Generative Models

> 这是我写的第一篇关于深度学习的学习笔记，本来想用英文写的，毕竟看的都是英文的论文，自己也身处英语环境里，但想到将来时要回归国内环境，而且本文面向的群体也都是中文受众，所以就主体以中文来写吧~

生成模型类别

这里会总结我学习过的模型类别并根据我的理解，按顺序给出我觉得重要的点。后续会持续更新~

- AE (Autoencoder)
- VAE (Variational Autoencoder)
- GAN (Generative Adversarial Network)
- Diffusion models
- NeRF (Neural Radiance Fields)

1. 自编码器 (Autoencoder)

首先贴上原文: Reducing the Dimensionality of Data with Neural Networks。

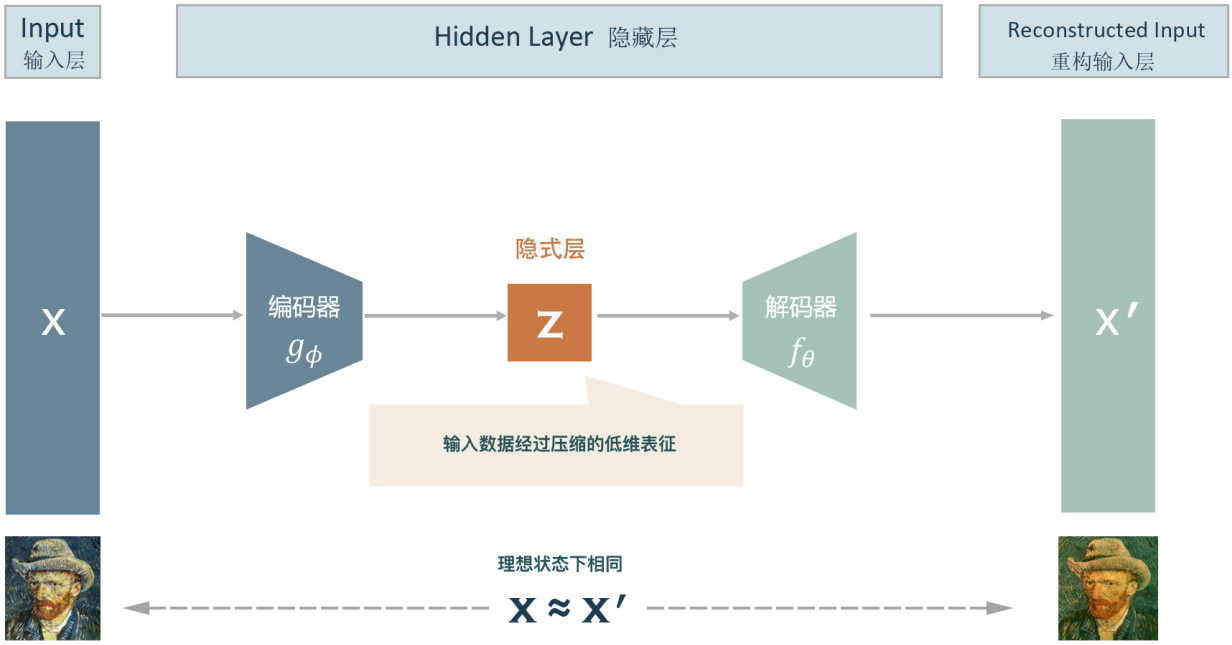
AE是为了**表征学习**（**representation learning**）所使用的一种**无监督学习**（**unsupervised learning**）方法。相较于传统深度学习把数据直接输入的方法，AE会通过给网络加一个**隐式**（**latent**）层的方式把输入数据强制压缩成表征形式，从而使后续其他网络在使用该输入数据时可以在低维度的（low-dimentional）表征层面（即参数）进行。通过这种压缩降维的方式，可以令深度学习网络的效率得到提高。

