CS-E4891 Deep Generative Models
Lecture 4: Diffusion models

Harri Lähdesmäki

Department of Computer Science
Aalto University

April 14, 2025

Outline

- Denoising diffusion probabilistic model (Section 25.2 from (Murphy, 2023))
- Continuous-time diffusion models (Section 25.4 from (Murphy, 2023))
- Latent diffusion (Section 25.5 from (Murphy, 2023))
- Conditional diffusion models (Section 25.6 from (Murphy, 2023))

- Reading: parts of Sec. 25 from (Murphy, 2023), Sec. 18 from (Prince, 2023), and other literature (see references at the end)

Denoising diffusion probabilistic model

Diffusion models build on a simple idea:

- It is hard to convert pure noise to structured data
- It is easy to convert structured data into a noise

Denoising diffusion probabilistic model

Diffusion models build on a simple idea:

- It is hard to convert pure noise to structured data
- It is easy to convert structured data into a noise

Denoising diffusion probabilistic model (DDPM) can be thought of as a hierarchical VAE, where

- All latent layers $\boldsymbol{x}_{1:T} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ have the same dimensionality as data $\boldsymbol{x}_0$
- Encoder model (variational inference) is a simple and fixed linear Gaussian model
- Decoder model (generative model) is shared across layers

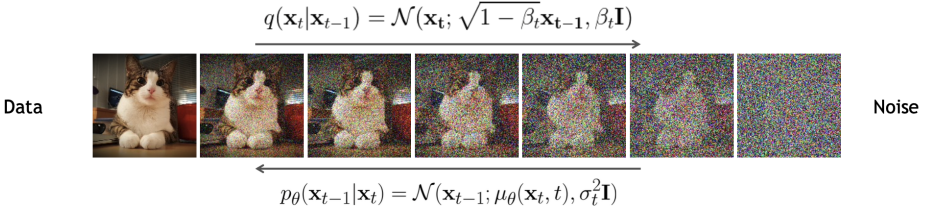


Illustration: Figure 25.1 from (Murphy, 2023)

DDPM: forward process

The forward diffusion process (=encoder) is a simple Gaussian noise model

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

where $\beta_t \in (0, 1)$ is defined by a noise schedule $\beta_1, \beta_2, \ldots, \beta_T$

The joint distribution over latent variables is

$$q(\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$$

DDPM: forward process

The forward diffusion process (=encoder) is a simple Gaussian noise model

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

where $\beta_t \in (0, 1)$ is defined by a noise schedule $\beta_1, \beta_2, \ldots, \beta_T$

The joint distribution over latent variables is

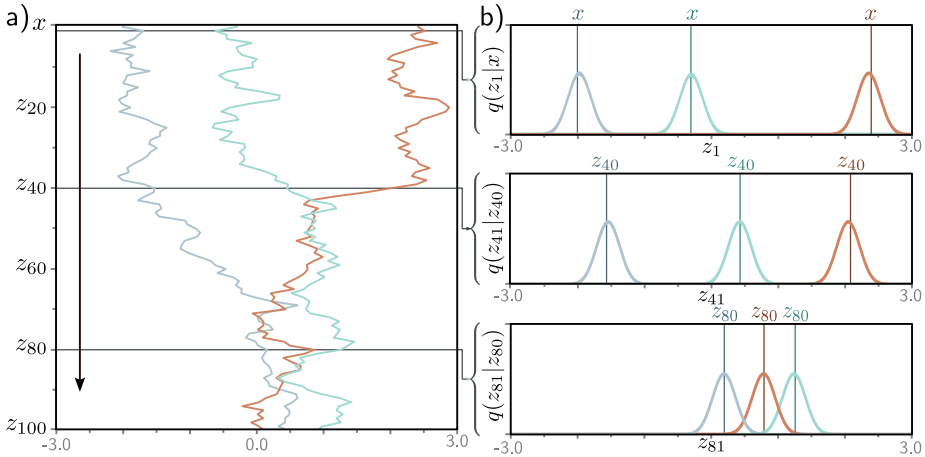
$$q(\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$$

This is a linear Markov chain for which marginals can be computed analytically (see (Prince, 2023) for details)

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t \mid \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I}), \quad \text{where} \quad \alpha_t = 1 - \beta_t \quad \text{and} \quad \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

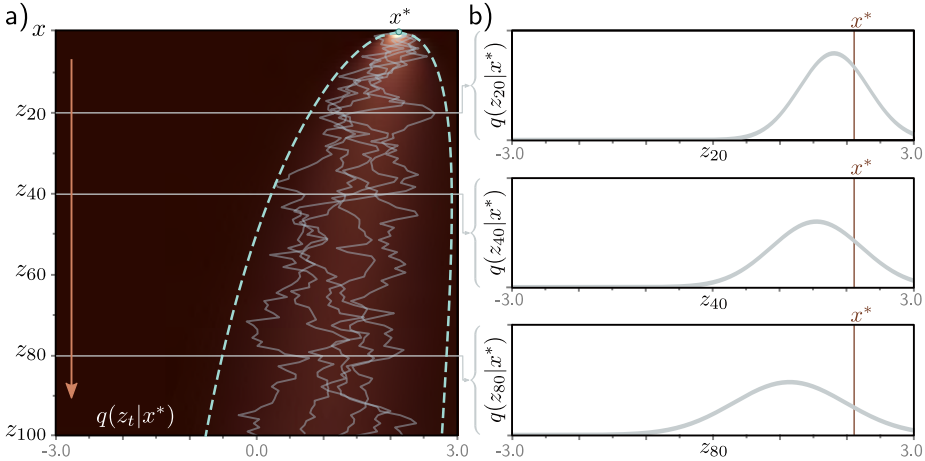
Choose noise schedule such that $\bar{\alpha}_T \approx 0$ and $q(\boldsymbol{x}_T \mid \boldsymbol{x}_0) \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$

