Diffusion Model + Text Generation

Paper Reading : RenZHi Wang
2022.11.22

Catalogue

1、**Diffusion Model**(Score Based Model+Langevin Dynamics+SDE)

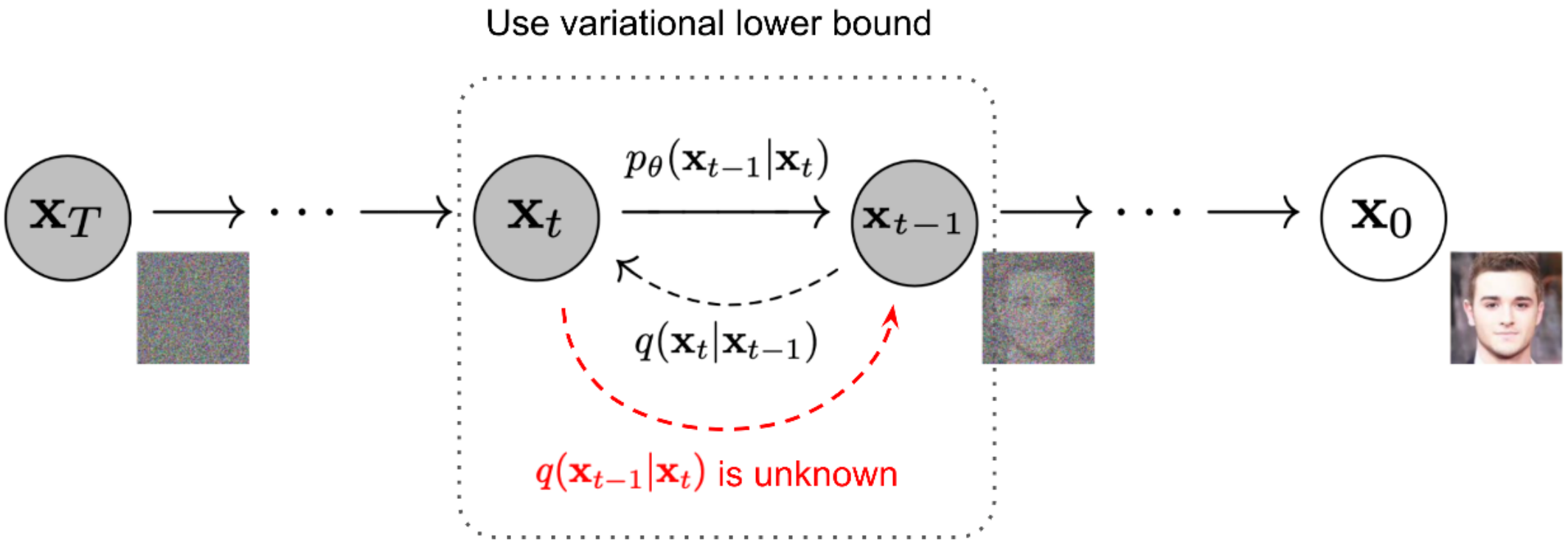
- DDPM
- DDIM
- Stable-Diffusion

2、**Text Generation**

- D3PM
- Diffusion-LM
- DiffuSeq
- SSD-LM
- DiffuER

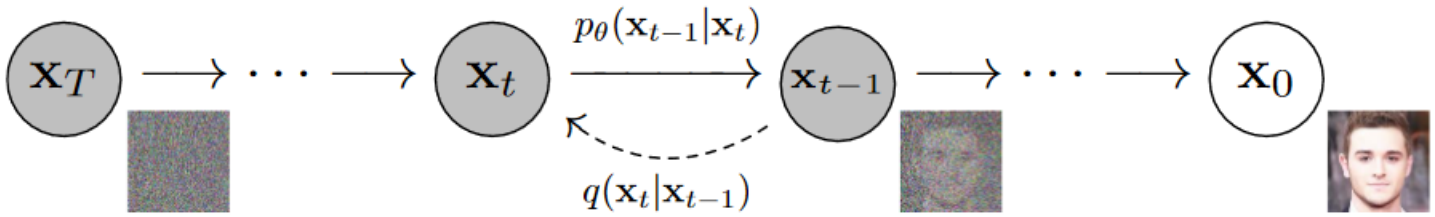
DDPM

《Denoising Diffusion Probabilistic Models》



DDPM

Forward process



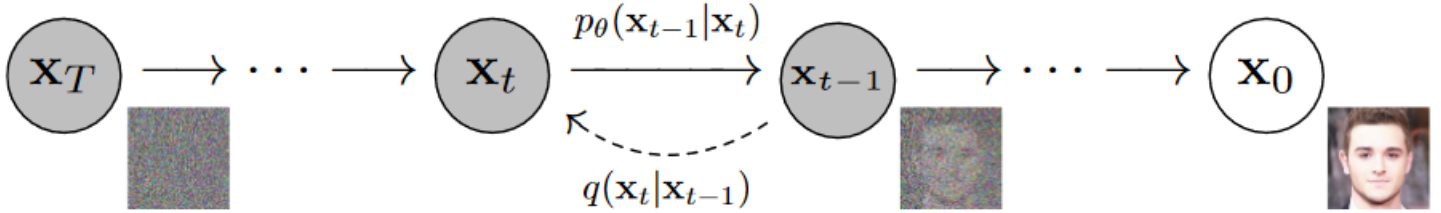
Given a data point sampled from a real data distribution $\mathbf{x}_0 \sim q(\mathbf{x})$, let us define a *forward diffusion process* in which we add small amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples $\mathbf{x}_1, \dots, \mathbf{x}_T$. The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^T$.

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

DDPM

Forward process

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



前向过程有一个很好的特性是我们不需要迭代计算 x_t，任意时刻的 x_t都可以利用 x_0和 β进行直接表示：

首先假设 $\alpha_t = 1 - \beta_t$， $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_t$，采用重参数法：

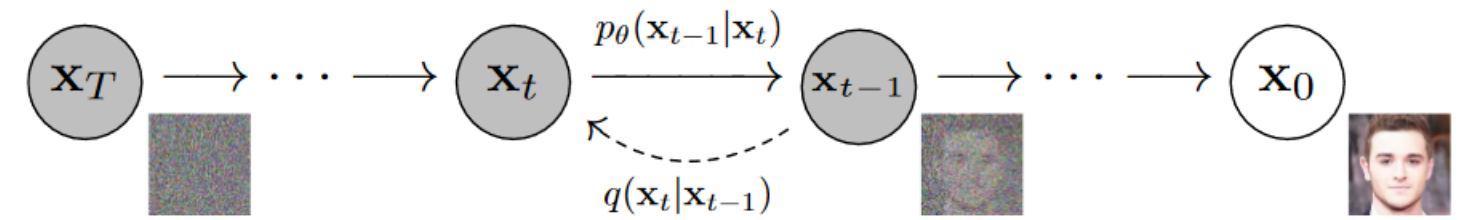
$$\begin{aligned}
x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_1 \quad \text{where } \epsilon_1, \epsilon_2 \sim \mathcal{N}(0, \mathbf{I}) \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_2\right) + \sqrt{1-\alpha_t}\epsilon_1 \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \left(\sqrt{\alpha_t\left(1-\alpha_{t-1}\right)}\epsilon_2 + \sqrt{1-\alpha_t}\epsilon_1\right) \qquad \text{(第三行)} \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\epsilon}_2 \quad \text{where } \bar{\epsilon}_2 \sim \mathcal{N}(0, \mathbf{I}) \qquad \text{(第四行)} \\
&= \cdots \\
&= \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}
\end{aligned}$$

由此，对于任意时刻 x_t ，我们有 $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$

> 这里第三行到第四行是由于对于两个高斯分布 $X \sim \mathcal{N}(\mu_1, \sigma_1^2)$ 和 $Y \sim \mathcal{N}(\mu_2, \sigma_2^2)$，$aX + bY$ 服从 $\mathcal{N}(\mu, \sigma^2)$，其中 $\mu = a\mu_1 + b\mu_2$，$\sigma^2 = a^2\sigma_1^2 + b^2\sigma_2^2$

DDPM

Reverse process



$$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$$

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

It is noteworthy that the reverse conditional probability is tractable when conditioned on $\mathbf{x}_0$:

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

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

Using Bayes' rule, we have:

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

(前向过程已算出)

$$\propto \exp\left(-\frac{1}{2}\left(\frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{1-\bar{\alpha}_t}\right)\right)$$

$$= \exp\left(-\frac{1}{2}\left(\frac{\mathbf{x}_t^2 - 2\sqrt{\alpha_t}\mathbf{x}_t\mathbf{x}_{t-1} + \alpha_t\mathbf{x}_{t-1}^2}{\beta_t} + \frac{\mathbf{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\mathbf{x}_{t-1} + \bar{\alpha}_{t-1}\mathbf{x}_0^2}{1-\bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{1-\bar{\alpha}_t}\right)\right)$$

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

DDPM

Reverse process

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

where $C(\mathbf{x}_t, \mathbf{x}_0)$ is some function not involving $\mathbf{x}_{t-1}$ and details are omitted. Following the standard Gaussian density function, the mean and variance can be parameterized as follows (recall that $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$):

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

=> $q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t\mathbf{I})$

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

DDPM

$$q\left(x_{t-1} \mid x_t, x_0\right)=\mathcal{N}\left(x_{t-1} ; \tilde{\mu}_t\left(x_t, x_0\right), \tilde{\beta}_t \mathbf{I}\right)$$

$$\tilde{\mu}_t\left(x_t, x_0\right)=\frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t} x_t+\frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1-\bar{\alpha}_t} x_0$$

$$\tilde{\beta}_t=\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \beta_t$$

$$x_t=\sqrt{\bar{\alpha}_t} x_0+\sqrt{1-\bar{\alpha}_t} \bar{\epsilon}$$

Thanks to the nice property, we can represent $\mathbf{x}_0=\frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t-\sqrt{1-\bar{\alpha}_t} \boldsymbol{\epsilon}_t\right)$ and plug it into the above equation and obtain:

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

Diffusion Optimization

As demonstrated in Fig. 2., such a setup is very similar to VAE and thus we can use the variational lower bound to optimize the negative log-likelihood.

