

什么是Diffusion模型

这是一篇关于 [What are Diffusion Models](#) 的译文。很多地方借鉴了[由浅入深了解Diffusion Model](#)

本文主要是介绍什么是DDPM：Denoising Diffusion Probabilistic Models(DDPM; [Ho et al. 2020](#))。

Basic Knowledge

- Normal Distribution: $\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
- Conditional Probability: $P(A | B) = \frac{P(A \cap B)}{P(B)}$
- Bayes' theorem: $P(A | B) = \frac{P(B | A)P(A)}{P(B)}$
- KL divergence: $D_{\text{KL}}(q(x)||p(x)) = \mathbb{E}_{q(x)} \log[q(x)/p(x)]$

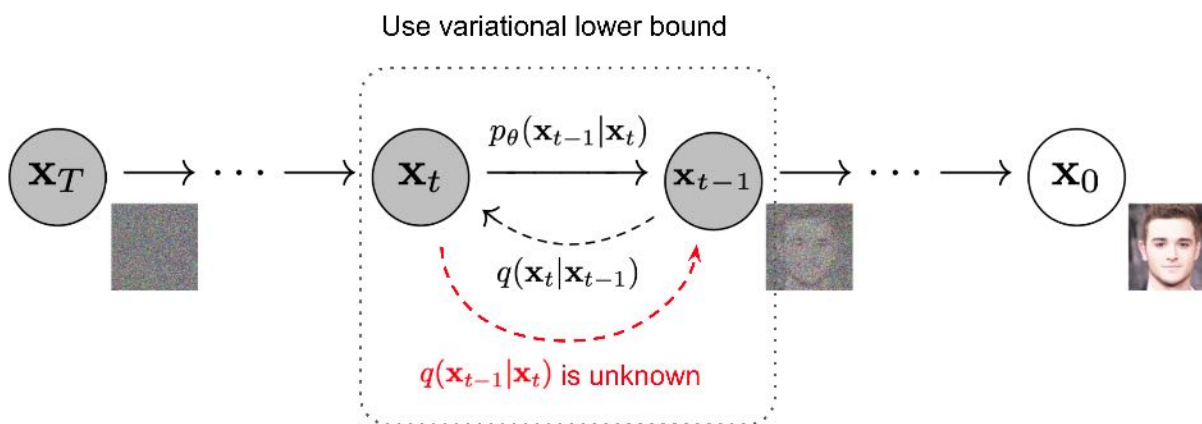
DDPM

Forward diffusion process

从给定的真实数据集中采样一个数据样本： $\mathbf{x}_0 \sim q(\mathbf{x})$ ，我们定义前向扩散过程，在这个过程中，我们逐步向样本中添加少量高斯噪声，产生一系列噪声样本 $\mathbf{x}_1, \dots, \mathbf{x}_T$ 。步长由方差schedule控制 $\{\beta_t \in (0, 1)\}_{t=1}^T$

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

随着步长 t 的变大，数据样本 \mathbf{x}_0 逐渐失去其特征。最终，当 $T \rightarrow \infty$ ， \mathbf{x}_T 等价于各向同性高斯分布。



上述过程有一个很好的特性，我们可以使用重参数化技巧以封闭形式在任意时间步 t 采样 \mathbf{x}_t 。

定义 $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t} \mathbf{x}_{t-1} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_{t-1} && \text{;where } \boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \dots \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
&= \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_t \alpha_{t-1}} \bar{\boldsymbol{\epsilon}}_{t-2} && \text{;where } \bar{\boldsymbol{\epsilon}}_{t-2} \text{ merges two Gaussians (*)}. \\
&= \dots && (*) \text{ 合并两个高斯分布} \\
&= \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon} \\
q(\mathbf{x}_t | \mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}) \\
\mathcal{N}(\mathbf{0}, \sigma_1^2 \mathbf{I}) \text{ 和 } \mathcal{N}(\mathbf{0}, \sigma_2^2 \mathbf{I}), &\text{ 得到的新的高斯分布是 } \mathcal{N}(\mathbf{0}, (\sigma_1^2 + \sigma_2^2) \mathbf{I}) \text{ 合并后的标准差是: } \sqrt{(1 - \alpha_t) + \alpha_t(1 - \alpha_{t-1})} = \sqrt{1 - \alpha_t \alpha_{t-1}}
\end{aligned}$$

Reverse diffusion process

前向过程是扩散（加噪）过程，那么逆向过程就是去噪。如果我们以反转扩散过程，得到逆转后的分布 $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$ ，就可以从标准高斯分布 $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 中还原出原图分布 \mathbf{x}_0 。

