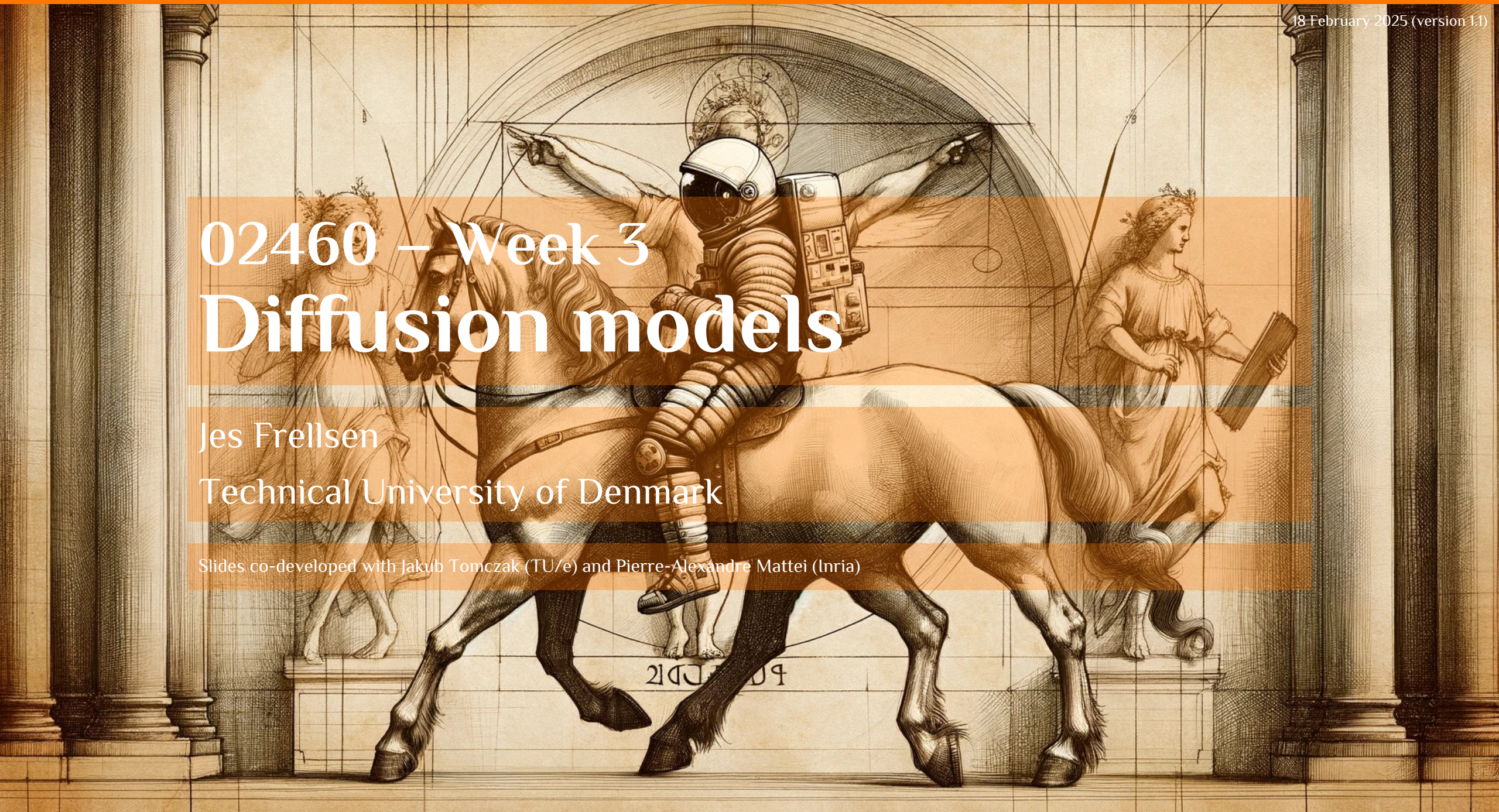
18 February 2025 (version 1.1)



02460 – Week 3
Diffusion models

Jes Frellsen

Technical University of Denmark

Slides co-developed with Jakub Tomczak (TU/e) and Pierre-Alexandre Mattei (Inria)

Menu of the day

Module 1: Generative models

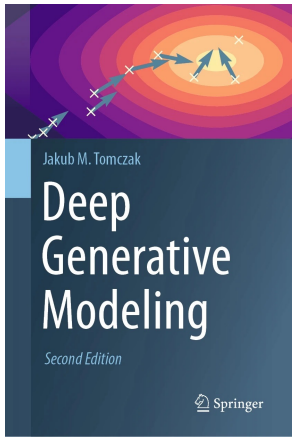
- Week 1: DLVMs
- Week 2: Flows
- Week 3: Diffusion models
- Week 4: Mini-project 1

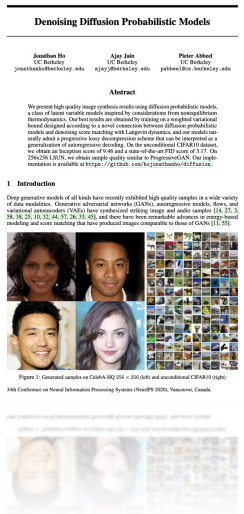
Lecture

- DDPMs
  - Section 5.5.3
  - The DDPM paper[1]
- SDE based models
  - Sections 9.1–9.3

Exercises

- **Theoretical**: Derivation of results for DDPMs (e.g., ELBO)
- **Programming**: Implement the ELBO and sampling for DPPMs





[1] Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. NeurIPS, 2020.

Lecture outline

- Introduction to diffusion models
- Denoising Diffusion Probabilistic Models (DDPMs)
  - **Learning and sampling**
- SDE-based diffusion models
  - **Learning** using score matching
  - Sampling using Euler's method
- Mini-project 1

Diffusion models

- Still an **emerging field**
- Covers several types of models:
  - Score based models
  - Discrete time diffusion models
    - Easy to understand
    - Work very well
  - Continuous time diffusion models
- Are **state-of-the-art** for image generation!

Images from: Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. NeurIPS, 2020.

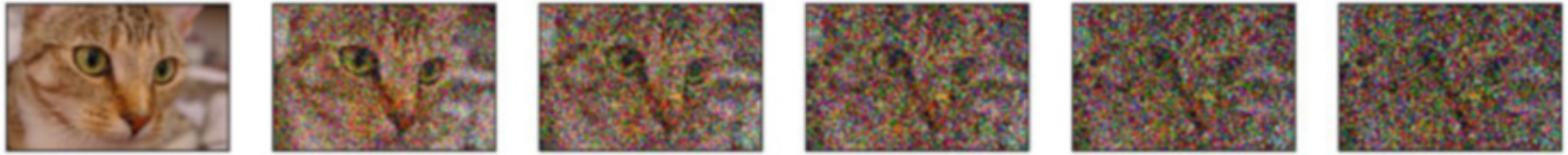
General idea

Why is this a sensible thing to do?

Recall: all the generative models we have seen so far work by transforming noise into data

- We gradually **transform data into noise**
- Can we learn the **reverse process** (i.e., denoising)?

Forward process (Gaussian diffusion)



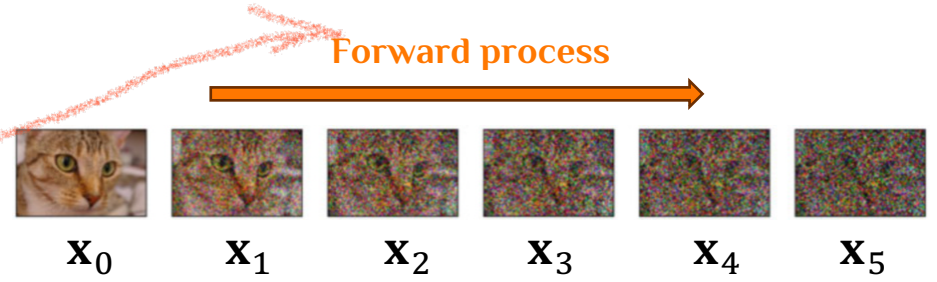
Reverse process (learned)

- The **reverse process** is a generative model

"Creating data from noise is generative modelling" – Yang Song et al.

Image from: Tomczak JM. Deep Generative Modeling. Springer, 2021.
Quote from abstract of: Song Y, Sohl-Dickstein J, Kingma DP, Kumar A, Ermon S, Poole B. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR, 2021.

