

Introduction to diffusion models

François Rousseau



IMT Atlantique
Bretagne • Pays de la Loire
École Mines-Télécom

Ressources

Diffusion Models: A Comprehensive Survey of Methods and Applications

LING YANG*, Peking University, China

ZHILONG ZHANG*, Peking University, China

YANG SONG, OpenAI, USA

SHENDA HONG, Peking University, China

RUNSHENG XU, University of California, Los Angeles, USA

YUE ZHAO, Carnegie Mellon University, USA

YINGXIA SHAO, Beijing University of Posts and Telecommunications, China

WENTAO ZHANG, Mila - Quebec AI Institute, HEC Montréal, Canada

BIN CUI, Peking University, China

MING-HSUAN YANG, University of California at Merced, USA.

Diffusion models have emerged as a powerful new family of deep generative models with record-breaking performance in many applications, including image synthesis, video generation, and molecule design. In this survey, we provide an overview of the rapidly expanding body of work on diffusion models, categorizing the research into three key areas: efficient sampling, improved likelihood estimation, and handling data with special structures. We also discuss the potential for combining diffusion models with other generative models for enhanced results. We further review the wide-ranging applications of diffusion models in fields spanning from computer vision, natural language processing, temporal data modeling, to interdisciplinary applications in other scientific disciplines. This survey aims to provide a contextualized, in-depth look at the state of diffusion models, identifying key areas of focus and pointing to potential areas for further exploration. GitHub: <https://github.com/YangLing91/Diffusion-Models-Papers-Survey-Taxonomy>.

CVPR 2022 Tutorial:

Denoising Diffusion-based Generative Modeling: Foundations and Applications

Date: Sunday June 19, 8:30 am - 12:10 pm (CDT)



Generative Modeling by Estimating Gradients of the Data Distribution

This blog post focuses on a promising new direction for generative modeling. We can learn score functions (probabilities of log probability density function) from a batch of real-world data distributions, then generate samples with Langevin-type sampling. The resulting generative model, often called ScoreNet or ScoreGAN, has several important advantages over existing model families: GAN-level sample quality without adversarial training, flexible model architecture, exact log-likelihood computation, and inverse gradient solving without training models. In this blog post, we will show you more details like intuition, basic concepts, and potential applications of score-based generative models.

Author:
Yang Song

Published:
January 1, 2021

Updated:
May 8, 2021

UFLog 2+

PAGE · ACTIVE · 26132 · TAGS · 147 · RECENT · SEARCH · LOGOUT

What are Diffusion Models?

July 17, 2021 · 31 min · Edit · Permalink

Table of Contents

Updated on 2021-07-18 · Highly recommended this blog post on score-based generative modeling by Yang Song is one of several key papers in the field.

Updated on 2021-08-21 · Adamo JelLINE et al. introduce SDEs, SDEUR, and RHO-G.

Updated on 2021-08-21 · Aditya John et al. introduce model.

Summary: This article gives an overview of generative models, SDEs, VAEs, and Pixelcnn models. They introduce SDEs (Score-based denoising flow) as a generalization of the standard forward Euler method. SDEs are known for potentially avoidable training and less difficulty in generating samples. It can accomplish learning rates as VAEs, but in a simpler form. These models have some specific fixed conditions to maintain invertible functions.

Diffusion models are inspired by non-equilibrium thermodynamics. They define a Markov chain of diffusions that is slowly adding noise to data and then back to zero. The diffusions converge to generate unconditional samples from the noise. Unlike GANs or Pixelcnn, diffusion models are learned with a fixed procedure and the latent variable has high cross-correlation (noise vs. the original image).

Stochastic Differential Equations and Diffusion Models

Written: October 2020 · 11 min · Edit · Permalink

Diffusion models (Song, Schulman, & Ermon, 2019) are one of the hottest topics of generative models in this decade (not because of solving this puzzle). They have been shown to outperform GANs in certain settings (Belghazi, Makhzani, & Courville, 2018), and even trained on raw sensor data (Song et al., 2020). There are two main reasons that are also very elegant. The model is trained toward a particular corruption process which samples a target distribution (say the distribution of fake images in a database). It approximates a Gaussian noise. Once the model has been trained, we can generate new samples from the target distribution by applying Gaussian noise and corrupt through the corruption process.

In particular, the diffusion models use a multi-step corruption procedure where, at each step, the input is randomly multiplied by a random loss function, and an average weight (which will come from some mean Gaussian noise). That is to say, the output of modeling is, e.g., a frame from a Gaussian measurement or noise using a downsampled version of its input. Consequently, then, the theoretical distribution characterizing such steps would resemble a Gaussian. In this process, the scaling factor and the variance for the covariance matrix in case of multi-dimensional data are often functions of the data number.

The inversion process is defined in a manner similar to the corruption process and goes from the last step to the first. The output of an inversion step is defined as a sample from a Gaussian distribution, but now instead of Gaussian having a scaled version of the input, it adds noise.

Reading time:

estimated 1 min

diffusion · score

scorenet · scoregan

scoreur · rho-g

adamo-jeline · sde

aditya-john · model

Contents

Introduction · Previous

Forward-Backward · Previous

Pixelcnn · Previous

Sampling · Previous

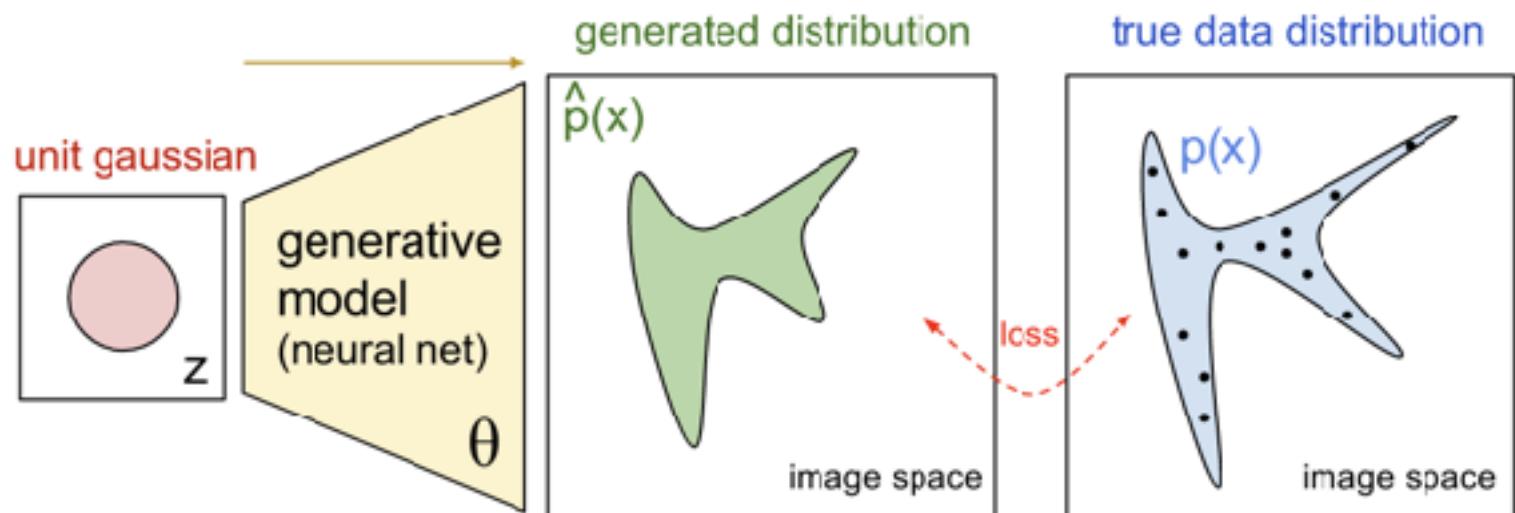
Diffusion Models · Previous

ScoreNet

ScoreGAN

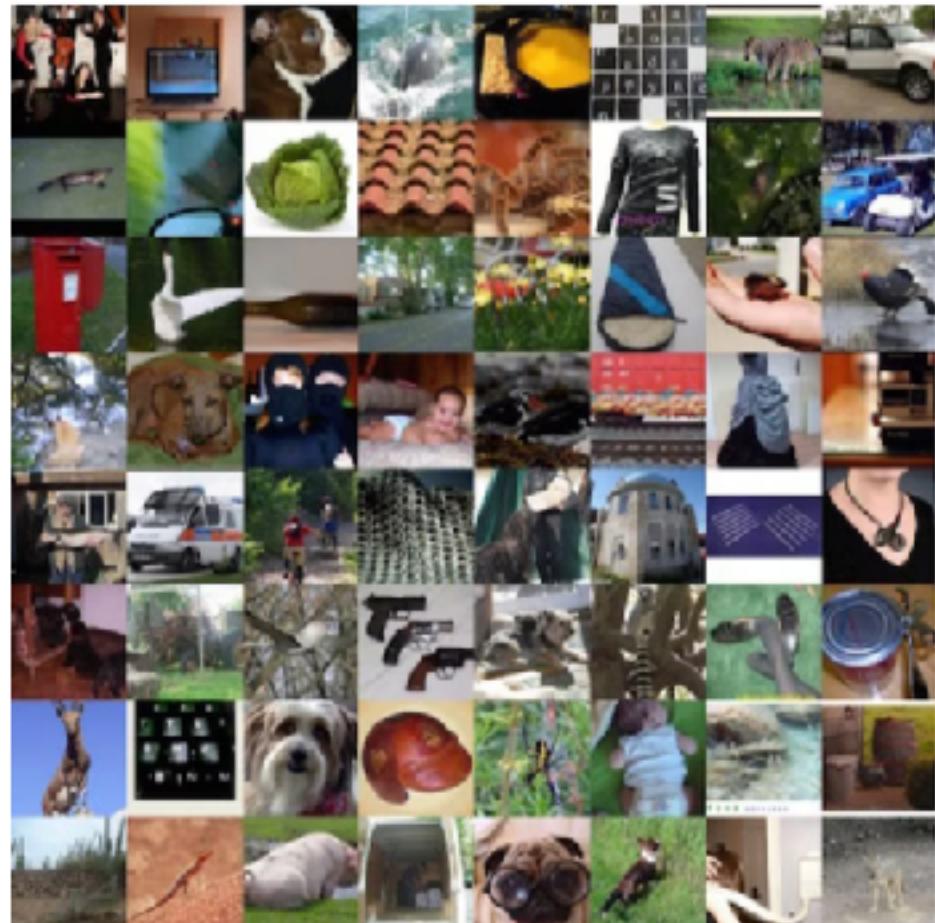
Generative modeling

Objective : draw a random sample z to generate a data x

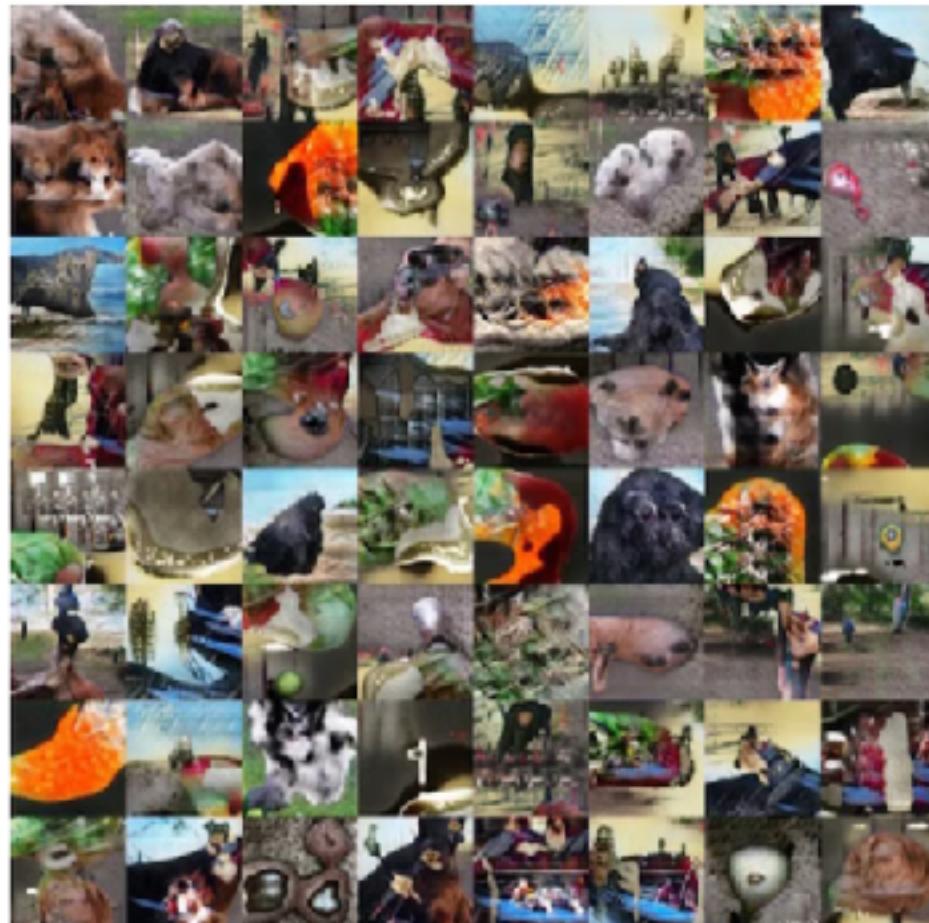


[openAI]

Generative modeling using GAN

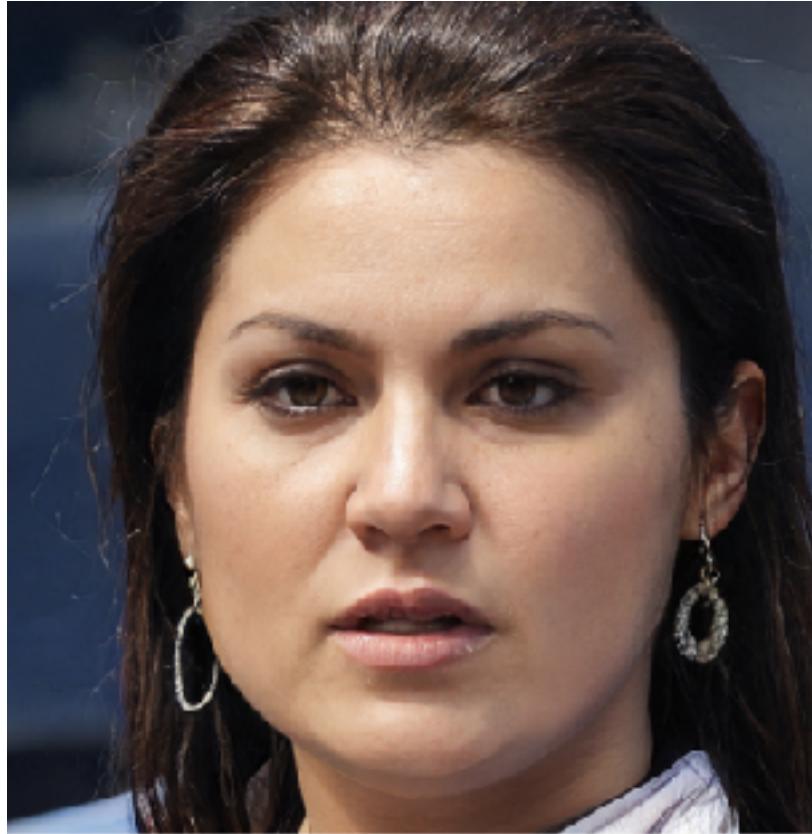


Real images (ImageNet)

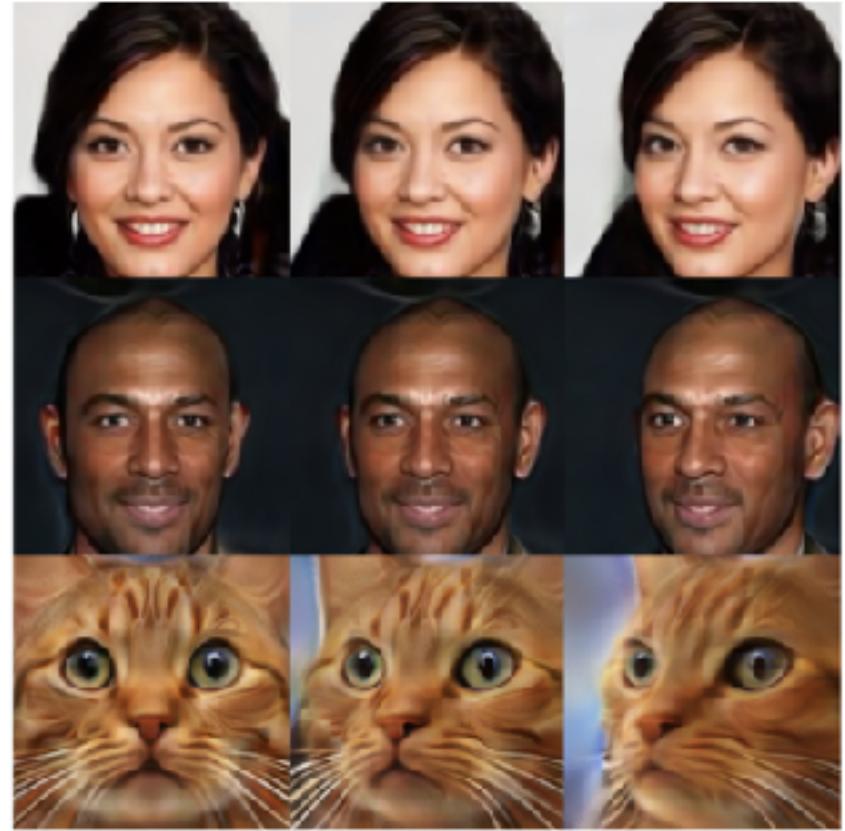


Generated images

Generative modeling using GAN



[This person does not exist !]



