Diffusion Models

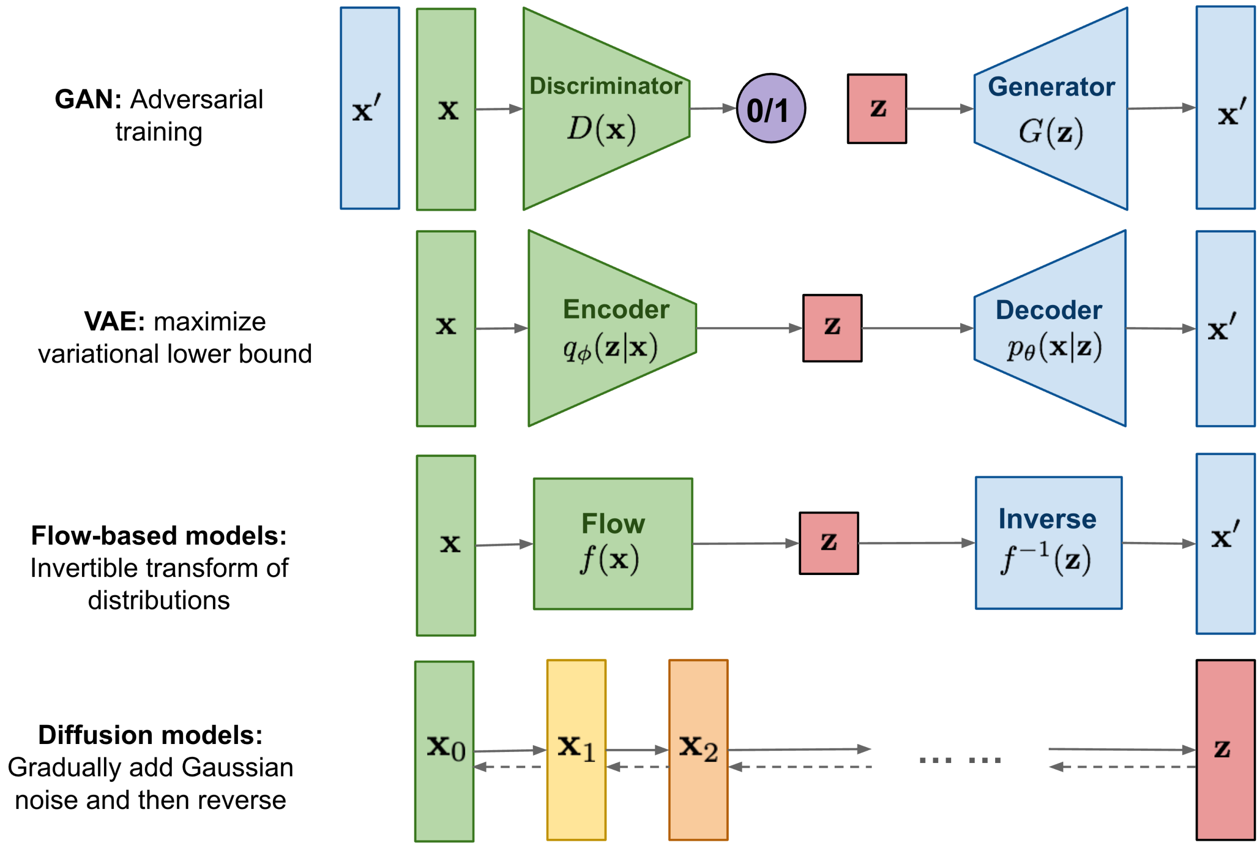
DL4DS – Spring 2025

Based on Rocca, 2022, "Understanding Diffusion Probabilistic Models (DPMs)", Towards Data Science and other references…

Outline

- Contextualizing Diffusion Models
- Theory behind diffusion models
- Architectures and Training
- Applications

