Diffusion Model 2

CSE 849 Deep Learning
Spring 2025

Zijun Cui

Slides are revised based on Deep Learning CMU by Hao Chen, cs231n Stanford by Ruiqi Gao, and online resources

Content

- Diffusion Model Basics
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Variants of Diffusion Models
  - Denoising Diffusion Implicit Model (DDIM)
  - Latent Diffusion Models
  - Conditional Diffusion Models
- Applications of Diffusion Models

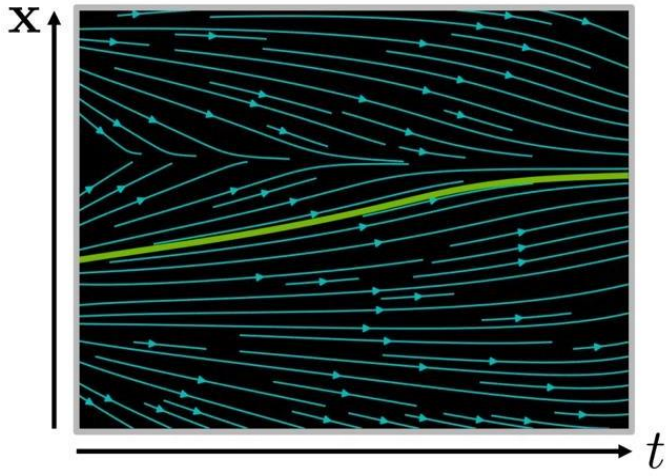
Why SDEs?

- A unified framework for interpreting diffusion models and score-based generation models
  - Variants of diffusion-based and flow-based models

Stochastic Differential Equations

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

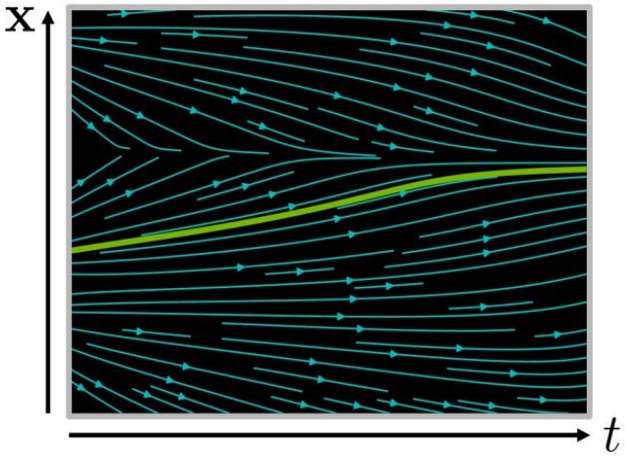
Iterative Numerical $\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

Stochastic Differential Equations

Ordinary Differential Equation (ODE):

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



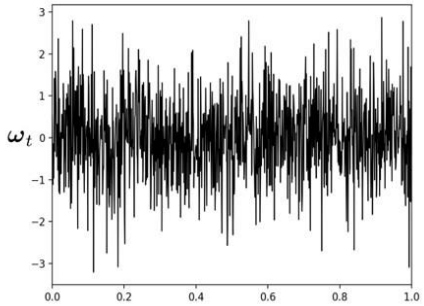
Analytical Solution: $\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}, \tau)\mathrm{d}\tau$

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

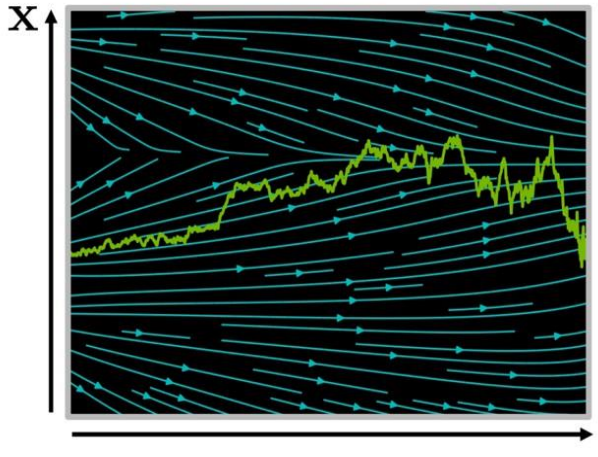
Stochastic Differential Equation (SDE):

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

Wiener Process (Gaussian White Noise)



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

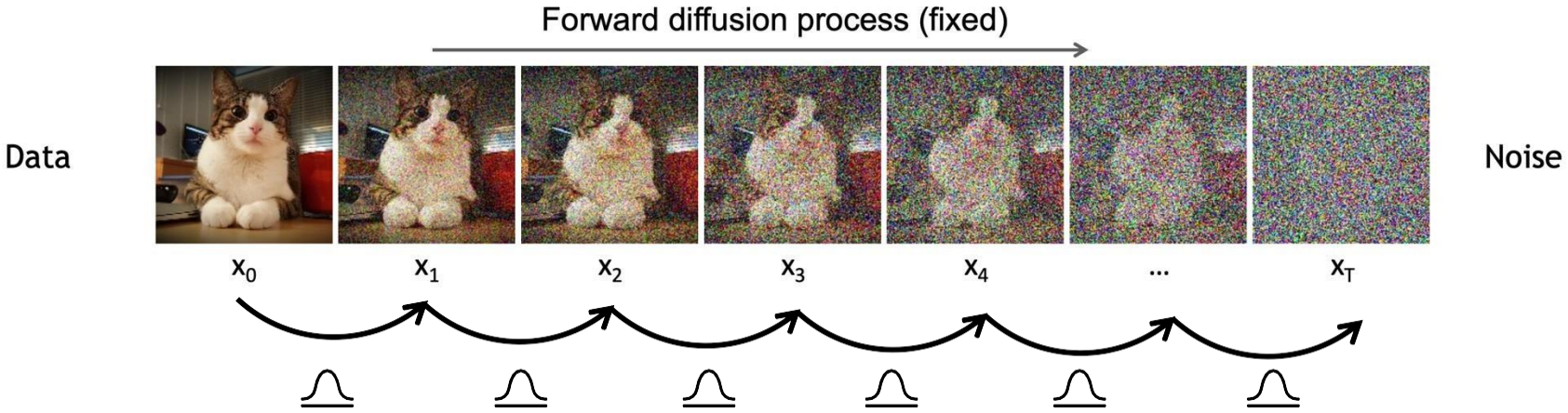


$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t + \sigma(\mathbf{x}(t), t)\sqrt{\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

with $\omega_t \approx \frac{1}{\sqrt{\Delta t}} \cdot \mathcal{N}(0, I)$

Continuous-time diffusion models

Stochastic differential equation framework



- Consider a forward process with many many small steps (still discrete)

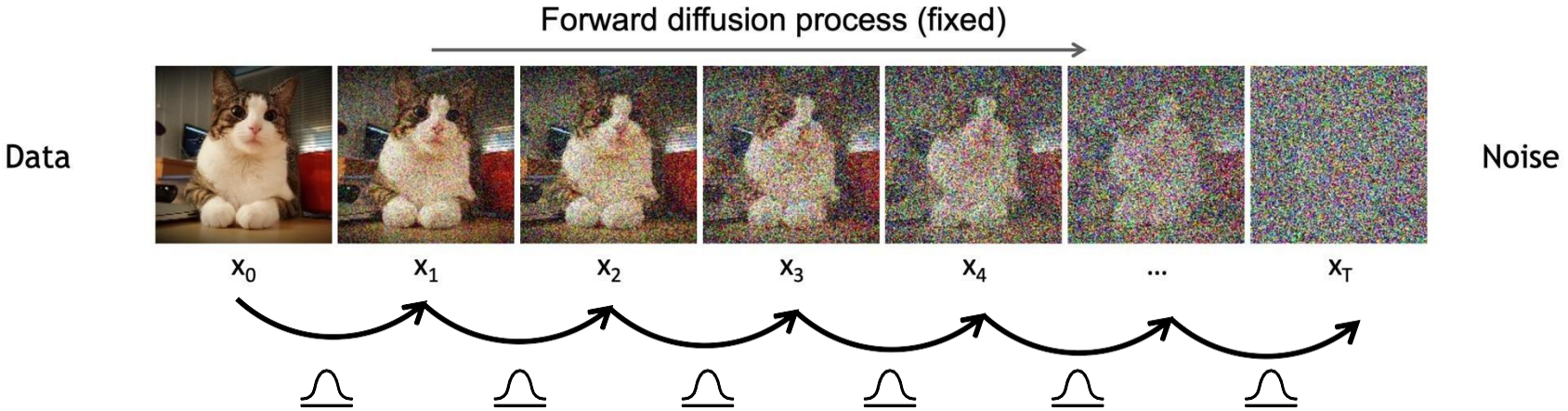
$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t \mathbf{I}\right)$$

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathcal{N}(\mathbf{0}, \mathbf{I})$$

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

Continuous-time diffusion models

Stochastic differential equation framework



- Consider a forward process with many many small steps

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

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathcal{N}(\mathbf{0}, \mathbf{I})$$

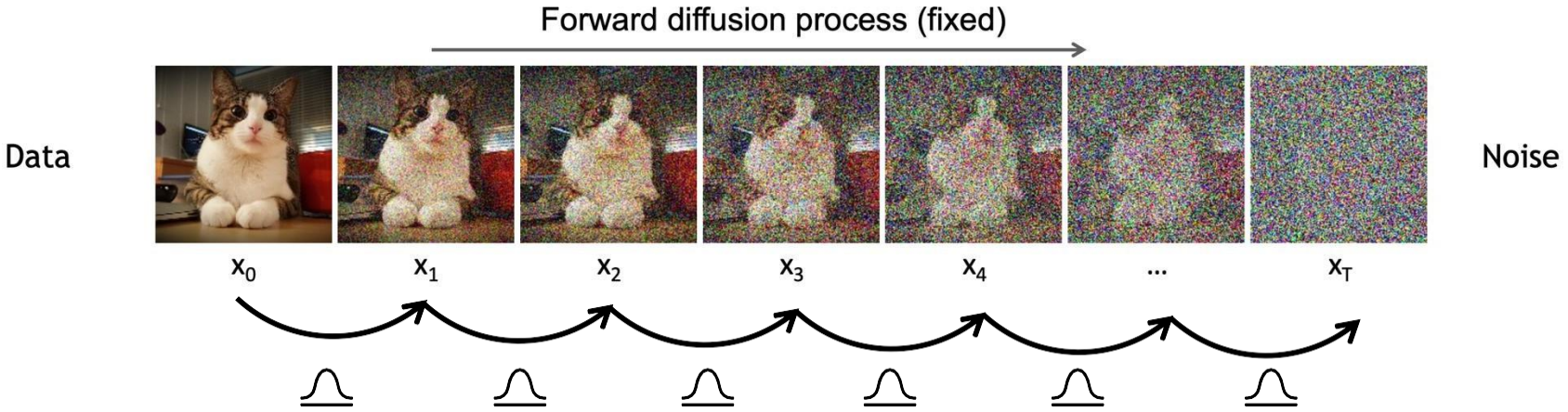
$$= \sqrt{1-\beta(t)\Delta t}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\mathcal{N}(\mathbf{0}, \mathbf{I}) \qquad (\beta_t := \beta(t)\Delta t)$$

Allows different size along t

Step size

Continuous-time diffusion models

Stochastic differential equation framework



- Consider a forward process with many many small steps

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

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathcal{N}(\mathbf{0}, \mathbf{I})$$