如果 β_t 足够小的话， $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$ 也是高斯分布。

不过我们无法简单得到 $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$ 。因此我们使用深度学习模型 p_θ 来预测这样一个逆向分布。

$$\begin{aligned}
p_\theta(\mathbf{x}_{0:T}) &= p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) \\
p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) &= \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))
\end{aligned}$$

我们无法得到逆向后的分布 $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$ ，但如果知道 \mathbf{x}_0 ，是可以通过贝叶斯公式得到 $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ 为：

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\boldsymbol{\beta}}_t \mathbf{I})$$

具体的推导如下：

$$\begin{aligned}
q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) &= q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_0)}{q(\mathbf{x}_t | \mathbf{x}_0)} \\
&\propto \exp \left(-\frac{1}{2} \left(\frac{(\mathbf{x}_t - \sqrt{\alpha_t} \mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}} \mathbf{x}_0)^2}{1 - \bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0)^2}{1 - \bar{\alpha}_t} \right) \right) \\
&= \exp \left(-\frac{1}{2} \left(\frac{\mathbf{x}_t^2 - 2\sqrt{\alpha_t} \mathbf{x}_t \mathbf{x}_{t-1} + \alpha_t \mathbf{x}_{t-1}^2}{\beta_t} + \frac{\mathbf{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}} \mathbf{x}_0 \mathbf{x}_{t-1} + \bar{\alpha}_{t-1} \mathbf{x}_0^2}{1 - \bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0)^2}{1 - \bar{\alpha}_t} \right) \right) \\
&= \exp \left(-\frac{1}{2} \left(\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} \right) \mathbf{x}_{t-1}^2 - \left(\frac{2\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0 \right) \mathbf{x}_{t-1} + C(\mathbf{x}_t, \mathbf{x}_0) \right) \right)
\end{aligned}$$

其中第一步贝叶斯公式部分：

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_0)}{q(\mathbf{x}_t | \mathbf{x}_0)}$$

$$P(A | B, C) = \frac{P(B|A, C) P(A|C)}{P(B|C)}$$

▼ Proof:

$$\begin{aligned}
P(A | B, C) &= \frac{P(A, B, C)}{P(B, C)} \\
&= \frac{P(B | A, C) P(A, C)}{P(B, C)} \\
&= \frac{P(B | A, C) P(A | C) P(C)}{P(B, C)} \\
&= \frac{P(B | A, C) P(A | C) P(C)}{P(B | C) P(C)} \\
&= \frac{P(B | A, C) P(A | C)}{P(B | C)}
\end{aligned}$$

一般的高斯密度函数的指数部分应该写为：

$$\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \exp\left(-\frac{1}{2}\left(\frac{1}{\sigma^2}x^2 - \frac{2\mu}{\sigma^2}x + \frac{\mu^2}{\sigma^2}\right)\right)$$

其中 $C(\mathbf{x}_t, \mathbf{x}_0)$ 是与 \mathbf{x}_{t-1} 无关的常数项。按照标准高斯密度函数，均值和方差可以参数化如下（回想一下 $\alpha_t = 1 - \beta_t$ 和 $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$ ）：

$$\begin{aligned}
\tilde{\beta}_t &= 1 / \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} \right) = 1 / \left(\frac{\alpha_t - \bar{\alpha}_t + \beta_t}{\beta_t(1 - \bar{\alpha}_{t-1})} \right) = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \cdot \beta_t \\
\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) &= \left(\frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0 \right) / \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} \right) \\
&= \left(\frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0 \right) \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \cdot \beta_t \\
&= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0
\end{aligned}$$

因为：

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

可以整理成：

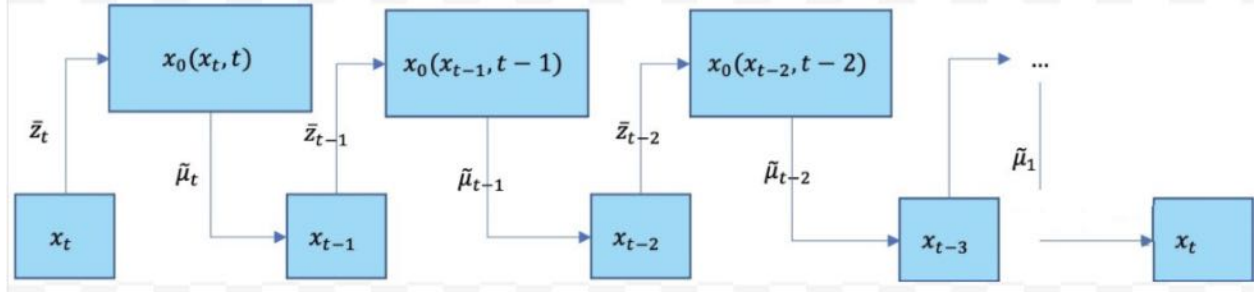
$$\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_t)$$

