CPEN 455: Deep Learning

Lecture 11: Diffusion Models

Felix Fu, Qi Yan, Renjie Liao

University of British Columbia

Winter, Term 2, 2024

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.

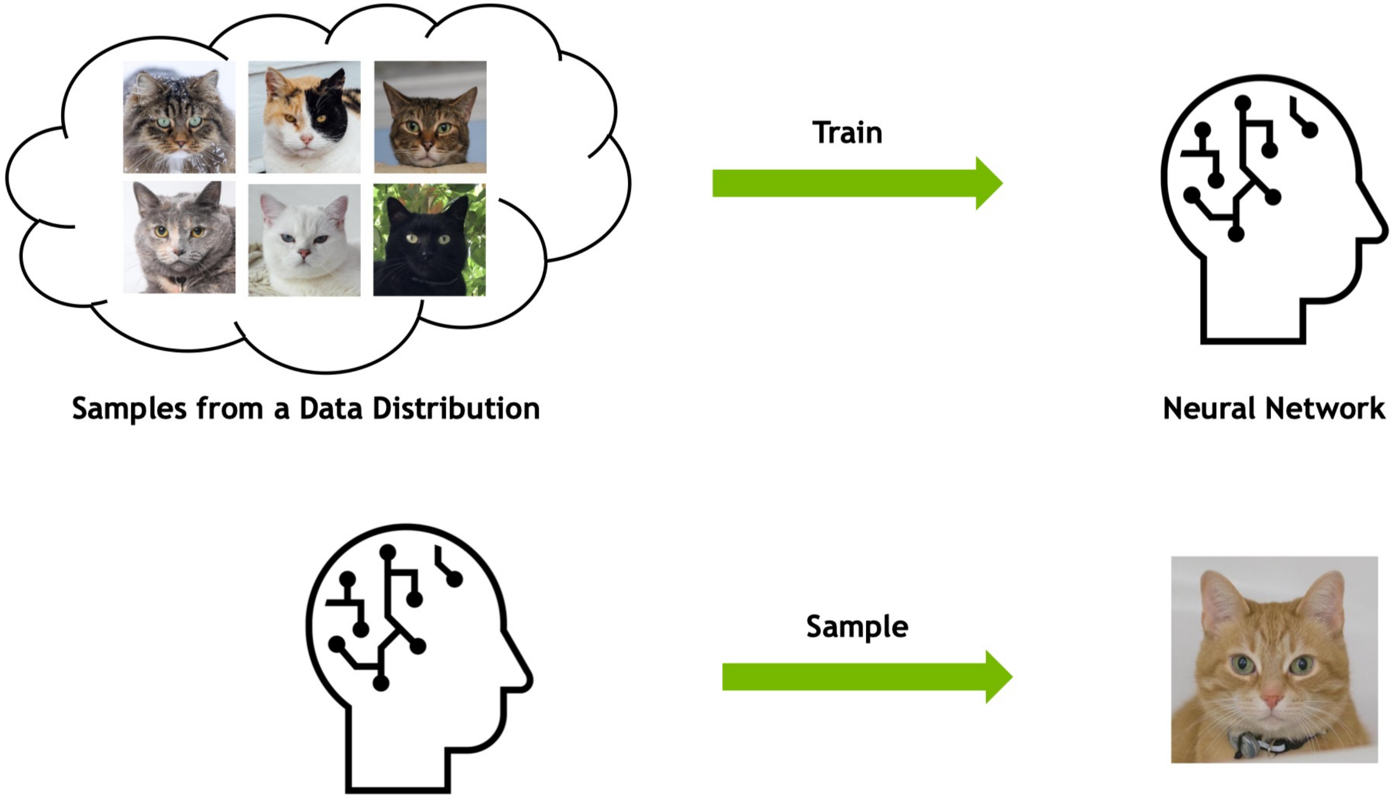


Image credit: [1]

The Landscape of Deep Generative Learning

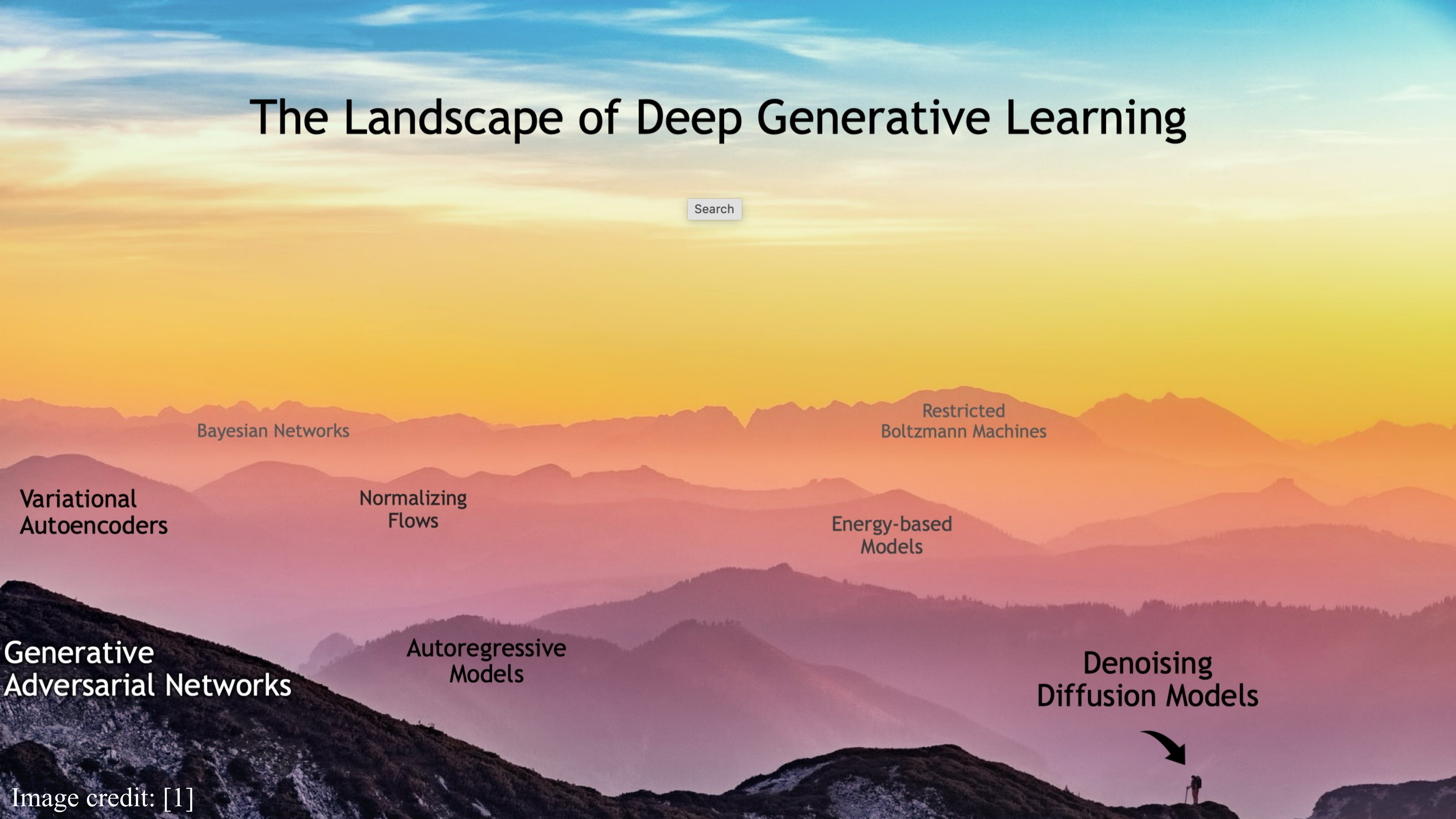


Image credit: [1]