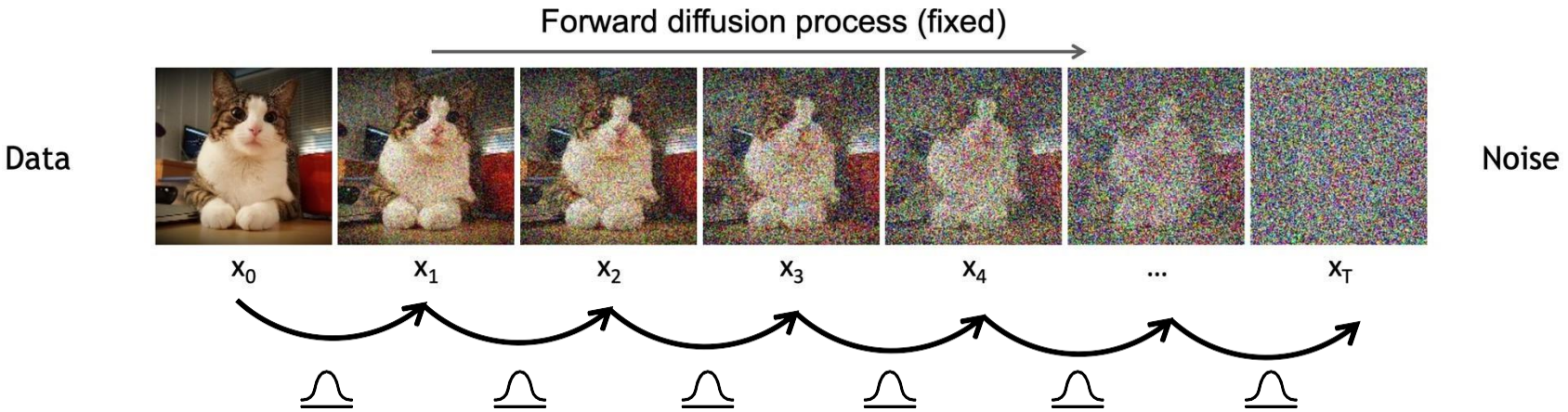
$$= \sqrt{1-\beta(t)\Delta t}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\mathcal{N}(\mathbf{0}, \mathbf{I}) \qquad (\beta_t := \beta(t)\Delta t)$$

$$\approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\mathcal{N}(\mathbf{0}, \mathbf{I}) \quad \text{(Taylor expansion } \sqrt{1-\beta(t)\Delta t} \approx 1 - \frac{1}{2}\beta(t)\Delta t\text{)}$$

Note: $\sqrt{1-x} \approx 1 - \frac{1}{2}x \quad \text{when } x \ll 1$

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

Forward Diffusion Process as SDEs



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

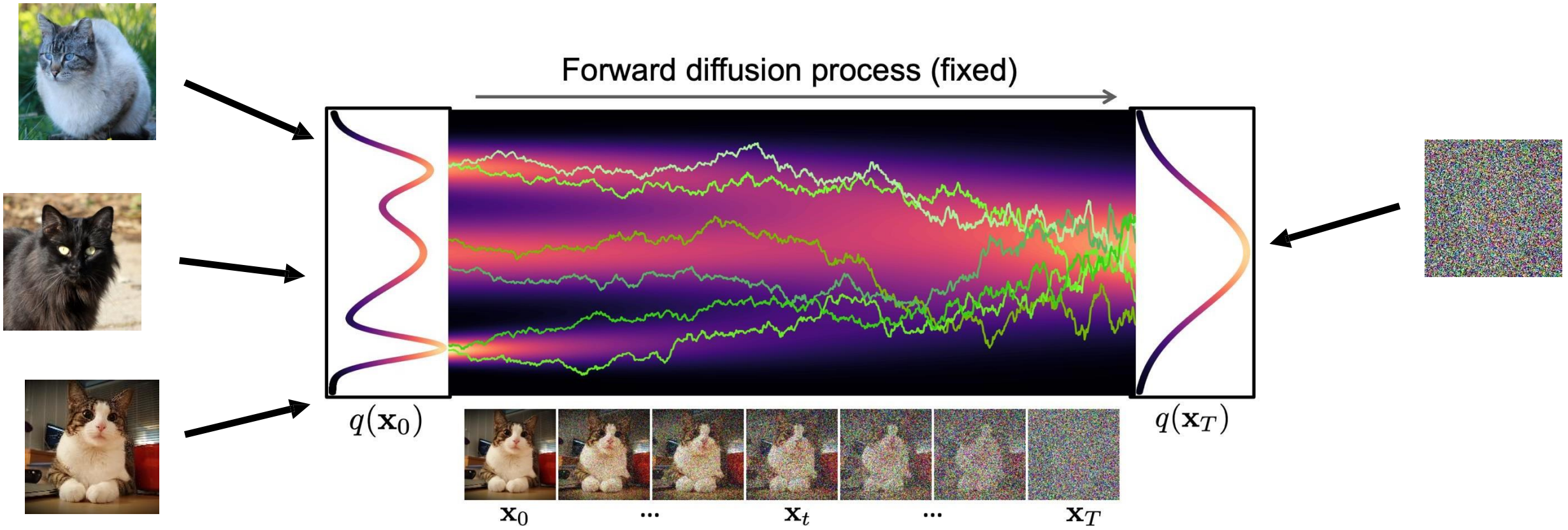
$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\boldsymbol{\omega}_t$$

describing the diffusion in infinitesimal limit
i.e., continuous time
Stochastic Differential Equation (SDE)

An iterative update that can be viewed as SDEs

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

Forward Diffusion Process as SDEs



Forward Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t}_{\text{Drift Term}} + \underbrace{\sqrt{\beta(t)}\mathrm{d}\boldsymbol{\omega}_t}_{\text{Diffusion Term}}$$

Drift Term (Pulls toward the mode)

Diffusion Term (Injects Noise)

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

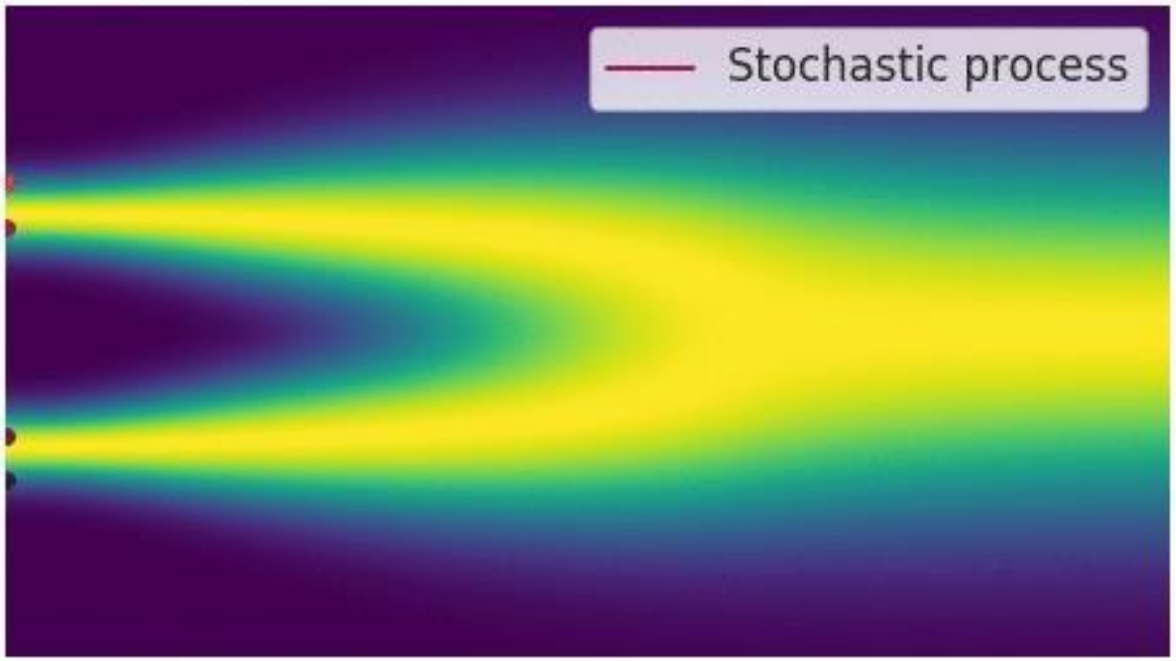


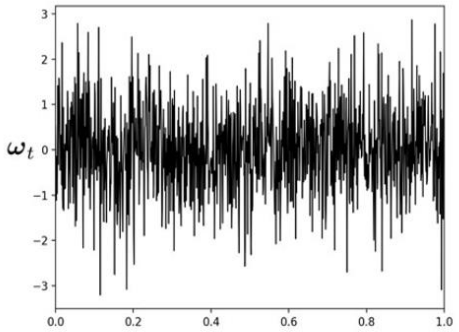
Figure credit to: https://yang-song.net/blog/2021/score/

Stochastic Differential Equations

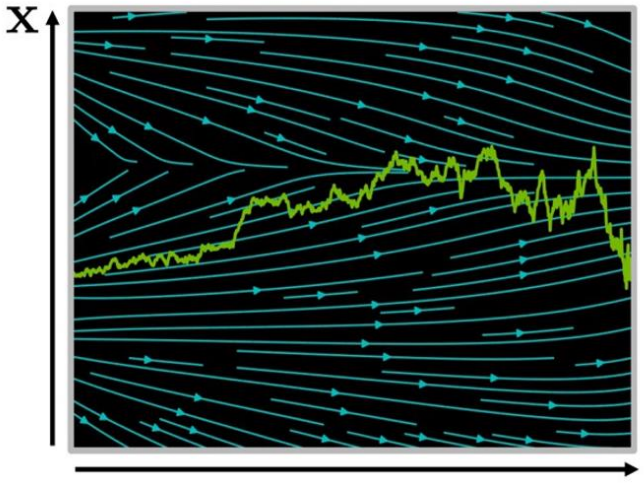
Stochastic Differential Equation (SDE):

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

Wiener Process (Gaussian White Noise)



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



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

Given an SDE

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\, dt + \boldsymbol{\sigma}(\mathbf{x}, t)\, d\omega_t$$

We have the **reverse-time SDE**

$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \boldsymbol{\sigma}(\mathbf{x}, t)\boldsymbol{\sigma}^\top(\mathbf{x}, t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] dt + \boldsymbol{\sigma}(\mathbf{x}, t)\, d\bar{\omega}_t$$

- $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is the **score function**
- $d\bar{\omega}_t$ is a **reverse-time Wiener process**
- $\boldsymbol{\sigma}\boldsymbol{\sigma}^\top$ accounts for the diffusion covariance

Stochastic Differential Equations

Given an SDE $$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\, dt + \boldsymbol{\sigma}(\mathbf{x}, t)\, d\omega_t$$

We have the **reverse-time SDE**

$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \boldsymbol{\sigma}(\mathbf{x}, t)\boldsymbol{\sigma}^{\top}(\mathbf{x}, t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] dt + \boldsymbol{\sigma}(\mathbf{x}, t)\, d\bar{\omega}_t$$

$p_t(\mathbf{x})$ = density of the solution $\mathbf{x}_t$ at time t

This gradient tells us in which direction should I perturb $\boldsymbol{x}_t$ to move to high-density

We call it score function, and the reverse drift can be treated as:

$$\text{drift}_{\text{reverse}} = \text{drift}_{\text{forward}} - \text{score-guided correction}$$

A quick detour: Score Function

- General form of probability density function

$$p_\theta(\mathbf{x}) = \frac{e^{-f_\theta(\mathbf{x})}}{Z_\theta}$$

  - This is an energy-based model (EBM).
  - $f_\theta(\boldsymbol{x})$ is the energy function. $Z_\theta = \int e^{-f_\theta(\boldsymbol{x})}$ is the partition function, often intractable.