> AE就是把自己先变低维再变高维变回自己，生成低维参数。

1.1 AE结构介绍

AE可以被视为一个三层神经网络结构：输入层、隐藏层和输出层。其中隐藏层由两个网络构成：

- *Encoder* network （*编码器* 网络）：保证输入层神经元个数大于重构输出层神经元个数，将高维数据转化成低维数据。
- *Decoder* network （*解码器* 网络）：将数据从低维还原到高维，可能会使输入层增大。



在这个AE模型中，输入数据 $\mathbf{x}$ 在经过以 ϕ 为参数的的编码器函数 $g_\phi(.)$ 压缩后变成低维数据 $\mathbf{z}=g_\phi(\mathbf{x})$,再经过以 θ 为参数的解码器函数 $f_\theta(.)$ 还原成高维数据 $\mathbf{x}'=f_\theta(g_\phi(\mathbf{x}))$ 。两个参数 ϕ 和 θ 是同时训练的，目的是使 $\mathbf{x}\approx\mathbf{x}'$,以均方误差举例即满足：

$$L_{AE}(\theta)=\frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}^i-f_\theta(g_\phi(\mathbf{x}^i)))^2$$

1.2 稀疏自编码（Sparse Autoencoder）

作为当前深度学习领域各网络所追求的很重要的特性，**稀疏性**（**sparse**）一直是解决高维度数据计算量问题很有效的方式，因为具有稀疏性的方法可以避免过拟合来提升鲁棒性。

若在AE模型中，令输出层神经元个数大于输入层神经元个数，然后再**损失函数**（**loss function**）构造上加上稀疏约束，就变成了**稀疏自编码**（**sparse autoencoder**）模型。稀疏自编码通过约束不为0的神经元个数，来使用尽可能少的神经元表示输入数据 $\mathbf{x}$,从而实现稀疏降维。

1.3 收缩自编码（Contractive Autoencoder）

Rifai et al. 提出类似于稀疏自编码的方法，在损失函数中加入相对于输入维度所有偏导数平方和为正则项：

$$||J_f(\mathbf{x})||_F^2=\sum_{ij}(\frac{\partial h_j(\mathbf{x})}{\partial x_i})^2$$

其中 h_j 代表每个数据 x_i 对应的低维数据 $z_i=g_\phi(x_i)$。
该方法通过减小损失函数对输入层的敏感性，来提高网络对于训练数据小扰动的鲁棒性。

2. 变分自编码器 (Variational Autoencoder)

首先贴上原文: Auto-Encoding Variational Bayes

因为AE及其拓展都没有很好的通过分布来理解和控制对变量的合理识别，故而并不能产生新的东西，即不满足**生成模型**（**generative model**）的需求。VAE的提出，使编码器能通过添加约束的方式，强迫它产生服从标准正态分布的参数，从而理解并控制合理的潜在变量。这样，通过VAE我们就可以从标准正态分布中进行采样并产生我们想要的内容。

> VAE让网络有了控制生成内容的能力

为了让网络能够控制合理潜在变量,VAE需要隐式层的**隐变量**（**latent variable**）$\mathbf{z}$ 趋近于服从标准正态分布，即 $\mathbf{z} \sim \mathcal{N}(0,1)$ 。所以我们为每一个输入样本 x^k 匹配一个对应的正态分布 $\mathcal{N}(\mu_k, \sigma_k^2)$ ，通过拟合 $\mu_k = f_1(x_k)$ 和 $log\sigma^2 = f_2(x_k)$ 这两个函数，即可达到目的。

