

EECE 571F: Advanced Topics in Deep Learning

Lecture 7: Diffusion Models

Felix Fu, Qi Yan, Renjie Liao

University of British Columbia

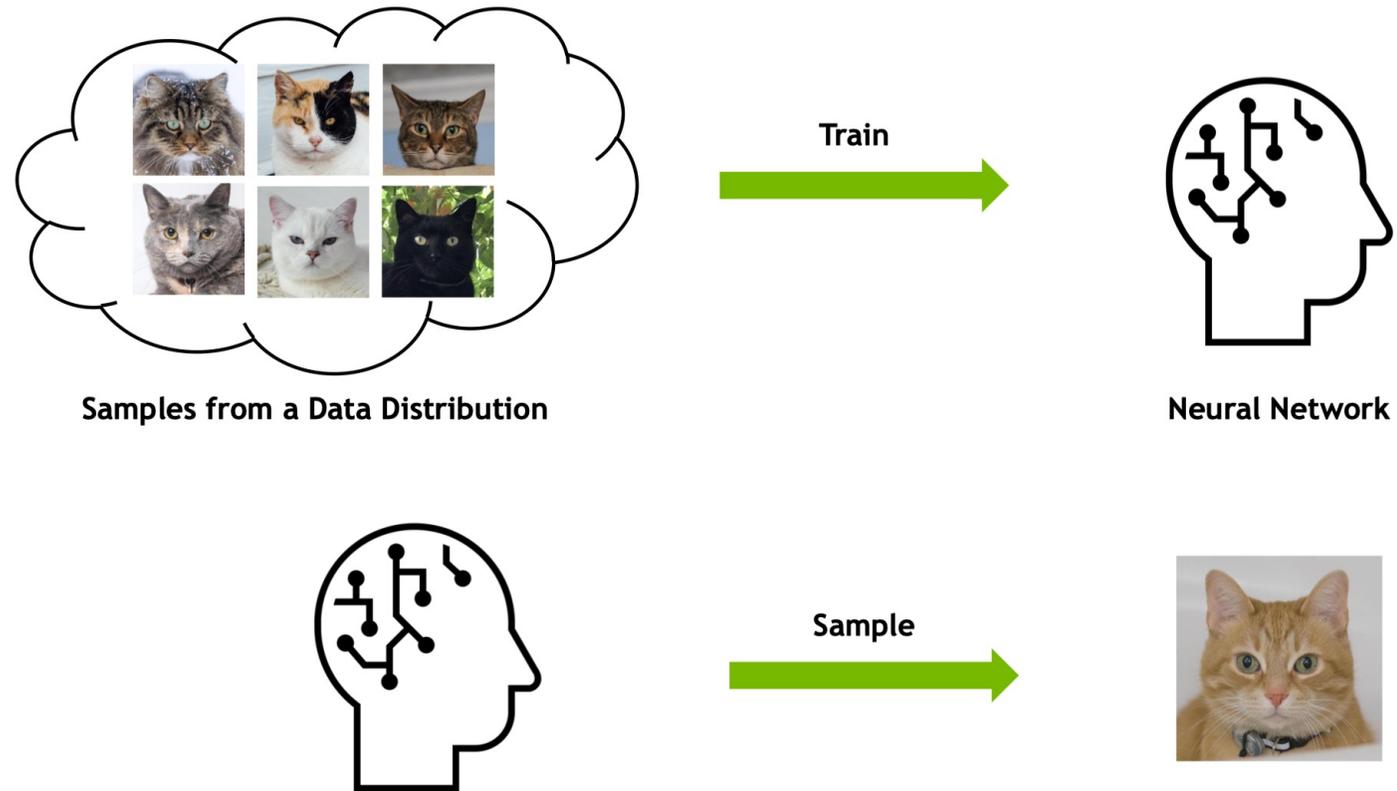
Winter, Term 2, 2025

Outline

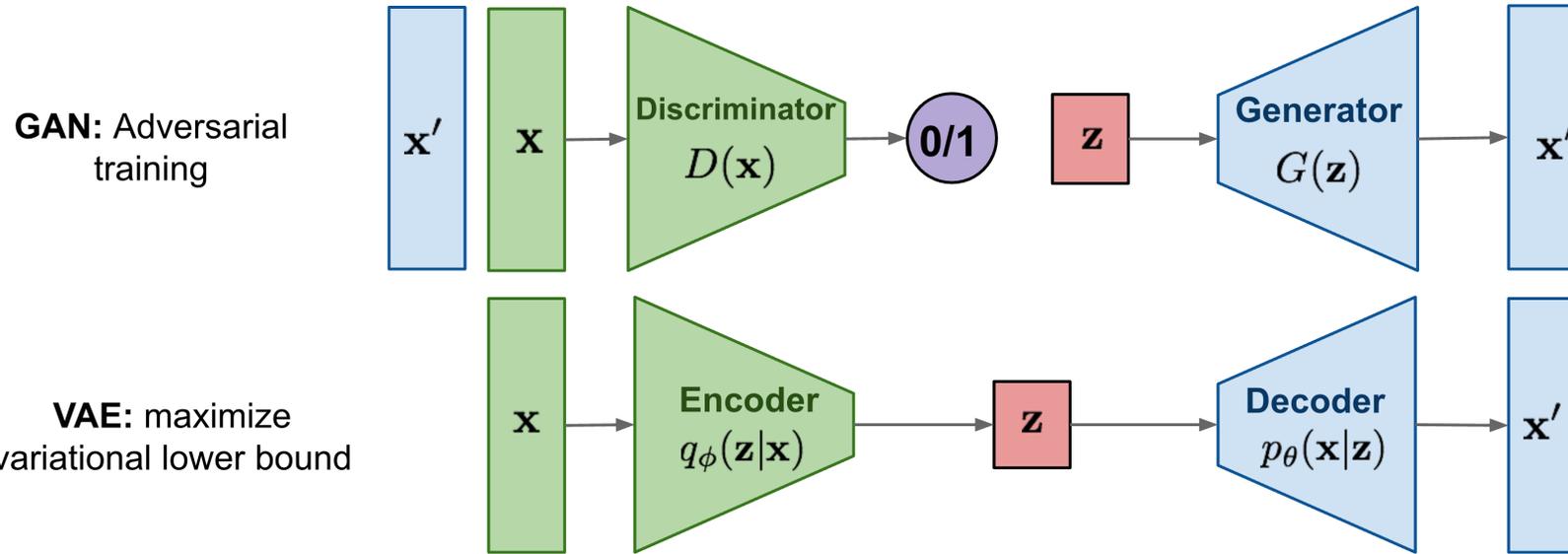
- Denoising Diffusion Probabilistic Models (DDPMs)
 - Forward Process
 - Reverse Process
 - ELBO
 - Noise vs Data Prediction
 - Inference
- Other Diffusion Models
 - Score Matching & Score-based Models
 - Score SDEs

Generative models

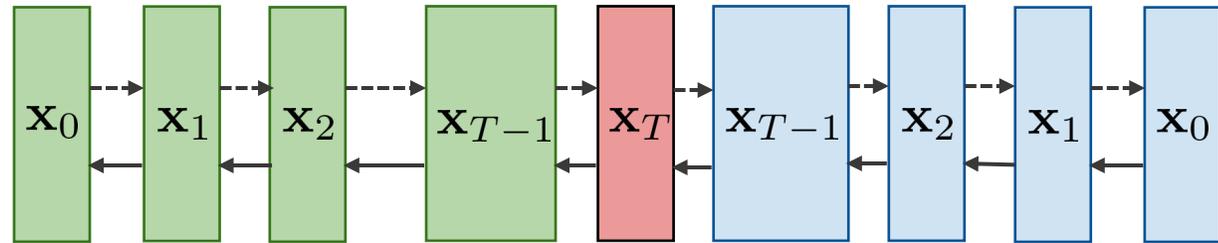
- Learning to generate data.



Generative models



Flow-based models: invertible transformations of distributions



Diffusion models: gradually add Gaussian noise and then reverse it

This lecture:

AI Artworks

📅 2021



Stable Diffusion

AI Artworks



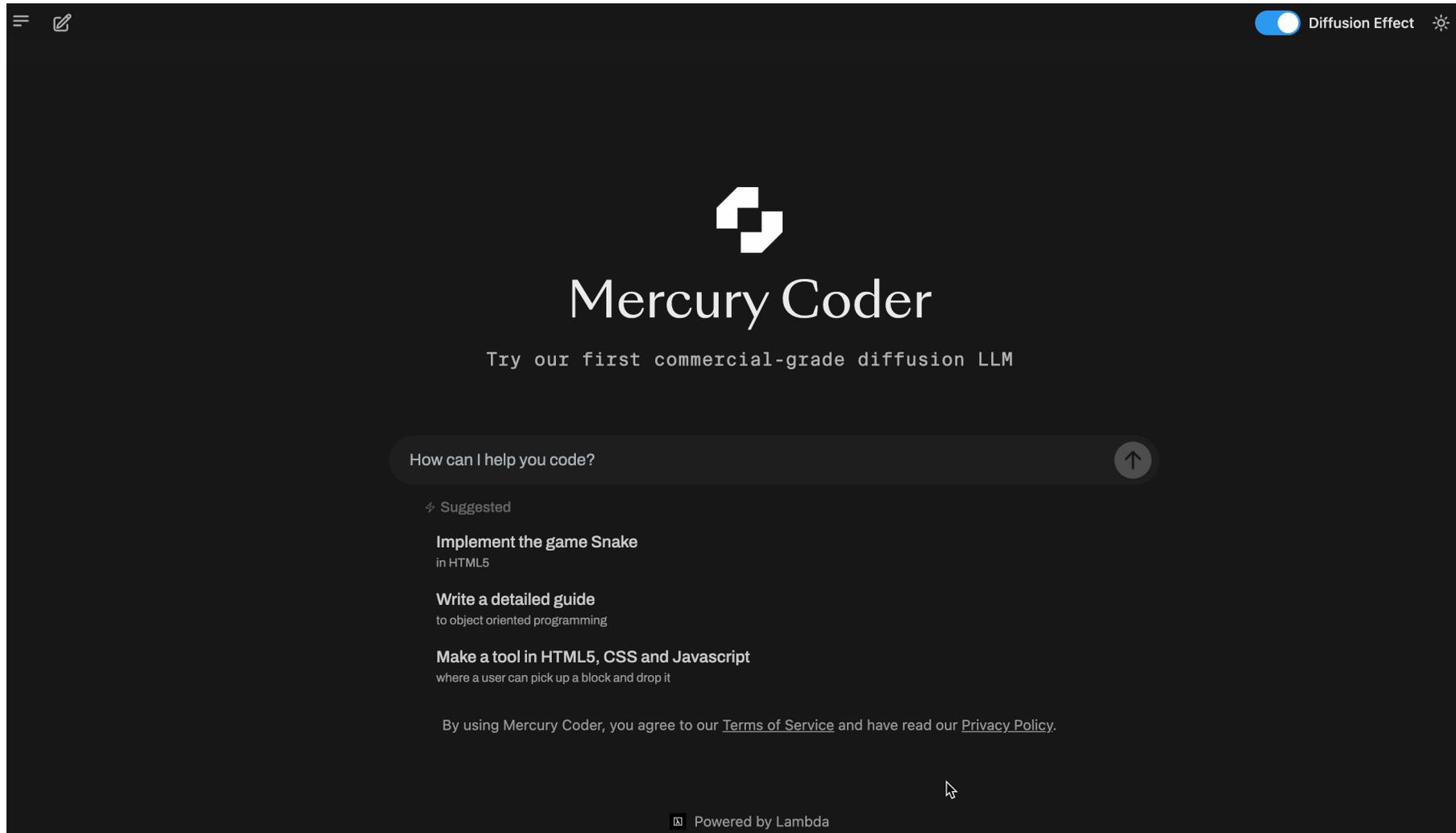
Midjourney

A fast-involving field



Sora

A fast-involving field



Mercury Coder

Motivations

Probabilistic models suffer from a trade-off:

- Some models are *tractable* but not *flexible*. (e.g. Laplace, Gaussian distributions)
- Some models are *flexible* but not *tractable*. (e.g. energy-based models)

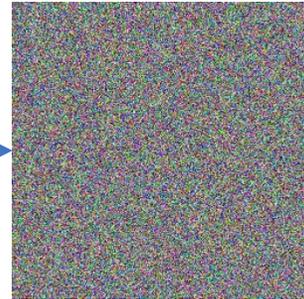
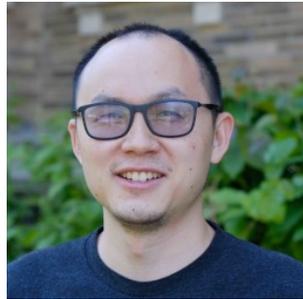
We build a generative Markov Chain (Markov Process) that

- converts a simple distribution into a target distribution.
- **has an analytical transition kernel between two intermediate states.**
- based on non-equilibrium statistical physics.