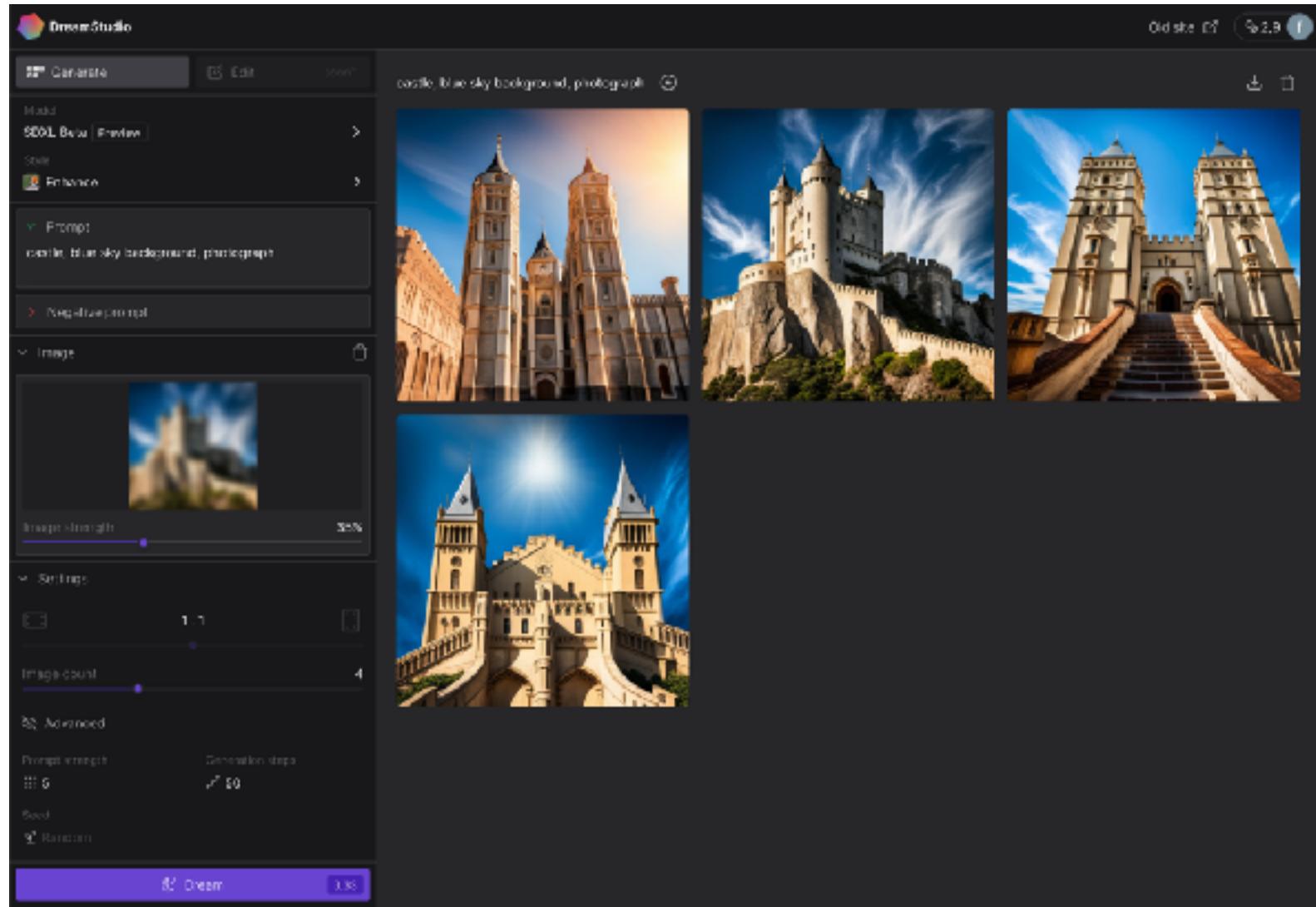
[Chan et al. pi-GAN: Periodic Implicit Generative Adversarial Networks for 3D-Aware Image Synthesis. CVPR 2021]

Generative modeling with diffusion models



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

Diffusion models for everyone



[stability.ai]

Diffusion models and Hugging Face Diffusers API

minimalist_hf_text-to-image.ipynb

Fichier Modifier Affichage Insérer Exécution Outils Aide

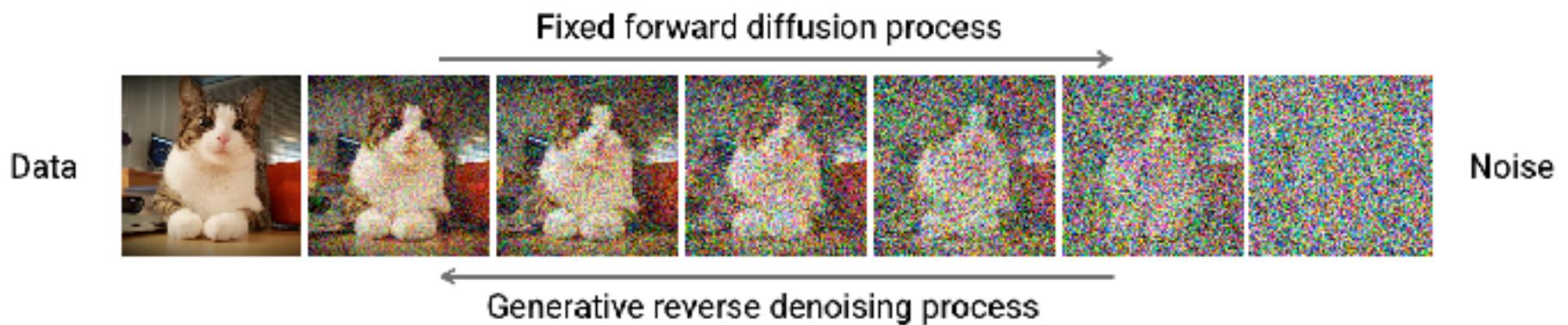
+ Code + Texte

```
[1]: %pip install -qq -L diffusers
[2]: from diffusers import DiffusionPipeline
[3]: pipeline = DiffusionPipeline.from_pretrained("runwayml/stable-diffusion-v1-5")
    pipeline.to("cuda")
[4]: pipeline("An image of a squirrel in Picasso style").images[0]
```

100% 50/50 [0:24<00:00, 2.07ms]

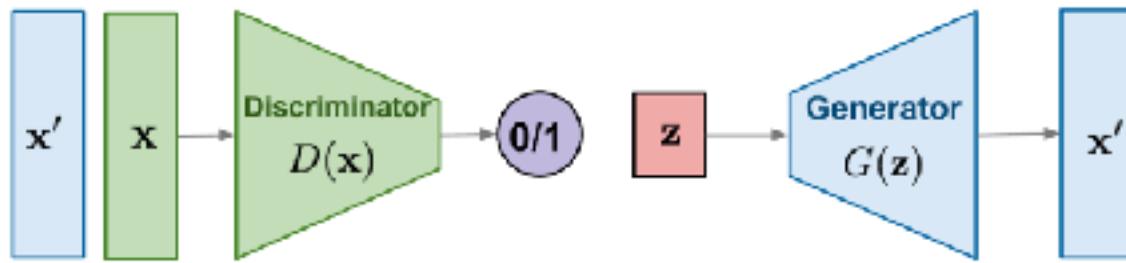
A generated image of a squirrel in a Picasso-style painting. The squirrel is depicted with thick, expressive brushstrokes in shades of blue, white, and black. It has a large, bushy tail and is set against a background of warm colors like orange, red, and yellow. The overall composition is abstract and artistic, capturing the form of the squirrel through geometric shapes and bold lines.