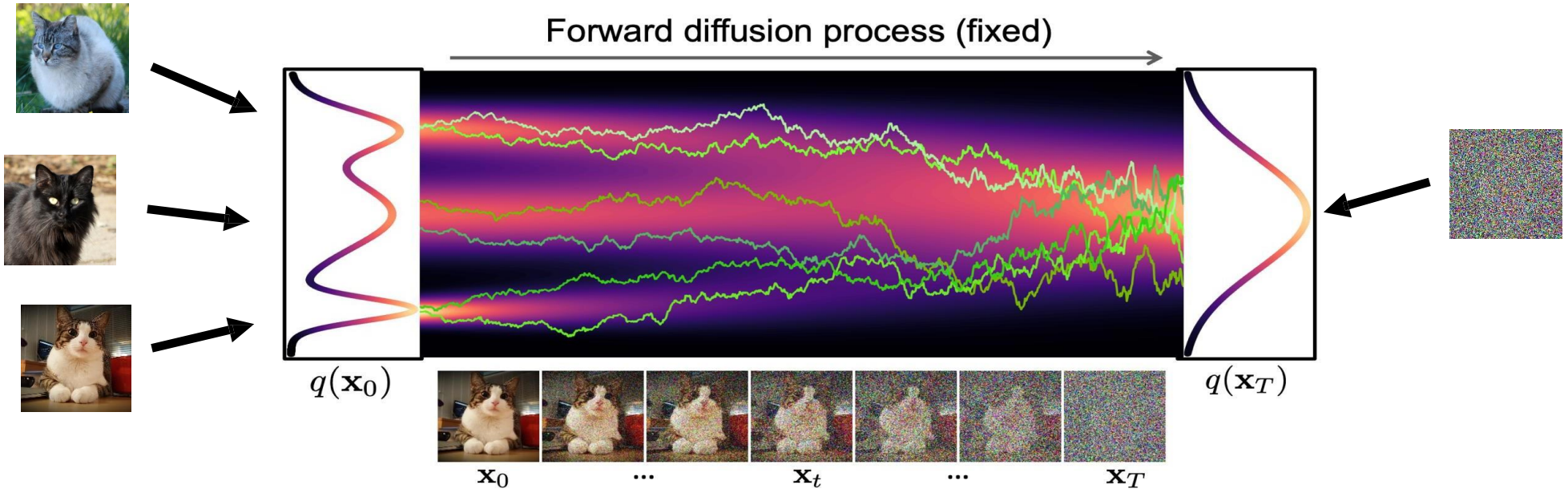
- Log-likelihood is hard: $\log p_\theta(\mathbf{x}) = -f_\theta(\mathbf{x}) - \log Z_\theta$
  - Because of Z_θ

- Instead of modeling $p_\theta(\boldsymbol{x})$, we can model its gradient $\nabla_{\mathbf{x}} \log p(\mathbf{x})$
  - It avoids calculating Z_θ because:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}) = -\nabla_{\mathbf{x}} f_\theta(\mathbf{x})$$

This gradient $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ is called score function

The Generative Reverse Stochastic Differential Equation



Forward Diffusion SDE:

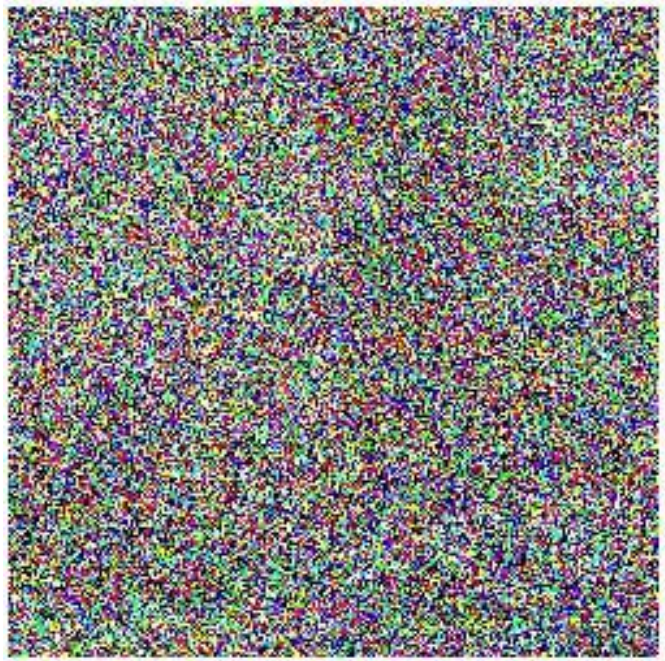
$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

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!**

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/



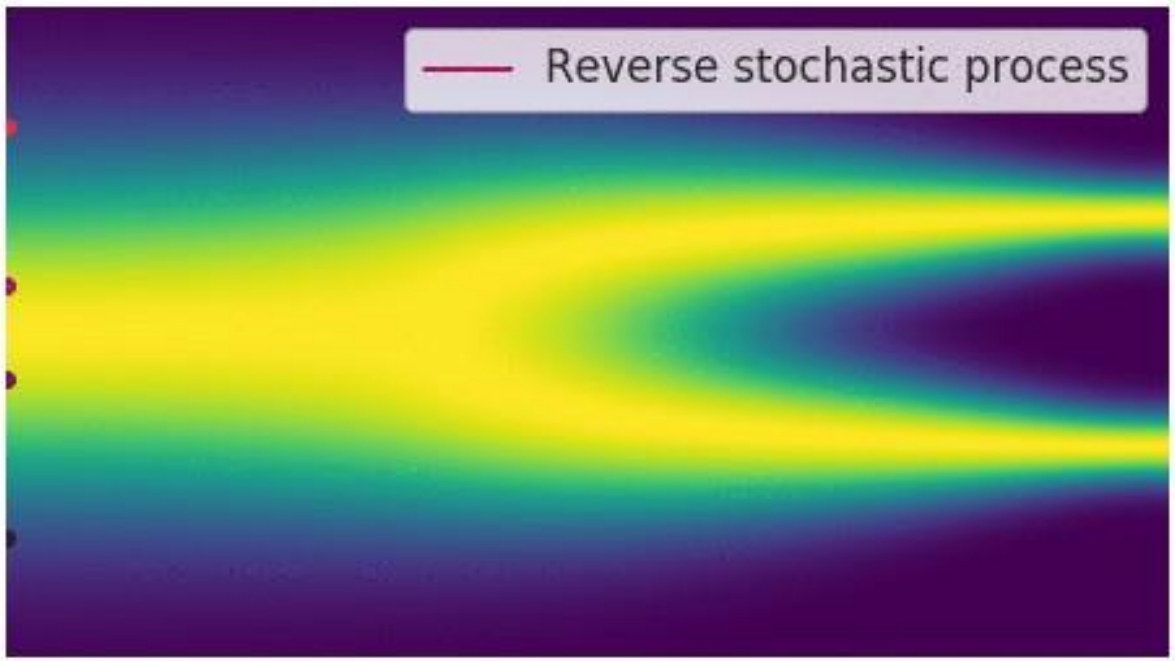
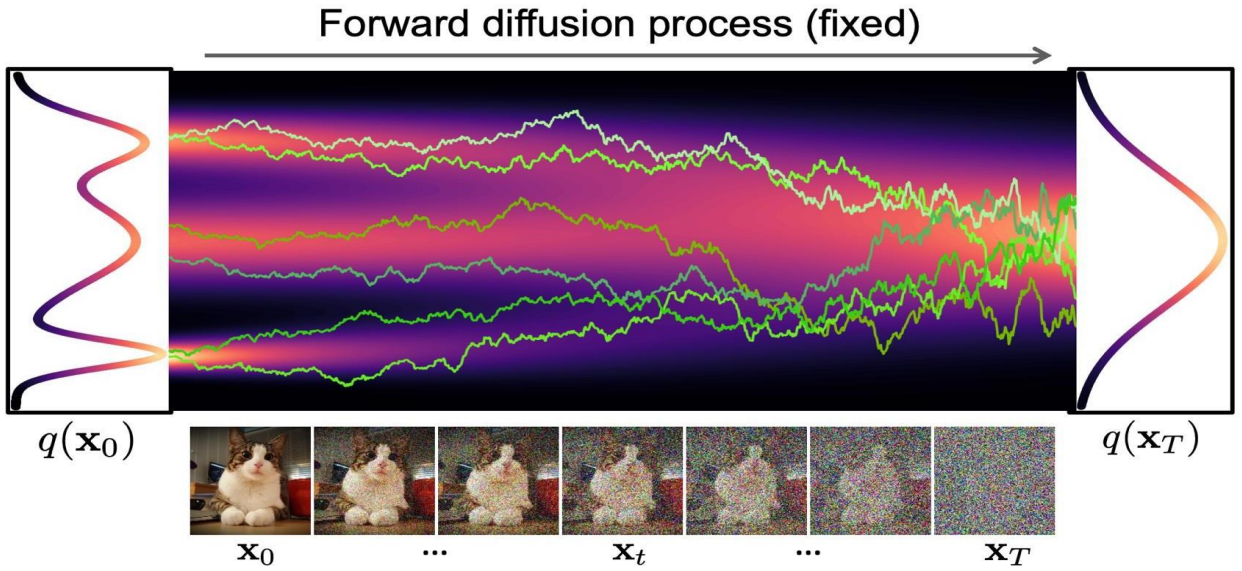


Figure credit to: https://yang-song.net/blog/2021/score/

Generative Reverse SDEs



- The forward SDE has a reverse form:

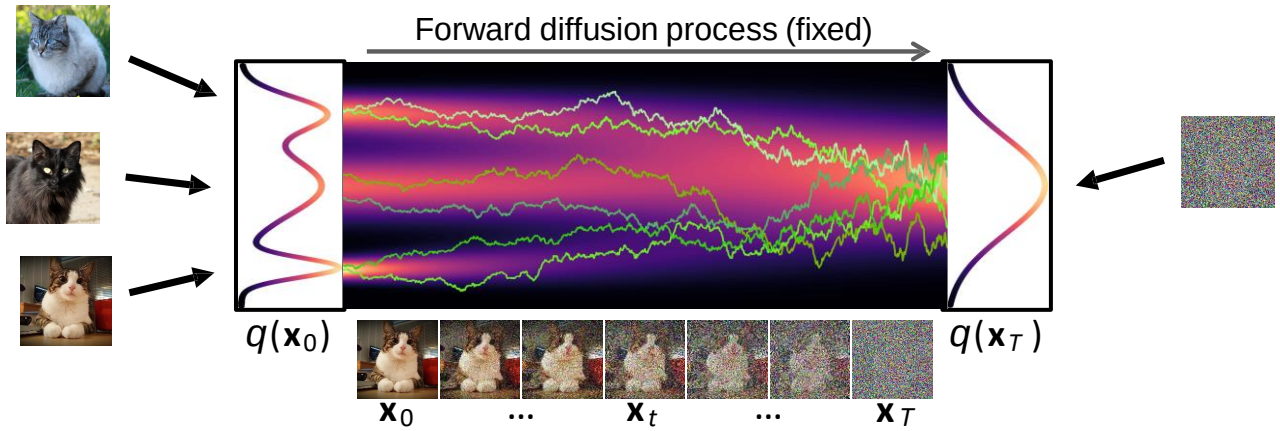
$$\mathrm{d}\mathbf{x}_t = \left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\nabla_{\mathbf{x}_t}\log q_t\left(\mathbf{x}_t\right)\right]\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\overline{\boldsymbol{\omega}}_t$$

How to get the score function?