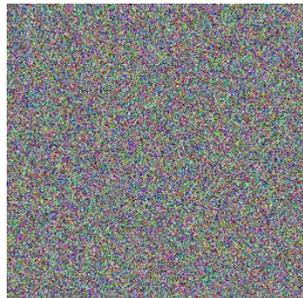
Motivations

1. A **defined** forward process that transforms data to noise (more *tractable*).
2. A **learned** reverse process that transforms noise to data (more *flexible*).

Forward



Reverse



Motivations

1. A **defined** forward process that transforms data to noise (more *tractable*).
2. A **learned** reverse process that transforms noise to data (more *flexible*).

Forward



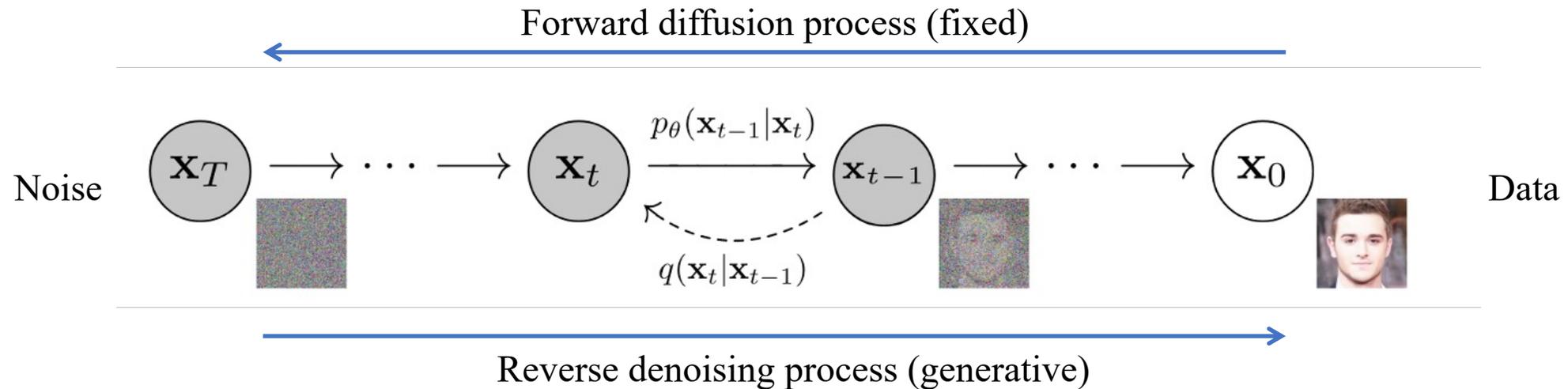
Reverse



Denoising Diffusion Probabilistic Models

Denoising diffusion models consist of two processes:

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



Outline

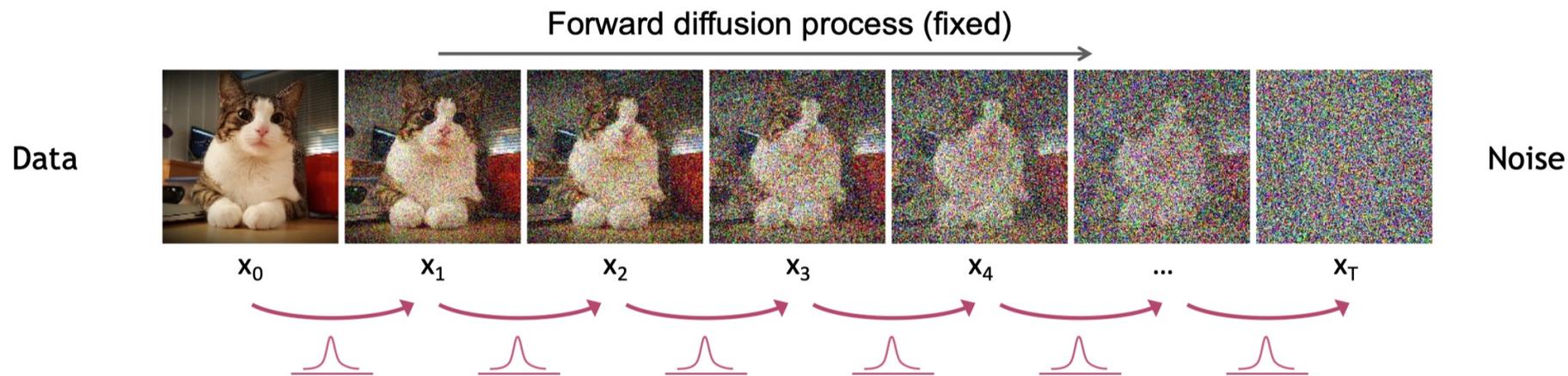
- Denoising Diffusion Probabilistic Models (DDPMs)
 - **Forward Process**
 - Reverse Process
 - ELBO
 - Noise vs Data Prediction
 - Inference
- Other Diffusion Models
 - Score Matching & Score-based Models
 - Score SDEs

Forward Process

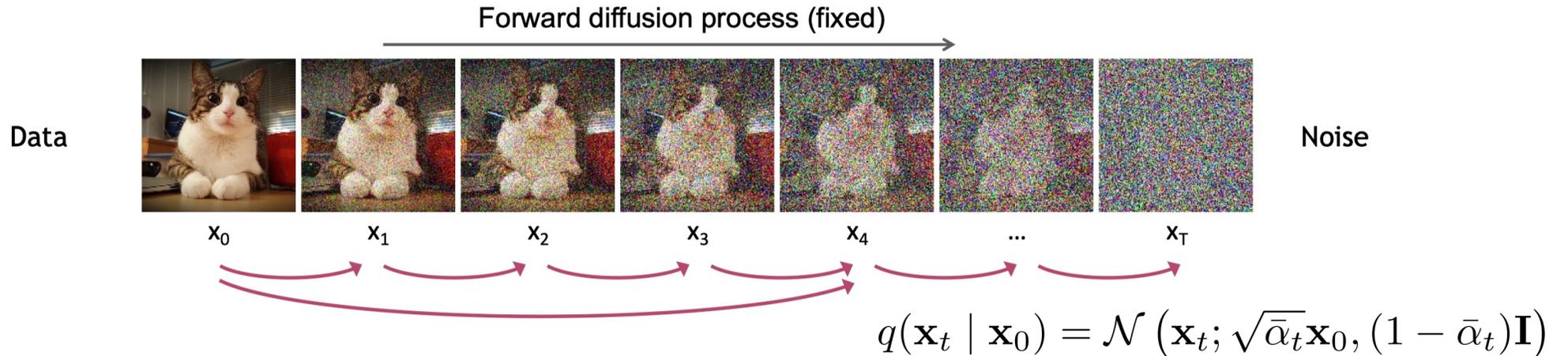
- Given real data distribution $\mathbf{x}_0 \sim q(\mathbf{x})$, we gradually adding Gaussian noise according to a schedule $\{\beta_t \in (0, 1)\}_{t=1}^T$.

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

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



Forward Process



- The forward process allows sampling of \mathbf{x}_t at arbitrary timestep t in *tractable*, closed form:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad \bar{\alpha}_t = \prod_{s=1}^t (1 - \beta_s) \quad \alpha_t \triangleq 1 - \beta_t$$

- The noise schedule is designed such that $q(\mathbf{x}_T | \mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$.

Forward Process

- Change of variable.
- We want to sample $\mathbf{x}_t \sim q(\mathbf{x}_t | \mathbf{x}_0)$ based on noise and data in one pass.

$$\begin{aligned}\mathbf{x}_t &= \sqrt{\alpha_t} \mathbf{x}_{t-1} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_{t-1} && \boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \dots \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ &= \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_t \alpha_{t-1}} \bar{\boldsymbol{\epsilon}}_{t-2} && \text{where } \bar{\boldsymbol{\epsilon}}_{t-2} \text{ merges two Gaussians.} \\ &= \dots \\ &= \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}\end{aligned}$$

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

Noise Schedule

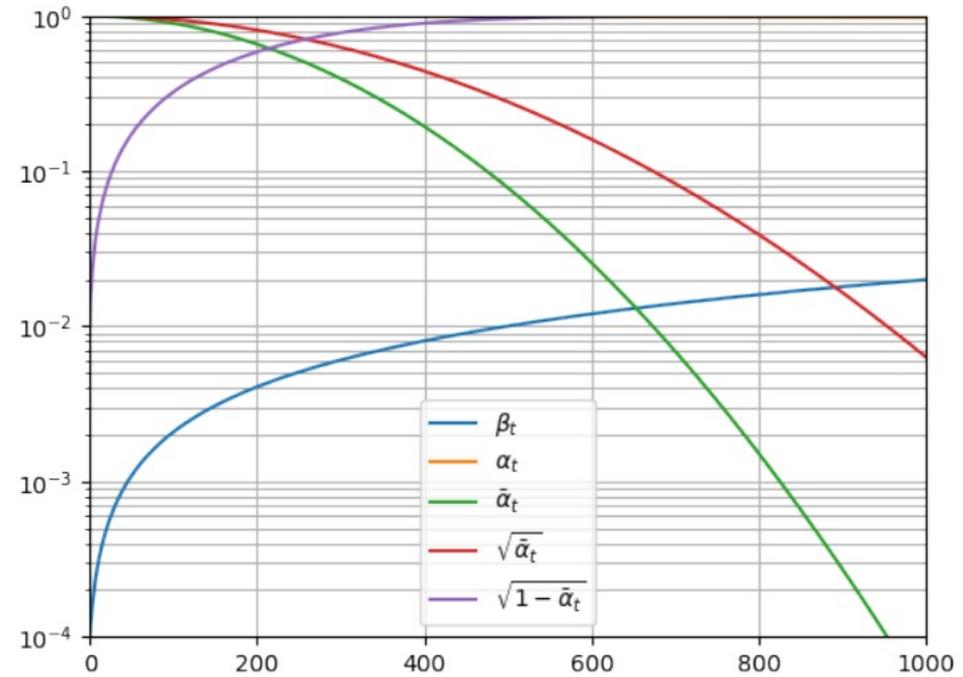
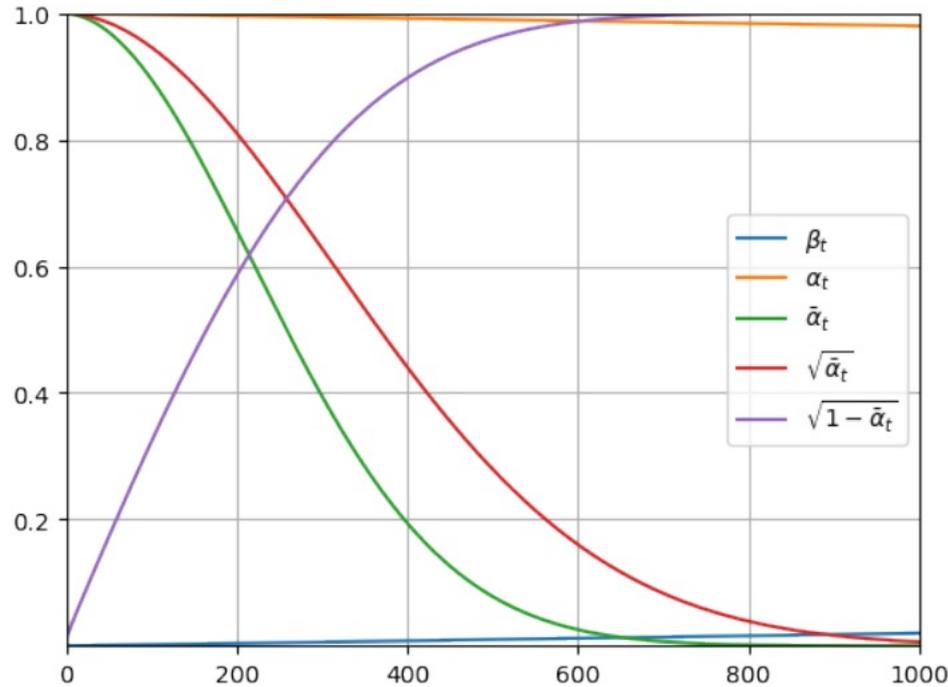


Figure 2: Parameter values for $\beta = [10^{-4}, 0.02]$ over 1000 time steps t using a linear schedule. The information in the two figures are the same, but the right-hand side uses log-scale on the y -axis to show the speed of which $\bar{\alpha}_t$ goes towards zero.