Generative models

GAN: Adversarial training

Last lecture: **VAE:** maximize variational lower bound

Flow-based models: Invertible transform of distributions

Diffusion models: Gradually add Gaussian noise and then reverse

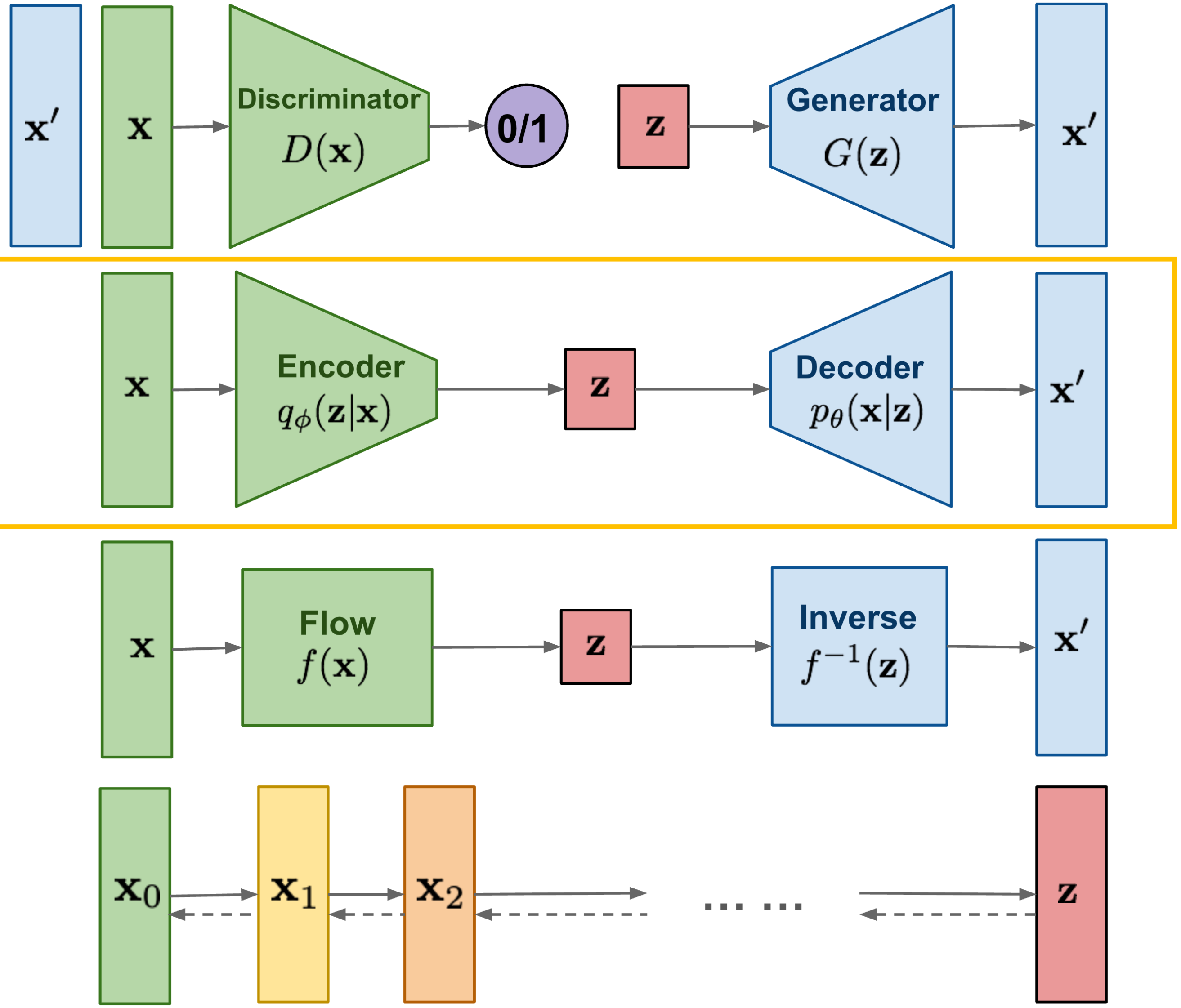
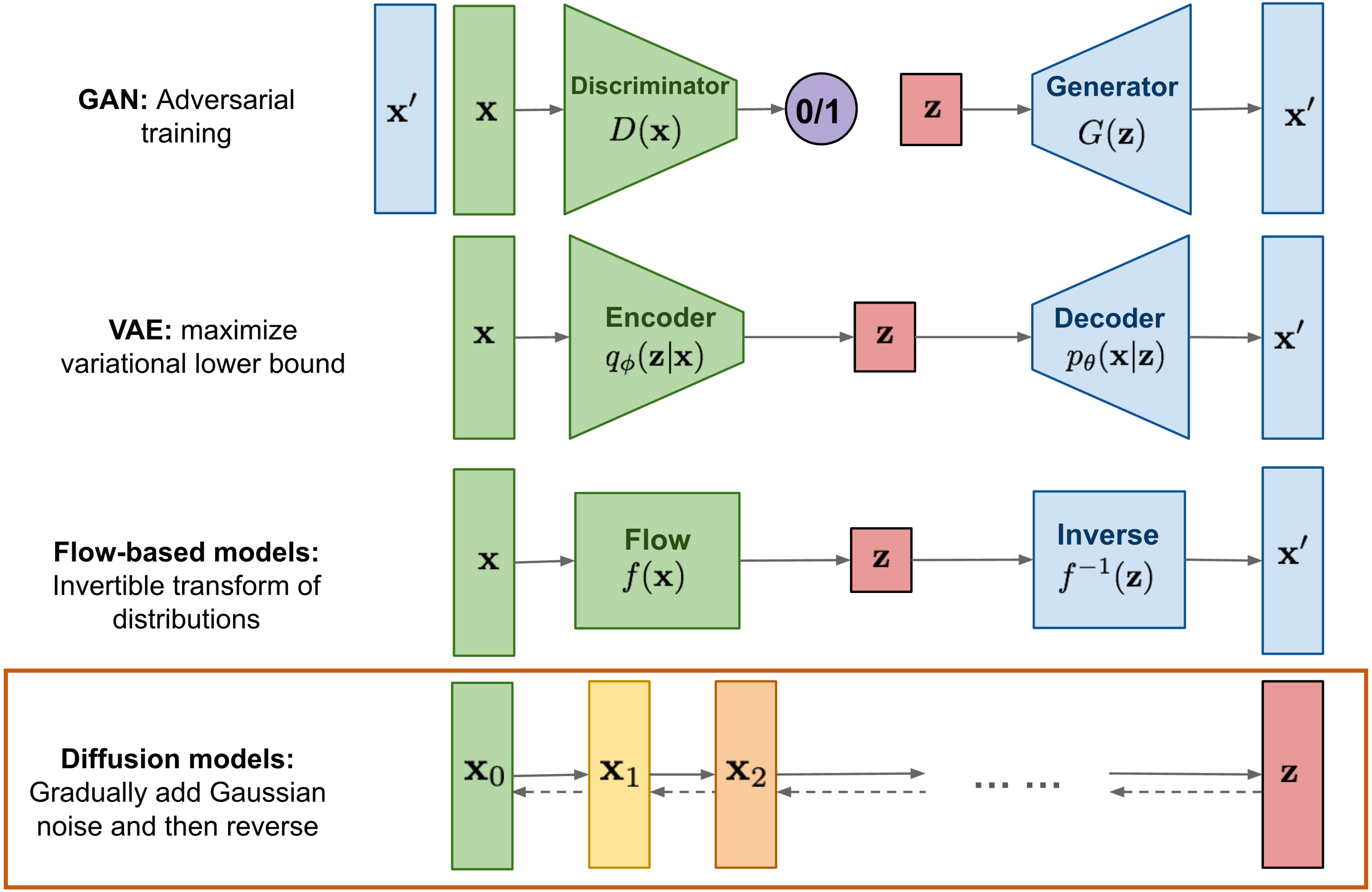


Image credit: [2]

Generative models



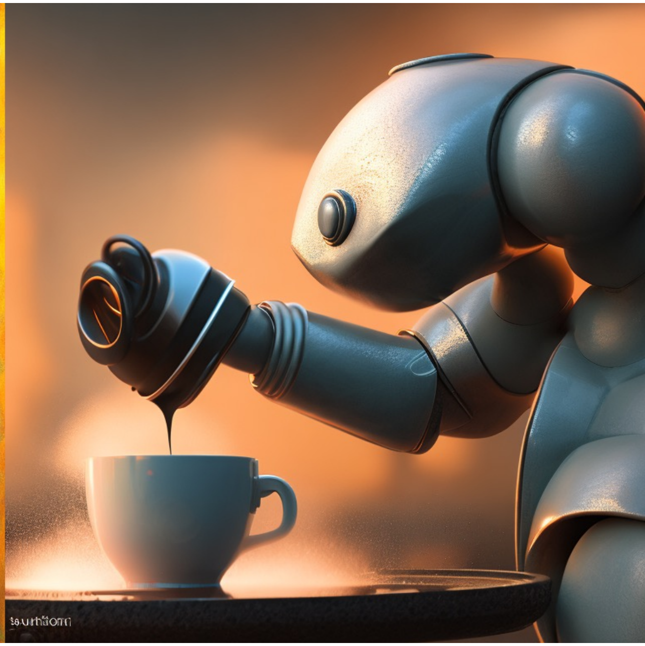
This lecture:

Image credit: [2]

AI Artworks

2021



Stable Diffusion

Image credit: [3]

AI Artworks

2022

Midjourney

Image credit: [14]

A fast-involving field

2024

Sora

Video credit: [15]

A fast-involving field

2025

Mercury Coder

Video credit: [16]

Motivations

2015

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 that

- converts a simple distribution into a target distribution.
- has an analytically evaluable probability at each step, thus the full chain.
- based on non-equilibrium statistical physics.

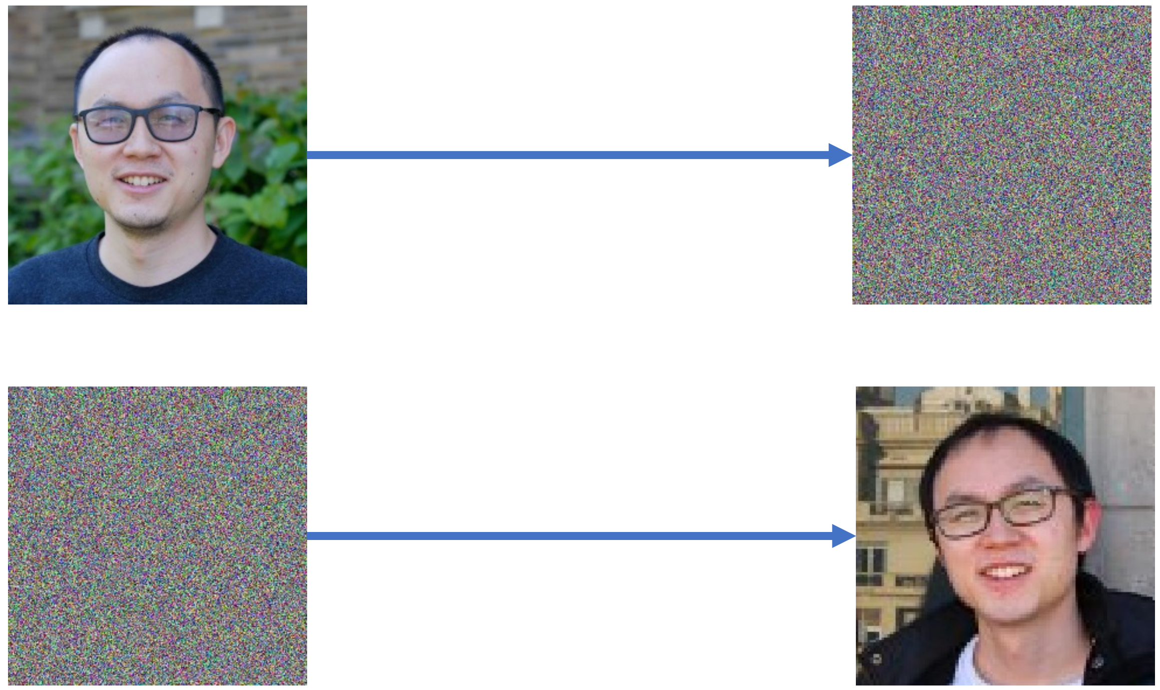
Motivations

2015

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

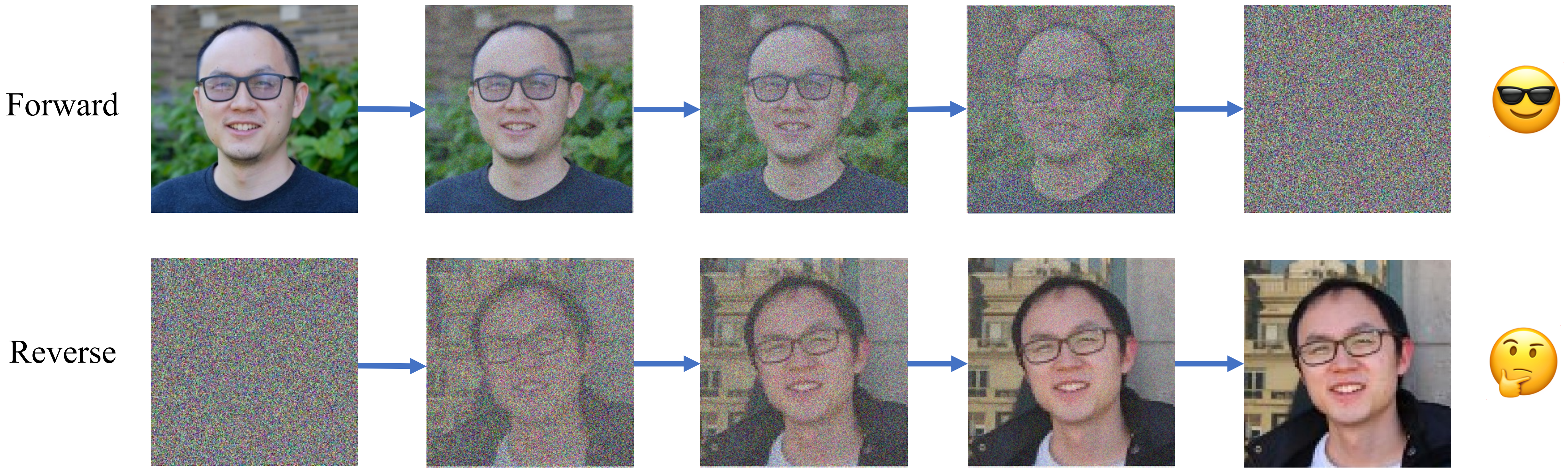


What does this remind you of ?

