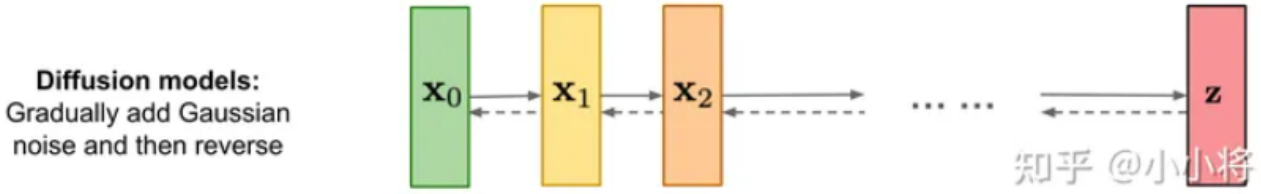
Diffusion model

简单来说，扩散模型包含两个过程：**前向扩散过程**和**反向生成过程**，前向扩散过程是对一张图像逐渐添加高斯噪声直至变成**随机噪声**，而反向生成过程是**去噪声过程**，我们将从一个随机噪声开始逐渐去噪声直至生成一张图像，这也是我们要求解或者训练的部分。



扩散模型原理

扩散模型包括两个过程：**前向过程（forward process）**和**反向过程（reverse process）**，其中前向过程又称为**扩散过程（diffusion process）**，如下图所示。无论是前向过程还是反向过程都是一个**参数化的马尔可夫链（Markov chain）**，其中反向过程可以用来生成数据，这里我们将通过变分推断来进行建模和求解。

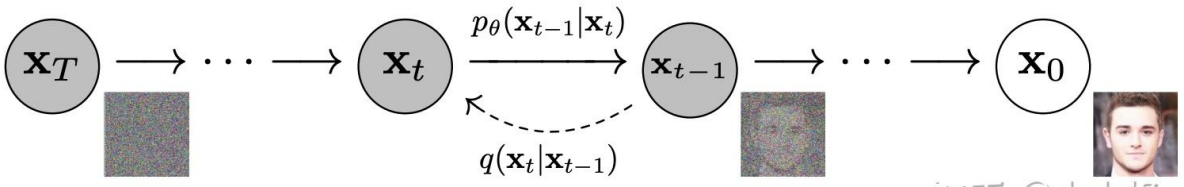


Figure 2: The directed graphical model considered in this work.

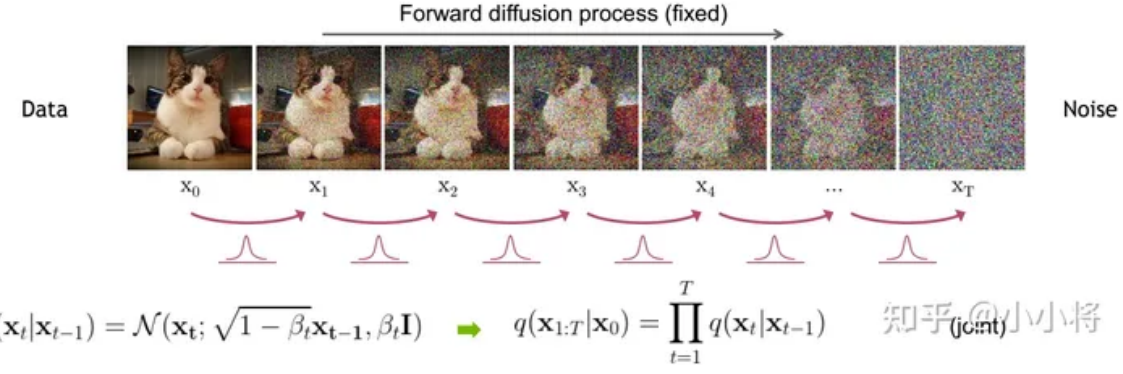
扩散过程

扩散过程是指的对数据逐渐增加高斯噪声直至数据变成随机噪声的过程。对于原始数据 $\mathbf{x}_0 \sim q(\mathbf{x}_0)$，总共包含 T 步的扩散过程的每一步都是对上一步得到的数据 $\mathbf{x}_{t-1}$ 按如下方式增加高斯噪声：