Overview of diffusion models

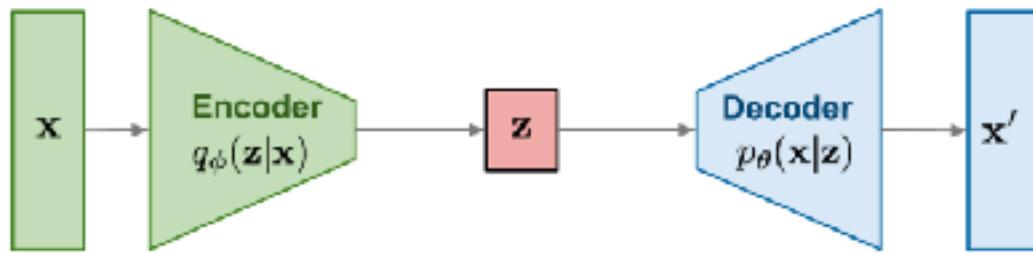


Overview of generative models

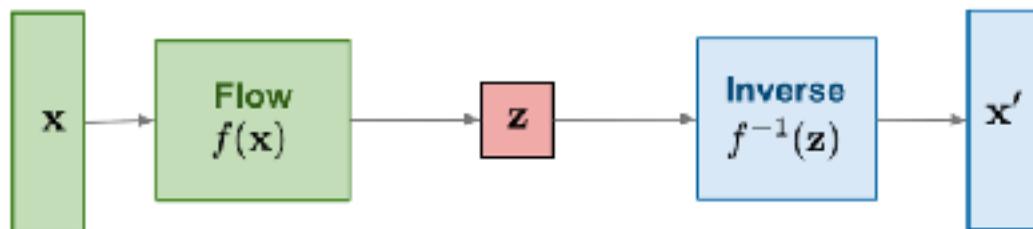
GAN: Adversarial training



VAE: maximize variational lower bound



Flow-based models:
Invertible transform of distributions

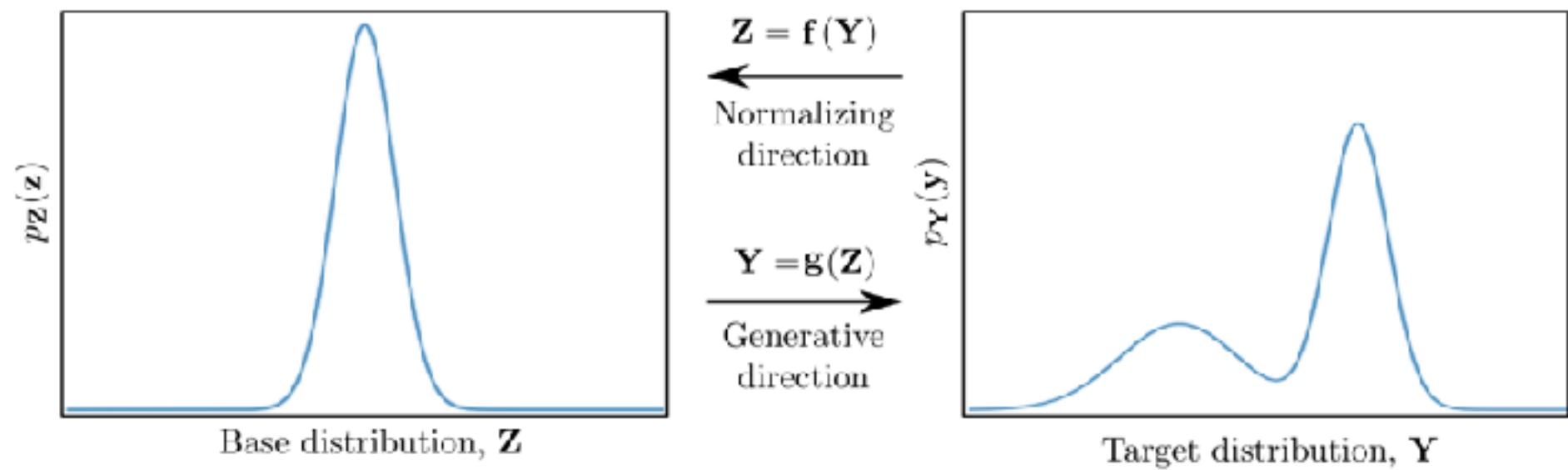


Diffusion models:
Gradually add Gaussian noise and then reverse



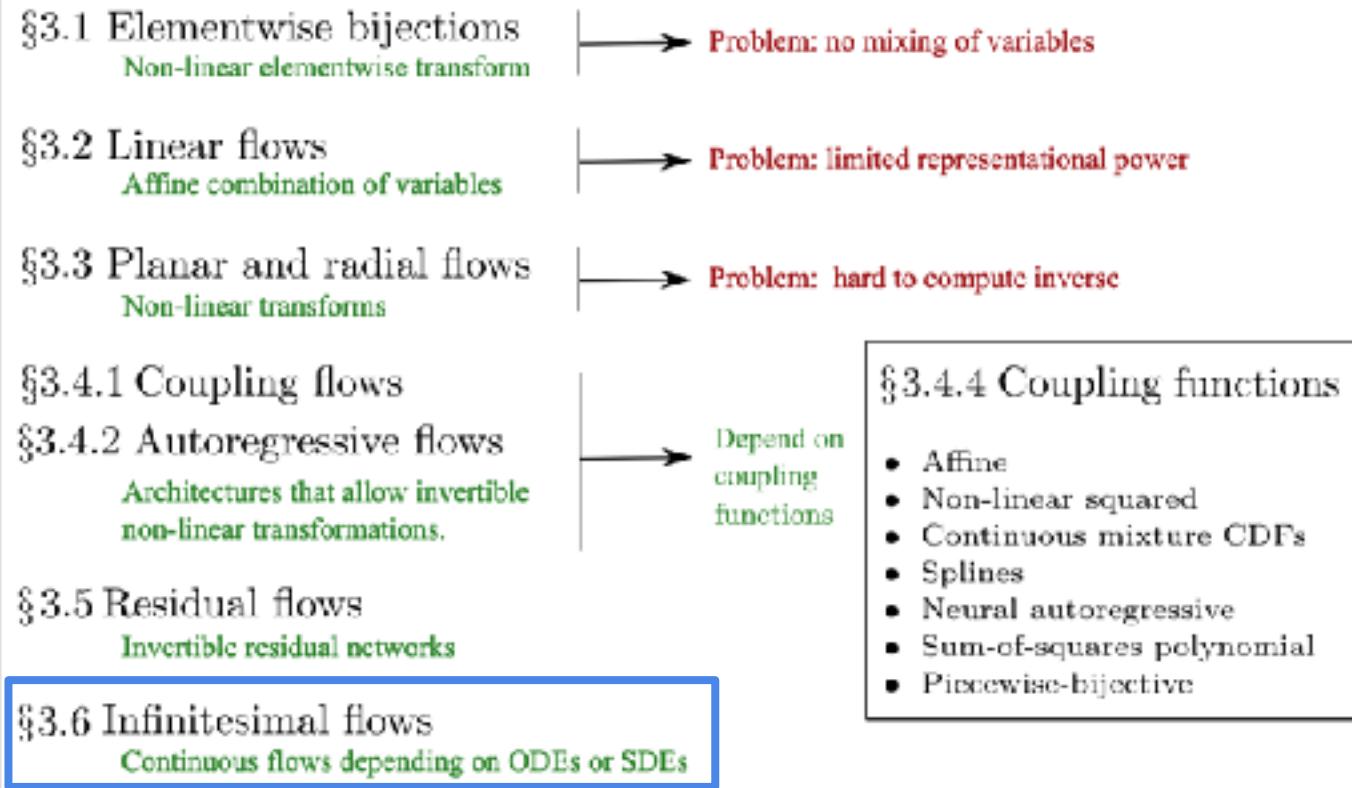
Density mapping

Objective of generative modeling : draw a random sample z to generate a data



[Kobyzev et al. Normalizing flows: an introduction
and review of currents methods. IEEE PAMI 2020]

Normalizing flows



Note, that even though Langevin flows manifest nice mathematical properties, they have not found practical applications.

[Kobyzev et al. Normalizing flows: an introduction and review of currents methods. IEEE PAMI 2020]

Discrete-Time Markov Processes

Sequence of random variables, with Markov properties:

$$P(X_0 = x_0, X_1 = x_1, X_2 = x_2, \dots) = P(X_0 = x_0) \prod_{t \in \{1, 2, \dots\}} P(X_t = x_t | X_{t-1} = x_{t-1})$$

Such a Markov process is completely defined by the initial state and the transition probabilities.