Motivations

2015

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

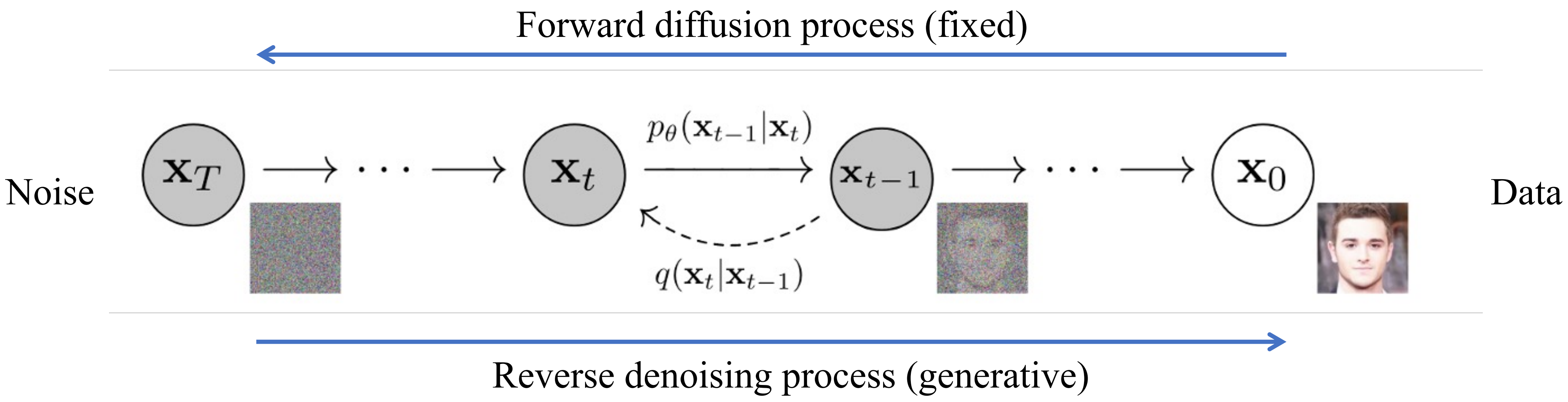


Denoising Diffusion Probabilistic Models

2020

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

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

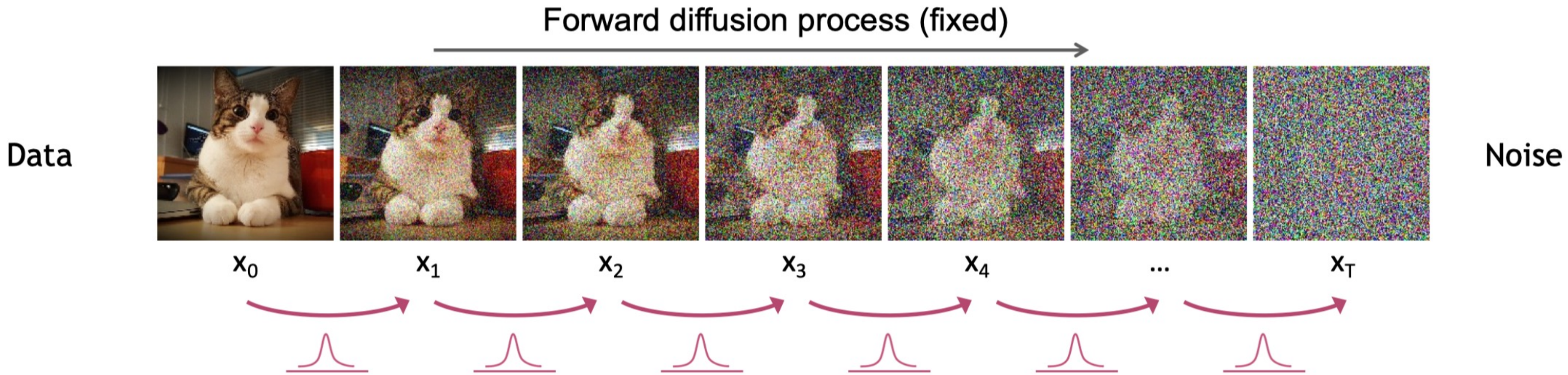
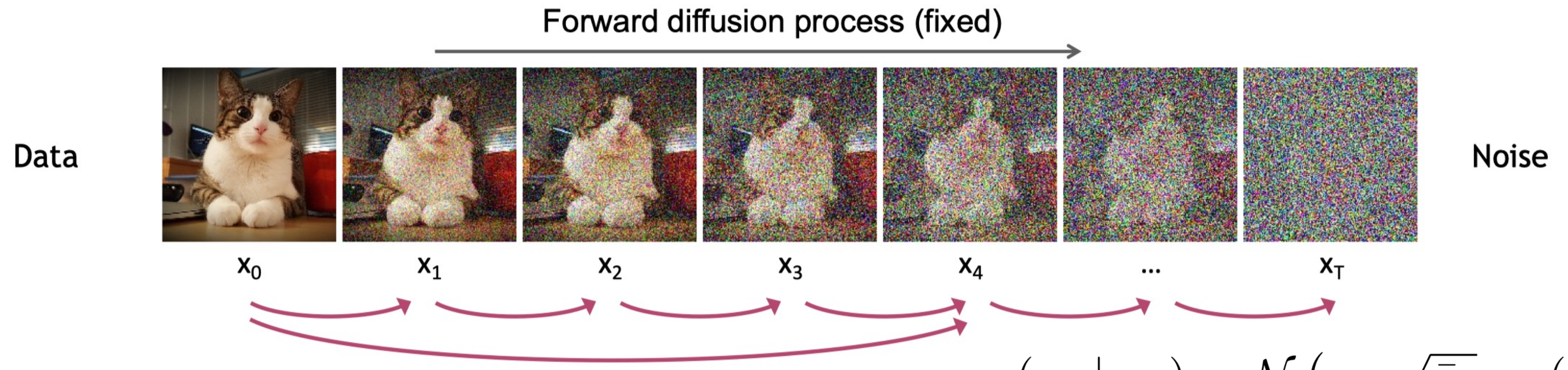


Image credit: [1]

Forward Process



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

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

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

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

Image credit: [1]

Forward Process

- The re-parametrization trick.
- Blackboard time!

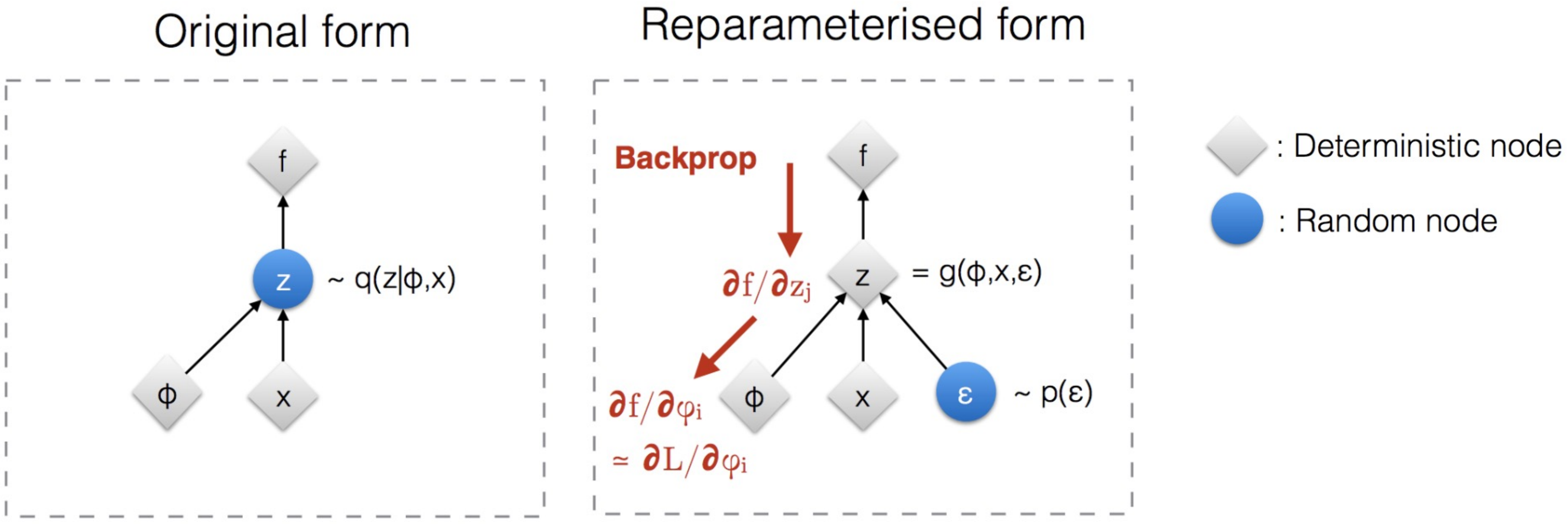


Image credit: [21]

Noise Schedule

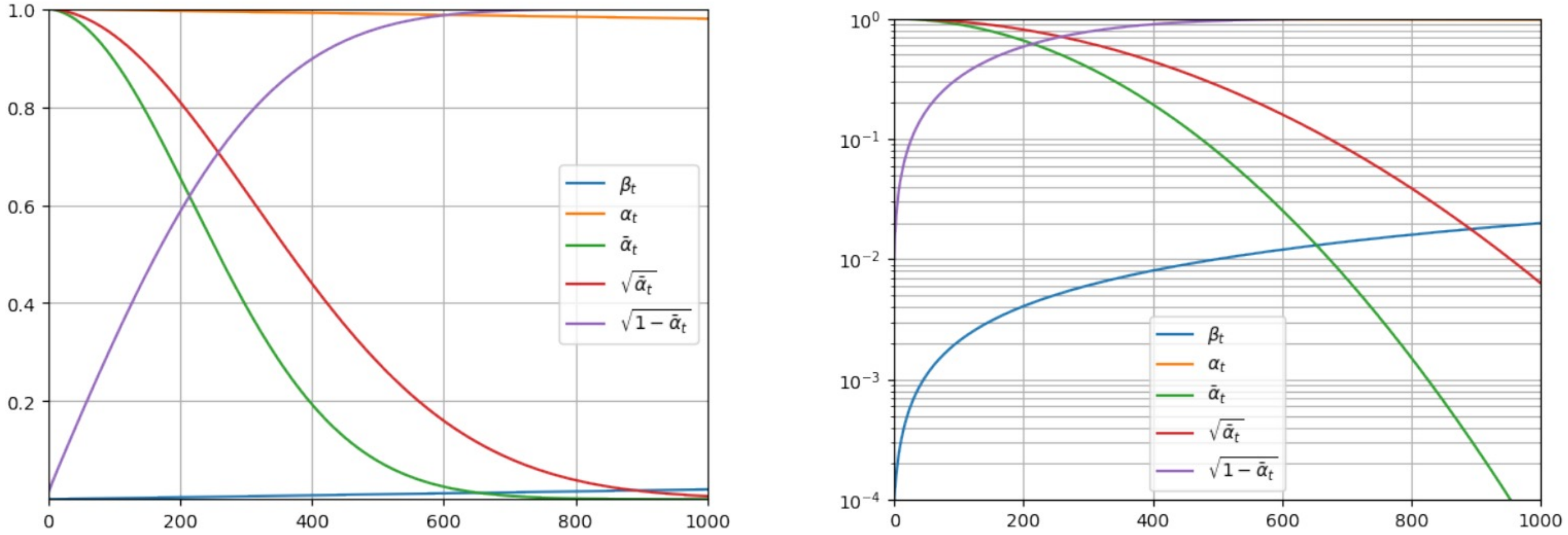


Figure 2: Parameter values for $\beta = [10^{-4}, 0.02]$ over 1000 time steps t using a linear schedule. The information in the two figures are the same, but the right-hand side uses log-scale on the y-axis to show the speed of which $\bar{\alpha}_t$ goes towards zero.