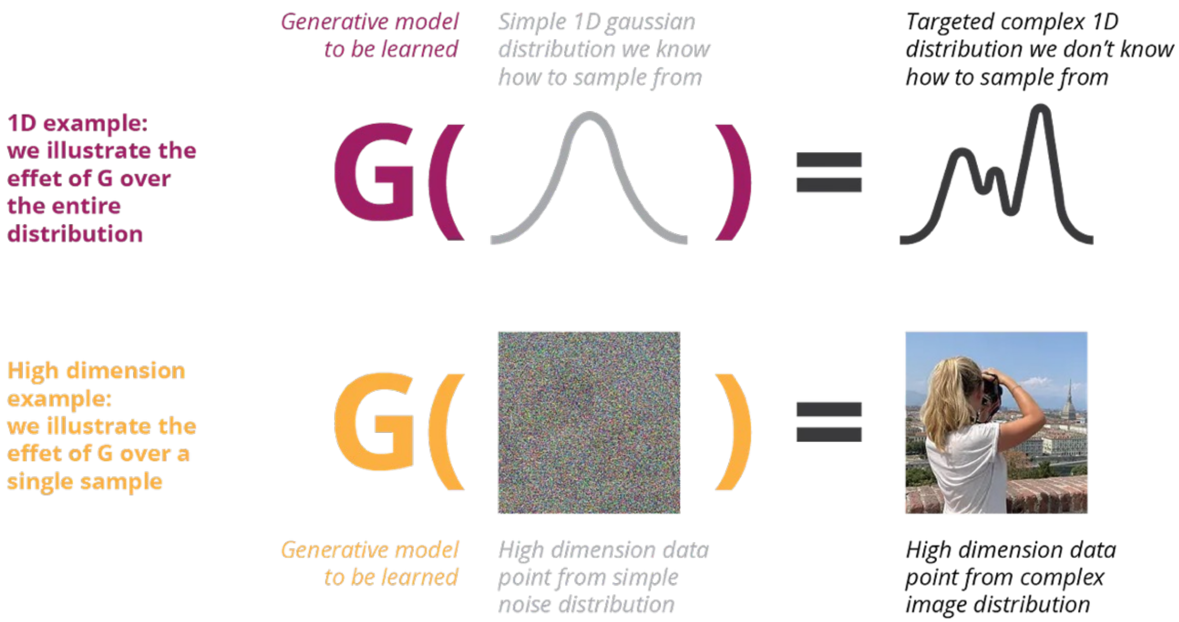
Different Generative Models



L. Weng, "What are Diffusion Models?", blog post, 2021

Given a probability distribution only described by some available samples, how can one generate a new sample?

Introduction



Generative models aims at learning a function that takes data from a simple distribution and transform it into data from a complex distribution.

Retrospective

- 2013: Kingma and Welling introduce Variational AutoEncoder.
- 2014: Goodfellow et al introduced Generative Adversarial Networks (GANs).
- 2015: Sohl-Dickstein "Deep Unsupervised Learning using Nonequilibrium Thermodynamics"
- 2020 (DDPM): J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models"
- 2022 (DDIM): J. Song, C. Meng, and S. Ermon, "Denoising Diffusion Implicit Models" – more efficient
- 2022 (Stable Diffusion): R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models"
- 2023 (Diffusion Transformer): W. Peebles and S. Xie, "Scalable Diffusion Models with Transformers"

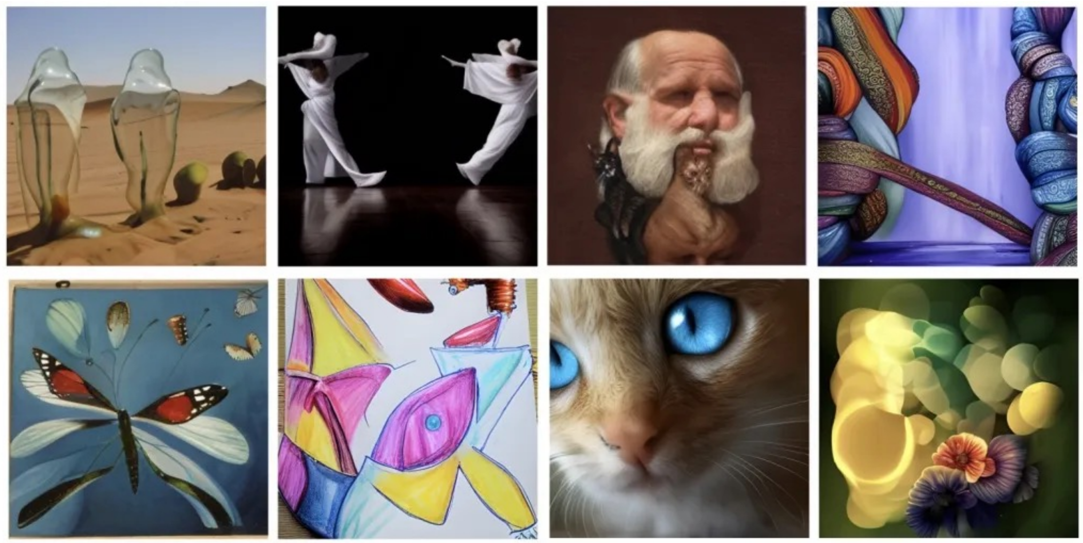
UDL Chapter 18 – Diffusion Models

- 18.2 Encoder (forward process)
  - Defines the noise diffusion process
- 18.3 Decoder model (reverse process)
  - Derives the denoising steps based on neural networks
- 18.4 Training
  - Derives a lower bound (ELBO) on the loss function
  - Initial diffusion loss function
- 18.5 Reparameterization of loss function
  - Shown to work better empirically
- 18.6 Implementaiton
  - Training and sampling algorithms

Commercial Diffusion Solutions

- Dall-E 2 and 3 from OpenAI
- Imagen from Google
- Make-A-Scene from Meta
- Imagen Video from Google
- Make-A-Video from Meta
- Stable Diffusion 3 from stability.ai
- Model Version 6 from midjourney.com

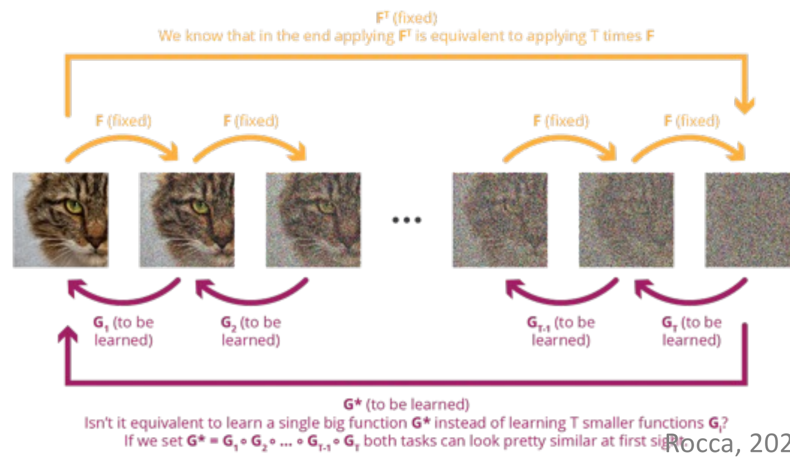
Examples



Examples above have been generated by Meta Make-A-Scene model, that generates images from both a text prompt and a basic sketch for greater level of creative control.

