Lecture 18: Deep Generative Models VII: DDIM

Lan Xu
SIST, ShanghaiTech
Fall, 2024

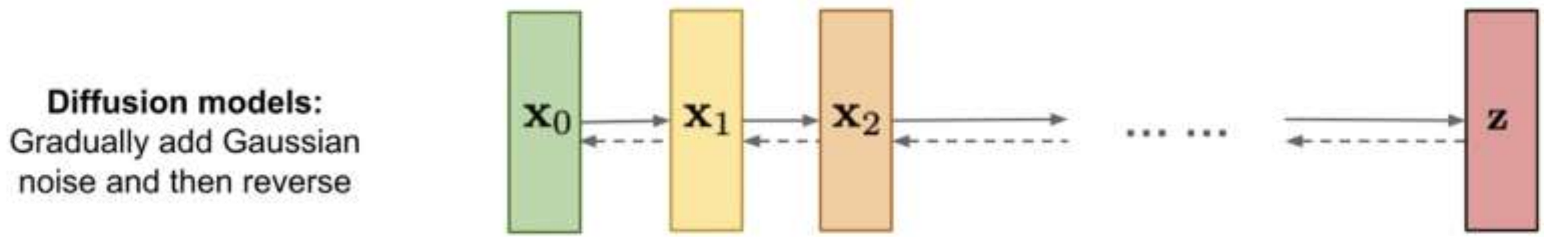
Outline

- Revisit DDPM
- A score-based view angle
- Acceleration, mostly DDIM

Revisit DDPM

Anatomy of a Diffusion Model

- Forward Process
- Reverse Process



DDPM

- Denoising Diffusion Probabilistic Models
- Target: understand the training and sampling phases!

Algorithm 1 Training

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

Algorithm 2 Sampling

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

Diffusion models

- What if we add a bunch of Gaussian noise to an image?

Diffusion models

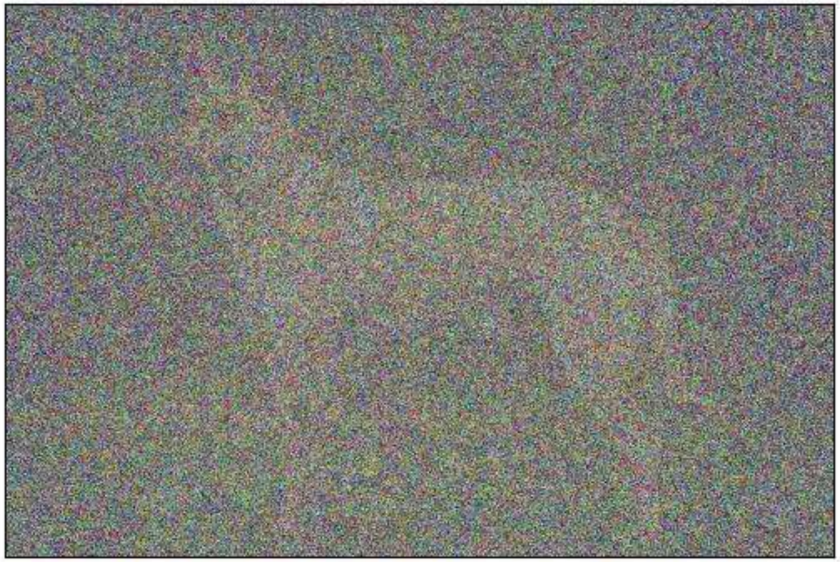
- and again…

Diffusion models

- and again…

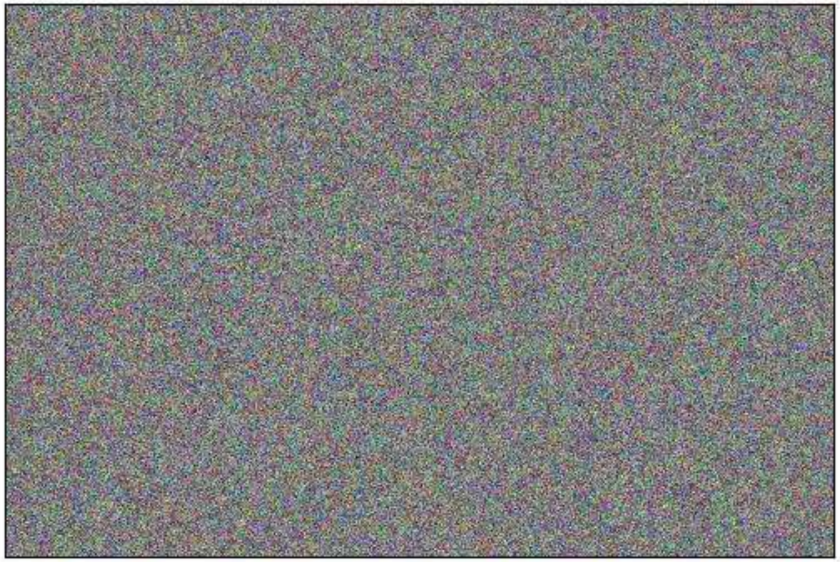
Diffusion models

- and again…



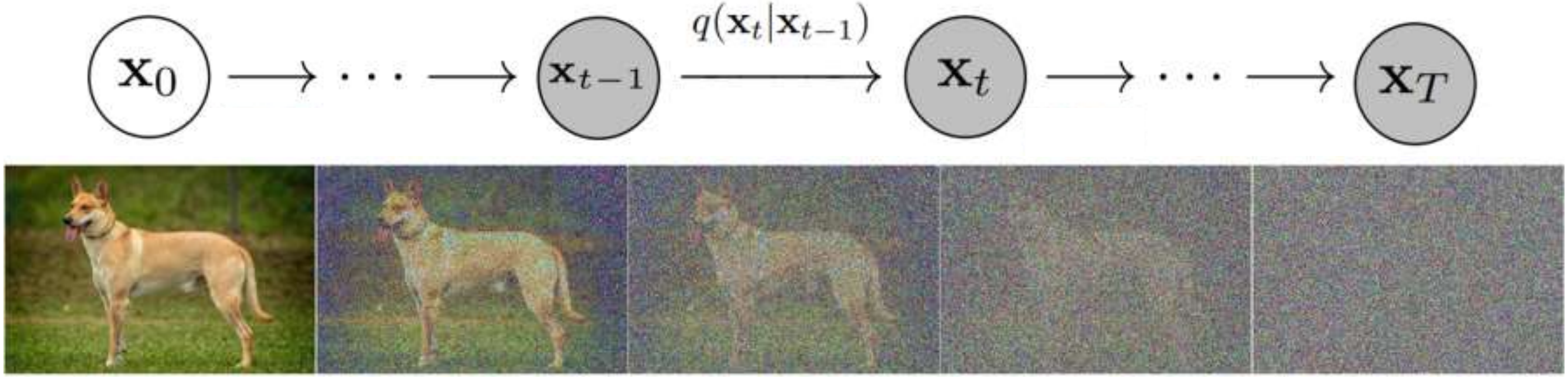
Diffusion models

- … until it resembles pure noise



Diffusion models

- Forward Process



- Take a datapoint **x_0** and gradually add very small amounts of Gaussian noise to it
- Let **x_t** be the datapoint after t iterations
- This is called the **forward diffusion process**
- Repeat this process for **T** steps — over time, more and more features of the original input are destroyed until you get something resembling **pure noise**

Diffusion models

- Forward Process

$$\mathbf{x}_0 \longrightarrow \cdots \longrightarrow \mathbf{x}_{t-1} \xrightarrow{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \mathbf{x}_t \longrightarrow \cdots \longrightarrow \mathbf{x}_T$$

- More formally, we update each image over time as

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0,\, I)$$

where $\{\beta_t \in (0,1)\}_{t=1}^{T}$

is called the **noise schedule** (basically a hyperparameter describing how much noise to add at a given timestep).
The update above can equivalently be written as a sampling process from the following Gaussian distribution:

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

Diffusion models

- A neat (reparametrization) trick! $\{\beta_t \in (0,1)\}_{t=1}^{T}$

$$\beta_1 < \beta_2 < \ldots < \beta_T$$

- Define:

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

- Then:

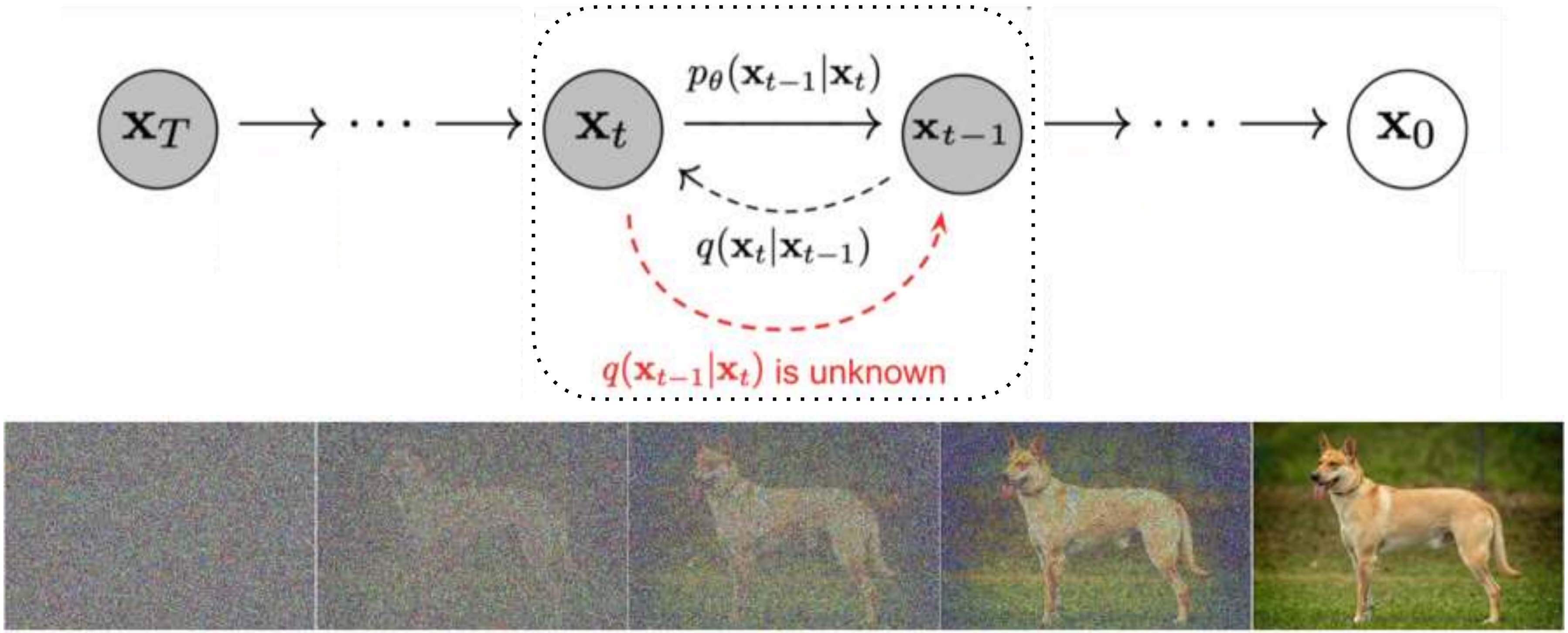
$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}\right)$$

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}) \\
&= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon \\
&= \ldots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon
\end{aligned}$$

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

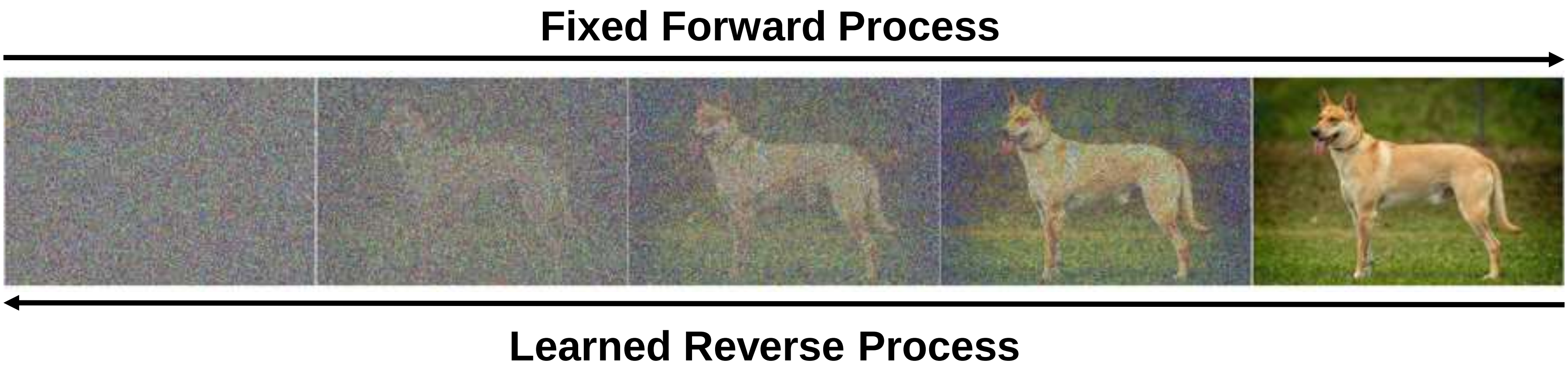
Diffusion models

- Can we go in the other direction?