Outline

- Denoising Diffusion Probabilistic Models (DDPMs)
 - Forward Process
 - **Reverse Process**
 - ELBO
 - Noise vs Data Prediction
 - Inference
- Other Diffusion Models
 - Score Matching & Score-based Models
 - Score SDEs

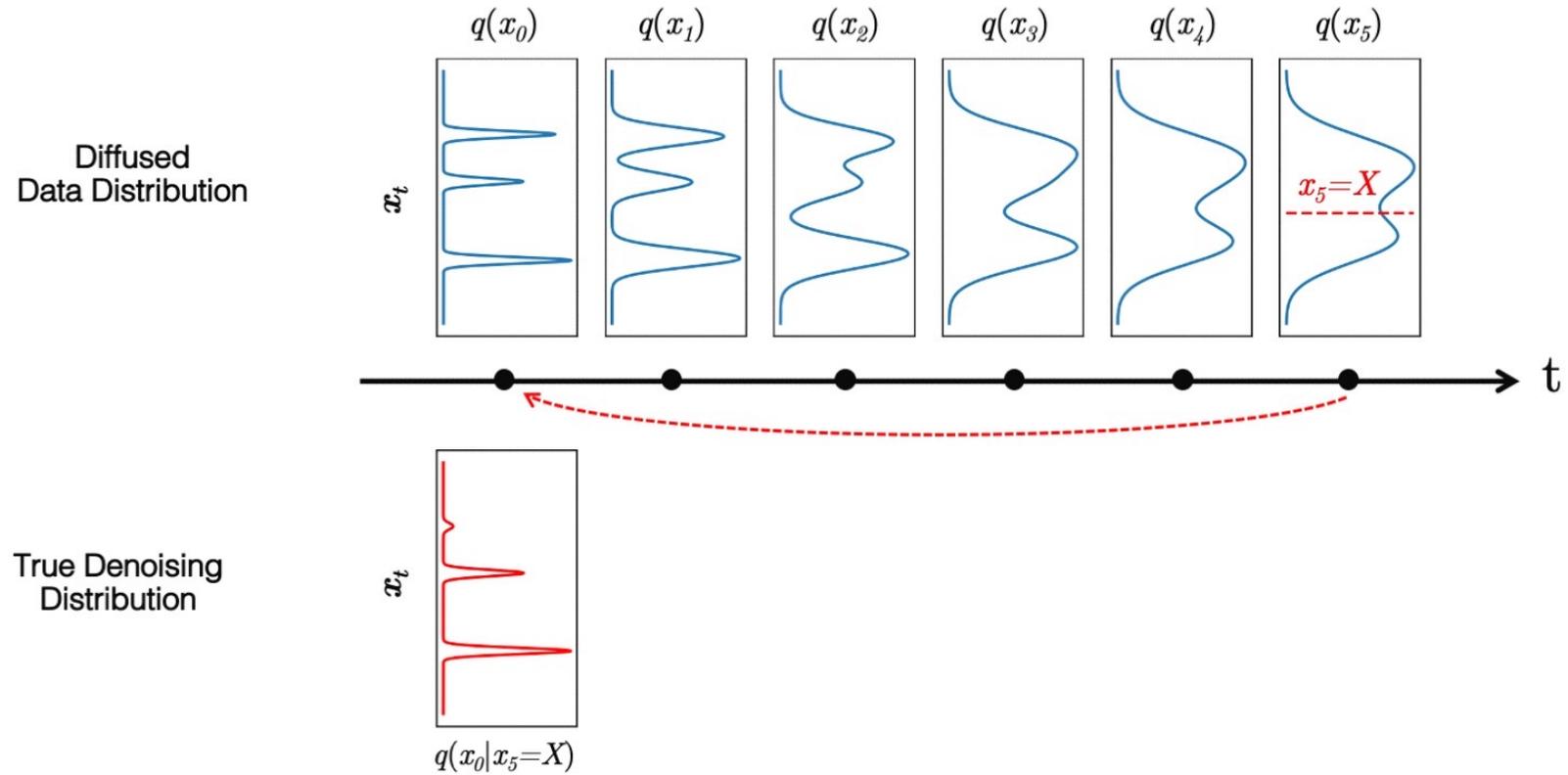
Reverse Process

- How to generate data?
 - Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$
 - Iteratively sample from the reversed Markov chain $\mathbf{x}_{t-1} \sim q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$
- But $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is **unknown and intractable!**
- Luckily, if we condition on the data, we arrive at something tractable $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$
- That is to say, we have a **closed-form** posterior distribution as the target. Yay!

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

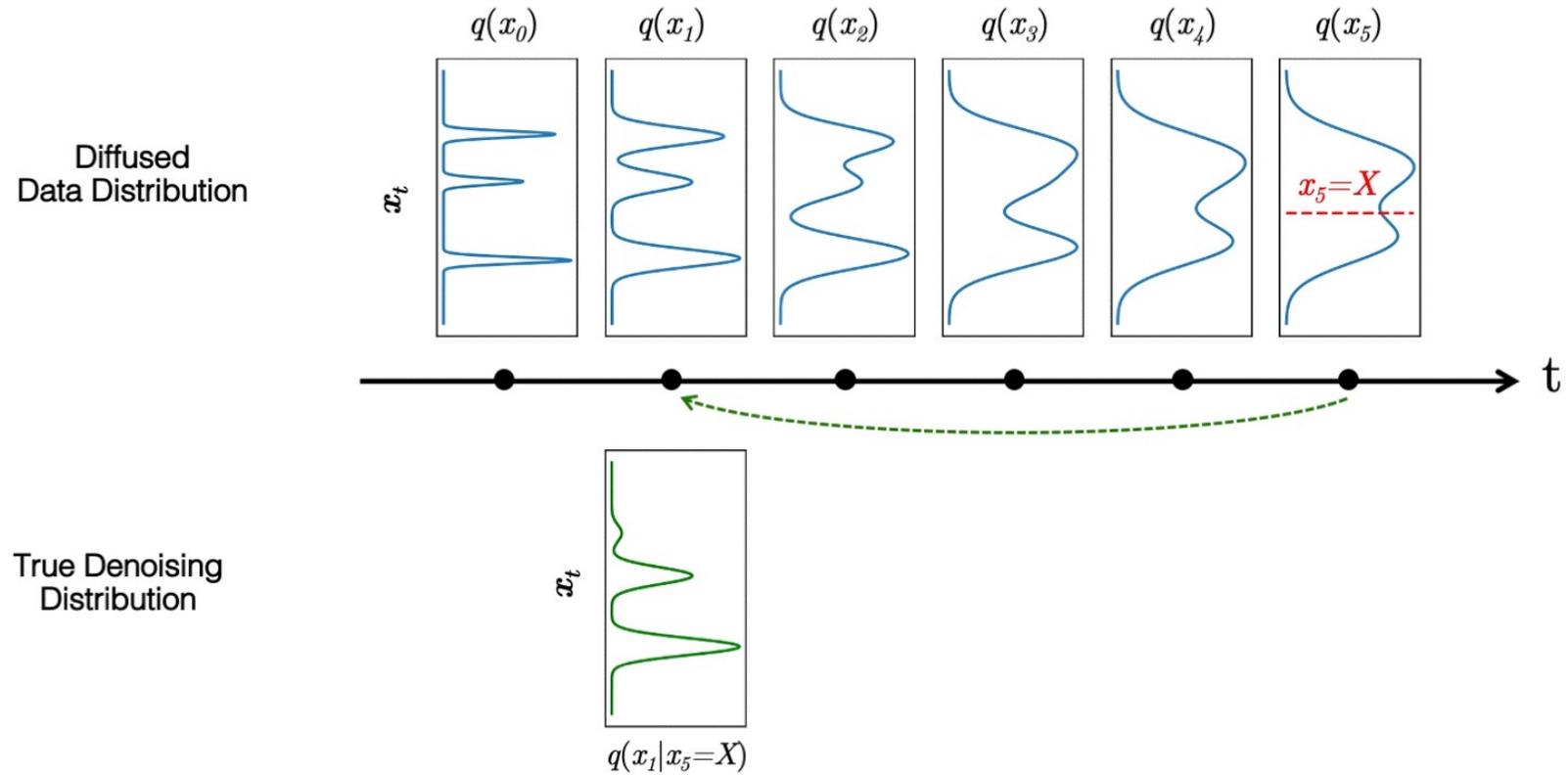
$$\text{where } \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0 + \frac{\sqrt{1 - \beta_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t \quad \text{and} \quad \tilde{\boldsymbol{\beta}}_t := \frac{1 - \bar{\alpha}_{t-1}}{1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

Reverse Process

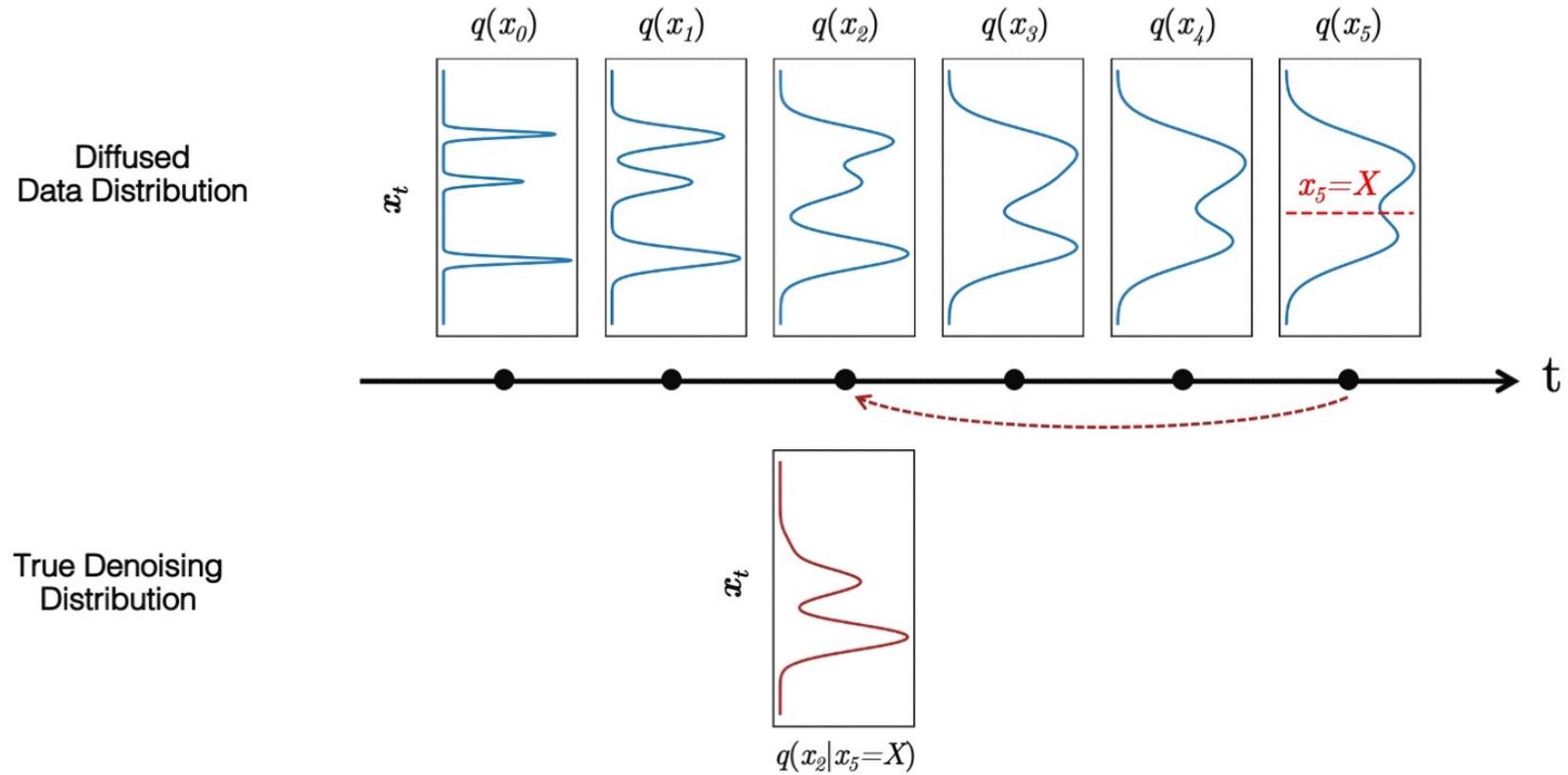


Complex distribution!