Score matching is the typical and foundational strategy used in diffusion models to learn the score function

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

Score Matching



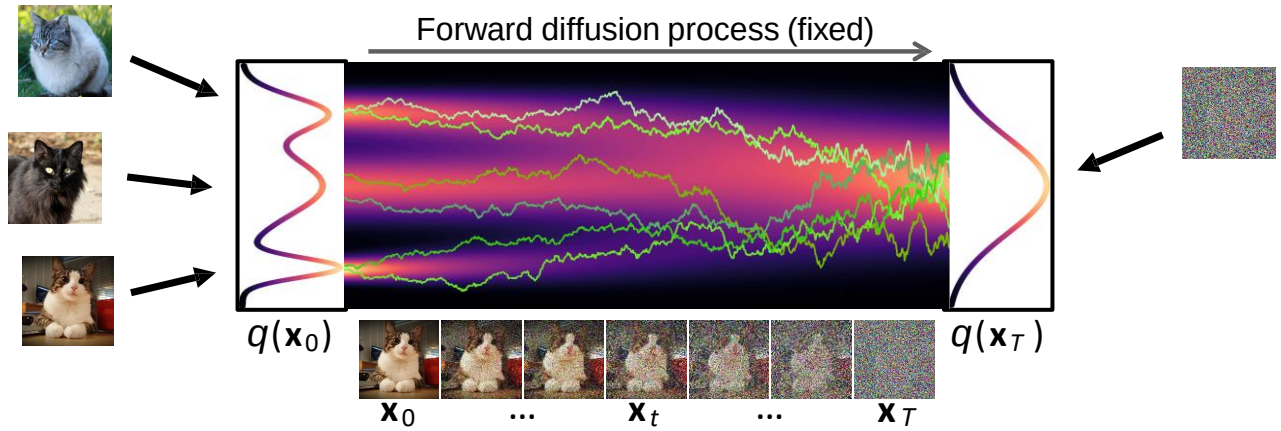
- Naïve idea, learn model for the score function by direct regression?

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

➡ **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!**

Vincent, "A Connection Between Score Matching and Denoising Autoencoders", *Neural Computation*, 2011
Song and Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution", *NeurIPS*, 2019
slide from https://cvpr2022-tutorial-diffusion-models.github.io/

Denoising Score Matching



- Instead, diffuse individual data points $\mathbf{x}_0$. Diffused $q_t(\mathbf{x}_t|\mathbf{x}_0)$ ***is*** tractable!
- **Denoising Score Matching:**

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

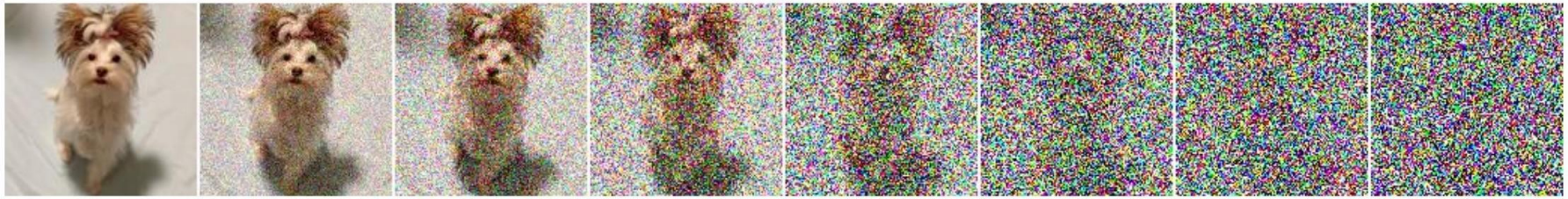
➡ **After expectations,** $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$**!**

Vincent, in *Neural Computation*, 2011
Song and Ermon, *NeurIPS*, 2019
Song et al. *ICLR*, 2021
slide from https://cvpr2022-tutorial-diffusion-models.github.io/

Denoising Score Matching

Forward SDE (data → noise)

$\mathbf{x}(0)$ —— $\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$ ——→ $\mathbf{x}(T)$



score function

$\mathbf{x}(0)$ ←—— $\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$ —— $\mathbf{x}(T)$

Reverse SDE (noise → data)

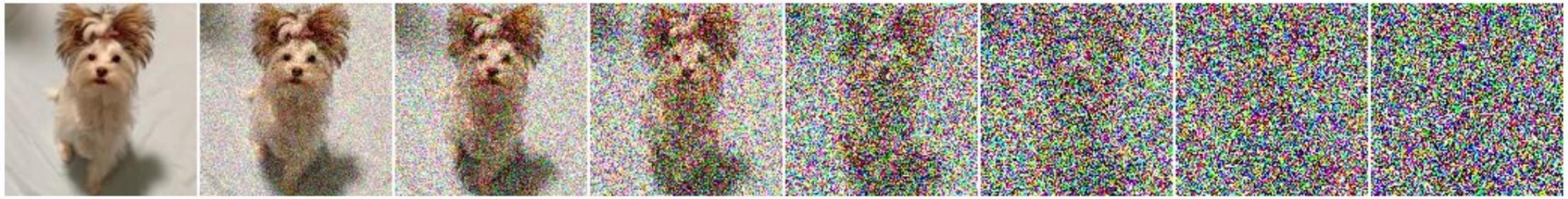
Figure credit to: https://yang-song.net/blog/2021/score/

Denoising Score Matching

Forward SDE (data → noise)

$\mathbf{x}(0)$ — $\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$ → $\mathbf{x}(T)$

$-\frac{1}{2}\beta(t)\mathbf{x}_t$ $\quad$ $\sqrt{\beta(t)}$



score function

$\mathbf{x}(0)$ ← $\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$ — $\mathbf{x}(T)$

Reverse SDE (noise → data)

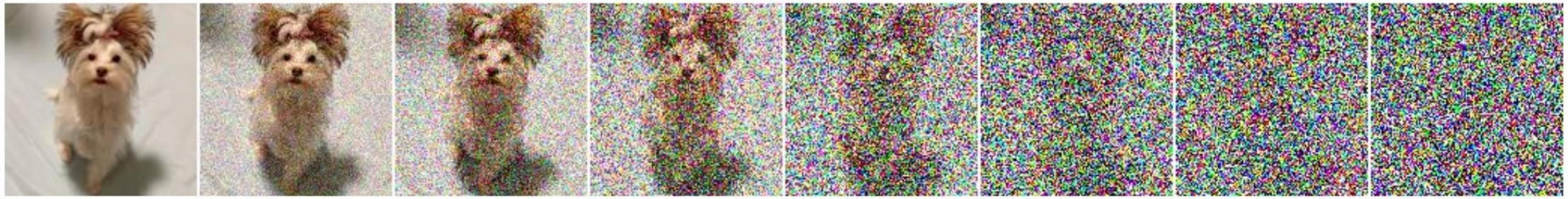
$$\mathrm{d}\mathbf{x}_t = \left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)\right]\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\bar{\boldsymbol{\omega}}_t$$

Figure credit to: https://yang-song.net/blog/2021/score/

Denoising Score Matching

Forward SDE (data → noise)

$\mathbf{x}(0)$ — $\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$ → $\mathbf{x}(T)$



score function

$\mathbf{x}(0)$ ← $\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$ — $\mathbf{x}(T)$

Reverse SDE (noise → data)

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

Figure credit to: https://yang-song.net/blog/2021/score/

Denoising Score Matching

- Denoising score matching objective

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

- Re-parametrized sampling:

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

- Score function:

$$\nabla_{\mathbf{x}_t} \log q_t\left(\mathbf{x}_t \mid \mathbf{x}_0\right) = -\nabla_{\mathbf{x}_t} \frac{\left(\mathbf{x}_t - \alpha_t \mathbf{x}_0\right)^2}{2\sigma_t^2} = -\frac{\mathbf{x}_t - \alpha_t \mathbf{x}_0}{\sigma_t^2} = -\frac{\alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon} - \alpha_t \mathbf{x}_0}{\sigma_t^2} = -\frac{\boldsymbol{\epsilon}}{\sigma_t}$$

- Denoising network:

$$\mathbf{s}_{\boldsymbol{\theta}}\left(\mathbf{x}_t, t\right) := -\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\mathbf{x}_t, t\right)}{\sigma_t}$$

- Final objective:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)} \mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \hat{w}(t) \cdot \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\mathbf{x}_t, t\right) \right\|_2^2 \quad \hat{w}(t) = \frac{\tilde{w}(t)}{\sigma_t}$$

What is the ELBO for continuous-time diffusion models?

- Denoising Score Matching:

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

- [Kingma et al, 2021] showed that the variational upper bound (negative ELBO) can be reduced to a simple variant:

$$-\text{ELBO}(\theta) = \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}-\frac{d\lambda}{dt}\cdot||\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)||_2^2+\text{const}$$

$$\text{where } \mathbf{x}_t = \alpha_t\mathbf{x}_0+\sigma_t\boldsymbol{\epsilon}$$

$$\lambda = \log\frac{\alpha_t^2}{\sigma_t^2} \text{ (signal-to-noise ratio of timestep } t)$$

- In practice, different loss weightings trade off between model with good perceptual quality vs. high log-likelihood:
  - Perceptual quality, e.g., $\hat{w}(t)=1$
  - High likelihood: -ELBO

➡ How to explain such discrepancy?
Can we still trust ELBO / maximum likelihood?

Kingma et al., “Variational diffusion models”, NeurIPS 2021.

Weighted diffusion objectives

Understanding diffusion objectives as the ELBO with data augmentation

- Summarize different types of diffusion objectives as the following **weighted diffusion objective**:

$$\mathcal{L}_w(\boldsymbol{\theta}) = \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[w(t)\cdot -\frac{d\lambda}{dt}\cdot ||\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)||_2^2\right]$$

- [Kingma & Gao, 2023] showed that if $w(t)$ is a monotonically increasing function, then the weighted diffusion objective is equivalent to the **negative ELBO with data augmentation** (Gaussian addition noise):

$$\mathcal{L}_w(\boldsymbol{\theta}) \geq \underbrace{\mathbb{E}_{t\sim p_w(t)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)}\left[-\log p(\mathbf{x}_t)\right]}_{\text{Neg. log lik. of noise-perturbed data}} + \text{const}$$

$$\text{where } p_w(t) \propto \frac{dw(t)}{dt}$$

- **Takeaway:** By using a certain weighting function w(t) (monotonically increasing) during training, you're essentially performing maximum likelihood estimation on a noise-perturbed data distribution.

- Therefore, the ELBO objective is compatible with perceptual quality when combined with simple data augmentation (additive noise)

Kingma & Gao, “Understanding Diffusion Objectives\\as the ELBO with Data Augmentation”, 2023.