Stochastic process



- We gradually **transform data into noise**
  - We can model this as a stochastic process $\{\mathbf{x}_t\}_{t\in\mathcal{T}}$ with $\mathrm{x}_t \in \mathcal{X}$
  - Where $\mathbf{x}_0$ is the data and $\mathbf{x}_T$ is complete noise
- Choices for index set $\mathcal{T}$ and state space $\mathcal{X}$

	discrete-valued $\mathbf{x} \in \mathcal{X}$, e.g., $\mathcal{X} = \{0, 1, \dots, 255\}^{28\times 28}$	continuous-valued $\mathbf{x} \in \mathcal{X}$, e.g., $\mathcal{X} = \mathbb{R}^{28\times 28}$
discrete-time $t \in \mathcal{T}$, e.g., $\mathcal{T} = \{0, 1, \dots, T\}$	Markov chain (MC)	DDPM
continuous-time $t \in \mathcal{T}$, e.g., $\mathcal{T} = [0, T]$	Continuous-time Markov chain (CTMC)	SDE based models

Top image from: Image from: Tomczak JM. Deep Generative Modeling. Springer, 2021.

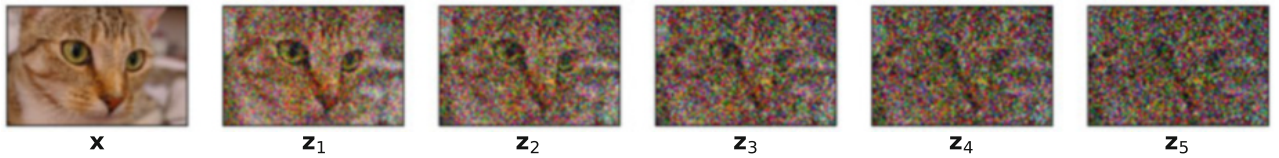
Discrete-time continuous-valued diffusion models

Denoising Diffusion Probabilistic Models (DDPMs)

DDPM

How to corrupt the data?

Forward process



- We blend the image with Gaussian noise

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

where

- $\beta_1, \dots, \beta_T \in (0,1)$ is the noise schedule

Typically, linearly spaced

Ensures the mean get closer to 0

Ensures the variance gets closer to 1

- We can express this as a Gaussian

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

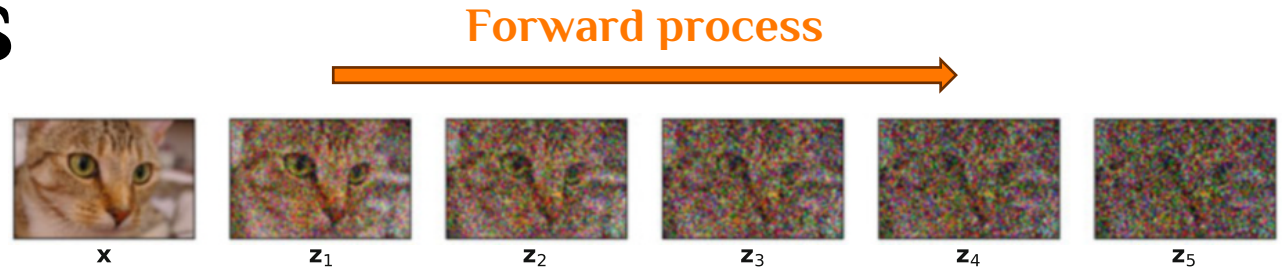
- The entire forward process of T steps is the Markov chain

$$q(\mathbf{z}_{1:T}|\mathbf{x}) = q(\mathbf{z}_1|\mathbf{x})\prod_{t=2}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1})$$

Based on: Bishop CM, Bishop H. Deep learning: foundations and concepts. Spring Nature, 2024. Image from: Tomczak JM. Deep Generative Modeling. Springer, 2021.

DDPM

The forward process



- The forward process up to $\mathbf{z}_t$ is

$$q(\mathbf{z}_{1:t}|\mathbf{x}) = q(\mathbf{z}_1|\mathbf{x}) \prod_{\tau=2}^{t} q(\mathbf{z}_\tau|\mathbf{z}_{\tau-1})$$

- If we marginalise over $\mathbf{z}_{1:t-1}$, we get a Gaussian

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

where $\bar{\alpha}_t$ is the cumulative product $\bar{\alpha}_t = \prod_{\tau=1}^{t}(1-\beta_t)$

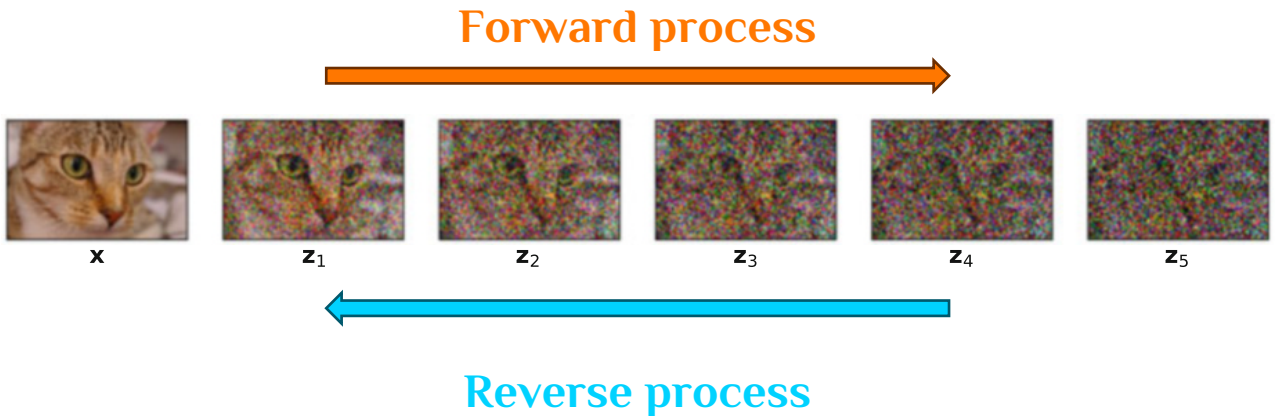
- For large T, datum $\mathbf{x}$ is **transformed to noise**

$$q(\mathbf{z}_T|\mathbf{x}) \to \mathcal{N}(\mathbf{z}_T|\mathbf{0}, \mathbf{I}) \quad \text{as} \quad T \to \infty$$

Based on: Bishop CM, Bishop H. Deep learning: foundations and concepts. Spring Nature, 2024. Image from: Tomczak JM. Deep Generative Modeling. Springer, 2021.

DDPM

Why is the reverse process difficult?



- Using Bayes, we can write

$$q(\mathbf{z}_{t-1}|\mathbf{z}_t) = \frac{q(\mathbf{z}_t|\mathbf{z}_{t-1})q(\mathbf{z}_{t-1})}{q(\mathbf{z}_t)}$$

- However, the marginal $q(\mathbf{z}_t)$ is intractable

$$q(\mathbf{z}_t) = \int q(\mathbf{z}_t|\mathbf{x})p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

  since we do not have $p(\mathbf{x})$.

- We **cannot reverse the forward** process analytically
- Let's **learn** it!

DDPM

How to learn the reverse process (1)?

- Let us assume a **Markovian reverse process** of the form

$$p(\mathbf{x}, \mathbf{z}_{1:T}) = p(\mathbf{x}|\mathbf{z}_1) \prod_{t=1}^{T} p(\mathbf{z}_{t-1}|\mathbf{z}_t)\, p(\mathbf{z}_T)$$

where

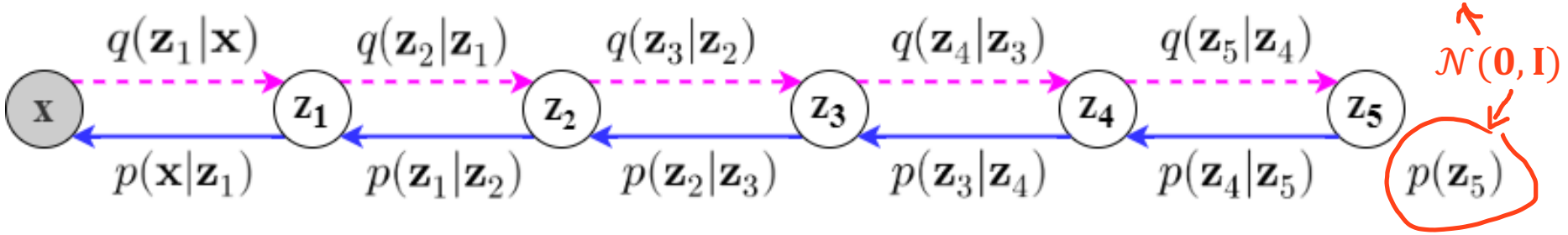
- $p(\mathbf{z}_T) = \mathcal{N}(\mathbf{z}_T|\mathbf{0}, \mathbf{I})$
- $p(\mathbf{z}_{t-1}|\mathbf{z}_t) = \mathcal{N}(\mathbf{z}_{t-1}|\mu_\theta(\mathbf{z}_t, t), \beta_t \mathbf{I})$
- $p(\mathbf{x}|\mathbf{z}_1)$ is a sensible output density

A neural network with learnable weights θ

DDPM

How to learn the reverse process (2)?