Diffusion models

- Reverse Process

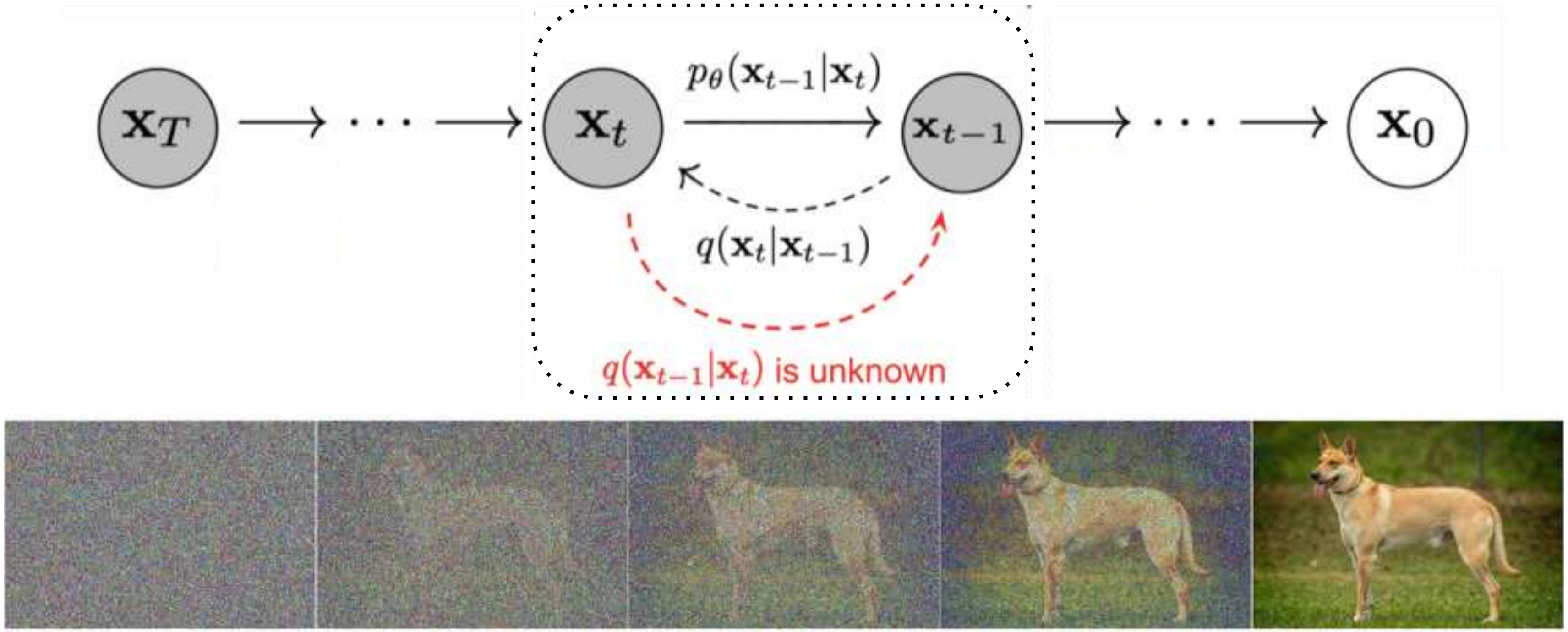


- The goal of a diffusion model is to **learn the reverse denoising process** to iteratively **undo** the forward process
- In this way, the reverse process appears as if it is generating new data from random noise!

Diffusion models

- Reverse Process

We are given **q(x_t | x_{t-1})**. How do we find **q(x_{t-1} | x_t)**?



Diffusion models

- Finding the exact distribution is hard
- Remember Bayes rule?

$$f(\theta \mid x) = \frac{f(\theta, x)}{f(x)} = \frac{f(\theta)\, f(x \mid \theta)}{f(x)} \quad \Longrightarrow \quad q(x_{t-1} \mid x_t) = q(x_t \mid x_{t-1}) \frac{q(x_{t-1})}{q(x_t)}$$

$$q(x_t) = \int q(x_t \mid x_{t-1}) q(x_{t-1}) \mathrm{d}x$$

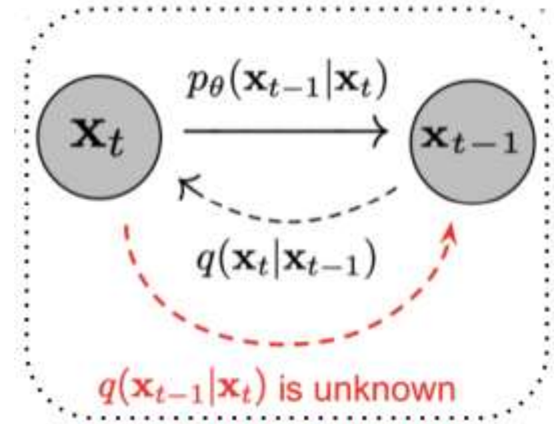
- The distribution of each timestep and q(x_t | x_{t-1}) depends on the entire data distribution:
- This is computationally intractable
  - Need to integrate over the whole data distribution to find **q(x_t)** and **q(x_{t-1})**
  - Where else have we seen this dilemma?
- We still need the posterior distribution to carry out the reverse process. Can we approximate this somehow?

Diffusion models

- Can we go in the other direction?
- A naïve solution, don't work:

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon$$

$$x_{t-1} = (x_t - \sqrt{\beta_t}\epsilon_{t-1})/\sqrt{1-\beta_t}$$



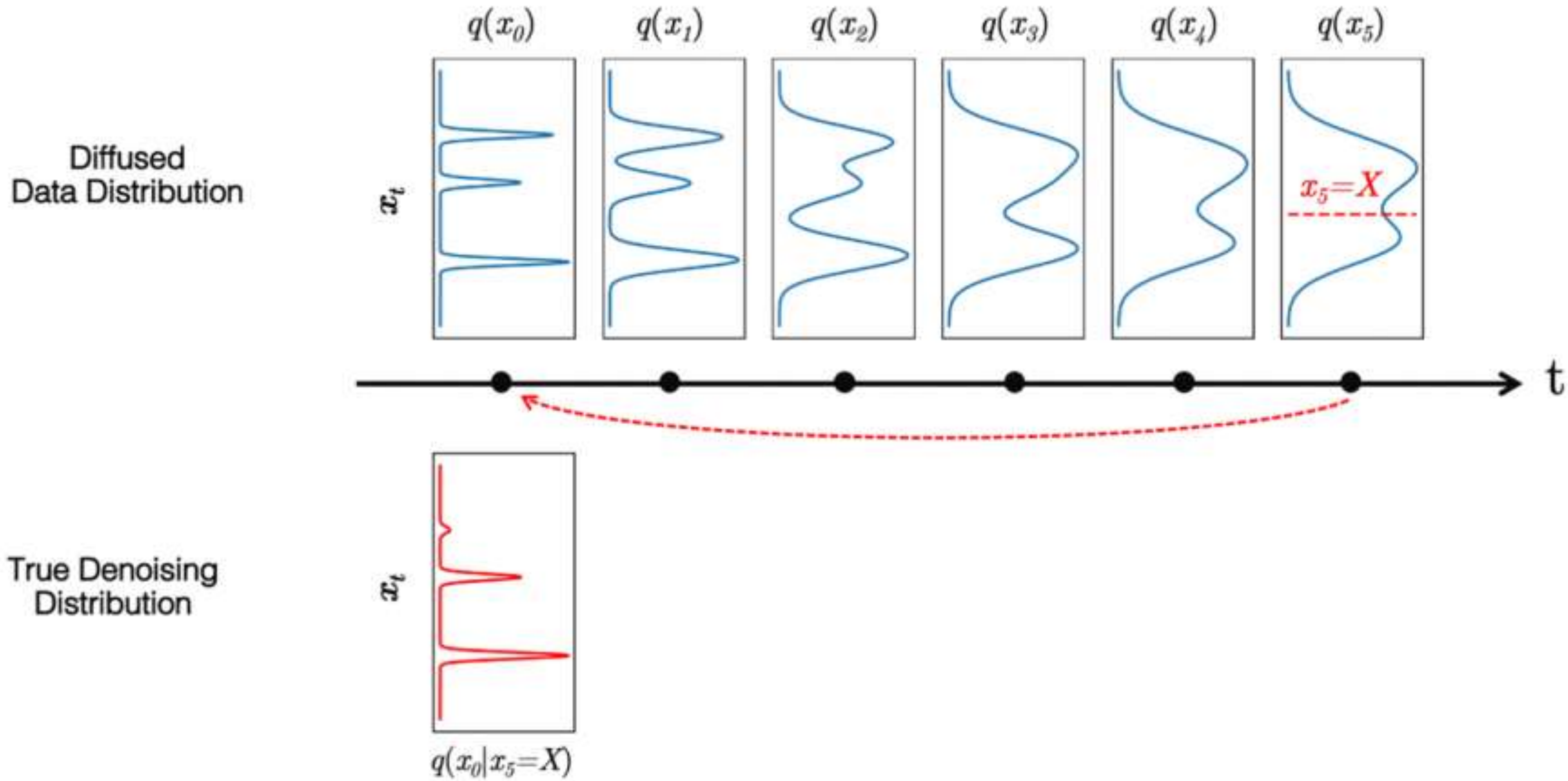
- Then, let a NN estimate ϵ_{t-1}

$$x_{t-1} = (x_t - \sqrt{\beta_t}\epsilon_\theta(x_t, t))/\sqrt{1-\beta_t}$$

- Problem: interactive training, super non-efficient

- **Solution in DDPM**: use the reparametrization trick, from $q(x_{t-1}|x_t)$ to $q(x_{t-1}|x_t, x_0)$

Denoising Diffusion Probabilistic Models



Denoising Diffusion Probabilistic Models

- What does the final reverse process look like?

- In practice, we choose **our noise schedule** such that the forward process steps are **very small**.

$$\{\beta_t \in (0,1)\}_{t=1}^{T}$$

- Thus, we approximate the reverse posterior distributions

$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ as Gaussians and **learn** their parameters (i.e., the mean and variance) via neural networks

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

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

- Cool, we now have an idea of what the model looks like. How do we train it?

DDPM models

- A preliminary objective

We want to maximize the log-likelihood of the data generated by a reverse process.

Remember that VAEs tried to do something similar but they maximized a lower bound on the likelihood instead because the actual likelihood is computationally intractable

$$\boxed{-L_{\text{VAE}} = \log p_\theta(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})}$$

We can apply the same trick to diffusion!

$$-L = \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}\right] \leq \log p_\theta(\mathbf{x}_0)$$

DDPM models

- A preliminary objective

The VAE (ELBO) loss is a bound on the true log likelihood (also called the *variational lower bound*)

$$\boxed{-L_{\text{VAE}} = \log p_\theta(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})}$$

Apply the same trick to diffusion:

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

In details:

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

DDPM models

- A preliminary objective

The VAE (ELBO) loss is a bound on the true log likelihood (also called the *variational lower bound*)

$$\boxed{-L_{\text{VAE}} = \log p_\theta(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})}$$

Apply the same trick to diffusion:

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

Expanding out,

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

DDPM models

- A more thorough derivation

$$
\begin{aligned}
L &= \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] \\
&= \mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{\text{L\_T}} + \underbrace{\sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{\text{L\_t}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{\text{L\_0}}\right]
\end{aligned}
$$

DDPM models

- A simplified objective: use the **reparametrization trick**, from $q(x_{t-1}|x_t)$ to $q(x_{t-1}|x_t, x_0)$

- The reverse step conditioned on x_0 is a **Gaussian**:

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

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

- After **doing some algebra**, each loss term can be approximated by:

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, I)$$

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

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

DDPM models

- Some algebra here ……
- The reverse step conditioned on x_0 is a **Gaussian**

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_{t-1}, x_t, x_0)}{q(x_t, x_0)} = \frac{q(x_t|x_{t-1}, x_0) \cdot q(x_{t-1}, x_0)}{q(x_t, x_0)} = \frac{q(x_t|x_{t-1}, x_0) \cdot q(x_{t-1}|x_0)}{q(x_t|x_0)}$$
$$= \frac{q(x_t|x_{t-1}) \cdot q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

- Note that: $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t \boldsymbol{I})$
- Let's handle: $q(x_t|x_0)$ using the **reparametrization trick**

$$\begin{aligned}
q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) &= \mathcal{N}\left(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right) \\
\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}) \\
&= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon \\
&= \sqrt{\alpha_t \alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t \alpha_{t-1}}\epsilon \\
&= \ldots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \\
q(\mathbf{x}_t \mid \mathbf{x}_0) &= \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\right)
\end{aligned}$$

$$\alpha_t = 1 - \beta_t$$

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

DDPM models

- Some algebra here …… $\frac{q(x_t|x_{t-1}) \cdot q(x_{t-1}|x_0)}{q(x_t|x_0)}$
- All are Gaussians now：
  → If $x \sim \mathcal{N}(\mu, \sigma^2)$, then $q(x) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(x-\mu)^2}{2\sigma^2})$
- Thus, $\boxed{q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t \boldsymbol{I})}$

where $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$ and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$

- Recall $x_t = \sqrt{\overline{\alpha}_t} x_0 + \sqrt{1 - \overline{\alpha}_t} \overline{\epsilon}_t$ and $x_0 = \frac{1}{\sqrt{\overline{\alpha}_t}}(x_t - \sqrt{1-\overline{\alpha}_t}\overline{\epsilon}_t)$
- Thus, $\boxed{\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\overline{\epsilon}_t)}$, use NN to estimate it !!
- Only rely on $\overline{\epsilon}_t$, from x_0 to x_t, with only one sampling!