Basic idea of Diffusion Probabilistic Models

- learn the *reverse process* of
- a well defined *stochastic forward process* that progressively destroys information, taking data from our complex target distribution and bringing them to a simple gaussian distribution.
- *reverse process* is then expected to take the path in the opposite direction, taking gaussian noise as an input and generating data from the distribution of interest.



Rocca, 2022

Outline

First:

- Stochastic Process
- Diffusion Process

Then intuition behind DPMs

Then some math basis

Then how trained in practice

Markov stochastic process

Stochastic Processes

- Discrete: $X_n, \quad \forall\, n \in \mathbb{N}$
- Continuous: $X_t, \quad \forall t \geq 0$

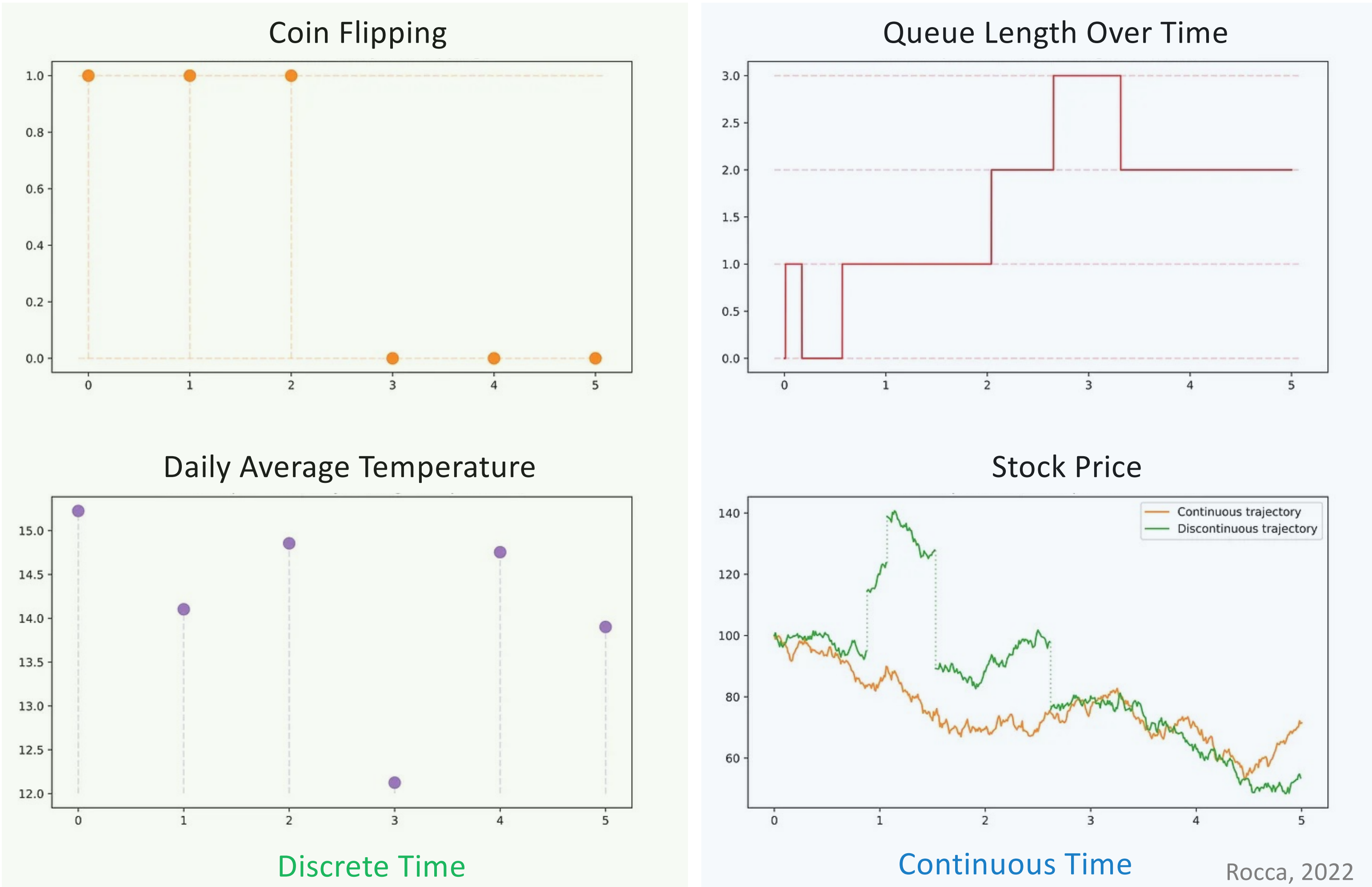
Realization of a random variable → sample

