Lecture 18: Diffusion Model II: Score-based Methods

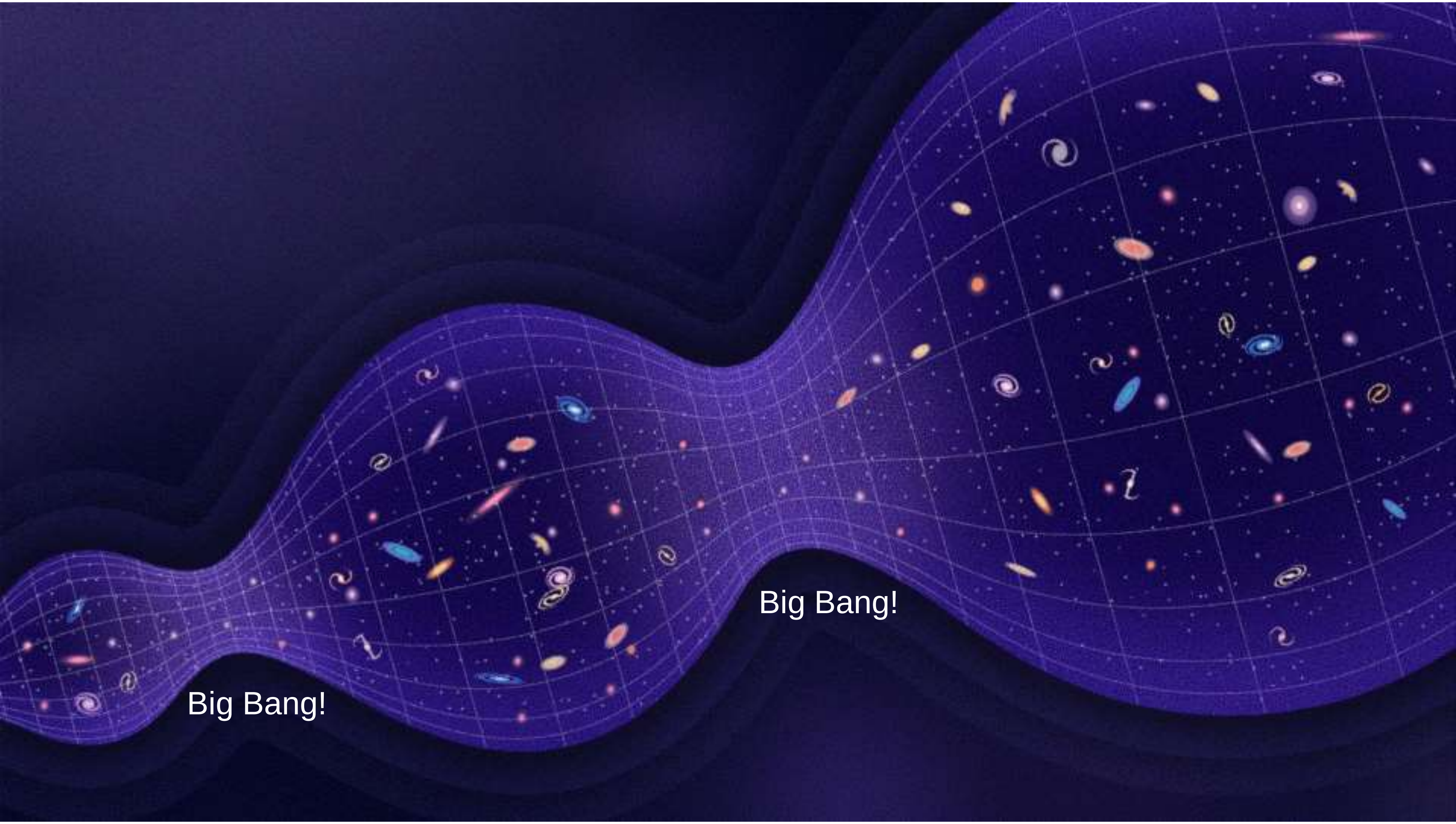
Lan Xu
SIST, ShanghaiTech
Fall, 2023

Outline

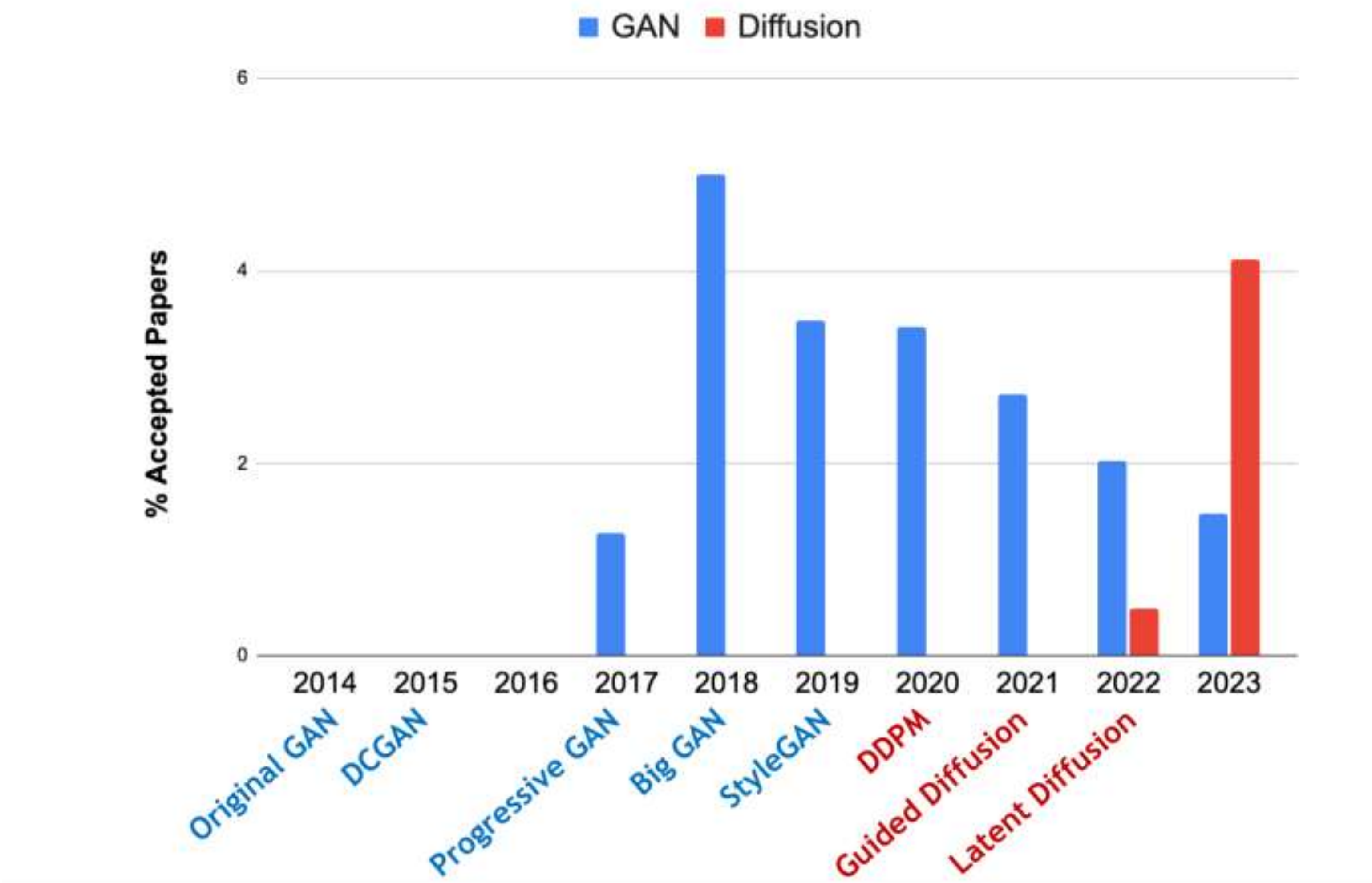
- Diffusion: Denoising Diffusion Probabilistic Models
- Score-based Diffusion Models
- Accelerated Sampling
- Conditional Generation and Guidance

The Landscape of Generative Models

Diffusion: A Generative Learning Big Bang



Diffusion: new trend in CVPR



Recall DDPM

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

Algorithm 1 Training

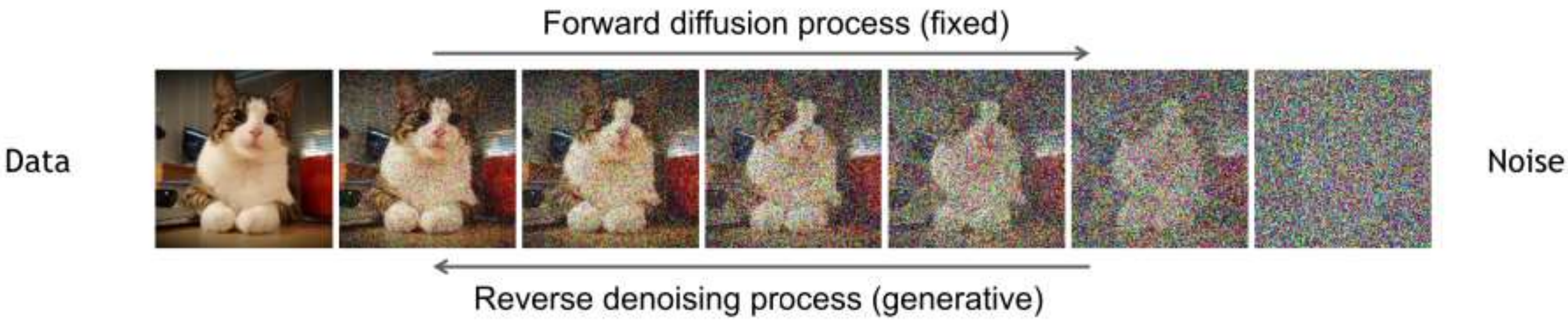
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

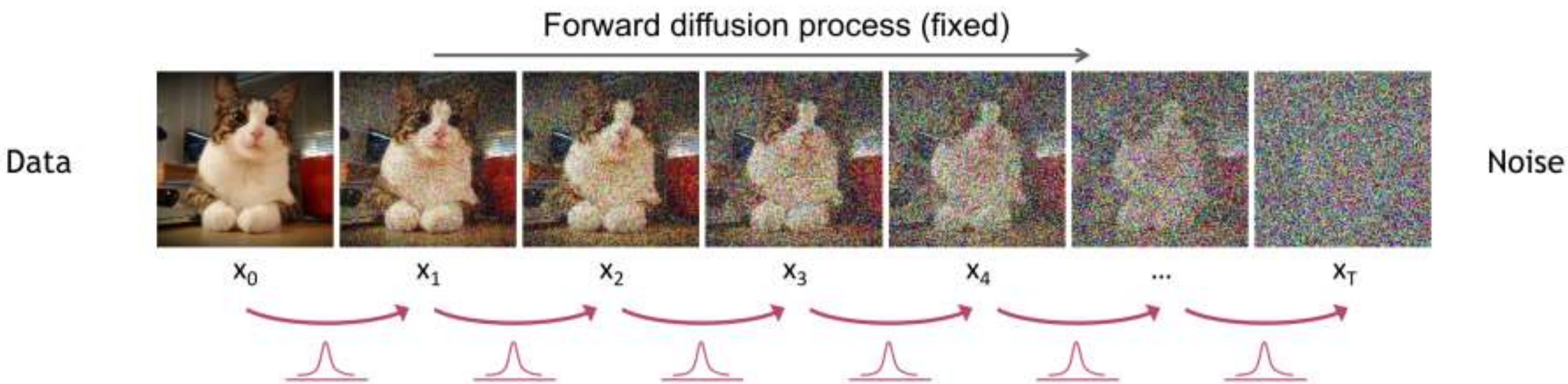
Recall DDPM

- Learning to generate by denoising
- **Forward diffusion** process that gradually adds noise to input
- **Reverse denoising** process that learns to generate data by denoising



Recall DDPM

- Forward Diffusion Process in T steps

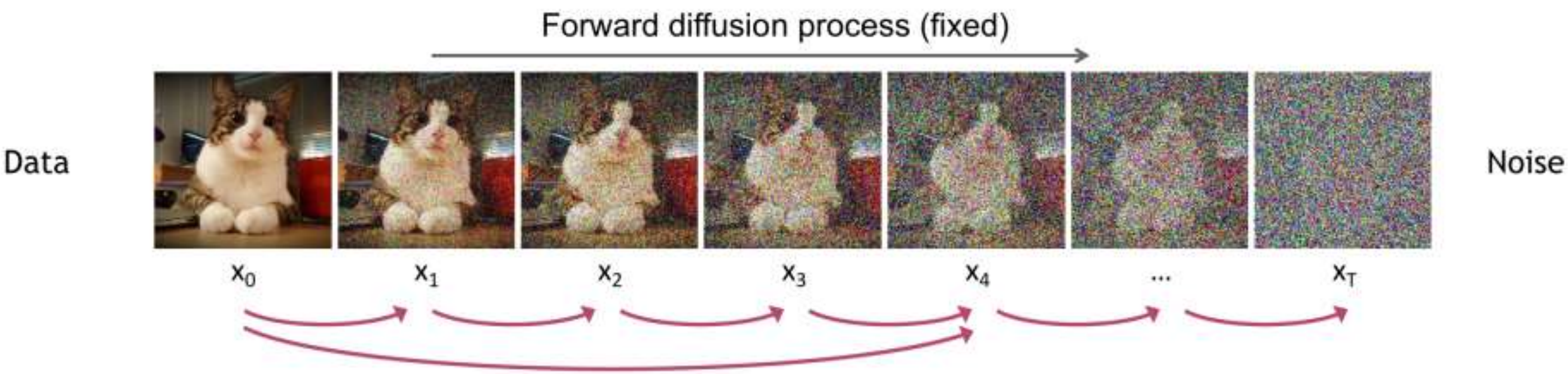


$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

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

Recall DDPM

- Diffusion Kernel and Reparametrization Trick



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

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

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

$$\Rightarrow$$

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

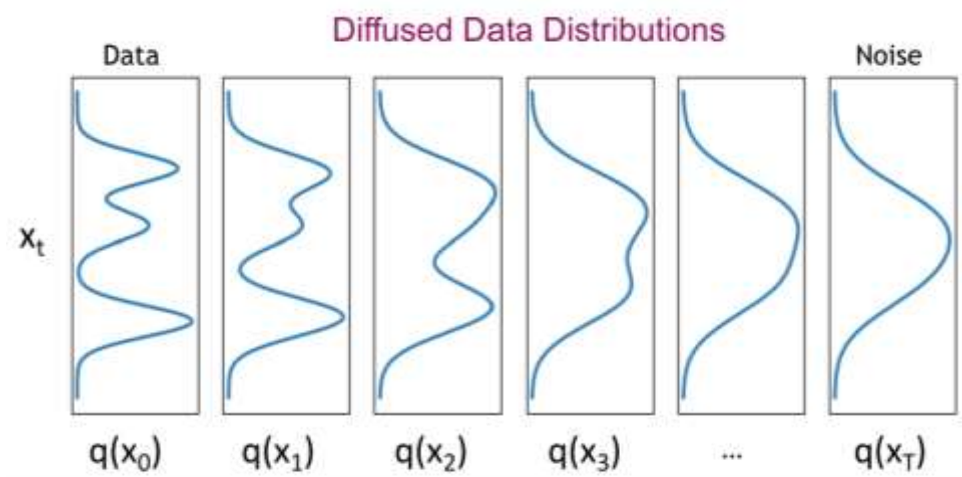
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\, \epsilon$$