Weighted Diffusion Objective

- Denoising score matching objective with loss weighting in general

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

- Loss weights trade-off between
  - good perceptual quality: $\lambda(t)=\sigma_t^2$
  - maximum likelihood: $\lambda(t)=\beta(t)$

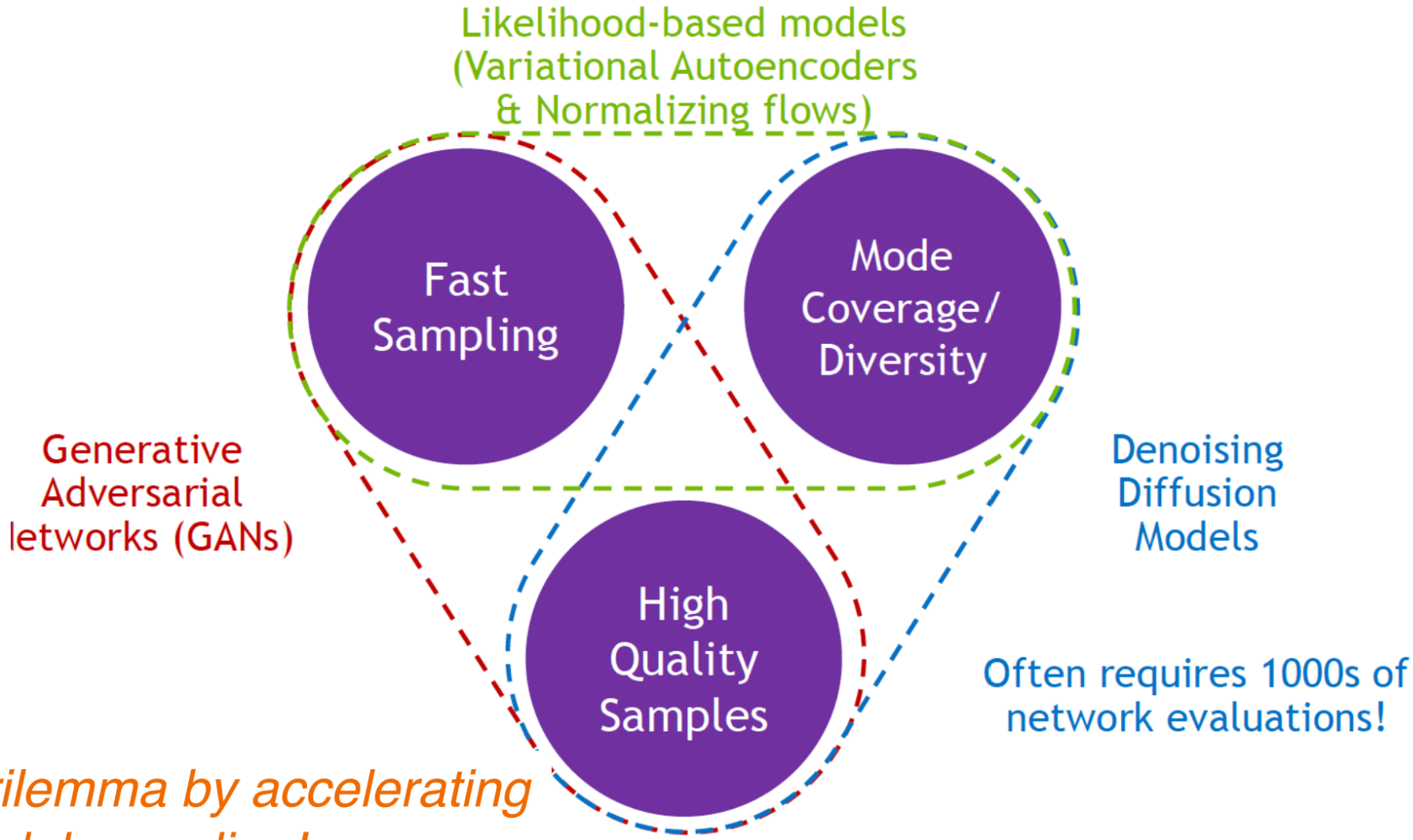
- More complicated model parametrization and loss weighting leads to different diffusion model variants in the literature!

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

Content

- Diffusion Model Basics
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Variants of Diffusion Models
  - Denoising Diffusion Implicit Model (DDIM); Distillation guided Diffusion
  - Latent Diffusion Models
  - Conditional Diffusion Models
- Applications of Diffusion Models

The generative learning trilemma



Tackle the trilemma by accelerating diffusion model sampling!

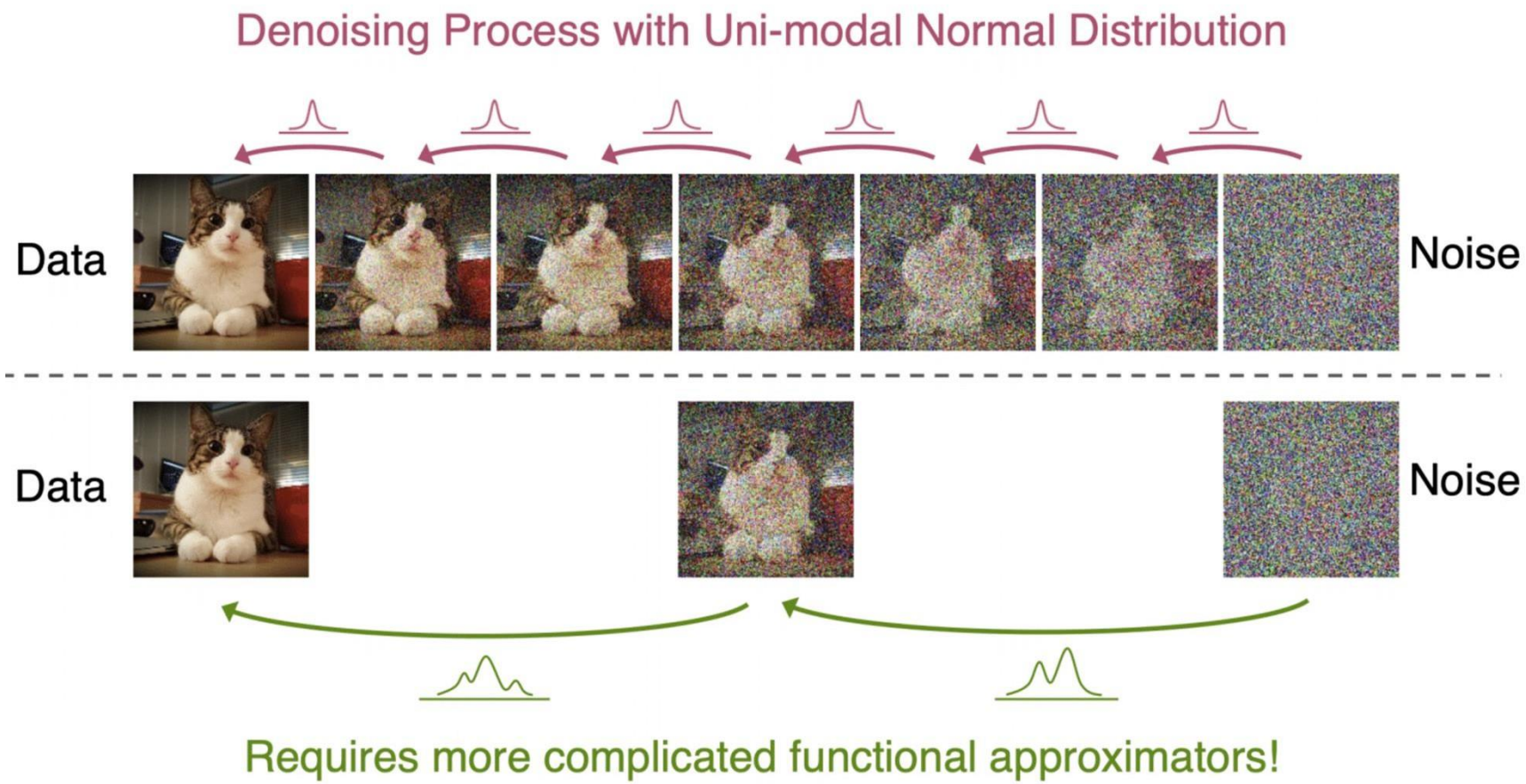
How to make sampling faster?

- One bottleneck of diffusion models is its slowness in sampling: need 10-1000+ steps to generate high quality samples
- Generative models need to be fast for practical use.

- Solutions:
  - Denoising Diffusion Implicit Model (DDIM) (ICLR 2021)
  - Distill diffusion models into models (using just 4-8 sampling steps)
    - *Progressive distillation for fast sampling of diffusion models, Salimans & Ho, ICLR 2022*
    - *On Distillation of Guided Diffusion Models, Meng et al., CVPR 2023*

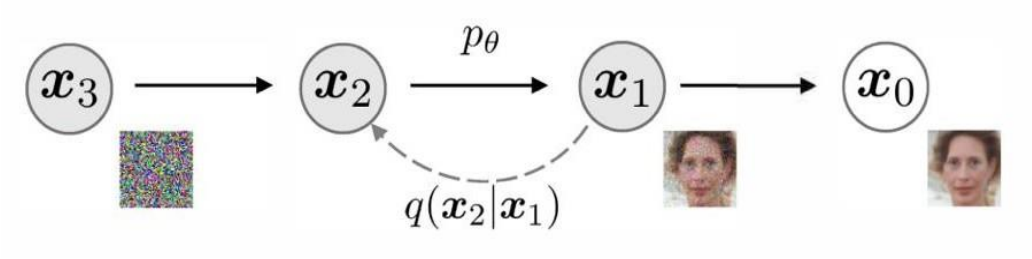
Can we do generation with less steps?

Denoising Diffusion Implicit Model



Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

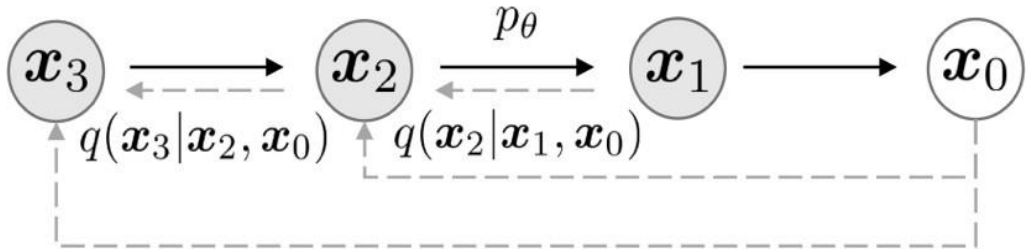
DDPM



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

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

DDIM



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

$$q_\sigma\left(\boldsymbol{x}_T \mid \boldsymbol{x}_0\right)=\mathcal{N}\left(\sqrt{\alpha_T} \boldsymbol{x}_0,\left(1-\alpha_T\right) \boldsymbol{I}\right)$$

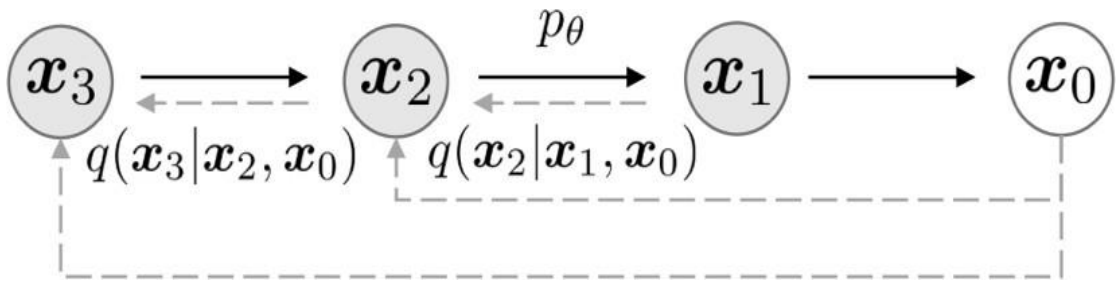
- A Non-Markovian Forward Process

$$q_\sigma\left(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_0\right)=\frac{q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right) q_\sigma\left(\boldsymbol{x}_t \mid \boldsymbol{x}_0\right)}{q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0\right)}$$

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

Song et al. Denoising Diffusion Implicit Models. 2021.