已知输入数据 $\mathbf{x}$ 以 θ 为参数的概率分布为：

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})d\mathbf{z}$$

为了使网络对于潜在合理参数的控制性尽可能的大，即令可能性 $p_\theta(\mathbf{x})$ 达到最大值，我们需要使关于 $\mathbf{x}$ 的最大似然对数尽可能大：

$$L = \sum_x log\ p_\theta(\mathbf{x})$$

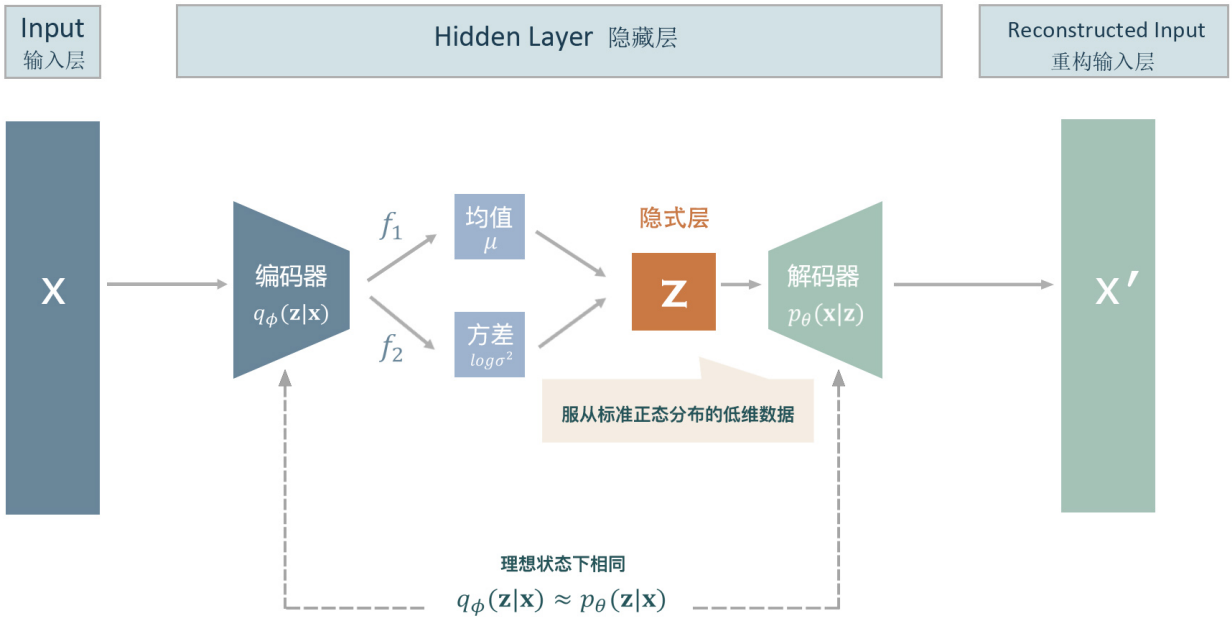
因为后验分布 $p_\theta(\mathbf{z}|\mathbf{x})$ 的求解很困难，所以我们引入$q_\phi(\mathbf{z}|\mathbf{x})$ 来逼近它，然后对 $log\ p_\theta(\mathbf{x}|\mathbf{z})$ 进行变换后可得：

$$\begin{aligned} log\ p_\theta(\mathbf{x}) &= \int q_\phi(\mathbf{x})p_\theta(\mathbf{z})d\mathbf{z} \\ &= \mathbb{E}_{\mathbf{z}\sim\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})} log\ p_\theta(\mathbf{x}|\mathbf{z}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) \\ &= L_b + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) \end{aligned}$$

因为KL散度始终为非负，$L_b = \mathbb{E}_{\mathbf{z}\sim\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})} log\ p_\theta(\mathbf{x}|\mathbf{z}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$即成了 $log\ p_\theta(\mathbf{x})$ 的下界。而因为 $q_\phi(\mathbf{z}|\mathbf{x})$ 与 $log\ p_\theta(\mathbf{x})$ 无关，所以当 $p_\theta(\mathbf{x}|)$ 固定时, KL散度越趋近于0，下界越趋近于 $log\ p_\theta(\mathbf{x})$ ，最大化 $log\ p_\theta(\mathbf{x})$ 等价于最大化 L_b 。