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

这里 $\{\beta_t\}_{t=1}^T$ 为每一步所采用的**方差**，它介于 0～1 之间。对于扩散模型，我们往往称不同 step 的方差设定为 **variance schedule** 或者 **noise schedule**，通常情况下，越后面的 step 会采用更大的方差，即满足 $\beta_1 < \beta_2 < \cdots < \beta_T$ 。在一个设计好的 **variance schedule** 下，的如果扩散步数 T 足够大，那么最终得到的 $\mathbf{x}_T$ 就完全丢失了原始数据而变成了一个随机噪声。 扩散过程的每一步都生成一个带噪声的数据 $\mathbf{x}_t$ ，整个扩散过程也就是一个**马尔卡夫链**：

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$



另外要指出的是，扩散过程往往是固定的，即采用一个预先定义好的 **variance**

schedule，比如 DDPM 就采用一个线性的 variance schedule。

扩散过程的一个重要特性是我们可以**直接基于原始数据 $\mathbf{x}_0$ 来对任意 t 步的 $\mathbf{x}_t$ 进行采样：** $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$ 。这里定义 $\alpha_t = 1 - \beta_t$ ，通过**重参数技巧**（和 VAE 类似），那么有：

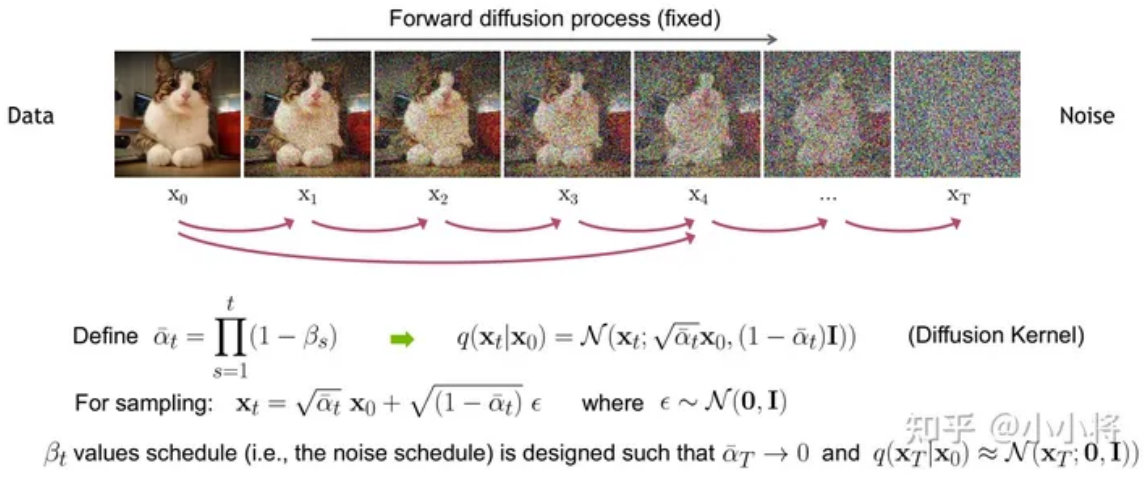
$$
\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} && ;\text{where } \epsilon \\
&= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_{t-2}) + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}^2 + \sqrt{1-\alpha_t}^2}\bar{\epsilon}_{t-2} && ;\text{where } \bar{\epsilon}_{t-} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\epsilon}_{t-2} \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon
\end{aligned}
$$

其中 $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ ，也即，任一时刻的的分布都可以通过 x_0 得到， ϵ_t 是符合高斯分布的噪声。

上述推到过程利用了两个方差不同的高斯分布 $\mathcal{N}(\mathbf{0}, \sigma_1^2\mathbf{I})$ 和 $\mathcal{N}(\mathbf{0}, \sigma_2^2\mathbf{I})$ 相加等于一个新的高斯分布 $\mathcal{N}(\mathbf{0}, (\sigma_1^2 + \sigma_2^2)\mathbf{I})$ 。反重参数化后，我们得到：