Realization of a stochastic process → sample path or trajectory

Different types of stochastic processes

Discrete
Value

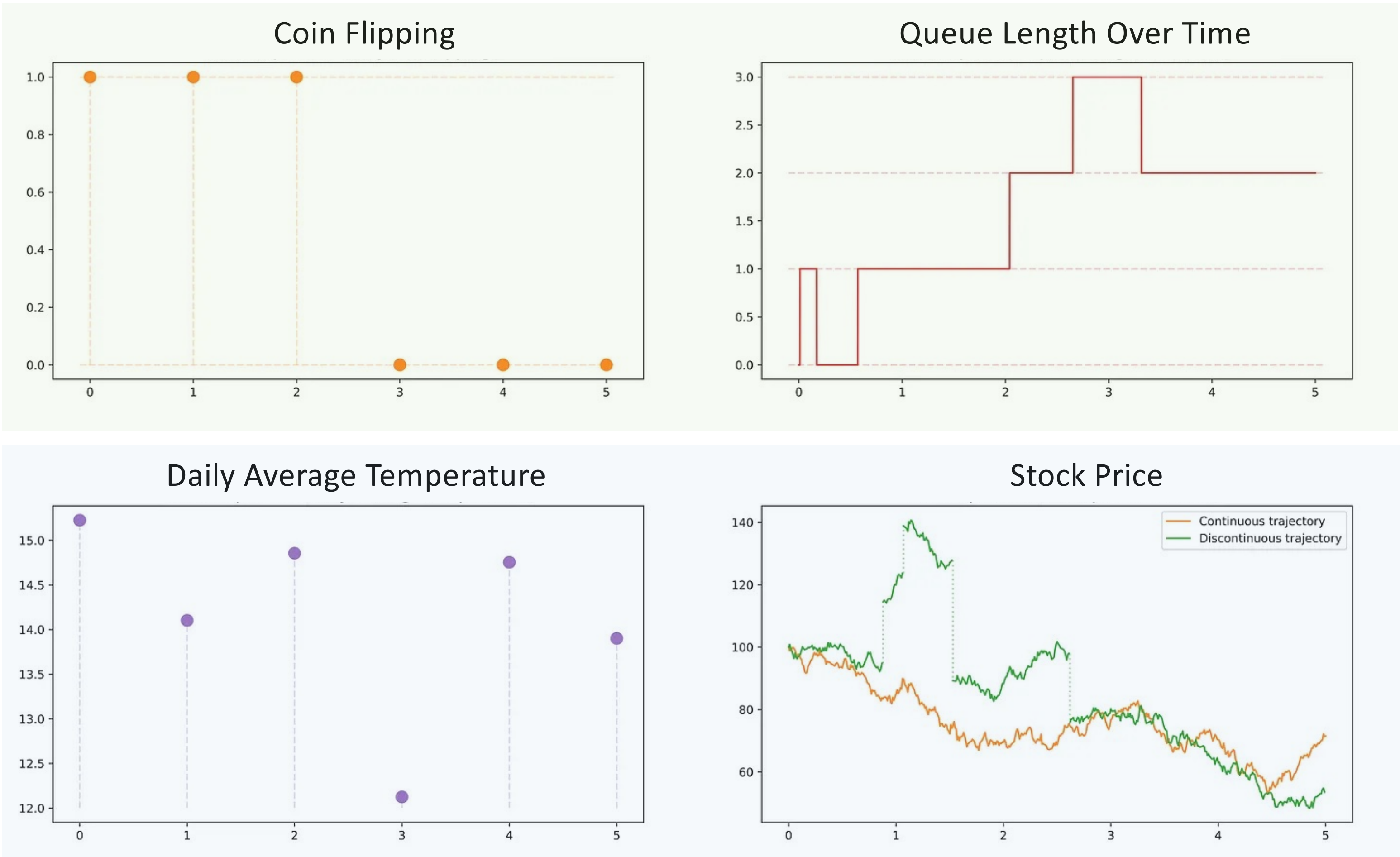
Continuous
Value



Different types of stochastic processes

Discrete Value

Continuous Value



Discrete Time

Continuous Time

Rocca, 2022

Markov (Stochastic) Process

A *Markov process* is a *stochastic process* with no memory.

Future behavior only depends on the present, or

Present only depends on the previous sample.

$$P\left(X_{t_n} \middle| X_{t_{n-1}}, \dots, X_{t_0}\right) = P\left(X_{t_n} \middle| X_{t_{n-1}}\right) \quad \forall t_0 < t_1 < \dots < t_{n-1} < t_n$$

Diffusion Process

Any *diffusion process* can be described by a *stochastic differential equation* (SDE)

$$dX_t = a(X_t, t)dt + \sigma(X_t, t)dW_t$$

where:

$a(\cdot)$ is called the drift coefficient
$\sigma(\cdot)$ is called the diffusion coefficient
W is the Wiener process

Both a and σ are a function of the value and time

Diffusion Process

Any *diffusion process* can be described by a *stochastic differential equation* (SDE)

$$\underbrace{dX_t = a(X_t, t)dt}_{\text{Simple differential equation}} + \underbrace{\sigma(X_t, t)dW_t}_{\text{Stochastic part}}$$

where:

$a(\cdot)$ is called the drift coefficient
$\sigma(\cdot)$ is called the diffusion coefficient
W is the Wiener process