Ref: [9]

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

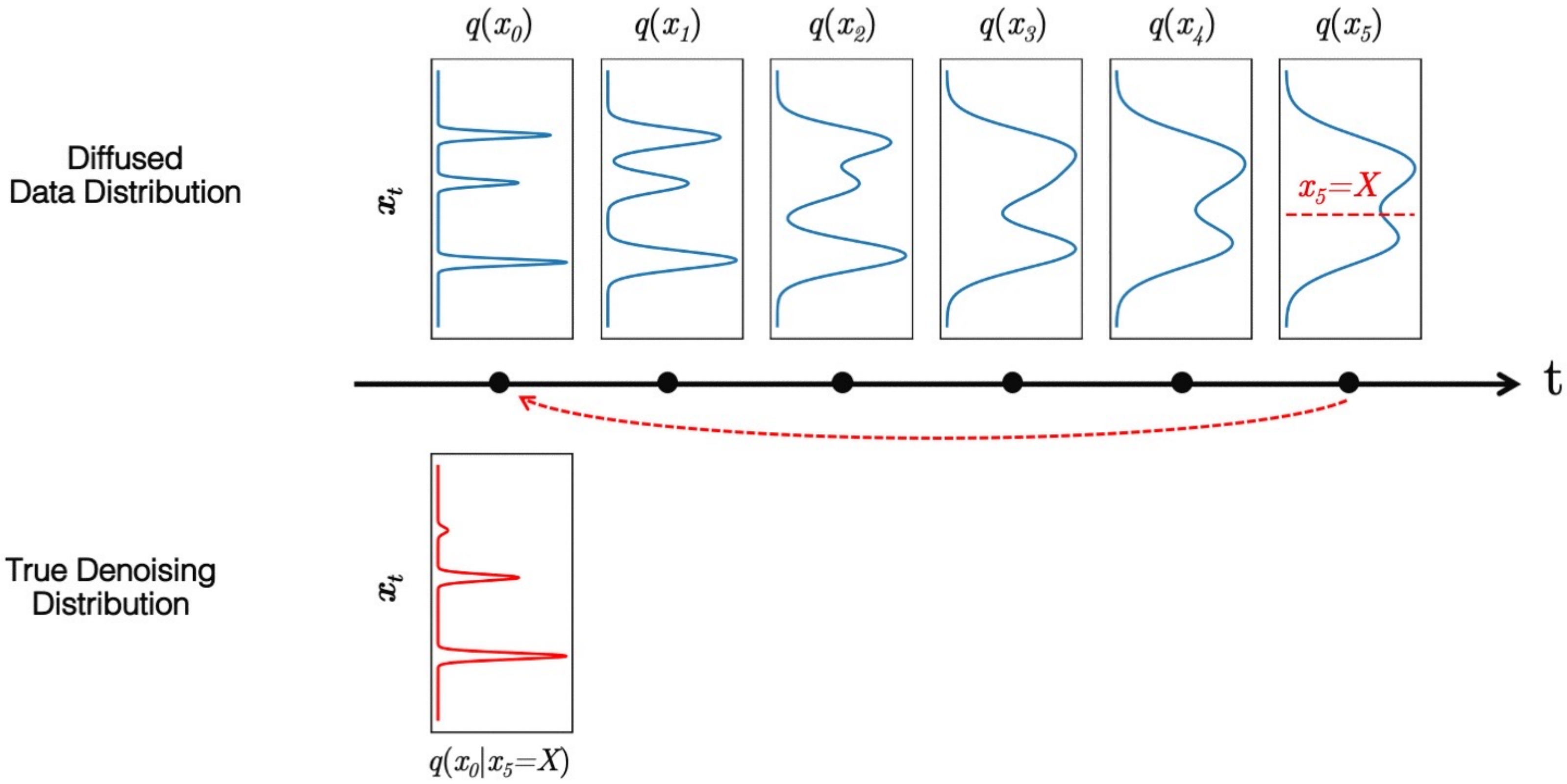


Image credit: [22]

Reverse Process

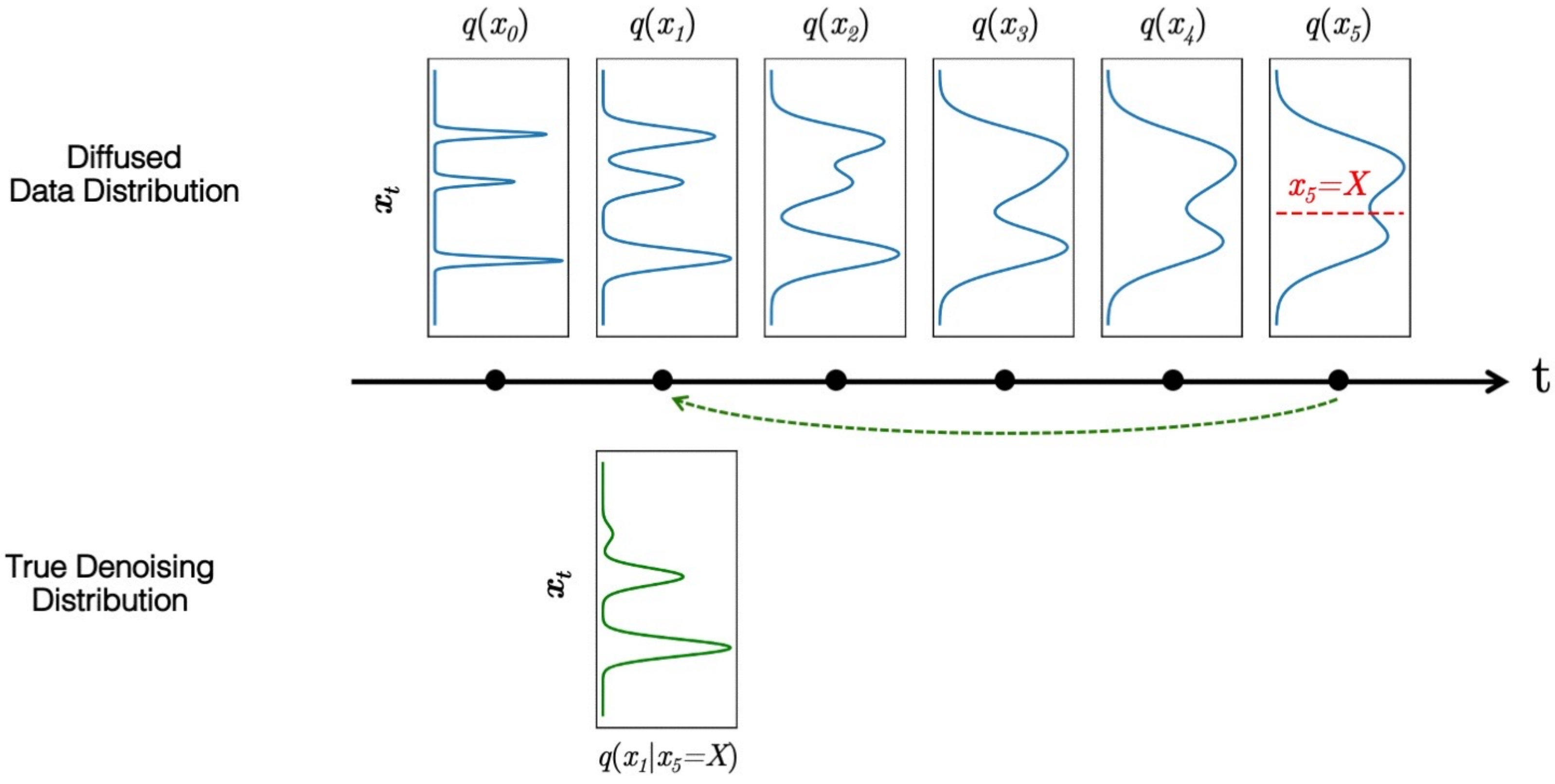


Image credit: [22]

Reverse Process

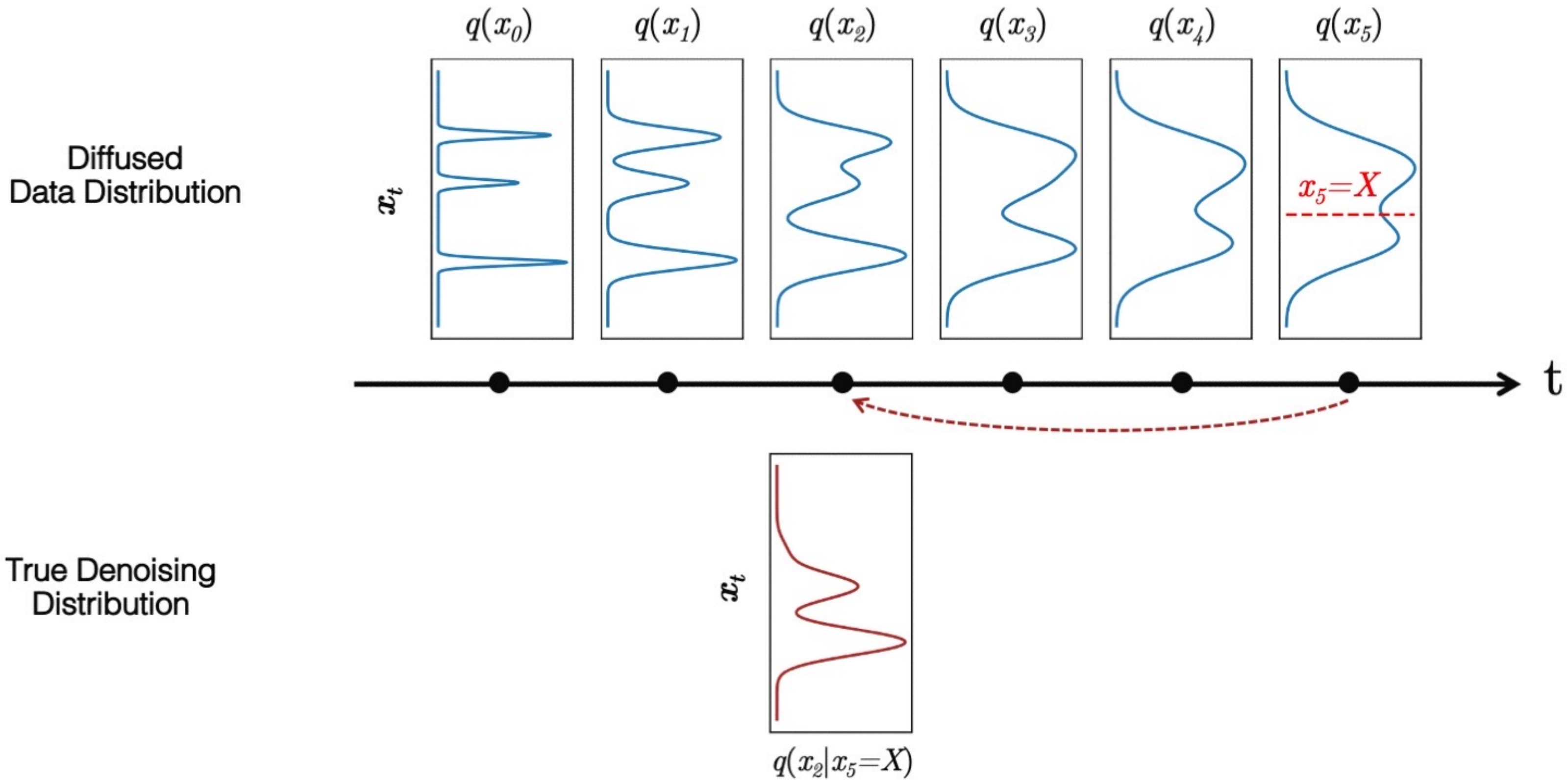


Image credit: [22]