DDPM: forward process



1-D Illustration of $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$ with $\beta_t = 0.03$. Figure 18.2 from (Prince, 2023)

DDPM: forward process



1-D Illustration of $q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)$. Figure 18.3 from (Prince, 2023)

DDPM: reverse process

To generate data, we would ideally like to directly reverse the forward process $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$

$$q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \frac{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) q(\boldsymbol{x}_{t-1})}{q(\boldsymbol{x}_t)}$$

DDPM: reverse process

To generate data, we would ideally like to directly reverse the forward process $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$

$$q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \frac{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) q(\boldsymbol{x}_{t-1})}{q(\boldsymbol{x}_t)}$$

but we cannot analytically solve $q(\boldsymbol{x}_{t-1})$ or $q(\boldsymbol{x}_t)$ because

$$q(\boldsymbol{x}_t) = \int q(\boldsymbol{x}_t \mid \boldsymbol{x}) p_{\mathcal{D}}(\boldsymbol{x}) d\boldsymbol{x}$$

DDPM: reverse process

To generate data, we would ideally like to directly reverse the forward process $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$

$$q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \frac{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})q(\boldsymbol{x}_{t-1})}{q(\boldsymbol{x}_t)}$$

but we cannot analytically solve $q(\boldsymbol{x}_{t-1})$ or $q(\boldsymbol{x}_t)$ because

$$q(\boldsymbol{x}_t) = \int q(\boldsymbol{x}_t \mid \boldsymbol{x}) p_{\mathcal{D}}(\boldsymbol{x}) d\boldsymbol{x}$$

If the input $\boldsymbol{x}_0$ was known, using properties of Gaussians (see e.g. (Prince, 2023) for details)

$$\begin{aligned} q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) &= \mathcal{N}(\boldsymbol{x}_{t-1} \mid \tilde{\mu}_t(\boldsymbol{x}_t, \boldsymbol{x}_0), \tilde{\beta}_t \mathbf{I}) \\ \tilde{\mu}_t(\boldsymbol{x}_t, \boldsymbol{x}_0) &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t \\ \tilde{\beta}_t &= \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \end{aligned}$$

This does not serve us because $\boldsymbol{x}_0$ is not known when generating new data

DDPM: learnable reverse process

We can learn a reverse diffusion process (=decoder) that aims to invert the forward process

Generator can be chosen to have the following form

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1} \mid \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)), \quad \text{where often} \quad \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \sigma_t^2 \mathbf{I}$$

where $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ is trainable neural net

DDPM: learnable reverse process

We can learn a reverse diffusion process (=decoder) that aims to invert the forward process

Generator can be chosen to have the following form

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1} \mid \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)), \quad \text{where often} \quad \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \sigma_t^2 \mathbf{I}$$

where $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ is trainable neural net

The joint distribution of generated variables is

$$\begin{aligned} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{0:T}) &= p(\boldsymbol{x}_T) p_{\boldsymbol{\theta}}(\boldsymbol{x}_{T-1} \mid \boldsymbol{x}_T) \cdots p_{\boldsymbol{\theta}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_1) \\ &= p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) \end{aligned}$$

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

DDPM model components

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

Data

Noise

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$$

Illustration: Figure 25.1 from (Murphy, 2023)

DDPM: model fitting

We can fit the model using ELBO

$$
\begin{aligned}
\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0) &= \log \left[\int p_{\boldsymbol{\theta}}(\boldsymbol{x}_0, \boldsymbol{x}_{1:T}) d\boldsymbol{x}_{1:T} \right] \\
&= \log \left[\int \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)} q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0) d\boldsymbol{x}_{1:T} \right] \\
&= \log \mathbb{E}_{q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)} \left[\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)} \right] \\
&\overset{\text{Jensen}}{\geq} \mathbb{E}_q \left[\log \left(\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)} \right) \right] \\
&= \mathbb{E}_q \left[\log \left(p(\boldsymbol{x}_T) \prod_{t=1}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})} \right) \right] \\
&= \mathbb{E}_q \left[\log p(\boldsymbol{x}_T) + \sum_{t=1}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})} \right] \\
&= \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{x}_0)
\end{aligned}
$$

DDPM: model fitting details

By the Markov property and Bayes's rule we have

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0)}$$

DDPM: model fitting details

By the Markov property and Bayes's rule we have

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0)}$$

By plugging $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$ into $\mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{x}_0)$ we get

$$\mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{x}_0) = \mathbb{E}_{q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)} \left[\log p(\boldsymbol{x}_T) + \sum_{t=2}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0)} + \underbrace{\sum_{t=2}^{T} \log \frac{q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)}}_{\text{* telescoping sum}} + \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)}{q(\boldsymbol{x}_1 \mid \boldsymbol{x}_0)} \right],$$