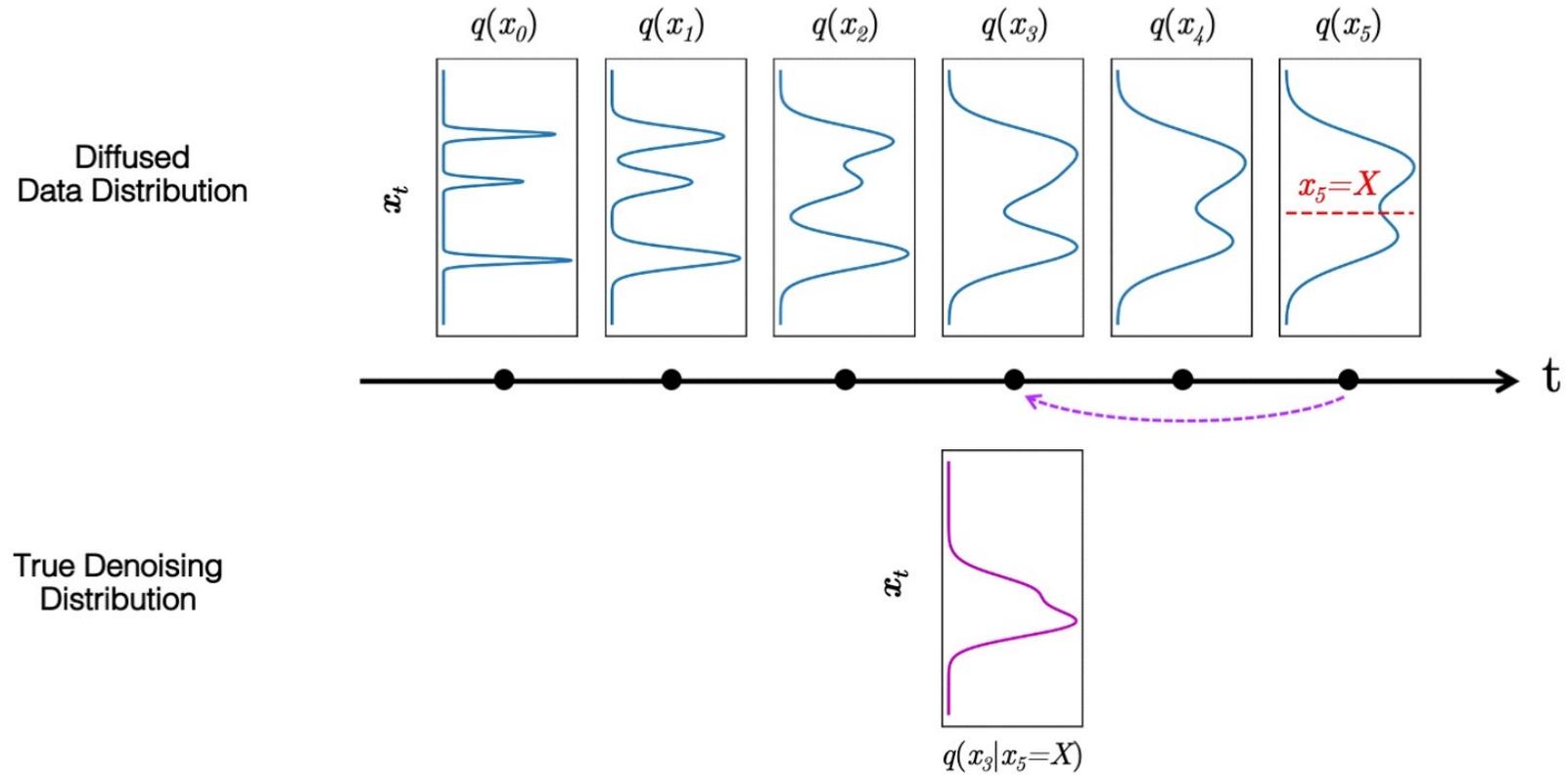
Reverse Process



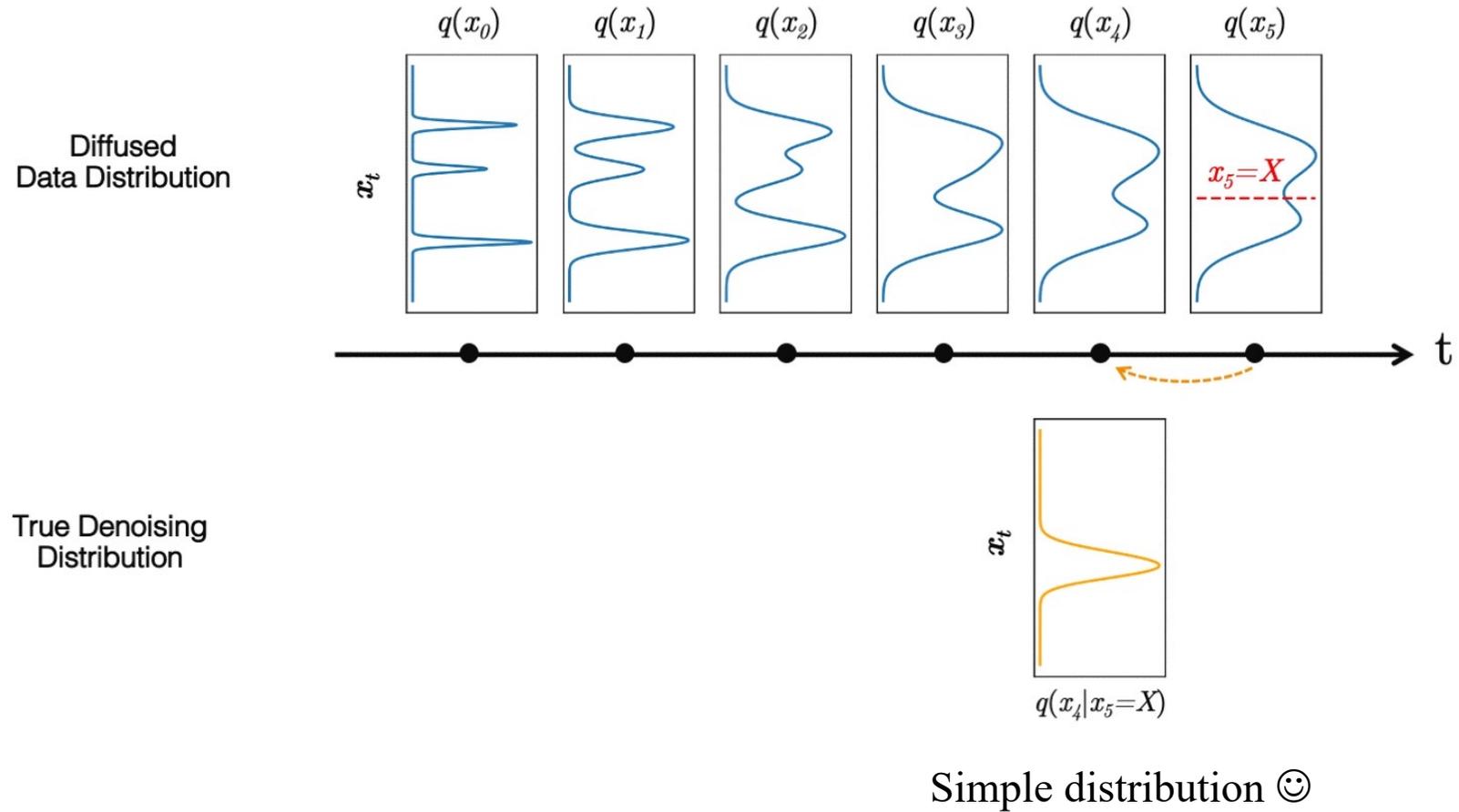
Reverse Process



Reverse Process



Reverse Process



Reverse Process

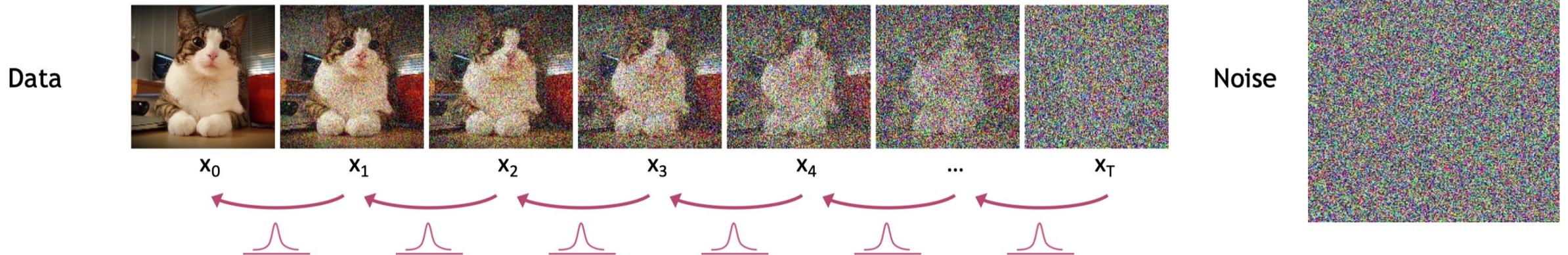
- We predict the mean and covariance of added Gaussian noise.

$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

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

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

Reverse denoising process (generative)



Outline

- Denoising Diffusion Probabilistic Models (DDPMs)
 - Forward Process
 - Reverse Process
 - **ELBO**
 - Noise vs Data Prediction
 - Inference
- Other Diffusion Models
 - Score Matching & Score-based Models
 - Score SDEs

ELBO

- Connection with Variational Autoencoders

$$\begin{aligned}\log p(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \right] + \mathcal{D}_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) \| p_\theta(\mathbf{z} | \mathbf{x})) \\ &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x} | \mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x} | \mathbf{z})] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \right] \\ &= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x} | \mathbf{z})]}_{\text{Reconstruction}} - \underbrace{\mathcal{D}_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) \| p(\mathbf{z}))}_{\text{Prior Matching}}\end{aligned}$$

ELBO

- Connection with Variational Autoencoders

$$\begin{aligned}\log p(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \right] + \mathcal{D}_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) \| p_\theta(\mathbf{z} | \mathbf{x})) \\ &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x} | \mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x} | \mathbf{z})] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \right] \\ &= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x} | \mathbf{z})]}_{\text{Reconstruction}} - \underbrace{\mathcal{D}_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) \| p(\mathbf{z}))}_{\text{Prior Matching}}\end{aligned}$$

$$\begin{aligned}\log p(\mathbf{x}_0) &= \log \int p(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \\ &= \log \int \frac{p(\mathbf{x}_{0:T}) q(\mathbf{x}_{1:T} | \mathbf{x}_0)}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} d\mathbf{x}_{1:T} \\ &= \log \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right] \\ &\geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right] \\ &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{\prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1})} \right]\end{aligned}$$

ELBO

- Connection with Variational Autoencoders

$$\begin{aligned}
 \log p(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \right] + \mathcal{D}_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) \| p_\theta(\mathbf{z} | \mathbf{x})) \\
 &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x} | \mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \right] \\
 &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x} | \mathbf{z})] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \right] \\
 &= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x} | \mathbf{z})]}_{\text{Reconstruction}} - \underbrace{\mathcal{D}_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) \| p(\mathbf{z}))}_{\text{Prior Matching}}
 \end{aligned}$$

VAE

$$\begin{aligned}
 \log p(\mathbf{x}_0) &= \log \int p(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \\
 &= \log \int \frac{p(\mathbf{x}_{0:T}) q(\mathbf{x}_{1:T} | \mathbf{x}_0)}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} d\mathbf{x}_{1:T} \\
 &= \log \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right] \\
 &\geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right] \\
 &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{\prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1})} \right]
 \end{aligned}$$

Diffusion

ELBO

- Connection with Variational Autoencoders

$$\begin{aligned}
 \log p(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \right] + \mathcal{D}_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) \| p_\theta(\mathbf{z} | \mathbf{x})) \\
 &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x} | \mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \right] \\
 &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x} | \mathbf{z})] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \right] \\
 &= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x} | \mathbf{z})]}_{\text{Reconstruction}} - \underbrace{\mathcal{D}_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) \| p(\mathbf{z}))}_{\text{Prior Matching}}
 \end{aligned}$$

VAE

$$\begin{aligned}
 \log p(\mathbf{x}_0) &= \log \int p(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \\
 &= \log \int \frac{p(\mathbf{x}_{0:T}) q(\mathbf{x}_{1:T} | \mathbf{x}_0)}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} d\mathbf{x}_{1:T} \\
 &= \log \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right] \\
 &\geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right] \\
 &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{\prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1})} \right]
 \end{aligned}$$

Diffusion

ELBO

$$\begin{aligned}\log p(\mathbf{x}_0) &\geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{\prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1})} \right] \\ &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log p(\mathbf{x}_T) + \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] && \text{Logarithmic rules} \\ &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log p(\mathbf{x}_T) + \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_{t-1})}{q(\mathbf{x}_t)} \right] && \text{Bayes' rule} \\ &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T)} + \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log q(\mathbf{x}_0) \right] && \text{Telescoping products}\end{aligned}$$

Intractable!