DDIM



- Reverse process

$$p_\theta^{(t)}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \begin{cases} \mathcal{N}\left(f_\theta^{(1)}(\boldsymbol{x}_1), \sigma_1^2 \boldsymbol{I}\right) & \text{if } t = 1 \\ q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, f_\theta^{(t)}(\boldsymbol{x}_t)\right) & \text{otherwise,} \end{cases}$$

$$f_\theta^{(t)}(\boldsymbol{x}_t) := \left(\boldsymbol{x}_t - \sqrt{1-\alpha_t} \cdot \epsilon_\theta^{(t)}(\boldsymbol{x}_t)\right) / \sqrt{\alpha_t}$$

Trying to estimate $\boldsymbol{x}_0$

Song et al. Denoising Diffusion Implicit Models. 2021.

DDPM vs DDIM

Algorithm DDPM Sampling

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

for all t from T to 1 **do**

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

$$\mu \leftarrow \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t))$$

$\mathbf{x}_{t-1} \leftarrow \mu + \sigma_t \epsilon$ Stochastic

end for

return $\mathbf{x}_0$

Algorithm DDIM Sampling

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

for all t from T to 1 **do**

$\bar{\epsilon} \leftarrow \epsilon_\theta(\mathbf{x}_t, t)$

$$\bar{\mathbf{x}}_0 \leftarrow \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}}{\sqrt{\bar{\alpha}_t}}$$ Estimate $\mathbf{x}_0$

$\mathbf{x}_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\bar{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\bar{\epsilon}$

end for

return $\mathbf{x}_0$

DDIM with Fewer Steps Sampling

DDIM

Algorithm Original DDIM Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for all t from T to 1 **do**
  $\bar{\epsilon} \leftarrow \epsilon_\theta(\mathbf{x}_t, t)$
  $\bar{\mathbf{x}}_0 \leftarrow \dfrac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}}{\sqrt{\bar{\alpha}_t}}$
  $\mathbf{x}_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\bar{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\bar{\epsilon}$
end for
return $\mathbf{x}_0$

Increasing
Sub-sequence

$$[1,\ldots,T] \Longrightarrow [\tau_0 = 0,\ldots,\tau_S = T]$$

E.g., $\tau = [0,10,20,30,\ldots,1000]$

Algorithm Fewer-Steps DDIM Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for all s from S to 1 **do**
  $t \leftarrow \tau_s$
  $t' \leftarrow \tau_{s-1}$
  $\bar{\epsilon} \leftarrow \epsilon_\theta(\mathbf{x}_t, t)$
  $\bar{\mathbf{x}}_0 \leftarrow \dfrac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}}{\sqrt{\bar{\alpha}_t}}$
  $\mathbf{x}_{t'} \leftarrow \sqrt{\bar{\alpha}_{t'}}\bar{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t'}}\bar{\epsilon}$
end for
return $\mathbf{x}_0$

DDIM Results

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

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

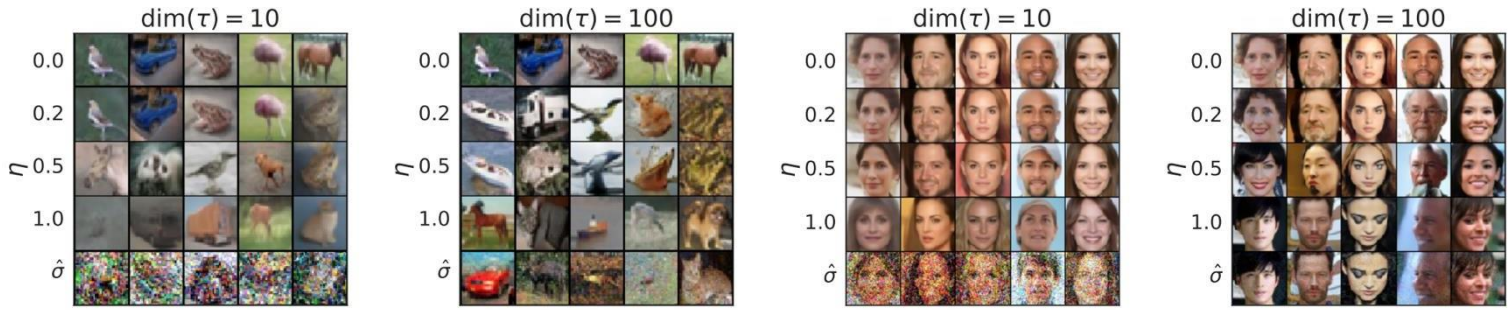
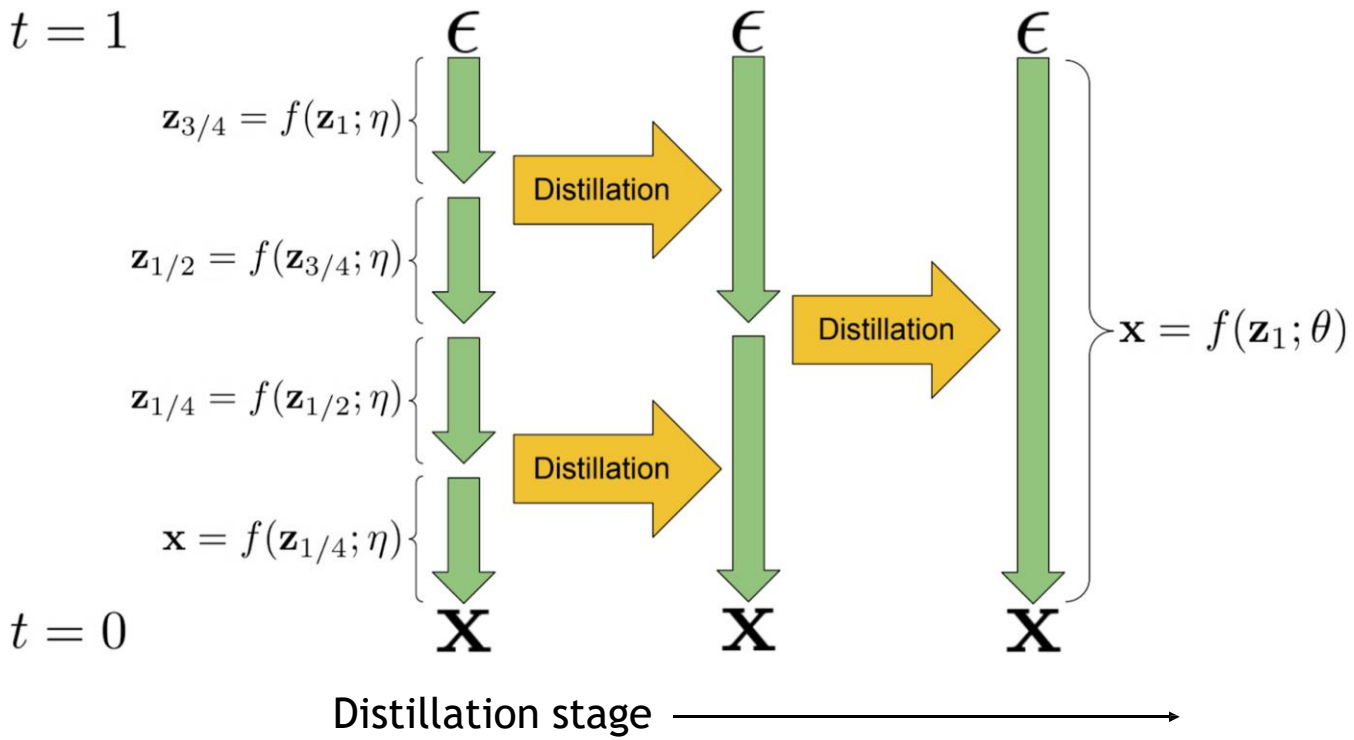


Figure 3: CIFAR10 and CelebA samples with $\dim(\tau) = 10$ and $\dim(\tau) = 100$.

Progressive distillation

How to make sampling faster?

- Distill a deterministic ODE sampler (i.e. DDIM 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.

On Distillation of Guided Diffusion Models

Meng et al., CVPR 2023 award nominated

Text-guided generation (1 step)

Text-guided generation (2 steps)

Class-conditional generation (1 step)

Now also works with

- CF-Guidance
- Stochastic sampling
- Text-to-image/video
- Image-to-image
- Inpainting
- Latent Diffusion

Text-guided generation (4 steps)

Input Mask Result 1 Result 2

Image inpainting (2 steps)

Input Output (different styles)

Image to image translation (3 steps)

Diffusion Model in Pixel Space

- Pros
  - Intuitive Understanding: Diffusion in pixel space directly affects image pixels, making the changes visually easy to understand.

- Cons
  - Computational Cost

    The larger the number of pixels, the greater the computation.
  - Memory Usage

    Handling high-resolution images requires substantial memory.

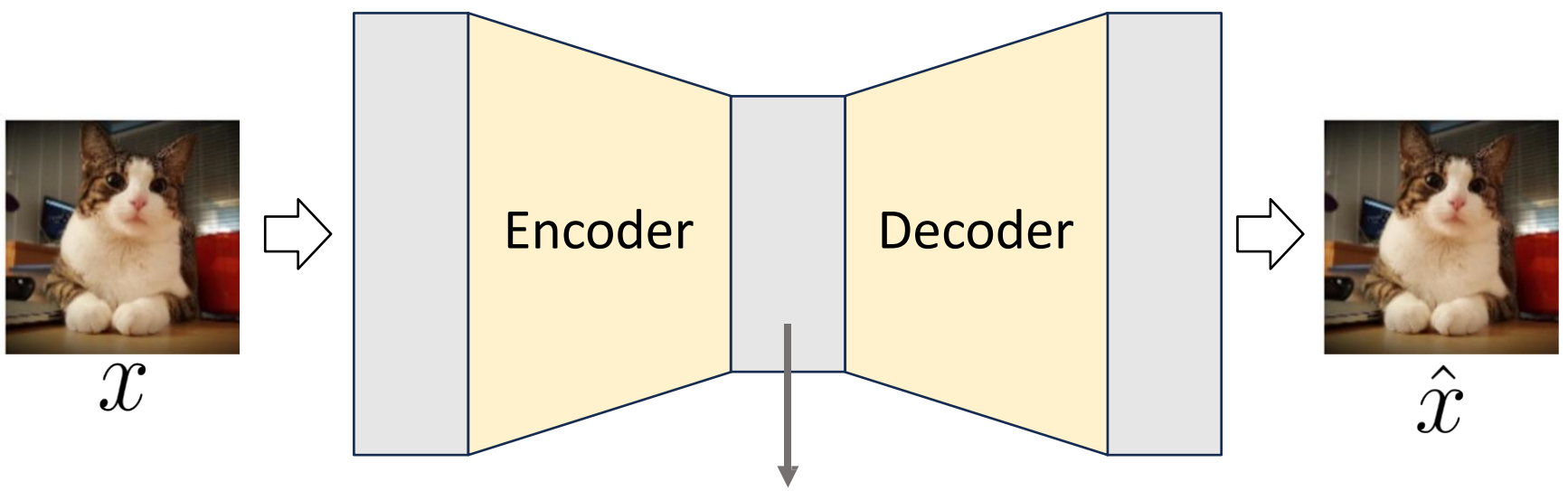
Latent Diffusion Model

- Latent spaces typically have lower dimensions than pixel spaces, resulting in lower computational costs.
  - Pixel Space >> Latent Space

- Runs the diffusion process in the latent space instead of pixel space

Latent Diffusion Model

- Autoencoders can be particularly valuable as they enable a compressed yet remaining semantic and conceptual meaning of an image.