而最大化 L_b 需要最小化 $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$ 同时最大化期望 $\mathbb{E}_{\mathbf{z}\sim\mathbf{q}_\phi(\mathbf{z}|\mathbf{x})} log\ p_\theta(\mathbf{x}|\mathbf{z})$。在同时满足以上三个条件

最后总结VAE结构图如下：



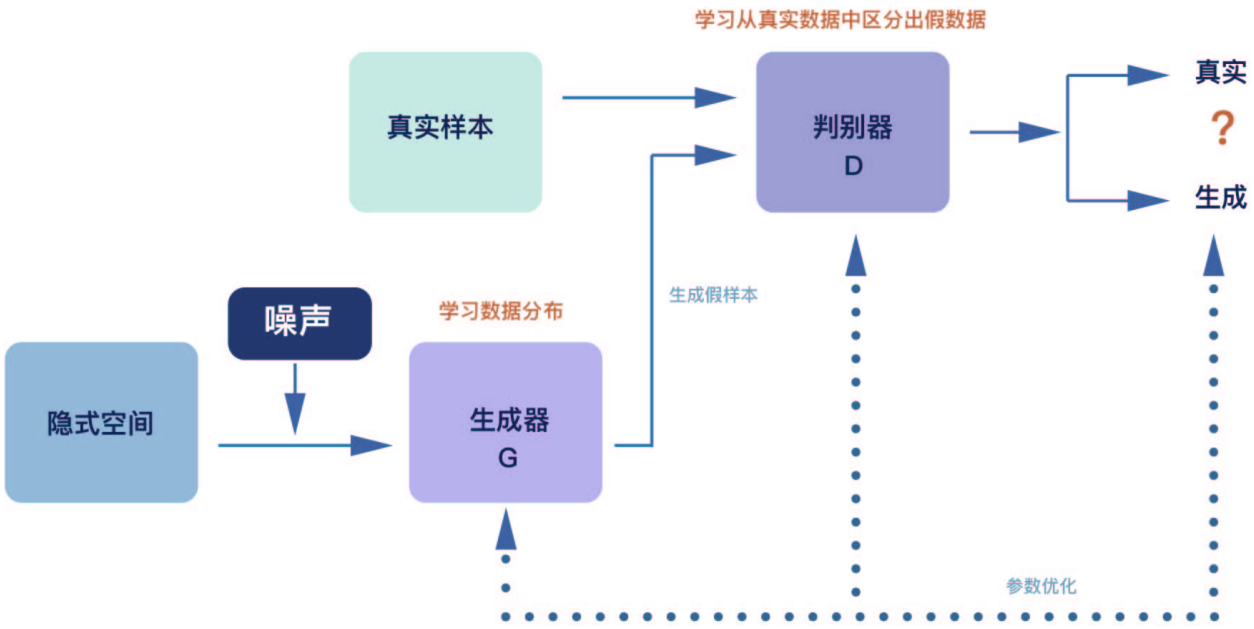
3. 对抗生成网络（GAN）

首先贴上原文: Generative Adversarial Nets

之前的VAE网络的生成内容并不够丰富和真实，而GAN的提出让人工智能产生了更强大的创造力。它主要由两个网络模型组成，一个是**生成模型**（**generative model**），其作用是不断试图产生欺骗判别器的真是样本；另一个是**判别模型**（**discriminator model**），其作用是不断试图分辨出真实模型和生成样本。顾名思义，通过不断地对抗博弈，它的建模能力得到不断地提高，直到可以产生各种人类才能创造出来的产品（目前以图像为主流）。

> GAN能通过自我博弈训练来不断提高生成内容的真实性

其结构图如下：



GAN通过向隐式空间中添加噪声数据来输入生成器，生成器输出的假样本随机被输入到判别器中与真实样本做对比。通过判别器判定该假数据来自于真实数据的可能性值来返回优化生成器和判别器的参数，使得生成器不断提高学习数据分布的能力，同时判别器不断提高其对假数据的分辨能力，如此往复。