- We have samples from $p(\mathbf{x})$
- We know analytically: $q(\mathbf{z}_{1:T}|\mathbf{x}) = q(\mathbf{z}_1|\mathbf{x}) \prod_{t=2}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1})$ ← Variational posterior
- We want to learn: $p(\mathbf{x}, \mathbf{z}_{1:T}) = p(\mathbf{x}|\mathbf{z}_1) \prod_{t=1}^{T} p(\mathbf{z}_{t-1}|\mathbf{z}_t)\, p(\mathbf{z}_T)$ ← Generative model

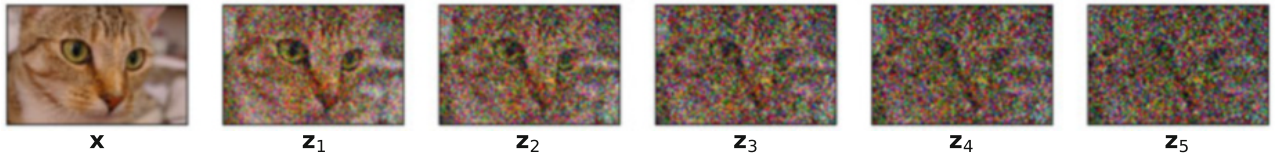


- This looks like a **hieratical VAE!**
- Let's write down an ELBO for $p(\mathbf{x})$

Image by JM Tomczak.

DDPM

The ELBO (1)



We can write the standard ELBO as

$$\log p(\mathbf{x}) \leq \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x})}\left[\log \frac{p(\mathbf{x}, \mathbf{z}_{1:T})}{q(\mathbf{z}_{1:T}|\mathbf{x})}\right]$$

$$= \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x})}\left[-\log p(\mathbf{z}_T) - \sum_{t=1}^{T} \log \frac{p(\mathbf{z}_{t-1}|\mathbf{z}_t)}{q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x})} - \log q(\mathbf{z}_1|\mathbf{x}) + \log p(\mathbf{x}|\mathbf{z}_1)\right]$$

Using Bayes rule on $q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x})$ gives us (see appendix A[1])

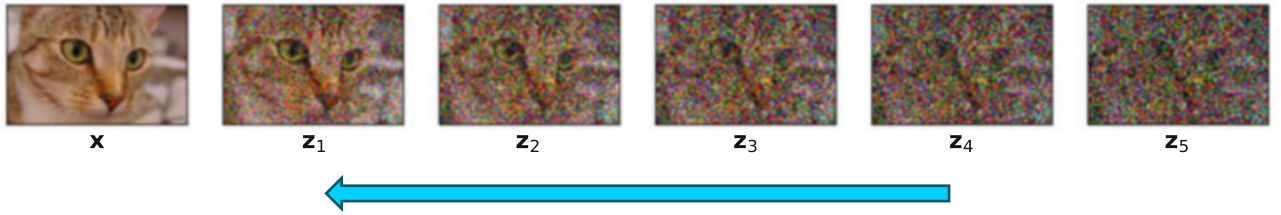
$$\mathcal{L} = \underbrace{-\mathrm{KL}[q(\mathbf{z}_T|\mathbf{x})||p(\mathbf{z}_T)]}_{\mathcal{L}_T} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}\mathrm{KL}[q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) \parallel p(\mathbf{z}_{t-1}|\mathbf{z}_t)]}_{\mathcal{L}_{t-1}} + \underbrace{\mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})}[\ln p(\mathbf{x}|\mathbf{z}_1)]}_{\mathcal{L}_0}$$

$\mathcal{L}_0$ (reconstruction term)

[1]Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. NeurIPS, 2020.
Top image from: Image from: Tomczak JM. Deep Generative Modeling. Springer, 2021.

DDPM

The ELBO (2)



$$\mathcal{L} = \underbrace{-\mathrm{KL}[q(\mathbf{z}_T|\mathbf{x})||p(\mathbf{z}_T)]}_{\mathcal{L}_T} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}\mathrm{KL}[q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x}) \parallel p(\mathbf{z}_{t-1}|\mathbf{z}_t)]}_{\mathcal{L}_{t-1}} + \underbrace{\mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})}[\ln p(\mathbf{x}|\mathbf{z}_1)]}_{\mathcal{L}_0}$$

- We can ignore $\mathcal{L}_T$ as both $q(\mathbf{z}_T|\mathbf{x})$ and $p(\mathbf{z}_T)$ are fixed
- In $\mathcal{L}_{t-1}$, q is Gaussian

$$q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x}) = \mathcal{N}\left(\mathbf{z}_{t-1}\middle|\tilde{\mu}_t(z_t,\mathbf{x}),\tilde{\beta}_t\mathbf{I}\right)$$

  where $\tilde{\mu}_t$ and $\tilde{\beta}_t$ have closed forms.

$$\tilde{\mu}(\mathbf{z}_t,\mathbf{x}) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{z}_t$$

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

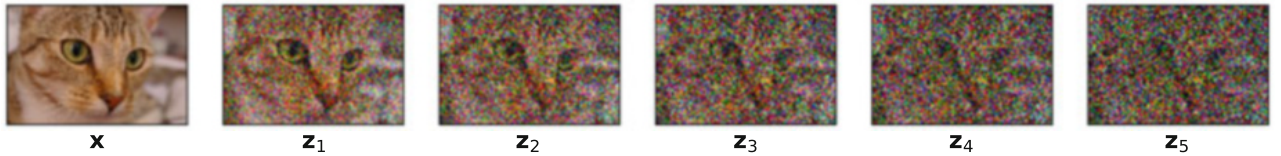
- Since both q and p are Gaussian, the KL in $\mathcal{L}_{t-1}$ have closed form

$$\mathcal{L}_{t-1} = \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}[(2\beta_t)^{-1}\|\tilde{\mu}_t(\mathbf{z}_t,\mathbf{x}) - \mu_\theta(\mathbf{z}_t,t)\|^2] + C$$

[1]Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. NeurIPS, 2020.

DDPM

The ELBO (3)



$$\mathcal{L} = \underbrace{-\mathrm{KL}[q(\mathbf{z}_T|\mathbf{x})||p(\mathbf{z}_T)]}_{\mathcal{L}_T} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}\mathrm{KL}[q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x}) \parallel p(\mathbf{z}_{t-1}|\mathbf{z}_t)]}_{\mathcal{L}_{t-1}} + \underbrace{\mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})}[\ln p(\mathbf{x}|\mathbf{z}_1)]}_{\mathcal{L}_0}$$

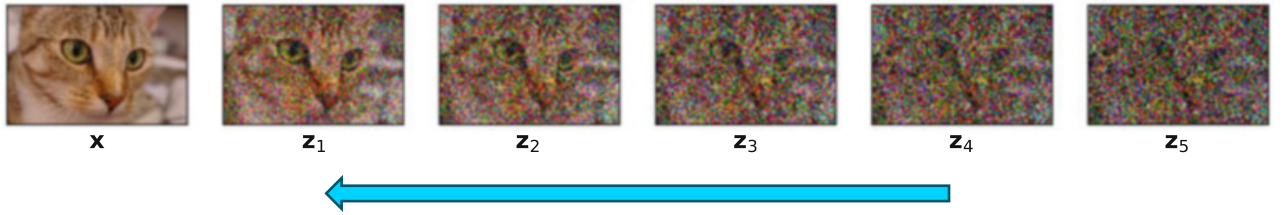
$$(T-1)\mathbb{E}_{t\sim\mathcal{U}\{2,T\}}$$

- $\mathcal{L}_0$ is **tractable by Monte Carlo** as long as $p(\mathbf{x}|\mathbf{z}_1)$ has closed form
  - So, this is a tractable ELBO!
- But for large values of T, it is expensive to evaluate!
  - Using the **law of the unconscious statistician (LOTUS)**, we can replace the sum with an expectation over the uniform distribution on $\{2, 3, \ldots, T\}$
  - Then we can do a **Monte Carlo estimate of the sum**

[1]Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. NeurIPS, 2020.