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

DDPM models

- A simplified objective: use the **reparametrization trick**
- Instead of predicting the **mu**, Ho et al. say that we should predict **epsilon** instead!

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right) \Longrightarrow \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right)$$

- Thus, our loss becomes:

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

DDPM models

- Some algebra here ……
$$\frac{q(x_t|x_{t-1})\cdot q(x_{t-1}|x_0)}{q(x_t|x_0)}$$
- All are Gaussians now：

  → If $x \sim \mathcal{N}(\mu, \sigma^2)$, then $q(x) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(x-\mu)^2}{2\sigma^2})$
- Thus, $\boxed{q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t \boldsymbol{I})}$

where $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$ and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$

- Recall $x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\bar{\epsilon}_t$ and $x_0 = \frac{1}{\sqrt{\overline{\alpha}_t}}(x_t - \sqrt{1-\overline{\alpha}_t}\bar{\epsilon}_t)$
- Thus, $\boxed{\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\bar{\epsilon}_t)}$, use NN to estimate it !!
- Only rely on $\bar{\epsilon}_t$, from x_0 to x_t, with only one sampling!

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

DDPM models

- Rethinking the Training and Sampling processes…..

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
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** $t = T, \ldots, 1$ **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** $\mathbf{x}_0$

- During training, add noise from 0 to t, then estimate it
- During sampling, note that $\sigma_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$
- As t increases, $\bar{\alpha}_t$ decreases, $\sqrt{1-\bar{\alpha}_t}$ increases
- Thus, $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ works as denoise auto-encoder for various noise levels!

DDPM models

- If we have the noise, sampling by using Gaussians:

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

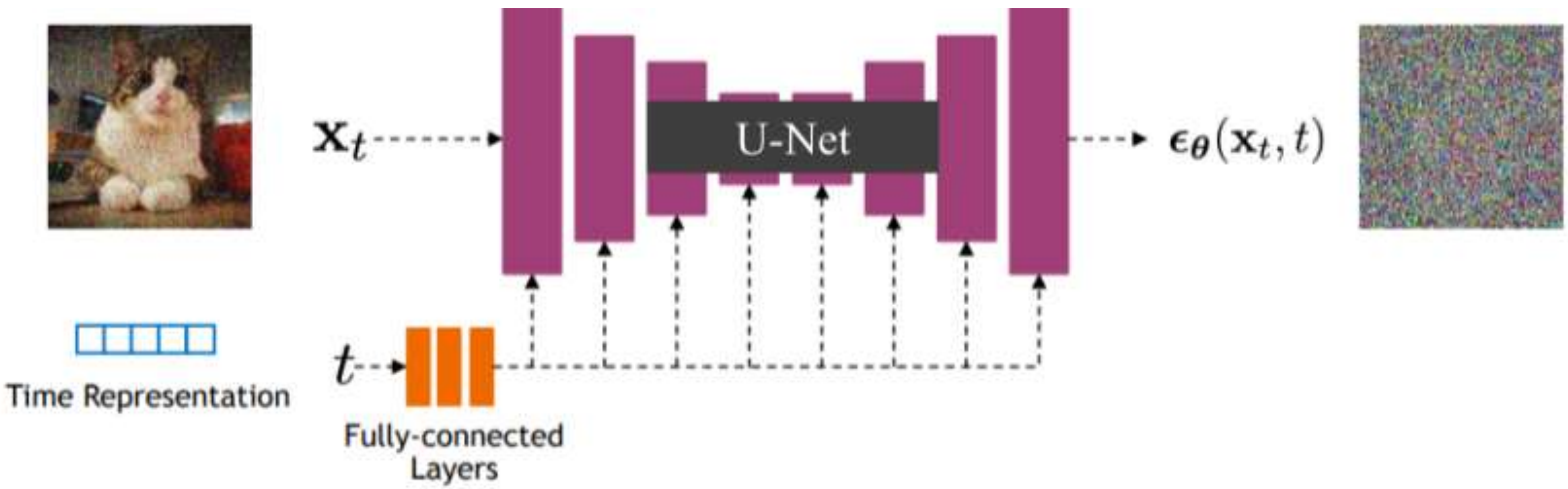
- 1) sampling z_t
- 2) sampling x_t-1, using the estimated noise

$$x_{t-1} = \tilde{\mu}_t + \tilde{\beta}_t \cdot z_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\bar{\epsilon}_t) + \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \cdot z_t$$

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)) + \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \cdot z_t$$

DDPM models

- Network Structure



Algorithm 1 Training

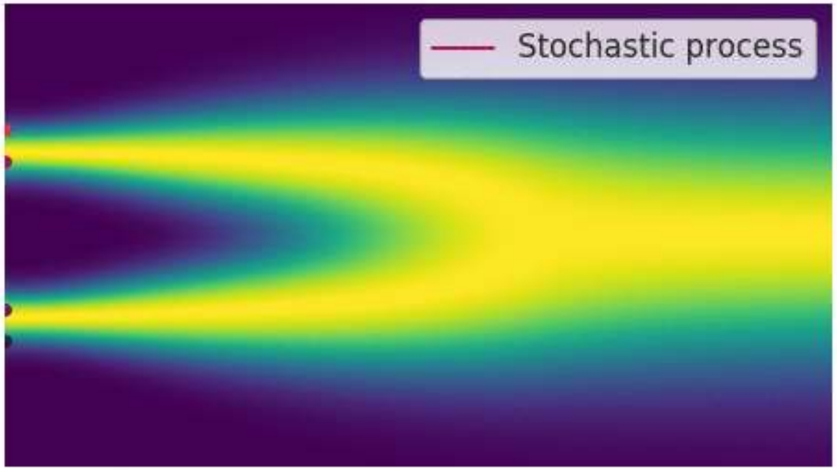
1: **repeat**
2: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $\quad t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad$ Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

Algorithm 2 Sampling

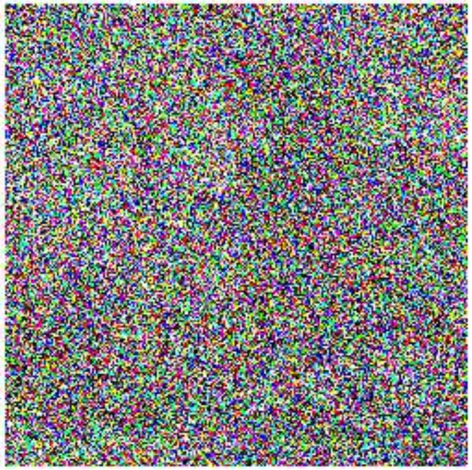
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \boxed{\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}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

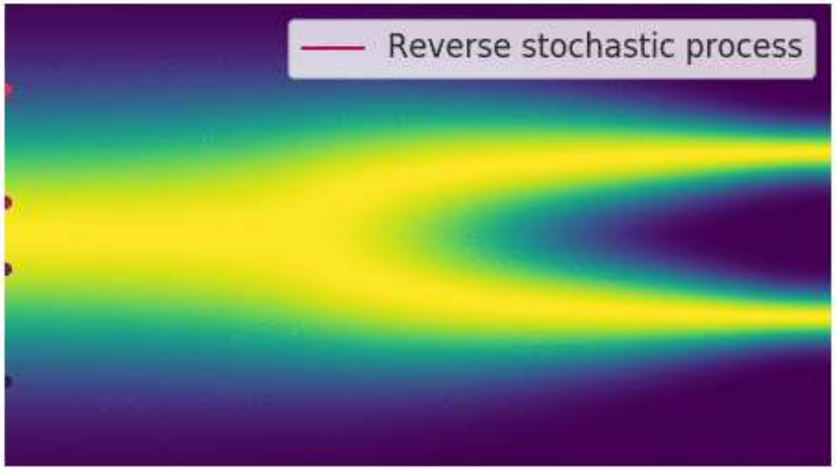
DDPM models

- Forward/Reverse process for Image Generation



Forward process: converting the image distribution to pure noise





Reverse process: sampling from the image distribution, starting with pure noise

Score-based Diffusion Models

Diffusion and Score Matching

- Diffusion Models are closely related to **Score Matching**.
- Score Matching is one solution to **Energy-based Models**.
- Energy-based Models:
  - can be probabilistic or non-probabilistic
  - can be generative or discriminative
- Many useful concepts in diffusion co-evolved w/ score matching
  - Annealed importance sampling [Neal 1998]
  - Denoising score matching [Vincent 2011]
  - Noise Conditional Score Network [Song & Ermon 2019]

What is Score function

- Probability density function (pdf)

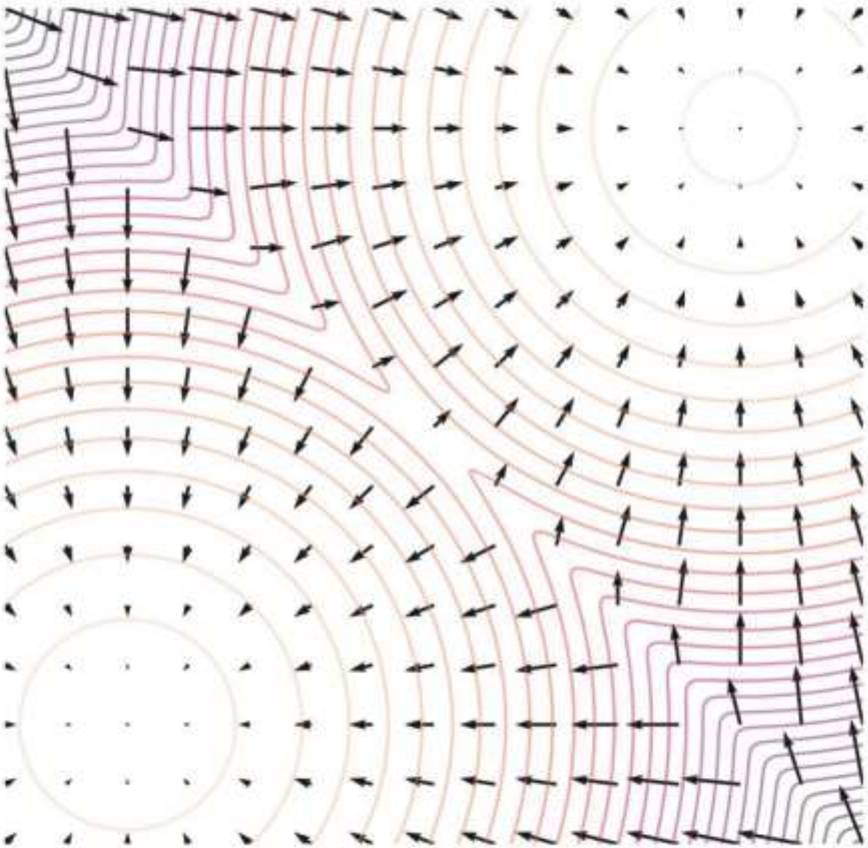
$$p(\mathrm{X})$$

- Score function

$$\nabla_{\mathrm{x}} \log p(\mathrm{X})$$

- e.g. Gaussian distribution

$$p_\theta(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$
$$\longrightarrow s_\theta(x) = -\frac{x-\mu}{\sigma^2}$$