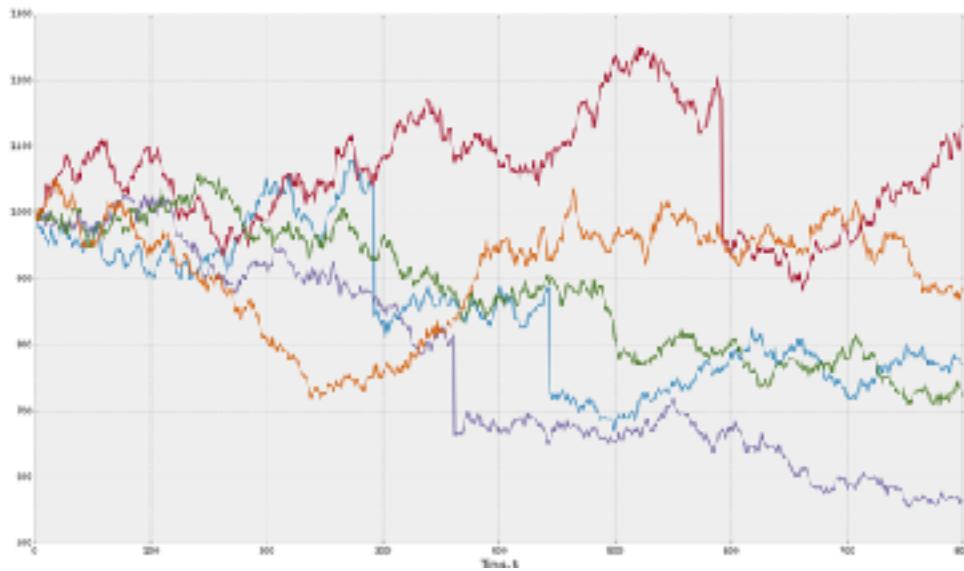
Given two states, one can write:

$$P(X_t = x_t | X_s = x_s) = \sum_k P(X_t = x_t | X_m = k) P(X_m = k | X_s = x_s)$$

Continuous-time Markov processes

In 1931, Kolmogorov found that there are two kinds of continuous time Markov processes, depending on the assumed behavior over small intervals of time:

- Jump processes (like Poisson process)
- Diffusion processes (represented by diffusion and by Brownian motion)



The Kolmogorov forward equation

Evolution of the conditional density function over a small interval:

$$p(x; t + dt|y; s) = \int_{-\infty}^{\infty} p(x; t + dt|m; t)p(m; t|y; s)dm$$

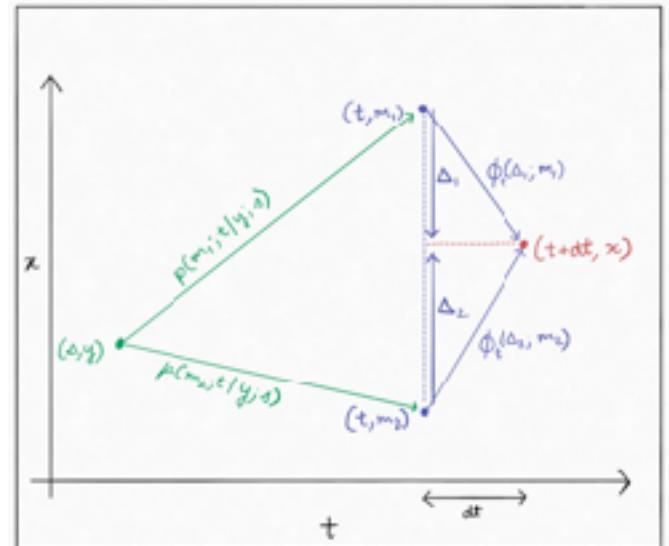
Let define: $\phi_t(\Delta; z) = p(z + \Delta; t + dt|z; t)$

Rewrite using: $m = x - \Delta$

$$\Rightarrow dm = -d\Delta$$

$$m = \pm\infty \Rightarrow \Delta = \mp\infty$$

$$\begin{aligned}\Rightarrow p(x; t + dt|y; s) &= - \int_{\infty}^{-\infty} \phi_t(\Delta; m)p(m; t|y; s)d\Delta \\ &= \int_{-\infty}^{+\infty} \phi_t(\Delta; m)p(m; t|y; s)d\Delta\end{aligned}$$



[T. S. Dabral, 2021]

The Kolmogorov forward equation

Taylor's theorem: $f(x) = f(a) + \frac{f'(a)}{1!} (x - a) + \frac{f^{(2)}(a)}{2!} (x - a)^2 + \cdots + \frac{f^{(n)}(a)}{n!} (x - a)^n + R_n(x)$

Using Taylor's theorem around the point x :

$$\begin{aligned} p(x; t + dt|y; s) &= \int_{-\infty}^{+\infty} \phi_t(\Delta; x) p(x; t|y; s) d\Delta \\ &\quad - \int_{-\infty}^{+\infty} \Delta \frac{\partial}{\partial x} \phi_t(\Delta; x) p(x; t|y; s) d\Delta \\ &\quad + \int_{-\infty}^{+\infty} \frac{\Delta^2}{2} \frac{\partial^2}{\partial x^2} \phi_t(\Delta; x) p(x; t|y; s) d\Delta \\ &\quad \vdots \end{aligned}$$

We get:

$$\begin{aligned} p(x; t + dt|y; s) - p(x; t|y; s) &= -\frac{\partial}{\partial x} (\mathbb{E}_{\Delta \sim \phi_t(\cdot; x)} [\Delta] p(x; t|y; s)) \\ &\quad + \frac{1}{2} \frac{\partial^2}{\partial x^2} (\mathbb{E}_{\Delta \sim \phi_t(\cdot; x)} [\Delta^2] p(x; t|y; s)) \\ &\quad \vdots \end{aligned}$$

The Kolmogorov forward equation

Let define:

$$\mathbb{E}_{\Delta \sim \phi_t(\cdot; x)}[\Delta] := f(x, t)dt$$

$$\mathbb{E}_{\Delta \sim \phi_t(\cdot; x)} [\Delta^2] := g^2(x, t)dt$$

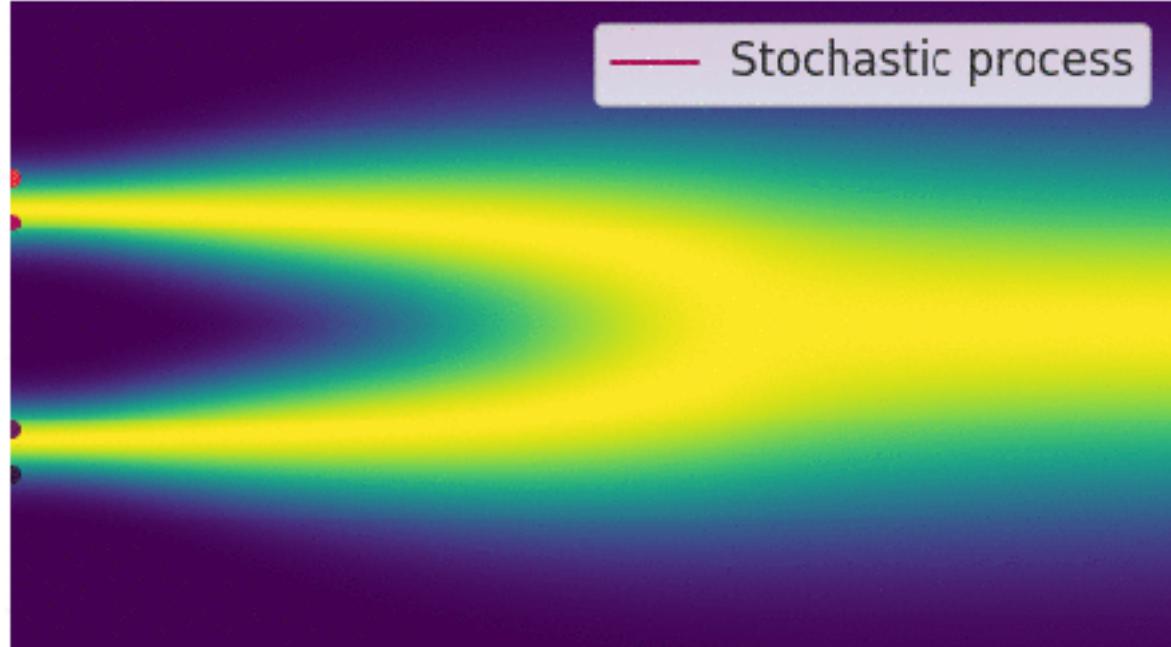
By truncating the Taylor series up to the second moment, we get the following partial differential equation:

$$\frac{\partial}{\partial t} p(x; t|y; s) = -\frac{\partial}{\partial x} (f(x, t)p(x; t|y; s)) + \frac{1}{2} \frac{\partial^2}{\partial x^2} (g^2(x; t)p(x; t|y; s))$$

This is the Kolmogorov forward equation, also known as the Fokker-Planck equation.

F is called the *drift coefficient* of the diffusion process and g is called the *diffusion coefficient*.