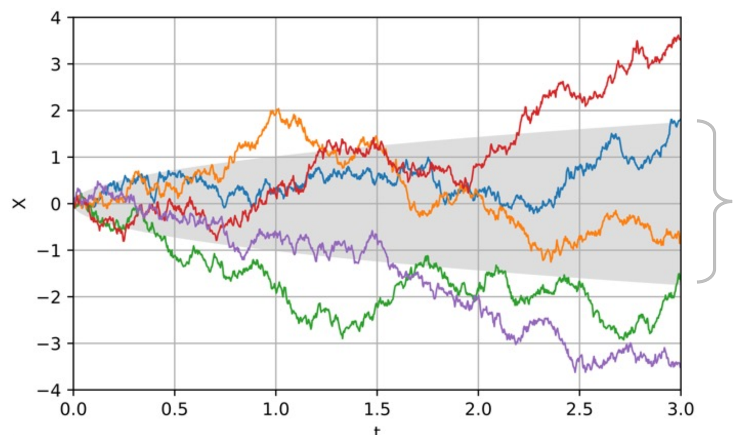
Wiener Process (Brownian Motion)

Continuous time stochastic process

The Wiener process W_t is characterised by the following properties:[2]

1. $W_0 = 0$ almost surely
2. W has independent increments: for every $t > 0$, the future increments $W_{t+u} - W_t$, $u \geq 0$, are independent of the past values W_s, $s < t$.
3. W has Gaussian increments: $W_{t+u} - W_t$ is normally distributed with mean 0 and variance u, $\boxed{W_{t+u} - W_t \sim \mathcal{N}(0, u).}$
4. W has almost surely continuous paths: W_t is almost surely continuous in t.

Five sampled processes



Expected standard deviation

https://en.wikipedia.org/wiki/Wiener_process

Norbert Weiner

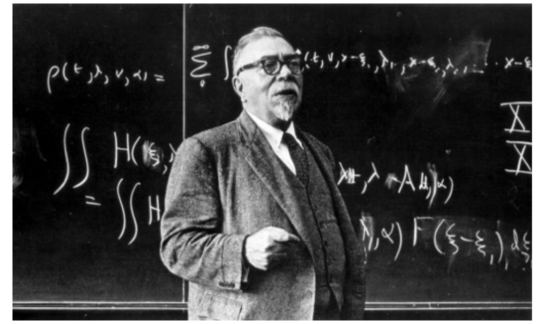
Norbert Wiener (November 26, 1894 – March 18, 1964) was an American computer scientist, mathematician and philosopher. He became a professor of mathematics at the Massachusetts Institute of Technology (MIT).

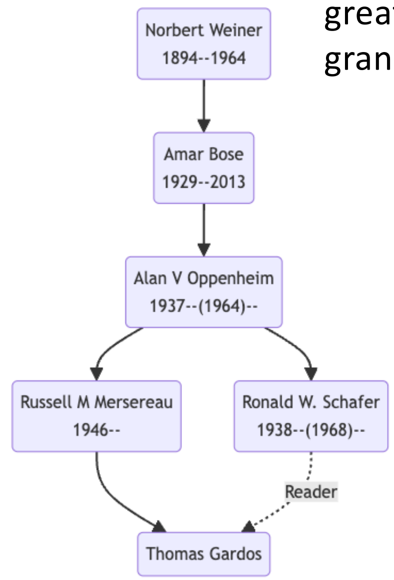
A child prodigy, Wiener later became *an early researcher in stochastic and mathematical noise processes*, contributing work relevant to electronic engineering, electronic communication, and control systems.

Wiener is considered the originator of cybernetics, the science of communication as it relates to living things and machines.

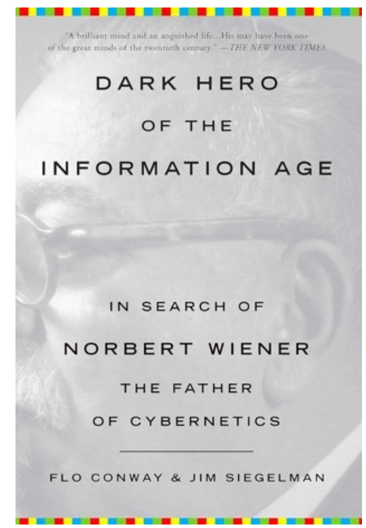
Heavily influenced John von Neumann, Claude Shannon, etc…

Wrote "The Machine Age" in 1949 anticipating robots, etc.





great, great
grand advisor ☺



https://en.wikipedia.org/wiki/Norbert_Wiener
https://www.nytimes.com/2013/05/21/science/mit-scholars-1949-essay-on-machine-age-is-found.html

Discretizing

So

$$dW_t \approx W_{t+dt} - W_t \sim \mathcal{N}(0, \underbrace{dt})$$

Property of Weiner Process: The std is equal to the time step.