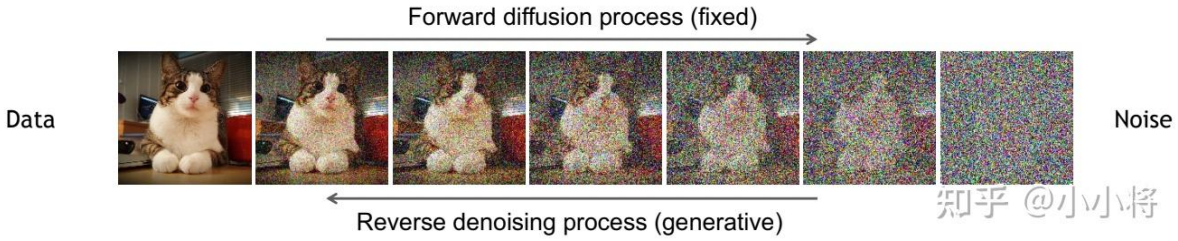
$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

扩散过程的这个特性很重要。首先，我们可以看到 $\mathbf{x}_t$ 其实可以看成是原始数据 $\mathbf{x}_0$ 和随机噪声 ϵ 的线性组合，其中 $\sqrt{\bar{\alpha}_t}$ 和 $\sqrt{1-\bar{\alpha}_t}$ 为组合系数，它们的平方和等于 1，我们也可以称两者分别为 `signal_rate` 和 `noise_rate` 。更近一步地，我们可以基于 $\bar{\alpha}_t$ 而不是 β_t 来定义 **noise schedule**，因为这样处理更直接，比如我们直接将 $\bar{\alpha}_T$ 设定为一个接近 0 的值，那么就可以保证最终得到的 $\mathbf{x}_T$ 近似为一个随机噪声。其次，后面的建模和分析过程将使用这个特性。



反向过程

扩散过程是将数据噪声化，那么反向过程就是**一个去噪的过程**，如果我们知道反向过程的每一步的真实分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$，那么从一个随机噪声 $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 开始，逐渐去噪就能生成一个真实的样本，所以反向过程也就是**生成数据的过程**。



估计分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 需要用到整个训练样本，我们可以用神经网络来估计这些分布。这里，我们将反向过程也定义为一个马尔卡夫链，只不过它是由一系列**用神经网络参数化的高斯分布**来组成：

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

这里 $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$，而 $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 为参数化的高斯分布，它们的均值和方差由训练的网络 $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ 和 $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)$ 给出。实际上，**扩散模型就是要得到这些训练好的网络，因为它们构成了最终的生成模型。**

虽然分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 是不可直接处理的，但是加上条件 $\mathbf{x}_0$ 的后验分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ 却是可处理的，因为先验概率和证据直接求解不到，通过先前推导的公式可以使用 x_0 求解得到。根据先前的式子：

$$q(x_{t-1}|x_0) = \sqrt{\overline{\alpha}_{t-1}}x_0 + \sqrt{1-\overline{\alpha}_{t-1}}\epsilon \sim \mathcal{N}(\sqrt{\overline{\alpha}_{t-1}}x_0, 1-\overline{\alpha}_{t-1}))$$

$$q(x_t|x_0) = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon \sim \mathcal{N}(\sqrt{\overline{\alpha}_t}x_0, 1-\overline{\alpha}_t))$$

$$q(x_t|x_{t-1}, x_0) = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon \sim \mathcal{N}(\sqrt{\alpha_t}x_{t-1}, 1-\alpha_t))$$

这里有：

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

下面我们来具体推导这个分布，首先根据贝叶斯公式，我们有：

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

由于扩散过程的马尔卡夫链特性，我们知道分布 $q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0)=q(\mathbf{x}_t|\mathbf{x}_{t-1})=\mathcal{N}(\mathbf{x}_t;\sqrt{1-\beta_t}\mathbf{x}_{t-1},\beta_t\mathbf{I})$ （这里条件 $\mathbf{x}_0$ 是多余的），而由前面得到的扩散过程特性可知：