DDPM: model fitting details

By the Markov property and Bayes's rule we have

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0)}$$

By plugging $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$ into $\mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{x}_0)$ we get

$$\mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{x}_0) = \mathbb{E}_{q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)} \left[\log p(\boldsymbol{x}_T) + \sum_{t=2}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0)} + \underbrace{\sum_{t=2}^{T} \log \frac{q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)}}_{* \text{ telescoping sum}} + \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)}{q(\boldsymbol{x}_1 \mid \boldsymbol{x}_0)} \right],$$

where telescoping sum reduces to

$$\begin{aligned} * &= \Big(\log q(\boldsymbol{x}_1 \mid \boldsymbol{x}_0) - \log q(\boldsymbol{x}_2 \mid \boldsymbol{x}_0) \Big) + \Big(\log q(\boldsymbol{x}_2 \mid \boldsymbol{x}_0) - \log q(\boldsymbol{x}_3 \mid \boldsymbol{x}_0) \Big) + \dots \\ &\quad + \Big(\log q(\boldsymbol{x}_{T-1} \mid \boldsymbol{x}_0) - \log q(\boldsymbol{x}_T \mid \boldsymbol{x}_0) \Big) \\ &= \log q(\boldsymbol{x}_1 \mid \boldsymbol{x}_0) - \log q(\boldsymbol{x}_T \mid \boldsymbol{x}_0) \end{aligned}$$

DDPM: model fitting details

ELBO reduces to

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{x}_0) &= \mathbb{E}_{q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)} \left[\log p(\boldsymbol{x}_T) + \sum_{t=2}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0)} + \log \frac{q(\boldsymbol{x}_1 \mid \boldsymbol{x}_0)}{q(\boldsymbol{x}_T \mid \boldsymbol{x}_0)} + \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)}{q(\boldsymbol{x}_1 \mid \boldsymbol{x}_0)} \right] \\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)} \left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T \mid \boldsymbol{x}_0)} + \sum_{t=2}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0)} + \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_1) \right] \\
&= \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1 \mid \boldsymbol{x}_0)} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_1) \right]}_{L_0(\boldsymbol{x}_0)} - \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)} \underbrace{\left[D_{\mathbb{KL}}(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \mid\mid p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)) \right]}_{L_{t-1}(\boldsymbol{x}_0)} \\
&\quad + \underbrace{D_{\mathbb{KL}}(q(\boldsymbol{x}_T \mid \boldsymbol{x}_0) \mid\mid p(\boldsymbol{x}_T))}_{L_T(\boldsymbol{x}_0)}
\end{aligned}$$

All KL divergences can be computed analytically because all distributions are Gaussians

DDPM: model fitting details for $L_{t-1}(\boldsymbol{x}_0)$

Recall that the KL between two multivariate Gaussians is:

$$D_{\mathbb{KL}}\left(\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \,||\, \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)\right) = \frac{1}{2}\Big[\mathrm{tr}(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) - D + \log\left(\frac{\det(\boldsymbol{\Sigma}_2)}{\det(\boldsymbol{\Sigma}_1)}\right)\Big]$$

DDPM: model fitting details for $L_{t-1}(\boldsymbol{x}_0)$

Recall that the KL between two multivariate Gaussians is:

$$D_{\mathbb{KL}}\left(\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \,||\, \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)\right) = \frac{1}{2}\Big[\mathrm{tr}(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) - D + \log\left(\frac{\det(\boldsymbol{\Sigma}_2)}{\det(\boldsymbol{\Sigma}_1)}\right)\Big]$$

Thus, $L_{t-1}(\boldsymbol{x}_0)$ term can be written as

$$\begin{aligned} D_{\mathbb{KL}}(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)) &= \frac{1}{2\sigma_t^2}\|\tilde{\mu}_t(\boldsymbol{x}_t, \boldsymbol{x}_0) - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\|^2 + C \\ &= \frac{1}{2\sigma_t^2}\Big\| \underbrace{\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t}_{\mathbb{E}[\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0, \boldsymbol{x}_t]} - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\Big\|^2 + C \end{aligned}$$

DDPM: model fitting details for $L_{t-1}(\boldsymbol{x}_0)$

Recall that the KL between two multivariate Gaussians is:

$$D_{\mathbb{KL}}\left(\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \,||\, \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)\right) = \frac{1}{2}\Big[\operatorname{tr}(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) - D + \log\left(\frac{\det(\boldsymbol{\Sigma}_2)}{\det(\boldsymbol{\Sigma}_1)}\right)\Big]$$

Thus, $L_{t-1}(\boldsymbol{x}_0)$ term can be written as

$$\begin{aligned} D_{\mathbb{KL}}(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)) &= \frac{1}{2\sigma_t^2}\|\tilde{\mu}_t(\boldsymbol{x}_t, \boldsymbol{x}_0) - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\|^2 + C \\ &= \frac{1}{2\sigma_t^2}\Big\| \underbrace{\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t}_{\mathbb{E}[\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0, \boldsymbol{x}_t]} - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\Big\|^2 + C \end{aligned}$$

