模型为$\boldsymbol{\mu}(\boldsymbol{x}_t)$，那么容易想到学习方案就是最小化两者的欧氏距离：

$$\left\Vert\boldsymbol{x}_{t-1} - \boldsymbol{\mu}(\boldsymbol{x}_t)\right\Vert^2 \tag{7}$$

其实这已经非常接近最终的DDPM模型了，接下来让我们将这个过程做得更精细一些。首先“拆楼”的式(3)可以改写为$\boldsymbol{x}_{t-1} = \frac{1}{\alpha_t}(\boldsymbol{x}_t - \beta_t\boldsymbol{\varepsilon}_t)$，这启发我们或许可以将“建楼”模型$\boldsymbol{\mu}(\boldsymbol{x}_t)$设计成

$$\boldsymbol{\mu}(\boldsymbol{x}_t) = \frac{1}{\alpha_t}\left(\boldsymbol{x}_t - \beta_t\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right) \tag{8}$$

的形式，其中$\boldsymbol{\theta}$是训练参数，将其代入到损失函数，得到

$$\left\Vert\boldsymbol{x}_{t-1} - \boldsymbol{\mu}(\boldsymbol{x}_t)\right\Vert^2 = \frac{\beta_t^2}{\alpha_t^2}\left\Vert\boldsymbol{\varepsilon}_t - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right\Vert^2 \tag{9}$$

前面的因子$\frac{\beta_t^2}{\alpha_t^2}$代表loss的权重，这个我们可以暂时忽略，最后代入结合式(6)和(3)所给出$\boldsymbol{x}_t$的表达式

$$\boldsymbol{x}_t = \alpha_t\boldsymbol{x}_{t-1} + \beta_t\boldsymbol{\varepsilon}_t = \alpha_t\left(\bar{\alpha}_{t-1}\boldsymbol{x}_0 + \bar{\beta}_{t-1}\bar{\boldsymbol{\varepsilon}}_{t-1}\right) + \beta_t\boldsymbol{\varepsilon}_t = \bar{\alpha}_t\boldsymbol{x}_0 + \alpha_t\bar{\beta}_{t-1}\bar{\boldsymbol{\varepsilon}}_{t-1} +$$

得到损失函数的形式为

$$\left\Vert\boldsymbol{\varepsilon}_t - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\bar{\alpha}_t\boldsymbol{x}_0 + \alpha_t\bar{\beta}_{t-1}\bar{\boldsymbol{\varepsilon}}_{t-1} + \beta_t\boldsymbol{\varepsilon}_t, t)\right\Vert^2 \tag{11}$$

可能读者想问为什么要回退一步来给出$\boldsymbol{x}_t$，直接根据式(6)来给出$\boldsymbol{x}_t$可以吗？答案是不

行，因为我们已经事先采样了$\boldsymbol{\varepsilon}_t$，而$\boldsymbol{\varepsilon}_t$跟$\bar{\boldsymbol{\varepsilon}}_t$不是相互独立的，所以给定$\boldsymbol{\varepsilon}_t$的情况下，我们不能完全独立地采样$\bar{\boldsymbol{\varepsilon}}_t$。

降低方差

原则上来说，损失函数(11)就可以完成DDPM的训练，但它在实践中可能有方差过大的风险，从而导致收敛过慢等问题。要理解这一点并不困难，只需要观察到式(11)实际上包含了4个需要采样的随机变量：

> 1、从所有训练样本中采样一个$\boldsymbol{x}_0$；
>
> 2、从正态分布$\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$中采样$\bar{\boldsymbol{\varepsilon}}_{t-1},\boldsymbol{\varepsilon}_t$（两个不同的采样结果）；
>
> 3、从$1\sim T$中采样一个t。

要采样的随机变量越多，就越难对损失函数做准确的估计，反过来说就是每次对损失函数进行估计的波动（方差）过大了。很幸运的是，我们可以通过一个积分技巧来将$\bar{\boldsymbol{\varepsilon}}_{t-1},\boldsymbol{\varepsilon}_t$合并成单个正态随机变量，从而缓解一下方差大的问题。

这个积分确实有点技巧性，但也不算复杂。由于正态分布的叠加性，我们知道$\alpha_t\bar{\beta}_{t-1}\bar{\boldsymbol{\varepsilon}}_{t-1}+\beta_t\boldsymbol{\varepsilon}_t$实际上相当于单个随机变量$\bar{\beta}_t\boldsymbol{\varepsilon}|\boldsymbol{\varepsilon}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$，同理$\beta_t\bar{\boldsymbol{\varepsilon}}_{t-1}-\alpha_t\bar{\beta}_{t-1}\boldsymbol{\varepsilon}_t$实际上相当于单个随机变量$\bar{\beta}_t\boldsymbol{\omega}|\boldsymbol{\omega}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$，并且可以验证$\mathbb{E}[\boldsymbol{\varepsilon}\boldsymbol{\omega}^{\top}]=\boldsymbol{0}$，所以这是两个相互独立的正态随机变量。