ELBO

$$\begin{aligned}
 \mathbb{E}_{q(\mathbf{x}_0)} \left[\log p_\theta(\mathbf{x}_0) \right] &\geq \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[\log p_\theta(\mathbf{x}_T) - \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)} \right] && \text{Rao-blackwellization} \\
 &= \mathbb{E}_q \left[\log p_\theta(\mathbf{x}_T) - \sum_{t=2}^T \log \left(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} \right) - \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)} \right] && \text{Bayes' rule} \\
 &= \mathbb{E}_q \left[\log p_\theta(\mathbf{x}_T) - \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} - \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)} \right] && \text{Logarithmic rules} \\
 &= \mathbb{E}_q \left[\log p_\theta(\mathbf{x}_T) - \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} - \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)} \right] && \text{Telescoping products} \\
 &= \mathbb{E}_q \left[\log \frac{p_\theta(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} - \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \right] \\
 \log p_\theta(\mathbf{x}_0) &\geq \underbrace{-\mathcal{D}_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}_{\text{Prior Matching } (L_T)} - \sum_{t=2}^T \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \underbrace{[\mathcal{D}_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))]}_{\text{Prior Matching } (L_{t-1})} - \underbrace{\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)]}_{\text{Reconstruction } (L_0)}
 \end{aligned}$$

Parameterizing DDPM

- KL divergence has a simple form between Gaussians.

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

Parameterizing DDPM

- KL divergence has a simple form between Gaussians.

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

- Recall the re-parameterization of the forward process $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Parameterizing DDPM

- KL divergence has a simple form between Gaussians.

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

- Recall the re-parameterization of the forward process $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

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

Parameterizing DDPM

- KL divergence has a simple form between Gaussians.

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

- Recall the re-parameterization of the forward process $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

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

- Trainable network predicts the noise mean.

Parameterizing DDPM

- KL divergence has a simple form between Gaussians.

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

- Recall the re-parameterization of the forward process $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

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

- Trainable network predicts the noise mean.

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

Parameterizing DDPM

- KL divergence has a simple form between Gaussians.

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

- Recall the re-parameterization of the forward process $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

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

- Trainable network predicts the noise mean.

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

- Final objective:

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\frac{\beta_t^2}{2\sigma_t^2(1 - \beta_t)(1 - \bar{\alpha}_t)} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\underbrace{\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}}_{\mathbf{x}_t}, t \right) \right\|^2 \right] + C$$

Simplified Training Objective

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\underbrace{\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}}_{\lambda_t} \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

Simplified Training Objective

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\frac{\beta_t^2}{\underbrace{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}_{\lambda_t}} \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

- λ_t adjusts the weights for correct maximum likelihood estimation.

Simplified Training Objective

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\underbrace{\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}}_{\lambda_t} \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

- λ_t adjusts the weights for correct maximum likelihood estimation.
- In DDPM, the training objective gets simplified to:

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

Training and Inference

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

Generated Samples

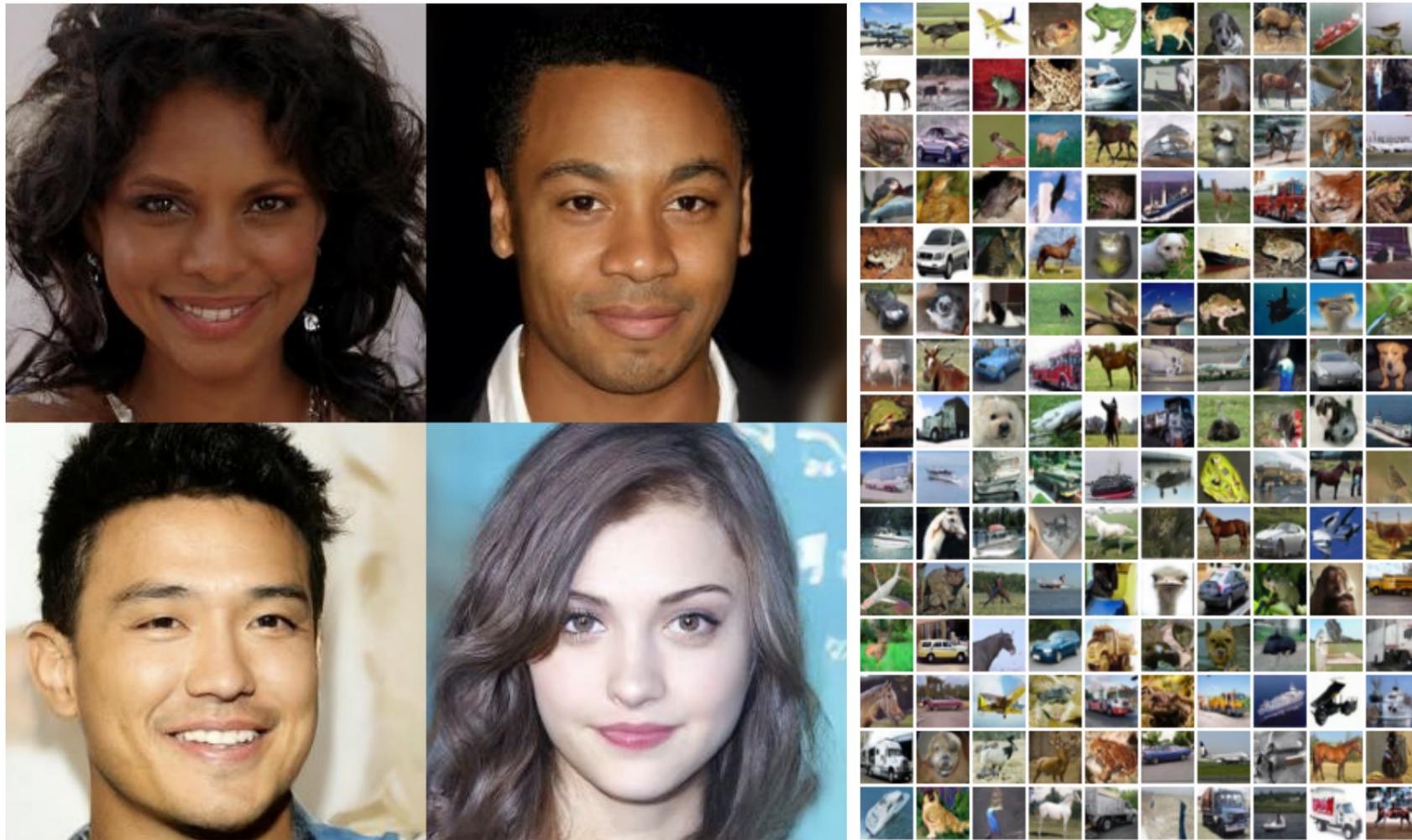


Figure 1: Generated samples on CelebA-HQ 256×256 (left) and unconditional CIFAR10 (right)

Outline

- Denoising Diffusion Probabilistic Models (DDPMs)
 - Forward Process
 - Reverse Process
 - ELBO
 - **Noise vs Data Prediction**
 - Inference
- Other Diffusion Models
 - Score Matching & Score-based Models
 - Score SDEs

Noise vs Data Prediction

- DDPM noise estimation loss:

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

Noise vs Data Prediction

- DDPM noise estimation loss:

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

- Recall the forward process:

$$\begin{aligned} \mathbf{x}_t &= \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, & \epsilon &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ \epsilon &= \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0}{\sqrt{1 - \bar{\alpha}_t}} \end{aligned}$$

Noise vs Data Prediction

- DDPM noise estimation loss:

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

- Recall the forward process:

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

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

- Let's use \mathbf{x}_0 centered parameterization:

$$\mathbf{D}_{\theta}(\mathbf{x}_t, t) \approx \mathbf{x}_0$$

$$\epsilon_{\theta} = \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{D}_{\theta}(\mathbf{x}_t, t)}{\sqrt{1 - \bar{\alpha}_t}}$$

Noise vs Data Prediction

- DDPM data estimation loss:

$$\begin{aligned} L_{\text{simple}} &= \mathbb{E}_{\mathbf{x}_0, \epsilon, t} \left[\|\epsilon - \epsilon_{\theta}(\mathbf{x}_t, t)\|^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon, t} \left[\left\| \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0}{\sqrt{1 - \bar{\alpha}_t}} - \epsilon_{\theta}(\mathbf{x}_t, t) \right\|^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon, t} \left[\left\| \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0}{\sqrt{1 - \bar{\alpha}_t}} - \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{D}_{\theta}(\mathbf{x}_t, t)}{\sqrt{1 - \bar{\alpha}_t}} \right\|^2 \right] \\ L'_{\text{simple}} &:= \mathbb{E}_{\mathbf{x}_0, \epsilon, t} \left[\|\mathbf{x}_0 - \mathbf{D}_{\theta}(\mathbf{x}_t, t)\|^2 \right] \end{aligned}$$

- Here, we show a simplified objective. See more discussion in [5].

Outline

- Denoising Diffusion Probabilistic Models (DDPMs)
 - Forward Process
 - Reverse Process
 - ELBO
 - Noise vs Data Prediction
 - **Inference**
 - **Guided conditional generation**
 - Classifier-free guidance
 - DDIM
- Other Diffusion Models
 - Score Matching & Score-based Models
 - Score SDEs

Inference: guided conditional generation

- Goal: generation with *conditions* (controllability)



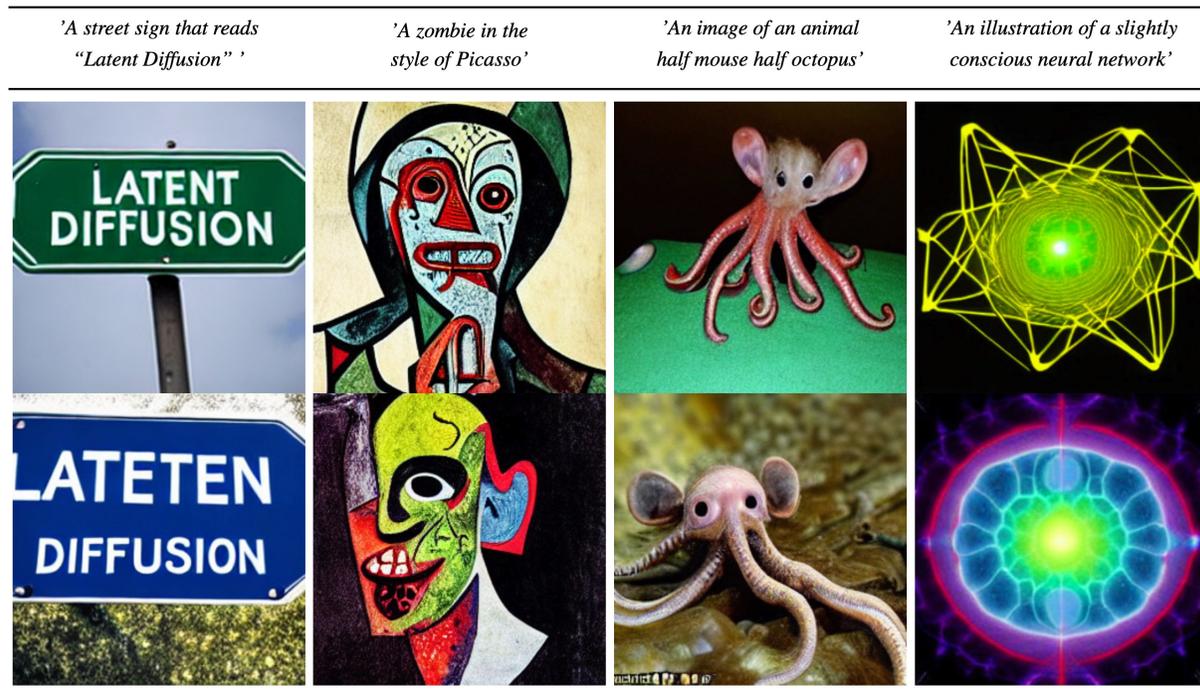
Unconditional generation



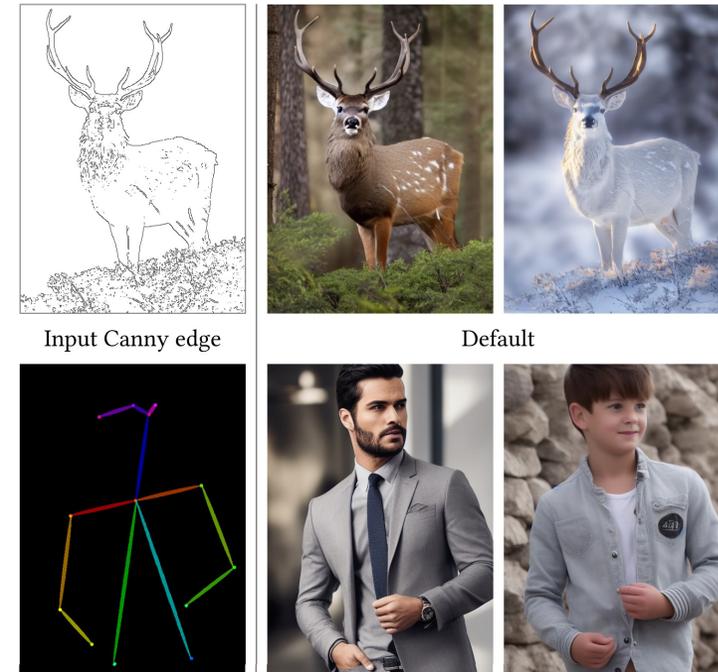
Class-conditional generation

Inference: guided conditional generation

- Goal: generation with *conditions* (controllability)



Text-to-Image generation



Visual cue-based generation

Inference: guided conditional generation

- Naïve approach: explicit training using the data-condition pairs (\mathbf{x}, \mathbf{y})
- Generative modeling objective:

$$q(\mathbf{x} \mid \mathbf{y})$$

through denoiser network:

$$\epsilon_{\theta}(\mathbf{x}_t, t, \mathbf{y})$$

Inference: guided conditional generation

- Naïve approach: explicit training using the data-condition pairs (\mathbf{x}, \mathbf{y})
- Generative modeling objective:

$$q(\mathbf{x} \mid \mathbf{y})$$

through denoiser network:

$$\epsilon_{\theta}(\mathbf{x}_t, t, \mathbf{y})$$

- Caveats:
 - Data scarcity: what if one condition appears rarely in the dataset?
 - Flexibility: control “strength” of conditioning?