DDPM

Predicting the noise



- We can reparametrize the expectation $\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}[f(\mathbf{z}_t)]$ as

$$\mathbb{E}_{\mathbf{z}_t \sim q(\mathbf{z}_t|\mathbf{x})}[f(\mathbf{z}_t)] = \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathrm{I})}\left[f\left(z_t(\mathbf{x}, \boldsymbol{\epsilon})\right)\right]$$

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

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

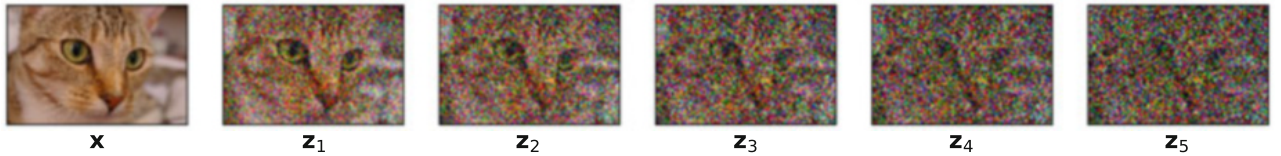
- Then we can write $\mathcal{L}_{t-1}$ as

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

- The **network** need to predict the difference
- We input z_t to the **network**, so can just need to predict the **noise**

DDPM

Simplified $\mathcal{L}_{t-1}$



Reverse process

- This give us the simplified $\mathcal{L}_{t-1}$-term

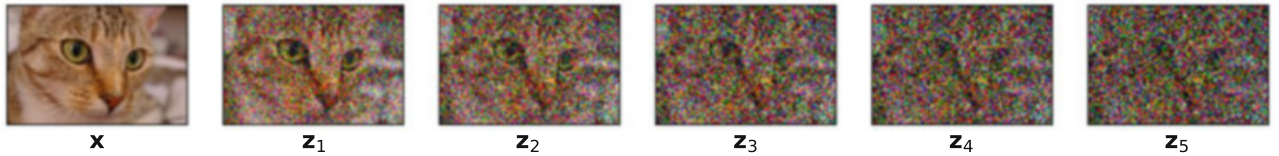
$$\mathcal{L}_{t-1} = \mathbb{E}_{\epsilon}\left[\frac{\beta_t^2}{2\beta_t\alpha_t(1-\bar{\alpha}_t)}\|\epsilon - \epsilon_\theta(z_t(\mathbf{x},\epsilon),t)\|^2\right]$$

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

- Empirically, training works better if we drop the **pre-term**

DDPM

Learning algorithm



Reverse process

For Gaussian output, the learning algorithm[1] for datum $\mathbf{x}$

1. **repeat**
2. $t \sim \mathcal{U}\{1, T\}$
3. $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4. $z_t(\mathbf{x}, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\mathbf{x} + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}$
5. $\mathcal{L}_{t-1} = \|\boldsymbol{\epsilon} - \epsilon_\theta(z_t(\mathbf{x}, \boldsymbol{\epsilon}), t)\|^2$
6. take gradient step on θ using $\nabla_\theta \mathcal{L}_{t-1}$
7. **until** convergence

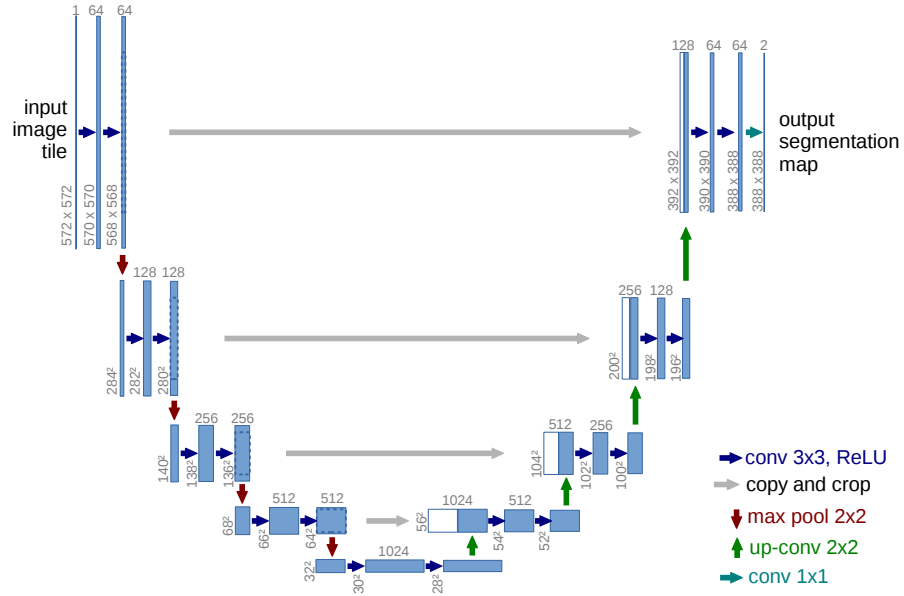
You will implement this today!

[1]Algorithm 1, Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. NeurIPS, 2020.

DDPM

How to parametrize the reverse process?

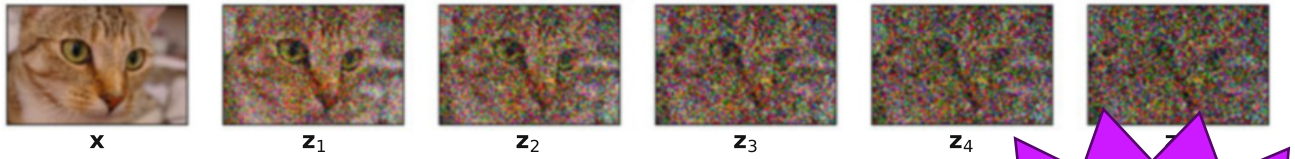
- We want a network that
  - Inputs and output the **same dimensions**
  - Is good at **denoising**
- **U-nets** are does exactly that
- The time is inputted to the network
  - Often using embeddings



Top image: Ronneberger O, Fischer P, Brox MICCAI 2015.
Bottom image: https://cvpr2023-tutorial-diffusion-models.github.io

DDPM

How to sample from the model?



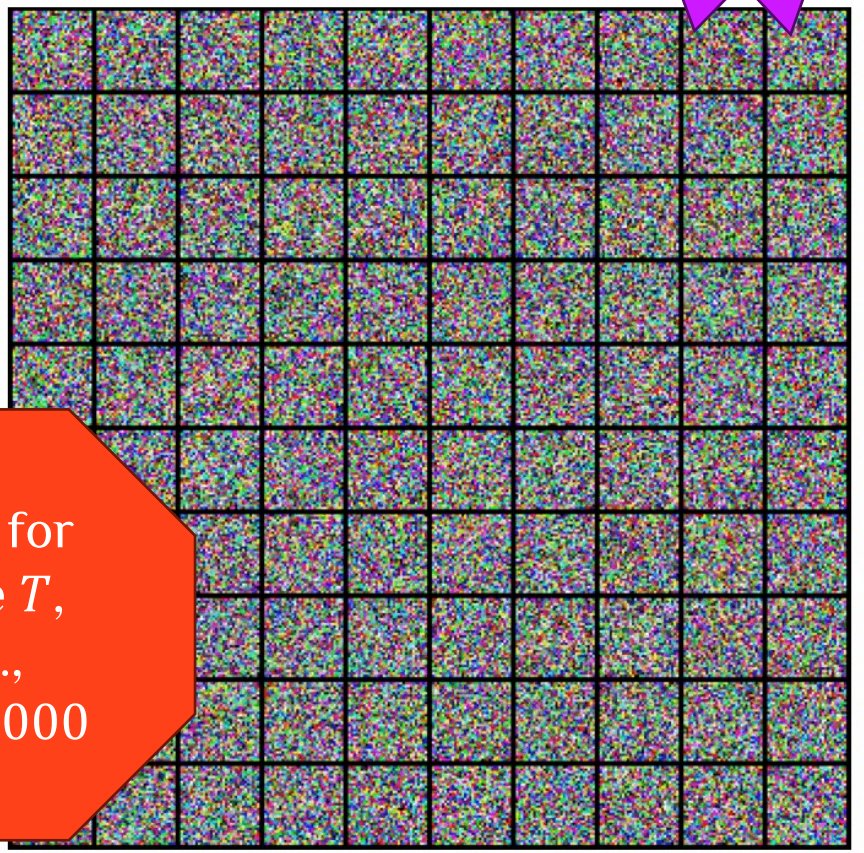
Reverse process

You will implement this today!