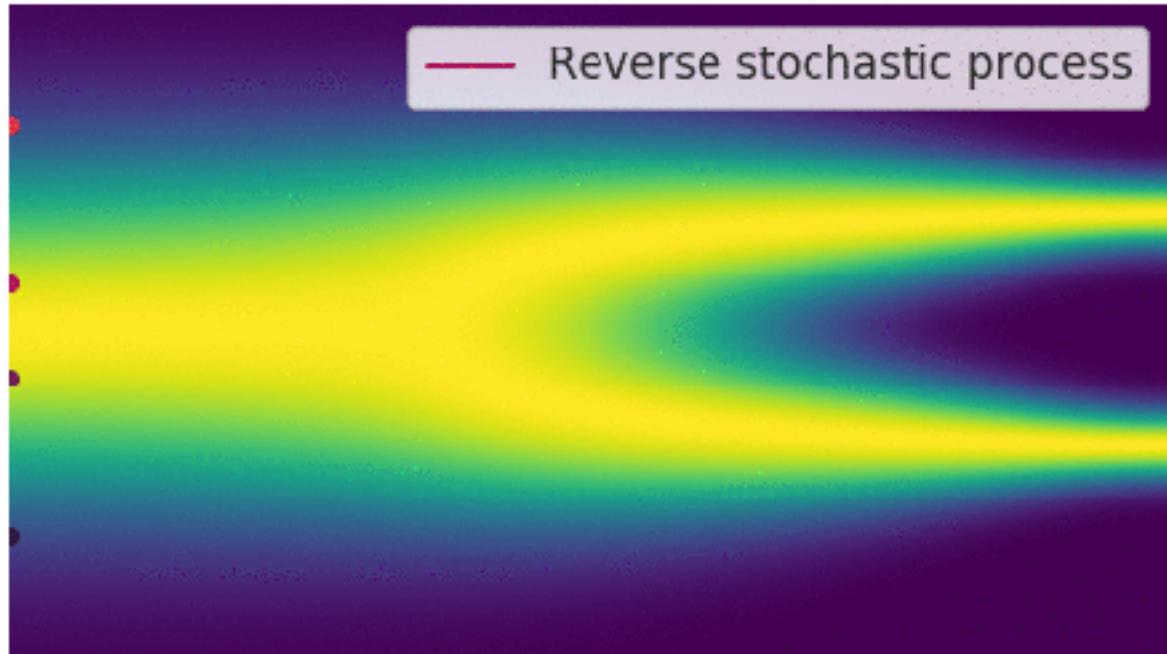
Diffusion models and SDE



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

drift coefficient diffusion coefficient Brownian motion

Sample generation by reversing the SDE

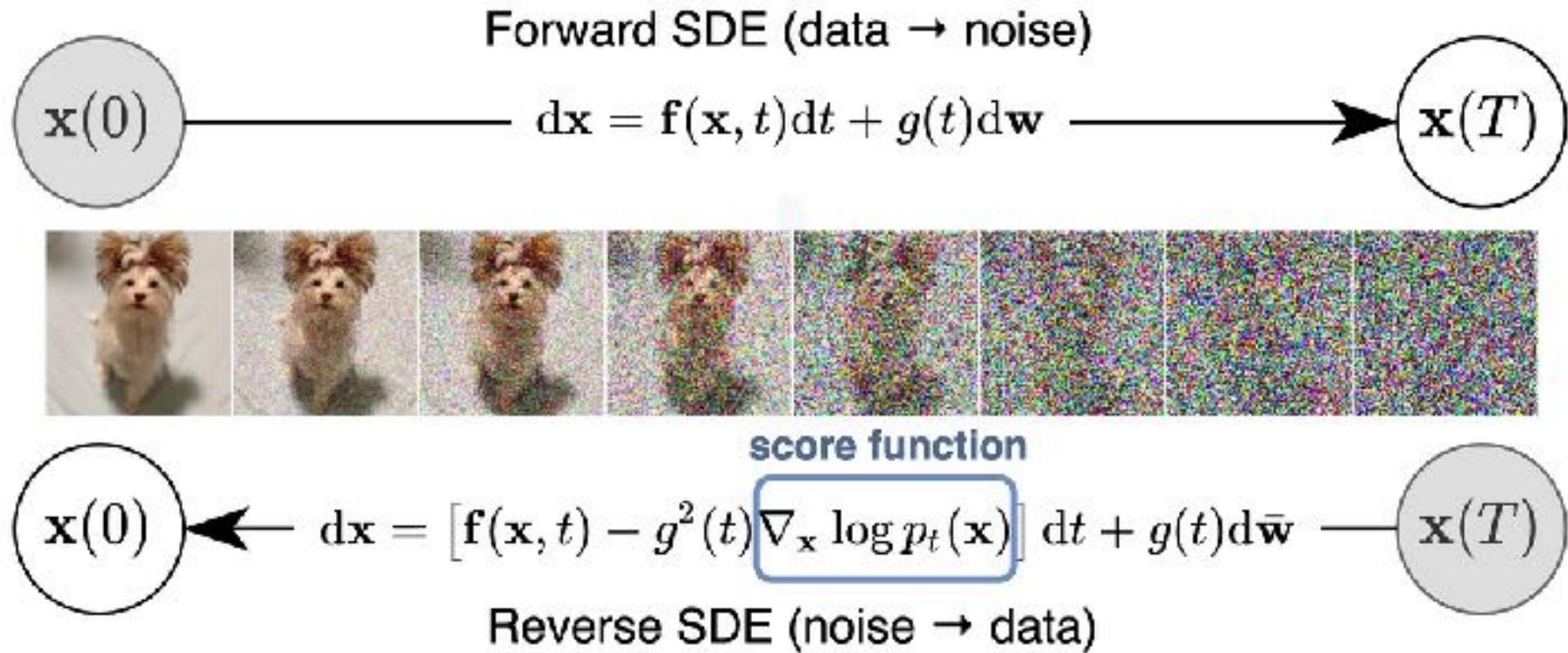


$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t) \nabla_{\mathbf{x}} \log p_t(\mathbf{x})] dt + g(t) d\mathbf{w}$$

|
Score function
(to be estimated)

[Anderson, Reverse-time diffusion equations models, 1982; Song et al., ICLR 2021]

Overview of the SDE-based generative modeling



How to learn the score function?

Direct regression approach:

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

The diagram shows the optimization equation above. A red rectangular stamp with the words "NOT TRACTABLE" in red is placed over the term $\|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, t)\|_2^2$. Below the stamp, blue lines connect the terms to their respective labels: "diffusion time" under $\mathbb{E}_{t \sim \mathcal{U}(0, T)}$, "diffused data" under $\mathbb{E}_{p_t(\mathbf{x})}$, "score of diffused data" under $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, and "neural network" under $\mathbf{s}_\theta(\mathbf{x}, t)$.

Conditioning to initial state:

$$\min_{\theta} \mathbb{E}_{t \sim \mathcal{U}(0, T)} \mathbb{E}_{\mathbf{x}_0 \sim p_0(\mathbf{x})} \mathbb{E}_{\mathbf{x}_t \sim p_t(\mathbf{x}_t | \mathbf{x}_0)} \|\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | \mathbf{x}_0) - \mathbf{s}_\theta(\mathbf{x}, t)\|_2^2$$

The diagram shows the optimization equation above. The term $\mathbb{E}_{\mathbf{x}_t \sim p_t(\mathbf{x}_t | \mathbf{x}_0)}$ is expanded into two separate expectations, each with a blue line connecting it to its label: "initial data sample" under $\mathbb{E}_{\mathbf{x}_0 \sim p_0(\mathbf{x})}$ and "conditioning to initial data sample" under the original $\mathbb{E}_{\mathbf{x}_t \sim p_t(\mathbf{x}_t | \mathbf{x}_0)}$ term.

It can be shown that, after expectations, $\mathbf{s}_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$

Variance preserving SDE

Consider the following discrete Markov chain:

$$\mathbf{x}_i = \sqrt{1 - \beta_i} \mathbf{x}_{i-1} + \sqrt{\beta_i} \mathbf{z}_{i-1}, \mathbf{z}_{i-1} \sim \mathcal{N}(0, I), \beta_i \in [0, 1]$$

It corresponds to the following SDE:

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

It is called the variance preserving SDE.