Reverse Process

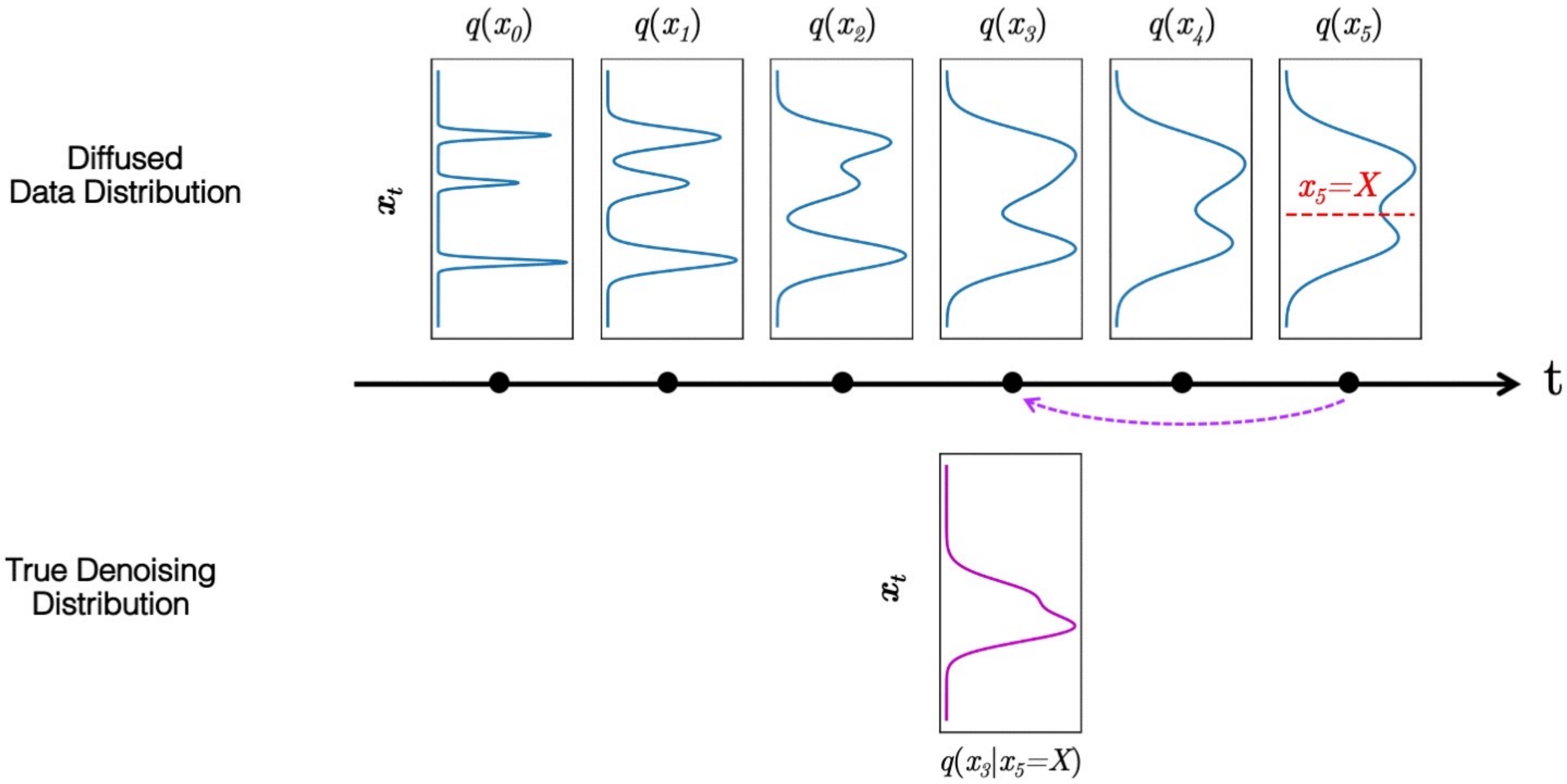


Image credit: [22]

Reverse Process

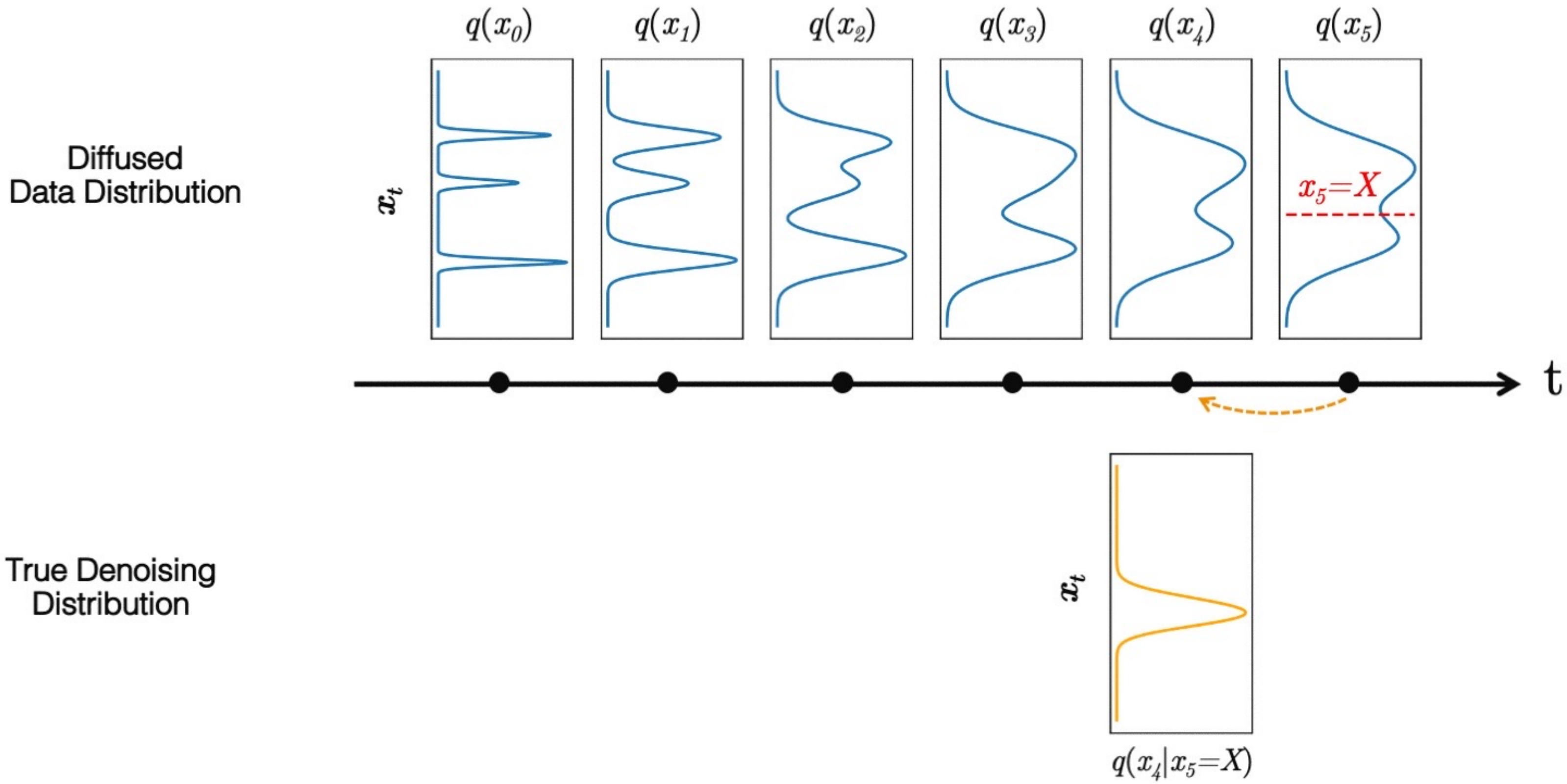


Image credit: [22]

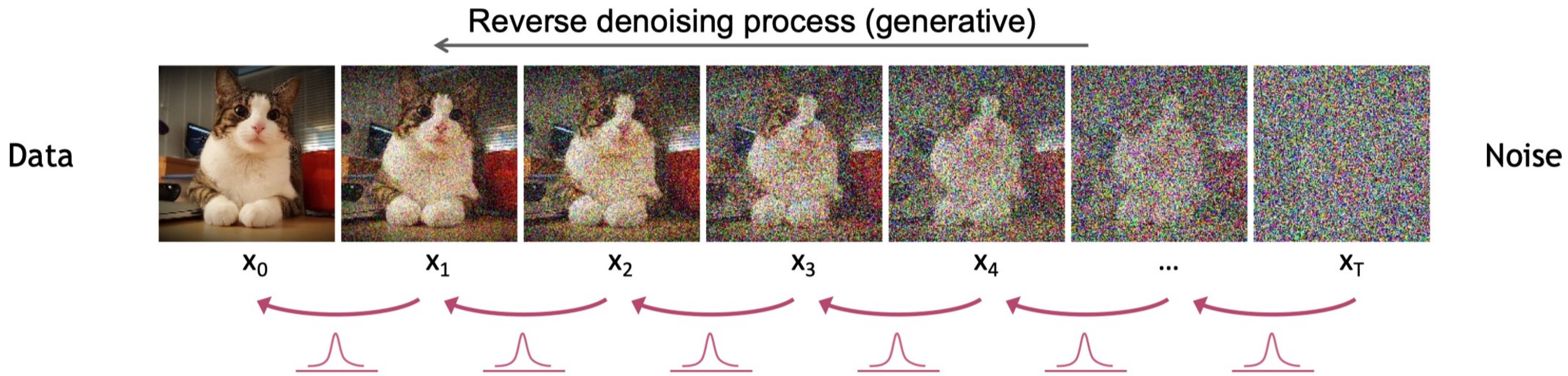
Reverse Process

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

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

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

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



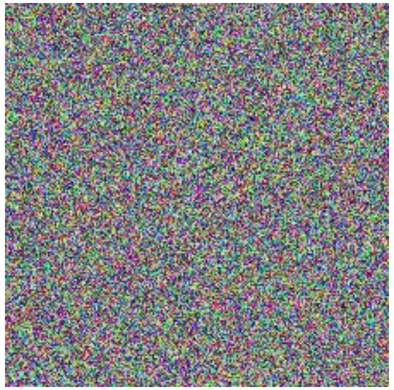


Image credit: [1]

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. Yay!