The noise schedule) is designed such that $\bar{\alpha}_T \to 0$

Recall DDPM

- What happens to a distribution in the forward diffusion?
- The diffusion kernel $q(\mathbf{x}_t|\mathbf{x}_0)$ is **Gaussian convolution**!

$$\underbrace{q(\mathbf{x}_t)}_{\text{Diffused data dist.}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\text{Joint dist.}} \, d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\text{Input data dist.}} \, \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\text{Diffusion kernel}} \, d\mathbf{x}_0$$



- We can sample $\mathbf{x}_t \sim q(\mathbf{x}_t)$ by first sampling $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, and then sampling $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$ (ancestral sampling).

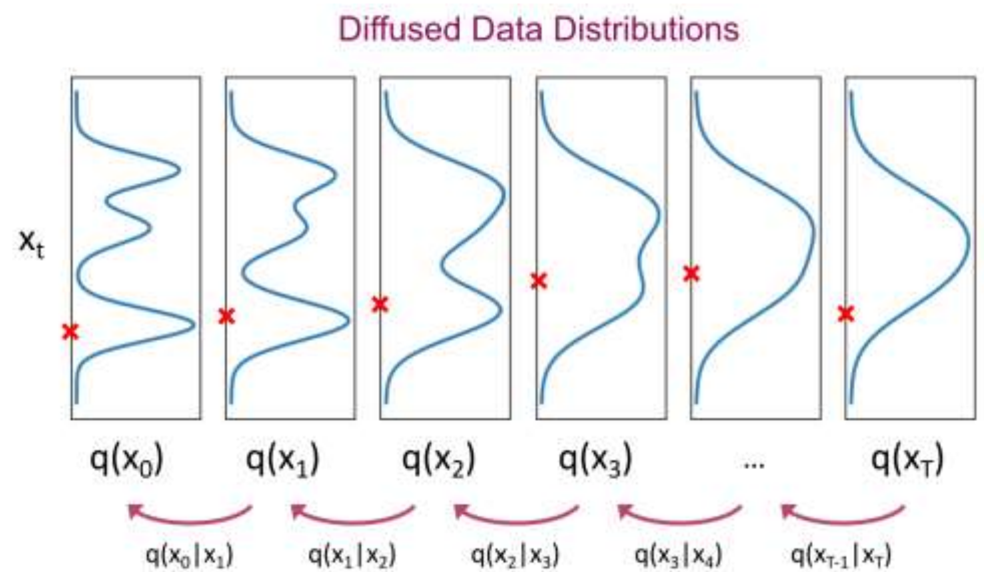
Recall DDPM

- Reverse: Generative Learning by Denoising
- Diffusion parameters are designed: $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Generation:

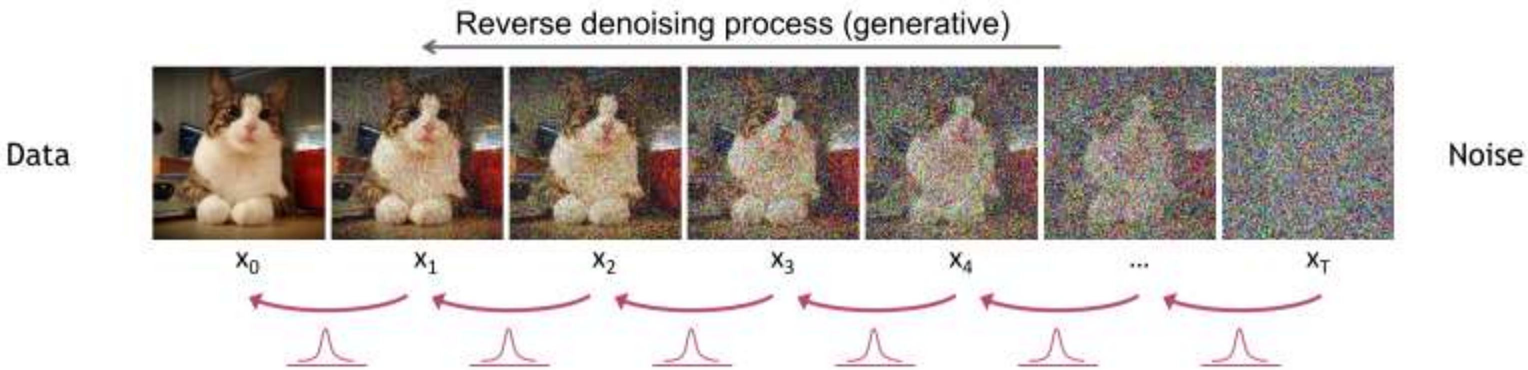
Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

Iteratively sample $\mathbf{x}_{t-1} \sim \underbrace{q(\mathbf{x}_{t-1}|\mathbf{x}_t)}_{\text{True Denoising Dist.}}$



Recall DDPM

- Reverse Denoising Process



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

- From $q(x_{t-1}|x_t)$ to $q(x_{t-1}|x_t, x_0)$

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

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

Recall DDPM

- Learning Denoising Model

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

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

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

- Since $x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}_t)$, 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 regress the noise:

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

$$L = \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0), t\sim\mathcal{U}\{1,T\}, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\lambda_t||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon, t)||^2\right]$$

Recall DDPM

- The authors of DDPM say that it's fine to drop all that baggage in the front and instead just use

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

- Note that this is not a variational lower bound on the log-likelihood anymore: in fact, you can view it as a **reweighted version of ELBO** that emphasizes reconstruction quality!

Recall DDPM

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

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

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

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

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

Recall DDPM

- Rethinking the Training and Sampling processes…..

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

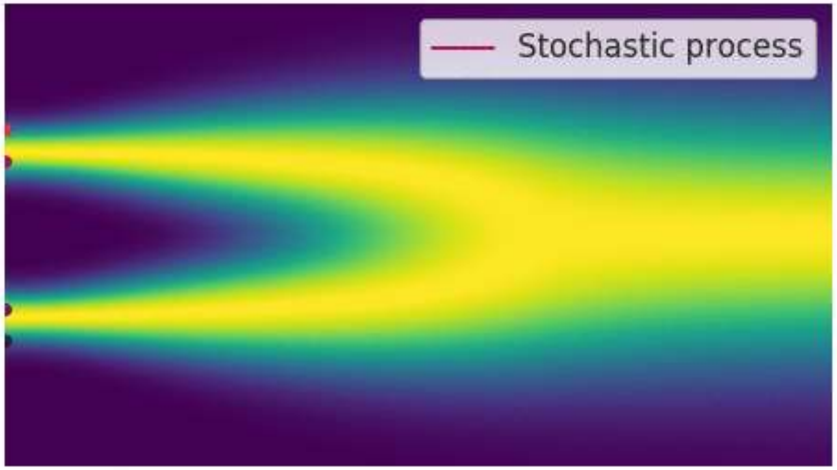
Algorithm 2 Sampling

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** $\mathbf{x}_0$

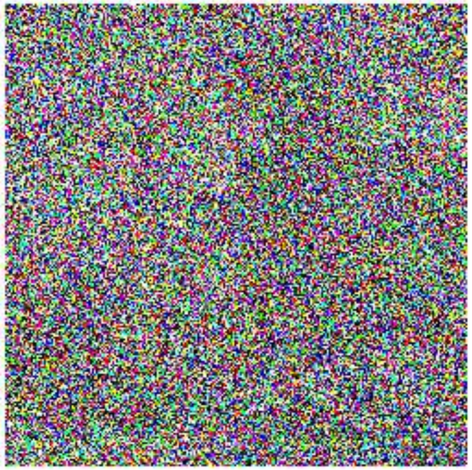
- During training, add noise from 0 to t, then estimate it
- During sampling, note that $\sigma_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$
- As t increases, $\bar{\alpha}_t$ decreases, $\sqrt{1-\bar{\alpha}_t}$ increases
- Thus, $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ works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

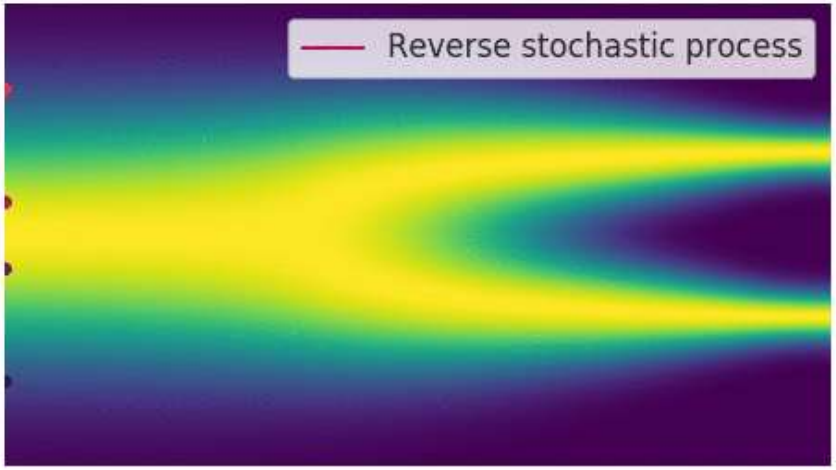
Recall DDPM

- Forward/Reverse process for Image Generation



Forward process: converting the image distribution to pure noise

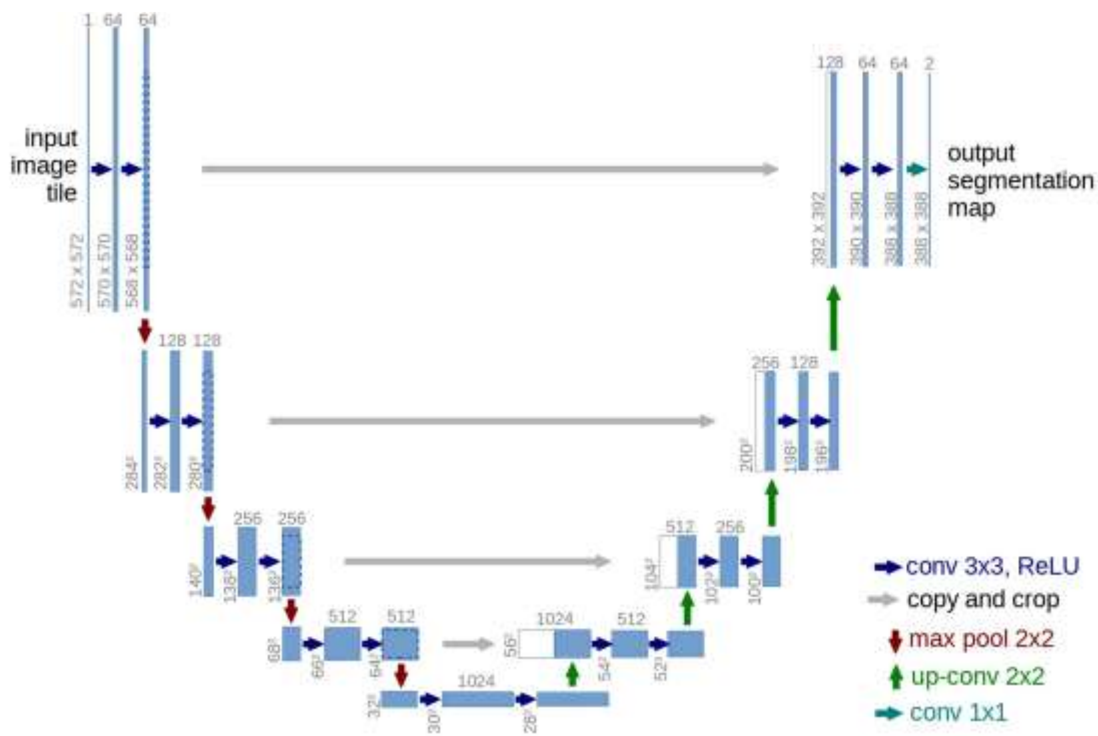




Reverse process: sampling from the image distribution, starting with pure noise

Implementation Considerations

- UNet + Other Stuff



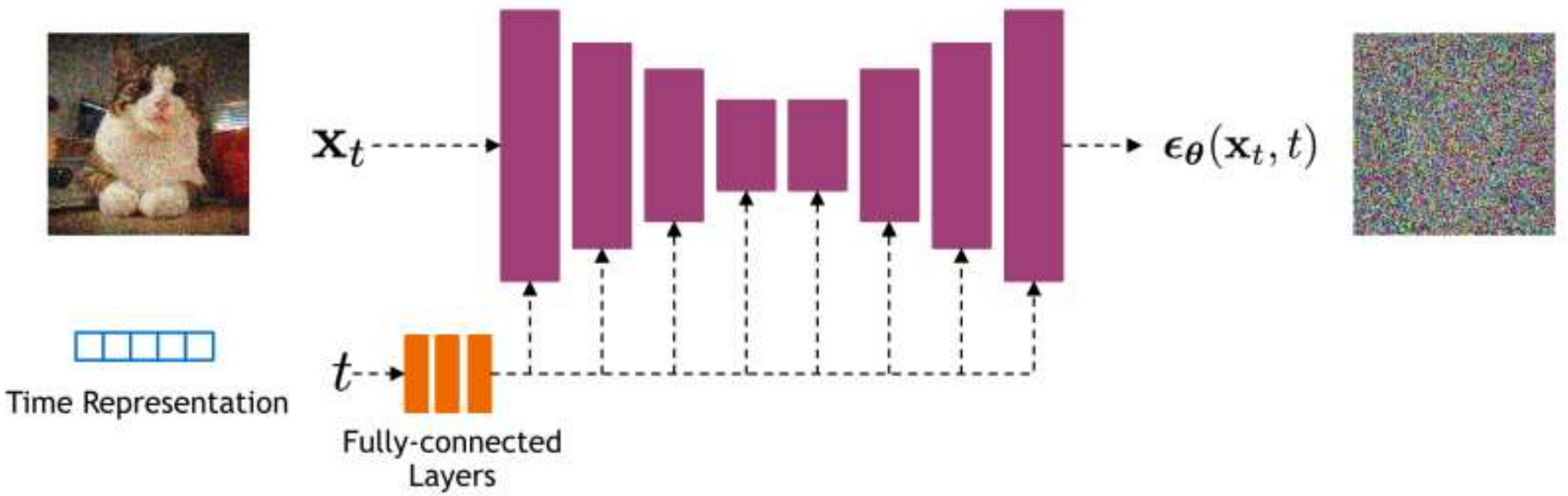
Diffusion models typically use a U-Net on steroids as the noise predictive model — you take the good ol' model that you are already familiar with and add:

- Positional Embeddings
- ResNet Blocks
- ConvNext Blocks
- Attention Modules
- Group Normalization
- Swish and GeLU

It's a massive kitchen sink of modern CV tricks

Implementation Considerations

- Diffusion models often use **U-Net architectures** with ResNet blocks and self-attention layers



- Time representation: sinusoidal positional embeddings or random Fourier features.
- Time features are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers.

Outline

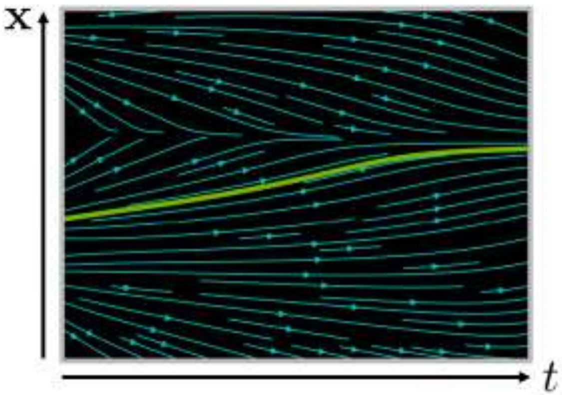
- Diffusion: Denoising Diffusion Probabilistic Models
- Score-based Diffusion Models
- Accelerated Sampling
- Conditional Generation and Guidance

Crash Course in Differential Equations

- ODE and SDE

Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t) \quad \text{or} \quad \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t$$



Analytical Solution: $\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}, \tau)\mathrm{d}\tau$

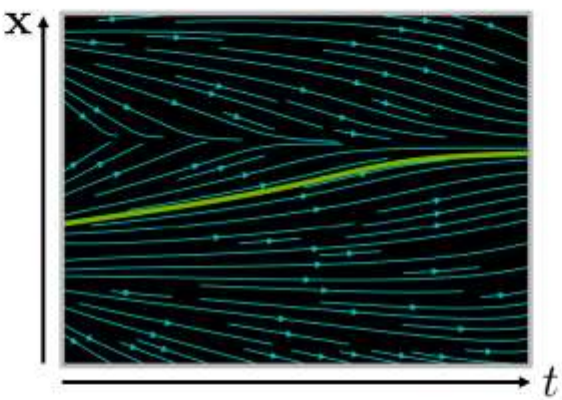
Iterative Numerical Solution: $\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$

Crash Course in Differential Equations

- ODE and SDE

Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t) \quad \text{or} \quad \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t$$



Analytical Solution: $\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}, \tau)\mathrm{d}\tau$

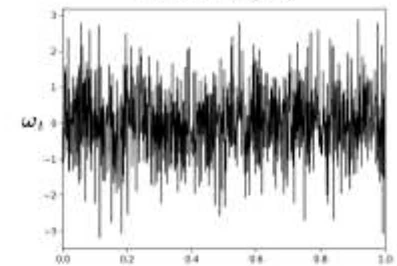
Iterative Numerical Solution: $\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$

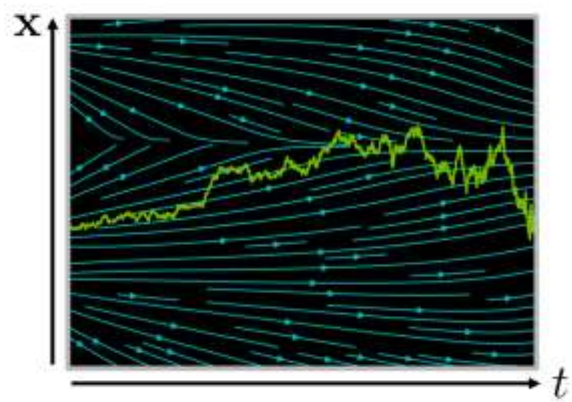
Stochastic Differential Equation (SDE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \underbrace{\mathbf{f}(\mathbf{x}, t)}_{\text{drift coefficient}} + \underbrace{\sigma(\mathbf{x}, t)}_{\text{diffusion coefficient}}\boldsymbol{\omega}_t$$

Wiener Process (Gaussian White Noise)

$$\left(\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + \sigma(\mathbf{x}, t)\mathrm{d}\boldsymbol{\omega}_t\right)$$

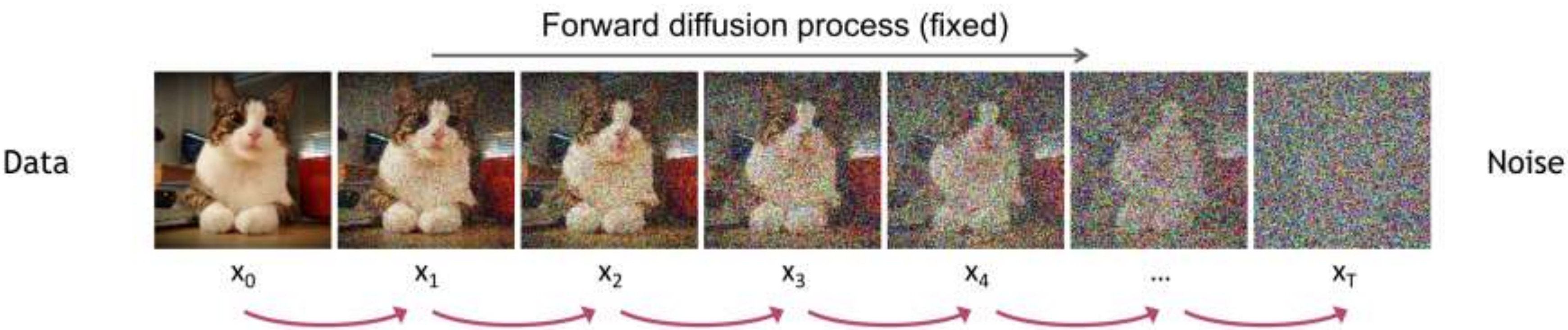




$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t + \sigma(\mathbf{x}(t), t)\sqrt{\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

Score-based Diffusion Models

- **Forward Diffusion Process** as Stochastic Differential Equation (continuous form)
- Consider the limit of many small steps:



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

discretization of

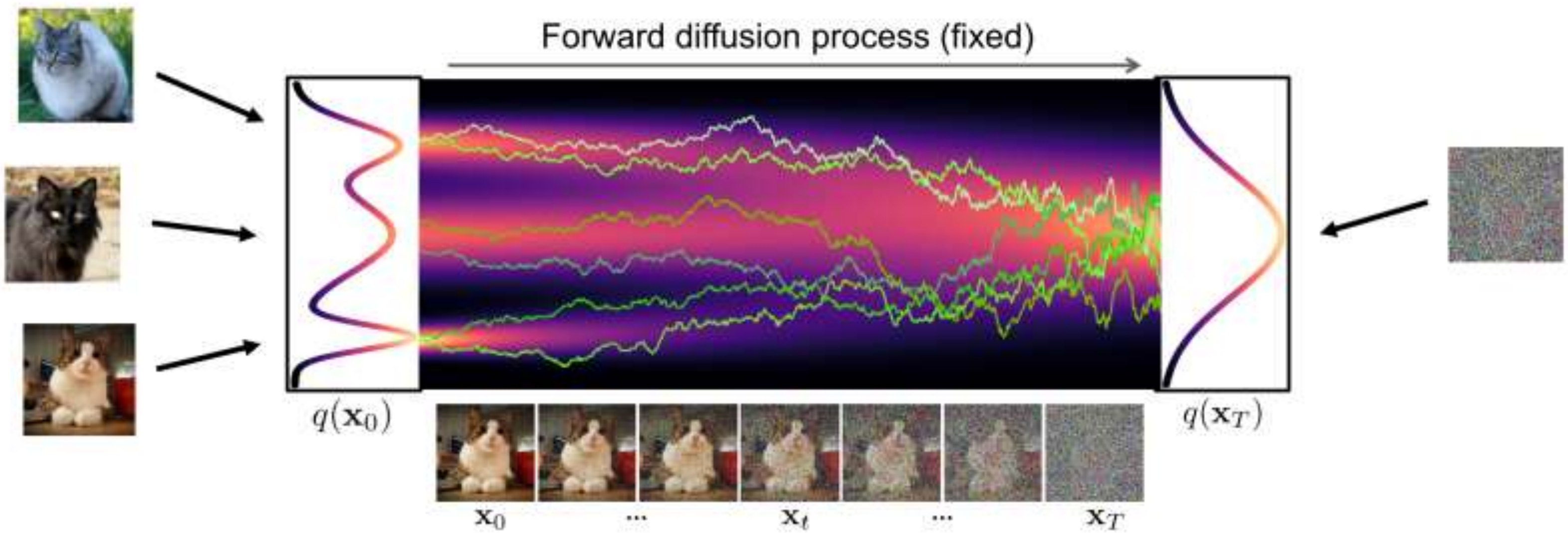
$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

Stochastic Differential Equation (SDE)

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

Score-based Diffusion Models

- **Forward Diffusion Process** as Stochastic Differential Equation (continuous form)

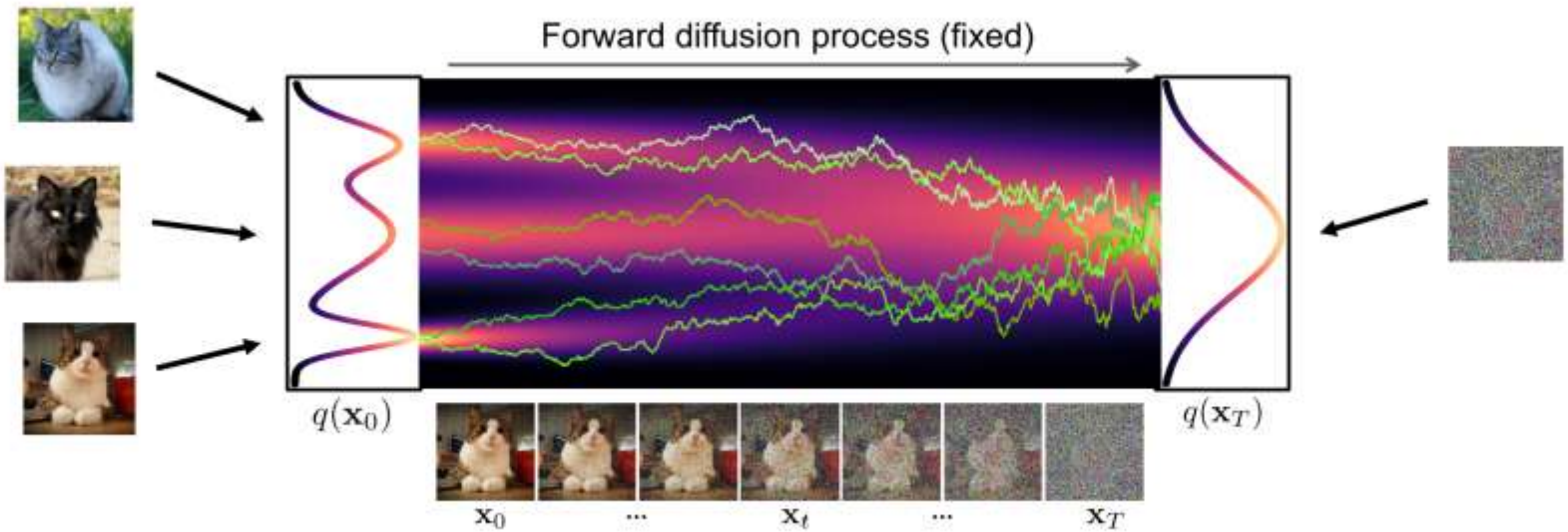


Forward Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t}_{\substack{\text{drift term} \\ \text{(pulls towards mode)}}} + \underbrace{\sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t}_{\substack{\text{diffusion term} \\ \text{(injects noise)}}}$$

Score-based Diffusion Models

- The Generative Reverse Stochastic Differential Equation



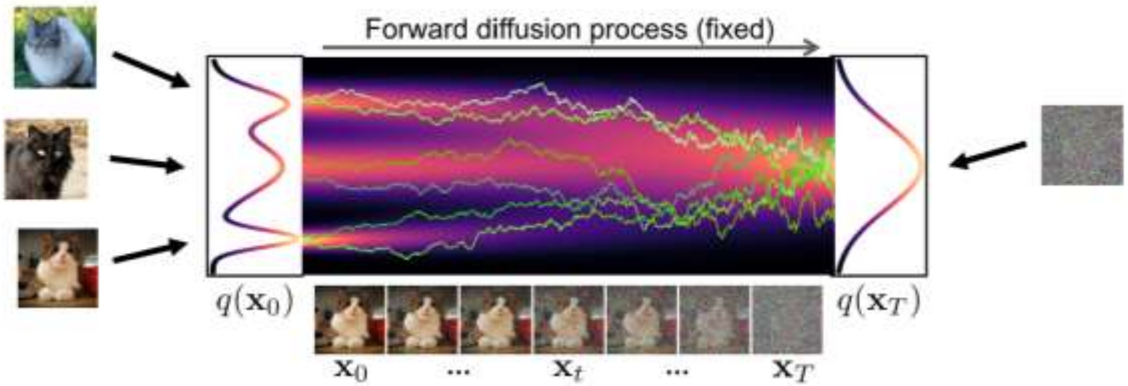
Forward Diffusion SDE:

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

- But what about the **reverse direction**, necessary for generation?