接下来，我们反过来将$\boldsymbol{\varepsilon}_t$用$\boldsymbol{\varepsilon},\boldsymbol{\omega}$重新表示出来

$$\boldsymbol{\varepsilon}_t=\frac{(\beta_t\boldsymbol{\varepsilon}-\alpha_t\bar{\beta}_{t-1}\boldsymbol{\omega})\bar{\beta}_t}{\beta_t^2+\alpha_t^2\bar{\beta}_{t-1}^2}=\frac{\beta_t\boldsymbol{\varepsilon}-\alpha_t\bar{\beta}_{t-1}\boldsymbol{\omega}}{\bar{\beta}_t} \tag{12}$$

代入到式(11)得到

$$\begin{aligned}&\mathbb{E}_{\bar{\boldsymbol{\varepsilon}}_{t-1},\boldsymbol{\varepsilon}_t\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}\left[\left\Vert\boldsymbol{\varepsilon}_t - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\bar{\alpha}_t\boldsymbol{x}_0 + \alpha_t\bar{\beta}_{t-1}\bar{\boldsymbol{\varepsilon}}_{t-1} + \beta_t\boldsymbol{\varepsilon}_t, t)\right\Vert^2\right]\\ =&\,\mathbb{E}_{\boldsymbol{\omega},\boldsymbol{\varepsilon}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}\left[\left\Vert\frac{\beta_t\boldsymbol{\varepsilon} - \alpha_t\bar{\beta}_{t-1}\boldsymbol{\omega}}{\bar{\beta}_t} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\bar{\alpha}_t\boldsymbol{x}_0 + \bar{\beta}_t\boldsymbol{\varepsilon}, t)\right\Vert^2\right]\end{aligned}\tag{13}$$

注意到，现在损失函数关于$\boldsymbol{\omega}$只是二次的，所以我们可以展开然后将它的期望直接算出来，结果是

$$\frac{\beta_t^2}{\bar{\beta}_t^2}\mathbb{E}_{\boldsymbol{\varepsilon}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}\left[\left\Vert\boldsymbol{\varepsilon} - \frac{\bar{\beta}_t}{\beta_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\bar{\alpha}_t\boldsymbol{x}_0 + \bar{\beta}_t\boldsymbol{\varepsilon}, t)\right\Vert^2\right] + \text{常数}\tag{14}$$

再次省掉常数和损失函数的权重，我们得到DDPM最终所用的损失函数：

$$\left\Vert\boldsymbol{\varepsilon} - \frac{\bar{\beta}_t}{\beta_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\bar{\alpha}_t\boldsymbol{x}_0 + \bar{\beta}_t\boldsymbol{\varepsilon}, t)\right\Vert^2\tag{15}$$

（**提示：**原论文中的$\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$实际上就是本文的$\frac{\bar{\beta}_t}{\beta_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$，所以大家的结果是完全一样的。）

递归生成

至此，我们算是把DDPM的整个训练流程捋清楚了。内容写了不少，你要说它很容易，那肯定说不上，但真要说非常困难的地方也几乎没有——没有用到传统的能量函数、得分匹配等工具，甚至连变分推断的知识都没有用到，只是借助“拆楼-建楼”的类比和一些基本的概率论知识，就能得到完全一样的结果。所以说，以DDPM为代表的新兴起的生成扩散模型，实际上没有很多读者想象的复杂，它可以说是我们从“拆解-重组”的过程中学习新知识的形象建模。

训练完之后，我们就可以从一个随机噪声$\boldsymbol{x}_T\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$出发执行$T$步式(8)来进行生成：

$$\boldsymbol{x}_{t-1} = \frac{1}{\alpha_t}\left(\boldsymbol{x}_t - \beta_t\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right)\tag{16}$$

这对应于自回归解码中的Greedy Search。如果要进行Random Sample，那么需要补上

噪声项：

$$\boldsymbol{x}_{t-1} = \frac{1}{\alpha_t}\left(\boldsymbol{x}_t - \beta_t \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{z},\quad \boldsymbol{z}\sim\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{17}$$

一般来说，我们可以让$\sigma_t = \beta_t$，即正向和反向的方差保持同步。这个采样过程跟传统扩散模型的朗之万采样不一样的地方在于：DDPM的采样每次都从一个随机噪声出发，需要重复迭代T步来得到一个样本输出；朗之万采样则是从任意一个点出发，反复迭代无限步，理论上这个迭代无限步的过程中，就把所有数据样本都被生成过了。所以两者除了形式相似外，实质上是两个截然不同的模型。

从这个生成过程中，我们也可以感觉到它其实跟Seq2Seq的解码过程是一样的，都是串联式的自回归生成，所以生成速度是一个瓶颈，DDPM设了$T = 1000$，意味着每生成一个图片，需要将$\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$反复执行1000次，因此DDPM的一大缺点就是采样速度慢，后面有很多工作都致力于提升DDPM的采样速度。而说到“图片生成 + 自回归模型 + 很慢”，有些读者可能会联想到早期的PixelRNN、PixelCNN等模型，它们将图片生成转换成语言模型任务，所以同样也是递归地进行采样生成以及同样地慢。那么DDPM的这种自回归生成，跟PixelRNN/PixelCNN的自回归生成，又有什么实质区别呢？为什么PixelRNN/PixelCNN没大火起来，反而轮到了DDPM？