Expectations can be estimated using Monte Carlo

$$\mathbb{E}[L_{t-1}(\boldsymbol{x}_0)] \approx \frac{1}{S}\frac{1}{\cdot 2\sigma_t^2}\sum_{s=1}^{S}\left\| \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t^{(s)} - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right\|^2, \quad \text{where} \quad \boldsymbol{x}_t^{(s)} \overset{\text{i.i.d}}{\sim} q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)$$

DDPM: reparameterization

Earlier we had that

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

which means

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

DDPM: reparameterization

Earlier we had that

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

which means

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

Insert the expression of $\boldsymbol{x}_0$ into the exact decoder model $q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_{t-1} \mid \tilde{\mu}_t(\boldsymbol{x}_t, \boldsymbol{x}_0), \tilde{\beta}_t\mathbf{I})$ to get

$$\begin{aligned}
\tilde{\mu}_t(\boldsymbol{x}_t, \boldsymbol{x}_0) &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t \\
&= \dots \\
&= \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right)
\end{aligned}$$

This reparameterized model depends explicitly on $\boldsymbol{x}_t$ and $\boldsymbol{\epsilon}$ (and only implicitly on $\boldsymbol{x}_0$)

DDPM: reparameterization

Previously we had a decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1} \mid \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t))$ that predicts the mean of the denoised version of $\boldsymbol{x}_{t-1}$ given its noisy input $\boldsymbol{x}_t$

Reparametrization allows to train a neural network model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ that, given $\boldsymbol{x}_t$, predicts the specific noise $\boldsymbol{\epsilon}$ (sampled from $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$) that was added to $\boldsymbol{x}_{t-1}$ such that it resulted in $\boldsymbol{x}_t$

DDPM: reparameterization

Previously we had a decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1} \mid \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t))$ that predicts the mean of the denoised version of $\boldsymbol{x}_{t-1}$ given its noisy input $\boldsymbol{x}_t$

Reparametrization allows to train a neural network model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ that, given $\boldsymbol{x}_t$, predicts the specific noise $\boldsymbol{\epsilon}$ (sampled from $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$) that was added to $\boldsymbol{x}_{t-1}$ such that it resulted in $\boldsymbol{x}_t$

In other words, given noisy input $\boldsymbol{x}_t$, predict $\boldsymbol{\epsilon}$ to get mean of $\boldsymbol{x}_{t-1}$

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

DDPM: reparameterized objective

The reparameterized loss

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

DDPM: reparameterized objective

The reparameterized loss

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

Empirical evidence suggests that $\lambda_t = 1$ gives better results

A simplified loss (averaged over samples $\boldsymbol{x}_0$ and diffusion layers t)

$$L_{\text{simple}} = \mathbb{E}_{\boldsymbol{x}_0\sim p_{\mathcal{D}}(\boldsymbol{x}),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),t\sim\text{Unif}(\{1,\ldots,T\})}\left[\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\underbrace{\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}}_{\boldsymbol{x}_t},t)\|^2\right]$$

DDPM: training algorithm, simple model

Algorithm 25.1: Training a DDPM model with L_{simple}.

while *not converged* **do**
  $\boldsymbol{x}_0 \sim q_0(\boldsymbol{x}_0)$
  $t \sim \text{Unif}(\{1, \ldots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on $\nabla_{\boldsymbol{\theta}} ||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}} \left(\sqrt{\overline{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t\right)||^2$

Algorithm 25.1 from (Murphy, 2023)

DDPM: ancestral sampling algorithm

Algorithm 25.2: Sampling from a DDPM model.

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

Algorithm 25.2 from (Murphy, 2023)

UNet architecture

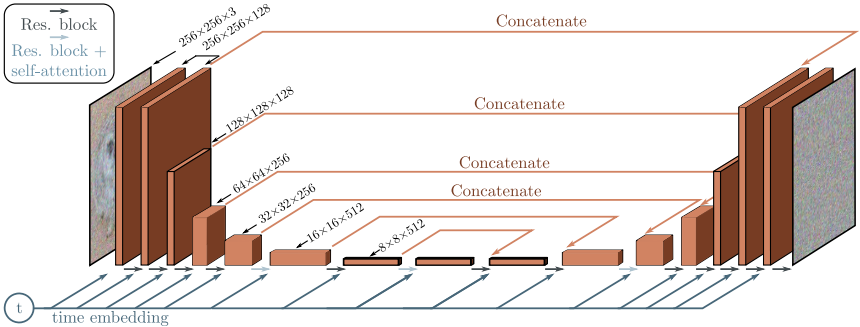


Figure 18.9 from (Prince, 2023)