Score-based Diffusion Models

- The Generative Reverse Stochastic Differential Equation



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 Diffusion SDE:

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

➡ **Simulate reverse diffusion process: Data generation from random noise!**

Score function example

- Probability density function (pdf)

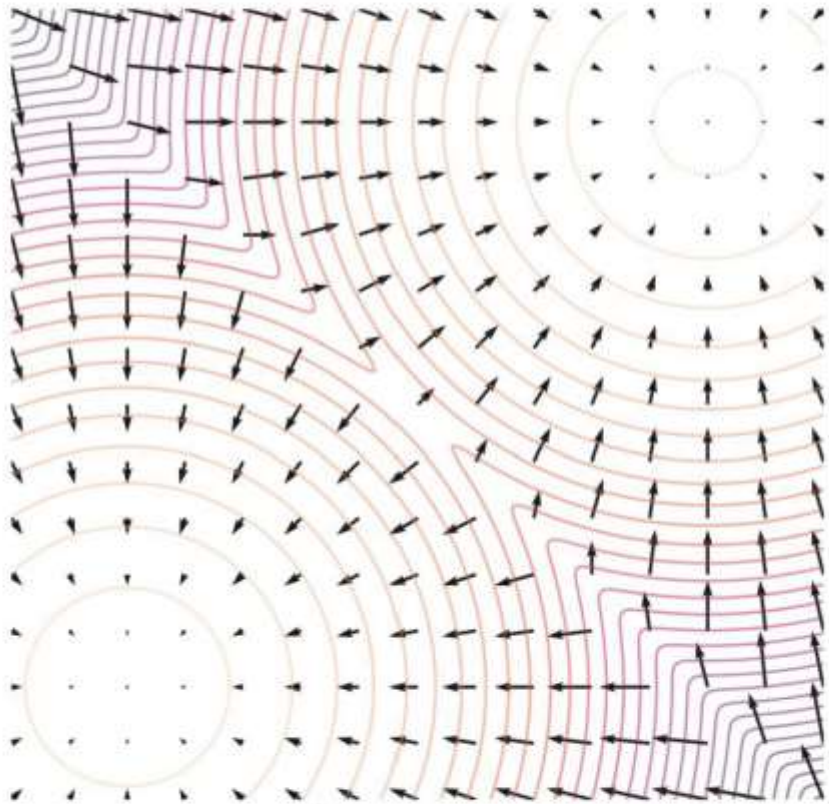
$$p(\mathrm{X})$$

- 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

Score-based Diffusion Models

- The Generative Reverse Stochastic Differential Equation

But how to get the score function $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$?

Forward Diffusion SDE:

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

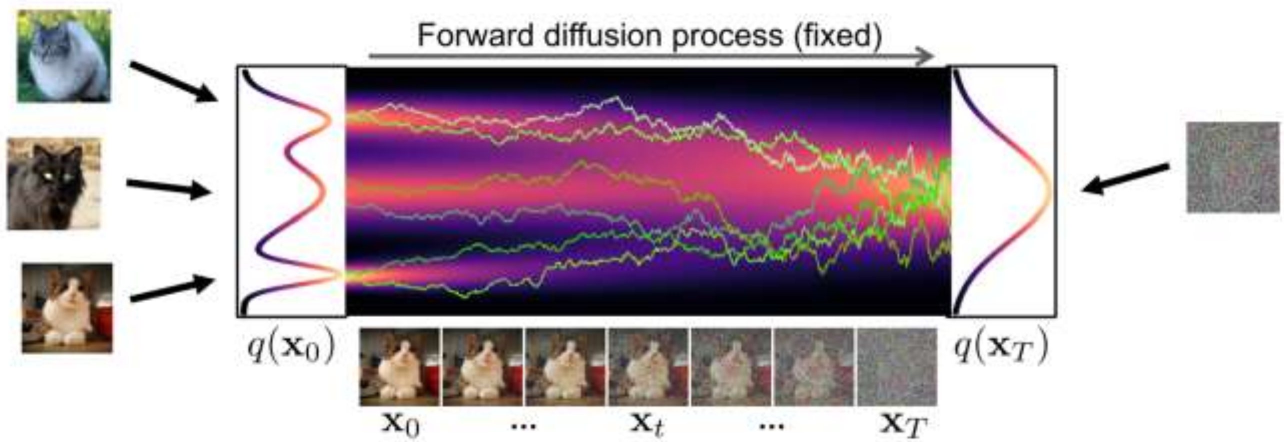
Reverse Generative Diffusion SDE:

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

➡ Simulate reverse diffusion process: Data generation from random noise!

Score Matching

- Naïve idea, learn model for the score function by direct regression?



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

➡ **But $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$ (score of the *marginal diffused density* $q_t(\mathbf{x}_t)$) is not tractable!**

Denoising Score Matching

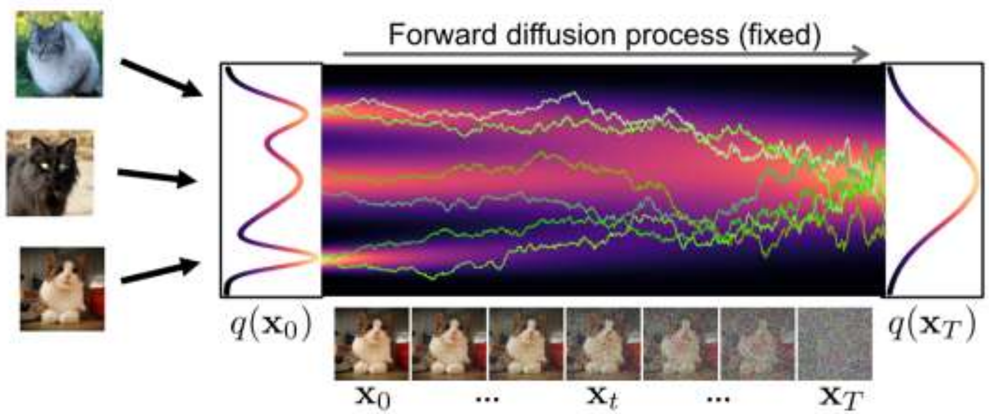
- Instead, diffuse individual data points $\mathbf{x}_0$

  Diffused $q_t(\mathbf{x}_t|\mathbf{x}_0)$ is Trackable!

"Variance Preserving" SDE:

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

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



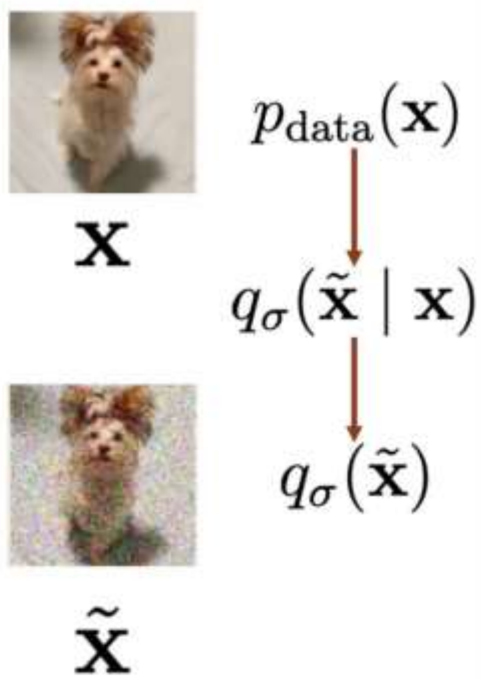
- Denoising Score Matching:

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

➡ **After expectations,** $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$**!**

Denoising Score Matching

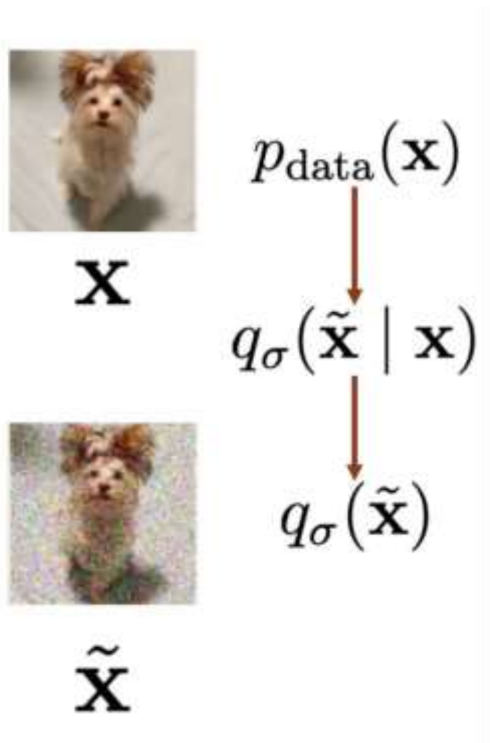
- Matching the score of a noise-perturbed distribution



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

Denoising Score Matching

- Matching the score of a noise-perturbed distribution



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

Denoising Score Matching

- Matching the score of a noise-perturbed distribution

$\mathbf{x}$

$\tilde{\mathbf{x}}$

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

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

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

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

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

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

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

Denoising Score Matching

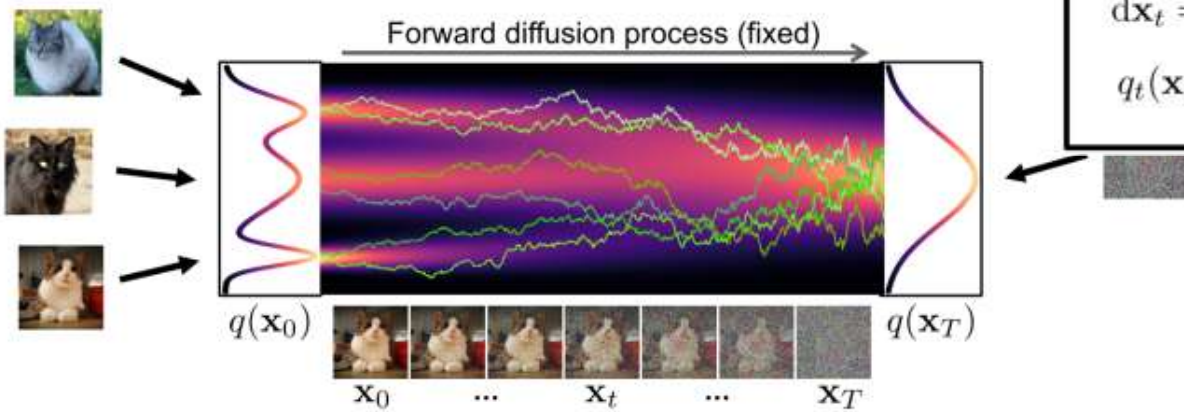
- Instead, diffuse individual data points $\mathbf{x}_0$

  Diffused $q_t(\mathbf{x}_t|\mathbf{x}_0)$ is Trackable!

"Variance Preserving" SDE:

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

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



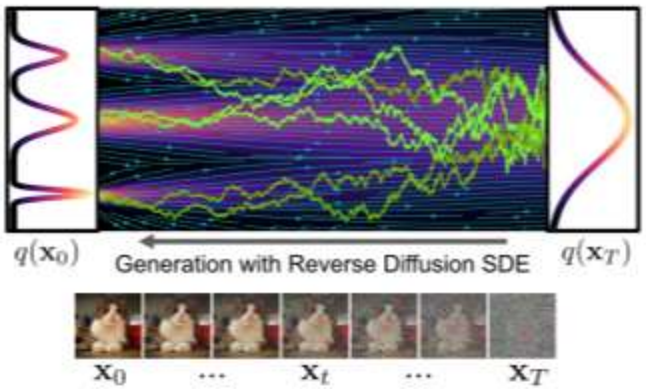
- Denoising Score Matching:

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

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

Synthesis with SDE vs. ODE

- How to solve the generative SDE or ODE in practice?



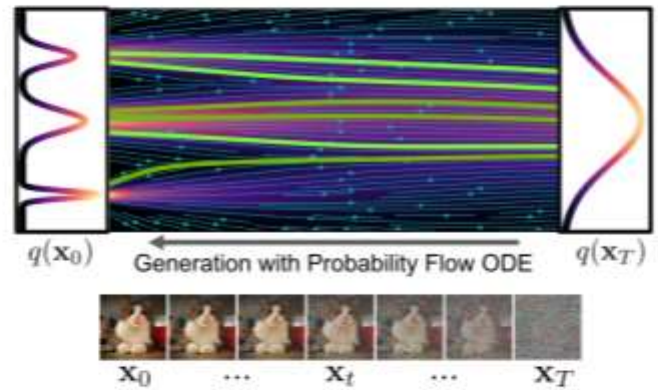
Generative Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + 2\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t$$

➡ *Euler-Maruyama*:

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \frac{1}{2}\beta(t)\left[\mathbf{x}_t + 2\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\Delta t + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

➡ *Ancestral Sampling* (Part 1) is also a generative SDE sampler!