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

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

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

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

ELBO

- Connection with Variational Autoencoders

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

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

ELBO

- Connection with Variational Autoencoders

$$\begin{aligned}
\log p(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\mid\mathbf{x})}\right] + \mathcal{D}_{\mathrm{KL}}\left(q_\phi(\mathbf{z}\mid\mathbf{x}) \,\|\, p_\theta(\mathbf{z}\mid\mathbf{x})\right) \\
&\geq \boxed{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}\mid\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z}\mid\mathbf{x})}\right]} \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}\mid\mathbf{z})\right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{z})}{q_\phi(\mathbf{z}\mid\mathbf{x})}\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}\mid\mathbf{z})\right]}_{\text{reconstruction term}} - \underbrace{\mathcal{D}_{\mathrm{KL}}\left(q_\phi(\mathbf{z}\mid\mathbf{x}) \,\|\, p(\mathbf{z})\right)}_{\text{prior matching term}}
\end{aligned}$$

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

ELBO

$$\log p(\mathbf{x}_0) \geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{\prod_{t=1}^{T} \boxed{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})}}\right]$$

Intractable!

$$\mathbb{E}_{q(\mathbf{x}_0)}\Big[\log p_\theta(\mathbf{x}_0)\Big] \geq \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log p_\theta(\mathbf{x}_T) - \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big]$$

$$= \mathbb{E}_q\Big[\log p_\theta(\mathbf{x}_T) - \sum_{t=2}^{T} \log \Big(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}\Big) - \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big]$$

Markov Chain

$$= \mathbb{E}_q\Big[\log p_\theta(\mathbf{x}_T) - \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} - \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big]$$

Logarithmic rules

$$= \mathbb{E}_q\Big[\log p_\theta(\mathbf{x}_T) - \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} - \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big]$$

Telescoping products

$$= \mathbb{E}_q\Big[\log \frac{p_\theta(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} - \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\Big]$$

$$= \mathbb{E}_q[\underbrace{-\mathcal{D}_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}_{\text{Prior Matching } (L_T)} - \sum_{t=2}^{T} \underbrace{\mathcal{D}_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{\text{Consistency } (L_t)} \underbrace{+ \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{\text{Reconstruction } (L_0)}]$$

Parameterizing DDPM

- KL divergence has a simple form between Gaussians.

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

Parameterizing DDPM

- KL divergence has a simple form between Gaussians.

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

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

Parameterizing DDPM

- KL divergence has a simple form between Gaussians.

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

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

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

Parameterizing DDPM

- KL divergence has a simple form between Gaussians.

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

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

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

- Trainable network predicts the noise mean.

Parameterizing DDPM

- KL divergence has a simple form between Gaussians.

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

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

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

- 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.

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

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

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

- 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

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

Simplified Training Objective

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

- λ_t adjusts the weights for correct maximum likelihood estimation.

Simplified Training Objective

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

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

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

Training and Inference

Algorithm 1 Training

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

Algorithm 2 Sampling

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

Ref: [5]

Generated Samples

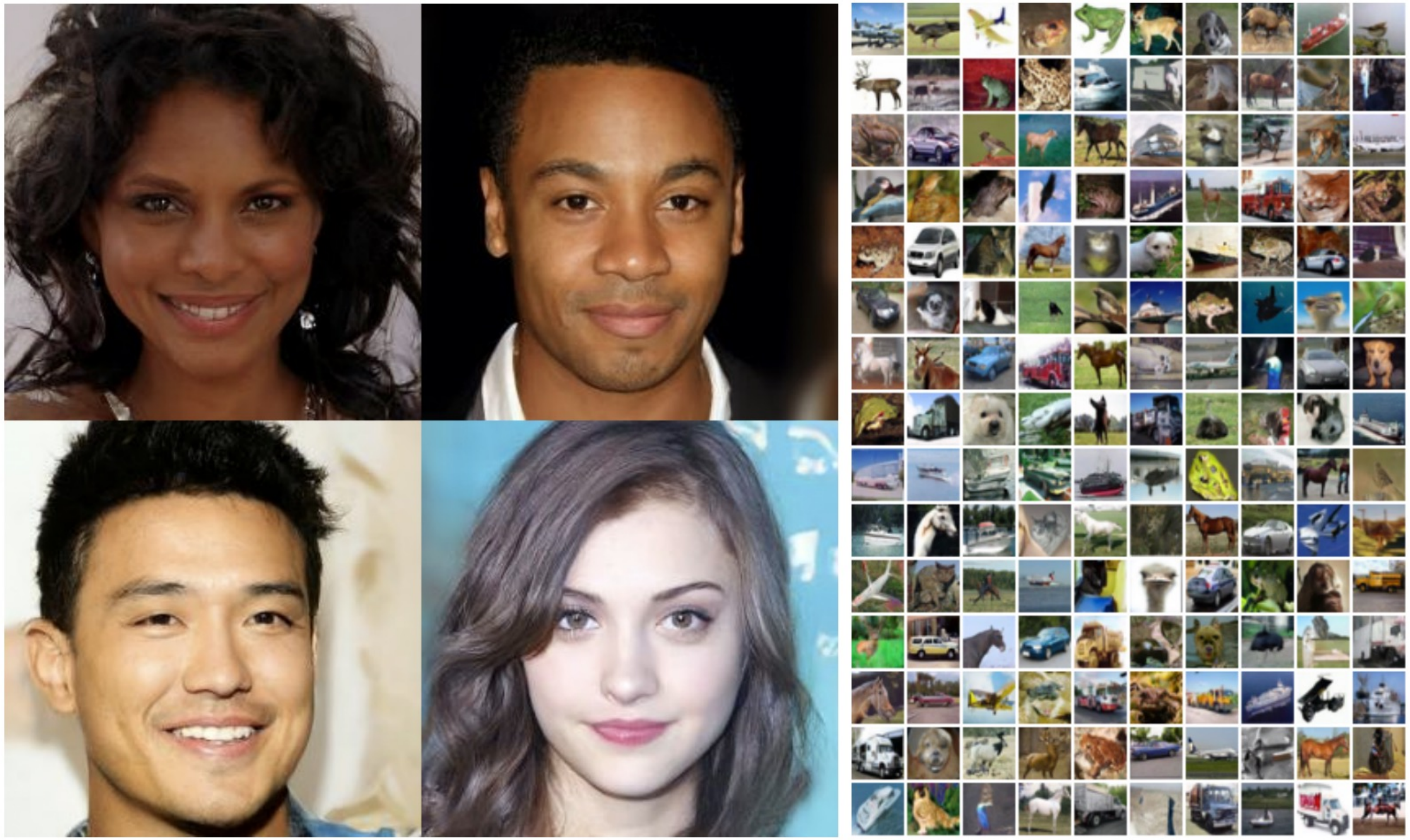


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

Image credit: [5]

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:

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

Noise vs Data Prediction

- DDPM noise estimation loss:

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon},t}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t,\, t\right)\right\|^2\right]$$

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

- Recall the forward process:

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

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

Noise vs Data Prediction

- DDPM noise estimation loss:

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

- Recall the forward process:

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

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

- Let’s use x_0 centered parameterization:

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

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

Noise vs Data Prediction

- DDPM data estimation loss:

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

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

Ref: [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)

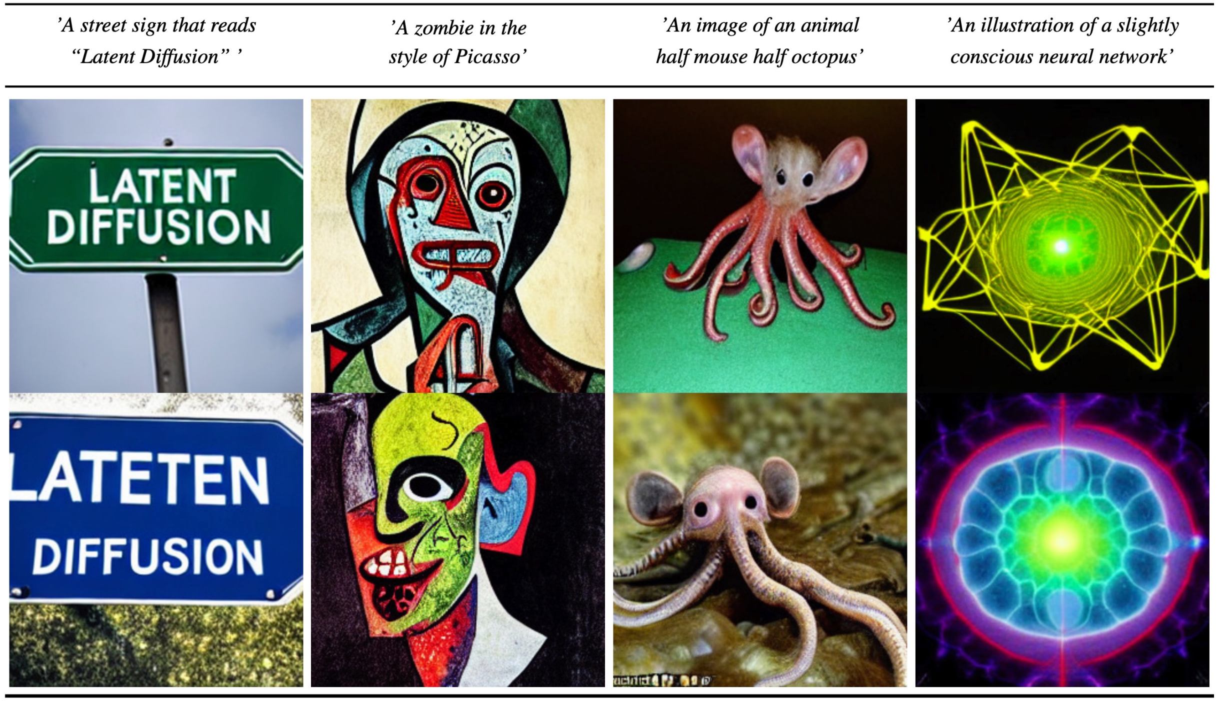
Unconditional generation

Class-conditional generation

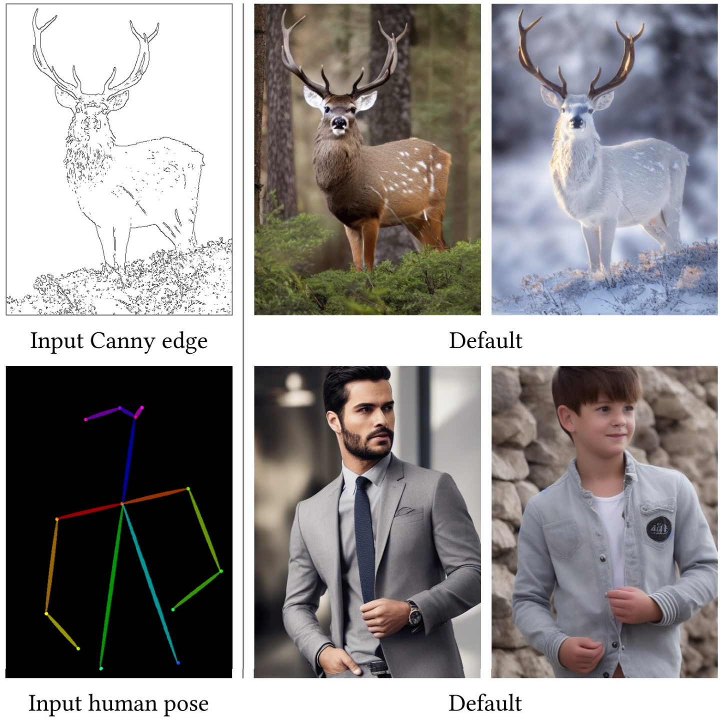
Image credit: [6]

Inference: guided conditional generation

- Goal: generation with *conditions* (controllability)



Text-to-Image generation



Visual cue-based generation

Image credit: [7, 8]

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:

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

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

Ref: [9]

Inference: classifier guidance

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

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

$$\log \underbrace{\boxed{q(\mathbf{x}_t|y)}}_{\text{Cond. distribution}} = \log \underbrace{\boxed{q(\mathbf{x}_t)}}_{\text{Uncond. distribution}} + \log \underbrace{\boxed{q(y|\mathbf{x}_t)}}_{\text{Classifier objective}} - \log q(y)$$

Ref: [9]

Inference: classifier guidance

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

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

$$\log \underbrace{\boxed{q(\mathbf{x}_t|y)}}_{\text{Cond. distribution}} = \log \underbrace{\boxed{q(\mathbf{x}_t)}}_{\text{Uncond. distribution}} + \log \underbrace{\boxed{q(y|\mathbf{x}_t)}}_{\text{Classifier objective}} - \log q(y)$$

- Take gradient w.r.t. data:

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

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

Ref: [9]

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:

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

Ref: [10]

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

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

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

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

Ref: [10]

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:

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

- 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:

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

Ref: [9]

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 $\mathcal{N}(\mu + \boxed{s\Sigma \nabla_{x_t} \log f_\phi(y|x_t)}, \Sigma)$
end for
return x_0

Image credit: [9]

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

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

  - 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

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

  - 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.

$$\text{Unconditional:} \quad \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) = -\frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y=\varnothing)}{\sqrt{1-\bar{\alpha}_t}}$$

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

Ref: [11]

Inference: classifier-free guidance

- Drawbacks of classifier guidance

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

  - 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.