DDPM: illustration

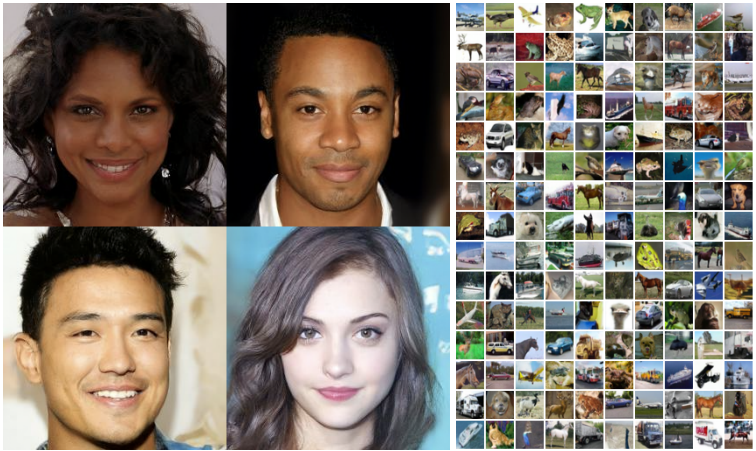
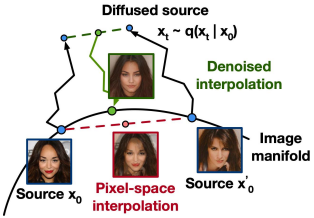


Figure from (Ho et al., 2020)

DDPM: interpolation

Example: DDPM model can be used e.g. to interpolate two data points $\boldsymbol{x}$ and $\boldsymbol{x}'$

- $\boldsymbol{x}_t \sim q(\boldsymbol{x}_t \mid \boldsymbol{x})$ and $\boldsymbol{x}'_t \sim q(\boldsymbol{x}_t \mid \boldsymbol{x}')$
- $\bar{\boldsymbol{x}}_t = (1-\lambda)\boldsymbol{x}_t + \lambda \boldsymbol{x}'_t$
- $\boldsymbol{x}_0 \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}_0 \mid \bar{\boldsymbol{x}}_t)$



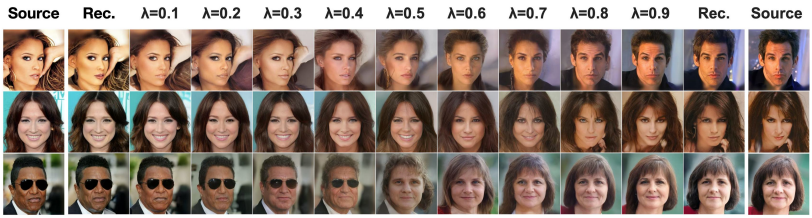


Figure from (Ho et al., 2020)

DDPM: noise schedule

The original DDPM model chose the noise schedule $\beta_1, \ldots, \beta_T$ from a set of constant, linear, and quadratic schedules, such that $q(\boldsymbol{x}_T \mid \boldsymbol{x}_0) \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$, with $T = 1000$

A number of noise schedules have been proposed in the literature both with

- empirical evidence
- theoretical benefits

Continuous-time diffusion models: forward diffusion SDE

We can consider $t \in [0, T]$ to be continuous to obtain continuous-time diffusion models

Consider noise level β_t to be $\beta(t)\Delta t$, where Δt is a small diffusion time increment

Diffusion model $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t \mid \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \mathbf{I})$ becomes

$$\boldsymbol{x}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon} = \sqrt{1-\beta(t)\Delta t} \cdot \boldsymbol{x}_{t-1} + \sqrt{\beta(t)\Delta t} \cdot \boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Continuous-time diffusion models: forward diffusion SDE

We can consider $t \in [0, T]$ to be continuous to obtain continuous-time diffusion models

Consider noise level β_t to be $\beta(t)\Delta t$, where Δt is a small diffusion time increment

Diffusion model $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t \mid \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \mathbf{I})$ becomes

$$\boldsymbol{x}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon} = \sqrt{1-\beta(t)\Delta t} \cdot \boldsymbol{x}_{t-1} + \sqrt{\beta(t)\Delta t} \cdot \boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

For small Δt, by the first order Tailor approximation

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} - \frac{\beta(t)\Delta t}{2} \cdot \boldsymbol{x}_{t-1} + \sqrt{\beta(t)\Delta t} \cdot \boldsymbol{\epsilon}$$

Continuous-time diffusion models: forward diffusion SDE

We can consider $t \in [0, T]$ to be continuous to obtain continuous-time diffusion models

Consider noise level β_t to be $\beta(t)\Delta t$, where Δt is a small diffusion time increment

Diffusion model $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t \mid \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \mathbf{I})$ becomes

$$\boldsymbol{x}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon} = \sqrt{1-\beta(t)\Delta t} \cdot \boldsymbol{x}_{t-1} + \sqrt{\beta(t)\Delta t} \cdot \boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

For small Δt, by the first order Tailor approximation

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} - \frac{\beta(t)\Delta t}{2} \cdot \boldsymbol{x}_{t-1} + \sqrt{\beta(t)\Delta t} \cdot \boldsymbol{\epsilon}$$

Thus

$$\frac{\boldsymbol{x}_t - \boldsymbol{x}_{t-1}}{\Delta t} = -\frac{\beta(t)}{2}\boldsymbol{x}_{t-1} + \frac{\sqrt{\beta(t)}}{\sqrt{\Delta t}}\boldsymbol{\epsilon}$$

and when $\Delta t \to 0$

$$\frac{d\boldsymbol{x}(t)}{dt} = -\frac{\beta(t)}{2}\boldsymbol{x}(t) + \sqrt{\beta(t)}\frac{d\boldsymbol{w}(t)}{dt}$$

we have a stochastic differential equation (SDE) model, where $\boldsymbol{w}$ is Brownian motion