Probability Flow ODE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t$$

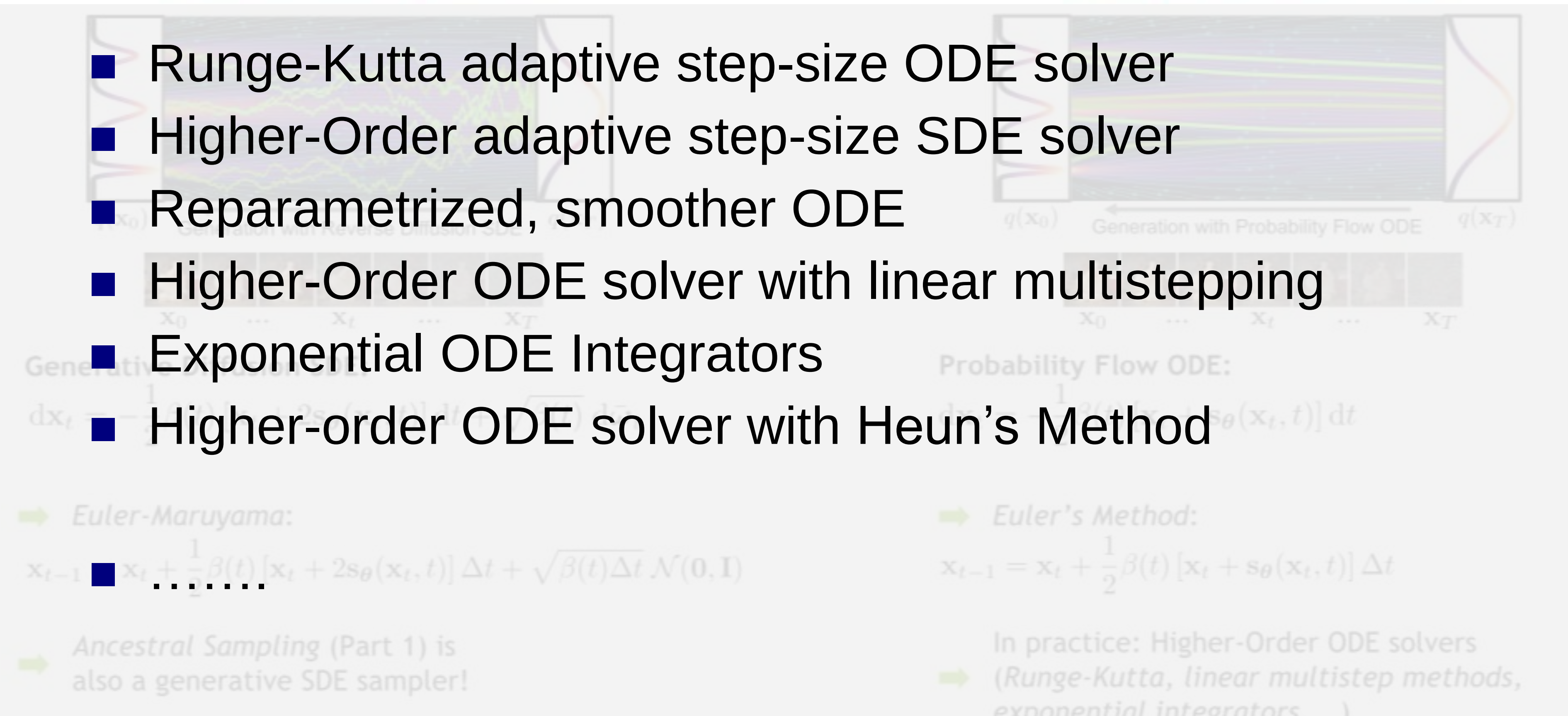
➡ *Euler's Method*:

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \frac{1}{2}\beta(t)\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\Delta t$$

➡ In practice: Higher-Order ODE solvers (*Runge-Kutta, linear multistep methods, exponential integrators, ...*)

Synthesis with SDE vs. ODE

- How to solve the generative SDE or ODE in practice?



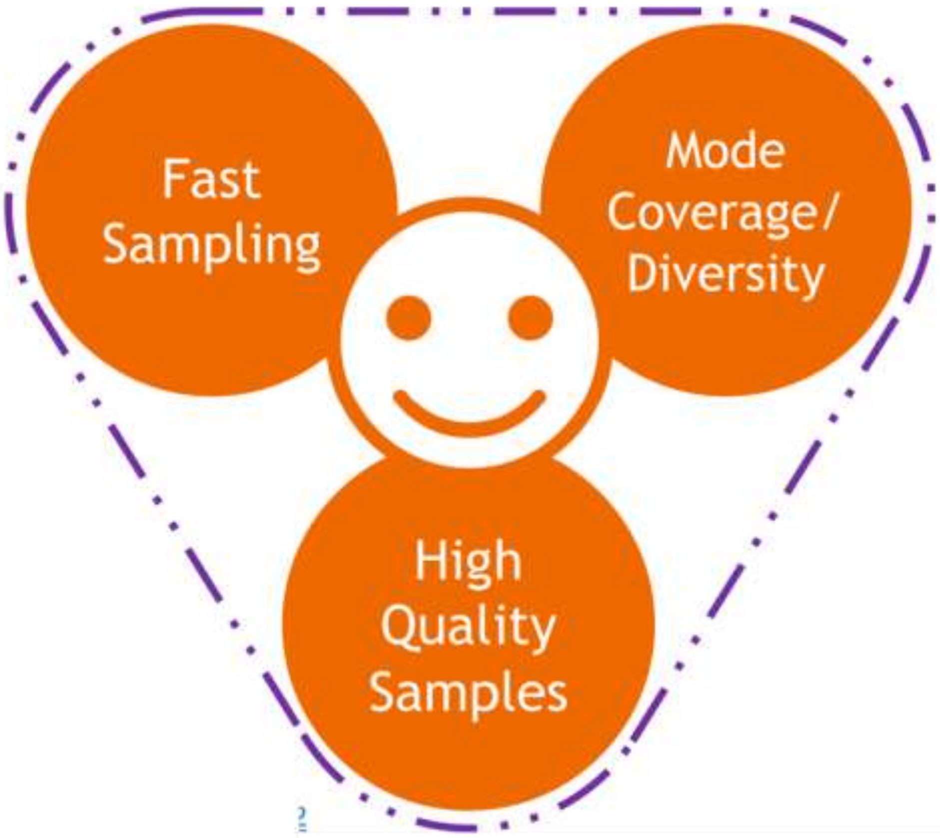
- Runge-Kutta adaptive step-size ODE solver
- Higher-Order adaptive step-size SDE solver
- Reparametrized, smoother ODE
- Higher-Order ODE solver with linear multistepping
- Exponential ODE Integrators
- Higher-order ODE solver with Heun's Method
- …….

Outline

- Diffusion: Denoising Diffusion Probabilistic Models
- Score-based Diffusion Models
- Accelerated Sampling
- Conditional Generation and Guidance

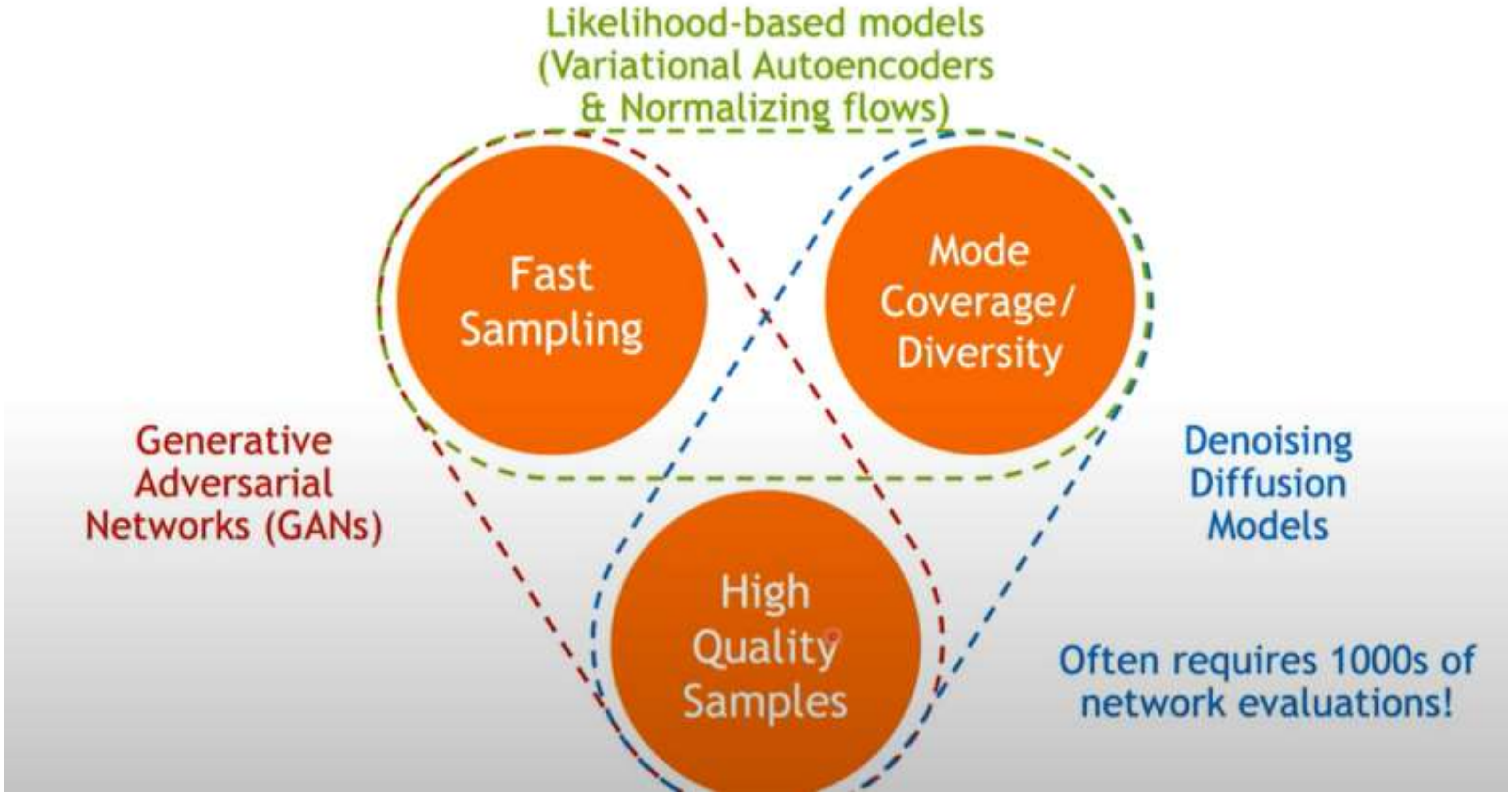
What makes a good generative model?

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



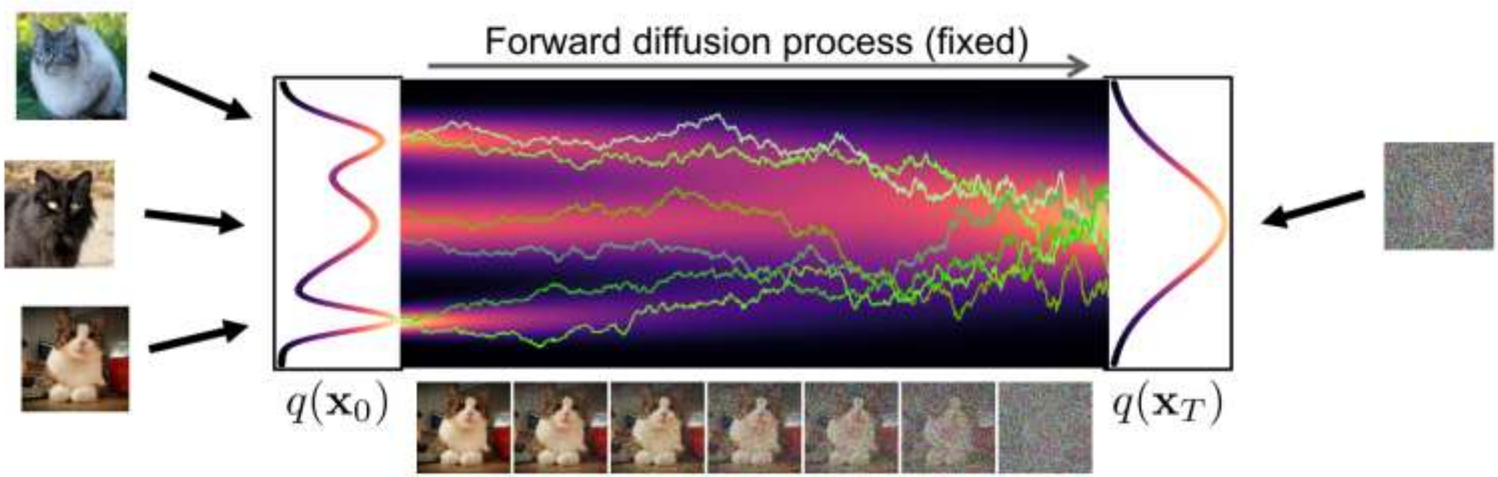
What makes a good generative model?