定义生成器为 G，判别器为 D，我们做如下假定：

符号	意义
p_z	噪声 z 的数据分布
p_g	生成器关于 x 的数据分布
p_r	真实样本关于 x 的数据分布
$G(z): z \rightarrow x$	噪声 z 到数据 x 的映射
$D(G(z))$	判断数据 x 是否来自于**真实数据**（**ground truth**）的可能性值

首先对于生成器，为了产生与真实数据更相近的数据，需要最小化期望 $\mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$ 来让 D 来对生成数据判断出更大的可能性值。

其次对于判别器，为了保证 D 能准确的区分真实数据与生成数据，需要最大化期望 $\mathbb{E}_{x\sim p_r(x)}[log\ D(x)]$。同时，为了使 D 对生成的数据的判断可能性值 $D(G(z))$ 尽可能的小，又需要最小化期望 $\mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$。

为了同时优化两个网络模型，得到了以下损失方程：

$$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

在给定 G 时，可以通过最大化 $L(G,D)$ 来得到 D 的最优值 D^*：

$$\max_D L(D,G) = \int_x (p_r(x)log(D(x)) + p_g(x)log(1-D(x)))dx$$

$$D^* = \frac{p_r(x)}{p_r(x)+p_g(x)} \in [0,1]$$

因此， $\max_D L(G,D)$ 可以被重写为：

$$\max_D L(D,G) = 2D_{JS}(p_r(x)||p_g(x)) - 2log2$$

当生成器被训练到最佳状态，即 p_r 与 p_g 足够接近时，$D^*(x) = 1/2$ 。此时可以得到 $L(G,D)$ 的全局最优解 $L(G,D^*) = -2\log 2$ 。

4. 扩散模型（Diffusion）

首先贴上原文： Diffusion Models Beat GANs on Image Synthesis 和 Denoising Diffusion Probabilistic Models

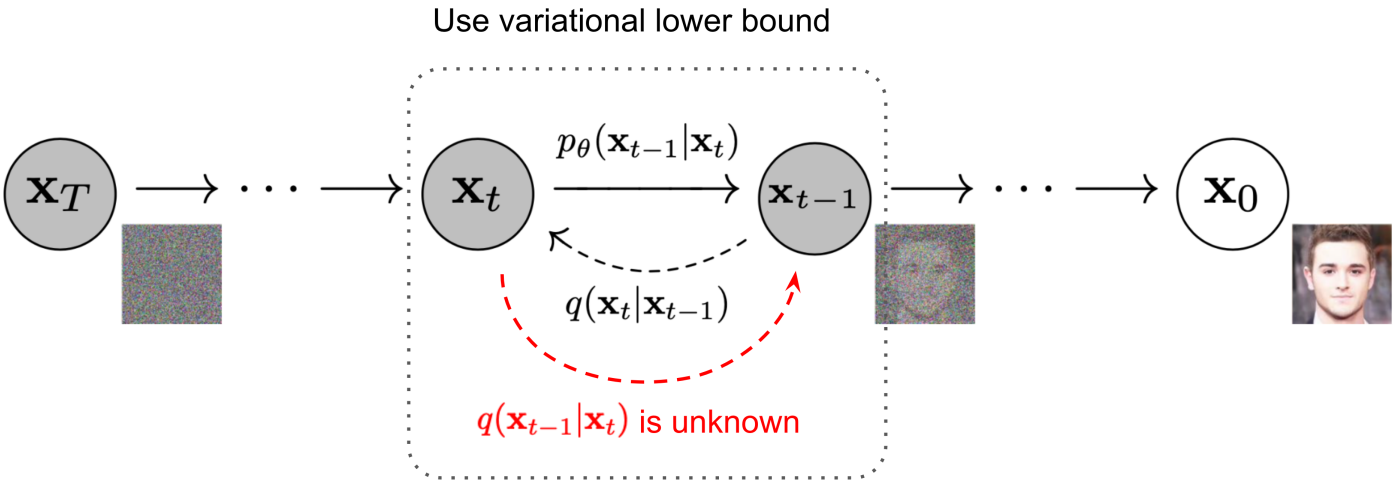
扩散模型（**diffusion model**）作为最近热度较高的生成模型，顾名思义，通过不断添加随机噪声到数据中，来逆向训练网络，从而都检出所需的数据样本，但其学习方式和添加噪声的规则都是固定的。其在多个领域的出色表现，且在图片生成效果中表现尤其优异，超越了GAN。