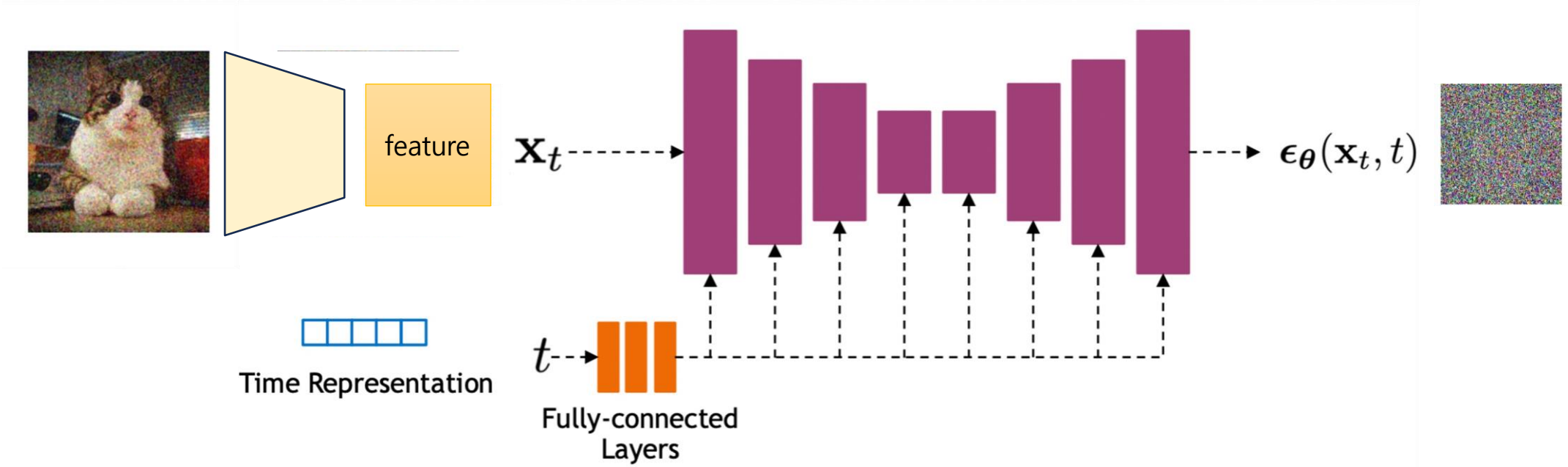


x $\hat{x}$

$$loss = \frac{1}{n}\sum_{i=0}^{n}(x_i - \hat{x}_i)^2$$

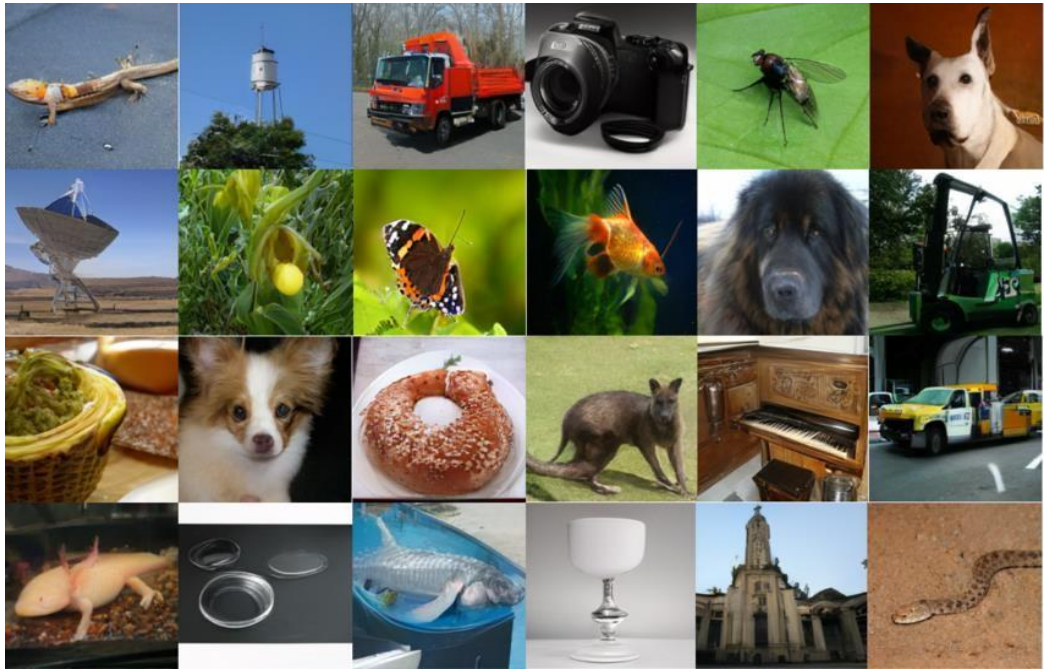
Latent Diffusion Model

- 2 Stage Training : Auto-Encoder + Latent Diffusion



Conditional Diffusion Models

- Un-conditional



$$p\left(\boldsymbol{x}_{0:T}\right)=p\left(\boldsymbol{x}_T\right) \prod_{t=1}^T p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t\right)$$

- Conditional

$$p\left(\boldsymbol{x}_{0:T} \mid y\right)=p\left(\boldsymbol{x}_T\right) \prod_{t=1}^T p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, y\right)$$

More controllable!

Conditional Score Matching

- Score matching with conditional information

$$
\begin{aligned}
\nabla \log p\left(\boldsymbol{x}_t \mid y\right) &= \nabla \log \left(\frac{p\left(\boldsymbol{x}_t\right) p\left(y \mid \boldsymbol{x}_t\right)}{p(y)} \right) \\
&= \nabla \log p\left(\boldsymbol{x}_t\right) + \nabla \log p\left(y \mid \boldsymbol{x}_t\right) - \nabla \log p(y) \\
&= \underbrace{\nabla \log p\left(\boldsymbol{x}_t\right)}_{\text{unconditional score}} + \underbrace{\nabla \log p\left(y \mid \boldsymbol{x}_t\right)}_{\text{adversarial gradient}}
\end{aligned}
$$

Classifier Guidance

- Use a discriminative classifier for $\nabla \log p\left(y \mid \boldsymbol{x}_t\right)$

$$\nabla \log p\left(\boldsymbol{x}_t \mid y\right) = \nabla \log p\left(\boldsymbol{x}_t\right) + \gamma \nabla \log p\left(y \mid \boldsymbol{x}_t\right)$$

- γ controls the strength of the condition
- Limitations:
  - Need a separate classifier
  - Conditioning depends on the performance of classifier

Dhariwal and Nichol, “Diffusion models beat GANs on image synthesis”, NeurIPS 2021.

Classifier-Free Guidance

- Score matching with conditional information

$$\nabla \log p\left(\boldsymbol{x}_t \mid y\right) = \nabla \log p\left(\boldsymbol{x}_t\right) + \gamma \nabla \log p\left(y \mid \boldsymbol{x}_t\right)$$

- By Bayes' rule we can define an implicit classifier

$$\nabla \log p\left(y \mid \boldsymbol{x}_t\right) = \nabla \log p\left(\boldsymbol{x}_t \mid y\right) - \nabla \log p\left(\boldsymbol{x}_t\right)$$

- Classifier-free guidance

$$\begin{aligned}
\nabla \log p\left(\boldsymbol{x}_t \mid y\right) &= \nabla \log p\left(\boldsymbol{x}_t\right) + \gamma\left(\nabla \log p\left(\boldsymbol{x}_t \mid y\right) - \nabla \log p\left(\boldsymbol{x}_t\right)\right) \\
&= \nabla \log p\left(\boldsymbol{x}_t\right) + \gamma \nabla \log p\left(\boldsymbol{x}_t \mid y\right) - \gamma \nabla \log p\left(\boldsymbol{x}_t\right) \\
&= \underbrace{\gamma \nabla \log p\left(\boldsymbol{x}_t \mid y\right)}_{\text{conditional score}} + \underbrace{(1-\gamma) \nabla \log p\left(\boldsymbol{x}_t\right)}_{\text{unconditional score}}
\end{aligned}$$

Ho et al. Classifier-Free Diffusion Guidance. 2022.

Training of Classifier-Free Guidance

- For conditional embeddings
  - Randomly drop **p** original conditionals with an additional unconditional class

Embedding of label y

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

Replace it with unconditional case with probability p, i.e.,

$$\epsilon_\theta(z_t,t,\varnothing)$$

Ho et al. Classifier-Free Diffusion Guidance. 2022.

Content

- Diffusion Model Basics
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Variants of Diffusion Models
  - Denoising Diffusion Implicit Model (DDIM)
  - Latent Diffusion Models
  - Conditional Diffusion Models
- Applications of Diffusion Models

Case study: Imagen

Imagen: text-to-image diffusion models

By Google (imagen.research.google)

Input: text; Output: 1kx1k images

- An unprecedented degree of photorealism
  - SOTA automatic scores & human ratings
- A deep level of language understanding
- Extremely simple
  - no latent space, no quantization

A brain riding a rocketship heading towards the moon.

Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

Imagen

By Google (imagen.research.google)

A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.

Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

Imagen

By Google (imagen.research.google)

A dragon fruit wearing karate belt in the snow.

Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

“toilet paper with real cactus spikes”

Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

What goes to Imagen?

Data

- Image-text pairs
- LAION-400M
- Internal (~500M images)

Model

- Diffusion models
- Cascading super-res
- Frozen text encoders

Sampler

- Classifier-free guidance
- Maximizing text-alignment

Scaling up

Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

What goes to Imagen?

Data

- Image-text pairs
- LAION-400M
- Internal (~500M images)