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

Ref: [11]

Inference: classifier-free guidance

- Drawbacks of classifier guidance

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

  - 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.

$$\begin{aligned}
\nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) &= \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|y) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \\
&= -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\Big(\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y=\varnothing)\Big) \\
\bar{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t, y) &= \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - \sqrt{1-\bar{\alpha}_t}\, w \nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) \\
&= \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) + w\boxed{\big(\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\big)} \\
&= (w+1)\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - w\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)
\end{aligned}$$

Guidance scale

Ref: [11]

Inference: classifier-free guidance

Algorithm 1 Joint training a diffusion model with classifier-free guidance

Require: p_{uncond}: probability of unconditional training

repeat
- $(\mathbf{x}, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{c})$ ▷ Sample data with conditioning from the dataset
- $\mathbf{c} \leftarrow \varnothing$ with probability p_{uncond} ▷ Randomly discard conditioning to train unconditionally
- $\lambda \sim p(\lambda)$ ▷ Sample log SNR value
- $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- $\mathbf{z}_\lambda = \alpha_\lambda \mathbf{x} + \sigma_\lambda \boldsymbol{\epsilon}$ ▷ Corrupt data to the sampled log SNR value
- Take gradient step on $\nabla_\theta \left\| \boldsymbol{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) - \boldsymbol{\epsilon} \right\|^2$ ▷ Optimization of denoising model

until converged

Algorithm 2 Conditional sampling with classifier-free guidance

Require: w: guidance strength
Require: $\mathbf{c}$: conditioning information for conditional sampling
Require: $\lambda_1, \ldots, \lambda_T$: increasing log SNR sequence with $\lambda_1 = \lambda_{\min}$, $\lambda_T = \lambda_{\max}$

$\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

for $t = 1, \ldots, T$ **do**
- ▷ Form the classifier-free guided score at log SNR λ_t
- $\tilde{\boldsymbol{\epsilon}}_t = (1 + w)\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, \mathbf{c}) - w\boldsymbol{\epsilon}_\theta(\mathbf{z}_t)$
- ▷ Sampling step (could be replaced by another sampler, e.g. DDIM)
- $\tilde{\mathbf{x}}_t = (\mathbf{z}_t - \sigma_{\lambda_t} \tilde{\boldsymbol{\epsilon}}_t)/\alpha_{\lambda_t}$
- $\mathbf{z}_{t+1} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_{\lambda_{t+1}|\lambda_t}(\mathbf{z}_t, \tilde{\mathbf{x}}_t), (\tilde{\sigma}^2_{\lambda_{t+1}|\lambda_t})^{1-v}(\sigma^2_{\lambda_t|\lambda_{t+1}})^v)$ if $t < T$ else $\mathbf{z}_{t+1} = \tilde{\mathbf{x}}_t$

end for

return $\mathbf{z}_{T+1}$

Image credit: [11]

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

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

Image credit: [4]

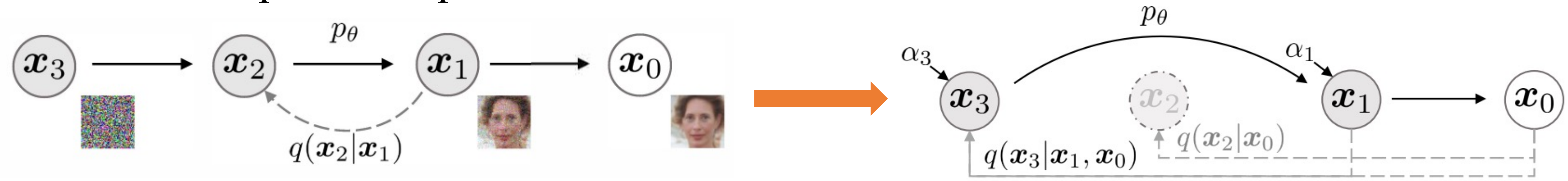
Inference: DDIM

- Let's skip some steps in the middle!

Image credit: [12]

Inference: DDIM

- Let's skip some steps in the middle!



- Recall the data prediction formulation:

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

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

Image credit: [12]

Inference: DDIM

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

Algorithm 1: Fewer-Steps DDIM Sampling

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

$$[1, \ldots, T] \Longrightarrow [\tau_0 = 0, \ldots, \tau_S = T], \text{e.g.}, \tau = [0, 10, 20, 30, \ldots, 1000]$$