Variance preserving SDE

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

$$\begin{aligned}\gamma_t &= e^{-\frac{1}{2} \int_0^t \beta(s) ds} \\ \sigma_t^2 &= 1 - e^{-\int_0^t \beta(s) ds}\end{aligned}$$

Re-parametrized sampling: $\mathbf{x}_t = \gamma_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}$ $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Score function: $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | \mathbf{x}_0) = -\nabla_{\mathbf{x}_t} \frac{(\mathbf{x}_t - \gamma_t \mathbf{x}_0)^2}{2\sigma_t^2} = -\frac{\mathbf{x}_t - \gamma_t \mathbf{x}_0}{\sigma_t^2} = -\frac{\boldsymbol{\epsilon}}{\sigma_t}$

Neural network model: $\mathbf{s}_{\theta}(\mathbf{x}_t, t) := -\frac{\epsilon_{\theta}(\mathbf{x}_t, t)}{\sigma_t}$

Overall loss: $\min_{\theta} \mathbb{E}_{t \sim \mathcal{U}(0, T)} \mathbb{E}_{\mathbf{x}_0 \sim p_0(\mathbf{x})} \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \frac{1}{\sigma_t^2} \|\boldsymbol{\epsilon} - \epsilon_{\theta}(\mathbf{x}_t, t)\|_2^2$

Implementation considerations

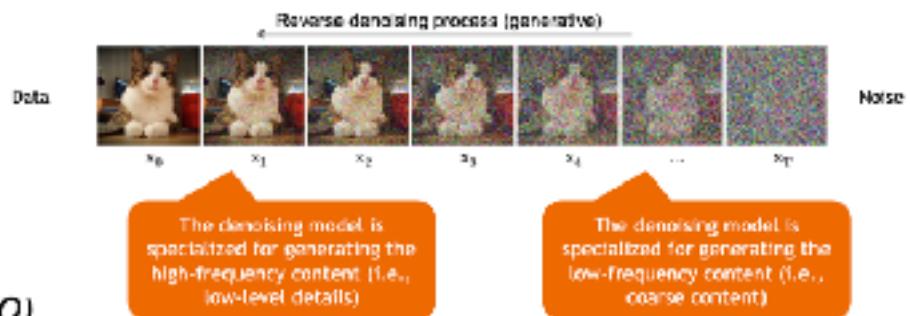
1. Network architecture for score function estimation
2. Noise scheduler
3. Weighting of the loss for different timesteps:

$$\min_{\theta} \mathbb{E}_{t \sim \mathcal{U}(0, T)} \mathbb{E}_{\mathbf{x}_0 \sim p_0(\mathbf{x})} \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \frac{\lambda(t)}{\sigma_t^2} \|\epsilon - \epsilon_{\theta}(\mathbf{x}_t, t)\|_2^2$$

Different loss weightings trade off between model with good perceptual quality vs high log-likelihood:

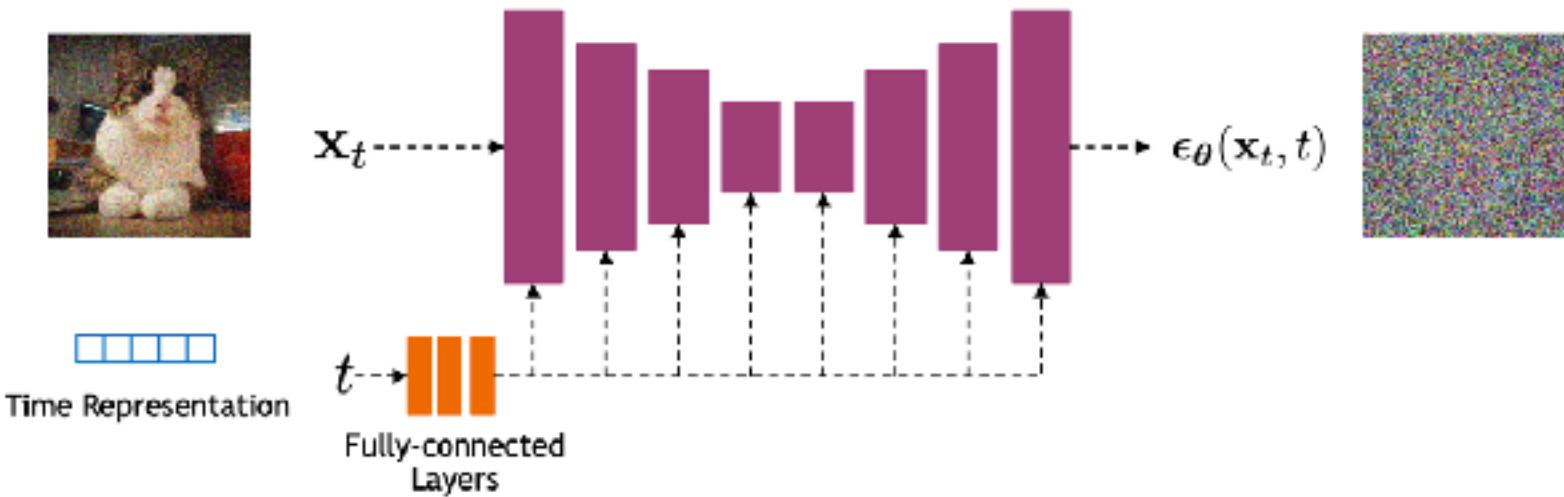
Perceptual quality: $\lambda(t) = \sigma_t^2$

Maximum log-likelihood: $\lambda(t) = \beta(t)$ (*negative ELBO*)



Network architecture

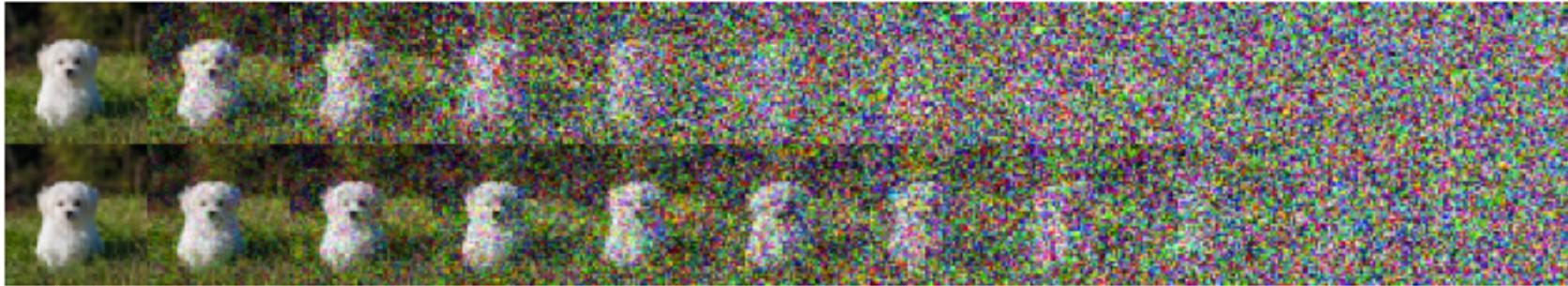
Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$



Time representation: sinusoidal positional embeddings or random Fourier features.

Noise scheduler

Linear

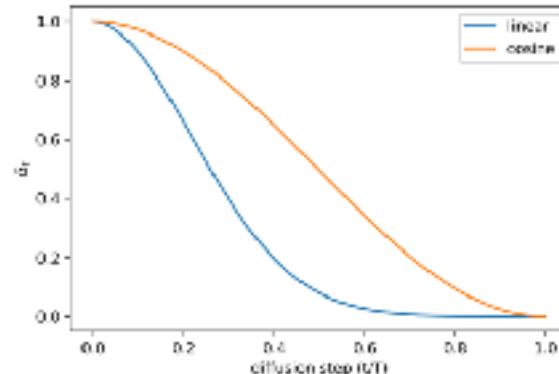


Cosine

Linear schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$

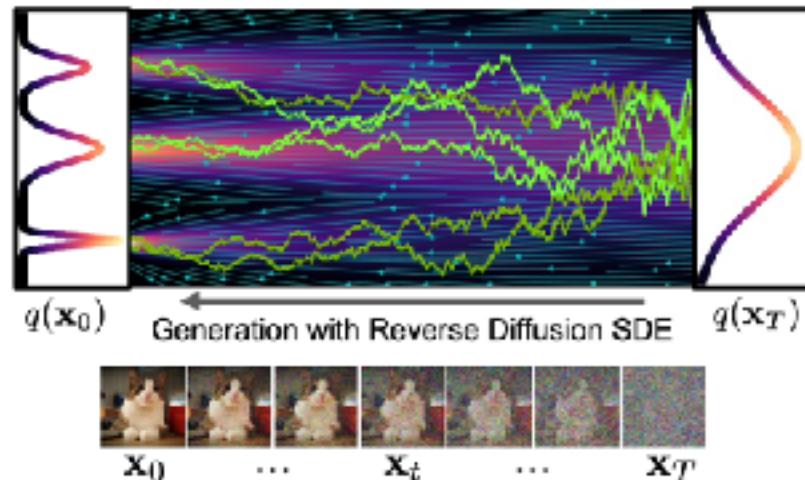
Cosine schedule: $\beta_t = \text{clip}\left(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, 0.999\right)$ $\bar{\alpha}_t = \frac{f(t)}{f(0)}$ where $f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2$

$\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=0}^t \alpha_s$



Sampling data, i.e. how to solve the reverse SDE?