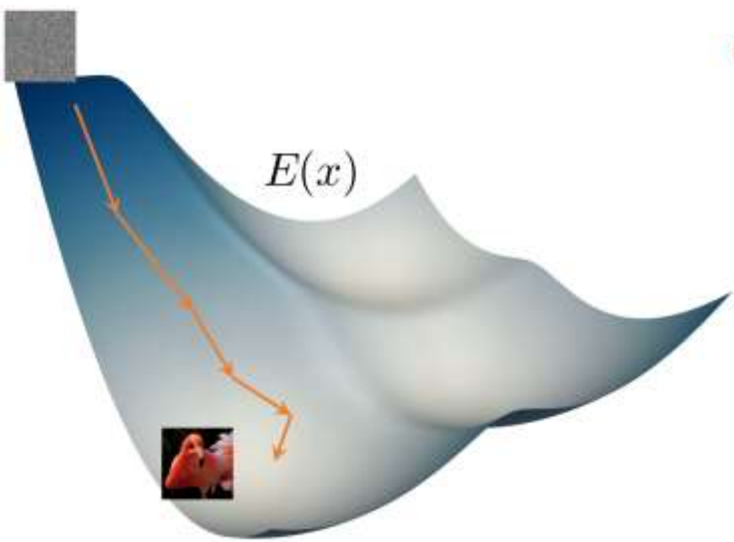
Density function: Contours
Score function: Vector field

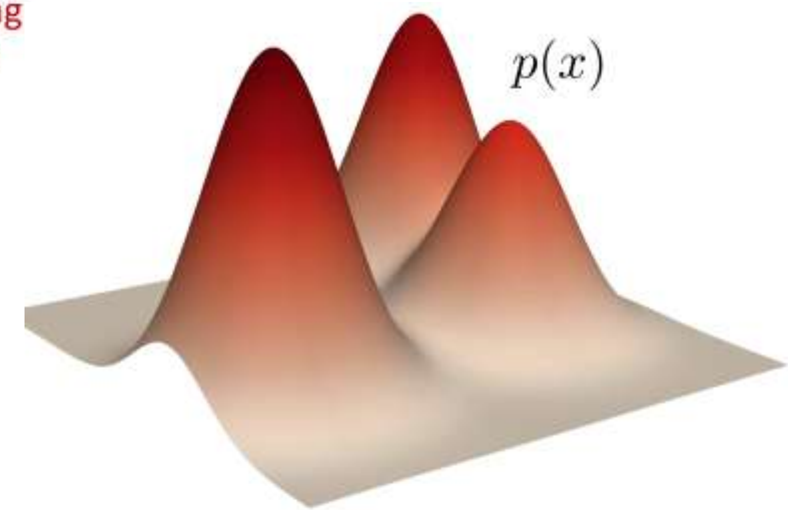
Scores function for Energy-based Models

- Define a scalar function, called “energy”.
- At inference time, find x that minimizes energy
- We can use an energy to model a probability distribution

$$p(x) = \frac{\exp(-E(x))}{Z}$$

normalizing
constant





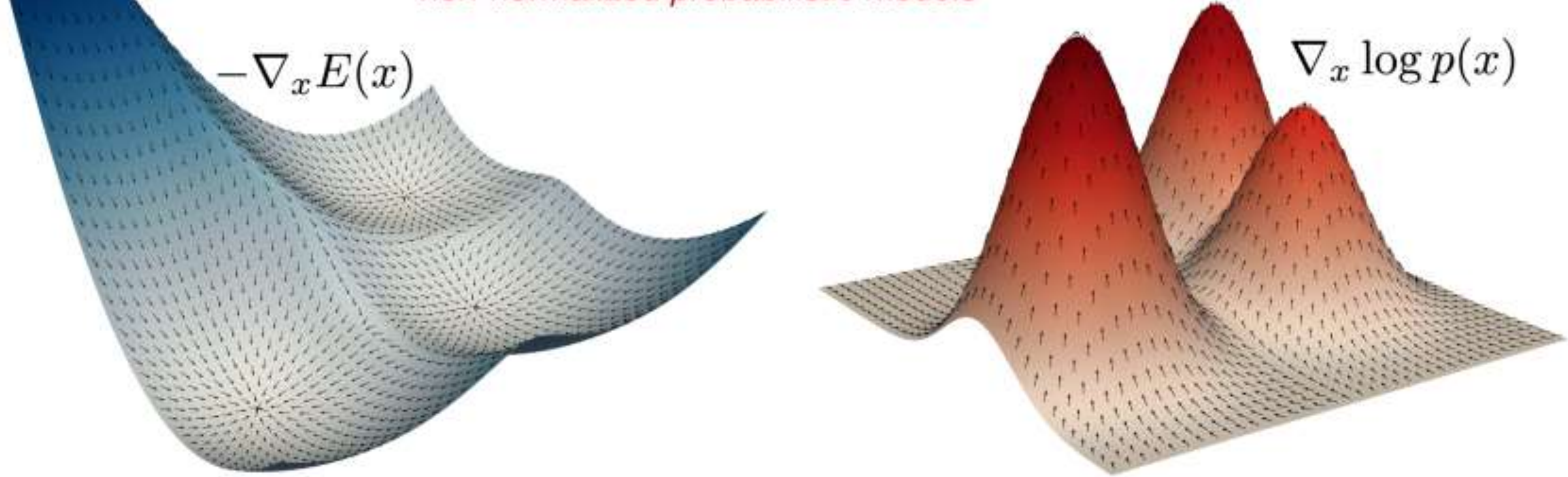
From 6.S978 Fall 2024, EECS, MIT

Scores function for Energy-based Models

- "Score function": gradient of log-probability

$$\nabla_x \log p(x) = -\nabla_x E(x)$$

"non-normalized probabilistic models"

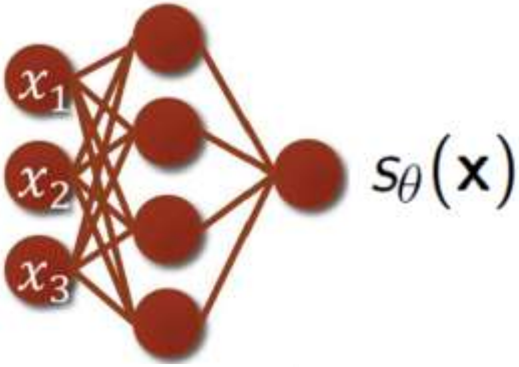


From 6.S978 Fall 2024, EECS, MIT

Score matching

- Score function:

$$s_\theta(\mathbf{x}) := \nabla_\mathbf{x} \log p_\theta(\mathbf{x})$$



- Fisher divergence between $p(\mathrm{x})$ and $q(\mathrm{x})$:

$$D_F(p, q) := \frac{1}{2} E_{\mathbf{x} \sim p}[\|\nabla_\mathbf{x} \log p(\mathbf{x}) - \nabla_\mathbf{x} \log q(\mathbf{x})\|_2^2]$$

- Score matching:

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}}[\|\nabla_\mathbf{x} \log p_{\text{data}}(\mathbf{x}) - s_\theta(\mathbf{x})\|_2^2]$$

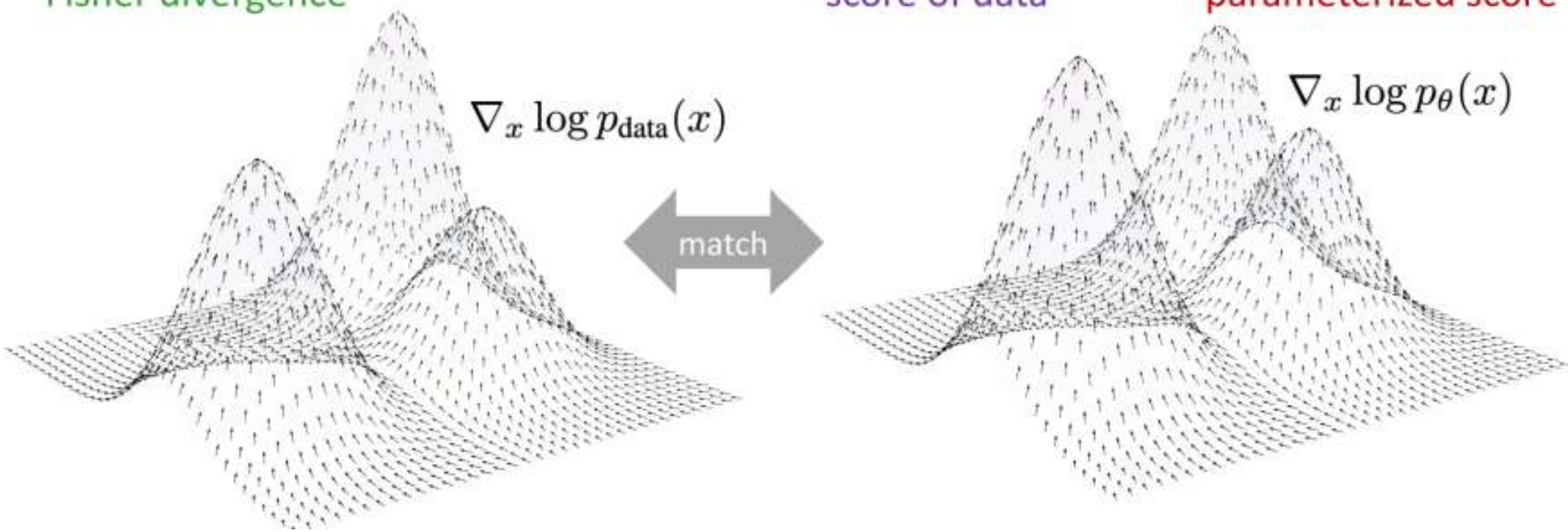
- What if data score is undefined? Denoising score matching

$$\frac{1}{2} E_{\tilde{\mathbf{x}} \sim q_\sigma}[\|\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) - \boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2]$$

Score matching

- Instead of parametrizing *p*, we can parametrize the score

$$\underbrace{D_F(p_{\text{data}}(x) \parallel p_\theta(x))}_{\text{Fisher divergence}} = \mathbb{E}_{p_{\text{data}}(x)}\left[\frac{1}{2}\left\|\underbrace{\nabla_x \log p_{\text{data}}(x)}_{\text{score of data}} - \underbrace{\nabla_x \log p_\theta(x)}_{\text{parameterized score}}\right\|^2\right]$$



From 6.S978 Fall 2024, EECS, MIT

Denoising Score Matching

- Matching the score of a noise-perturbed distribution

$\mathbf{x}$

$$p_{\text{data}}(\mathbf{x}) \xrightarrow{} q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \xrightarrow{} q_\sigma(\tilde{\mathbf{x}})$$

$\tilde{\mathbf{x}}$

$$\frac{1}{2} E_{\tilde{\mathbf{x}} \sim q_\sigma}[\|\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) - \boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2]$$

$$= \frac{1}{2} \int q_\sigma(\tilde{\mathbf{x}}) \, \|\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) - \boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2 \, \mathrm{d}\tilde{\mathbf{x}}$$

$$= \frac{1}{2} \int q_\sigma(\tilde{\mathbf{x}}) \, \|\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})\|_2^2 \, \mathrm{d}\tilde{\mathbf{x}} + \frac{1}{2} \int q_\sigma(\tilde{\mathbf{x}}) \, \|\boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2 \, \mathrm{d}\tilde{\mathbf{x}}$$

$$- \int q_\sigma(\tilde{\mathbf{x}}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}}$$

$$= \text{const.} + \frac{1}{2} E_{\tilde{\mathbf{x}} \sim q_\sigma}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2] - \int q_\sigma(\tilde{\mathbf{x}}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})^\mathsf{T} \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}}$$

Denoising Score Matching

- Matching the score of a noise-perturbed distribution

$\mathbf{x}$

$$p_{\text{data}}(\mathbf{x}) \downarrow q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \downarrow q_\sigma(\tilde{\mathbf{x}})$$

$\tilde{\mathbf{x}}$

$$\begin{aligned}
& -\int q_\sigma(\tilde{\mathbf{x}}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}} \\
&= -\int q_\sigma(\tilde{\mathbf{x}}) \frac{1}{q_\sigma(\tilde{\mathbf{x}})} \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}} \\
&= -\int \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}})^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}} \\
&= -\int \nabla_{\tilde{\mathbf{x}}} \left(\int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \, \mathrm{d}\mathbf{x} \right)^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}} \\
&= -\int \left(\int p_{\text{data}}(\mathbf{x}) \nabla_{\tilde{\mathbf{x}}} q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \, \mathrm{d}\mathbf{x} \right)^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}} \\
&= -\int \left(\int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \, \mathrm{d}\mathbf{x} \right)^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}} \\
&= -\iint p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\mathbf{x} \, \mathrm{d}\tilde{\mathbf{x}} \\
&= -E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})} [\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}})]
\end{aligned}$$