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

It is also straightforward to get the same result using Jensen's inequality. Say we want to minimize the cross entropy as the learning objective,

$$
\begin{aligned}
L_{\mathrm{CE}} &= -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0) \\
&= -\mathbb{E}_{q(\mathbf{x}_0)} \log \Big(\int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}\Big) \\
&= -\mathbb{E}_{q(\mathbf{x}_0)} \log \Big(\int q(\mathbf{x}_{1:T}|\mathbf{x}_0) \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} d\mathbf{x}_{1:T}\Big) \\
&= -\mathbb{E}_{q(\mathbf{x}_0)} \log \Big(\mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big) \\
&\leq -\mathbb{E}_{q(\mathbf{x}_{0:T})} \log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \\
&= \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] = L_{\mathrm{VLB}}
\end{aligned}
$$

Diffusion Optimization

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

$$\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] && \text{联合概率拆分} \\
&= \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] && \text{Log把累乘转换为累加} \\
&= \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] && \text{第一项单独取出} \\
&= \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] && \text{贝叶斯公式} \\
&= \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] && \text{累成相消} \\
&= \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[\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}]
\end{aligned}$$

Diffusion Optimization

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

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

补全（*）为:

$$(*) = \mathbb{E}_{q(x_{0:T})}\left[\log \frac{q\left(\mathbf{x}_T \mid \mathbf{x}_0\right)}{p_\theta\left(\mathbf{x}_T\right)} + \sum_{t=2}^{T} \log \frac{q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right)}{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)} - \log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)\right]$$

$$= \mathbb{E}_{q(x_0)}[\underbrace{D_{\text{KL}}\left(q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_T\right)\right)}_{L_T}] + \sum_{t=2}^{T} \mathbb{E}_{q(x_t, x_0)}[\underbrace{D_{\text{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)}_{L_{t-1}}] - \mathbb{E}_{q(x_1, x_0)}[\underbrace{\log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}_{L_0}]$$

Let's label each component in the variational lower bound loss separately:

$$\begin{aligned} L_{\text{VLB}} &= L_T + L_{T-1} + \cdots + L_0 \\ \text{where } L_T &= D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T)) \\ L_t &= D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \le t \le T-1 \\ L_0 &= -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \end{aligned}$$

Diffusion Optimization

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

Recall that we need to learn a neural network to approximate the conditioned probability distributions in the reverse diffusion process, $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$. We would like to train $\boldsymbol{\mu}_\theta$ to predict $\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)$. Because $\mathbf{x}_t$ is available as input at training time, we can reparameterize the Gaussian noise term instead to make it predict $\boldsymbol{\epsilon}_t$ from the input $\mathbf{x}_t$ at time step t:

$$q\left(x_{t-1} \mid x_t, x_0\right) = \mathcal{N}\left(x_{t-1}; \tilde{\mu}_t\left(x_t, x_0\right), \tilde{\beta}_t \mathbf{I}\right)$$

$$\tilde{\mu}_t\left(\mathbf{x}_t, t\right) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t\left(\mathbf{x}_t, t\right)\right)$$

$$\tilde{\mu}_t\left(x_t, x_0\right) = \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0$$

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

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

The loss term L_t is parameterized to minimize the difference from $\tilde{\boldsymbol{\mu}}$:

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

$$D_{KL}\left(\mathcal{N}\left(\mu_1, \Sigma_1\right) \| \mathcal{N}\left(\mu_2, \Sigma_2\right)\right) = \log\frac{\left|\Sigma_2\right|}{\left|\Sigma_1\right|} - K + \operatorname{Tr}\left(\Sigma_2^{-1}\Sigma_1\right) + \left(\mu_1 - \mu_2\right)^\top \Sigma_2^{-1}\left(\mu_1 - \mu_2\right)$$

Diffusion Optimization

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

Simplification

Empirically, Ho et al. (2020) found that training the diffusion model works better with a simplified objective that ignores the weighting term:

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

The final simple objective is:

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

where C is a constant not depending on θ.

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

Improve：

《Improved Denoising Diffusion Probabilistic Models》

- 方差Σ变成可学习的 $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \exp(\mathbf{v} \log \beta_t + (1-\mathbf{v}) \log \tilde{\beta}_t)$ (model predicting a mixing vector v)
- β由线性变化变成非线性变换 $\beta_t = \text{clip}(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, 0.999) \quad \bar{\alpha}_t = \frac{f(t)}{f(0)} \quad \text{where } f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)$
- 改变损失函数 $L_{\text{hybrid}} = L_{\text{simple}} + \lambda L_{\text{VLB}}$

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

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

Model	ImageNet	CIFAR
Glow (Kingma & Dhariwal, 2018)	3.81	3.35
Flow++ (Ho et al., 2019)	3.69	3.08
PixelCNN (van den Oord et al., 2016c)	3.57	3.14
SPN (Menick & Kalchbrenner, 2018)	3.52	-
NVAE (Vahdat & Kautz, 2020)	-	2.91
Very Deep VAE (Child, 2020)	3.52	2.87
PixelSNAIL (Chen et al., 2018)	3.52	2.85
Image Transformer (Parmar et al., 2018)	3.48	2.90
Sparse Transformer (Child et al., 2019)	3.44	**2.80**
Routing Transformer (Roy et al., 2020)	**3.43**	-
DDPM (Ho et al., 2020)	3.77	3.70
DDPM (cont flow) (Song et al., 2020b)	-	2.99
Improved DDPM (ours)	**3.53**	**2.94**

Table 3. Comparison of DDPMs to other likelihood-based models on CIFAR-10 and Unconditional ImageNet 64 × 64. NLL is reported in bits/dim. On ImageNet 64 × 64, our model is competitive with the best convolutional models, but is worse than fully transformer-based architectures.

Score based model

Suppose our dataset consists of i.i.d. samples $\{\mathbf{x}_i \in \mathbb{R}^D\}_{i=1}^N$ from an unknown data distribution $p_{\text{data}}(\mathbf{x})$. We define the *score* of a probability density $p(\mathbf{x})$ to be $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. The *score network* $\mathbf{s}_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{R}^D$ is a neural network parameterized by $\boldsymbol{\theta}$, which will be trained to approximate the score of $p_{\text{data}}(\mathbf{x})$. The goal of generative modeling is to use the dataset to learn a model for generating new samples from $p_{\text{data}}(\mathbf{x})$. The framework of score-based generative modeling has two ingredients: score matching and Langevin dynamics.

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log q(\mathbf{x})$$

Score based model

Score matching：

Objective function：就是两个score function（Data pdf的score function 和Model pdf的score function）的MSE（均方误差），又称为Fisher Divergence：

$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}}\left[\left\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})-\nabla_{\mathbf{x}}\log p_{\text{data}}(\mathbf{x})\right\|_2^2\right]$$

改写为：

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\operatorname{tr}(\nabla_{\mathbf{x}}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}))+\frac{1}{2}\left\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})\right\|_2^2\right]$$

变换后的公式不需要显式计算P_{data}（x），但损失函数里面包含有梯度的计算。

$$\begin{aligned}
&\frac{1}{2}E_{p_{\text{data}}}\left[\left\|s_\theta(x)-\nabla_x\log p_{\text{data}}(x)\right\|_2^2\right]\\
=&\frac{1}{2}E_{p_{\text{data}}}\left[\left\|s_\theta(x)\right\|_2^2+(\nabla_x\log p_{\text{data}}(x))(\nabla_x\log p_{\text{data}}(x))^T-2s_\theta(x)\cdot(\nabla_x\log p_{\text{data}}(x))^T\right]\\
=&\frac{1}{2}E_{p_{\text{data}}}\left[\left\|s_\theta(x)\right\|_2^2\right]+0-E_{p_{\text{data}}}\left[s_\theta(x)\cdot\frac{\nabla_x p_{\text{data}}(x)}{p_{\text{data}}(x)}\right]\\
=&\frac{1}{2}E_{p_{\text{data}}}\left[\left\|s_\theta(x)\right\|_2^2\right]-\int s_\theta(x)\cdot\nabla_x p_{\text{data}}(x)dx\\
=&E_{p_{\text{data}}}\left[\frac{1}{2}\left\|s_\theta(x)\right\|_2^2\right]-\left[s_\theta(x)\cdot p_{\text{data}}(x)\right|_{x\to-\infty}^{x\to+\infty}-\int p_{\text{data}}(x)ds_\theta(x)]\\
=&E_{p_{\text{data}}}\left[\frac{1}{2}\left\|s_\theta(x)\right\|_2^2\right]+E_{p_{\text{data}}}\left[\operatorname{tr}(\nabla_x s_\theta(x))\right]
\end{aligned}$$

核心思想：先用一种特定的噪声分布来扰动原始数据$p_{data}(x)$，我们令扰动后的分布为$q_\sigma(\hat{x}|x)$，用score matching的方法去估计扰动后的数据分布的score。

$$q_\sigma(\hat{x}) \triangleq \int q_\sigma(\hat{x}|x)p_{data}(x)dx$$

当噪声十分小的时候，有近似:$q_\sigma(x) \approx p_{data}(x)$。

我们的目标变为：

$$E_{q_\sigma(\hat{x})}[||s_\theta(\hat{x}) - \nabla_{\hat{x}} \log q_\sigma(\hat{x})||_2^2]$$

可以经过推导转换为：

$$E_{q_\sigma(\hat{x}|x)p_{data}(x)}[||s_\theta(\hat{x}) - \nabla_{\hat{x}} \log q_\sigma(\hat{x}|x)||_2^2]$$

从最终结论可以看出：我们添加噪声的方式不影响我们的目标。为了简单计算，可以令添加噪声的类型为线性高斯噪声。

Training Objective:

$$E_{p_{data}(x)} E_{\hat{x}\sim\mathcal{N}(x,\sigma^2 I)}[||s_\theta(\hat{x}) - \frac{x-\hat{x}}{\sigma^2}||_2^2]$$

2.2 Sampling with Langevin dynamics

Langevin dynamics can produce samples from a probability density $p(\mathbf{x})$ using only the score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. Given a fixed step size $\epsilon > 0$, and an initial value $\tilde{\mathbf{x}}_0 \sim \pi(\mathbf{x})$ with π being a prior distribution, the Langevin method recursively computes the following

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\epsilon}{2} \nabla_{\mathbf{x}} \log p(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\epsilon}\, \mathbf{z}_t, \tag{4}$$

where $\mathbf{z}_t \sim \mathcal{N}(0, I)$. The distribution of $\tilde{\mathbf{x}}_T$ equals $p(\mathbf{x})$ when $\epsilon \to 0$ and $T \to \infty$, in which case $\tilde{\mathbf{x}}_T$ becomes an exact sample from $p(\mathbf{x})$ under some regularity conditions [62].

用Score based model刻画DDPM

The schedule of increasing noise levels resembles the forward diffusion process. If we use the diffusion process annotation, the score approximates $\mathbf{s}_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)$. Given a Gaussian distribution $\mathbf{x} \sim \mathcal{N}(\mu, \sigma^2\mathbf{I})$, we can write the derivative of the logarithm of its density function as $\nabla_{\mathbf{x}} \log p(\mathbf{x}) = \nabla_{\mathbf{x}}\Big(-\frac{1}{2\sigma^2}(\mathbf{x}-\boldsymbol{\mu})^2 \Big) = -\frac{\mathbf{x}-\boldsymbol{\mu}}{\sigma^2} = -\frac{\boldsymbol{\epsilon}}{\sigma}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Recall that $q(\mathbf{x}_t|\mathbf{x}_0) \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$ and therefore,

$$\mathbf{s}_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) = \mathbb{E}_{q(\mathbf{x}_0)}[\nabla_{\mathbf{x}_t} q(\mathbf{x}_t|\mathbf{x}_0)] = \mathbb{E}_{q(\mathbf{x}_0)}\Big[-\frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\sqrt{1-\bar{\alpha}_t}} \Big] = -\frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\sqrt{1-\bar{\alpha}_t}}$$

《Score-Based Generative Modeling through Stochastic Differential Equations》

更完备的描述:

$$\tilde{\mu}_t\left(\mathbf{x}_t, t\right)=\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t-\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_t\left(\mathbf{x}_t, t\right)\right)$$

$$\mathbf{s}_\theta(\mathbf{x}_t, t) \approx -\frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\sqrt{1-\bar{\alpha}_t}}$$

Sohl-Dickstein et al. (2015); Ho et al. (2020) consider a sequence of positive noise scales $0 < \beta_1, \beta_2, \cdots, \beta_N < 1$. For each training data point $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$, a discrete Markov chain $\{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_N\}$ is constructed such that $p(\mathbf{x}_i \mid \mathbf{x}_{i-1}) = \mathcal{N}(\mathbf{x}_i; \sqrt{1-\beta_i}\mathbf{x}_{i-1}, \beta_i\mathbf{I})$, and therefore $p_{\alpha_i}(\mathbf{x}_i \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_i; \sqrt{\alpha_i}\mathbf{x}_0, (1-\alpha_i)\mathbf{I})$, where $\alpha_i := \prod_{j=1}^{i}(1-\beta_j)$. Similar to SMLD, we can denote the perturbed data distribution as $p_{\alpha_i}(\tilde{\mathbf{x}}) := \int p_{\text{data}}(\mathbf{x}) p_{\alpha_i}(\tilde{\mathbf{x}} \mid \mathbf{x}) \mathrm{d}\mathbf{x}$. The noise scales are prescribed such that $\mathbf{x}_N$ is approximately distributed according to $\mathcal{N}(\mathbf{0}, \mathbf{I})$. A variational Markov chain in the reverse direction is parameterized with $p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i) = \mathcal{N}(\mathbf{x}_{i-1}; \frac{1}{\sqrt{1-\beta_i}}(\mathbf{x}_i + \beta_i \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_i, i)), \beta_i\mathbf{I})$, and trained with a re-weighted variant of the evidence lower bound (ELBO):