把它带入上述方程得到：

$$\begin{aligned}
\tilde{\mu}_t &= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_t) \\
&= \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_t \right)
\end{aligned}$$

如此DDPM的每一步推断可以总结为：

1. 每个时间步通过 \mathbf{x}_t 和 t 来预测高斯噪声 $\boldsymbol{\epsilon}_t$ ，然后根据 $\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_t \right)$ 得到 μ_t
2. DDPM中方差部分 $\Sigma_\theta(x_t, t)$ 与 $\tilde{\beta}_t$ 相关
3. 根据 $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$ ，利用重参数得到 x_{t-1} ：



Diffusion训练

如上文所描述的diffusion逆向过程，我们需要训练diffusion模型得到可用的 $\mu_\theta(x_t, t)$ 。

Diffusion的设置与 VAE 非常相似，因此我们可以使用变分下限来优化负对数似然：

$$\begin{aligned}
 \log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + \text{DKL}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)||p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
 &= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T} \sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)} \right] \\
 &= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0) \right] \\
 &= \mathbb{E}_q \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \right] \\
 \text{Let } L_{\text{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \right] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0)
 \end{aligned}$$

进一步对 L_{VLB} 推导，可以得到熵与多个KL散度的累加，推导过程

$$\begin{aligned}
L_{\text{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \right] \\
&= \mathbb{E}_q \left[\log \frac{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \right] \\
&= \mathbb{E}_q \left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \right] \\
&= \mathbb{E}_q \left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)} \right] \\
&= \mathbb{E}_q \left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \left(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} \right) + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)} \right] \\
&= \mathbb{E}_q \left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)} \right] \\
&= \mathbb{E}_q \left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)} \right] \\
&= \mathbb{E}_q \left[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \right] \\
&= \mathbb{E}_q \left[\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} \right]
\end{aligned}$$

▼ Notes:

- basic knowledge about logarithms:

$$\log A + \log B = \log AB$$

$$\log A - \log B = \log A/B$$

$$\log A^n = n \log A$$

- line 4 → line 5: Markov property of the forward process, and the Bayes' rule

$$\begin{aligned}
q(x_t|x_{t-1}) &= q(x_t|x_{t-1}, x_0) \\
P(A | B, C) &= \frac{P(B | A, C) P(A | C)}{P(B | C)}
\end{aligned}$$

也可以写作：

$$L_{\text{VLB}} = L_T + L_{T-1} + \dots + L_0$$

where $L_T = D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))$

$$L_t = D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1$$

$$L_0 = -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)$$

前向 q 没有可学习参数, x_T 是纯噪声, L_T 可以当作是常数项忽略。而 L_t 则可以看做拉近两个高斯分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ 和 $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, 根据多元高斯分布的KL散度求解：

参数化 L_t

我们需要学习一个神经网络来逼近反向扩散过程中的条件概率分布： $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$ 。我们需要训练一个 μ_θ 来预测 $\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_t \right)$ 。因为 \mathbf{x}_t 在训练时是当作输入参数，我们可以重新参数化高斯噪声项，使其根据 timestep t 的输入 x_t 来预测 ϵ_t ：

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right)$$

$$\text{Thus } \mathbf{x}_{t-1} = \mathcal{N}(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right), \Sigma_\theta(\mathbf{x}_t, t))$$

The loss term L_t

$$\begin{aligned} L_t &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2 \|\Sigma_\theta(\mathbf{x}_t, t)\|_2^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2 \|\Sigma_\theta\|_2^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_t \right) - \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) \right\|^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t) \|\Sigma_\theta\|_2^2} \|\epsilon_t - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t) \|\Sigma_\theta\|_2^2} \|\epsilon_t - \epsilon_\theta(\sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \epsilon_t, t)\|^2 \right] \end{aligned}$$



diffusion训练的核心就是取学习高斯噪声 ϵ_t 和 ϵ_θ 之间的MSE

简化 L_t

$$\begin{aligned} L_t^{\text{simple}} &= \mathbb{E}_{t \sim [1, T], \mathbf{x}_0, \epsilon_t} \left[\|\epsilon_t - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right] \\ &= \mathbb{E}_{t \sim [1, T], \mathbf{x}_0, \epsilon_t} \left[\|\epsilon_t - \epsilon_\theta(\sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \epsilon_t, t)\|^2 \right] \end{aligned}$$

最后的目标函数是：

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

训练过程可以归纳为：

1. 从真实数据分布中随机sample一个 x_0 , 从 $1 \dots T$ 随机采样一个 t .
2. 从标准高斯分布采样一个噪声 $\epsilon_t \sim \mathcal{N}(0, I)$
3. 最小化 $\|\epsilon_t - \epsilon_\theta(\sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \epsilon_t, t)\|$