Denoising Score Matching

- Matching the score of a noise-perturbed distribution

$$\tilde{x} := x + \epsilon$$

$\mathbf{x}$

$$p_{\text{data}}(\mathbf{x}) \longrightarrow q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \longrightarrow q_\sigma(\tilde{\mathbf{x}})$$

$\tilde{\mathbf{x}}$

$$\frac{1}{2} E_{\tilde{\mathbf{x}} \sim q_\sigma}[\|\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) - \boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2]$$

$$= \text{const.} + \frac{1}{2} E_{\tilde{\mathbf{x}} \sim q_\sigma}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2] - \int q_\sigma(\tilde{\mathbf{x}}) \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) \, \mathrm{d}\tilde{\mathbf{x}}$$

$$= \text{const.} + \frac{1}{2} E_{\tilde{\mathbf{x}} \sim q_\sigma}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}})\|_2^2] - E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}[\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})^\intercal \boldsymbol{s}_\theta(\tilde{\mathbf{x}})]$$

$$= \text{const.} + \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2]$$

$$- \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}[\|\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2]$$

$$= \text{const.} + \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2] + \text{const.}$$

$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2] + \text{const.}$$

Denoising Score Matching

- Matching the score of a noise-perturbed distribution

$$\frac{1}{2}E_{\tilde{\mathbf{x}}\sim p_{\text{data}}}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}})\|_2^2]$$
$$=\frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x}),\tilde{\mathbf{x}}\sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}[\|\boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\mid\mathbf{x})\|_2^2] + \text{const}$$

- How to calculate $\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}_i \mid \mathbf{x}_i)$?

$$q_\sigma(\tilde{\mathbf{x}}\mid\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}\mid\mathbf{x}, \sigma^2\boldsymbol{I})$$ **Gaussian perturbation**

$$\nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}\mid\mathbf{x}) = -\frac{\tilde{\mathbf{x}}-\mathbf{x}}{\sigma^2}$$

- σ need to be small enough such that $q_\sigma(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$
- How to optimize?
  - Sample a minibatch
  - Stochastic gradient descent

$$\frac{1}{2n}\sum_{i=1}^{n}\left[\left\|\boxed{\boldsymbol{s}_\theta(\tilde{\mathbf{x}}_i)} + \frac{\tilde{\mathbf{x}}_i-\mathbf{x}_i}{\sigma^2}\right\|_2^2\right]$$

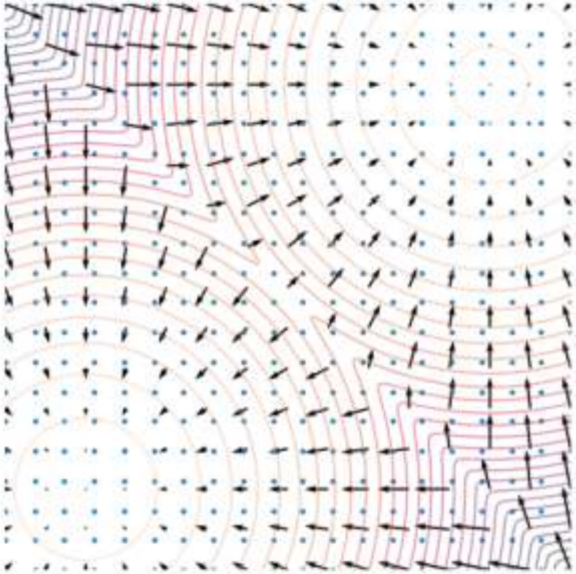
How to sample: Langevin Dynamics

- Suppose we trained a score-based model $s_\theta(\mathrm{x}) = \nabla_{\mathrm{x}} \log p(\mathrm{x})$.
  How can we draw a sample from the distribution $p(\mathrm{x})$?

- **Langevin dynamics**
  - Sample from $p(\mathrm{x})$ using only the score $\nabla_{\mathrm{x}} \log p(\mathrm{x})$
  - Process:
    - Initialize from a prior distribution $\mathrm{x}^0 \sim \pi(\mathrm{x})$
    - Repeat for $t = 1, 2, \dots, T$

$$\mathbf{z}^t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

$$\mathbf{x}^t \leftarrow \mathbf{x}^{t-1} + \frac{\epsilon}{2} \nabla_{\mathbf{x}} \log p(\mathbf{x}^{t-1}) + \sqrt{\epsilon}\, \mathbf{z}^t$$



  - If $\epsilon \to 0$ and $T \to \infty$, we will have $\mathrm{x}^T \sim p(\mathrm{x})$

How to sample: Langevin Dynamics

- Given a score function, we can sample *x* from *p* by iterating:

$$x_t \leftarrow x_{t-1} + \frac{\sigma^2}{2} \underline{\underline{\nabla_x \log p_\theta(x_{t-1})}} + \sigma z_t$$

step size

score function

(don't need to know p)

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

(neg) gradient of energy

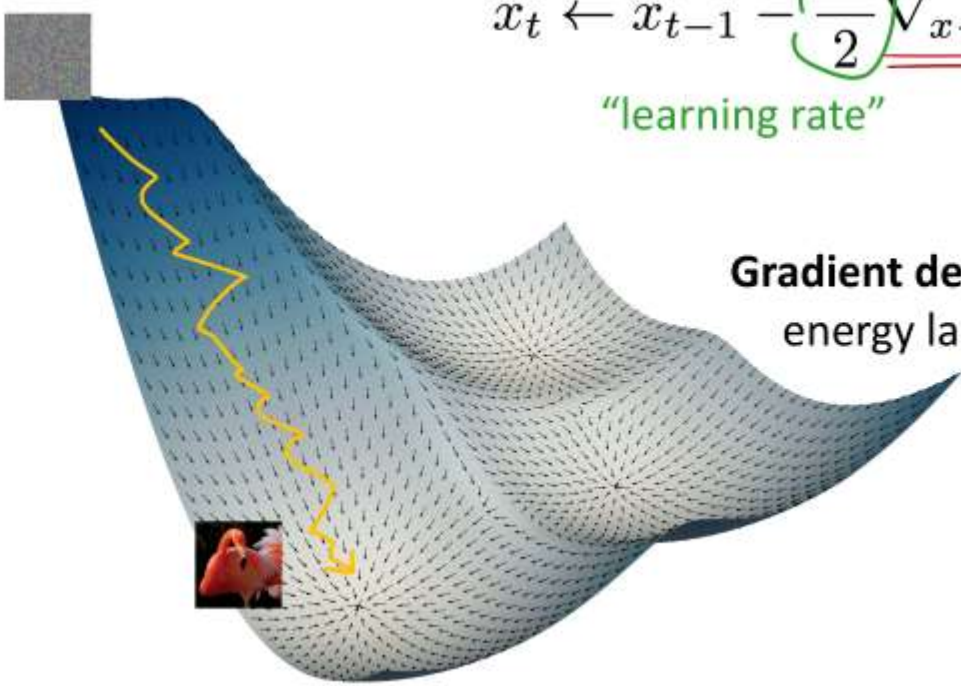
$$-\nabla_x E_\theta(x_{t-1})$$

From 6.S978 Fall 2024, EECS, MIT

How to sample: Langevin Dynamics

- Given a score function, we can sample *x* from *p* by iterating:

$$x_t \leftarrow x_{t-1} - \underbrace{\frac{\sigma^2}{2}}_{\text{“learning rate”}} \underbrace{\nabla_x E_\theta(x_{t-1})}_{\text{gradient}} + \underbrace{\sigma z_t}_{\text{perturbation}}$$



From 6.S978 Fall 2024, EECS, MIT

(Recap) Diffusion algorithm

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \underbrace{\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)}_{\text{score function}} \right\|^2$$

6: **until** converged

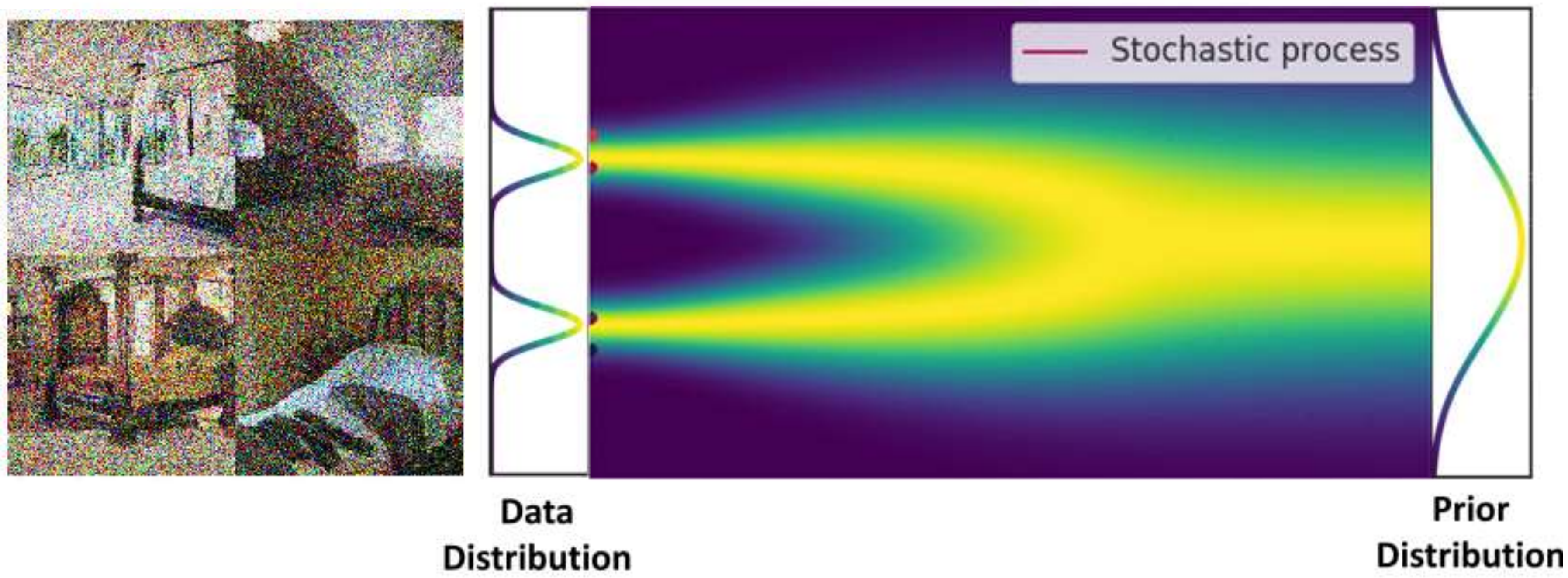
Algorithm 2 Sampling

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: $\underbrace{\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \underbrace{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}_{\text{score function}}\right) + \sigma_t \mathbf{z}}_{\text{Langevin Dynamics}}$
5: **end for**
6: **return** $\mathbf{x}_0$