Continuous-time diffusion models: forward diffusion SDE

SDE models can be generally written as

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

and can be numerically simulated (solved) e.g. by the Euler-Maruyama algorithm

$$\boldsymbol{x}(t + \Delta t) = \boldsymbol{x}(t) + \boldsymbol{f}(\boldsymbol{x}(t), t)\Delta t + g(t)\sqrt{\Delta t} \cdot \boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Continuous-time diffusion models: forward diffusion SDE

SDE models can be generally written as

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

and can be numerically simulated (solved) e.g. by the Euler-Maruyama algorithm

$$\boldsymbol{x}(t + \Delta t) = \boldsymbol{x}(t) + \boldsymbol{f}(\boldsymbol{x}(t), t)\Delta t + g(t)\sqrt{\Delta t} \cdot \boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Continuous-time limit of the DDPM model is obtained with $\boldsymbol{f}(\boldsymbol{x}, t) = -\frac{1}{2}\beta(t)\boldsymbol{x}$ and $g(t) = \sqrt{\beta(t)}$

$$d\boldsymbol{x} = -\frac{1}{2}\beta(t)\boldsymbol{x}dt + \sqrt{\beta(t)}d\boldsymbol{w}$$

Continuous-time diffusion models: illustration

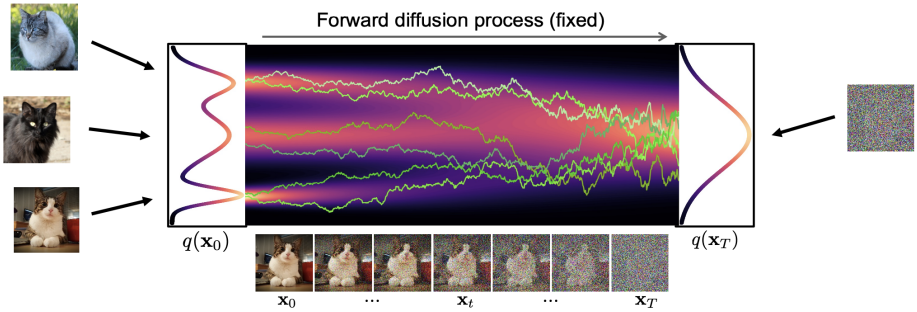


Figure 25.6 from (Murphy, 2023)

Continuous-time diffusion models: reverse diffusion SDE

As with discrete-time diffusion models, we need to revert the noising process

Reverse-time SDE is known and has the following form

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

where $\bar{\boldsymbol{w}}$ is Brownian motion to the reverse diffusion time direction

Continuous-time diffusion models: reverse diffusion SDE

As with discrete-time diffusion models, we need to revert the noising process

Reverse-time SDE is known and has the following form

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

where $\bar{\boldsymbol{w}}$ is Brownian motion to the reverse diffusion time direction

Reverse-time SDE model corresponding to the DDPM model is

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

We need to estimate the so-called score-function

$$\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \approx \nabla_{\boldsymbol{x}_t} \log q_t(\boldsymbol{x}_t)$$

Continuous-time diffusion models: reverse diffusion SDE

Training loss for continuous-time diffusion model

$$\mathcal{L} = \mathbb{E}_{t\sim\mathrm{Unif}([0,T])}[\lambda(t)\mathbb{E}_{\boldsymbol{x}_0}\mathbb{E}_{\boldsymbol{x}_t|\boldsymbol{x}_0}[\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t) - \nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t \mid \boldsymbol{x}_0)\|]]$$

where

- $\lambda(t)$ is a positive weighting function
- $\nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t)$ can be computed analytically for simply (linear) forward diffusion models

Continuous-time diffusion models: reverse diffusion SDE

Training loss for continuous-time diffusion model

$$\mathcal{L} = \mathbb{E}_{t\sim\mathrm{Unif}([0,T])}[\lambda(t)\mathbb{E}_{\boldsymbol{x}_0}\mathbb{E}_{\boldsymbol{x}_t|\boldsymbol{x}_0}[\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t) - \nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t \mid \boldsymbol{x}_0)\|]]$$

where

- $\lambda(t)$ is a positive weighting function
- $\nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t)$ can be computed analytically for simply (linear) forward diffusion models

Samples can be generated using the Euler-Maruyama algorithm

$$\boldsymbol{x}_{t-\Delta t} = \boldsymbol{x}_t + \frac{1}{2}\beta(t)[\boldsymbol{x}_t + 2\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)]\Delta t + \sqrt{\beta(t)\Delta t}\cdot\boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$$

Continuous-time diffusion models: illustration

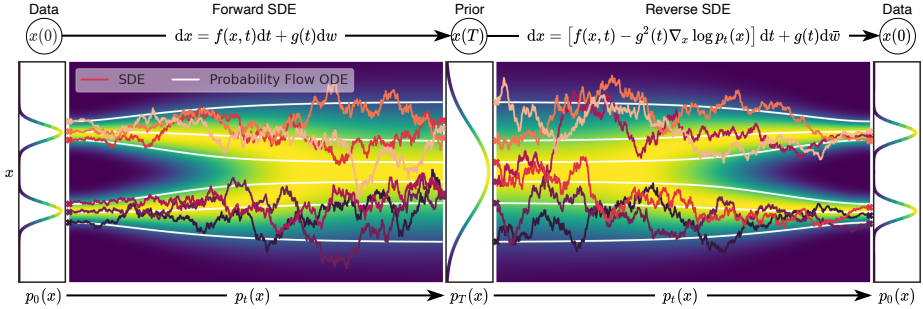


Figure from (Song et al., 2021)