We can sample from the model using ancestral sampling

1. $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ← $\mathbf{z}_T \sim p(\mathbf{z}_T)$
2. for $t = T, \ldots, 2$ do
3. $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4. $\mathbf{z}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{z}_t, t)\right) + \sqrt{\beta_t}\boldsymbol{\eta}$ ← $\mathbf{z}_{t-1} \sim p(\mathbf{z}_{t-1}|\mathbf{z}_t)$
5. end for
6. $\mathbf{x} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{z}_1 - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{z}_1, t)\right)$

Generations



Slow for large T, e.g., $T = 1000$

Continuous-time continuous-valued diffusion models

Score-based generative models (SBGM)

Continuous time diffusion models

Diffusion models can extend to **continuous-time** $t \in [0, T]$

- Use a **SDE** for the forward process
- Learn the **reverse SDE** using score matching

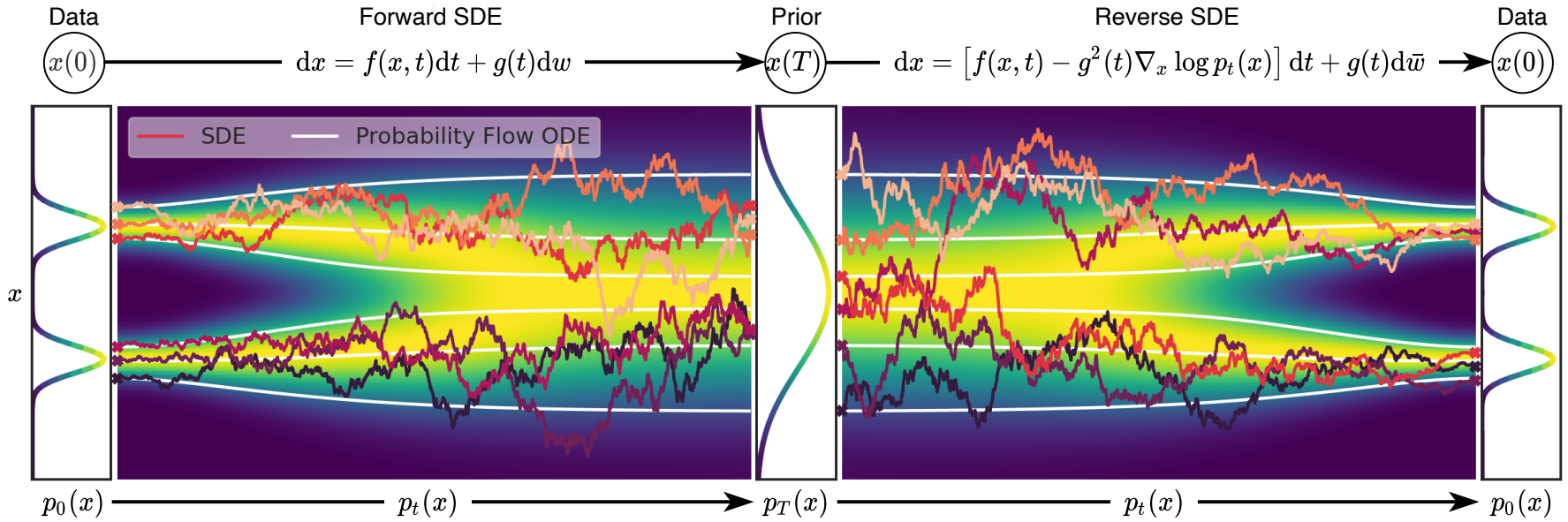


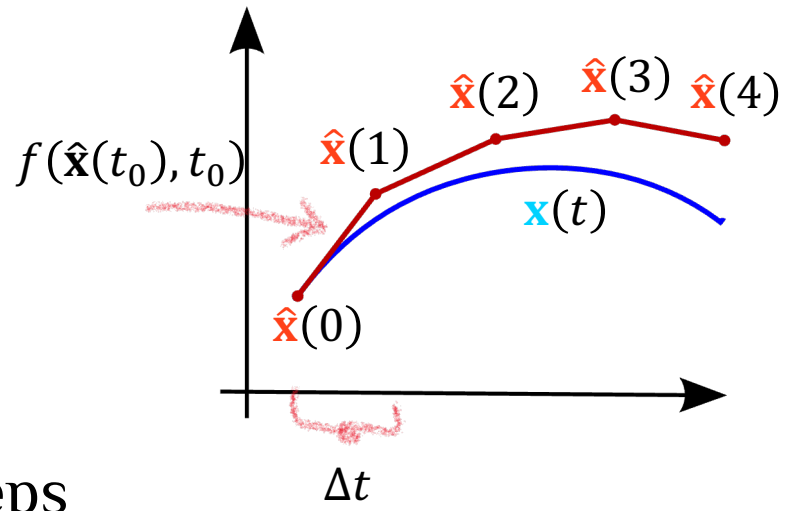
Image: Song Y, Sohl-Dickstein J, Kingma DP, Kumar A, Ermon S, Poole B. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR, 2021.

Ordinary differential equations (ODEs)

- Recall a (first-order) ODE of the form

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), t)$$

for $t \in \mathcal{T} \subseteq \mathbb{R}$ and $\mathbf{x}(t) \in \mathbb{R}^d$



- **Euler method** solves this numerically by
  - Dividing the integration interval $[t_0, T]$ into M steps

$$t_0 < t_1 < \cdots < t_M = T$$

such that $\Delta t = t_{i+1} - t_i$

  - For each step $i = 0, \ldots, M$, we approximate the solution as

$$\hat{\mathbf{x}}(t_{i+1}) = \hat{\mathbf{x}}(t_i) + f(\hat{\mathbf{x}}(t_i), t_i) \cdot \Delta t$$

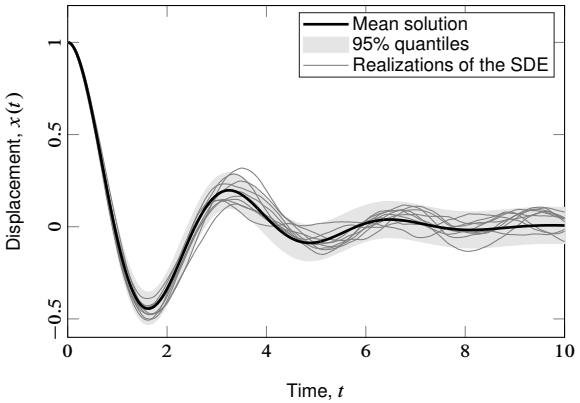
Image adopted from: https://en.wikipedia.org/wiki/Euler_method
See, e.g., section 2.6 in Särkkä S, Solin A. Applied Stochastic Differential Equations. Cambridge University Press, 2019.

Stochastic Differential Equations (SDEs)

- What happens if the ODE is noisy, i.e.,

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), t) + \text{"noise"}$$

Gaussian



- This can be described by an SDE

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

where $\mathbf{w}$ is a Wiener process.

drift — diffusion / dispersion

- **Euler–Maruyama method** solves this numerically by
  - Divide the integration interval $[t_0, T]$ into M steps such that $\Delta t = t_{i+1} - t_i$

$$t_0 < t_1 < \cdots < t_M = T$$

$\mathbf{w}_{i+1} - \mathbf{w}_i$ For each step $i = 0, \ldots, M$, we approximate the solution as

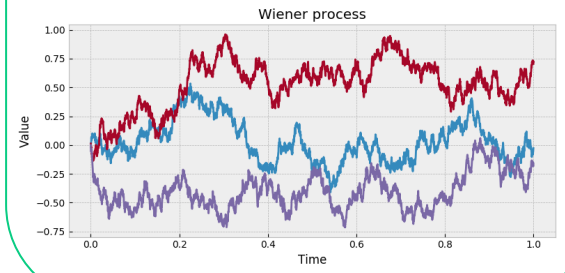
$$\Delta\mathbf{w}_i \sim \mathcal{N}(\mathbf{0}, \Delta t\, \mathbf{I})$$

$$\hat{\mathbf{x}}(t_{i+1}) = \hat{\mathbf{x}}(t_i) + f(\hat{\mathbf{x}}(t_i), t_i) \cdot \Delta t + g(\hat{\mathbf{x}}(t_i), t_i)\Delta\mathbf{w}_i$$