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N}(1-\alpha_i)\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{p_{\alpha_i}(\tilde{\mathbf{x}}|\mathbf{x})}[\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, i) - \nabla_{\tilde{\mathbf{x}}} \log p_{\alpha_i}(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2]. \tag{3}$$

After solving Eq. (3) to get the optimal model $\mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}, i)$, samples can be generated by starting from $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and following the estimated reverse Markov chain as below

$$\mathbf{x}_{i-1} = \frac{1}{\sqrt{1-\beta_i}}(\mathbf{x}_i + \beta_i \mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}_i, i)) + \sqrt{\beta_i}\mathbf{z}_i, \quad i = N, N-1, \cdots, 1. \tag{4}$$

We call this method *ancestral sampling*, since it amounts to performing ancestral sampling from the graphical model $\prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1} \mid \mathbf{x}_i)$. The objective Eq. (3) described here is L_{simple} in Ho et al. (2020), written in a form to expose more similarity to Eq. (1). Like Eq. (1), Eq. (3) is also a weighted sum of denoising score matching objectives, which implies that the optimal model, $\mathbf{s}_{\boldsymbol{\theta}*}(\tilde{\mathbf{x}}, i)$, matches the score of the perturbed data distribution, $\nabla_{\mathbf{x}} \log p_{\alpha_i}(\mathbf{x})$. Notably, the weights of the i-th summand in Eq. (1) and Eq. (3), namely σ_i^2 and $(1-\alpha_i)$, are related to corresponding perturbation kernels in the same functional form: $\sigma_i^2 \propto 1/\mathbb{E}[\|\nabla_{\mathbf{x}} \log p_{\sigma_i}(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2]$ and $(1-\alpha_i) \propto 1/\mathbb{E}[\|\nabla_{\mathbf{x}} \log p_{\alpha_i}(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2]$.

—— 《Score-Based Generative Modeling through Stochastic Differential Equations》

3.1 PERTURBING DATA WITH SDES

Our goal is to construct a diffusion process $\{\mathbf{x}(t)\}_{t=0}^{T}$ indexed by a continuous time variable $t \in [0, T]$, such that $\mathbf{x}(0) \sim p_0$, for which we have a dataset of i.i.d. samples, and $\mathbf{x}(T) \sim p_T$, for which we have a tractable form to generate samples efficiently. In other words, p_0 is the data distribution and p_T is the prior distribution. This diffusion process can be modeled as the solution to an Itô SDE:

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}, \tag{5}$$

where $\mathbf{w}$ is the standard Wiener process (a.k.a., Brownian motion), $\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ is a vector-valued function called the *drift* coefficient of $\mathbf{x}(t)$, and $g(\cdot) : \mathbb{R} \to \mathbb{R}$ is a scalar function known as the *diffusion* coefficient of $\mathbf{x}(t)$.

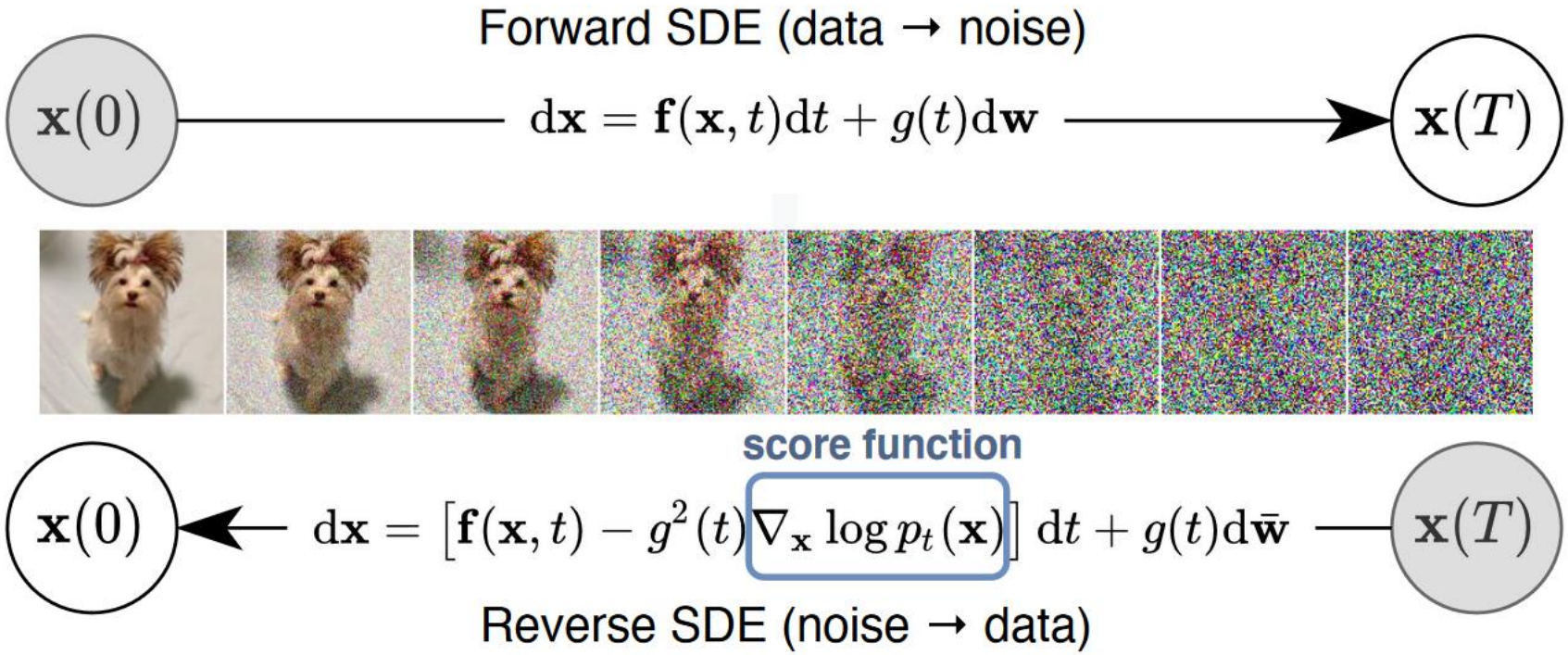
3.2 GENERATING SAMPLES BY REVERSING THE SDE

By starting from samples of $\mathbf{x}(T) \sim p_T$ and reversing the process, we can obtain samples $\mathbf{x}(0) \sim p_0$. A remarkable result from Anderson (1982) states that the reverse of a diffusion process is also a diffusion process, running backwards in time and given by the reverse-time SDE:

$$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}, \tag{6}$$

where $\bar{\mathbf{w}}$ is a standard Wiener process when time flows backwards from T to 0, and $\mathrm{d}t$ is an infinitesimal negative timestep. Once the score of each marginal distribution, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, is known for all t, we can derive the reverse diffusion process from Eq. (6) and simulate it to sample from p_0.

SCORE-BASED GENERATIVE MODELING WITH SDES



用SDE刻画DDPM

The noise perturbations used in SMLD and DDPM can be regarded as discretizations of two different SDEs. Below we provide a brief discussion and relegate more details to Appendix B.

When using a total of N noise scales, each perturbation kernel $p_{\sigma_i}(\mathbf{x} \mid \mathbf{x}_0)$ of SMLD corresponds to the distribution of $\mathbf{x}_i$ in the following Markov chain:

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2}\mathbf{z}_{i-1}, \quad i = 1, \cdots, N, \tag{8}$$

where $\mathbf{z}_{i-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and we have introduced $\sigma_0 = 0$ to simplify the notation. In the limit of $N \to \infty$, $\{\sigma_i\}_{i=1}^N$ becomes a function $\sigma(t)$, $\mathbf{z}_i$ becomes $\mathbf{z}(t)$, and the Markov chain $\{\mathbf{x}_i\}_{i=1}^N$ becomes a continuous stochastic process $\{\mathbf{x}(t)\}_{t=0}^1$, where we have used a continuous time variable $t \in [0, 1]$ for indexing, rather than an integer i. The process $\{\mathbf{x}(t)\}_{t=0}^1$ is given by the following SDE

$$\mathrm{d}\mathbf{x} = \sqrt{\frac{\mathrm{d}\left[\sigma^2(t)\right]}{\mathrm{d}t}}\mathrm{d}\mathbf{w}. \tag{9}$$

Likewise for the perturbation kernels $\{p_{\alpha_i}(\mathbf{x} \mid \mathbf{x}_0)\}_{i=1}^N$ of DDPM, the discrete Markov chain is

$$\mathbf{x}_i = \sqrt{1 - \beta_i}\mathbf{x}_{i-1} + \sqrt{\beta_i}\mathbf{z}_{i-1}, \quad i = 1, \cdots, N. \tag{10}$$

As $N \to \infty$, Eq. (10) converges to the following SDE,

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\mathbf{w}. \tag{11}$$

Therefore, the noise perturbations used in SMLD and DDPM correspond to discretizations of SDEs

DDIM

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

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

Because the generative model approximates the reverse of the inference process, we need to rethink the inference process in order to reduce the number of iterations required by the generative model. Our key observation is that the DDPM objective in the form of L_γ only depends on the marginals[2] $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$, but not directly on the joint $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$. Since there are many inference distributions (joints) with the same marginals, we explore alternative inference processes that are non-Markovian, which leads to new generative processes (Figure 1, right). These non-Markovian inference process lead to the same surrogate objective function as DDPM, as we will show below. In Appendix A, we show that the non-Markovian perspective also applies beyond the Gaussian case.

在训练DDPM所用到的L_{simple} loss中，我们甚至都没有采用跟$q(x_t|x_{t-1}, x_0)$相关的系数，而是直接选择将预测噪音的权重设置为1。由于噪音项是来自$q(x_t|x_0)$的采样，因此，DDPM的目标函数其实只由$q(x_t|x_0)$表达式决定。这其实也证实了L_{simple}与score-matching之间的联系。

换句话说，只要$q(x_t|x_0)$已知并且是高斯分布的形式，那么我们就可以用DDPM的预测噪音的目标函数L_{simple}来训练模型。

在DDPM中，基于马尔科夫性质我们认为$q(x_t|x_{t-1}, x_0)$=$q(x_t|x_{t-1})$，那么如果是服从非马尔科夫性质呢？$q(x_t|x_{t-1}, x_0)$是不是具有更一般的形式？以及，只要我们保证$q(x_t|x_0)$的形式不变，那么我们可以直接复用训好的DDPM，只不过使用新的概率分布来进行逆过程的采样。

接下来作者给出了一种非马尔科夫性质的前向扩散过程的公式以及后验概率分布的表达式，而该后验概率分布恰好满足DDPM中的边缘分布$q(x_t|x_0)$。

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$$

扩散过程

Let us consider a family $\mathcal{Q}$ of inference distributions, indexed by a real vector $\sigma \in \mathbb{R}^T_{\geq 0}$:

$$q_\sigma(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) := q_\sigma(\boldsymbol{x}_T|\boldsymbol{x}_0)\prod_{t=2}^{T} q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \tag{6}$$

where $q_\sigma(\boldsymbol{x}_T|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_T}\boldsymbol{x}_0, (1-\alpha_T)\boldsymbol{I})$ and for all $t > 1$,

$$q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}}, \sigma_t^2\boldsymbol{I}\right). \tag{7}$$

The mean function is chosen to order to ensure that $q_\sigma(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{x}_0, (1-\alpha_t)\boldsymbol{I})$ for all t (see Lemma 1 of Appendix B), so that it defines a joint inference distribution that matches the "marginals" as desired. The forward process[3] can be derived from Bayes' rule:

$$q_\sigma(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \frac{q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)q_\sigma(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}, \tag{8}$$

至此，我们仅需证明：$q_\sigma\left(x_t \mid x_0\right)$ 与DDPM中的q有相同的结构。

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$$

边缘高斯分布与条件高斯分布之间的推理关系

Marginal and Conditional Gaussians

Given a marginal Gaussian distribution for $\mathbf{x}$ and a conditional Gaussian distribution for $\mathbf{y}$ given $\mathbf{x}$ in the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \tag{2.113}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}) \tag{2.114}$$

the marginal distribution of $\mathbf{y}$ and the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ are given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \tag{2.115}$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \tag{2.116}$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}. \tag{2.117}$$

Lemma 1. *For $q_\sigma(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ defined in Eq. (6) and $q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ defined in Eq. (7), we have:*

$$q_\sigma(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{x}_0, (1-\alpha_t)\boldsymbol{I}) \tag{22}$$

Proof. Assume for any $t \leq T$, $q_\sigma(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{x}_0, (1-\alpha_t)\boldsymbol{I})$ holds, if:

$$q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0, (1-\alpha_{t-1})\boldsymbol{I}) \tag{23}$$

then we can prove the statement with an induction argument for t from T to 1, since the base case ($t = T$) already holds.