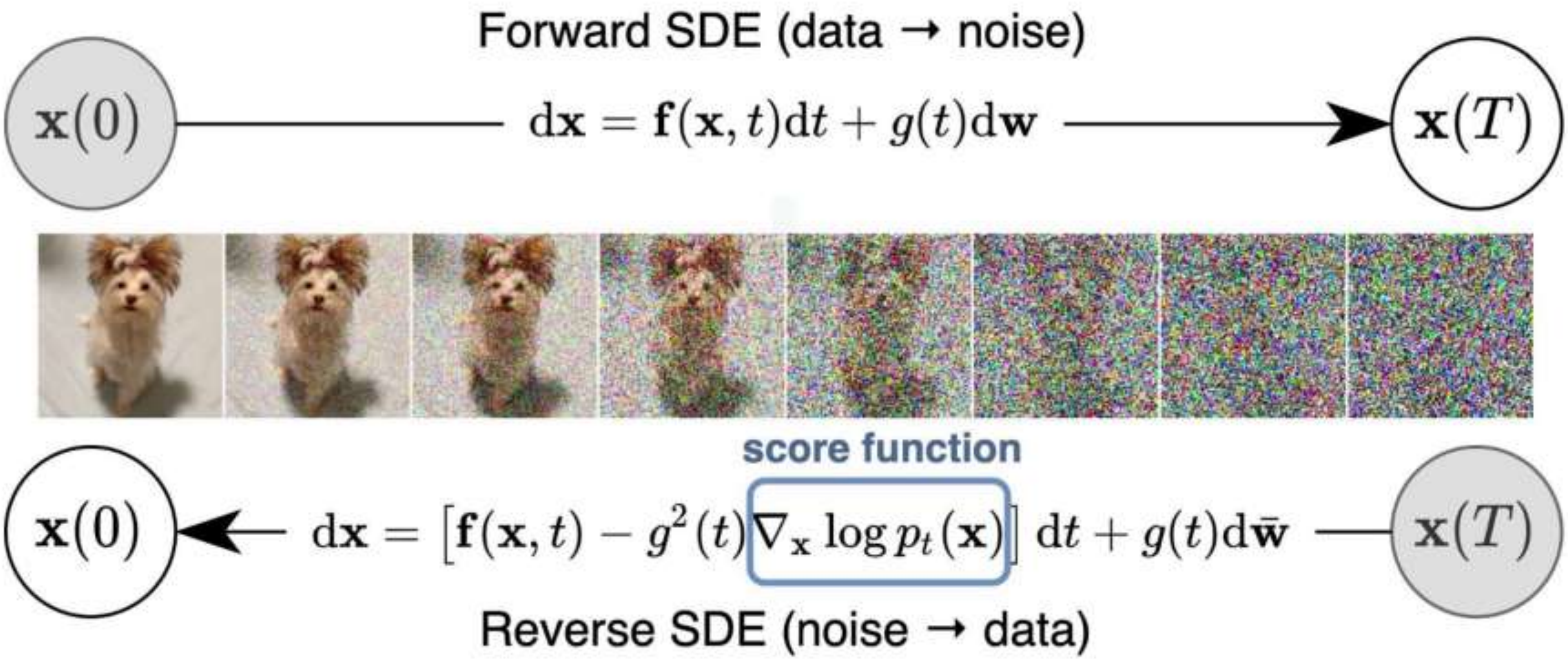
From 6.S978 Fall 2024, EECS, MIT

Score-based diffusion models via SDE

- Perturb data distribution with stochastic differential equations (SDEs)



Score-based diffusion models via SDE



Song et al, Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021

Score-based diffusion models via SDE

- Stochastic differential equation

$$\mathrm{d}\mathbf{x} = \underbrace{\boxed{\mathbf{f}(\mathbf{x}, t)\mathrm{d}t}}_{\textbf{Deterministic drift}} + \sigma(t)\underbrace{\boxed{\mathrm{d}\mathbf{w}}}_{\textbf{Infinitesimal white noise}}$$

- Forward SDE:

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

- Reverse-time SDE:

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\overbrace{\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}}^{\textbf{score function}}\right] \mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$$

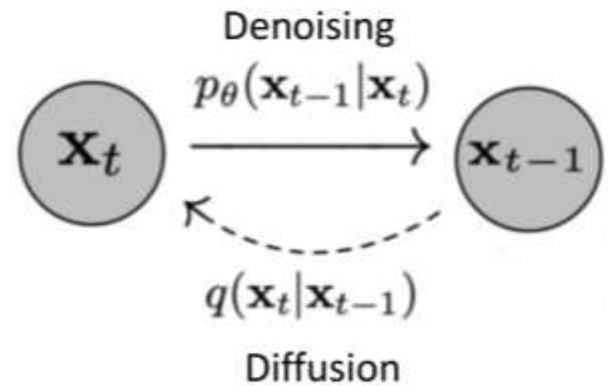
Sample from the reverse SDE

- Approximate the reverse SDE with the learned score-based model
$$d\mathbf{x} = [\,\boldsymbol{f}(\mathbf{x}, t) - g^2(t) s_\theta(\mathbf{x}, t)]dt + g(t)d\mathbf{w}$$

- Using the numerical SDE solvers. (**Euler-Maruyama**)
  - Initialize:
    - $t \leftarrow \mathrm{T}, \quad \mathbf{x} \sim p_T(\mathbf{x})$
  - Repeat:
    - $z \sim N(0, |\Delta t| I)$
    - $\Delta \mathrm{x} = [\,f(\mathrm{x}, t) - g^2(t) s_\theta(\mathrm{x}, t)]dt + g(t)\mathrm{z}$
    - $\mathrm{x} \leftarrow \mathrm{x} + \Delta \mathrm{x}$
    - $t \leftarrow t + \Delta \mathrm{t}$
  - Until $t = 0$

DDPM forward diffusion process as SDE

- Consider the diffusion process in **infinitesimal** step



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

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\,\mathbf{x}_{t-1} + \sqrt{\beta_t}\,\mathcal{N}(\mathbf{0},\mathbf{I}) \\
&= \sqrt{1-\beta(t)\Delta t}\,\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0},\mathbf{I}) \\
&\approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0},\mathbf{I})
\end{aligned}$$

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

Stochastic Differential Equation

DDPM forward diffusion process as SDE

- Forward diffusion SDE

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

- Reverse generative SDE

$$\mathrm{d}\mathbf{x}_t = \overbrace{\left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\underbrace{\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)}_{\text{“Score Function”}}\right]}^{\text{drift term}}\mathrm{d}t + \overbrace{\sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t}^{\text{diffusion term}}$$

Training DDPM via denoising score matching

- Consider the diffusion process in **infinitesimal** step

- Objective function

$$\min_{\theta} \underbrace{\mathbb{E}_{t\sim\mathcal{U}(0,T)}}_{\text{diffusion time } t} \underbrace{\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}}_{\text{data sample } \mathbf{x}_0} \underbrace{\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)}}_{\text{diffused data sample } \mathbf{x}_t} ||\underbrace{\mathbf{s}_\theta(\mathbf{x}_t, t)}_{\text{neural network}} - \underbrace{\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{x}_0)}_{\text{score of diffused data sample}}||_2^2$$

- Re-parameterized sampling:

$$\mathbf{x}_t = \gamma_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- Score function

$$\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{x}_0) = -\nabla_{\mathbf{x}_t} \frac{(\mathbf{x}_t - \gamma_t \mathbf{x}_0)^2}{2\sigma_t^2}$$

$$= -\frac{\mathbf{x}_t - \gamma_t \mathbf{x}_0}{\sigma_t^2} = -\frac{\gamma_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon} - \gamma_t \mathbf{x}_0}{\sigma_t^2} = -\frac{\boldsymbol{\epsilon}}{\sigma_t}$$

"Variance Preserving" SDE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

$$q_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \gamma_t \mathbf{x}_0, \sigma_t^2 \mathbf{I})$$

$$\gamma_t = e^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

$$\sigma_t^2 = 1 - e^{-\int_0^t \beta(s)ds}$$

- Can be implemented as noise prediction:

$$\mathbf{s}_\theta(\mathbf{x}_t, t) := -\frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\sigma_t} \quad \Rightarrow \quad \min_{\theta} \mathbb{E}_{t\sim\mathcal{U}(0,T)} \mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \frac{1}{\sigma_t^2} ||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)||_2^2$$

A quick and cute video summary

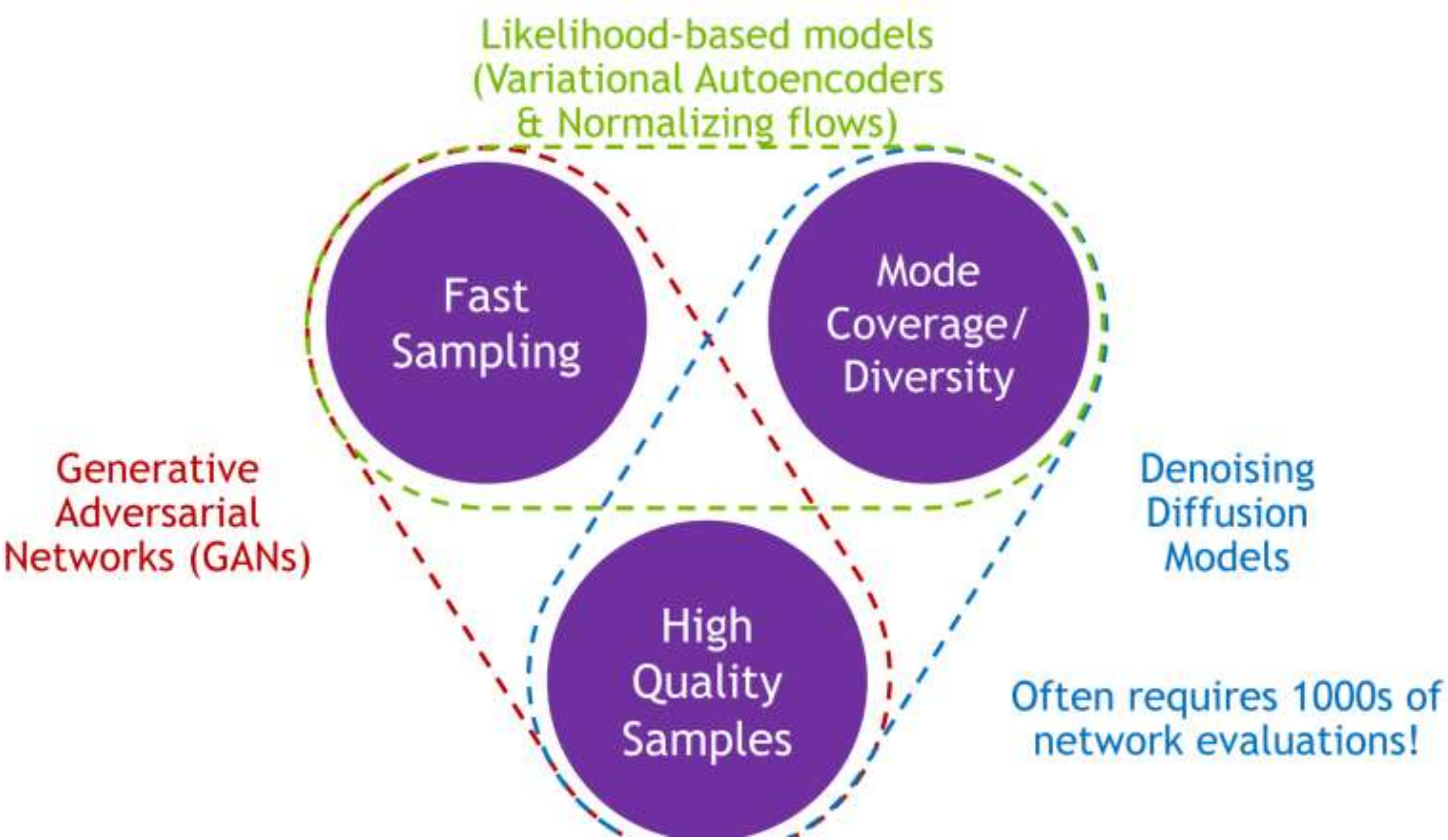
PROMPT:

https://www.youtube.com/watch?v=i2qSxMVeVLI

Acceleration of Diffusion Models

What makes a good generative model?

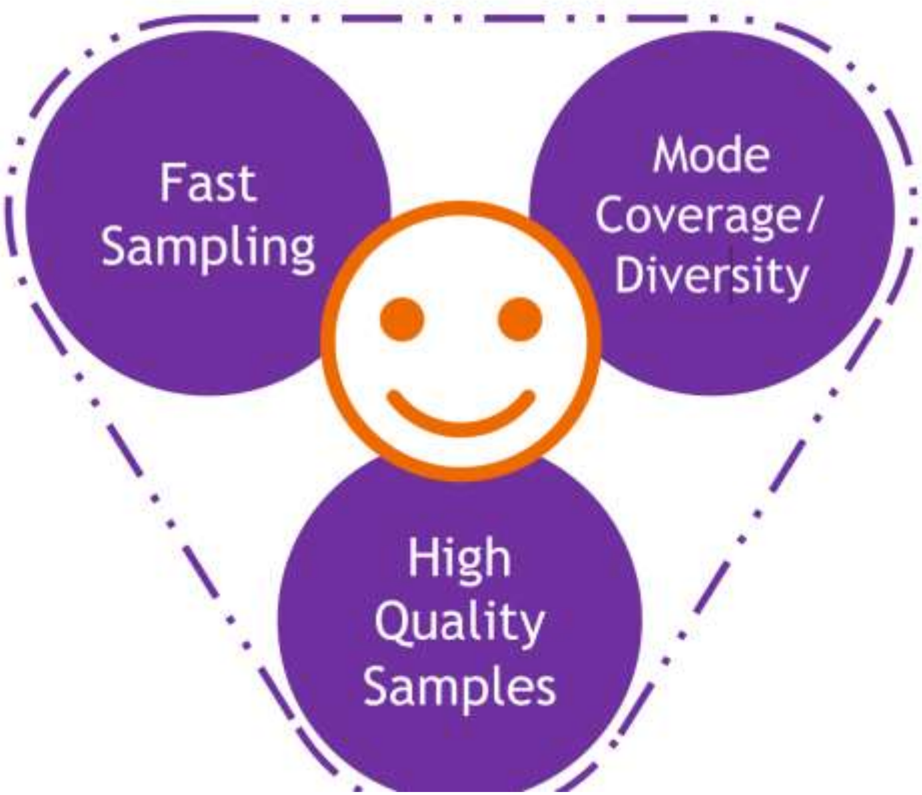
- The generative learning trilemma



Tackling the Generative Learning Trilemma with Denoising Diffusion GANs, ICLR 2022

What makes a good generative model?