Wiener process

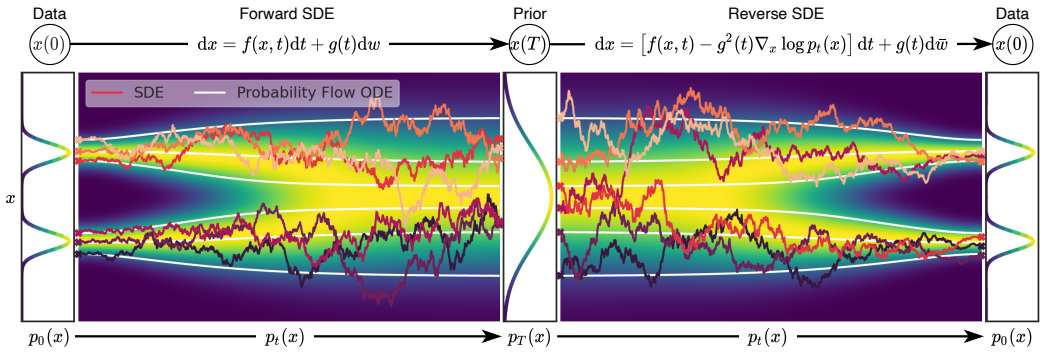
$$w(0) = 0$$

$$w(t) - w(s) \sim \mathcal{N}(0, t - s)$$



Top image from: Särkkä S, Solin A. Applied Stochastic Differential Equations. Cambridge University Press, 2019.
Bottom image from: https://stochastic.readthedocs.io/en/stable/continuous.html

Corrupting data with an SDE



- We construct a process $\{\mathbf{x}(t)\}_{t\in[0,T]}$ where

$$\mathbf{x}(0) \sim p_{\text{data}} \quad \text{and} \quad \mathbf{x}(T) \sim p\big(\mathbf{x}(T)\big) \text{ is tractable}$$

and the **evolution** of $\mathbf{x}_t$ follows the **SDE** Wiener process

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

Assumed known

- It can be shown that the **reverse process** is also an SDE Wiener process with time flows backwards

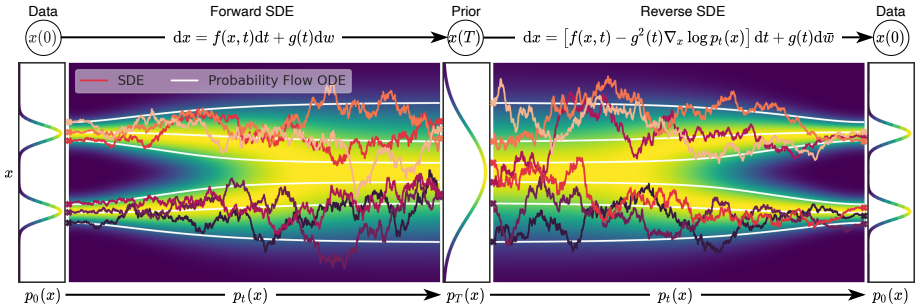
$$d\mathbf{x}(t) = \big[f(\mathbf{x}(t), t) - g(t)^2 \nabla_{\mathbf{x}(t)} \log p\big(\mathbf{x}(t)\big)\big]dt + g(t)d\bar{\mathbf{w}}$$

- We can use the reverse SDE as a generative model for $p\big(\mathbf{x}(0)\big)$

- However, the score $\nabla_{\mathbf{x}(t)} \log p\big(\mathbf{x}(t)\big)$ is analytically intractable

Song Y, Sohl-Dickstein J, Kingma DP, Kumar A, Ermon S, Poole B. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR, 2021.

Learning the score



- Learn a time dependent score model $\mathbf{s}_\theta(\mathbf{x}(t), t) \approx \nabla_{\mathbf{x}(t)} \log p(\mathbf{x}(t))$
- A valid learning objective is the **exact score matching**:

Intractable

Weight function

$$\mathcal{L}_{\text{ESM}}(\theta) = \mathbb{E}_{t\sim U(0,T)}[\lambda(t)\mathbb{E}_{\mathbf{x}(t)\sim p(\mathbf{x}(t))}\left[\left\|\mathbf{s}_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p(\mathbf{x}(t))\right\|_2^2\right]]$$

- $\mathcal{L}_{\text{ESM}}$ is equivalent to the **denoising score matching objective**:

$$\mathcal{L}_{\text{DSM}}(\theta) = \mathbb{E}_{t\sim U(0,T)}[\lambda(t)\mathbb{E}_{\substack{\mathbf{x}(0)\sim p(x)(x(0)) \\ \mathbf{x}(t)\sim p(\mathbf{x}(t)|\mathbf{x}(0))}}\left[\left\|\mathbf{s}_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p(x(t)|x(0)\right\|_2^2\right]]$$

- We still need to know $p(\mathbf{x}(t)|\mathbf{x}(0))$
  - If f and g are linear functions, then $p(\mathbf{x}(t)|\mathbf{x}(0))$ is Gaussian!

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

Song Y, Sohl-Dickstein J, Kingma DP, Kumar A, Ermon S, Poole B. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR, 2021.

Probability flow ODE

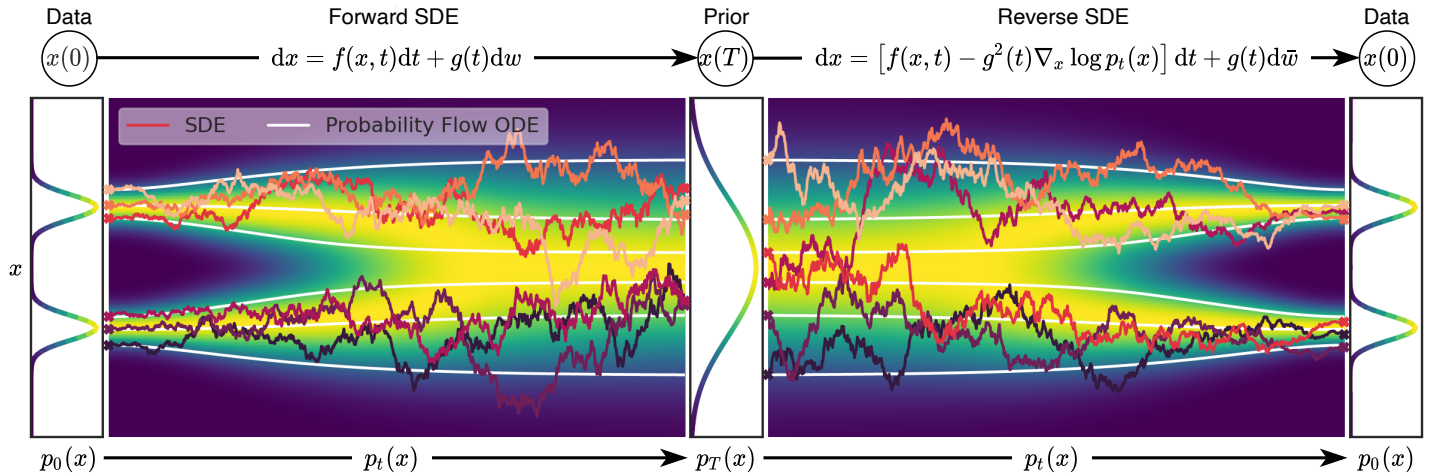
- For an SDE of the form

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

  there exist an ODE with the same marginal probabilities $\{p(\mathbf{x}(t))\}_{t\in[0,T]}$

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), t) - \frac{1}{2}g(t)^2\nabla_{\mathbf{x}(t)}\log p(\mathbf{x}(t))$$

- This ODE allows for
  - **More efficient sampling** using black-box ODE solvers
  - **Likelihood computation** using an ODE solver



Song Y, Sohl-Dickstein J, Kingma DP, Kumar A, Ermon S, Poole B. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR, 2021.

Let's put it all together: Variance preserving SDE ~ DDPM

In the book, Jakub describes the variance exploding SDE

- In the limit of steps $\to \infty$ the DDPM model corresponds to the SDE

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

with linear $\beta(t) = \bar{\beta}_{\text{min}} + t\left(\bar{\beta}_{\text{max}} - \bar{\beta}_{\text{min}}\right)$ and $t \in [0,1]$

In practice, we use $\bar{\beta}_{\text{min}} = 0.1$ and $\bar{\beta}_{\text{max}} = 20$

So for $t = 1$

$e^{-\frac{1}{2}t^2(\bar{\beta}_{\text{max}}-\bar{\beta}_{\text{min}})-\bar{\beta}_{\text{min}}} \approx 0$

and

$p(\mathbf{x}(t)|\mathbf{x}(0)) \approx \mathcal{N}(\mathbf{x}(t)|\mathbf{0},\mathbf{1})$

- You can show that this give the transition kernel

$$p(\mathbf{x}(t)|\mathbf{x}(0)) = \mathcal{N}\left(\mathbf{x}(t)\middle|\boldsymbol{\mu}_{\text{VP}}, \sigma^2_{\text{VP}}\text{I}\right)$$

where

$$\boldsymbol{\mu}_{\text{VP}} = e^{-\frac{1}{4}t^2(\bar{\beta}_{\text{max}}-\bar{\beta}_{\text{min}})-\frac{1}{2}t\bar{\beta}_{\text{min}}} \cdot \mathbf{x}(0)$$

$$\sigma^2_{\text{VP}}\text{I} = 1 - e^{-\frac{1}{2}t^2(\bar{\beta}_{\text{max}}-\bar{\beta}_{\text{min}})-t\bar{\beta}_{\text{min}}}$$

$\nabla_{\mathbf{x}(t)} \log \mathcal{N}(\mathbf{x}(t)|\boldsymbol{\mu}, \sigma^2\text{I})$
$= \frac{1}{\sigma^2}(\boldsymbol{\mu} - \mathbf{x}(t)) = -\frac{1}{\sigma}\boldsymbol{\epsilon}$

$\mathbf{x}(t) = \boldsymbol{\mu} + \boldsymbol{\epsilon}\sigma$
$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Song Y, Sohl-Dickstein J, Kingma DP, Kumar A, Ermon S, Poole B. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR, 2021.