了解PixelRNN/PixelCNN的读者都知道，这类生成模型是逐个像素逐个像素地生成图片的，而自回归生成是有序的，这就意味着我们要提前给图片的每个像素排好顺序，最终的生成效果跟这个顺序紧密相关。然而，目前这个顺序只能是人为地凭着经验来设计（这类经验的设计都统称为“Inductive Bias”），暂时找不到理论最优解。换句话说，PixelRNN/PixelCNN的生成效果很受Inductive Bias的影响。但DDPM不一样，它通过“拆楼”的方式重新定义了一个自回归方向，而对于所有的像素来说则都是平权的、无偏的，所以减少了Inductive Bias的影响，从而提升了效果。此外，DDPM生成的迭代步数是固定的T，而PixelRNN/PixelCNN则是等于图像分辨率（$\text{宽}\times\text{高}\times\text{通道数}$），所以DDPM生成高分辨率图像的速度要比PixelRNN/PixelCNN快得多。

超参设置

这一节我们讨论一下超参的设置问题。

在DDPM中，$T=1000$，可能比很多读者的想象数值要大，那为什么要设置这么大的T呢？另一边，对于α_t的选择，将原论文的设置翻译到本博客的记号上，大致上是

$$\alpha_t = \sqrt{1-\frac{0.02t}{T}} \tag{18}$$

这是一个单调递减的函数，那为什么要选择单调递减的α_t呢？

其实这两个问题有着相近的答案，跟具体的数据背景有关。简单起见，在重构的时候我们用了欧氏距离(7)作为损失函数，而一般我们用DDPM做图片生成，以往做过图片生成的读者都知道，欧氏距离并不是图片真实程度的一个好的度量，VAE用欧氏距离来重构时，往往会得到模糊的结果，除非是输入输出的两张图片非常接近，用欧氏距离才能得到比较清晰的结果，所以选择尽可能大的T，正是为了使得输入输出尽可能相近，减少欧氏距离带来的模糊问题。

选择单调递减的α_t也有类似考虑。当t比较小时，$\boldsymbol{x}_t$还比较接近真实图片，所以我们要缩小$\boldsymbol{x}_{t-1}$与$\boldsymbol{x}_t$的差距，以便更适用欧氏距离(7)，因此要用较大的α_t；当t比较大时，$\boldsymbol{x}_t$已经比较接近纯噪声了，噪声用欧式距离无妨，所以可以稍微增大$\boldsymbol{x}_{t-1}$与$\boldsymbol{x}_t$的差距，即可以用较小的α_t。那么可不可以一直用较大的α_t呢？可以是可以，但是要增大T。注意在推导(6)时，我们说过应该有$\bar{\alpha}_T\approx 0$，而我们可以直接估算

$$\log\bar{\alpha}_T = \sum_{t=1}^T \log\alpha_t = \frac{1}{2}\sum_{t=1}^T \log\left(1-\frac{0.02t}{T}\right) < \frac{1}{2}\sum_{t=1}^T \left(-\frac{0.02t}{T}\right) = -0.005(T$$

代入$T=1000$大致是$\bar{\alpha}_T\approx e^{-5}$，这个其实就刚好达到$\approx 0$的标准。所以如果从头到尾都用较大的$\alpha_t$，那么必然要更大的$T$才能使得$\bar{\alpha}_T\approx 0$了。

最后我们留意到，“建楼”模型中的$\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\bar{\alpha}_t\boldsymbol{x}_0+\bar{\beta}_t\boldsymbol{\varepsilon}, t)$中，我们在输入中显式地写出了$t$，这是因为原则上不同的$t$处理的是不同层次的对象，所以应该用不同的重构模型，即应该有T个不同的重构模型才对，于是我们共享了所有重构模型的参数，将t作为条件传入。按照论文附录的说法，t是转换成《Transformer升级之路：1、Sinusoidal位置编码追根溯源》介绍的位置编码后，直接加到残差模块上去的。

文章小结

本文从“拆楼-建楼”的通俗类比中介绍了最新的生成扩散模型DDPM，在这个视角中，我们可以通过较为“大白话”的描述以及比较少的数学推导，来得到跟原始论文一模一样的结果。总的来说，本文说明了DDPM也可以像GAN一样找到一个形象类比，它既可以不用到VAE中的“变分”，也可以不用到GAN中的“概率散度”、“最优传输”，从这个意义上来看，DDPM甚至算得上比VAE、GAN还要简单。

如果您需要引用本文，请参考：

苏剑林. (Jun. 13, 2022). 《生成扩散模型漫谈（一）：DDPM = 拆楼 + 建楼 》[Blog post]. Retrieved from https://spaces.ac.cn/archives/9119

```
@online{kexuefm-9119,
        title={生成扩散模型漫谈（一）：DDPM = 拆楼 + 建楼},
        author={苏剑林},
        year={2022},
        month={Jun},
        url={\url{https://spaces.ac.cn/archives/9119}},
}
```