- The generative learning trilemma
- Tackle the trilemma by accelerating diffusion models



Tackling the Generative Learning Trilemma with Denoising Diffusion GANs, ICLR 2022

How to accelerate diffusion models?

- A quick and cute video summary

Song et al., “Denoising Diffusion Implicit Models” (DDIM), ICLR 2021.

Denoising diffusion implicit models (DDIM)

- DDPM objective:

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q\left[\underbrace{\cancel{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\,\|\,p(\mathbf{x}_T))}}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\,\|\,p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{\cancel{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}}_{L_0}\right]$$

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

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar\alpha_t}\mathbf{x}_0, (1-\bar\alpha_t)\mathbf{I}))$$ (make sure $\mathbf{x}_t = \sqrt{\bar\alpha_t}\,\mathbf{x}_0 + \sqrt{(1-\bar\alpha_t)}\,\epsilon$)

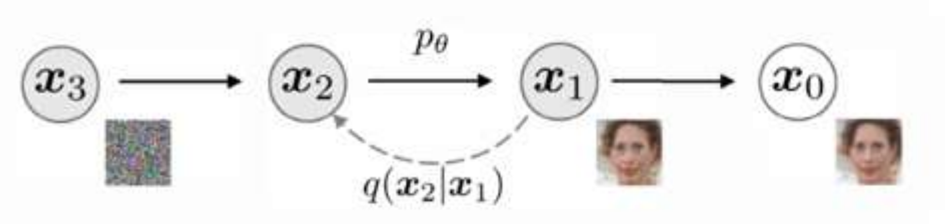
Forward process: $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde\mu_t(\mathbf{x}_t,\mathbf{x}_0), \tilde\sigma_t^2\mathbf{I}),\quad \tilde\mu_t(\mathbf{x}_t,\mathbf{x}_0) = a\mathbf{x}_t + b\epsilon = a\mathbf{x}_t + b\dfrac{\mathbf{x}_t - \sqrt{\bar\alpha_t}\mathbf{x}_0}{\sqrt{1-\bar\alpha_t}}$

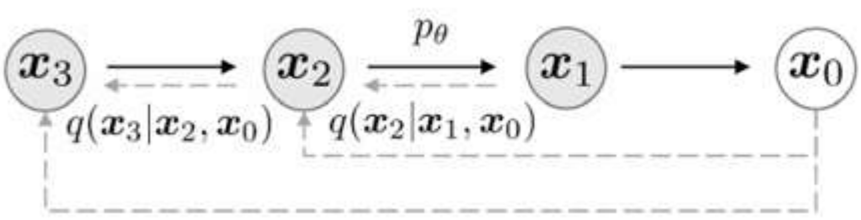
Reverse process: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t,t), \tilde\sigma_t^2\mathbf{I}),\quad \mu_\theta(\mathbf{x}_t,t) = a\mathbf{x}_t + b\epsilon_\theta(\mathbf{x}_t,t) = a\mathbf{x}_t + b\dfrac{\mathbf{x}_t - \sqrt{\bar\alpha_t}\hat{\mathbf{x}}_0}{\sqrt{1-\bar\alpha_t}}$

- No need to specify $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ to be a Markovian process!

Denoising diffusion implicit models (DDIM)

- Design a family of non-Markovian diffusion processes and corresponding reverse processes



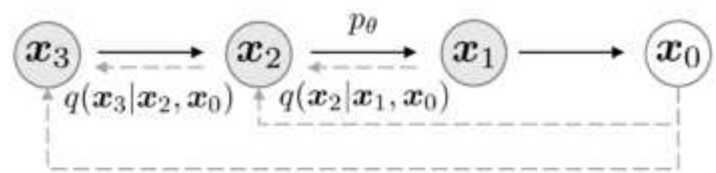


- Therefore, we can take a pre-trained diffusion model but with more choices in the sampling procedure.

Denoising diffusion implicit models (DDIM)

- Non-Markovian diffusion processes:

$$q_\sigma(x_{1:T}|x_0) := q_\sigma(x_T|x_0)\prod_{t=2}^{T} q_\sigma(x_{t-1}|x_t, x_0)$$



$$q_\sigma(x_T|x_0) \sim \mathcal{N}(\sqrt{\bar\alpha_T}\, x_0, (1-\bar\alpha_T)I)$$

Gaussian Distribution Additive Property

$$\mathcal{N}(0, \delta_1^2) + \mathcal{N}(0, \delta_2^2) = \mathcal{N}(0, \delta_1^2 + \delta_2^2)$$

$$x_{t-1} = \sqrt{\dot\alpha_{t-1}}x_0 + \sqrt{1-\dot\alpha_{t-1}}\epsilon_{t-1}$$

$$\sqrt{1-\bar\alpha_t-\delta_t^2}\,\epsilon_t \sim \mathcal{N}(0, 1-\bar\alpha_t-\delta_t^2)$$

$$\delta_t\epsilon \sim \mathcal{N}(0, \delta_t^2)$$

$$\sqrt{1-\bar\alpha}\,\epsilon_{t-1} \sim \mathcal{N}(0, 1-\bar\alpha_{t-1})$$

$$\begin{aligned} x_{t-1} &= \sqrt{\dot\alpha_{t-1}}x_0 + \sqrt{1-\dot\alpha_{t-1}}\epsilon_{t-1} \\ &= \sqrt{\bar\alpha_{t-1}}x_0 + \sqrt{1-\bar\alpha_{t-1}-\sigma_t^2}\epsilon_t + \sigma_t\epsilon \\ &= \sqrt{\bar\alpha_{t-1}}x_0 + \sqrt{1-\bar\alpha_{t-1}-\sigma_t^2}\frac{x_t-\sqrt{\bar\alpha_t}x_0}{\sqrt{1-\bar\alpha_t}} + \sigma_t\epsilon \end{aligned}$$

$$\boxed{\mathbf{x}_t = \sqrt{\bar\alpha_t}\,\mathbf{x}_0 + \sqrt{(1-\bar\alpha_t)}\,\epsilon}$$

Denoising diffusion implicit models (DDIM)

- Different "blending" selections during reversion:

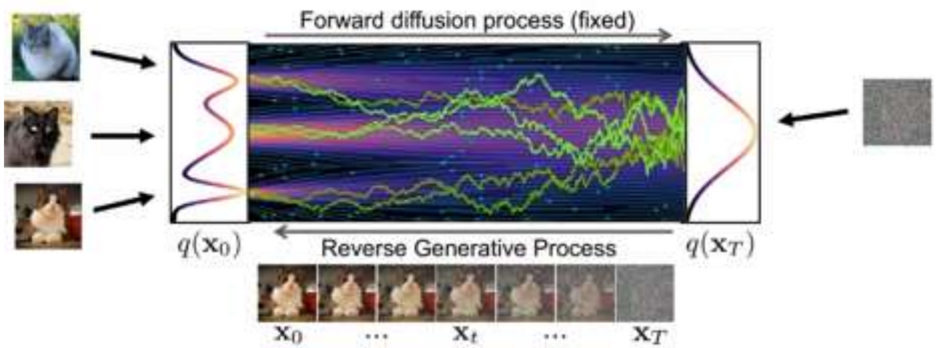
$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\frac{x_t-\sqrt{\bar{\alpha}_t}\,\hat{x}_0}{\sqrt{1-\bar{\alpha}_t}} + \sigma_t\epsilon_t^*$$

$$= \underbrace{\sqrt{\bar{\alpha}_{t-1}}\left(\frac{x_t-\sqrt{1-\bar{\alpha}_t}\,\hat{\epsilon}_t(x_t,t)}{\sqrt{\bar{\alpha}_t}}\right)}_{\text{predict } x_0} + \underbrace{\sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\quad\hat{\epsilon}_t(x_t,t)}_{\text{direction pointing to } x_t} + \underbrace{\sigma_t\epsilon_t^*}_{\text{random noise}}$$

$$\text{where}\quad \epsilon_t^* \sim \mathcal{N}(0,I)$$

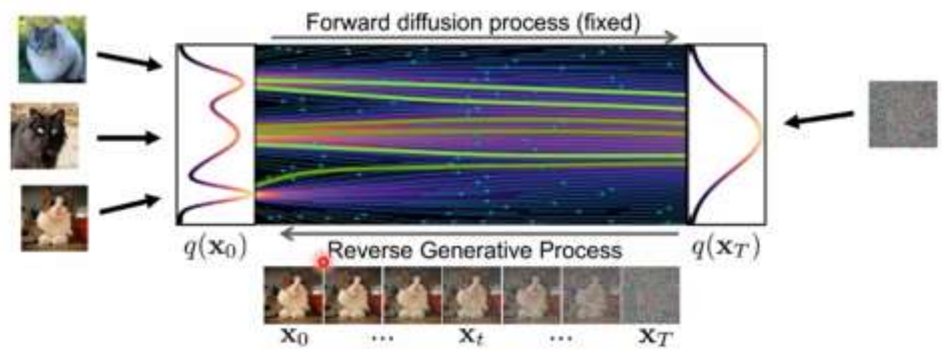
DDPM

$$\sigma^2 = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$$



DDIM

$$\sigma^2 = 0$$

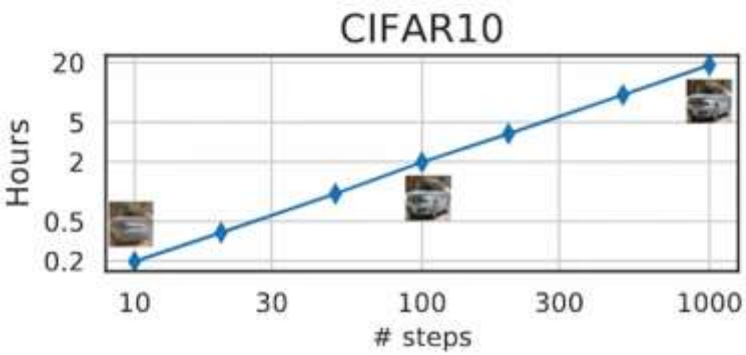


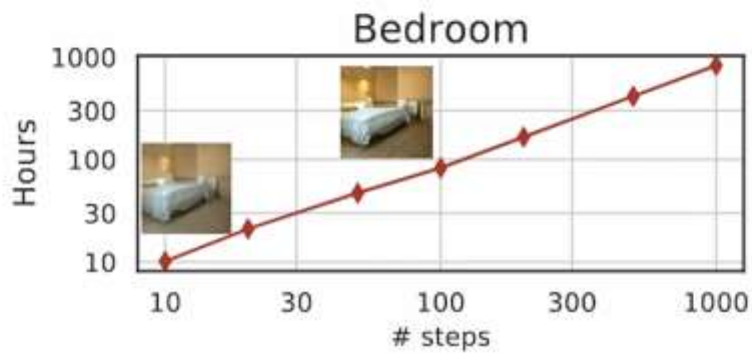
Denoising diffusion implicit models (DDIM)