First, we have that

$$q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0) := \int_{\boldsymbol{x}_t} q_\sigma(\boldsymbol{x}_t|\boldsymbol{x}_0) q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \mathrm{d}\boldsymbol{x}_t$$

and

$$q_\sigma(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{x}_0, (1-\alpha_t)\boldsymbol{I}) \tag{24}$$

$$q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}}, \sigma_t^2 \boldsymbol{I}\right). \tag{25}$$

From Bishop (2006) (2.115), we have that $q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$ is Gaussian, denoted as $\mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$ where

$$\mu_{t-1} = \sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \frac{\sqrt{\alpha_t}\boldsymbol{x}_0 - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}} \tag{26}$$

$$= \sqrt{\alpha_{t-1}}\boldsymbol{x}_0 \tag{27}$$

and

$$\Sigma_{t-1} = \sigma_t^2 \boldsymbol{I} + \frac{1-\alpha_{t-1}-\sigma_t^2}{1-\alpha_t}(1-\alpha_t)\boldsymbol{I} = (1-\alpha_{t-1})\boldsymbol{I} \tag{28}$$

Therefore, $q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0, (1-\alpha_{t-1})\boldsymbol{I})$, which allows us to apply the induction argument. □

$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$

加速采样:

$$q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)=\mathcal{N}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0+\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\frac{\boldsymbol{x}_t-\sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}},\sigma_t^2\boldsymbol{I}\right)$$

Let us consider the forward process as defined not on all the latent variables $\boldsymbol{x}_{1:T}$, but on a subset $\{\boldsymbol{x}_{\tau_1},\dots,\boldsymbol{x}_{\tau_S}\}$, where τ is an increasing sub-sequence of $[1,\dots,T]$ of length S. In particular, we define the sequential forward process over $\boldsymbol{x}_{\tau_1},\dots,\boldsymbol{x}_{\tau_S}$ such that $q(\boldsymbol{x}_{\tau_i}|\boldsymbol{x}_0)=\mathcal{N}(\sqrt{\alpha_{\tau_i}}\boldsymbol{x}_0,(1-\alpha_{\tau_i})\boldsymbol{I})$ matches the "marginals" (see Figure 2 for an illustration). The generative process now samples latent variables according to reversed(τ), which we term *(sampling) trajectory*. When the length of the sampling trajectory is much smaller than T, we may achieve significant increases in computational efficiency due to the iterative nature of the sampling process.

$$\boldsymbol{x}_{t-1}=\sqrt{\alpha_{t-1}}\underbrace{\left(\frac{\boldsymbol{x}_t-\sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}{\sqrt{\alpha_t}}\right)}_{\text{“ predicted }\boldsymbol{x}_0\text{”}}+\underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}_{\text{“direction pointing to }\boldsymbol{x}_t\text{”}}+\underbrace{\sigma_t\epsilon_t}_{\text{random noise}}$$

$$\sigma_{\tau_i}(\eta)=\eta\sqrt{(1-\alpha_{\tau_{i-1}})/(1-\alpha_{\tau_i})}\sqrt{1-\alpha_{\tau_i}/\alpha_{\tau_{i-1}}}$$

Table 1: CIFAR10 and CelebA image generation measured in FID. $\eta=1.0$ and $\hat{\sigma}$ are cases of DDPM (although Ho et al. (2020) only considered $T=1000$ steps, and $S<T$ can be seen as simulating DDPMs trained with S steps), and $\eta=0.0$ indicates DDIM.

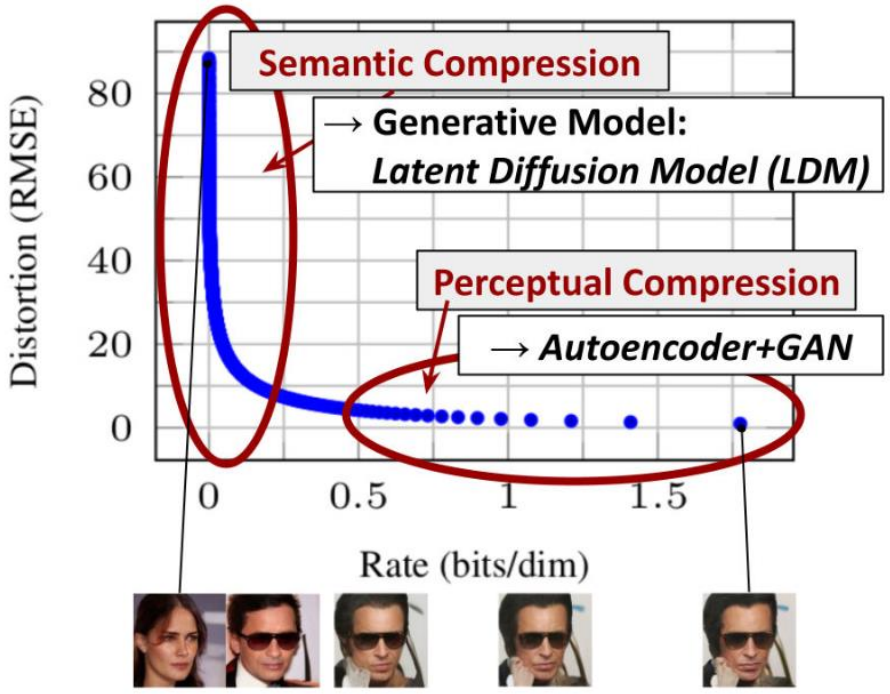
$\hat{\sigma}$: $\hat{\sigma}_{\tau_i}=\sqrt{1-\alpha_{\tau_i}/\alpha_{\tau_{i-1}}}$

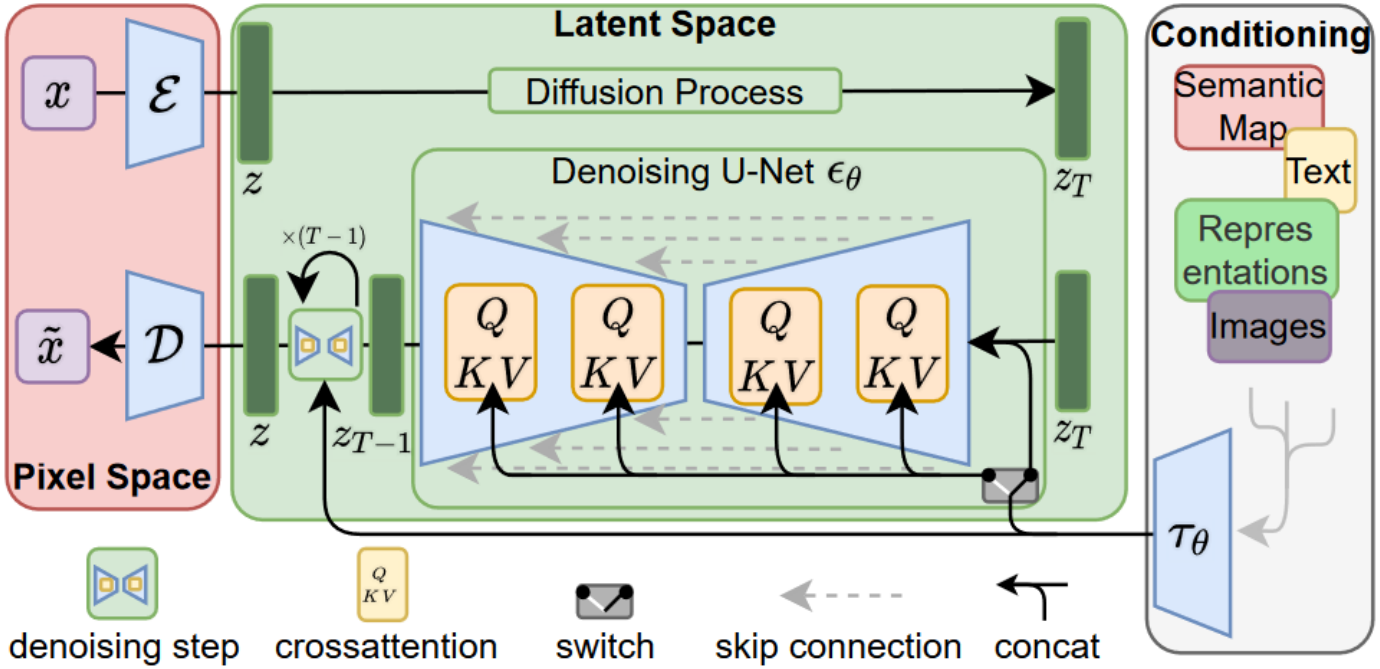
		CIFAR10 (32 × 32)					CelebA (64 × 64)				
	S	10	20	50	100	1000	10	20	50	100	1000
η	0.0	**13.36**	**6.84**	**4.67**	**4.16**	4.04	**17.33**	**13.73**	**9.17**	**6.53**	3.51
	0.2	14.04	7.11	4.77	4.25	4.09	17.66	14.11	9.51	6.79	3.64
	0.5	16.66	8.35	5.25	4.46	4.29	19.86	16.06	11.01	8.09	4.28
	1.0	41.07	18.36	8.01	5.78	4.73	33.12	26.03	18.48	13.93	5.98
$\hat{\sigma}$		367.43	133.37	32.72	9.99	**3.17**	299.71	183.83	71.71	45.20	**3.26**

Stable-Diffusion (*Latent diffusion model*)

《High-Resolution Image Synthesis With Latent Diffusion Models》

- Run the diffusion process in the latent space instead of pixel space, making training cost lower and inference speed faster





$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x),y,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta\left(z_t, t, \tau_\theta(y)\right)\|_2^2\right]$$

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\cdot\mathbf{V}$$

$$\text{where } \mathbf{Q} = \mathbf{W}_Q^{(i)}\cdot\varphi_i(\mathbf{z}_i),\ \mathbf{K} = \mathbf{W}_K^{(i)}\cdot\tau_\theta(y),\ \mathbf{V} = \mathbf{W}_V^{(i)}\cdot\tau_\theta(y)$$

$$\text{and } \mathbf{W}_Q^{(i)}\in\mathbb{R}^{d\times d_\epsilon^i},\ \mathbf{W}_K^{(i)},\mathbf{W}_V^{(i)}\in\mathbb{R}^{d\times d_\tau},\ \varphi_i(\mathbf{z}_i)\in\mathbb{R}^{N\times d_\epsilon^i},\ \tau_\theta(y)\in\mathbb{R}^{M\times d_\tau}$$