Learning for VP SDE

Due to vanishing gradients: $\tau = 10^{-3}$

1. **repeat**
2. $t \sim \mathcal{U}(\tau, 1)$
3. $\mathbf{x}(\mathbf{0}) \sim p_{\text{data}}$
4. $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5. $\mathbf{x}(t) = \boldsymbol{\mu}_{\text{VP}} + \sigma_{\text{VP}} \cdot \boldsymbol{\epsilon}$
6. $\mathcal{L}(\theta) = \lambda(t) \left\| \mathbf{s}_\theta(\mathbf{x}(t), t) + \frac{1}{\sigma_{\text{VP}}} \boldsymbol{\epsilon} \right\|_2^2 = \frac{\lambda(t)}{\sigma_{\text{VP}}^2} \| \tilde{\mathbf{s}}_\theta(\mathbf{x}(t), t) - \boldsymbol{\epsilon} \|_2^2$
7. take gradient step on θ using $\nabla_\theta \mathcal{L}(\theta)$
8. **until** convergence

$\nabla_{\mathbf{x}(t)} \log p(x(t)|x(0))$

$\tilde{\mathbf{s}}_\theta(\mathbf{x}(t), t) = -\sigma_{\text{VP}} \cdot \mathbf{s}_\theta(\mathbf{x}(t), t)$

Song Y, Sohl-Dickstein J, Kingma DP, Kumar A, Ermon S, Poole B. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR, 2021.

Sampling for VP SDE

$$\beta(t) = \bar{\beta}_{\min} + t\left(\bar{\beta}_{\max} - \bar{\beta}_{\min}\right)$$

We discretise the interval $[\tau, 1]$ in $T + 1$ steps with $\tau = 10^{-3}$

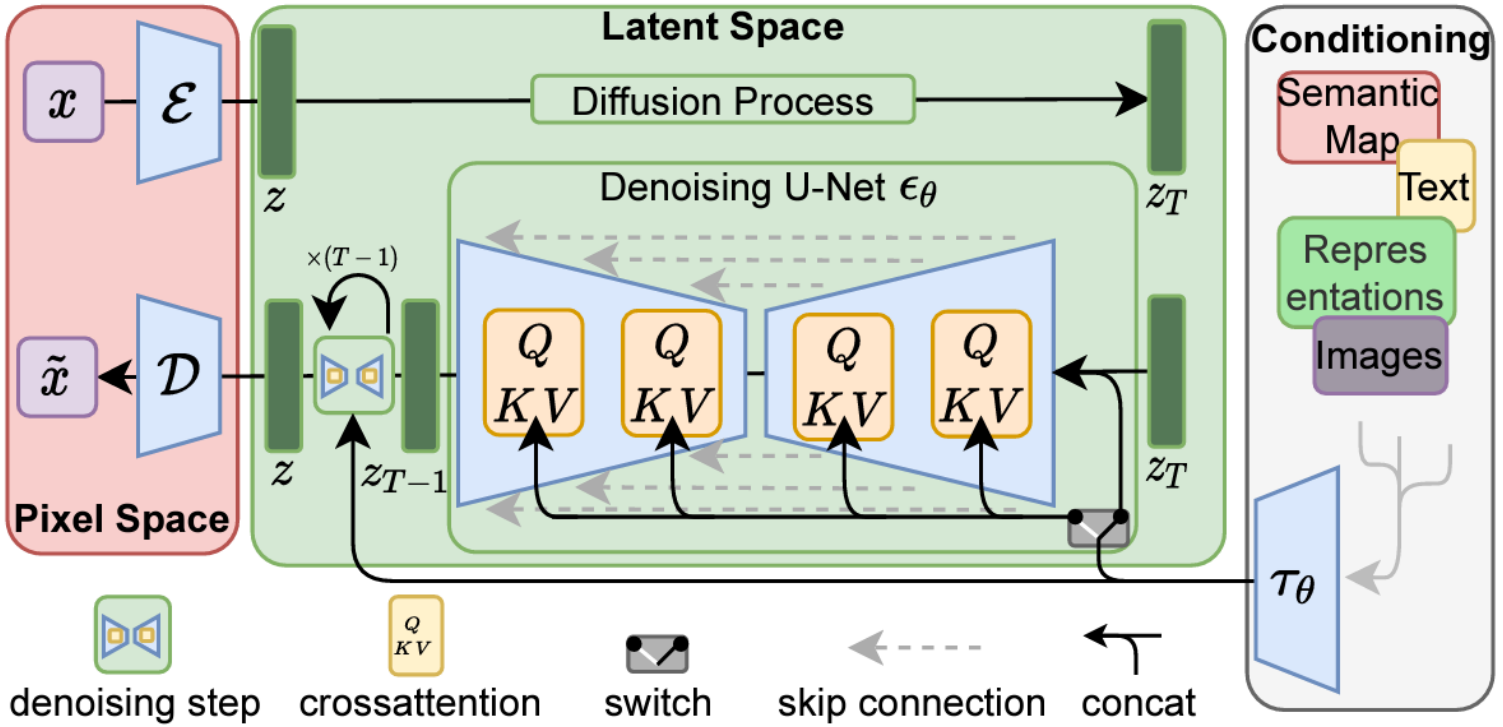
1. $\Delta t = \frac{1-\tau}{T}$
2. $\hat{\mathbf{x}}(t_T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
3. **for** $i = T - 1, \ldots, 0$ do
4. $\quad t_i = \tau + i \cdot \Delta t$
5. $\quad \Delta \mathbf{w}_{i+1} \sim \mathcal{N}(\mathbf{0}, \Delta t\, \mathbf{I})$
6. $\quad \hat{\mathbf{x}}(t_i) = \hat{\mathbf{x}}(t_{i+1}) + \frac{1}{2}\beta(t_{i+1})\hat{\mathbf{x}}(t_{i+1})\Delta t + \beta(t_{i+1})\mathbf{s}_\theta(\hat{\mathbf{x}}(t_{i+1}), t_{i+1})\Delta t + \sqrt{\beta(t_{i+1})}\Delta \mathbf{w}_{i+1}$
7. **end for**

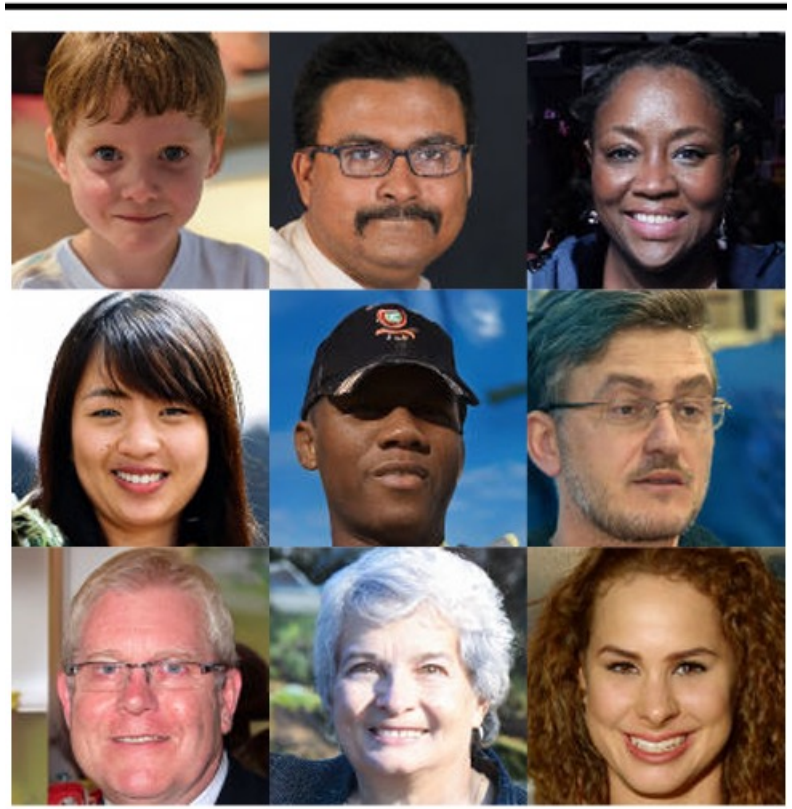
$$\hat{\mathbf{x}}(t_i) = \hat{\mathbf{x}}(t_{i+1}) - f(\hat{\mathbf{x}}(t_{i+1}), t_{i+1}) \cdot \Delta t + g(t_{i+1})^2 \nabla_{\mathbf{x}(t)} \log p\left(\hat{\mathbf{x}}(t_{i+1})\right) \cdot \Delta t + g(t_{i+1}) \Delta \mathbf{w}_{i+1}$$

Song Y, Sohl-Dickstein J, Kingma DP, Kumar A, Ermon S, Poole B. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR, 2021.