Inference: classifier guidance

- Can we steer the generation controllably using another neural net?
- Bayes' rule:

$$q(\mathbf{x}_t|y) = \frac{q(\mathbf{x}_t)q(y|\mathbf{x}_t)}{q(y)}$$

$$\log q(\mathbf{x}_t|y) = \log q(\mathbf{x}_t) + \log q(y|\mathbf{x}_t) - \log q(y)$$

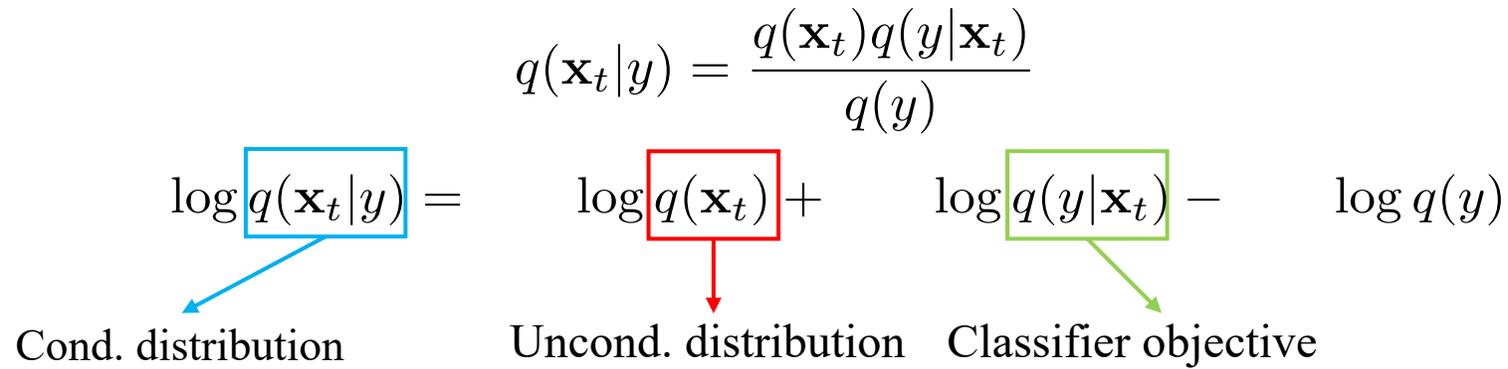
Inference: classifier guidance

- Can we steer the generation controllably using another neural net?
- Bayes' rule:

$$q(\mathbf{x}_t|y) = \frac{q(\mathbf{x}_t)q(y|\mathbf{x}_t)}{q(y)}$$

$\log q(\mathbf{x}_t|y) = \log q(\mathbf{x}_t) + \log q(y|\mathbf{x}_t) - \log q(y)$

Cond. distribution Uncond. distribution Classifier objective



Inference: classifier guidance

- Can we steer the generation controllably using another neural net?
- Bayes' rule:

$$q(\mathbf{x}_t|y) = \frac{q(\mathbf{x}_t)q(y|\mathbf{x}_t)}{q(y)}$$
$$\log q(\mathbf{x}_t|y) = \log q(\mathbf{x}_t) + \log q(y|\mathbf{x}_t) - \log q(y)$$

Cond. distribution Uncond. distribution Classifier objective

- Take gradient w.r.t. data:

$$\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|y) = \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log q(y|\mathbf{x}_t) - \cancel{\nabla_{\mathbf{x}_t} \log q(y)}$$
$$\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|y) = \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log q(y|\mathbf{x}_t)$$

Inference: classifier guidance

- Some background on *score function*:

$$\mathbf{s}(\mathbf{x}) := \nabla_{\mathbf{x}} \log q(\mathbf{x})$$

- Think about *gradient ascent*
- Gaussian distribution score function:

If $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 \mathbf{I})$, then $\nabla_{\mathbf{x}} \log q(\mathbf{x}) = \nabla_{\mathbf{x}} \left(-\frac{1}{2\sigma^2} (\mathbf{x} - \boldsymbol{\mu})^2 \right) = -\frac{\mathbf{x} - \boldsymbol{\mu}}{\sigma^2} = -\frac{\boldsymbol{\epsilon}}{\sigma}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Inference: classifier guidance

- Some background on *score function*:

$$\mathbf{s}(\mathbf{x}) := \nabla_{\mathbf{x}} \log q(\mathbf{x})$$

- Think about *gradient ascent*
- Gaussian distribution score function:

If $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 \mathbf{I})$, then $\nabla_{\mathbf{x}} \log q(\mathbf{x}) = \nabla_{\mathbf{x}} \left(-\frac{1}{2\sigma^2} (\mathbf{x} - \boldsymbol{\mu})^2 \right) = -\frac{\mathbf{x} - \boldsymbol{\mu}}{\sigma^2} = -\frac{\boldsymbol{\epsilon}}{\sigma}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

- **Denoising score matching (DSM) [10]:**

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

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

Inference: classifier guidance

- Can we steer the generation controllably using another neural net?
- Putting things together:

$$\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|y) = \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log q(y|\mathbf{x}_t)$$

- First term:

$$\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) \approx -\frac{\epsilon_{\theta}(\mathbf{x}_t, t)}{\sqrt{1 - \bar{\alpha}_t}}$$

- Second term:

$$\nabla_{\mathbf{x}_t} \log q(y|\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log f_{\phi}(y|\mathbf{x}_t)$$

- *Gradient* of a classifier

Inference: classifier guidance

- Can we steer the generation controllably using another neural net?
- Classifier guidance:

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|y) &= \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log q(y|\mathbf{x}_t) \\ &\approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t) \\ &= -\frac{1}{\sqrt{1-\bar{\alpha}_t}} (\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t} \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t))\end{aligned}$$

- Modified denoising process:

$$\bar{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t) = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t} w \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)$$

Inference: classifier guidance

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

Input: class label y , gradient scale s

$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$

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

$\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$

$x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log f_\phi(y|x_t), \Sigma)$

end for

return x_0

Outline

- Denoising Diffusion Probabilistic Models (DDPMs)
 - Forward Process
 - Reverse Process
 - ELBO
 - Noise vs Data Prediction
 - **Inference**
 - Guided conditional generation
 - **Classifier-free guidance**
 - DDIM
- Other Diffusion Models
 - Score Matching & Score-based Models
 - Score SDEs

Inference: classifier-free guidance

- Drawbacks of classifier guidance

$$\bar{\epsilon}_\theta(\mathbf{x}_t, t) = \epsilon_\theta(\mathbf{x}_t, t) - \sqrt{1 - \bar{\alpha}_t} w \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)$$

- Classifier must be separately trained, but it is usually not trained on the noisy data.
- Computing gradient in the denoising process is slow.

Inference: classifier-free guidance

- Drawbacks of classifier guidance

$$\bar{\epsilon}_\theta(\mathbf{x}_t, t) = \epsilon_\theta(\mathbf{x}_t, t) - \sqrt{1 - \bar{\alpha}_t} w \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)$$

- Classifier must be separately trained, but it is usually not trained on the noisy data.
 - Computing gradient in the denoising process is slow.
- **Classifier-free guidance (CFG)** uses a single neural net for both purposes.

$$\text{Unconditional: } \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) = -\frac{\epsilon_\theta(\mathbf{x}_t, t, y = \emptyset)}{\sqrt{1 - \bar{\alpha}_t}}$$

$$\text{Conditional: } \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|y) = -\frac{\epsilon_\theta(\mathbf{x}_t, t, y)}{\sqrt{1 - \bar{\alpha}_t}}$$

Inference: classifier-free guidance

- Drawbacks of classifier guidance

$$\bar{\epsilon}_\theta(\mathbf{x}_t, t) = \epsilon_\theta(\mathbf{x}_t, t) - \sqrt{1 - \bar{\alpha}_t} w \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)$$

- Classifier must be separately trained, but it is usually not trained on the noisy data.
 - Computing gradient in the denoising process is slow.
- **Classifier-free guidance (CFG)** uses a single neural net for both purposes.

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) &= \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|y) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \\ &= -\frac{1}{\sqrt{1 - \bar{\alpha}_t}} \left(\epsilon_\theta(\mathbf{x}_t, t, y) - \epsilon_\theta(\mathbf{x}_t, t, y = \emptyset) \right) \\ \bar{\epsilon}_\theta(\mathbf{x}_t, t, y) &= \epsilon_\theta(\mathbf{x}_t, t, y) - \sqrt{1 - \bar{\alpha}_t} w \nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) \\ &= \epsilon_\theta(\mathbf{x}_t, t, y) + w \left(\epsilon_\theta(\mathbf{x}_t, t, y) - \epsilon_\theta(\mathbf{x}_t, t) \right) \\ &= (w + 1) \epsilon_\theta(\mathbf{x}_t, t, y) - w \epsilon_\theta(\mathbf{x}_t, t)\end{aligned}$$

Inference: classifier-free guidance

- Drawbacks of classifier guidance

$$\bar{\epsilon}_\theta(\mathbf{x}_t, t) = \epsilon_\theta(\mathbf{x}_t, t) - \sqrt{1 - \bar{\alpha}_t} w \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)$$

- Classifier must be separately trained, but it is usually not trained on the noisy data.
- Computing gradient in the denoising process is slow.
- **Classifier-free guidance (CFG)** uses a single neural net for both purposes.

$$\begin{aligned} \nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) &= \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|y) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \\ &= -\frac{1}{\sqrt{1 - \bar{\alpha}_t}} \left(\epsilon_\theta(\mathbf{x}_t, t, y) - \epsilon_\theta(\mathbf{x}_t, t, y = \emptyset) \right) \end{aligned}$$

$$\begin{aligned} \bar{\epsilon}_\theta(\mathbf{x}_t, t, y) &= \epsilon_\theta(\mathbf{x}_t, t, y) - \sqrt{1 - \bar{\alpha}_t} w \nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) \\ &= \epsilon_\theta(\mathbf{x}_t, t, y) + w \left(\epsilon_\theta(\mathbf{x}_t, t, y) - \epsilon_\theta(\mathbf{x}_t, t) \right) \\ &= (w + 1) \epsilon_\theta(\mathbf{x}_t, t, y) - w \epsilon_\theta(\mathbf{x}_t, t) \end{aligned}$$

Guidance scale

Outline

- Denoising Diffusion Probabilistic Models (DDPMs)
 - Forward Process
 - Reverse Process
 - ELBO
 - Noise vs Data Prediction
 - **Inference**
 - Guided conditional generation
 - Classifier-free guidance
 - **DDIM**
- Other Diffusion Models
 - Score Matching & Score-based Models
 - Score SDEs

Inference: DDIM

- Sampling *for-loop* is slow.