Catalogue

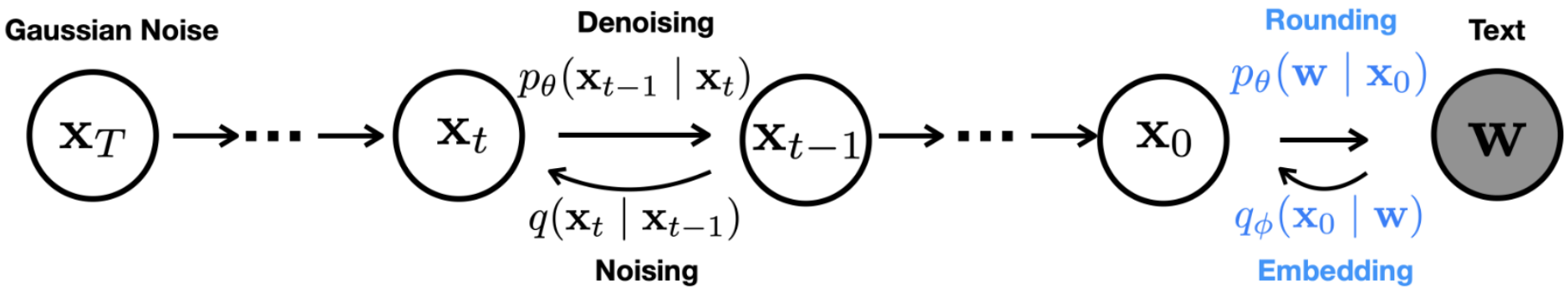
1、**Diffusion Model**(Score Based Model+Langevin Dynamics+SDE)

- DDPM
- DDIM
- Stable-Diffusion

2、**Text Generation**

- D3PM
- Diffusion-LM
- DiffuSeq
- SSD-LM
- DiffuER

Diffusion-LM 《Diffusion-LM Improves Controllable Text Generation》



$$\textsc{Emb}(\mathbf{w}) = [\textsc{Emb}(w_1), \ldots, \textsc{Emb}(w_n)] \in \mathbb{R}^{nd}.$$

As shown in Figure 2, our approach adds a Markov transition from discrete words $\mathbf{w}$ to $\mathbf{x}_0$ in the forward process, parametrized by $q_\phi(\mathbf{x}_0|\mathbf{w}) = \mathcal{N}(\textsc{Emb}(\mathbf{w}), \sigma_0 I)$. In the reverse process, we add a trainable rounding step, parametrized by $p_\theta(\mathbf{w} \mid \mathbf{x}_0) = \prod_{i=1}^{n} p_\theta(w_i \mid x_i)$, where $p_\theta(w_i \mid x_i)$ is a softmax distribution.

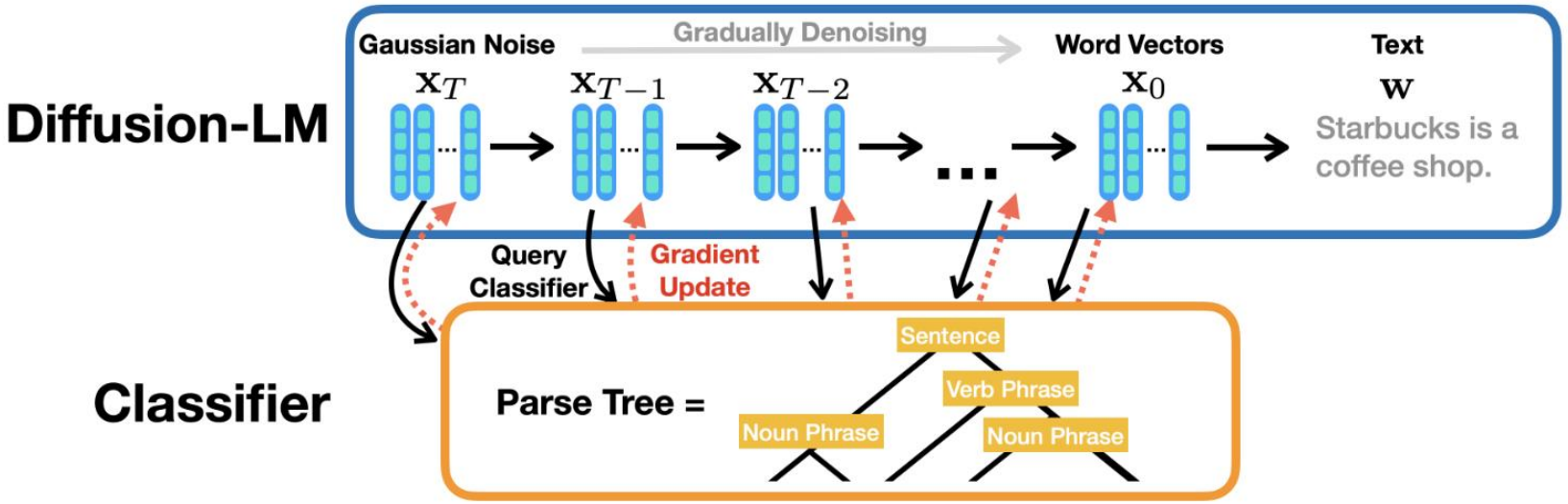
$$\mathcal{L}^{\text{e2e}}_{\text{simple}}(\mathbf{w}) = \mathop{\mathbb{E}}_{q_\phi(\mathbf{x}_{0:T}|\mathbf{w})} \left[\mathcal{L}_{\text{simple}}(\mathbf{x}_0) + ||\textsc{Emb}(\mathbf{w}) - \mu_\theta(\mathbf{x}_1, 1)||^2 - \log p_\theta(\mathbf{w}|\mathbf{x}_0)\right].$$

Reducing Rounding Errors:

1、Predict x0

2、Clamping Trick $\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}} \cdot \text{Clamp}(f_\theta(\mathbf{x}_t, t)) + \sqrt{1-\bar{\alpha}}\epsilon$

Diffusion-LM



Classifier Guided Diffusion

《Diffusion Models Beat GANs on Image Synthesis》

H Conditional Diffusion Process

In this section, we show that conditional sampling can be achieved with a transition operator proportional to $p_\theta(x_t|x_{t+1})p_\phi(y|x_t)$, where $p_\theta(x_t|x_{t+1})$ approximates $q(x_t|x_{t+1})$ and $p_\phi(y|x_t)$ approximates the label distribution for a noised sample x_t.

We start by defining a conditional Markovian noising process $\hat{q}$ similar to q, and assume that $\hat{q}(y|x_0)$ is a known and readily available label distribution for each sample.

$$\hat{q}(x_0) := q(x_0) \tag{28}$$

$$\hat{q}(y|x_0) := \text{Known labels per sample} \tag{29}$$

$$\hat{q}(x_{t+1}|x_t, y) := q(x_{t+1}|x_t) \tag{30}$$

$$\hat{q}(x_{1:T}|x_0, y) := \prod_{t=1}^{T} \hat{q}(x_t|x_{t-1}, y) \tag{31}$$

While we defined the noising process $\hat{q}$ conditioned on y, we can prove that $\hat{q}$ behaves exactly like q when not conditioned on y. Along these lines, we first derive the unconditional noising operator $\hat{q}(x_{t+1}|x_t)$:

$$\begin{aligned}
\hat{q}(x_{t+1}|x_t) &= \int_y \hat{q}(x_{t+1}, y|x_t)\, dy && (32)\\
&= \int_y \hat{q}(x_{t+1}|x_t, y)\hat{q}(y|x_t)\, dy && (33)\\
&= \int_y q(x_{t+1}|x_t)\hat{q}(y|x_t)\, dy && (34)\\
&= q(x_{t+1}|x_t) \int_y \hat{q}(y|x_t)\, dy && (35)\\
&= q(x_{t+1}|x_t) && (36)\\
&= \hat{q}(x_{t+1}|x_t, y) && (37)
\end{aligned}$$

Following similar logic, we find the joint distribution $\hat{q}(x_{1:T}|x_0)$:

$$\begin{aligned}
\hat{q}(x_{1:T}|x_0) &= \int_y \hat{q}(x_{1:T}, y|x_0)\, dy && (38)\\
&= \int_y \hat{q}(y|x_0)\hat{q}(x_{1:T}|x_0, y)\, dy && (39)\\
&= \int_y \hat{q}(y|x_0) \prod_{t=1}^{T} \hat{q}(x_t|x_{t-1}, y)\, dy && (40)\\
&= \int_y \hat{q}(y|x_0) \prod_{t=1}^{T} q(x_t|x_{t-1})\, dy && (41)\\
&= \prod_{t=1}^{T} q(x_t|x_{t-1}) \int_y \hat{q}(y|x_0)\, dy && (42)\\
&= \prod_{t=1}^{T} q(x_t|x_{t-1}) && (43)\\
&= q(x_{1:T}|x_0) && (44)
\end{aligned}$$

Using Equation 44, we can now derive $\hat{q}(x_t)$:

$$\begin{aligned}
\hat{q}(x_t) &= \int_{x_{0:t-1}} \hat{q}(x_0, ..., x_t)\, dx_{0:t-1} && (45)\\
&= \int_{x_{0:t-1}} \hat{q}(x_0)\hat{q}(x_1, ..., x_t|x_0)\, dx_{0:t-1} && (46)\\
&= \int_{x_{0:t-1}} q(x_0)q(x_1, ..., x_t|x_0)\, dx_{0:t-1} && (47)\\
&= \int_{x_{0:t-1}} q(x_0, ..., x_t)\, dx_{0:t-1} && (48)\\
&= q(x_t) && (49)\\
& && (50)
\end{aligned}$$

Using the identities $\hat{q}(x_t) = q(x_t)$ and $\hat{q}(x_{t+1}|x_t) = q(x_{t+1}|x_t)$, it is trivial to show via Bayes rule that the unconditional reverse process $\hat{q}(x_t|x_{t+1}) = q(x_t|x_{t+1})$.

One observation about $\hat{q}$ is that it gives rise to a noisy classification function, $\hat{q}(y|x_t)$. We can show that this classification distribution does not depend on x_{t+1} (a noisier version of x_t), a fact which we will later use:

$$\begin{aligned}
\hat{q}(y|x_t, x_{t+1}) &= \hat{q}(x_{t+1}|x_t, y)\frac{\hat{q}(y|x_t)}{\hat{q}(x_{t+1}|x_t)} && (51)\\
&= \hat{q}(x_{t+1}|x_t)\frac{\hat{q}(y|x_t)}{\hat{q}(x_{t+1}|x_t)} && (52)\\
&= \hat{q}(y|x_t) && (53)\\
& && (54)
\end{aligned}$$

We can now derive the conditional reverse process:

$$\begin{aligned}
\hat{q}(x_t|x_{t+1}, y) &= \frac{\hat{q}(x_t, x_{t+1}, y)}{\hat{q}(x_{t+1}, y)} && (55)\\
&= \frac{\hat{q}(x_t, x_{t+1}, y)}{\hat{q}(y|x_{t+1})\hat{q}(x_{t+1})} && (56)\\
&= \frac{\hat{q}(x_t|x_{t+1})\hat{q}(y|x_t, x_{t+1})\hat{q}(x_{t+1})}{\hat{q}(y|x_{t+1})\hat{q}(x_{t+1})} && (57)\\
&= \frac{\hat{q}(x_t|x_{t+1})\hat{q}(y|x_t, x_{t+1})}{\hat{q}(y|x_{t+1})} && (58)\\
&= \frac{\hat{q}(x_t|x_{t+1})\hat{q}(y|x_t)}{\hat{q}(y|x_{t+1})} && (59)\\
&= \frac{q(x_t|x_{t+1})\hat{q}(y|x_t)}{\hat{q}(y|x_{t+1})} && (60)\\
& && (61)
\end{aligned}$$

The $\hat{q}(y|x_{t+1})$ term can be treated as a constant since it does not depend on x_t. We thus want to sample from the distribution $Zq(x_t|x_{t+1})\hat{q}(y|x_t)$ where Z is a normalizing constant. We already have a neural network approximation of $q(x_t|x_{t+1})$, called $p_\theta(x_t|x_{t+1})$, so all that is left is an approximation of $\hat{q}(y|x_t)$. This can be obtained by training a classifier $p_\phi(y|x_t)$ on noised images x_t derived by sampling from $q(x_t)$.