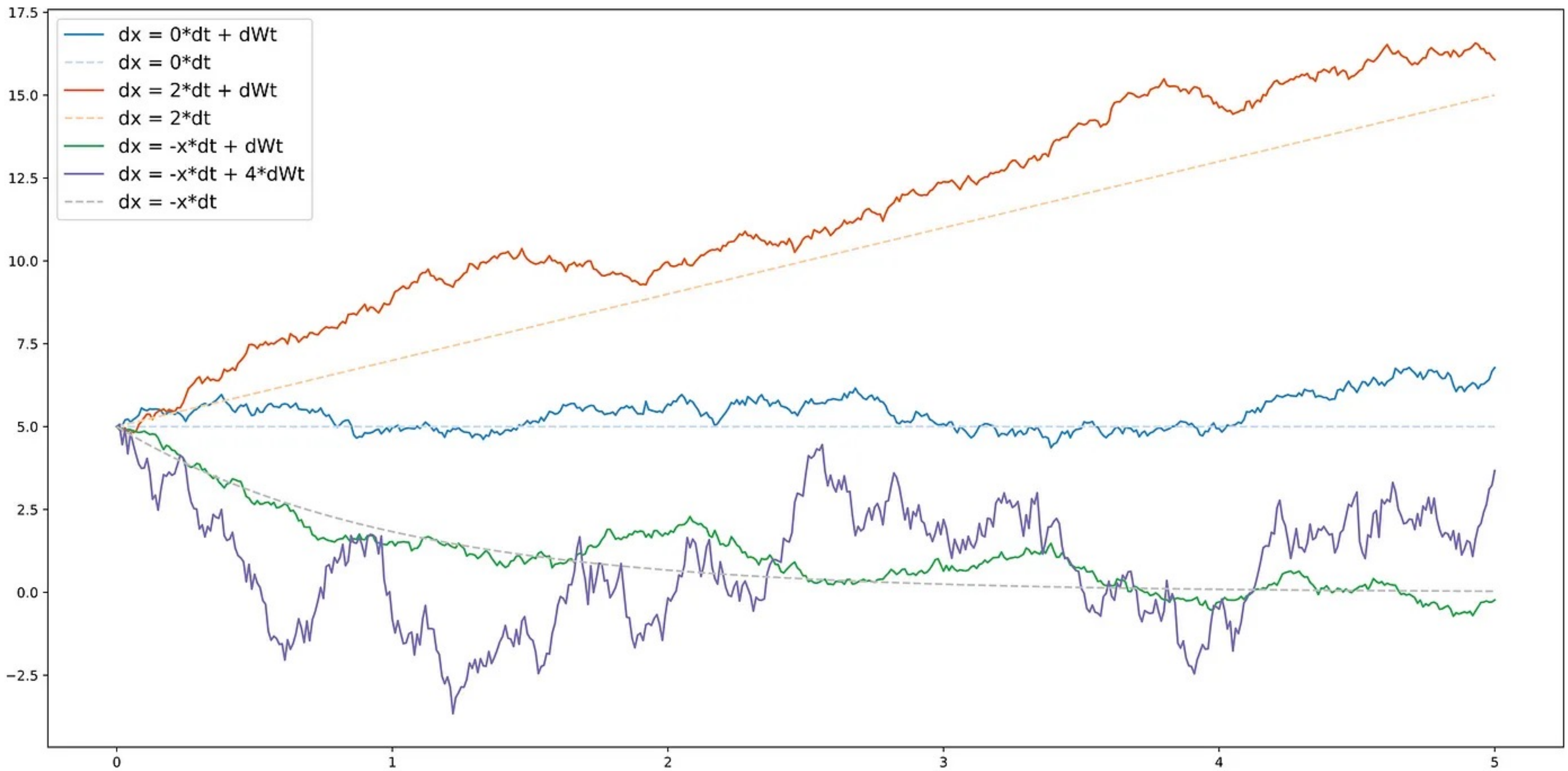
Discretizing the SDE

$$X_{t+dt} - X_t \approx a(X_t, t)dt + \sigma(X_t, t)U \quad \text{where} \quad U \sim \mathcal{N}(0, dt)$$

Which can also be rewritten

$$X_{t+dt} \approx X_t + \underbrace{a(X_t, t)dt}_{\text{Deterministic drift term}} + \underbrace{U'}_{\text{Normal RV with std proportional to diffusion term}} \quad \text{where} \quad U' \sim \mathcal{N}(0, \underbrace{\sigma(X_t, t)}dt)$$

Diffusion process samples



Reversed time process

If X_t is a diffusion process such that

$$dX_t = a(X_t, t)dt + \sigma(t)dW_t$$

then the reversed-time process, $\bar{X}_t = X_{T-t}$ is also a diffusion process

$$d\bar{X}_t = \left[a(\bar{X}_t, t) - \sigma^2(t)\nabla_{X_t} \log p(X_t)\right]dt + \sigma(t)dW_t$$