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 )
```

Image credit: [13]

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]:

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

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

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

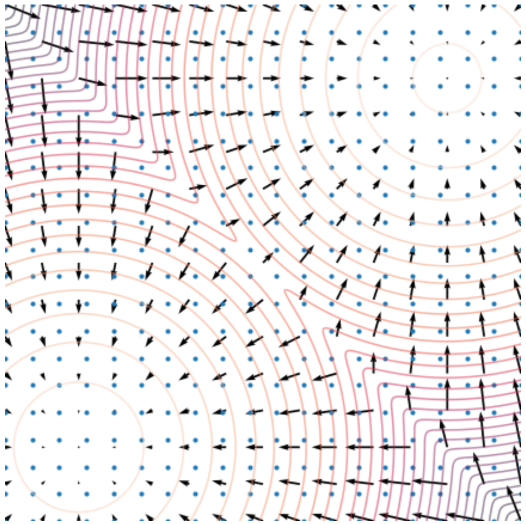
Ref: [10]

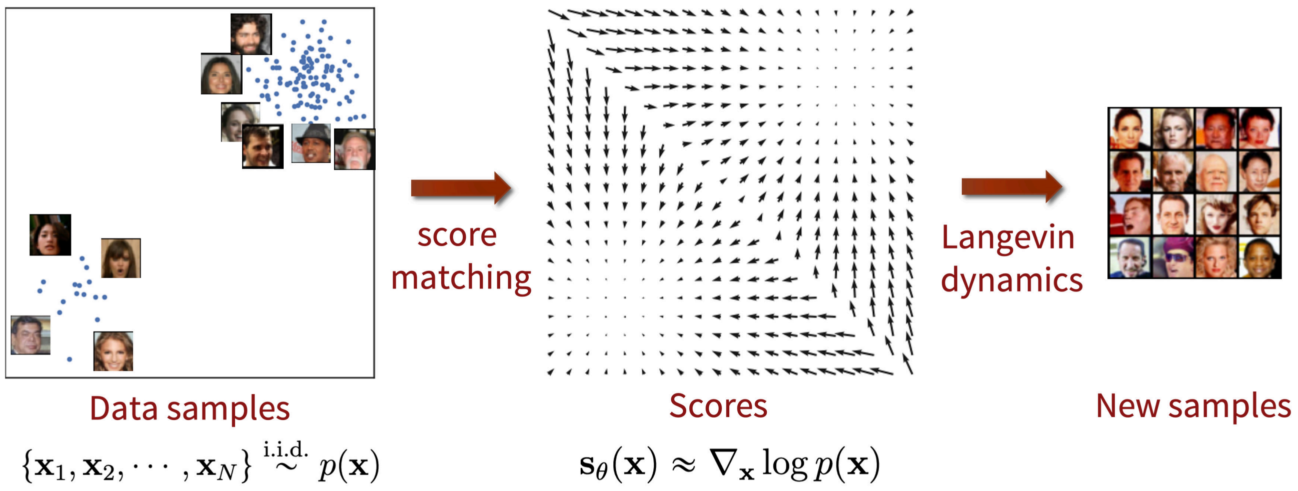
Score Matching & Score-based Models

- Score matching objective:

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

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





Ref: [17, 18]

Score Matching & Score-based Models

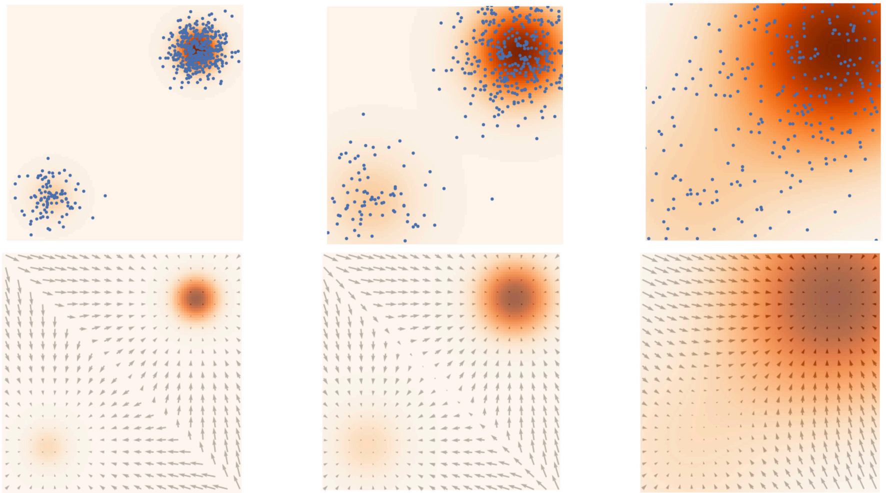
- What if we train score-based models from scratch (using DSM)?

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

$$\nabla_{\tilde{\mathbf{x}}} \log q_{\sigma_i}(\tilde{\mathbf{x}}) = \mathbb{E}_{q(\mathbf{x}_0)q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}_0)}\left[\nabla_{\tilde{\mathbf{x}}} \log q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}_0)\right]$$
$$\approx -\boldsymbol{s}_\theta(\tilde{\mathbf{x}}, \sigma_i)$$

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

$$\sigma_1 \quad < \quad \sigma_2 \quad < \quad \sigma_3$$



Ref: [17, 18]

Score Matching & Score-based Models

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

$$
\begin{aligned}
q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}_0) &\sim \mathcal{N}(\mathbf{x}_0, \sigma_i^2 \mathbf{I}) \\
\nabla_{\tilde{\mathbf{x}}} \log q_{\sigma_i}(\tilde{\mathbf{x}}) &= \mathbb{E}_{q(\mathbf{x}_0) q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}_0)} \left[\nabla_{\tilde{\mathbf{x}}} \log q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}_0) \right] \\
&\approx -\boldsymbol{s}_\theta(\tilde{\mathbf{x}}, \sigma_i) \\
\text{where } \sigma_{\min} &= \sigma_1 < \sigma_2 < \cdots < \sigma_N = \sigma_{\max}
\end{aligned}
$$

- Training:

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

Ref: [17, 18]

Score Matching & Score-based Models

- Sampling:

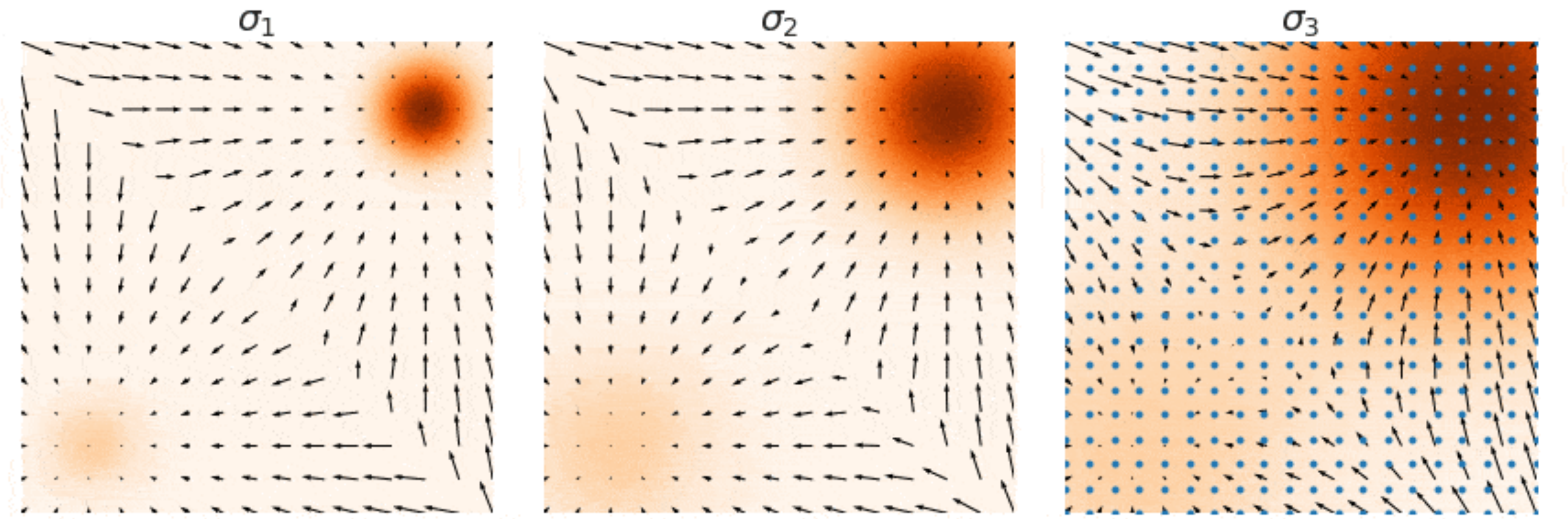
Algorithm 1 Langevin MCMC Sampling from Score-Based Model

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

Ref: [17, 18]

Score Matching & Score-based Models

- Sampling:



Ref: [17, 18]

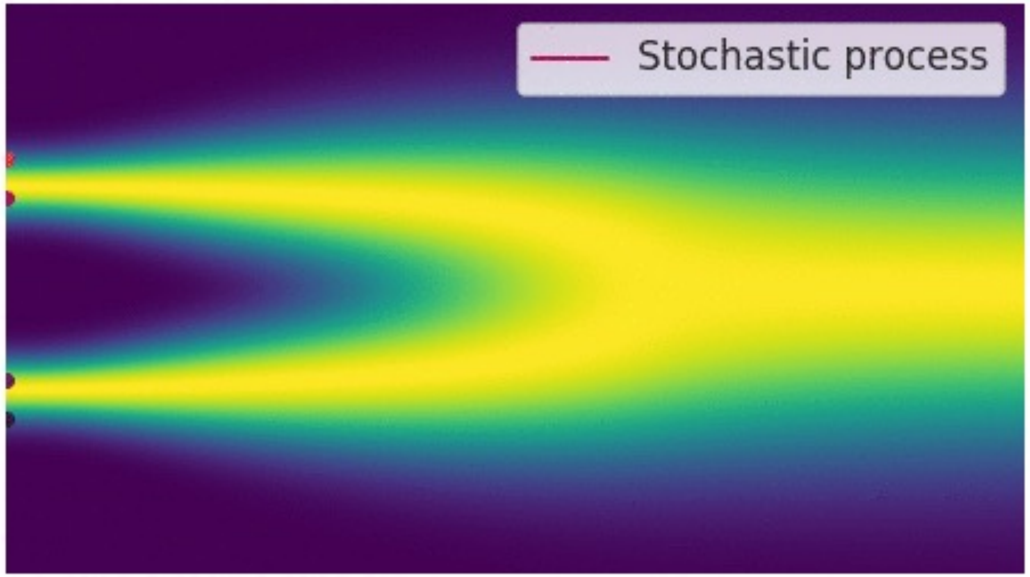
Outline

- 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?

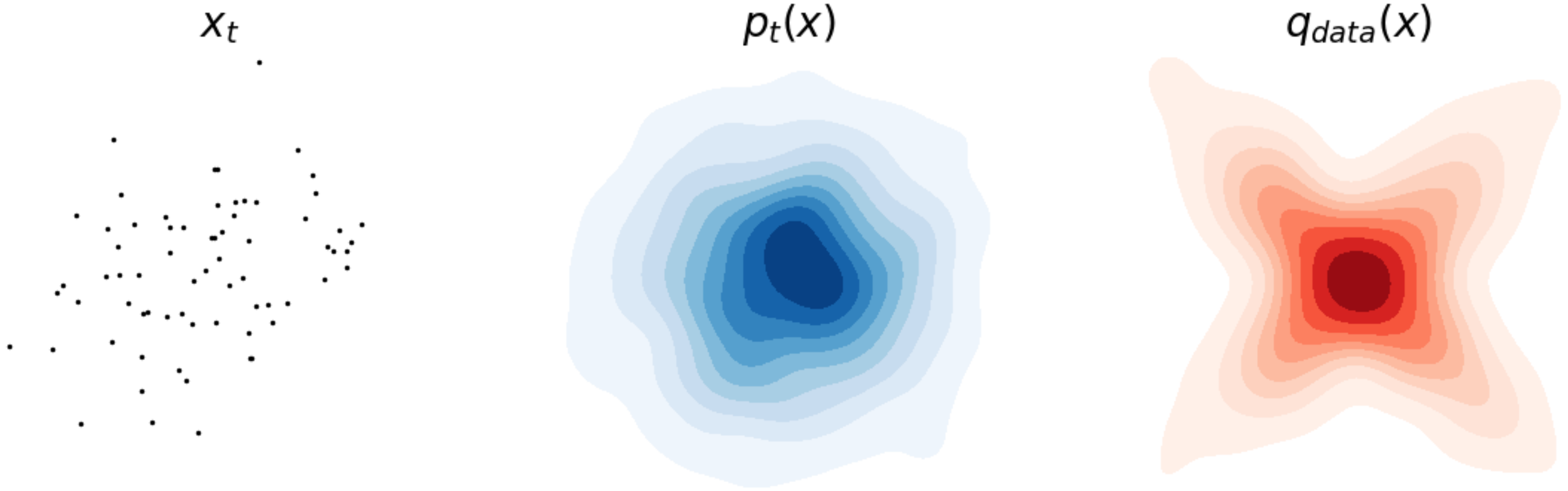
$$q_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}_0) \sim \mathcal{N}(\mathbf{x}_0, {\sigma_i}^2\mathbf{I})$$



Ref: [17, 18, 19]

Score SDEs

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



Ref: [17, 18, 19]

Score SDEs

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

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

- 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

Ref: [17, 18, 19]

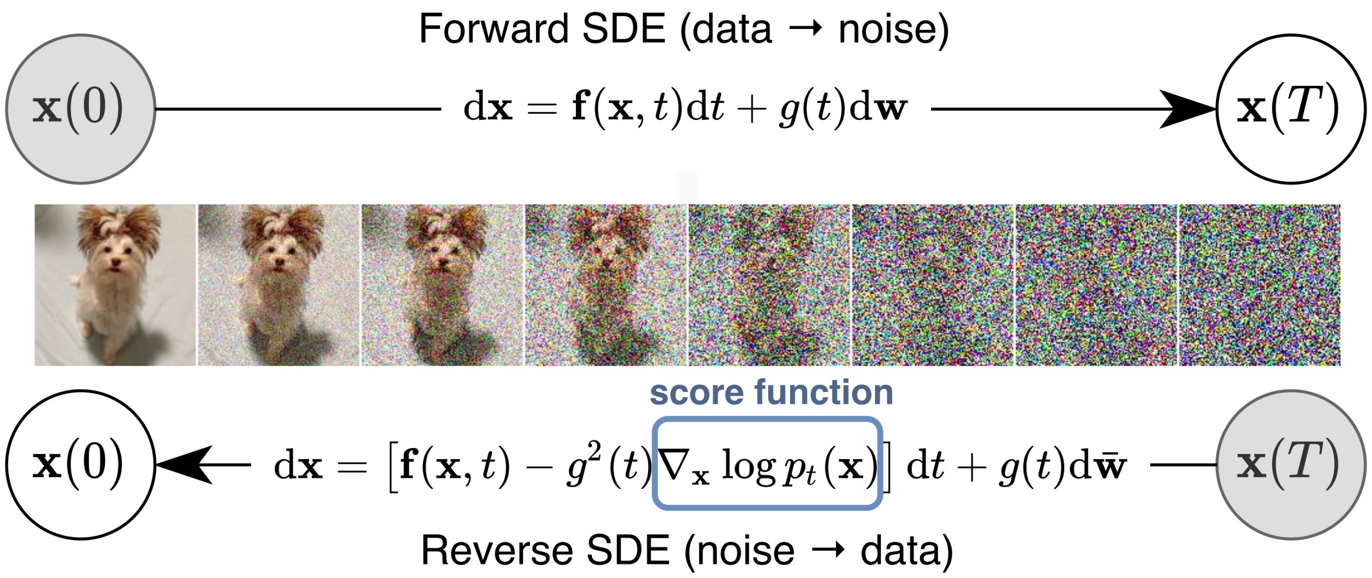
Score SDEs

- 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:

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



Ref: [17, 18, 19]

Score SDEs

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

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

- Reverse-time SDE corresponds to the sampling process:

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

- Training (based on forward SDEs):

$$\mathbb{E}_{t\in\mathcal{U}(0,T)}\mathbb{E}_{p_t(\mathbf{x})}[\lambda(t)\|\underbrace{\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, t)}}_{\text{Denoising score matching (or other variants)}}\|_2^2]$$

Ref: [17, 18, 19]

Score SDEs

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

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

- Reverse-time SDE corresponds to the sampling process:

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

- 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:

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

Ref: [17, 18, 19]

Score SDEs

- Sampling (based on reverse SDEs):

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

$$\Delta\mathbf{x} \leftarrow [\mathbf{f}(\mathbf{x}, t) - g^2(t)\mathbf{s}_\theta(\mathbf{x}, t)]\Delta t + g(t)\sqrt{|\Delta t|}\mathbf{z}_t$$
$$\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}, t \leftarrow t + \Delta t, \mathbf{z}_t \sim \mathcal{N}(0, I)$$

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

Ref: [17, 18, 19]

Score SDEs

- Sampling (based on reverse SDEs):

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

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

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

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

Ref: [17, 18, 19]

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. InProceedings 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?