- Experimental results of DDIM

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

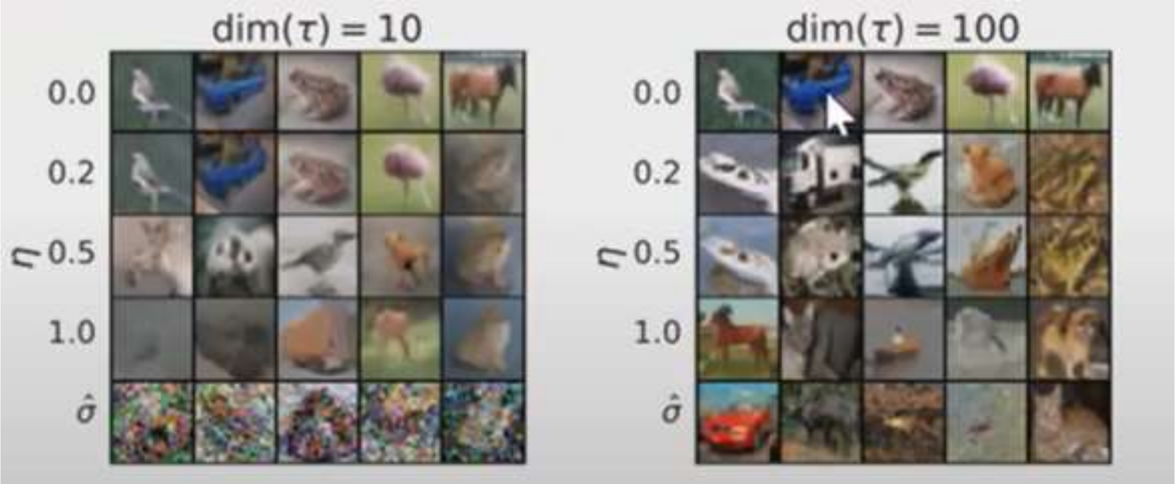
	S	CIFAR10 (32 × 32) 10	20	50	100	1000	CelebA (64 × 64) 10	20	50	100	1000
η	0.0	**13.36**	**6.84**	**4.67**	**4.16**	4.04	**17.33**	**13.73**	**9.17**	**6.53**	3.51
	0.2	14.04	7.11	4.77	4.25	4.09	17.66	14.11	9.51	6.79	3.64
	0.5	16.66	8.35	5.25	4.46	4.29	19.86	16.06	11.01	8.09	4.28
	1.0	41.07	18.36	8.01	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**

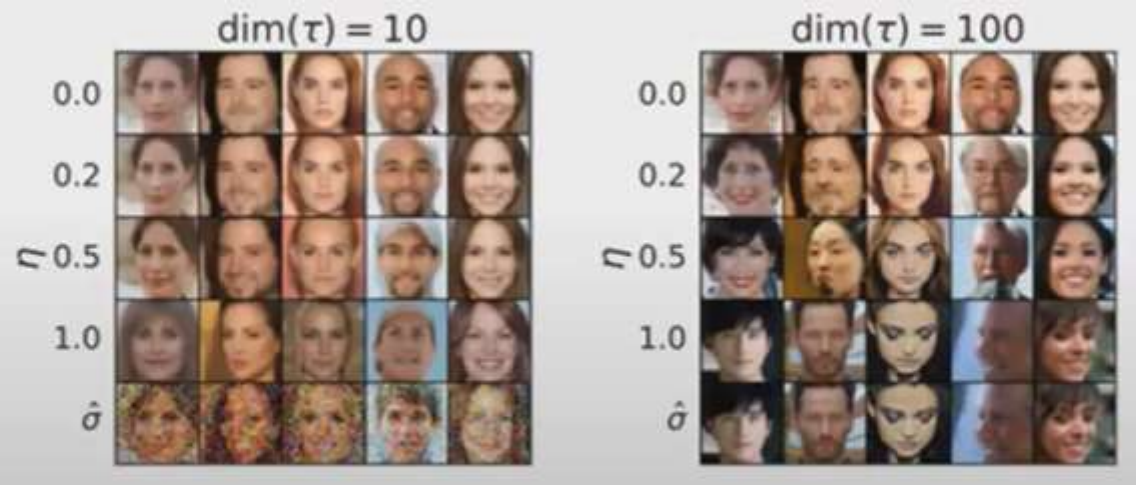




Denoising diffusion implicit models (DDIM)

- Experimental results of DDIM





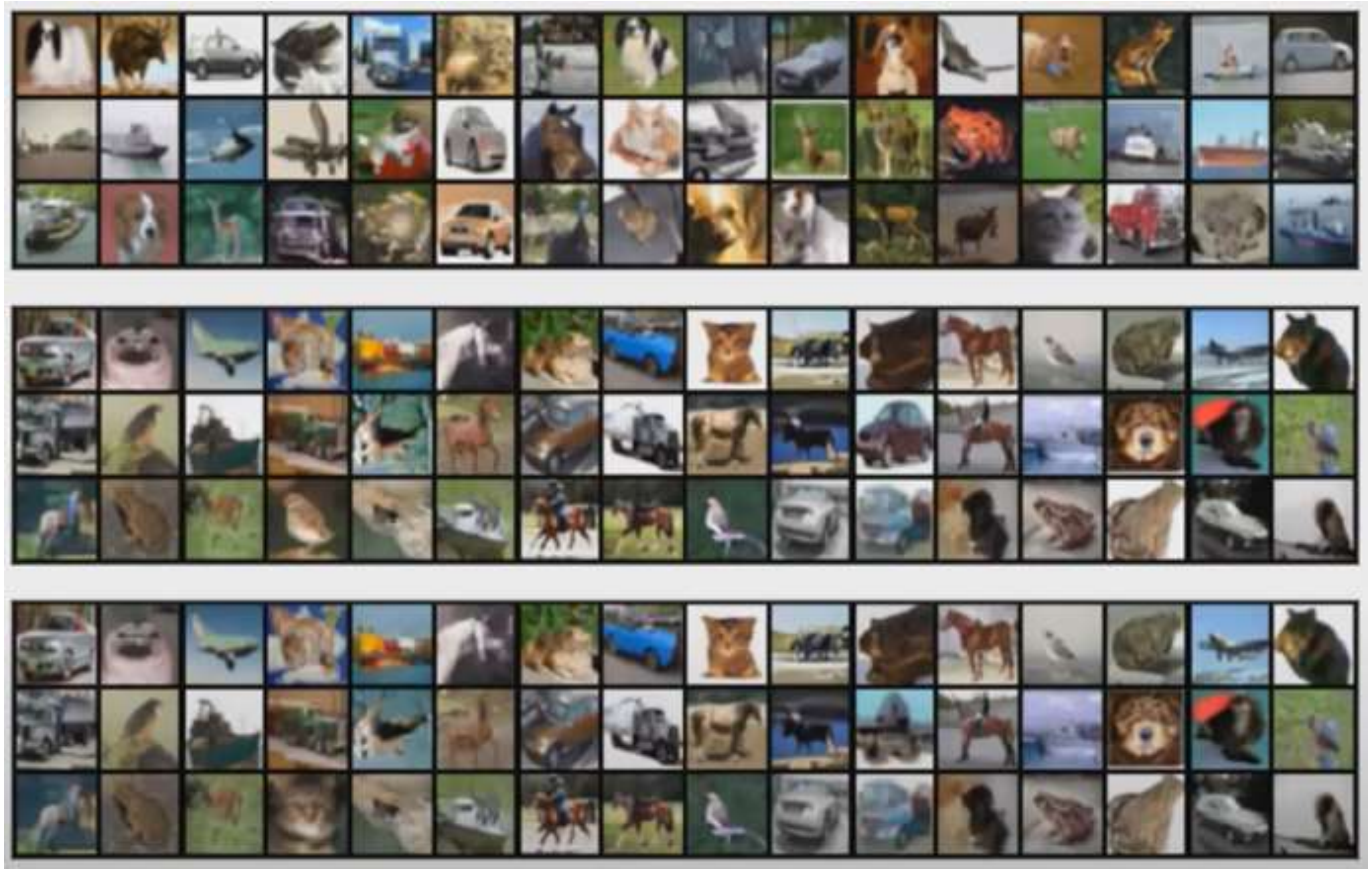
Denoising diffusion implicit models (DDIM)

- Experimental results of DDIM

DDPM
1000 steps

DDIM
1000 steps

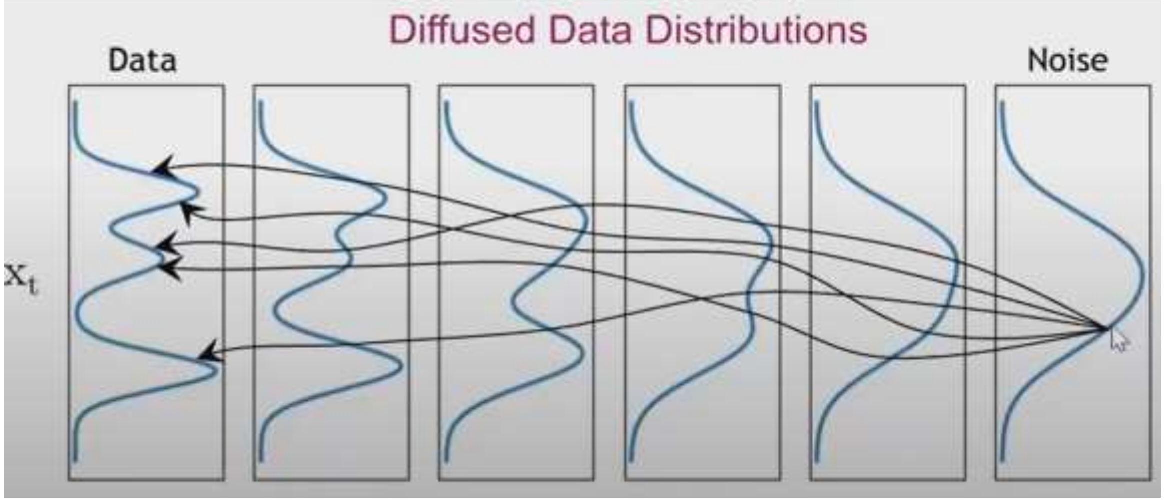
DDIM
100 steps



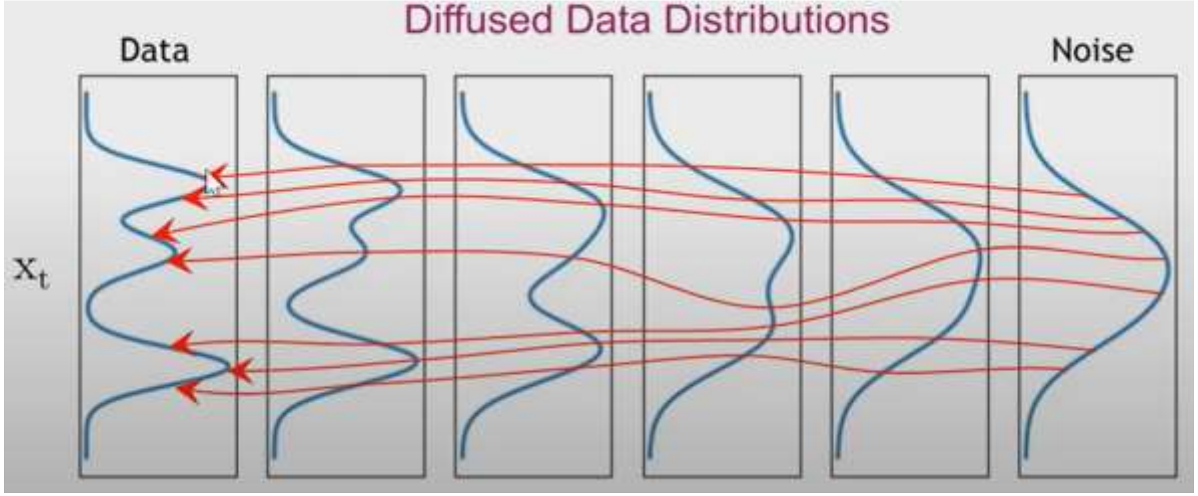
Denoising diffusion implicit models (DDIM)

- DDPM V.S. DDIM

DDPM



DDIM



Summary and Resources

- Deep Unsupervised Learning using Nonequilibrium Thermodynamics: https://arxiv.org/pdf/1503.03585.pdf
- Denoising Diffusion Probabilistic Models: https://arxiv.org/pdf/2006.11239.pdf
- Improved Denoising Diffusion Probabilistic Models: https://arxiv.org/pdf/2102.09672.pdf
- Diffusion Models Beat GANs on Image Synthesis: https://arxiv.org/pdf/2105.05233.pdf
- Classifier-free Diffusion Guidance: https://arxiv.org/pdf/2207.12598.pdf
- High Resolution Image Synthesis with Latent Diffusion Models: https://arxiv.org/pdf/2112.10752.pdf
- Denoising Diffusion Implicit Models: https://arxiv.org/pdf/1503.03585.pdf
- Generative Modeling by Estimating Gradients of the Data Distribution: https://yang-song.net/blog/2021/score/
- Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions: https://arxiv.org/pdf/2209.11215.pdf

Summary and Resources

- Lillian Weng's Blog: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
- The Annotated Diffusion Model: https://huggingface.co/blog/annotated-diffusion
- The Illustrated Stable Diffusion: https://jalammar.github.io/illustrated-stable-diffusion/
- PyTorch implementation of the DDPM Unet: https://nn.labml.ai/diffusion/ddpm/unet.html
- Guidance: a cheat code for diffusion models: https://benanne.github.io/2022/05/26/guidance.html
- Understanding Diffusion Models: A Unified Perspective: https://arxiv.org/pdf/2208.11970.pdf