Model

- Diffusion models
- Cascading super-res
- Frozen text encoders

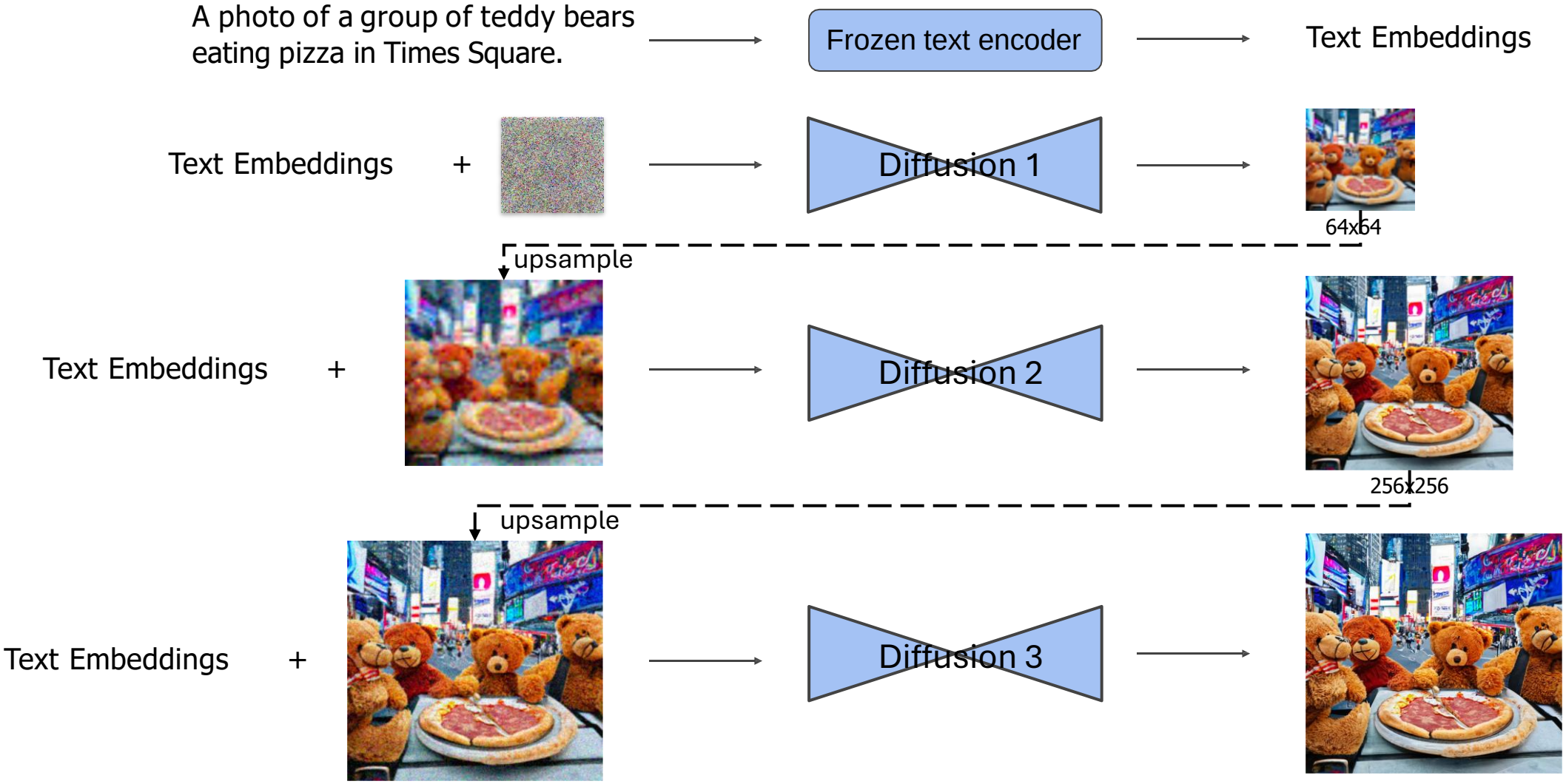
Sampler

- Classifier-free guidance
- Maximizing text-alignment

Scaling up

Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

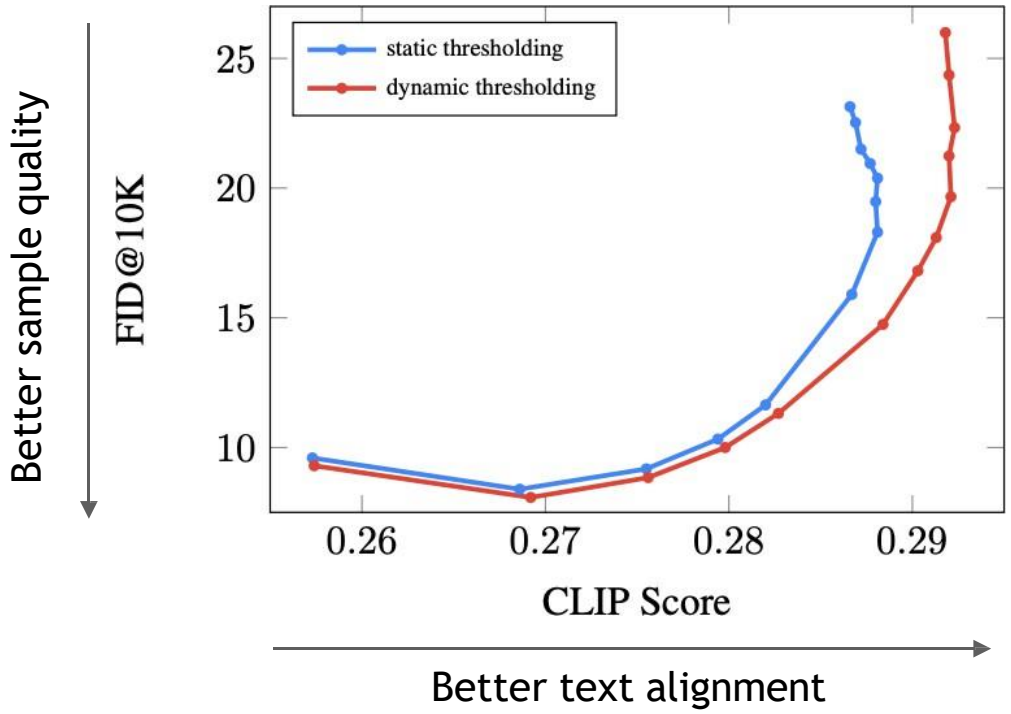
Imagen: Cascaded generation pipeline



Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

Classifier-free guidance in Imagen

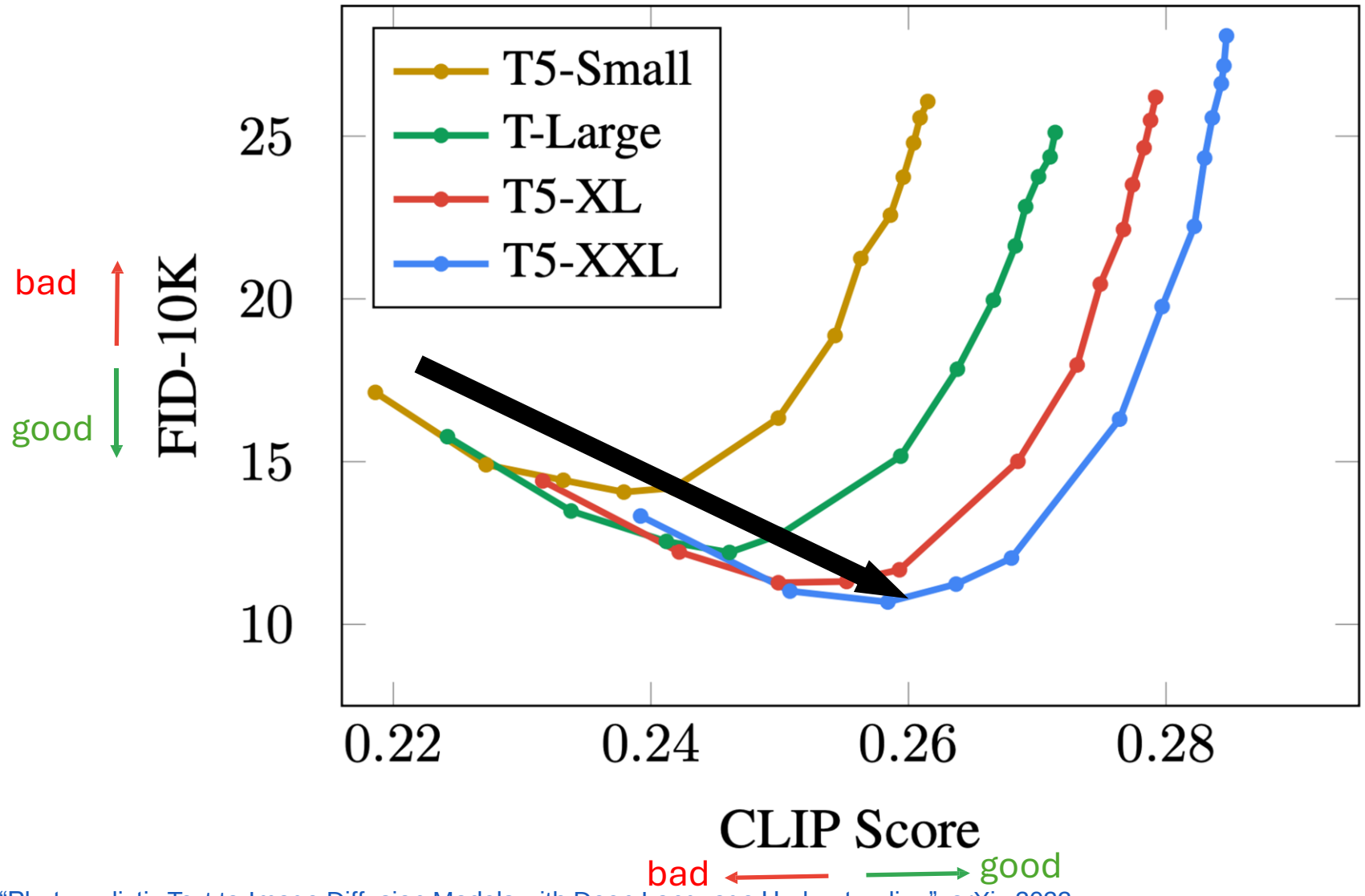
- Large classifier-free guidance weights → better text alignment, worse image fidelity



More weights on classifier-free guidance for condition

Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

Larger Text Encoders → Better Alignment, Better Fidelity



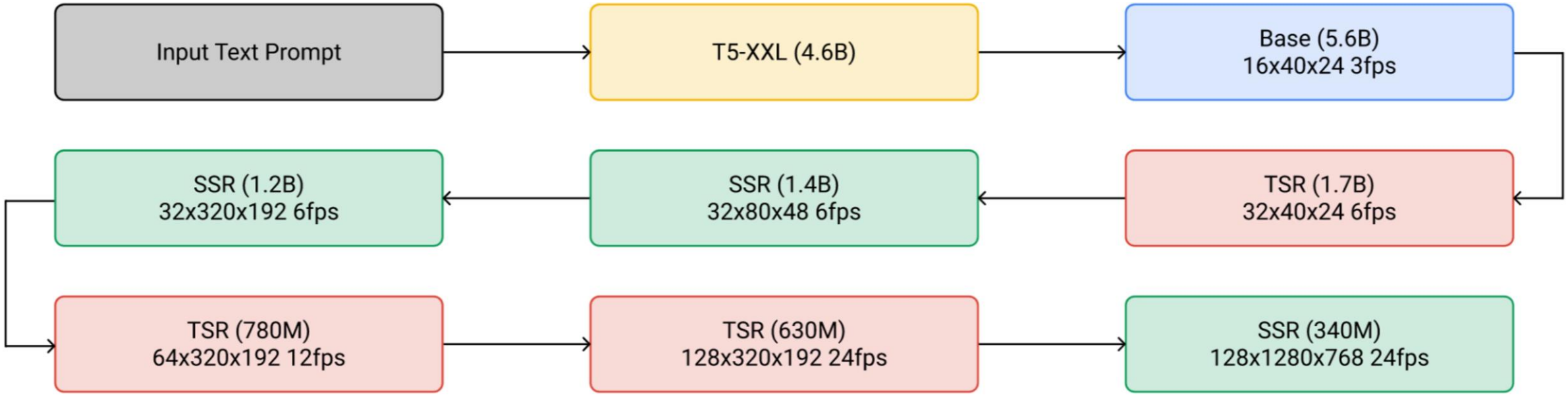
Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

Can we generalize this to video?

Imagen Video

Jonathan et al., “Imagen Video: High Definition Video Generation with Diffusion Models”, 2023.

Imagen Video generalizes **Imagen** to the video domain using a cascade of super-resolution diffusion models in space and time



TSR: Temporal Super Resolution
SSR: Spatial Super Resolution
Output shape: frames × width × height
fps: frames per second

Video diffusion models

Ho & Salimans, et al. https://video-diffusion.github.io/

- Image → Video: Just add another dimension to the data tensor
- Image architecture: 2D UNet
- Video architecture: 3D UNet, space-time separable
  - repeat the 2D UNet over frames
  - additional layers to mix over time using attention or convolution