Stable Diffusion

- The main idea: Do diffusion in the latent space for better scaling!

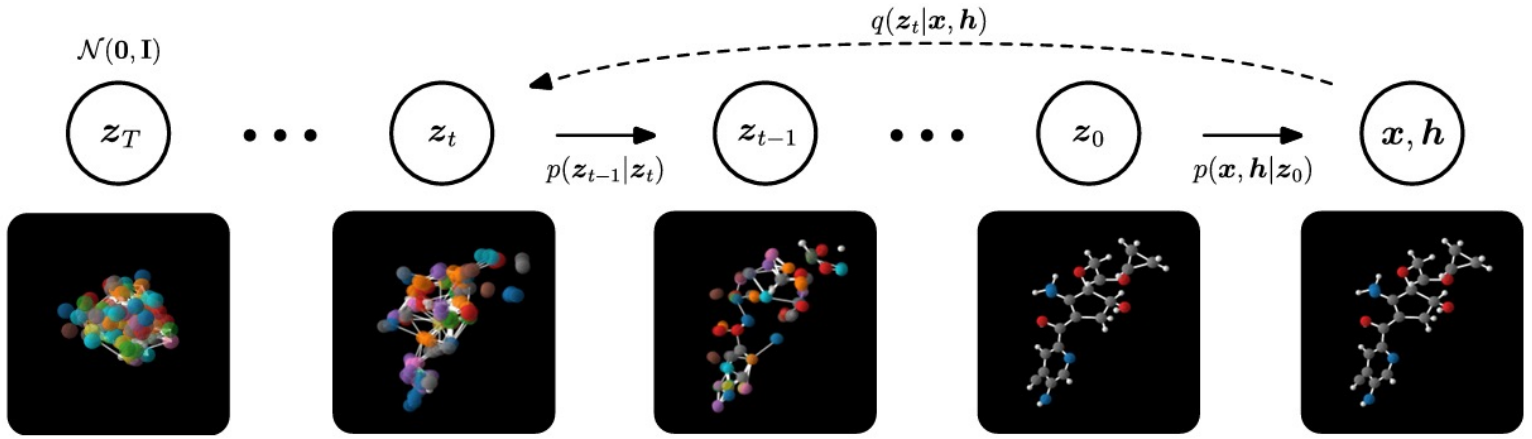


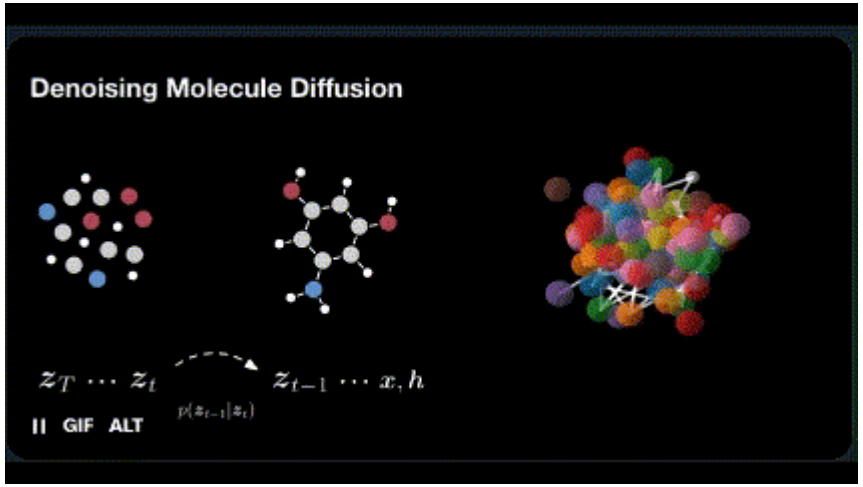


Rombach et al., CVPR 2022 "High-resolution image synthesis with latent diffusion models"

Molecule Generation with diffusion models

- Generating molecules in 3D





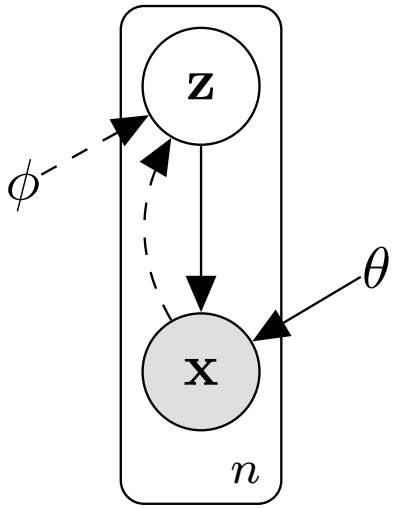
Hoogeboom et al., ICML 2022 "Equivariant Diffusion for Molecule Generation in 3D"

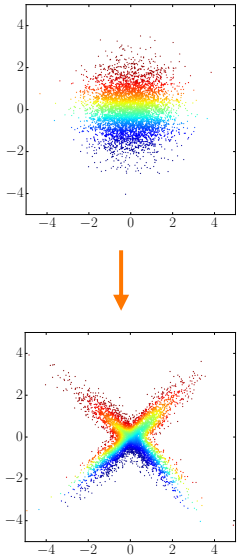
Take-away!

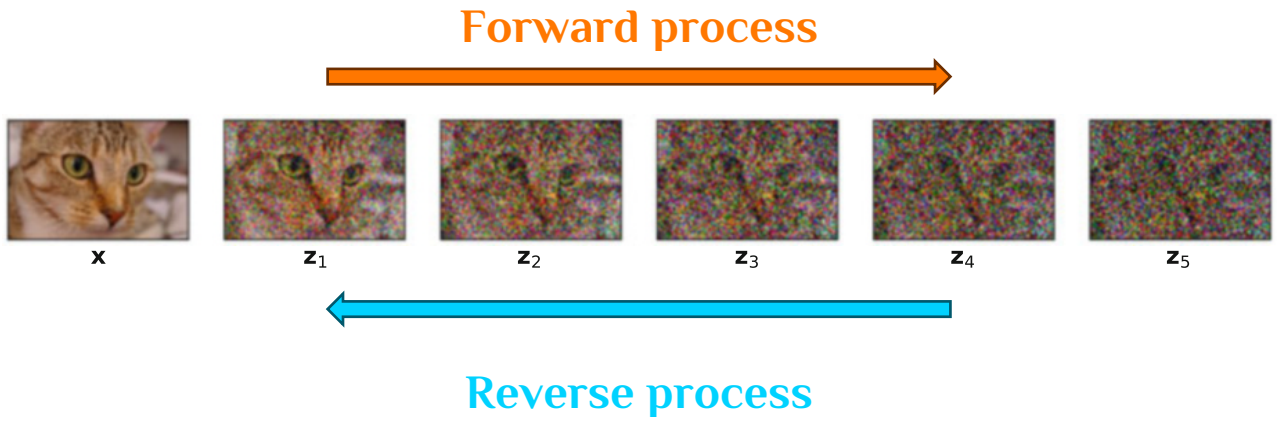
- **Diffusion models** produced high quality images
- The **DDPM** can be considered a very deep heretical VAE
  - Learned via an **ELBO**
  - Sapling using **ancestral sampling** (slow)
- **Applications**
  - Image synthesis (mainly!)
  - But also, music, audio, molecules...

Overview of models considered

Model	Training	Likelihood	Invertible	Bottleneck (latent)	Sampling speed
VAE	Stable	Approximate	No	Yes	Fast
Flow	Stable	Exact	Yes	No	Fast
DDPM	Stable	Approximate	No	No	Slow
SBGM	Stable	“Exact”	No / Yes (ODE)	No	Slow







Thank you!