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

所以，我们有：

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

这里的 $C(\mathbf{x}_t,\mathbf{x}_0)$ 是一个和 $\mathbf{x}_{t-1}$ 无关的部分，所以省略。

根据高斯分布的概率密度函数定义和上述结果（配平方），我们可以得到后验分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ 的均值和方差：

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

刚刚的推导可以归纳总结为，我们的目的是想在 x_t 的条件下求 x_{t-1} ，发现这个概率符合正态分布，而这个正态分布的均值是和 x_t 和 x_0 相关的。这就麻烦了，我们的目的是最终逐步向前推 x_{t-1} 得到 x_0 ，所以现在并不能得到 x_0 。

然而，我们可以先前的公式指出， x_0 已知可以得到任意的 x_t ，所以我们可以估计出 x_0 ：$x_0=\frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t-\sqrt{1-\bar{\alpha}_t}\epsilon_t\right)$ 将 x_0 带入上面的式子进行替换，得到最终结果： $\tilde{\mu}_t=\frac{1}{\sqrt{a_t}}\left(x_t-\frac{\beta_t}{\sqrt{1-\bar{a}_t}}\epsilon_t\right)$ 至此，均值和方差就都有了，前一个分布（上一张图片）也就可求了。可以看到方差是一个定量（扩散过程参数固

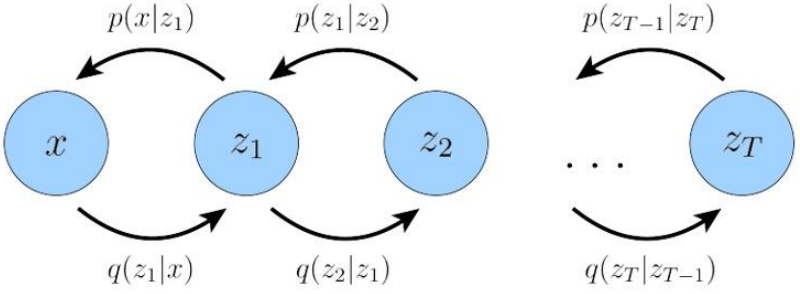
定），而均值是一个依赖 $\mathbf{x}_0$ 和 $\mathbf{x}_t$ 的函数。这个分布将会被用于推导扩散模型的优化目标。然而问题又来了，当下并不知道 ϵ_t 到底是多少。

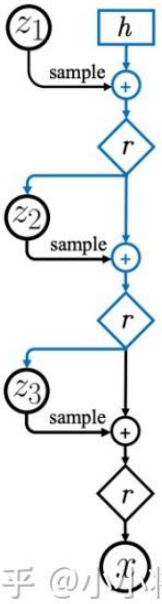
至此数学推导部分基本上已经到了尽头，推导推不出来的活，就交给神经网络来干吧。总的来说，我们的目的，已知 x_t 推导 x_{t-1} 的过程中，需要得到在 t 时刻的噪声 ϵ_t，所以目的便是训练一个网络，使得可以近似求出来一个噪声 $\epsilon_t^* = Z(x_t)$。

训练这个模型需要的东西包括：输入图片（这里是 x_t 已经有了），损失 loss（需要真实标签来进行损失计算）。这时候就巧了，前向过程中不断添加的噪声，不正是所谓的可以用来训练的标签么。

优化目标

上面介绍了扩散模型的扩散过程和反向过程，现在我们来从另外一个角度来看扩散模型：如果我们把中间产生的变量看成隐变量的话，那么扩散模型其实是包含 T 个隐变量的**隐变量模型（latent variable model）**，它可以看成是一个特殊的 **Hierarchical VAEs**。