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



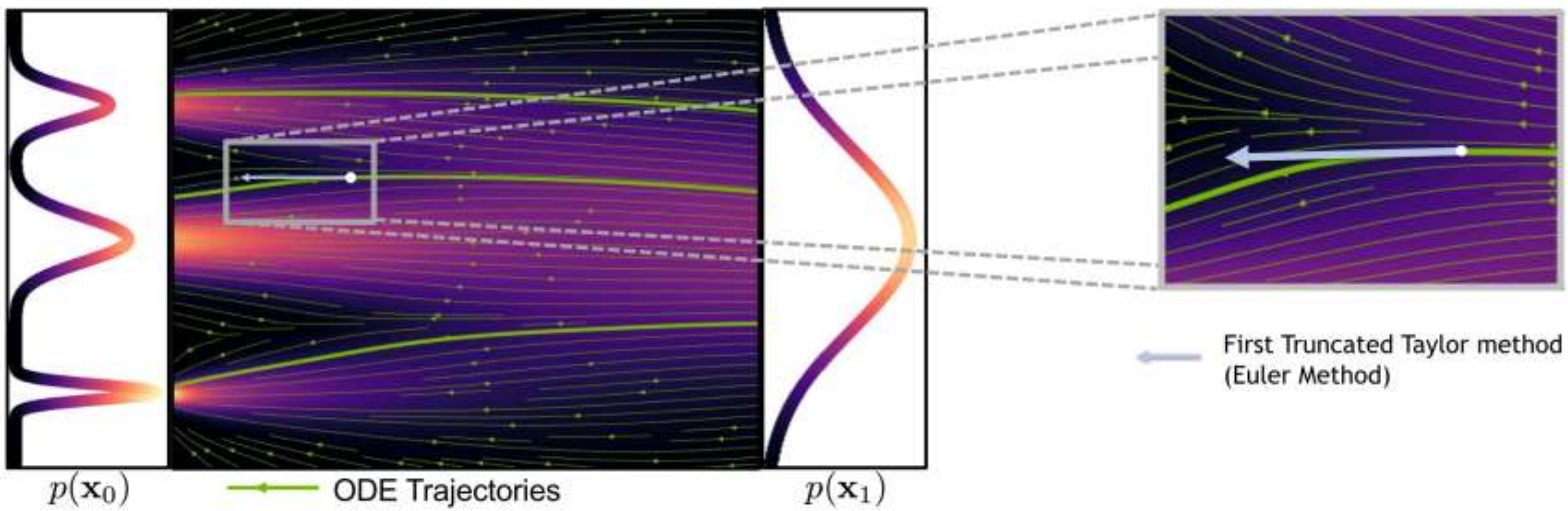
Accelerated Sampling

- Advanced ODE/SDE Solvers
- Distillation Techniques
- Low-dim Diffusion Processes
- … …



Generative ODEs

- Solve ODEs with as little function evaluations as possible! $dx = \epsilon_\theta(x, t)dt$

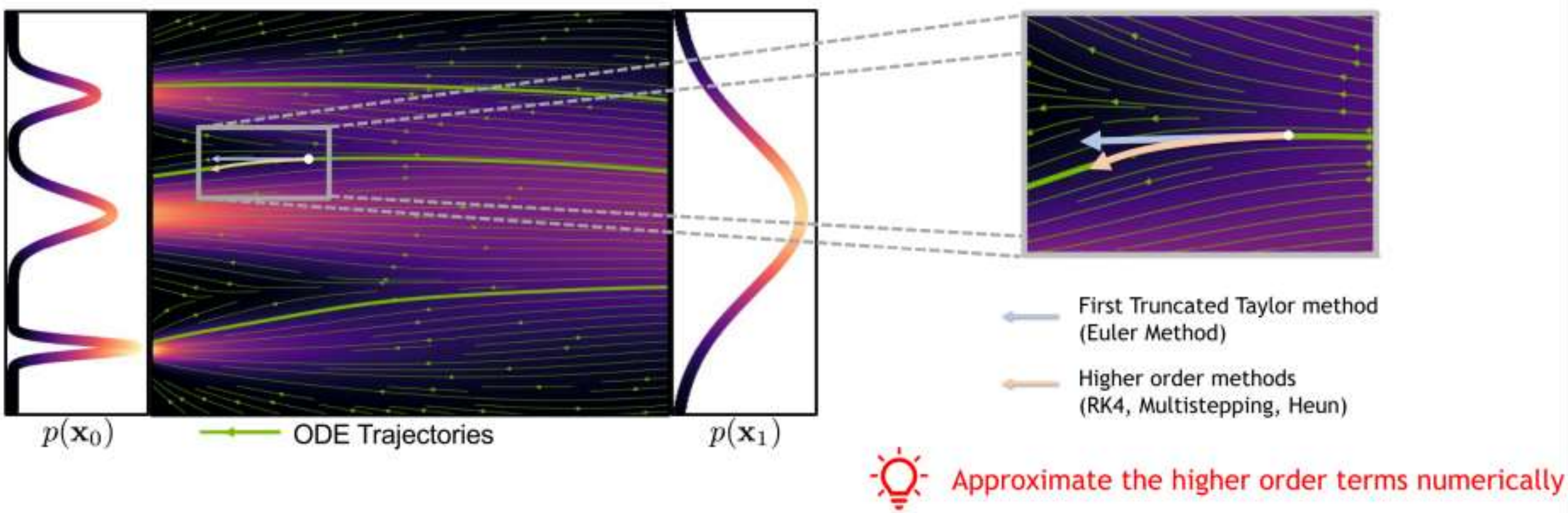


Song et al., "Denoising Diffusion Implicit Models (DDIM)", ICLR 2021
https://arxiv.org/abs/2010.02502

Generative ODEs

- Solve ODEs with as little function evaluations as possible!

$$dx = \epsilon_\theta(x, t)dt$$



Generative ODEs

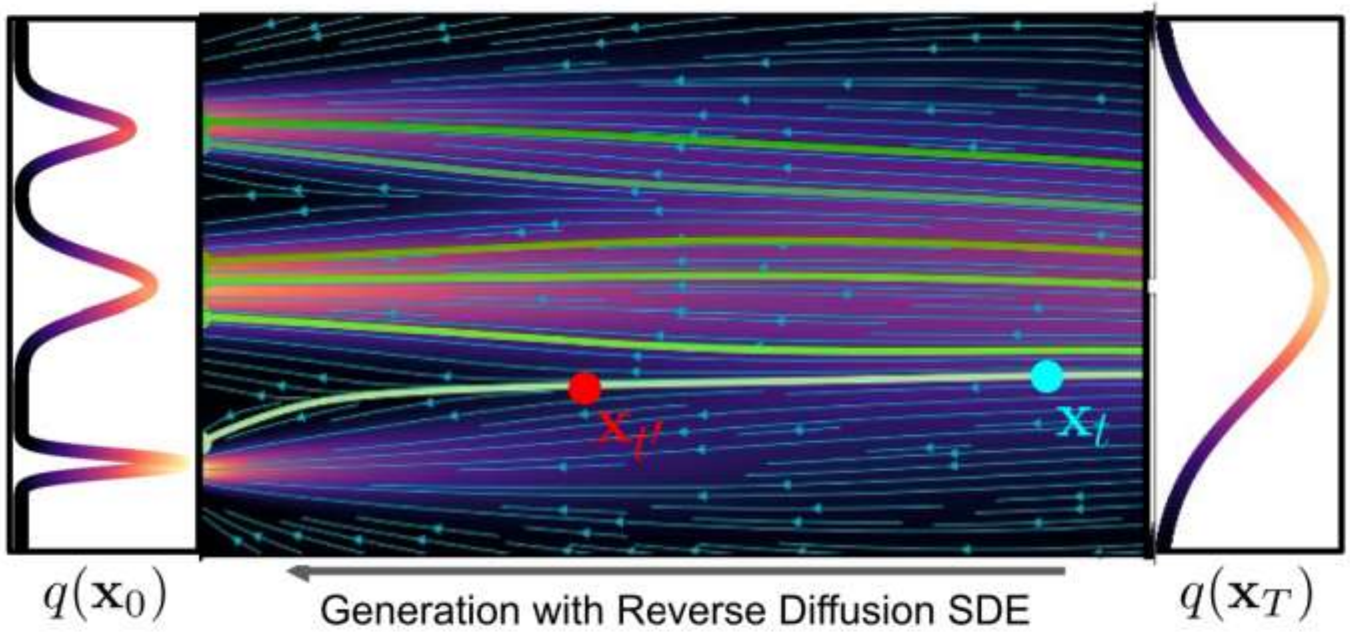
- Solve ODEs with as little function evaluations as possible!

$$dx = \epsilon_\theta(x, t)dt$$

  - Runge-Kutta adaptive step-size ODE solver: https://arxiv.org/abs/2011.13456
  - Higher-Order adaptive step-size SDE solver: https://arxiv.org/abs/2105.14080
  - Reparametrized, smoother ODE: gDDIM https://arxiv.org/abs/2206.05564
  - Higher-Order ODE solver with linear multistepping: https://arxiv.org/abs/2202.09778
  - Exponential ODE Integrators: DPM, DPM-Solver++ https://arxiv.org/abs/2206.00927
  - Higher-Order ODE solver with Heun's Method: https://arxiv.org/abs/2206.00364
  - Many more: https://arxiv.org/abs/2305.19947

Distillation Techniques

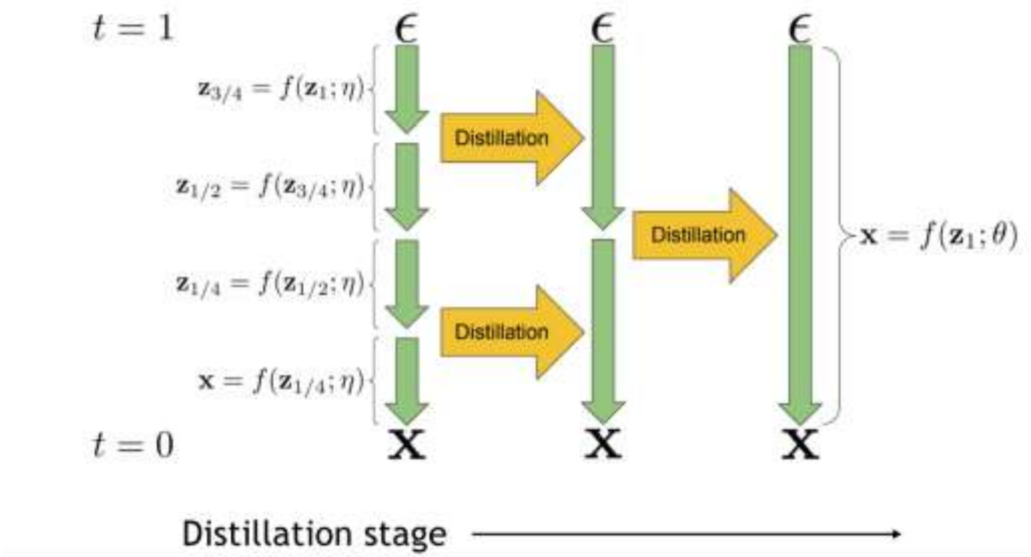
- ODE Distillation



Can we train a neural network to directly predict $\mathbf{x}_{t'}$ given $\mathbf{x}_t$?

Progressive Distillation

- Distill a deterministic ODE sampler to the same model architecture
- At each stage, a **"student"** model is learned to distill two adjacent sampling steps of the **"teacher"** model to one sampling step.
- At next stage, the **"student"** model from previous stage will serve as the new **"teacher"** model.



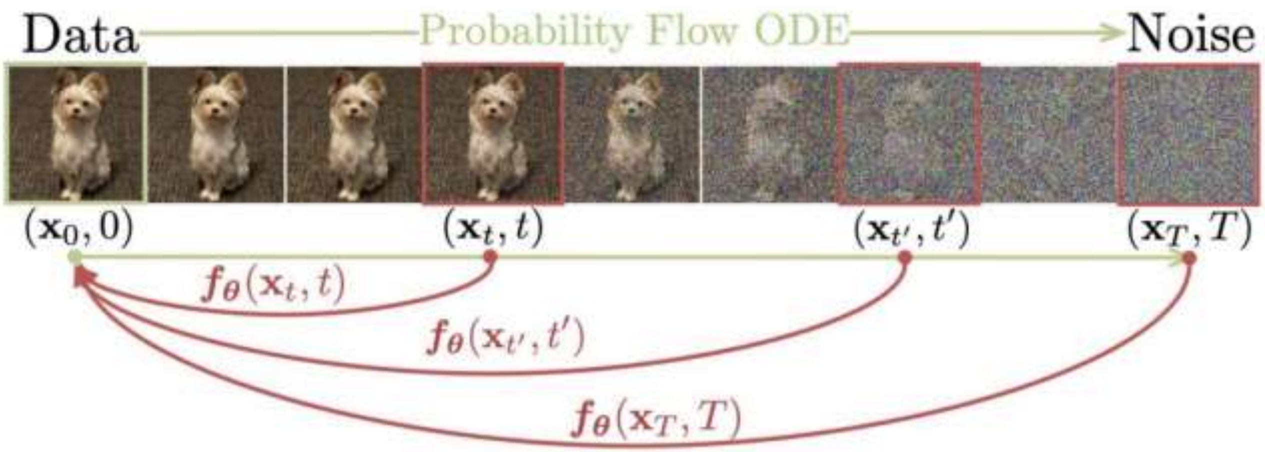
Salimans & Ho, "Progressive distillation for fast sampling of diffusion models", ICLR 2022. https://arxiv.org/abs/2202.00512

Progressive Distillation in Latent Space

(a) 2 denoising steps (b) 4 denoising steps (c) 8 denoising steps

Meng et al., “On Distillation of Guided Diffusion Models”, CVPR 2023 (Award Candidate). https://arxiv.org/abs/2210.03142

Consistency Distillation



Points on the same trajectory should generated the same $\mathbf{x}_0$

Assume $f_\theta(\mathbf{x}_t, t)$ is the current estimation of $\mathbf{x}_0$

Basic idea:

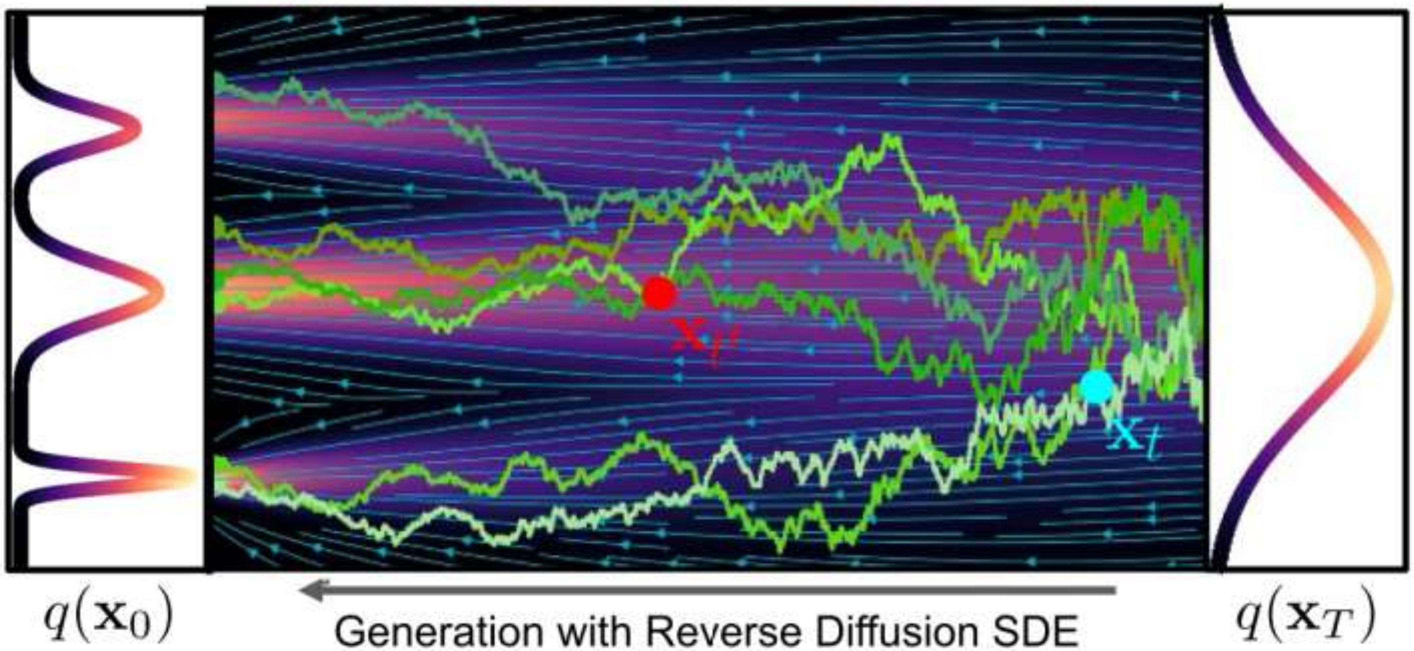
- Find $\mathbf{x}_t$ and $\mathbf{x}_{t'}$ on a trajectory by solving generative ODE in $[t, t']$
- Minimize:

$$\min_\theta \; ||f_{\text{EMA}}(\mathbf{x}_t, t) - f_\theta(\mathbf{x}'_t, t')||_2^2$$

Song et al., Consistency Models, ICML 2023, https://arxiv.org/abs/2303.01469

Distillation Techniques

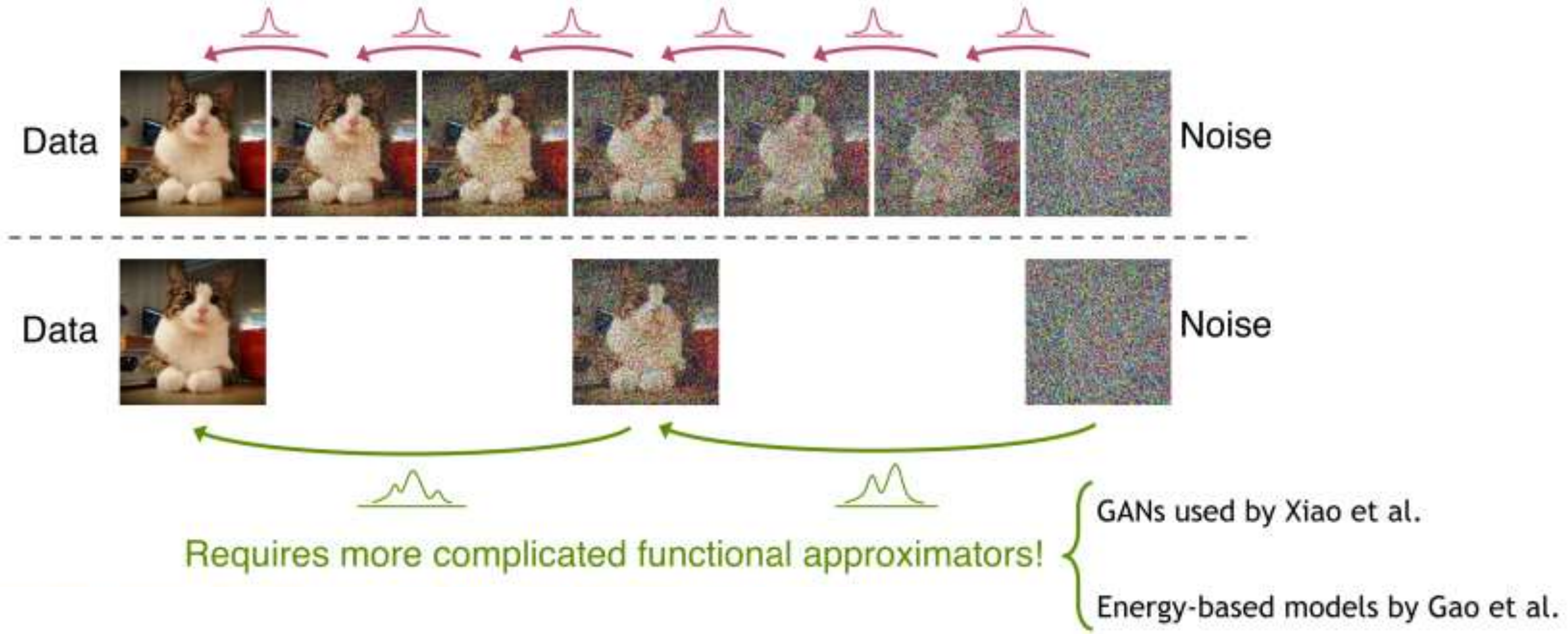
- SDE Distillation



Can we train a neural network to directly predict $\mathbf{x}_{t'}$ given $\mathbf{x}_t$?

Approximation of Reverse Process

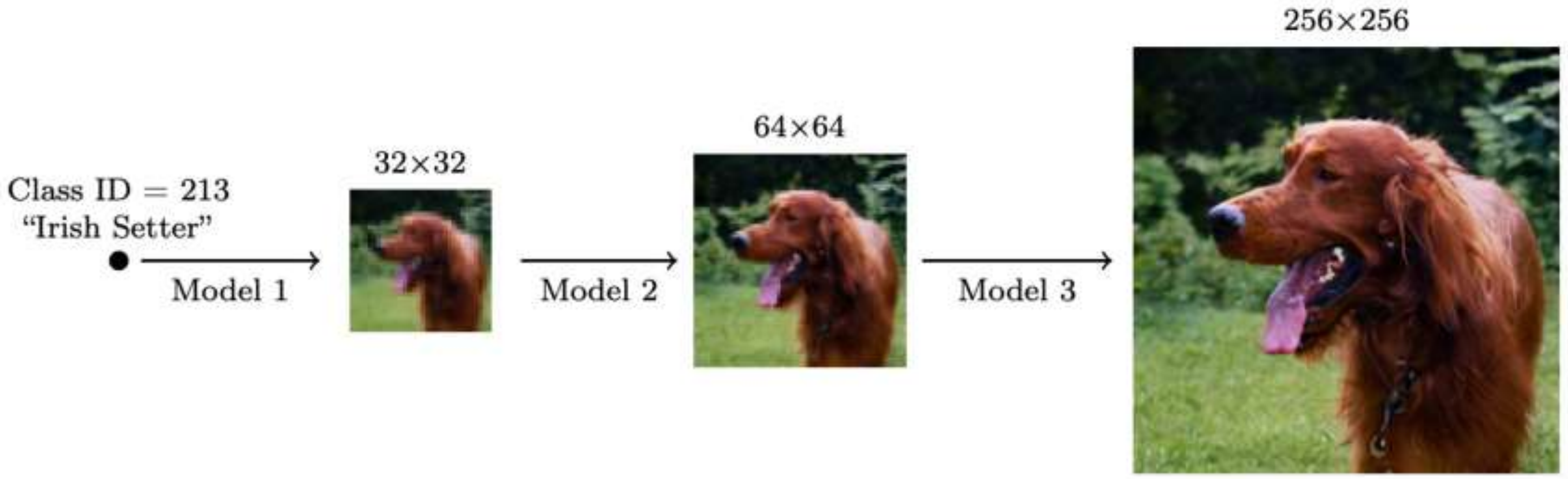
- Normal assumption in denoising distribution holds only for small step
- Denoising Process with Uni-modal Normal Distribution



Xiao et al., “Tackling the Generative Learning Trilemma with Denoising Diffusion GANs”, ICLR 2022.
Gao et al., “Learning energy-based models by diffusion recovery likelihood”, ICLR 2021.
https://arxiv.org/abs/2112.07804

Low-dim. Diffusion Processes

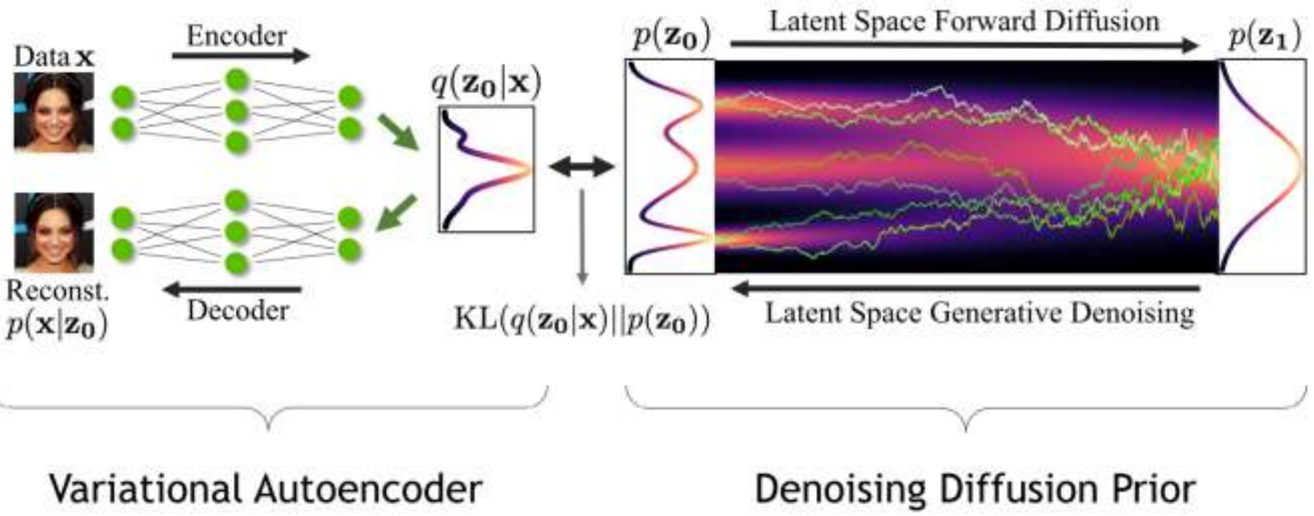
- Cascaded Generation
- Cascaded Diffusion Models outperform Big-GAN in FID and IS and VQ-VAE2 in Classification Accuracy Score.

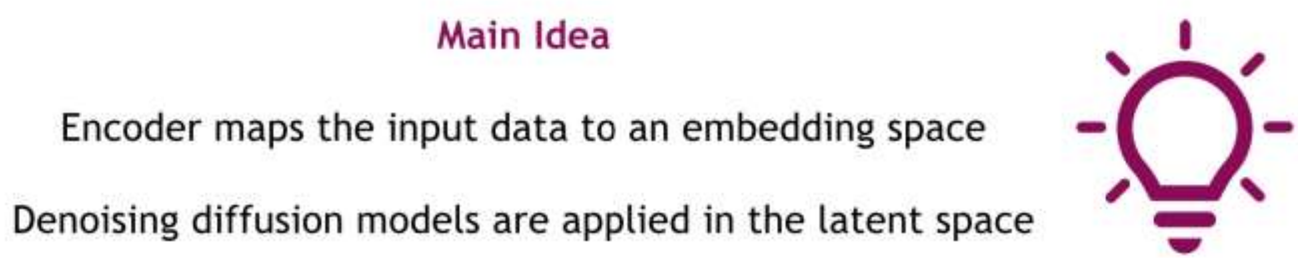


Ho et al., "Cascaded Diffusion Models for High Fidelity Image Generation", 2021. https://arxiv.org/abs/2106.15282
Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", 2022, https://arxiv.org/abs/2204.06125
Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", 2022, https://arxiv.org/abs/2205.11487