> Diffusion通过按固定规则添加随机噪声并逆向学习去噪来获得生成内容的能力。

Diffusion可以分为两个步骤：

- **正向扩散**（**forward diffusion process**）：根据定义扩散步骤的马尔科夫链，向数据中逐步添加随机噪声。
- **反向扩散**（**forward diffusion process**）：根据每一步添加的加噪数据与上一步的加噪数据比较来反向学习得出添加的噪声。

以**DDPM**举例说明Diffusion的结构如下：



根据马尔科夫链规则，从右至左/由左至右，逐步正向/反向扩散过程，得到添加/去除噪声后的样本。
每个时刻 t 的生成内容只和 $t-1$ 时刻有关。

（图片来源： DDPM ）

正向扩散

对于给定的一个从真实数据中得到的采样点 $\mathbf{x}_0 \sim q(\mathbf{x})$ ，经过 T 次累加步骤后产生一系列含噪样本 $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T$ ，其中时间步长由一组服从正态分布的方差 $\beta_t \in (0,1)_{t=1}^T$ 来控制。该过程可概括为：

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \qquad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

为了让采样 $\mathbf{x}_t$ 可微，引入重参数技巧把随机性转移到另一个独立的随机变量（ $\epsilon \in \mathcal{N}(0, \mathbf{I})$ ）上，令 $\alpha_t = 1-\beta_t$ 以及 $\bar{\alpha} = \prod_{i=1}^{t} \alpha_i$ 可得：

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$$

其满足：

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

在这个过程中，随着 t 的不断增大，时间步长会变的越来越大，而图片会逐渐变成纯噪声。

反向扩散

反向过程则是通过采样 $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ ，从而能从高斯噪声输入 $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ 中重建出正确的数据样本。当 β_t 足够小时，$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 也服从高斯分布。与VAE类似，因为预测 $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 需要整个数据集，很难显式求解，所以我们选择使用以 θ 作为参数的模型 $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 来拟合它，从而得到每一个时刻的分布 $p_\theta(\mathbf{x}_{0:T})$ ：

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

给定 x_0 ，通过贝叶斯公式可得：

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \\ &\propto \exp\Big(-\frac{1}{2}\big(\underbrace{(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}})\mathbf{x}_{t-1}^2}_{\mathbf{x}_{t-1}\text{的方差}} - \underbrace{(\frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0)\mathbf{x}_{t-1}}_{\mathbf{x}_{t-1}\text{的均值}} + \underbrace{C(\mathbf{x}_t, \mathbf{x}_0)}_{\text{与}\mathbf{x}_{t-1}\text{无关}}\big)\Big) \end{aligned}$$

那么 $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ 服从正态分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t), \tilde{\beta}_t\mathbf{I})$ 。其中：

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$$
$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\Big)$$

其中 $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t)$ 即为模型最终需要训练的 `ground truth`， $\boldsymbol{\epsilon}_t$ 为深度模型所预测的高斯噪声，若写作 $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ 则改写上式得到：

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\Big)$$

由此可以得到模型预测值 $\mathbf{x}_{t-1}$：

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\Big) + \sigma_\theta(\mathbf{x}_t, t)z \qquad z \sim \mathcal{N}(0, \mathbf{I})$$