相比 VAE 来说，扩散模型的隐变量是和原始数据同维度的，而且 encoder（即扩散过程）是固定的。既然扩散模型是隐变量模型，那么我们可以就可以基于**变分推断**来得到 variational lower bound（**VLB**，又称 **ELBO**）作为最大化优化目标，这里有：

$$\log p_\theta(\mathbf{x}_0) = \log \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}$$

$$
\begin{aligned}
&= \log \int \frac{p_\theta(\mathbf{x}_{0:T}) q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} d\mathbf{x}_{1:T} \\
&\geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} [\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}]
\end{aligned}
$$

这里最后一步是利用了 Jensen's inequality（不采用这个不等式的推导见博客 What are Diffusion Models?），对于网络训练来说，其训练目标为 **VLB 取负**：

$$
L = -L_{\text{VLB}} = \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} [-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}] = \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} [\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}]
$$

我们近一步对训练目标进行分解可得：

$$
\begin{aligned}
L &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \Big] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \Big[\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)} \Big] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \Big[-\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)} \Big] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \Big[-\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)} \Big] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)}{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)} \Big] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^{T} \log \Big(\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)} \Big) + \mathrm{l} \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \Big[-\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} \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \Big[-\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)} + \mathrm{l} \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \Big[\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) \Big] \\
&= \mathbb{E}_{q(\mathbf{x}_T|\mathbf{x}_0)} \Big[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} \Big] + \sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{x}_t, \mathbf{x}_{t-1}|\mathbf{x}_0)} \Big[\log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \Big] - \mathbb{E}_q \\
&= \mathbb{E}_{q(\mathbf{x}_T|\mathbf{x}_0)} \Big[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} \Big] + \sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \Big[q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,}{p\theta(\mathbf{x}_{t-1}|\mathbf{x}}
\end{aligned}
$$

$$= \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\Big[D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-}}_{L_{t-1}}$$

可以看到最终的优化目标共包含 T+1 项，其中 L_0 可以看成是原始数据重建，优化的是负对数似然， L_0 可以用估计的 $\mathcal{N}(\mathbf{x}_0; \boldsymbol{\mu}_\theta(\mathbf{x}_1, 1), \boldsymbol{\Sigma}_\theta(\mathbf{x}_1, 1))$ 来构建一个离散化的 decoder 来计算（见 DDPM 论文 3.3 部分）：

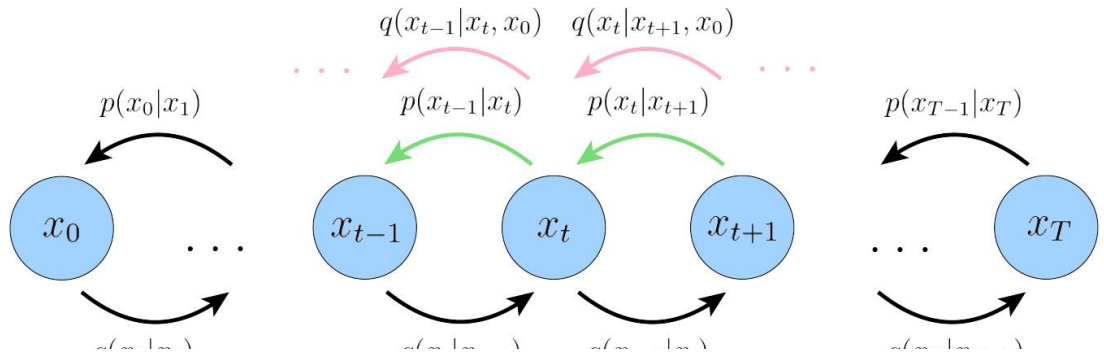
$$p_\theta(\mathbf{x}_0|\mathbf{x}_1) = \prod_{i=1}^{D} \int_{\delta-(x_0^i)}^{\delta+(x_0^i)} \mathcal{N}(x_0; \mu_\theta^i(x_1, 1), \Sigma_\theta^i(x_1, 1))dx$$

$$\delta_+(x) = \begin{cases} \infty & \text{if } x = 1 \\ x + \frac{1}{255} & \text{if } x < 1 \end{cases}$$

$$\delta_+(x) = \begin{cases} -\infty & \text{if } x = -1 \\ x - \frac{1}{255} & \text{if } x > -1 \end{cases}$$