Algorithm 1 Training

```

1: repeat
2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 
3:    $t \sim \text{Uniform}(\{1, \dots, T\})$ 
4:    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:   Take gradient descent step on
        $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)\|^2$ 
6: until converged

```

Algorithm 2 Sampling

```

1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  do
3:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = \mathbf{0}$ 
4:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 

```

小结



综上所述，理解DDPM可以分成3个部分：

1. Forward diffusion process: 前向扩散过程，这部分主要是通过重参数化，得到

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

2. Reverse diffusion process：反向扩散处理（去噪），这部分主要通过利用贝叶斯公式，利用 x_t, x_0 得到 x_{t-1}

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$$

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_t \right); \text{ where } \epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

3. Diffusion训练，这部分首先利用变分下限来优化负对数似然来得到：

$$L_{\text{VLB}} = L_T + L_{T-1} + \dots + L_0$$

where $L_t = D_{\text{KL}}(q(\mathbf{x}_t | \mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_{\theta}(\mathbf{x}_t | \mathbf{x}_{t+1}))$ for $1 \leq t \leq T - 1$

然后利用1, 2中的结论代入到 L_t :

$$\begin{aligned}
L_t &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2 \|\Sigma \theta(\mathbf{x}_t, t)\|_2^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_{\theta}(\mathbf{x}_t, t)\|_2^2 \right] \\
&= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2 \|\Sigma \theta\|_2^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_t \right) - \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) \right\|_2^2 \right] \\
&= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{(1 - \alpha_t)^2}{2 \alpha_t (1 - \bar{\alpha}_t) \|\Sigma \theta\|_2^2} \|\epsilon_t - \epsilon_{\theta}(\mathbf{x}_t, t)\|_2^2 \right] \\
&= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{(1 - \alpha_t)^2}{2 \alpha_t (1 - \bar{\alpha}_t) \|\Sigma \theta\|_2^2} \|\epsilon_t - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_t, t)\|_2^2 \right]
\end{aligned}$$

在训练的时候loss为：

$$\|\epsilon_t - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_t, t)\|_{\text{da}}$$

代码部分

Training

```

def forward(self, x_0):
    """

```

```

Algorithm 1.
"""
t = torch.randint(self.T, size=(x_0.shape[0],), device=x_0.device)
noise = torch.randn_like(x_0)
x_t = (
    extract(self.sqrt_alphas_bar, t, x_0.shape) * x_0
    + extract(self.sqrt_one_minus_alphas_bar, t, x_0.shape) * noise
)
loss = F.mse_loss(self.model(x_t, t), noise, reduction="none")
return loss

```

Sampling

```

class GaussianDiffusionSampler(nn.Module):
    def __init__(self, model, beta_1, beta_T, T):
        super().__init__()

        self.model = model
        self.T = T

        self.register_buffer("betas", torch.linspace(beta_1, beta_T, T).double())
        alphas = 1.0 - self.betas
        alphas_bar = torch.cumprod(alphas, dim=0)
        alphas_bar_prev = F.pad(alphas_bar, [1, 0], value=1)[:T]

        self.register_buffer("coeff1", torch.sqrt(1.0 / alphas))
        self.register_buffer(
            "coeff2", self.coeff1 * (1.0 - alphas) / torch.sqrt(1.0 - alphas_bar)
        )

        self.register_buffer(
            "posterior_var", self.betas * (1.0 - alphas_bar_prev) / (1.0 - alphas_bar)
        )

    def predict_xt_prev_mean_from_eps(self, x_t, t, eps):
        assert x_t.shape == eps.shape
        return (
            extract(self.coeff1, t, x_t.shape) * x_t
            - extract(self.coeff2, t, x_t.shape) * eps
        )

    def p_mean_variance(self, x_t, t):
        # below: only log_variance is used in the KL computations
        var = torch.cat([self.posterior_var[1:2], self.betas[1:]])
        var = extract(var, t, x_t.shape)

        eps = self.model(x_t, t)
        xt_prev_mean = self.predict_xt_prev_mean_from_eps(x_t, t, eps=eps)

        return xt_prev_mean, var

    def forward(self, x_T):
        """
        Algorithm 2.
        """
        x_t = x_T
        for time_step in reversed(range(self.T)):
            print(time_step)
            t = (
                x_t.new_ones(
                    [
                        x_T.shape[0],
                    ],
                    dtype=torch.long,
                )
                * time_step
            )
            mean, var = self.p_mean_variance(x_t=x_t, t=t)
            # no noise when t == 0
            if time_step > 0:
                noise = torch.randn_like(x_t)
            else:
                noise = 0
            x_t = mean + torch.sqrt(var) * noise

```



```
    assert torch.isnan(x_t).int().sum() == 0, "nan in tensor."  
    x_0 = x_t  
    return torch.clip(x_0, -1, 1)
```