Latent Diffusion Models

- VAE + score-based prior

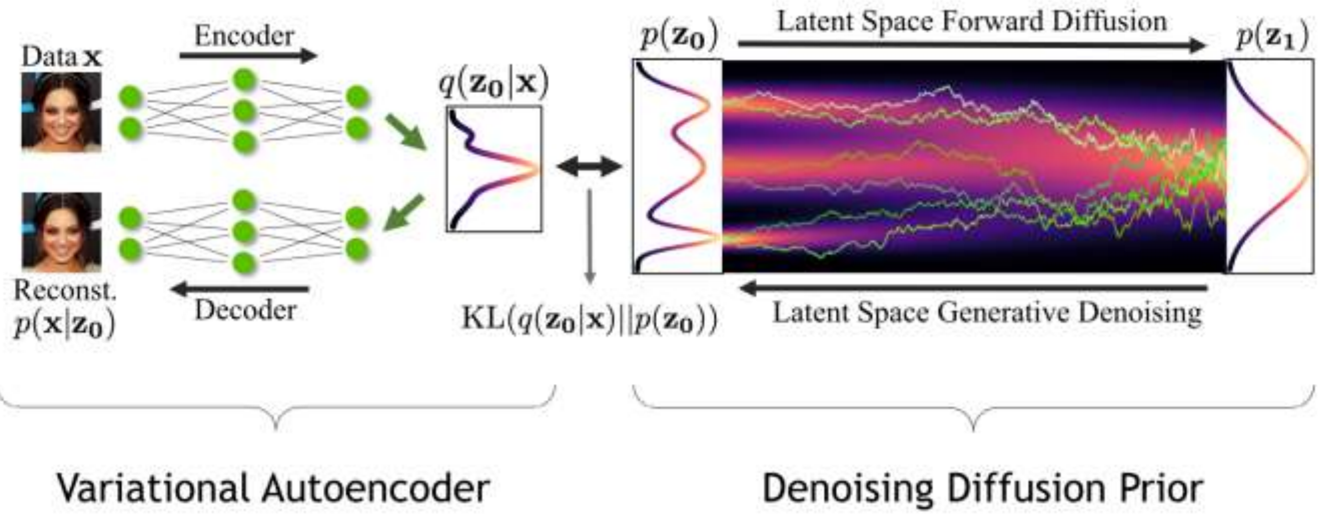




Vahdat et al., "Score-based generative modeling in latent space", NeurIPS 2021. https://arxiv.org/abs/2106.05931

Latent Diffusion Models

- VAE + score-based prior



Advantages:

(1) The distribution of latent embeddings close to Normal distribution → *Simpler denoising, Faster synthesis!*

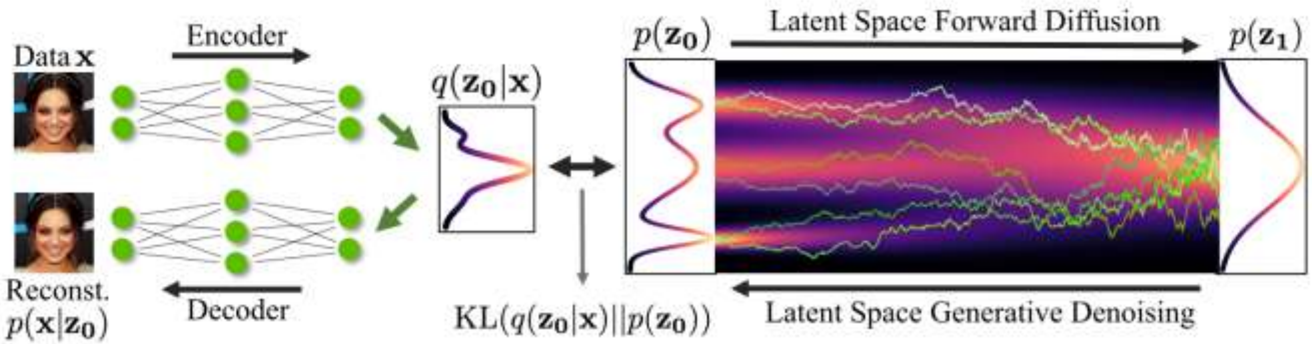
(2) Latent space → *More expressivity and flexibility in design!*

(3) Tailored Autoencoders → *More expressivity, Application to any data type (graphs, text, 3D data, etc.) !*

Vahdat et al., "Score-based generative modeling in latent space", NeurIPS 2021. https://arxiv.org/abs/2106.05931

Latent Diffusion Models

- End-to-End Training objective



$$\mathcal{L}(\mathbf{x},\phi,\theta,\psi) = \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[-\log p_\psi(\mathbf{x}|\mathbf{z}_0)\right] + \mathrm{KL}(q_\phi(\mathbf{z}_0|\mathbf{x})||p_\theta(\mathbf{z}_0))$$

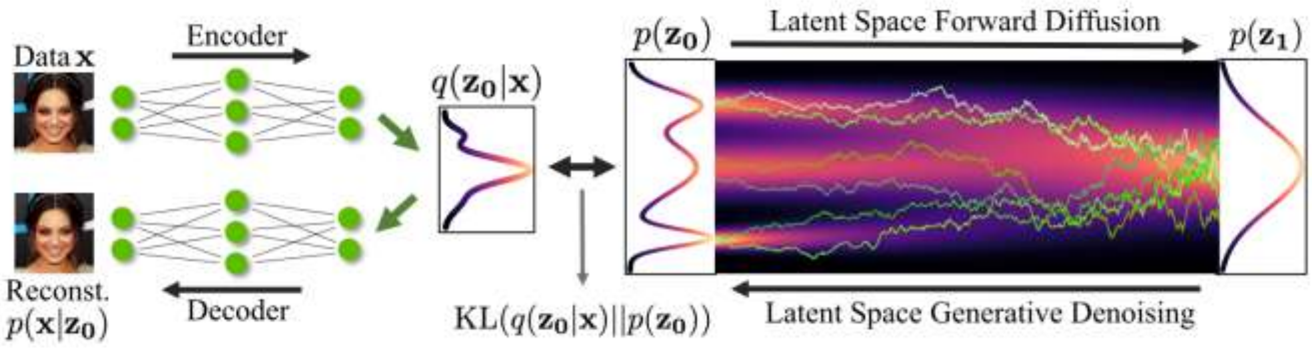
$$= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[-\log p_\psi(\mathbf{x}|\mathbf{z}_0)\right]}_{\text{reconstruction term}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[\log q_\phi(\mathbf{z}_0|\mathbf{x})\right]}_{\text{negative encoder entropy}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[-\log p_\theta(\mathbf{z}_0)\right]}_{\text{cross entropy}}$$

$$CE(q(\mathbf{z}_0|\mathbf{x})||p(\mathbf{z}_0)) = \underbrace{\mathbb{E}_{t\sim\mathcal{U}[0,1]}}_{\text{time sampling}}\left[\frac{g(t)^2}{2}\underbrace{\mathbb{E}_{q(\mathbf{z}_t,\mathbf{z}_0|\mathbf{x})}}_{\text{Forward diffusion}}\left[\underbrace{||\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{z}_0)}_{\text{Diffusion kernel}} - \underbrace{\nabla_{\mathbf{z}_t}\log p(\mathbf{z}_t)||_2^2}_{\text{Trainable score function}}\right]\right] + \underbrace{\frac{D}{2}\log\left(2\pi e\sigma_0^2\right)}_{\text{Constant}}$$

Vahdat et al., “Score-based generative modeling in latent space”, NeurIPS 2021. https://arxiv.org/abs/2106.05931

Latent Diffusion Models

- **Stable Diffusion:** the efficiency and expressivity of LDM+ open-source access fueled a large body of work in the community



- Two stage training: train autoencoder first, then train the diffusion prior
- Focus on compression without of any loss in reconstruction quality
- Demonstrated the expressivity of latent diffusion models on many conditional problems

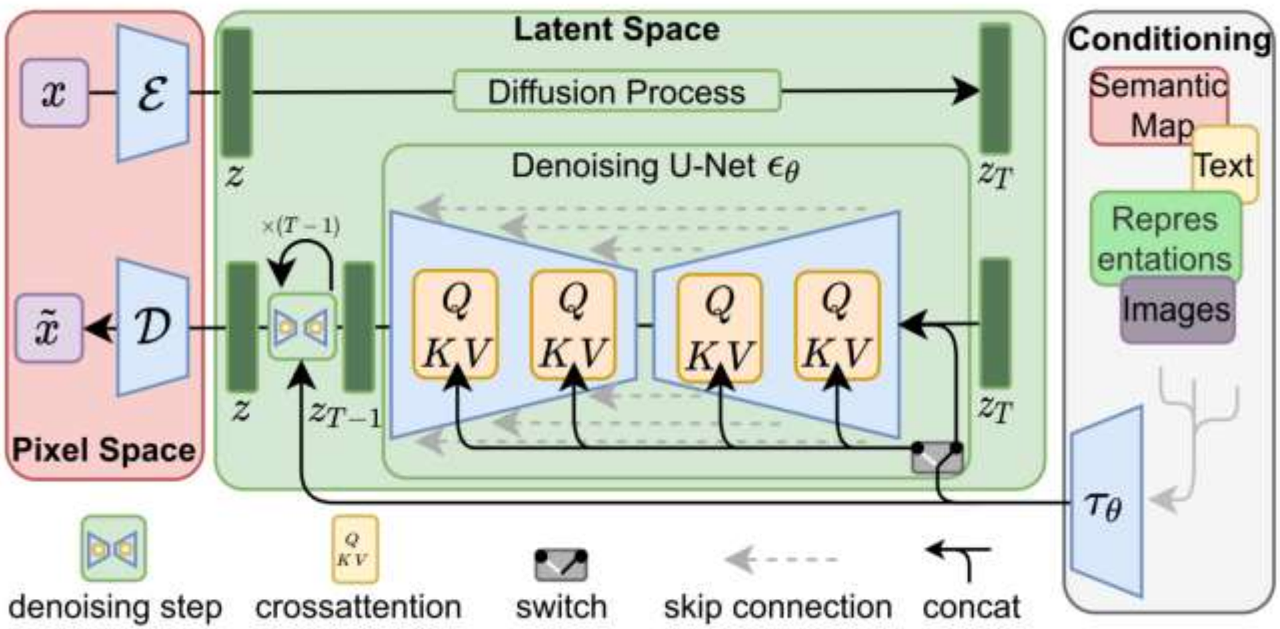
Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models", CVPR 2022. https://arxiv.org/abs/2112.10752

Latent Diffusion Models

- Training models in the pixel space is excessively computationally expensive (can easily multiple days on a V100 GPU)
  - Even image synthesis is very slow compared to GANs
  - Images are high dimensional → more things to model
- Researchers observed that most “bits” of an image contribute to its perceptual characteristics since aggressively compressing it usually maintains its semantic and conceptual composition
  - In layman’s terms, there are more bits for describing pixel-level details while less bits for describing “the meaning” within an image
  - Generative models should learn the latter
- Can we separate these two components?

Huge success of text-2-img!

- Stable Diffusion model (CVPR2022)



High-Resolution Image Synthesis with Latent Diffusion Models

Robin Rombach[1 *] Andreas Blattmann[1 *] Dominik Lorenz[1] Patrick Esser[R] Björn Ommer[1]
[1]Ludwig Maximilian University of Munich & IWR, Heidelberg University, Germany [R]Runway ML
https://github.com/CompVis/latent-diffusion

Latent Diffusion Models

Latent Diffusion Models can be divided **into two stages**:

1）Training perceptual compression models that strip away irrelevant high-level details and learn a latent space that is semantically equivalent to the high level image pixel-space

a. The loss is a combination of a reconstruction loss, an adversarial loss (remember GANs?) that promotes high quality decoder reconstruction, and regularization terms

$$L_{\text{Autoencoder}} = \min_{\mathcal{E},\mathcal{D}} \max_{\psi} \Big(L_{rec}(x, \mathcal{D}(\mathcal{E}(x))) - L_{adv}(\mathcal{D}(\mathcal{E}(x))) + \log D_{\psi}(x) + L_{reg}(x; \mathcal{E}, \mathcal{D}) \Big)$$

2）Performing a diffusion process *in this latent space*. There are several benefits to this:

a. The diffusion process is only focusing on the relevant semantic bits of the data
b. Performing diffusion in a low dimensional space is significantly more efficient

Huge success of text-2-img!

- Stable Diffusion model (CVPR2022)
- Long story between Stability AI, Runway ML and LAION-5B
- AI paradigm: data + algorithm + computing resource

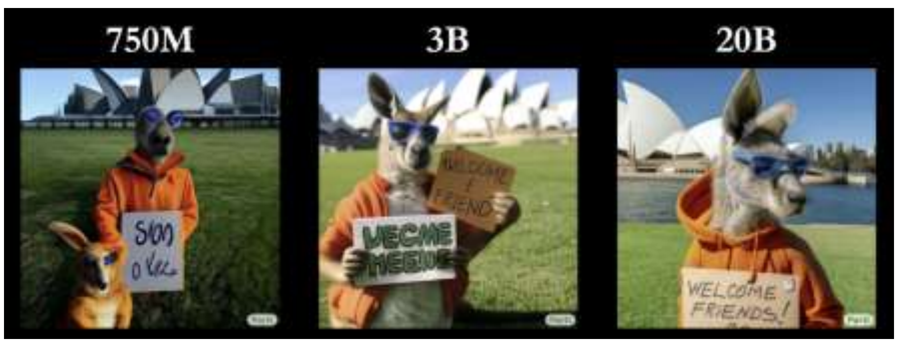
Computer Vision & Learning Group
Ludwig Maximilian University of Munich (LMU)

~4000 A100 from Stability AI

Huge text-image dataset from LAION

Huge success of text-2-img!

- The multi-modality framework is important
- The trend continues: big data, big modal ……
- Enjoy better text-encoder and suitable generator



Google: Parti Model, "Scaling Autoregressive Models for Content-Rich Text-to-Image Generation"

Stablity AI: DeepFloyd IF Model

T5-XXL as Text-encoder; pixel-level Diffusion

Huge success of text-2-img!

- Enjoy cross-modality abilities
- Enjoy downstream conditioning abilities

Subject-Driven Generation using **DreamBooth**

Conditioning using **ControlNet**

Editing Instructions using **InstructPix2Pix** (based on GPT-3)

Huge success of text-2-img-3D!

- Use NeRF as inherent representation to bridge 2D-DM with 3D scene
- More explicit disentanglement towards geometry, color, lighting …

3D Editing Instructions
using **InstructNeRF2NeRF**

NVDIA **Magic3D** 18 Nov 2022

OpenAI **Point-E** 21 Dec 2022

Summary

- Preliminary theory of diffusion (don’t worry if this is confusing!)
- Some tricks that modern diffusion models employ for image generation:
  - A U-Net architecture equipped with all kinds of modifications
  - Other architecture improvements
  - Several implementation tricks (different noise schedules, covariance parametrizations)
- Latent diffusion models for improving diffusion quality and efficiency