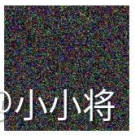
在 DDPM 中，会将原始图像的像素值从[0, 255]范围归一化到[−1, 1]，像素值属于离散化值，这样不同的像素值之间的间隔其实就是 2/255，我们可以计算高斯分布落在以 ground truth 为中心且范围大小为 2/255 时的概率积分即 CDF，具体实现见 https://github.com/hojonathanho/diffusion/blob/master/diffusion_tf/utils.py#L116-L133（不过后面我们的简化版优化目标并不会计算这个对数似然）。

而 L_T 计算的是最后得到的噪声的分布和先验分布的 KL 散度，这个 KL 散度没有训练参数，近似为 0，因为先验 $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ 而扩散过程最后得到的随机噪声 $q(\mathbf{x}_T|\mathbf{x}_0)$ 也近似为 $\mathcal{N}(\mathbf{0}, \mathbf{I})$；而 L_{t-1} 则是计算的是估计分布 $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 和真实后验分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ 的 KL 散度，这里希望我们估计的去噪过程和依赖真实数据的去噪过程近似一致：



$q(x_1|x_0)$ $q(x_t|x_{t-1})$ $q(x_{t+1}|x_t)$ $q(x_T|x_{T-1})$





之所以前面我们将 $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 定义为一个用网络参数化的高斯分布 $\mathcal{N}(\mathbf{x}_{t-1};\boldsymbol{\mu}_\theta(\mathbf{x}_t,t),\boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t))$，是因为要匹配的后验分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ 也是一个高斯分布。对于训练目标 L_0 和 L_{t-1} 来说，都是希望得到训练好的网络 $\boldsymbol{\mu}_\theta(\mathbf{x}_t,t)$ 和 $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t)$（对于 L_0， $t=1$）。DDPM 对 $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 做了近一步简化，**采用固定的方差**：$\boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t)=\sigma_t^2\mathbf{I}$，这里的 σ_t^2 可以设定为 β_t 或者 $\tilde{\beta}_t$（这其实是两个极端，分别是上限和下限，也可以采用可训练的方差，见论文 Improved Denoising Diffusion Probabilistic Models 和 Analytic-DPM: an Analytic Estimate of the Optimal Reverse Variance in Diffusion Probabilistic Models）。这里假定 $\sigma_t^2=\tilde{\beta}t$，那么：

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)=\mathcal{N}(\mathbf{x}_{t-1};\tilde{\boldsymbol{\mu}}(\mathbf{x}_t,\mathbf{x}_0),\sigma_t^2\mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)=\mathcal{N}(\mathbf{x}_{t-1};\boldsymbol{\mu}_\theta(\mathbf{x}_t,t),\sigma_t^2\mathbf{I})$$

对于两个高斯分布的 KL 散度，其计算公式为（具体推导见生成模型之 VAE）：

$$\mathrm{KL}(p_1||p_2)=\tfrac{1}{2}(\mathrm{tr}(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1)+(\boldsymbol{\mu_2}-\boldsymbol{\mu_1})^\top\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu_2}-\boldsymbol{\mu_1})-n+\log\frac{\det(\boldsymbol{\Sigma_2})}{\det(\boldsymbol{\Sigma_1})})$$

那么就有：

$$\begin{aligned}D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))&=D_{\mathrm{KL}}(\mathcal{N}(\mathbf{x}_{t-1};\tilde{\boldsymbol{\mu}}(\mathbf{x}_t,\mathbf{x}_0),\sigma_t^2\mathbf{I})\parallel\mathcal{N}(\mathbf{x}\\&=\frac{1}{2}(n+\frac{1}{\sigma_t^2}|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0)-\boldsymbol{\mu}_\theta(\mathbf{x}_t,t)|^2-\\&=\frac{1}{2\sigma_t^2}|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0)-\boldsymbol{\mu}_\theta(\mathbf{x}_t,t)|^2\end{aligned}$$