4.1 Conditional Reverse Noising Process

We start with a diffusion model with an unconditional reverse noising process $p_\theta(x_t|x_{t+1})$. To condition this on a label y, it suffices to sample each transition according to

$$p_{\theta,\phi}(x_t|x_{t+1}, y) = Z p_\theta(x_t|x_{t+1}) p_\phi(y|x_t) \tag{2}$$

where Z is a normalizing constant (proof in Appendix H). It is typically intractable to sample from this distribution exactly, but Sohl-Dickstein et al. [56] show that it can be approximated as a perturbed Gaussian distribution. Here, we review this derivation.

Recall that our diffusion model predicts the previous timestep x_t from timestep x_{t+1} using a Gaussian distribution:

$$p_\theta(x_t|x_{t+1}) = \mathcal{N}(\mu, \Sigma) \tag{3}$$

$$\log p_\theta(x_t|x_{t+1}) = -\frac{1}{2}(x_t - \mu)^T \Sigma^{-1}(x_t - \mu) + C \tag{4}$$

We can assume that $\log_\phi p(y|x_t)$ has low curvature compared to Σ^{-1}. This assumption is reasonable in the limit of infinite diffusion steps, where $||\Sigma|| \to 0$. In this case, we can approximate $\log p_\phi(y|x_t)$ using a Taylor expansion around $x_t = \mu$ as

$$\log p_\phi(y|x_t) \approx \log p_\phi(y|x_t)|_{x_t=\mu} + (x_t - \mu) \nabla_{x_t} \log p_\phi(y|x_t)|_{x_t=\mu} \tag{5}$$

$$= (x_t - \mu)g + C_1 \tag{6}$$

Here, $g = \nabla_{x_t} \log p_\phi(y|x_t)|_{x_t=\mu}$, and C_1 is a constant. This gives

$$\log(p_\theta(x_t|x_{t+1}) p_\phi(y|x_t)) \approx -\frac{1}{2}(x_t - \mu)^T \Sigma^{-1}(x_t - \mu) + (x_t - \mu)g + C_2 \tag{7}$$

$$= -\frac{1}{2}(x_t - \mu - \Sigma g)^T \Sigma^{-1}(x_t - \mu - \Sigma g) + \frac{1}{2} g^T \Sigma g + C_2 \tag{8}$$

$$= -\frac{1}{2}(x_t - \mu - \Sigma g)^T \Sigma^{-1}(x_t - \mu - \Sigma g) + C_3 \tag{9}$$

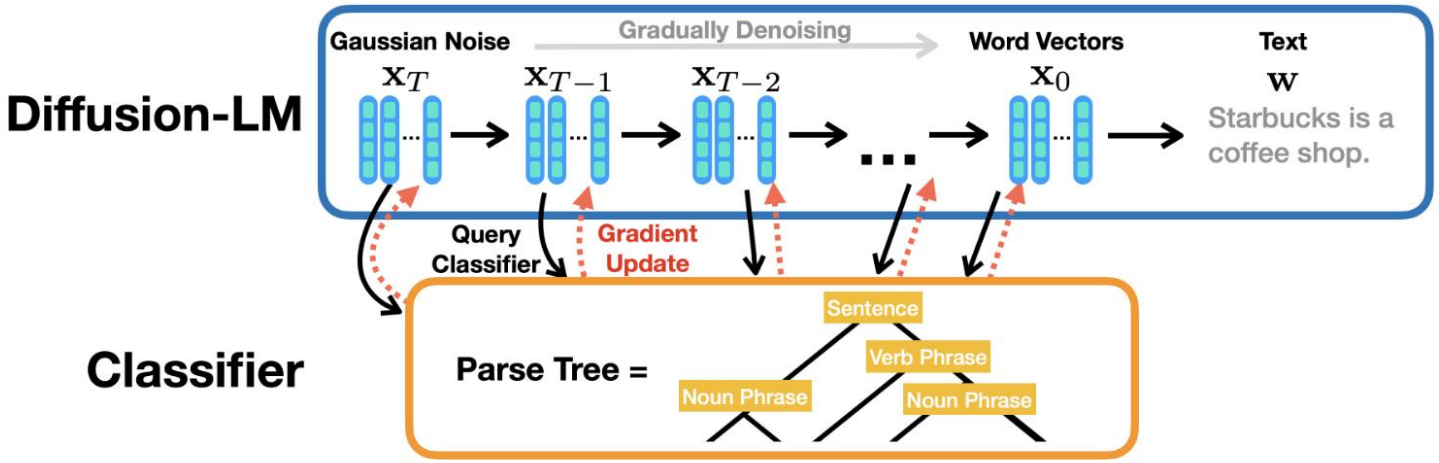
$$= \log p(z) + C_4, z \sim \mathcal{N}(\mu + \Sigma g, \Sigma) \tag{10}$$

We can safely ignore the constant term C_4, since it corresponds to the normalizing coefficient Z in Equation 2. We have thus found that the conditional transition operator can be approximated by a Gaussian similar to the unconditional transition operator, but with its mean shifted by Σg. Algorithm 1 summaries the corresponding sampling algorithm. We include an optional scale factor s for the gradients, which we describe in more detail in Section 4.3.

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
  $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$
end for
return x_0

Diffusion-LM



$$\nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}) = \nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) + \nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{c} \mid \mathbf{x}_{t-1}),$$

$$\log p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}) = \lambda \log p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) + \log p(\mathbf{c} \mid \mathbf{x}_{t-1}).$$

优点：可以实现complex、fine-grained、composed control

缺点：

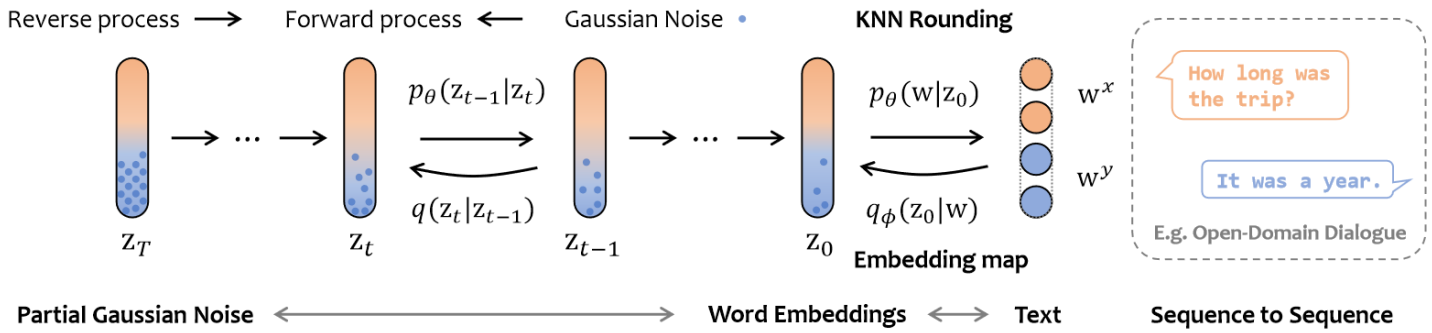
(1) it has higher perplexity;

(2) decoding is substantially slower;

(3) training converges more slowly

DiffuSeq

《DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models》



$$\text{EMB}(\mathbf{w}^{x\oplus y}) = [\text{EMB}(w_1^x), ..., \text{EMB}(w_m^x), \text{EMB}(w_1^y), ..., \text{EMB}(w_n^y)] \in \mathbb{R}^{(m+n)\times d}$$

$$q_\phi(\mathbf{z}_0|\mathbf{w}^{x\oplus y}) = \mathcal{N}(\text{EMB}(\mathbf{w}^{x\oplus y}), \beta_0\mathbf{I})$$

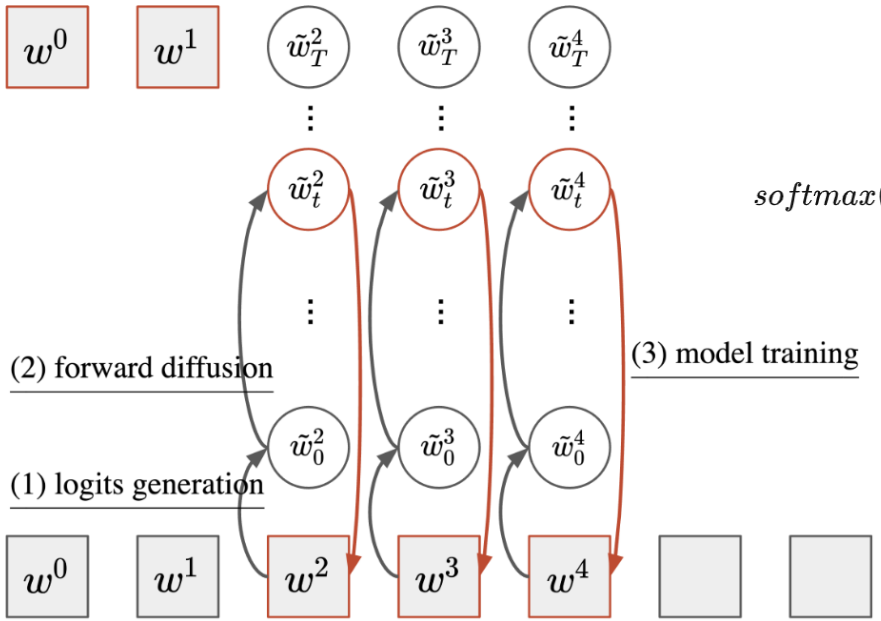
$$\min_\theta \mathcal{L}_{\text{VLB}} = \min_\theta \left[\sum_{t=2}^{T} ||\mathbf{z}_0 - f_\theta(\mathbf{z}_t, t)||^2 + ||\text{EMB}(\mathbf{w}^{x\oplus y}) - f_\theta(\mathbf{z}_1, 1)||^2 - \log p_\theta(\mathbf{w}^{x\oplus y}|\mathbf{z}_0)\right]$$

$$\rightarrow \min_\theta \sum_{t=2}^{T} \left[||\mathbf{y}_0 - \tilde{f}_\theta(\mathbf{z}_t, t)||^2\right] + ||\text{EMB}(\mathbf{w}^y) - \tilde{f}_\theta(\mathbf{z}_1, 1)|| + \mathcal{R}(||\mathbf{x}_0||^2),$$

here we use $\tilde{f}_\theta(\mathbf{z}_t, t)$ to denote the fractions of recovered $\mathbf{z}_0$ corresponding to $\mathbf{y}_0$. Note that although

SSD-LM

《SSD-LM: Semi-autoregressive Simplex-based Diffusion Language Model for Text Generation and Modular Control》



$$softmax(\tilde{w}_0^{c:c+B}) \rightarrow \cdots \rightarrow softmax(\tilde{w}_t^{c:c+B}) \rightarrow \cdots \rightarrow softmax(\tilde{w}_T^{c:c+B})$$

almost-one-hot simplex representation：

$\tilde{\boldsymbol{w}} \in \{-K, +K\}^{|V|}$:

$$\tilde{w}_{(i)} = \begin{cases} +K \text{ when } w = V_{(i)} \\ -K \text{ when } w \neq V_{(i)} \end{cases}$$