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  $\mathbf{x}_0$ 
```

Inference: DDIM

- Let's skip some steps in the middle!



Inference: DDIM

- Let's skip some steps in the middle!



- Recall the data prediction formulation:

$$\hat{\mathbf{x}}_0 = \frac{\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}}$$

- We can jump to data prediction and jump back to arbitrary noisy step.
- This is called **denoising diffusion implicit model (DDIM)**.

Inference: DDIM

- We can jump to data prediction and jump back to arbitrary noisy step.

Algorithm 1: Fewer-Steps DDIM Sampling

```
1:  $\mathbf{x}_T \sim \mathcal{N}(0, I)$ 
2: for  $s$  from  $S$  to 1 do
3:    $t \leftarrow \tau_s$ 
4:    $t' \leftarrow \tau_{s-1}$ 
5:    $\hat{\epsilon} \leftarrow \epsilon_\theta(\mathbf{x}_t, t)$ 
6:    $\hat{\mathbf{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \hat{\epsilon})$ 
7:    $\mathbf{x}_{t'} \leftarrow \sqrt{\bar{\alpha}_{t'}} \hat{\mathbf{x}}_0 + \sqrt{1 - \bar{\alpha}_{t'}} \hat{\epsilon}$ 
8: end for
9: return  $\hat{\mathbf{x}}_0$ 
```

$[1, \dots, T] \implies [\tau_0 = 0, \dots, \tau_S = T]$, e.g., $\tau = [0, 10, 20, 30, \dots, 1000]$

Inference: DDIM

- Application: StableDiffusion v1.5 at huggingface.co

DDIMScheduler

`class diffusers.DDIMScheduler`

[<source>](#)

```
( num_train_timesteps: int = 1000, beta_start: float = 0.0001, beta_end: float = 0.02, beta_schedule:
str = 'linear', trained_betas: typing.Union[numpy.ndarray, typing.List[float], NoneType] = None,
clip_sample: bool = True, set_alpha_to_one: bool = True, steps_offset: int = 0, prediction_type: str =
'epsilon', thresholding: bool = False, dynamic_thresholding_ratio: float = 0.995, clip_sample_range:
float = 1.0, sample_max_value: float = 1.0, timestep_spacing: str = 'leading', rescale_betas_zero_snr:
bool = False )
```

Outline

- Denoising Diffusion Probabilistic Models (DDPMs)
 - Forward Process
 - Reverse Process
 - ELBO
 - Noise vs Data Prediction
 - Inference
- Other Diffusion Models
 - **Score Matching & Score-based Models**
 - Score SDEs

Score Matching & Score-based Models

- Recall the aforementioned **denoising score matching** (DSM) [10]:

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

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

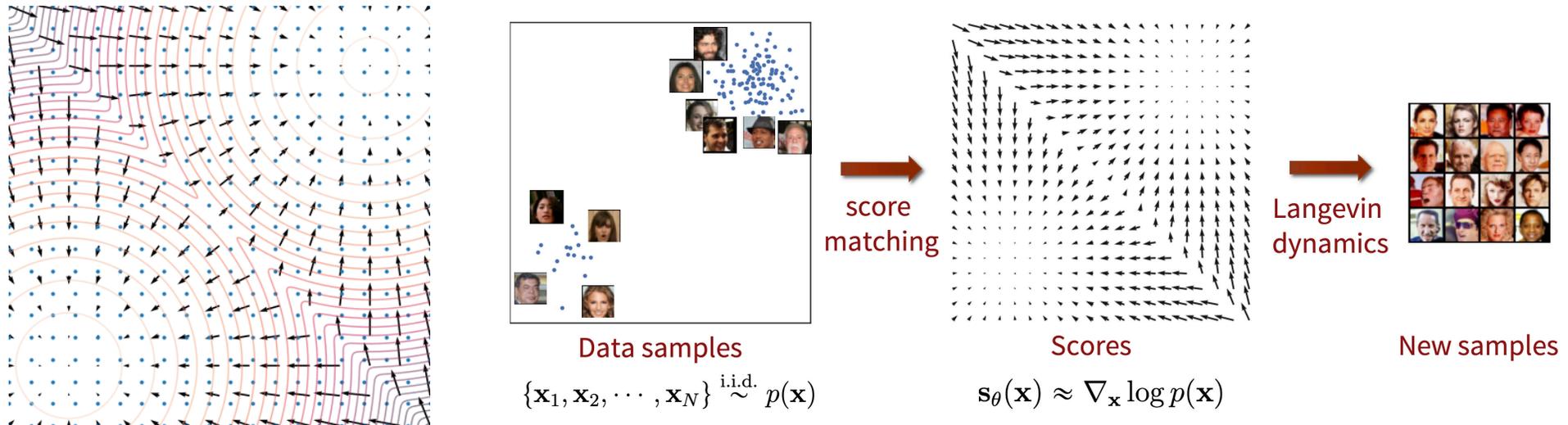
- Here, we introduce DSM within the DDPM objective. What if we train score-based models from scratch?

Score Matching & Score-based Models

- Score matching objective:

$$\mathbf{s}(\mathbf{x}) := \nabla_{\mathbf{x}} \log q(\mathbf{x})$$

- Interpretation of score function: the direction where data likelihood increases.

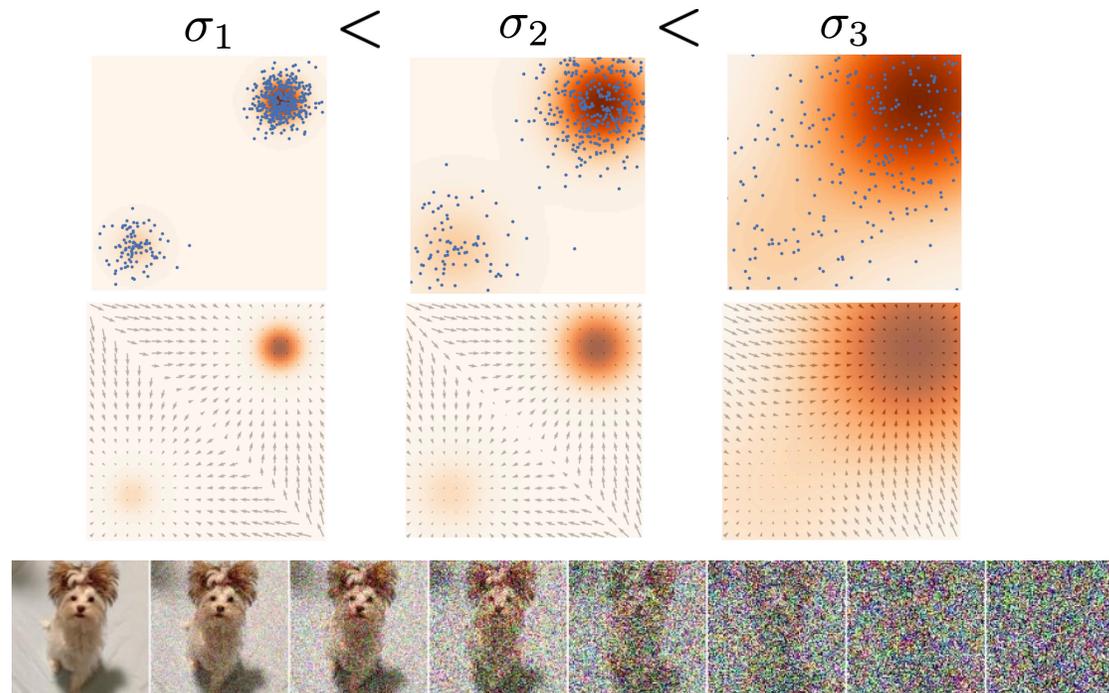


Score Matching & Score-based Models

- What if we train score-based models from scratch (using DSM)?

$$q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}_0) \sim \mathcal{N}(\mathbf{x}_0, \sigma_i^2 \mathbf{I})$$
$$\nabla_{\tilde{\mathbf{x}}} \log q_{\sigma_i}(\tilde{\mathbf{x}}) = \mathbb{E}_{q(\mathbf{x}_0)q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}_0)} [\nabla_{\tilde{\mathbf{x}}} \log q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}_0)]$$
$$\approx -\mathbf{s}_{\theta}(\tilde{\mathbf{x}}, \sigma_i)$$

where $\sigma_{\min} = \sigma_1 < \sigma_2 < \dots < \sigma_N = \sigma_{\max}$



Score Matching & Score-based Models

- What if we train score-based models from scratch (using DSM)?

$$\begin{aligned}q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}_0) &\sim \mathcal{N}(\mathbf{x}_0, \sigma_i^2 \mathbf{I}) \\ \nabla_{\tilde{\mathbf{x}}} \log q_{\sigma_i}(\tilde{\mathbf{x}}) &= \mathbb{E}_{q(\mathbf{x}_0)q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}_0)} [\nabla_{\tilde{\mathbf{x}}} \log q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}_0)] \\ &\approx -\mathbf{s}_{\theta}(\tilde{\mathbf{x}}, \sigma_i)\end{aligned}$$

where $\sigma_{\min} = \sigma_1 < \sigma_2 < \dots < \sigma_N = \sigma_{\max}$

- Training:

$$L = \sum_{i=1}^N \sigma_i^2 \mathbb{E}_{q(\mathbf{x}_0)} \mathbb{E}_{q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}_0)} \left[\|\mathbf{s}_{\theta}(\tilde{\mathbf{x}}, \sigma_i) - \nabla_{\tilde{\mathbf{x}}} \log q_{\sigma_i}(\tilde{\mathbf{x}} | \mathbf{x}_0)\|_2^2 \right]$$

Score Matching & Score-based Models

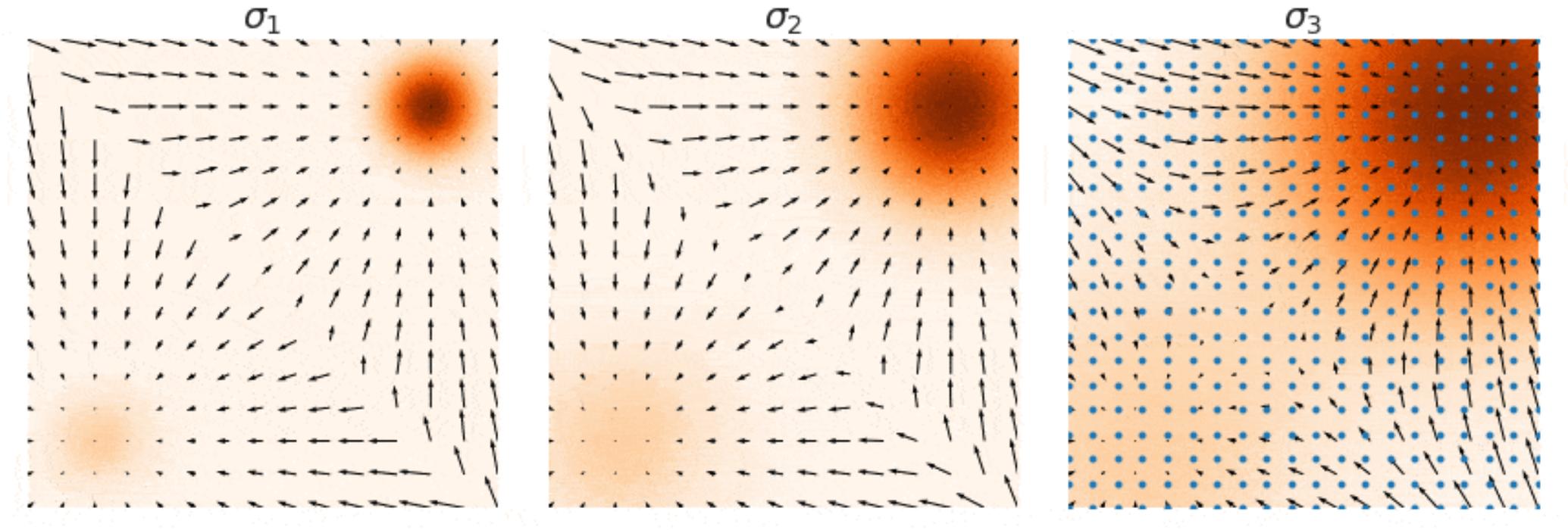
- Sampling:

Algorithm 1 Langevin MCMC Sampling from Score-Based Model