那么优化目标 L_{t-1} 即为：

$$L_{t-1}=\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[\frac{1}{2\sigma_t^2}|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0)-\boldsymbol{\mu}_\theta(\mathbf{x}_t,t)|^2\right]$$

从上述公式来看，我们是希望网络学习到的均值 $\boldsymbol{\mu}_\theta(\mathbf{x}_t,t)$ 和后验分布的均值 $\tilde{\boldsymbol{\mu}}(\mathbf{x}_t,\mathbf{x}_0)$ 一致。不过 DDPM 发现预测均值并不是最好的选择。根据前面得到的扩散过程的特性，我们有：

$$\mathbf{x_t}(\mathbf{x_0}, \epsilon) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \quad \text{where } \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

将这个公式带入上述优化目标（注意这里的损失我们加上了对 \mathbf{x}_0 的数学期望），可以得到：

$$\begin{aligned}
L_{t-1} &= \mathbb{E}\mathbf{x}_0\Big(\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\Big[\frac{1}{2\sigma_t^2}|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)|^2\Big]\Big) \\
&= \mathbb{E}_{\mathbf{x}_0, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\Big[\frac{1}{2\sigma_t^2}|\tilde{\boldsymbol{\mu}}_t\Big(\mathbf{x_t}(\mathbf{x_0}, \epsilon), \frac{1}{\sqrt{\bar{\alpha}_t}}\Big(\mathbf{x_t}(\mathbf{x_0}, \epsilon) - \sqrt{1-\bar{\alpha}_t}\epsilon\Big)\Big) - \boldsymbol{\mu} \\
&= \mathbb{E}_{\mathbf{x}_0, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\Big[\frac{1}{2\sigma_t^2}|\Big(\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t(\mathbf{x_0}, \epsilon) + \frac{\sqrt{\bar{\alpha}t-1}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}\big(\mathbf{x_t} \\
&= \mathbb{E}_{\mathbf{x}_0, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\Big[\frac{1}{2\sigma_t^2}|\frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t(\mathbf{x_0}, \epsilon) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\Big) - \boldsymbol{\mu}_\theta(\mathbf{x_t}(\mathbf{x_0}, \epsilon), t)
\end{aligned}$$

近一步地，我们对 $\boldsymbol{\mu}_\theta(\mathbf{x_t}(\mathbf{x_0}, \epsilon), t)$ 也进行重参数化，变成：

$$\boldsymbol{\mu}_\theta(\mathbf{x_t}(\mathbf{x_0}, \epsilon), t) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t(\mathbf{x_0}, \epsilon) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\theta\big(\mathbf{x}_t(\mathbf{x_0}, \epsilon), t\big)\Big)$$

这里的 ϵ_θ 是一个基于神经网络的拟合函数，这意味着我们**由原来的预测均值而换成预测噪声** ϵ 。我们将上述等式带入优化目标，可以得到：

$$\begin{aligned}
L_{t-1} &= \mathbb{E}\mathbf{x}_0, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})\Big[\frac{1}{2\sigma_t^2}|\frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t(\mathbf{x_0}, \epsilon) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\Big) - \boldsymbol{\mu}_\theta(\mathbf{x_t}(\mathbf{x_0} \\
&= \mathbb{E}_{\mathbf{x}_0, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\Big[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}|\epsilon - \epsilon_\theta\big(\mathbf{x}_t(\mathbf{x_0}, \epsilon), t\big)|^2\Big] \\
&= \mathbb{E}_{\mathbf{x}_0, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\Big[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}|\epsilon - \epsilon_\theta\big(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\big)|^2\Big]
\end{aligned}$$

DDPM 近一步对上述目标进行了简化，即去掉了权重系数，变成了：

$$L_{t-1}^{\text{simple}} = \mathbb{E}_{\mathbf{x}_0, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\Big[|\epsilon - \epsilon_\theta\big(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\big)|^2\Big]$$

这里的 t 在[1, T]范围内取值（如前所述，其中取 1 时对应 L_0）。由于去掉了不同 t 的权重系数，所以这个简化的目标其实是 VLB 优化目标进行了 reweight。