Numerical solvers provide approximate trajectories from SDE.



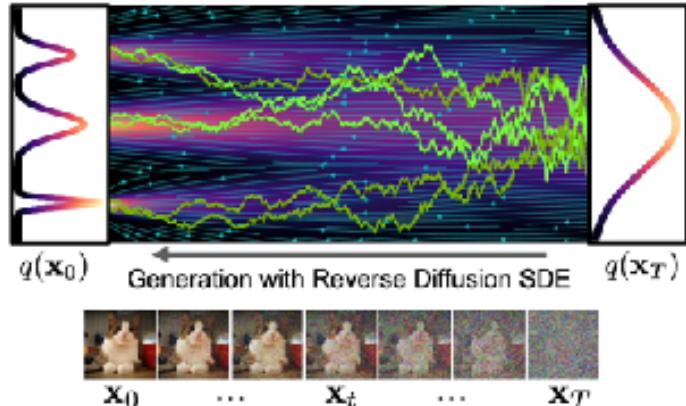
Generative Diffusion SDE:

$$d\mathbf{x}_t = -\frac{1}{2}\beta(t) [\mathbf{x}_t + 2\mathbf{s}_\theta(\mathbf{x}_t, t)] dt + \sqrt{\beta(t)} d\bar{\omega}_t$$

→ Euler-Maruyama:

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

Sampling: SDE vs ODE

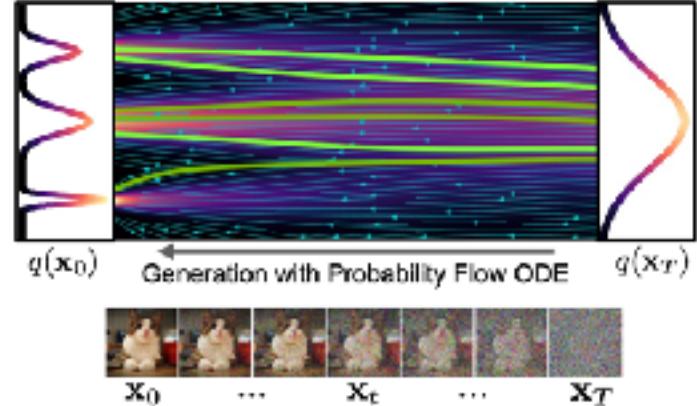


Generative Diffusion SDE:

$$d\mathbf{x}_t = -\frac{1}{2}\beta(t) [\mathbf{x}_t + 2\mathbf{s}_\theta(\mathbf{x}_t, t)] dt + \sqrt{\beta(t)} d\bar{\omega}_t$$

→ **Euler-Maruyama:**

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



Probability Flow ODE:

$$d\mathbf{x}_t = -\frac{1}{2}\beta(t) [\mathbf{x}_t + \mathbf{s}_\theta(\mathbf{x}_t, t)] dt$$

→ **Euler's Method:**

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \frac{1}{2}\beta(t) [\mathbf{x}_t + \mathbf{s}_\theta(\mathbf{x}_t, t)] \Delta t$$

What about denoising diffusion probabilistic models (DDPM)?

DDPM proposed by Ho et al. in 2020 is equivalent to variance preserving SDE with ancestral sampling.

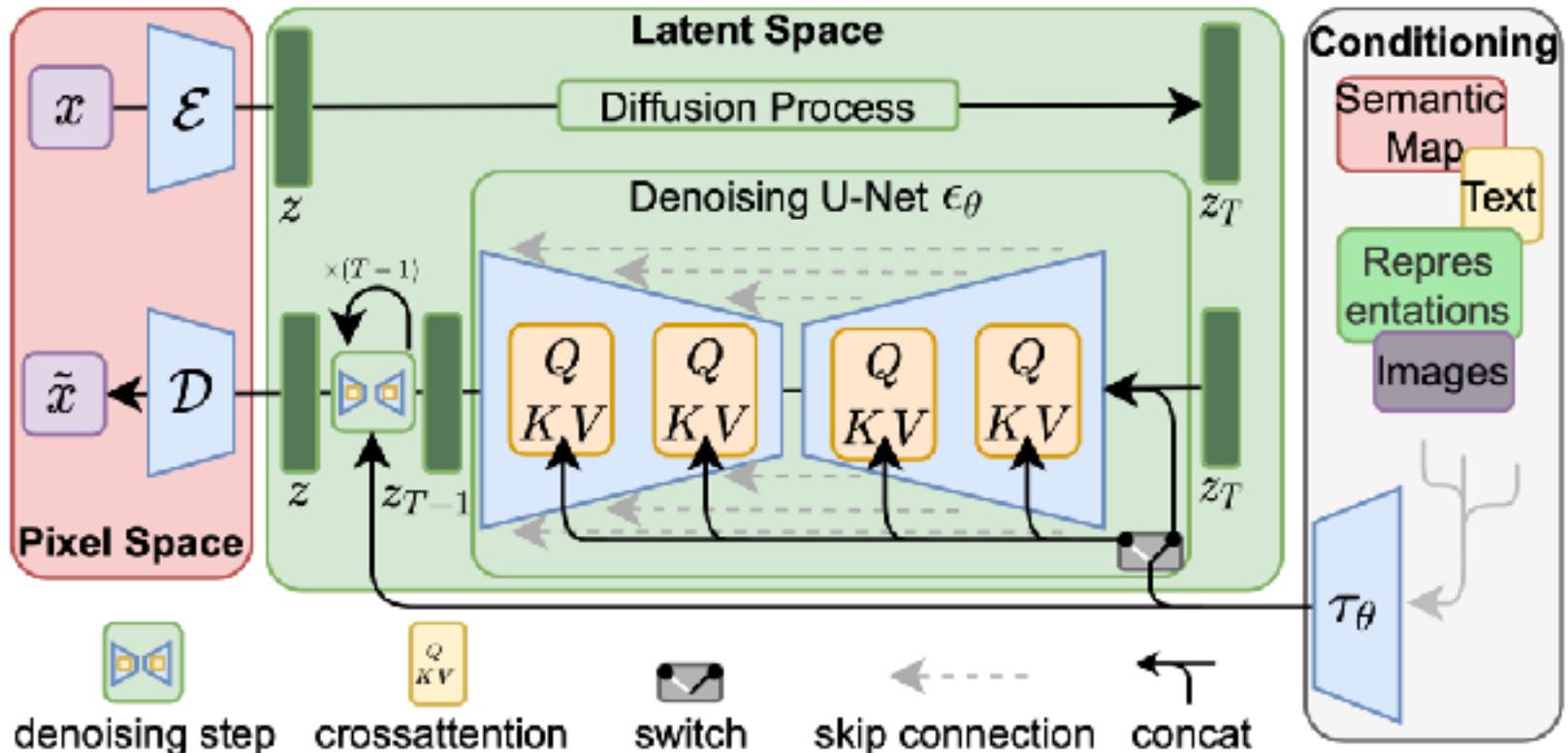
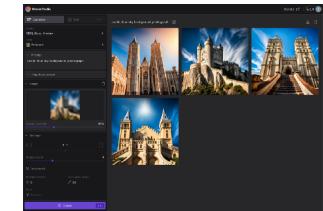
Algorithm 1 Training

```
1: repeat
2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 
3:    $t \sim \text{Uniform}(\{1, \dots, T\})$ 
4:    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:   Take gradient descent step on
       $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \alpha_t} \epsilon, t)\|^2$ 
6: until converged
```

Algorithm 2 Sampling

```
1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  do
3:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = \mathbf{0}$ 
4:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 
```

What about stable diffusion?



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

A not exhaustive list of key works

- Anderson, Reverse-time diffusion equation models, 1982
- Vincent, A connection between score matching and denoising auto encoders, Neural computation, 2011
- Sohl-Dickstein et al., Deep unsupervised learning using non equilibrium thermodynamics, ICML, 2015
- Ho et al. Denoising diffusion probabilistic models, 2020
- Song et al. Score-based generative modeling through stochastic differential equations, ICLR 2021
- Rombach et al. High-resolution image synthesis with latent diffusion models, CVPR 2022
- Yang et al., Diffusion models: a comprehensive survey of methods and applications, 2022

Take home messages

Diffusion models are very powerful and very easy to use

SDE formulation prodiges a clear mathematical framework

Connections to VAEs, neural ODEs, continuous normalizing flows

Key elements for implementations: denoising network (Unet) and a scheduler

Long to train and « slow » to sample

What about other data types? (meshes, text, etc.)