训练过程

以类似于VAE的变分下界来优化负对数似然值可得：

$$\begin{aligned} L &= \mathbb{E}_{q(\mathbf{x}_0)}[-log\ p_\theta(\mathbf{x_0})] \\ &\leqslant \mathbb{E}_{q(\mathbf{x}_{0:T})}[-log\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}] \qquad\qquad Jensen不等式 \\ &= -L_{VLB} \end{aligned}$$

对 L_{VLB} 进行进一步的推导可得多个KL散度和熵的累加：

$$L_{\text{VLB}} = \mathbb{E}_q[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^{T}\underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}]$$

其中对 L_t 进行多远高斯分布的KL散度求解可得：

$$L_t = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\Big[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\Big]$$

通过最小化 $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ 和 `ground truth` $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)$ 的MSE可进一步简化得到：

$$L_t^{\text{simple}} = \mathbb{E}_{t\sim[1,T], \mathbf{x}0, \boldsymbol{\epsilon}t}\Big[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\Big]$$

最终得到的优化目标损失函数为：

$$L_{simple} = L_t^{\text{simple}} + C$$

其中 C 是与 θ 无关的常数。

该训练过程可被总结为：

1. 获取输入 $\mathbf{x}_0$ ，并从 $[1, T]$ 中随机采样一个 t 。
2. 从标准正态分布随机采样一个噪声 $\bar{z}_t \sim \mathcal{N}(0, \mathbf{I})$ 。
3. 最小化 L_{simple} 。

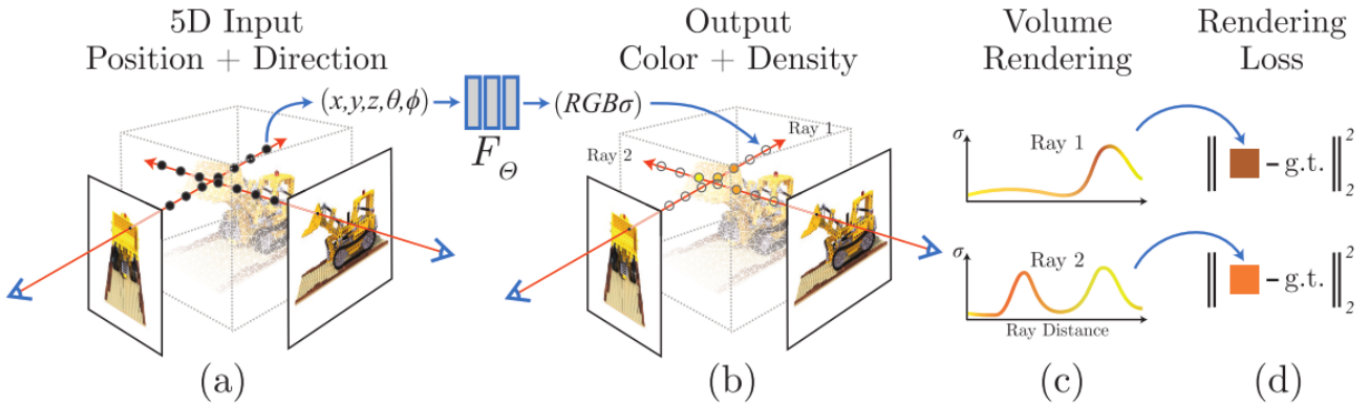
5. 神经辐射场（Neural Radiance Fields）

首先贴上原文：Representing Scenes as Neural Radiance Fields for View Synthesis。

不同于传统的三维重建方式，需要把场景通过点云、网格、体素等方式来进行显式表征，NeRF直接将场景建模成一个可微的5D辐射场隐式存储于神经网络中，只需要输入极少（稀疏，甚至有时只需要两张）不同角度带pose的图像就能训练得到一个NeRF模型，可以通过这个模型渲染出任意角度下清晰的图像。

> NeRF通过极少量图片可重构3D场景并渲染出任一角度的图像。

NeRF的结构可用下图做一个直观的理解：



图(a)通过相机光线采样得到5D坐标（位置+视角方向）； 图(b)把位置输入MLP来生成颜色和体积密度； 图(c)使用立体渲染技术，
利用这些值渲染新的图像；
图(d)利用该渲染函数的可导性来最小化合成图像与 `ground truth` 的损失函数来进行场景优化。

图片来源: Mildenhall et al.