虽然扩散模型背后的推导比较复杂，但是我们最终得到的优化目标非常简单，就是让网络预测的噪声和真实的噪声一致。DDPM 的训练过程也非常简单，如下图所示：随机选择一个训练样本–>从 1–T 随机抽样一个 t–>随机产生噪声–

计算当前所产生的带噪声数据（红色框所示）->输入网络预测噪声->计算产生的噪声和预测的噪声的 L2 损失->计算梯度并更新网络。

Algorithm 1 Training

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

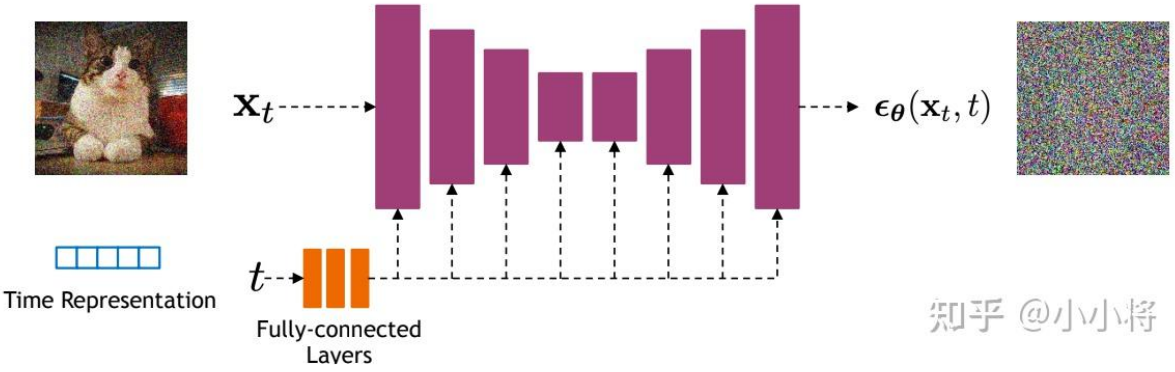
Algorithm 2 Sampling

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

一旦训练完成，其采样过程也非常简单，如上所示：我们从一个随机噪声开始，并用训练好的网络预测噪声，然后计算条件分布的均值（红色框部分），然后用均值加标准差乘以一个随机噪声，直至 t=0 完成新样本的生成（最后一步不加噪声）。不过实际的代码实现和上述过程略有区别（见 https://github.com/hojonathanho/diffusion/issues/5：先基于预测的噪声生成 $\mathbf{x}_0$，并进行了 **clip 处理**（范围[-1, 1]，原始数据归一化到这个范围），然后再计算均值。我个人的理解这应该算是一种约束，既然模型预测的是噪声，那么我们也希望用预测噪声重构处理的原始数据也应该满足范围要求。

模型设计

前面我们介绍了扩散模型的原理以及优化目标，那么扩散模型的核心就在于训练噪声预测模型，由于噪声和原始数据是同维度的，所以我们可以选择采用 **AutoEncoder 架构**来作为噪声预测模型。DDPM 所采用的模型是一个基于 residual block 和 attention block 的 **U-Net 模型**。如下所示：



U-Net 属于 encoder-decoder 架构，其中 encoder 分成不同的 stages，每个 stage 都包含下采样模块来降低特征的空间大小（H 和 W），然后 decoder 和 encoder 相反，是将 encoder 压缩的特征逐渐恢复。U-Net 在 decoder 模块中还引入了 **skip connection**，即 concat 了 encoder 中间得到的同维度特征，这有利于网络优化。DDPM 所采用的 U-Net 每个 stage 包含 **2 个 residual block**，而且部分 stage 还加入了 **self-attention 模块**增加网络的全局建模能力。 另外，扩散模型其实需要的是 T 个噪声预测模型，实际处理时，我们可以增加一个 **time embedding**（类似 transformer 中的 position embedding）来将 timestep 编码到网络中，从而只需要训练一个共享的 U-Net 模型。具体地，DDPM 在各个 residual block 都引入了 **time embedding**，如上图所示。