Continuous-time diffusion models: illustration

Figure from (Song et al., 2021)

Latent diffusion models

Diffusion models are slow to train and use

- Model needs to take many small steps on high-dimensional data

Latent diffusion models (LDM) use a two-stage training scheme

- First train a VAE
- Then train a diffusion model to the embeddings

Used e.g. in the stable diffusion system

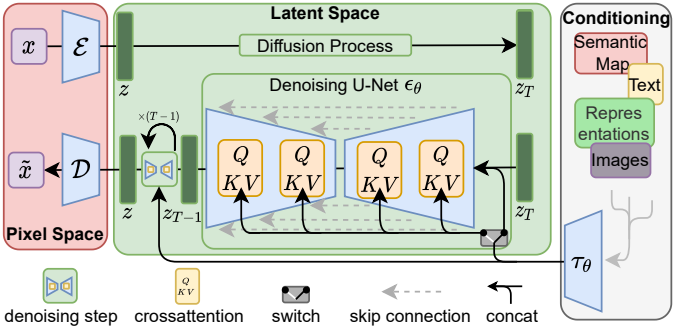


Figure 25.13 from (Murphy, 2023)

Conditional diffusion models: classifier guidance

Consider a case where data points $\boldsymbol{x}$

- Are associated with class labels $\boldsymbol{c}$
- We have access to a pre-trained classifier $p(\boldsymbol{c} \mid \boldsymbol{x})$

From Bayes theorem

$$\log p(\boldsymbol{x} \mid \boldsymbol{c}) = \log p(\boldsymbol{c} \mid \boldsymbol{x}) + \log p(\boldsymbol{x}) - \log p(\boldsymbol{c})$$

Conditional diffusion models: classifier guidance

Consider a case where data points $\boldsymbol{x}$

- Are associated with class labels $\boldsymbol{c}$
- We have access to a pre-trained classifier $p(\boldsymbol{c} \mid \boldsymbol{x})$

From Bayes theorem

$$\log p(\boldsymbol{x} \mid \boldsymbol{c}) = \log p(\boldsymbol{c} \mid \boldsymbol{x}) + \log p(\boldsymbol{x}) - \log p(\boldsymbol{c})$$

The score function becomes

$$\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, t)' = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x} \mid \boldsymbol{c}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{c} \mid \boldsymbol{x}) + \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$$

A weighted score function can be defined as

$$\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, t)' = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x} \mid \boldsymbol{c}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) + w \cdot \nabla_{\boldsymbol{x}} \log p(\boldsymbol{c} \mid \boldsymbol{x})$$

Conditional diffusion models: classifier guidance

Consider a case where data points $\boldsymbol{x}$

- Are associated with class labels $\boldsymbol{c}$
- We have access to a pre-trained classifier $p(\boldsymbol{c} \mid \boldsymbol{x})$

From Bayes theorem

$$\log p(\boldsymbol{x} \mid \boldsymbol{c}) = \log p(\boldsymbol{c} \mid \boldsymbol{x}) + \log p(\boldsymbol{x}) - \log p(\boldsymbol{c})$$

The score function becomes

$$\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, t)' = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x} \mid \boldsymbol{c}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{c} \mid \boldsymbol{x}) + \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$$

A weighted score function can be defined as

$$\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, t)' = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x} \mid \boldsymbol{c}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) + w \cdot \nabla_{\boldsymbol{x}} \log p(\boldsymbol{c} \mid \boldsymbol{x})$$

Analogously, classifier guidance can be implemented in DDPM by modifying the sampling step

$$\boldsymbol{x}_{x-1} = \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \right) + w \cdot \nabla_{\boldsymbol{x}} \log p(\boldsymbol{c} \mid \boldsymbol{x}) + \sigma_t \boldsymbol{\epsilon}_t$$

Classifier guidance affects only sample generation, not diffusion model training

Conditional diffusion models: classifier guidance example

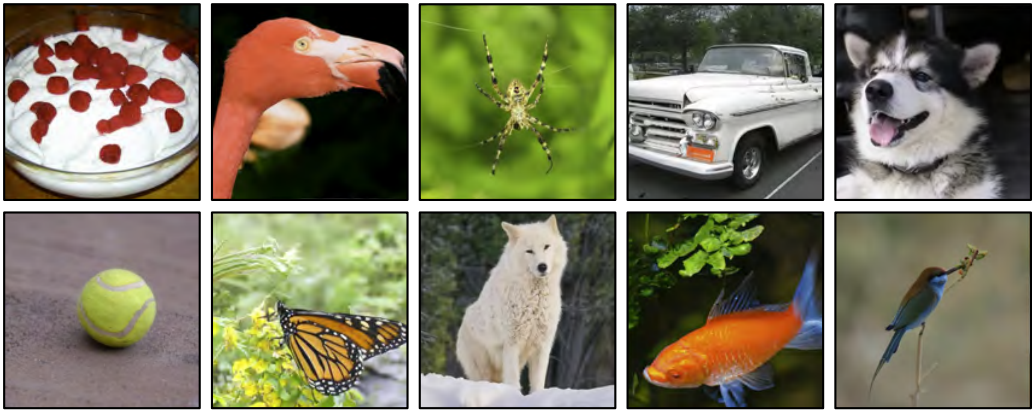


Figure 18.12 from (Prince, 2023)