NeRF的表征形式

输入已知的场景中点的位置 $\mathbf{x} = (x, y, z)$ 和二维观察方向 (θ, ϕ) ，MLP神经网络 F_Θ 会输出数据组 $(\mathbf{c}, \sigma(\mathbf{x}))$ 来表示该方向的自发光颜色 $\mathbf{c} = (r, g, b)$ 和该点体素密度 $\sigma(\mathbf{x})$。 然后使用 `classical volume rendering` 渲染方程即可得到光线 $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ 根据近点 t_n 和远点 t_f 所产生的颜色值 $C(\mathbf{r})$:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t)\mathbf{c}(\mathbf{r}(t)), \mathbf{d})dt$$

其中 $T(t) = exp\big(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\big)$ 代表了从近点 t_n 到 点 t 的累积透射比，$\mathbf{d}$ 是三维的观察方向矢量。

通过辐射场渲染体素

上述体素渲染方程式是可导的连续形式，但在实际网络训练过程中只能用离散形式进行近似计算。使用求积法则可以得到 $C(\mathbf{r})$ 的近似估计值 $\hat{C}(\mathbf{r})$:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - exp(-\sigma_i\delta_i))\mathbf{c}(i)$$

其中 $T_i = exp\big(-\sum_{j=1}^{i-1}\sigma_j\delta_j\big)$ ， $\sigma_i = t_{i+1} - t_i$ 表示相邻样本之间的距离，通过将视线路径均分成N段，再对每一段均匀随机采样体素用于渲染计算使得 $t_i \sim \mathcal{U}[t_n + \frac{i-1}{N}(t_f - t_n),\ t_n + \frac{i}{N}(t_f - t_n)]$ 。

NeRF的优化方式

位置编码（positional coding)

因为直接通过 隐式表征+体素渲染 的方式 会导致高频信息丢失，从而致使图像模糊。为了让神经网络模型更好的拟合高频信息，需要将位置向量 $\mathbf{x}$ 和 方向向量 $\mathbf{d}$ 应用从 $\mathbb{R}$ 到 $\mathbb{R}^{2\mathbb{L}}$ 的增维映射 γ :

$$\gamma(p) = (sin(2^0\pi p), cos(2^0\pi p), ..., sin(2^{L-1}\pi p), cos(2^{L-1}\pi p))$$

分级体素抽样 (hierarchical volume sampling)

因为**自由空间**（**free space**）和**遮挡区域**（**occluded region**）这样对于计算颜色值毫无贡献但仍被重复采样的区域存在，拖慢了NeRF的训练效率，NeRF采用了分级表征的方式同时优化两个网络：**粗糙**（**coarse**）和**精细**（**fine**）网络。

先通过分级采样得到 N_c 个点，通过粗糙网络的渲染方程计算得到:

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N} w_i c_i \qquad w_i = T_i(1 - exp(\sigma_i\delta_i))$$

然后对 w_i 进行归一化:

$$\hat{w}_i = \frac{w_i}{\sum_{j=1}^{N_c} w_j}$$

来产生分段常数概率密度函数，然后通过逆变换采样获得 N_f 个点，并添加至 N_c 个点中用于精细网络的渲染。通过二次采样的方式，可以使采样点更多采用对于计算颜色有贡献的体素进行计算。

最后通过同时优化两个网络的最小残差即可得到损失方程:

$$L = \sum_{r\in\mathbb{R}} \left[||\hat{C}_c(\mathbf{r}) - C(\mathbf{r})||_2^2 + ||\hat{C}_f(\mathbf{r}) - C(\mathbf{r})||_2^2\right]$$

最小化该损失方程即可使NeRF达到优化条件。