加噪声：

$$\tilde{\boldsymbol{w}}_0^{c:c+B} = \text{logits-generation}(\boldsymbol{w}^{c:c+B})$$

$$\tilde{\boldsymbol{w}}_t^{c:c+B} = \sqrt{\bar{\alpha}_t}\tilde{\boldsymbol{w}}_0^{c:c+B} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t$$

SSD-LM

Loss:在逆向过程中，用似然代替L2距离

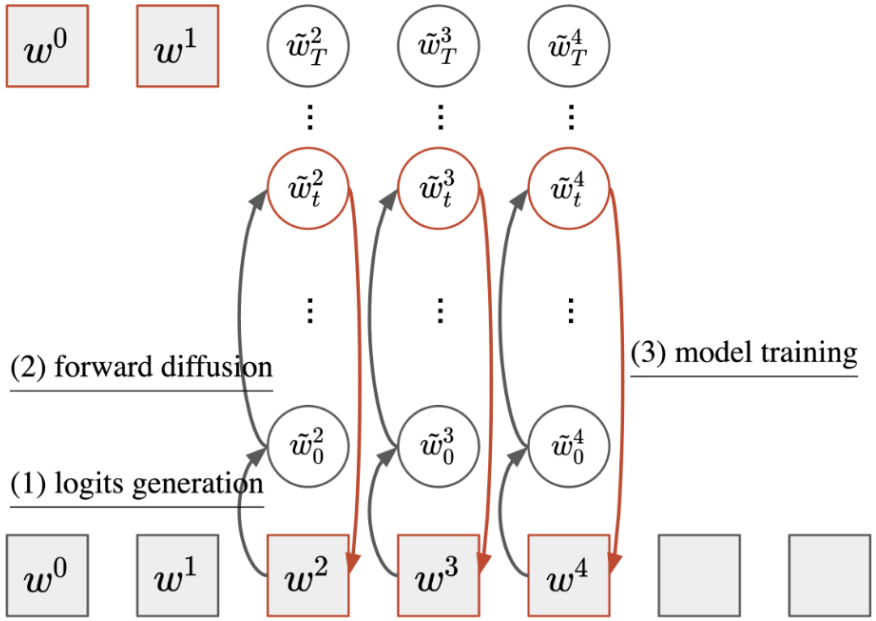
$$\mathcal{L}(\theta) = \mathbb{E}_{c\sim\mathcal{U}(1,L-B),t\sim\mathcal{U}(1,T)}[- \log p_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})]$$

$$= \mathbb{E}_{c\sim\mathcal{U}(1,L-B),t\sim\mathcal{U}(1,T)}[- \sum_{j=c}^{c+B-1} \log p_\theta(w^j \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})]$$

$$\log p_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})$$

$$= \log \frac{p_\theta(\boldsymbol{w}^{c:c+B} \mid \boldsymbol{w}^{<c})\, p_\theta(\tilde{\boldsymbol{w}}_t^{c:c+B} \mid \boldsymbol{w}^{c:c+B}, \boldsymbol{w}^{<c})}{p_\theta(\tilde{\boldsymbol{w}}_t^{c:c+B} \mid \boldsymbol{w}^{<c})}$$

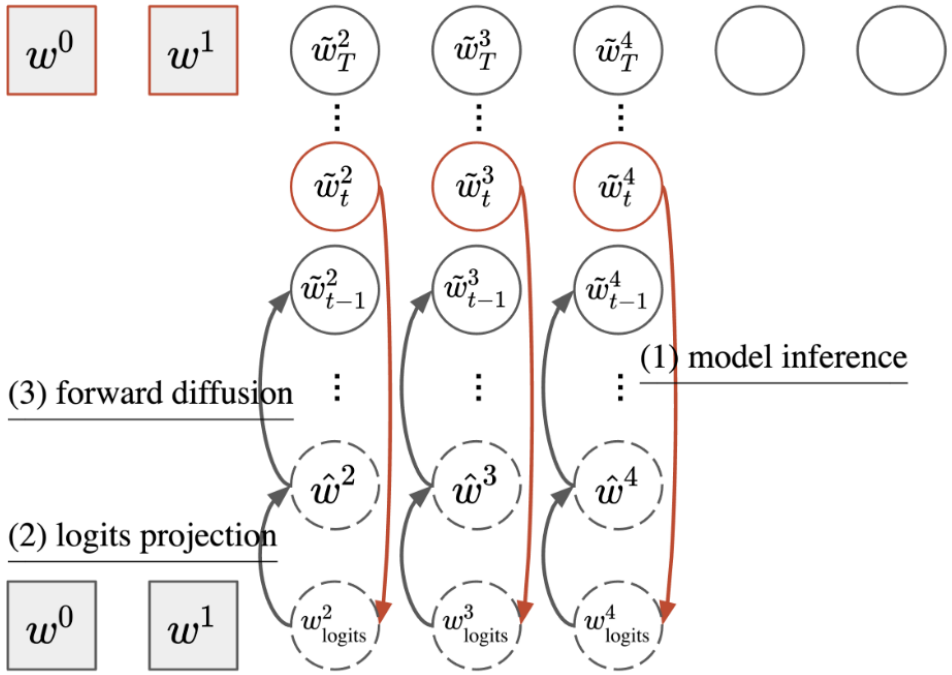
$$= \underbrace{\log p_\theta(\boldsymbol{w}^{c:c+B} \mid \boldsymbol{w}^{<c})}_{\text{likelihood of true data}} - \underbrace{\log p_\theta(\tilde{\boldsymbol{w}}_t^{c:c+B} \mid \boldsymbol{w}^{<c})}_{\text{likelihood of noisy data at timestep } t} + \underbrace{\log p(\tilde{\boldsymbol{w}}_t^{c:c+B} \mid \boldsymbol{w}^{c:c+B})}_{\text{forward diffusion process independent of } \theta}$$



Algorithm 1 Training

1: **repeat**
2: $\boldsymbol{w}^{0:L} \sim q(\boldsymbol{w}^{0:L})$
3: $c \sim \text{Uniform}(\{1, \ldots, L-B\})$
4: $\tilde{\boldsymbol{w}}_0^{c:c+B} = \text{logits-generation}(\boldsymbol{w}^{c:c+B})$
5: $t \sim \text{Uniform}(\{1, \ldots, T\})$
6: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$
7: $\tilde{\boldsymbol{w}}_t^{c:c+B} = \sqrt{\bar{\alpha}_t}\tilde{\boldsymbol{w}}_0^{c:c+B} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
8: Take gradient descent step on
$\nabla_\theta[-\sum_{j=c}^{c+B-1} \log p_\theta(w^j \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})]$
9: **until** converged

SSD-LM



$$\boldsymbol{w}_{\text{logits}}^{c:c+B}(\cdot,t) = \text{logits}_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})$$

$$\hat{\boldsymbol{w}}^{c:c+B}(\cdot,t) = \text{logits-projection}(\boldsymbol{w}_{\text{logits}}^{c:c+B}(\cdot,t))$$

$$\tilde{\boldsymbol{w}}_{t-1}^{c:c+B} = \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{w}}^{c:c+B}(\cdot,t) + \sqrt{1-\bar{\alpha}_{t-1}}\boldsymbol{z}$$

for $t = T,\ldots,1$ and $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$.

高度模块化可控: $\boldsymbol{w}_{\text{logits}}^{c:c+B}(\cdot,t) + \lambda\nabla_{\boldsymbol{w}_{\text{logits}}^{c:c+B}} f_\phi(y|\boldsymbol{w}_{\text{logits}}^{c:c+B}(\cdot,t), \boldsymbol{w}^{<c})$

- Greedy projection

$$\hat{w}_{(i)} = \begin{cases} +K \text{ if } i = \text{argmax}(\boldsymbol{w}_{\text{logits}}) \\ -K \text{ otherwise} \end{cases}$$

- top-p 采样:

$$\hat{w}_{(i)} = \begin{cases} +K \text{ if } i = \text{top-}p\text{-sampling}(\boldsymbol{w}_{\text{logits}}) \\ -K \text{ otherwise} \end{cases}$$

- *Multi-hot:* creates an almost-one-hot logit centered around *all* tokens which lie in the top-p nucleus.

$$\hat{w}_{(i)} = \begin{cases} +K \text{ if } i \in \text{top-}p\text{-all}(\boldsymbol{w}_{\text{logits}}) \\ -K \text{ otherwise} \end{cases}$$

Algorithm 2 Decoding (at a given c)

$\tilde{\boldsymbol{w}}_T^{c:c+B} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$
for $t = T,\ldots,1$ **do**
  $\boldsymbol{w}_{\text{logits}}^{c:c+B} = \text{logits}_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})$
  $\hat{\boldsymbol{w}}^{c:c+B} = \text{logits-projection}(\boldsymbol{w}_{\text{logits}}^{c:c+B})$ if uncontrolled, else $\hat{\boldsymbol{w}}^{c:c+B} = \text{logits-projection}(\boldsymbol{w}_{\text{logits}}^{c:c+B} + \lambda\nabla_{\boldsymbol{w}} f_\phi(\cdot))$
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$
  $\tilde{\boldsymbol{w}}_{t-1}^{c:c+B} = \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{w}}^{c:c+B} + \sqrt{1-\bar{\alpha}_{t-1}}\boldsymbol{z}$
end for
return $\text{argmax}\ \tilde{\boldsymbol{w}}_0^{c:c+B}$

SSD-LM

$$L = 200, B_{\text{train}} = 25, T_{\text{train}} = 5000, K = 5$$

We use an aggregated batch size of 6,144 and a learning rate of 1e-4 with an AdamW optimizer (Loshchilov and Hutter, 2019). We trained SSD-LM for 100K steps, which took about 6 days on 32 Nvidia V100 GPUs.

评价：

1. 引出利用扩散模型进行变长文本生成的一个思路，先分块，在块内缩放长度，但是引入新的超参，会让计算量变大
2. embedding层又给出一个新的思路，从对率和概率质量角度入手（对扩散模型的假设改了一大半），工作量有点大。
3. 难训练

结果:赢GPT-2

(Length 50)	MAUVE ↑	PPL $\overrightarrow{\text{gold}}$	$\lvert\Delta_{\log \text{PPL}}\rvert$ ↓	Dist-1 ↑	Dist-2 ↑	Dist-3 ↑	Zipf $\overrightarrow{\text{gold}}$	Rep ↓
Gold continuation	100.00	17.75	0.00	88.62	95.88	93.71	0.88	0.10
GPT2-medium (Best config)								
Top-p=0.95	96.57 ± 0.40	12.72 ± 0.07	0.33	66.31 ± 0.11	91.77 ± 0.03	92.75 ± 0.06	1.01	0.26 ± 0.04
GPT2-large (Best config)								
Top-p=0.95	96.41 ± 0.78	10.57 ± 0.05	0.51	64.91 ± 0.13	90.88 ± 0.06	92.38 ± 0.05	1.01	0.41 ± 0.06
GPT2-xl (Best config)								
Typical-τ=0.95	97.03 ± 0.50	10.33 ± 0.04	0.54	64.87 ± 0.15	90.69 ± 0.07	92.16 ± 0.05	1.01	0.37 ± 0.04
SSD-LM-"medium" (Top-3)								
Sampling p=0.99, T=1000	**97.89**	30.68	0.54	**68.99**	**92.60**	**92.94**	1.01	**0.16**
Sampling p=0.95, T=1000	96.64	27.34	0.43	67.75	92.16	92.91	1.01	**0.16**
Sampling p=0.9, T=2500	96.46	20.56	**0.14**	66.61	91.46	92.56	1.05	0.26

Table 1: Unconstrained generation evaluation of SSD-LM and GPT-2 models at length 50. For GPT-2 models, the results are averaged across 5 random seeds, and we show the best sampling parameter configuration. For our SSD-LM, we show the top-3 configurations. All configurations are ranked based on MAUVE, with original parameters from Pillutla et al. (2021).[10]

D3PM 《Structured Denoising Diffusion Models in Discrete State-Spaces》

- 泛化了扩散过程里的扩散方式（变高斯噪声为转移矩阵）
- 每个词都有一定概率被MASK（corrupt）或保持不变

t = 128 [MASK] [MASK] [MASK] [MASK] [MASK] [MASK]...
t = 25 In response [MASK] the demands , [MASK] [MASK]y Workers union said [MASK] backflow fund [MASK]s would face further investigation and a fine.
t = 0 In response to the demands , the Community Workers union said the backflow fund managers would face further investigation and a fine .