Conditional diffusion models: classifier-free guidance

Derive a classifier from a generative model $p(\boldsymbol{c} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{c})p(\boldsymbol{c})}{p(\boldsymbol{x})}$ to get

$$\log p(\boldsymbol{c} \mid \boldsymbol{x}) = \log p(\boldsymbol{x} \mid \boldsymbol{c}) + \log p(\boldsymbol{c}) - \log p(\boldsymbol{x})$$

Need two generative models

- Conditional model $p(\boldsymbol{x} \mid \boldsymbol{c})$
- Unconditional model $p(\boldsymbol{x}) = p(\boldsymbol{x} \mid \boldsymbol{c} = \emptyset)$

Conditional diffusion models: classifier-free guidance

Derive a classifier from a generative model $p(\boldsymbol{c} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{c})p(\boldsymbol{c})}{p(\boldsymbol{x})}$ to get

$$\log p(\boldsymbol{c} \mid \boldsymbol{x}) = \log p(\boldsymbol{x} \mid \boldsymbol{c}) + \log p(\boldsymbol{c}) - \log p(\boldsymbol{x})$$

Need two generative models

- Conditional model $p(\boldsymbol{x} \mid \boldsymbol{c})$
- Unconditional model $p(\boldsymbol{x}) = p(\boldsymbol{x} \mid \boldsymbol{c} = \emptyset)$

Use this implicit classifier similarly as in classifier guidance

$$\begin{aligned}\nabla_{\boldsymbol{x}}[\log p(\boldsymbol{x} \mid \boldsymbol{c}) + w \log p(\boldsymbol{c} \mid \boldsymbol{x})] &= \nabla_{\boldsymbol{x}}[\log p(\boldsymbol{x} \mid \boldsymbol{c}) + w(\log p(\boldsymbol{x} \mid \boldsymbol{c}) - p(\boldsymbol{x}))] \\ &= \nabla_{\boldsymbol{x}}[(1+w) \log p(\boldsymbol{x} \mid \boldsymbol{c}) - wp(\boldsymbol{x})]\end{aligned}$$

Conditional diffusion models

Conditional generation by training a diffusion model in $(\boldsymbol{c}, \boldsymbol{x})$ pairs

If the covariate $\boldsymbol{c}$ is a scalar:

- $\boldsymbol{c}$ can be mapped to an embedding vector
- Embedding can be incorporated into the neural network $\boldsymbol{\mu}(\boldsymbol{x}_t, t \mid \boldsymbol{c})$, $\boldsymbol{\epsilon}(\boldsymbol{x}_t, t \mid \boldsymbol{c})$ or $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t \mid \boldsymbol{c})$

If the covariate $\boldsymbol{c}$ is another image:

- We can concatenate $\boldsymbol{c}$ with $\boldsymbol{x}_t$ as an extra set of image channels

If the covariate $\boldsymbol{c}$ is a text prompt:

- We can embed $\boldsymbol{c}$
- Embedding can be incorporated into the neural network $\boldsymbol{\mu}(\boldsymbol{x}_t, t \mid \boldsymbol{c})$, $\boldsymbol{\epsilon}(\boldsymbol{x}_t, t \mid \boldsymbol{c})$ or $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t \mid \boldsymbol{c})$

UNet architecture

Embedding of covariate $\boldsymbol{c}$ can be e.g. shared across UNet layers similarly as time embedding

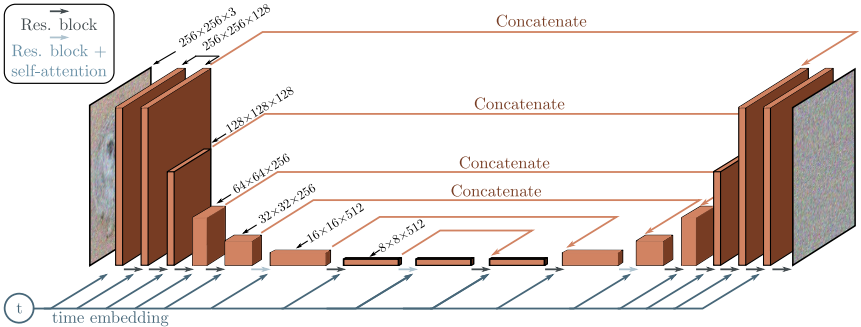


Figure 18.9 from (Prince, 2023)

Conditional diffusion models: example

Figure 18.13 from (Prince, 2023)

References

- Ho J, Jain A, Abbeel P, "Denoising Diffusion Probabilistic Models," *NeurIPS*, 2020.
- Murphy K, Probabilistic Machine Learning: Advanced Topics, 2023.
- Prince SJD, Understanding Deep Learning, The MIT Press, 2023.
- Song Y, Sohl-Dickstein J, Kingma DP, Kumar A, Ermon S, Poole B, "Score-Based Generative Modeling through Stochastic Differential Equations," *ICLR*, 2021.