```
1: Input: Number of noise scales  $N$ , step sizes  $\{\epsilon_i\}_{i=1}^N$ , number of MCMC
   steps  $M$ , score model  $\mathbf{s}_\theta(\mathbf{x}, \sigma)$ 
2: Initialize:  $\mathbf{x}_N^0 \sim \mathcal{N}(\mathbf{0}, \sigma_N^2 \mathbf{I})$ 
3: for  $i = N, N - 1, \dots, 1$  do ▷ Loop over noise scales
4:   for  $m = 1$  to  $M$  do ▷ Langevin MCMC at one noise level
5:     Sample  $\mathbf{z}_i^m \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
6:      $\mathbf{x}_i^m \leftarrow \mathbf{x}_i^{m-1} + \epsilon_i \mathbf{s}_\theta(\mathbf{x}_i^{m-1}, \sigma_i) + \sqrt{2\epsilon_i} \mathbf{z}_i^m$ 
7:   end for
8:   if  $i > 1$  then
9:      $\mathbf{x}_{i-1}^0 \leftarrow \mathbf{x}_i^M$ 
10:  end if
11: end for
12: Output: Sample  $\mathbf{x}_1^M \sim p_{\sigma_{\min}}(\mathbf{x})$ 
```

Score Matching & Score-based Models

- Sampling:



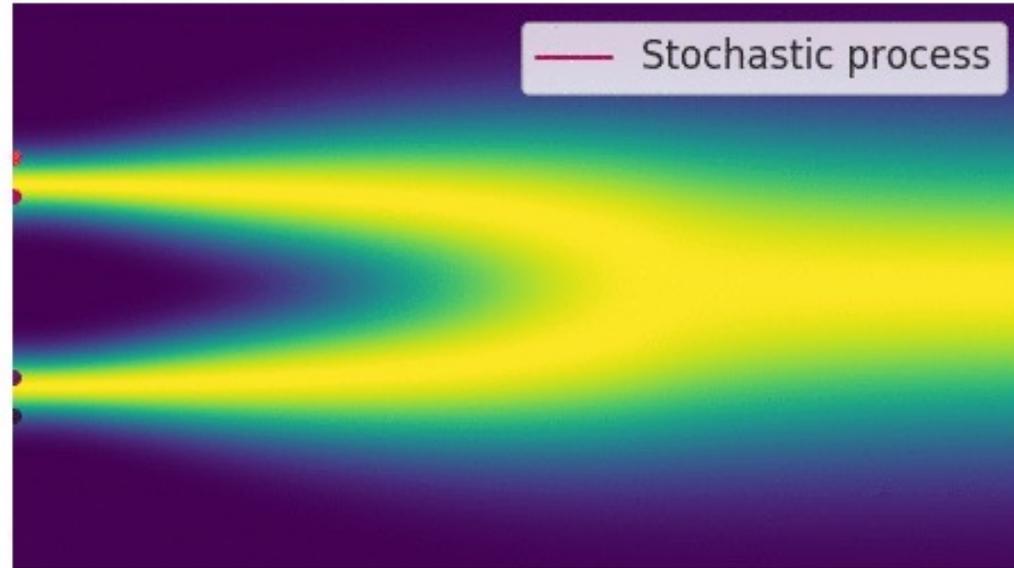
Outline

- Denoising Diffusion Probabilistic Models (DDPMs)
 - Forward Process
 - Reverse Process
 - ELBO
 - Noise vs Data Prediction
 - Inference
- Other Diffusion Models
 - Score Matching & Score-based Models
 - **Score SDEs**

Score SDEs

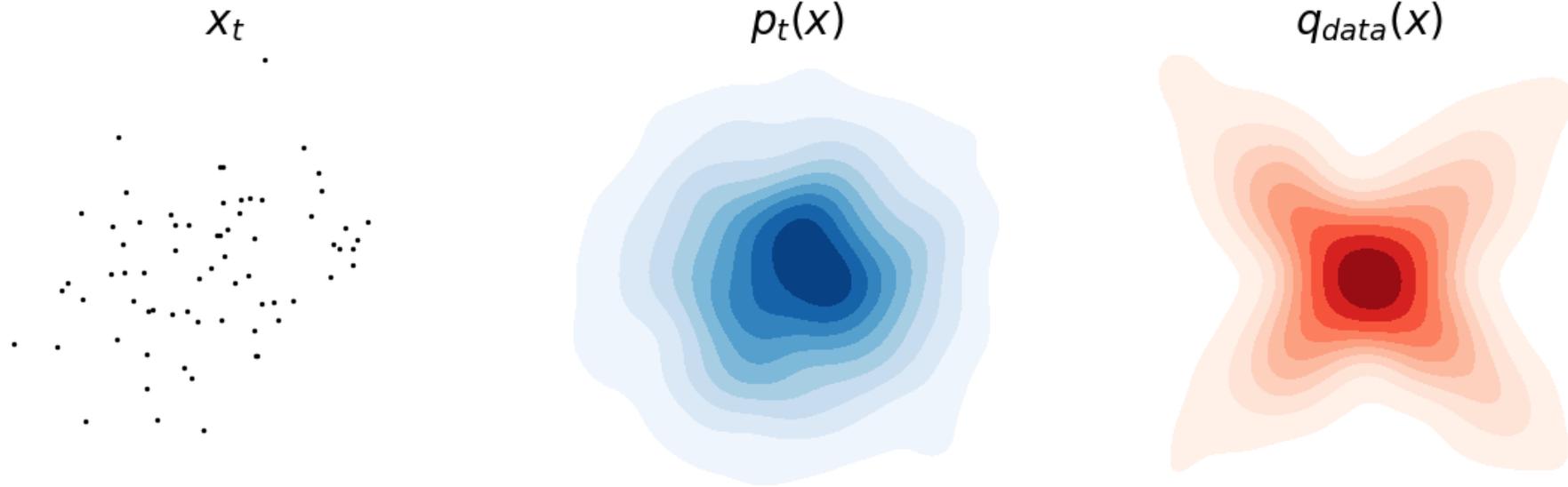
- What happens if we generalize the number of noise scales to infinity?

$$q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}_0) \sim \mathcal{N}(\mathbf{x}_0, \sigma_i^2 \mathbf{I})$$



Score SDEs

- What happens if we generalize the number of noise scales to infinity?



Score SDEs

- We use stochastic differential equation (SDEs) to represent the forward process:

$$d\mathbf{x} = \underbrace{\mathbf{f}(\mathbf{x}, t)}_{\text{Drift term (deterministic)}} dt + \underbrace{g(t)}_{\text{Diffusion term (stochastic)}} d\mathbf{w}$$

Wiener process

- SDE solutions $\{\mathbf{x}(t)\}_{t \in [0, T]}$ follow the distributions of $p_t(\mathbf{x})$, which are analogous to $p_{\sigma_i}(\mathbf{x})$ in the vanilla score-based models.
 - $p_T(\mathbf{x})$ tractable prior
 - $p_0(\mathbf{x})$ data distribution

Score SDEs

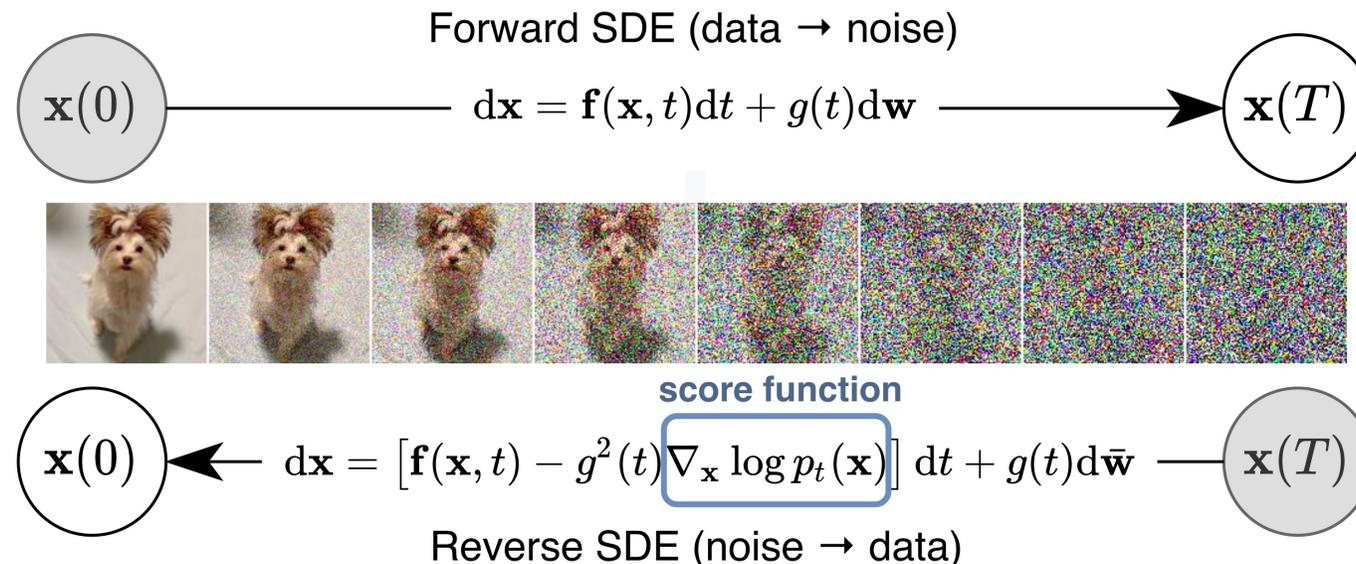
- We use stochastic differential equation (SDEs) to represent the forward process:

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

- Reverse-time SDE corresponds to the sampling process:

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

↓
Score function



Score SDEs

- We use stochastic differential equation (SDEs) to represent the forward process:

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

- Reverse-time SDE corresponds to the sampling process:

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

- Training (based on forward SDEs):

$$\mathbb{E}_{t \in \mathcal{U}(0, T)} \mathbb{E}_{p_t(\mathbf{x})} [\lambda(t) \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\theta}(\mathbf{x}, t)\|_2^2]$$

↓
Denoising score matching
(or other variants)

Score SDEs

- We use stochastic differential equation (SDEs) to represent the forward process:

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

- Reverse-time SDE corresponds to the sampling process:

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

- Training (based on forward SDEs):

$$\mathbb{E}_{t \in \mathcal{U}(0, T)} \mathbb{E}_{p_t(\mathbf{x})} [\lambda(t) \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\theta}(\mathbf{x}, t)\|_2^2]$$

- Neural reverse SDE:

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\mathbf{s}_{\theta}(\mathbf{x}, t)]dt + g(t)d\mathbf{w}$$

Score SDEs

- Sampling (based on reverse SDEs):

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\mathbf{s}_\theta(\mathbf{x}, t)]dt + g(t)d\mathbf{w}$$

$$\Delta\mathbf{x} \leftarrow [\mathbf{f}(\mathbf{x}, t) - g^2(t)\mathbf{s}_\theta(\mathbf{x}, t)]\Delta t + g(t)\sqrt{|\Delta t|}\mathbf{z}_t$$

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}, t \leftarrow t + \Delta t, \mathbf{z}_t \sim \mathcal{N}(0, I)$$

- Numerical solver design space:
 - Euler-Maruyama
 - Adaptive step size
 - High-order SDE solver

Score SDEs

- Sampling (based on reverse SDEs):

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\mathbf{s}_\theta(\mathbf{x}, t)]dt + g(t)d\mathbf{w}$$

- Transforming SDEs into probability flow ODEs (without noise):

$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right] dt$$

- Both share the same marginal distributions $\{p_t(\mathbf{x})\}_{t \in [0, T]}$
- ODEs allow for exact log-likelihood evaluation.

References

- [1] <https://cvpr2023-tutorial-diffusion-models.github.io/>
- [2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>
- [3] <https://github.com/CompVis/stable-diffusion?tab=readme-ov-file>
- [4] Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*. 2020;33:6840-51.
- [5] Karras T, Aittala M, Aila T, Laine S. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*. 2022 Dec 6;35:26565-77.
- [6] Ho J, Saharia C, Chan W, Fleet DJ, Norouzi M, Salimans T. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*. 2022;23(47):1-33.
- [7] Rombach R, Blattmann A, Lorenz D, Esser P, Ommer B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition 2022* (pp. 10684-10695).
- [8] Zhang L, Rao A, Agrawala M. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision 2023* (pp. 3836-3847).
- [9] Dhariwal P, Nichol A. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*. 2021 Dec 6;34:8780-94.
- [10] Vincent P. A connection between score matching and denoising autoencoders. *Neural computation*. 2011 Jul;23(7):1661-74.

References

- [11] Ho J, Salimans T. Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598. 2022 Jul 26.
- [12] Song J, Meng C, Ermon S. Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502. 2020 Oct 6.
- [13] https://huggingface.co/blog/stable_diffusion
- [14] <https://www.midjourney.com/explore?tab=top>
- [15] <https://openai.com/sora/?shareId=15>
- [16] <https://chat.inceptionlabs.ai/>
- [17] Song Y, Sohl-Dickstein J, Kingma DP, Kumar A, Ermon S, Poole B. Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456. 2020 Nov 26.
- [18] <https://yang-song.net/blog/2021/score/>
- [19] <https://iclr-blogposts.github.io/2024/blog/diffusion-theory-from-scratch/>
- [20] Strümke I, Langseth H. Lecture Notes in Probabilistic Diffusion Models. arXiv preprint arXiv:2312.10393. 2023 Dec 16.
- [21] <https://www.cs.toronto.edu/~duvenaud/courses/csc2541/slides/structured-encoders-decoders.pdf>
- [22] Xiao Z, Kreis K, Vahdat A. Tackling the generative learning trilemma with denoising diffusion GANs. arXiv preprint arXiv:2112.07804. 2021 Dec 15.

Questions?