Original: Caterpillar is eager to expand in Asia , where it trails local competitors such as Komatsu Ltd
Corrupted: Caterpillar is eager to expand in[MASK] , [MASK] it [MASK] s local competitors such as Komatsu Ltd
Reconstructed: Caterpillar is eager to expand in China , where it faces local competitors such as Komatsu Ltd

Using a trained D3PM absorbing model for LM1B to (top) generate new sentences and (bottom) reconstruct corrupted examples.

DiffuER

《DiffusER: Discrete Diffusion via Edit-based Reconstruction》

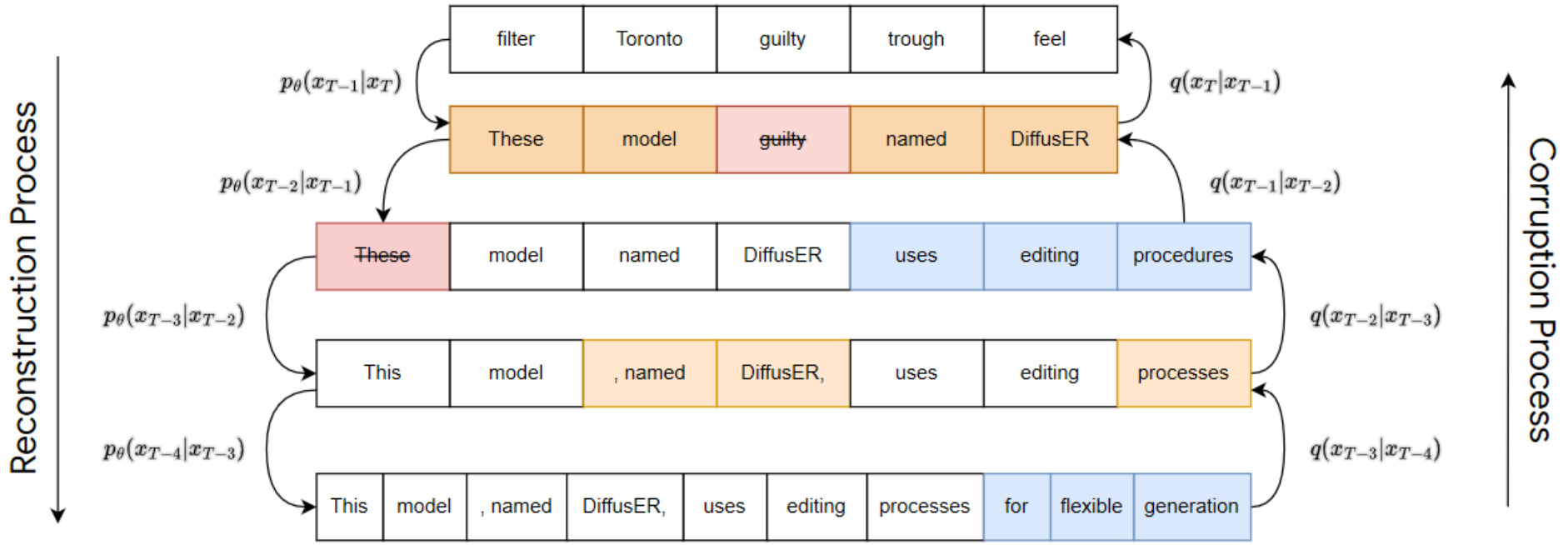


Figure 1: DIFFUSER's text generation process. Orange represents replacements, blue represents insertions, red represents deletions, and white represents keep operations. This process largely imitates a natural editing process (Reid & Neubig, 2022).

DiffuER

Corruption: $q(\mathbf{x}_i|\mathbf{x}_{i-1}; \mathcal{E}_t, \mathcal{E}_l)$

$\mathcal{E}_t$ the distribution over edit types (e.g. 60% keep, 20% replace, 10% delete, 10% insert)
$\mathcal{E}_l$ the distribution over edit length(a Poisson distribution)

Reconstruction: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = p_\theta^{\text{tag}}(\mathbf{e}_t|\mathbf{x}_t)p_\theta^{\text{gen}}(\mathbf{x}_{t-1}|\mathbf{x}_t|\mathbf{e}_t)$ $\mathbf{e}_t$:edit operations

tagging process :identify which edits should take place
generative process :deciding which tokens should go in these positions

DiffuER

实例：

Step 1	elf meantime Nano (¡ Aden Prepare ~~hue mere strictlyrights~~ hueHeat Goalsgeordnet LewisSession beet remindersrights rézes redund boldWisconsin-Port compl rocks@@~~actual Parish norm Lawyers Organisation deprecatedince eradicateewerkschaften~~ oyleingebracht naked Lawyers Organisation von Gewerkschaften al contestants negligible GeneIZE etablieren.HT
Step 2	elf meantime Nano (¡ ~~Aden Prepare hueHeat Goalsgeordnet~~ aggravatedabgeordnet LewisSession beet remindersrights rézes redund ~~boldWisconsinPort compl rocks@@oyleingebracht~~ boldWisconsin eingebracht naked Lawyers Organisation von Gewerkschaften al contestants negligible 2400 CLR GeneIZE etablieren.~~HT~~ isationatar ent
Step 3	~~elf meantime Nano (¡ aggravatedabgeordnet~~ containing Tai Prison Kongressabgeordnet und John LewisSession beet remindersrights ~~rézes redund boldWisconsin~~ rézesvorschlag eingebracht naked Lawyers Organisation von Gewerkschaften ~~al contestants negligible 2400 CLR GeneIZE~~ als Bürgerrecht zu etablieren.isationatar ent
Step 4	~~containing Tai Prison Kongressabgeordnet und John LewisSession beet remindersrights rézesvorschlag eingebrachtnaked Lawyers~~ Die Kongressabgeordneten Keith Ellison und John Lewis haben einen Gesetzesvorschlag eingebracht, um die Organisation von Gewerkschaften als Bürgerrecht zu etablieren. ~~isationatar ent~~
Target	Die Kongressabgeordneten Keith Ellison und John Lewis haben einen Gesetzesvorschlag eingebracht, um die Organisation von Gewerkschaften als Bürgerrecht zu etablieren.

Source Document	(CNN)They’re not gonna take it anymore. Really. Twisted Sister says that its 2016 tour will be its last, according to a press release. Next year marks the band’s 40th anniversary, and to celebrate, the tour is being titled ”Forty and F*ck It.” ”It’s official: Farewell,” Twisted Sister singer Dee Snider posted on Facebook. Snider also noted that the band will play with a new drummer, Mike Portnoy of Adrenaline Mob. Portnoy replaces A.J. Pero, who died March 20. The band will also perform two shows in Pero’s honor: one at Las Vegas’ Hard Rock Hotel and Casino, the other at the Starland Ballroom in Sayreville, New Jersey. The latter is in support of Pero’s family. Twisted Sister’s biggest hit, ”We’re Not Gonna Take It,” hit the Top Forty in 1984 and was featured in a popular video.
Step 1	~~(CNN)They’re not gonna take it anymore. Really.~~ Twisted Sister says that its 2016 tour will be its last, according to a press release. Next year marks the band’s 40th anniversary, and to celebrate, the tour is being titled ”Forty and F*ck It.” ”It’s official: Farewell,” Twisted Sister singer Dee Snider posted on Facebook. Snider also noted that the band will play with a new drummer, Mike Portnoy of Adrenaline Mob. Portnoy replaces A.J. Pero, who died March 20. The band will also perform two shows in Pero’s honor: one at Las Vegas’ Hard Rock Hotel and Casino, the other at the Starland Ballroom in Sayreville, New Jersey. The latter is in support of Pero’s family. Twisted Sister’s biggest hit, ”We’re Not Gonna Take It,” hit the Top Forty in 1984 and was featured in a popular video.
Step 2	Twisted Sister says that its 2016 tour will be its last, according to a press release. Next year marks the band’s 40th anniversary, and to celebrate, the tour is being titled ”Forty and F*ck It.” ~~”It’s official: Farewell,” Twisted Sister singer Dee Snider posted on Facebook. Snider also noted that the band will play with a new drummer, Mike Portnoy of Adrenaline Mob.~~ Portnoy replaces A.J. Pero, who died March 20. The band will also perform two shows in Pero’s honor: one at Las Vegas’ Hard Rock Hotel and Casino, the other at the Starland Ballroom in Sayreville, New Jersey. The latter is in support of Pero’s family. Twisted Sister’s biggest hit, ”We’re Not Gonna Take It,” hit the Top Forty in 1984 and was featured in a popular video.
Step 3	Twisted Sister says that its 2016 tour will be its last, according to a press release. Next year marks the band’s 40th anniversary, and to celebrate, the tour is being titled ”Forty and F*ck It.” Portnoy replaces A.J. Pero, who died March 20. The band will ~~also~~ perform two shows in Pero’s honor ~~: one at Las Vegas’ Hard Rock Hotel and Casino, the other at the Starland Ballroom in Sayreville, New Jersey. The latter is in support of Pero’s family. Twisted Sister’s biggest hit, ”We’re Not Gonna Take It,” hit the Top Forty in 1984 and was featured in a popular video~~in Las Vegas and New Jersey.
Step 4	Twisted Sister says that its 2016 tour will be its last, according to a press release. Next year marks the band’s 40th anniversary, and to celebrate, the tour is being titled ”Forty and F*ck It.” ~~Portnoy replaces~~ A.J. Pero, ~~who~~ died March 20. The band will perform two shows in Pero’s honor in Las Vegas and New Jersey.
Generated Summary	Twisted Sister says that its 2016 tour will be its last. Next year marks the band’s 40th anniversary, and to celebrate, the tour is being titled ”Forty and F*ck It.” A.J. Pero, died March 20. The band will perform two shows in Pero’s honor in Las Vegas and New Jersey.

DiffuER

Model	En-De (MT)	CNN-DM (Summ)
AR Transformer (Vaswani et al., 2017)	27.3	36.8
SUNDAE (Savinov et al., 2022)	26.3	37.0
CMLM (Ghazvininejad et al., 2019)	24.6	—
Levenshtein Transformer[2] (Gu et al., 2019)	23.7	—
DisCo (Kasai et al., 2020a)	24.7	—
Imputer	25.2	—
DIFFUSER	27.2	37.8
DIFFUSER + AR bootstrap	**28.8**	38.4
DIFFUSER + source bootstrap	24.5	**38.9**

Table 2: Machine Translation (MT) and Summarization (Summ) results on WMT'14 En-De (gold) and CNN-DailyMail. Experiments on MT use BLEU while summarization uses ROUGE. DIFFUSER is compatible with a standard autoregressive model, while outperforming previous methods.

Model	Accuracy	BLEU
Masker (Malmi et al., 2020)	40.9	14.5
Tag and Generate (Madaan et al., 2020)	86.2	19.8
LEWIS (Reid & Zhong, 2021)	**93.1**	24.0
DIFFUSER	87.6	**25.2**

Table 3: Results on Yelp dataset for text style transfer. Without task-specific training techniques, DIFFUSER performs comparably to previous task-specific methods.

Thanks for your listening

Paper Reading : RenZHi Wang
2022.11.22