$$= \bar{a}(\bar{X}_t, t)dt + \sigma(t)dW_t$$

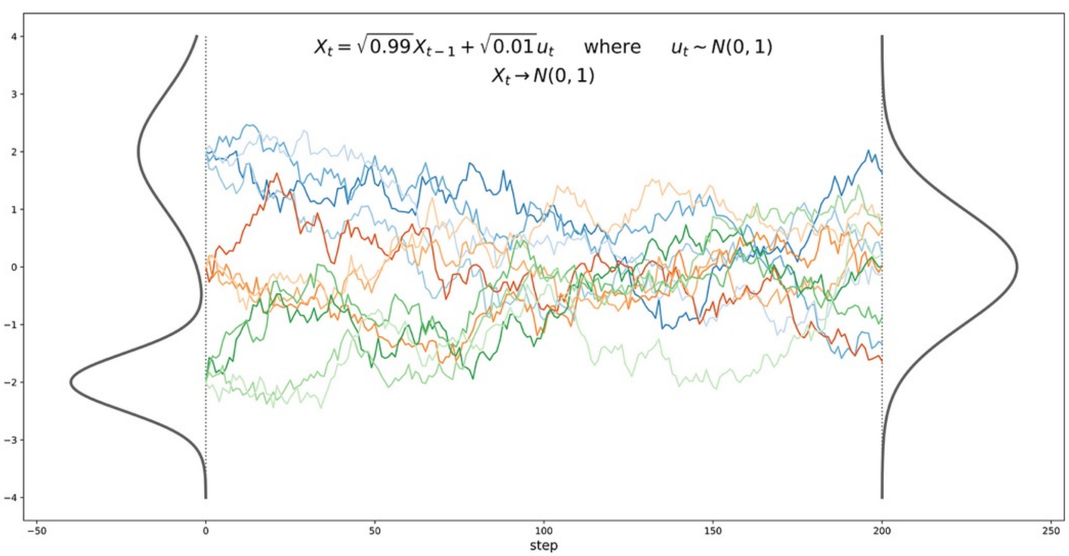
where $\nabla_{X_t} \log p(X_t)$ is called the *score function* and $p(X_t)$ is the *marginal probability* of X_t

Intuition behind diffusion processes

Progressively destroys relevant information

E.g. with shrinking ($|a| < 1$) *drift coefficient* and non-zero *diffusion coefficient* will turn complex distribution into isotropic gaussian

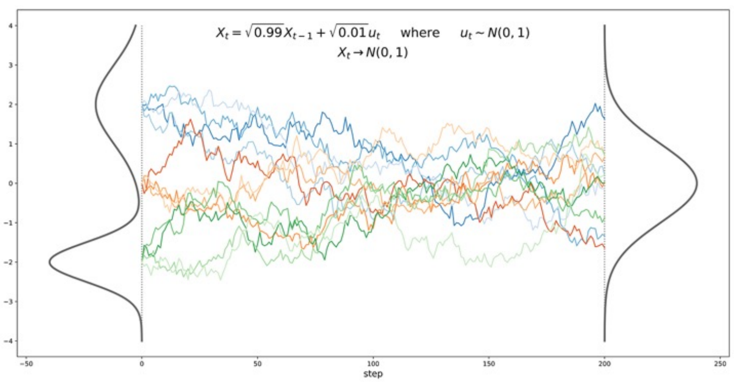
$$X_t = \sqrt{1-p}X_{t-1} + \sqrt{p}u_t \qquad \text{where} \qquad u_t \sim \mathcal{N}(0,1) \quad \text{and} \quad p = 0.01$$



Intuition behind diffusion processes

For the diffusion process

$$X_t = \sqrt{1-p}X_{t-1} + \sqrt{p}u_t \qquad \text{where} \qquad u_t \sim \mathcal{N}(0,1) \quad \text{and} \quad p = 0.01$$



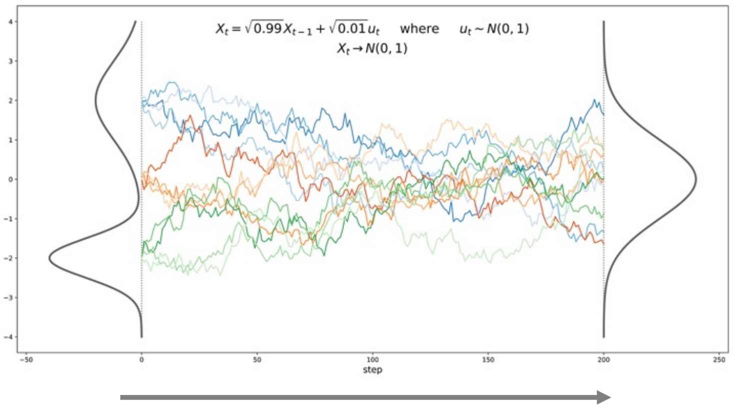
Towards Gaussian

After a given number of steps T we can write

$$\begin{aligned}
X_T &= \sqrt{1-p}X_{T-1} + \sqrt{p}u_T \\
&= (\sqrt{1-p})^2 X_{T-2} + \sqrt{1-p}\sqrt{p}u_{T-1} + \sqrt{p}u_T \\
&= \ldots \\
&= (\sqrt{1-p})^T X_0 + \sum_{i=0}^{T-1} \sqrt{p}(\sqrt{1-p})^i u_{T-i}
\end{aligned}$$

This is a sum of independent gaussians, so can express as single gaussian with variance the sum of the variances.

Intuition behind diffusion processes



Towards Gaussian

After a given number of steps T we can write

$$
\begin{aligned}
X_T &= \sqrt{1-p}X_{T-1} + \sqrt{p}u_T \\
&= (\sqrt{1-p})^2 X_{T-2} + \sqrt{1-p}\sqrt{p}u_{T-1} + \sqrt{p}u_T \\
&= \dots \\
&= (\sqrt{1-p})^T X_0 + \sum_{i=0}^{T-1} \sqrt{p}(\sqrt{1-p})^i u_{T-i}
\end{aligned}
$$

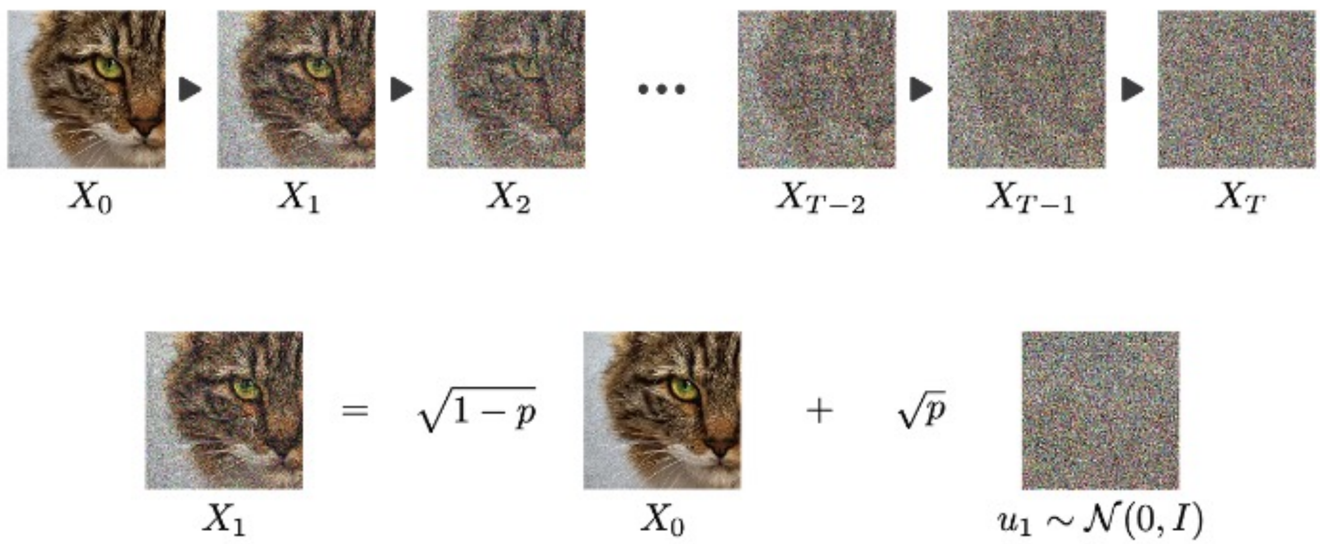
For number of steps T large enough, we have

$$
(\sqrt{1-p})^T \xrightarrow[T\to\infty]{} 0 \qquad \text{and} \qquad \sum_{i=0}^{T-1} \overbrace{\underbrace{\left(\sqrt{p}(\sqrt{1-p})^i\right)^2}_{\text{Variance of the gaussian}}}^{\text{Geometric series}} = p\frac{1-(1-p)^T}{1-(1-p)} \xrightarrow[T\to\infty]{} 1
$$

So for any starting point, we tend to a normal gaussian.

Same idea but for images

But in $H \times W \times C$ dimensions, e.g. $100 \times 100 \times 3$ for 100×100 resolution RGB images



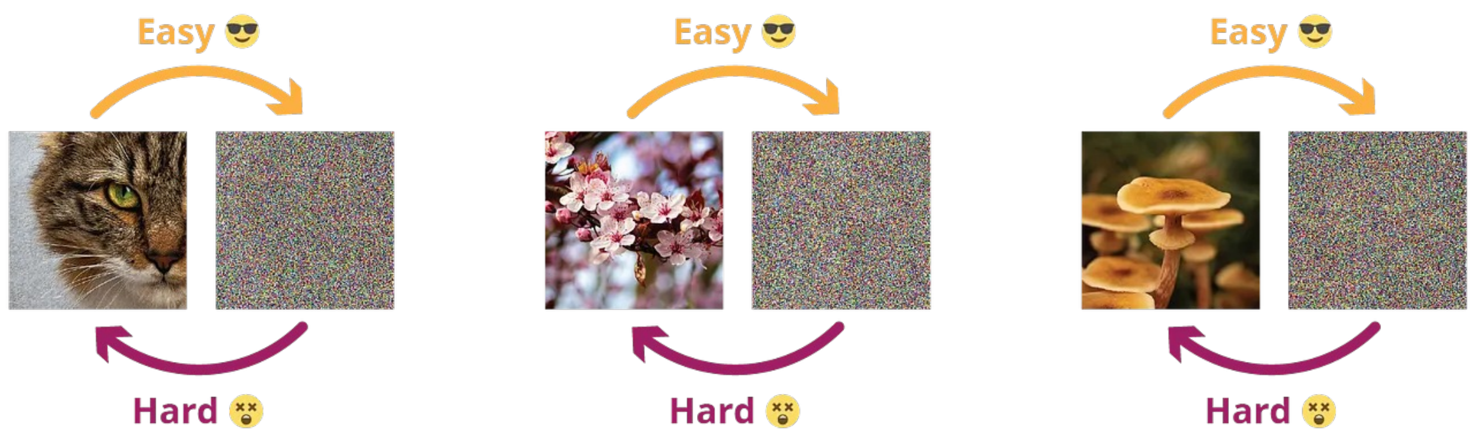
Why use diffusion?

Answer:

Gives us a progressive and structured way to go from a complex distribution to an isotropic gaussian noise

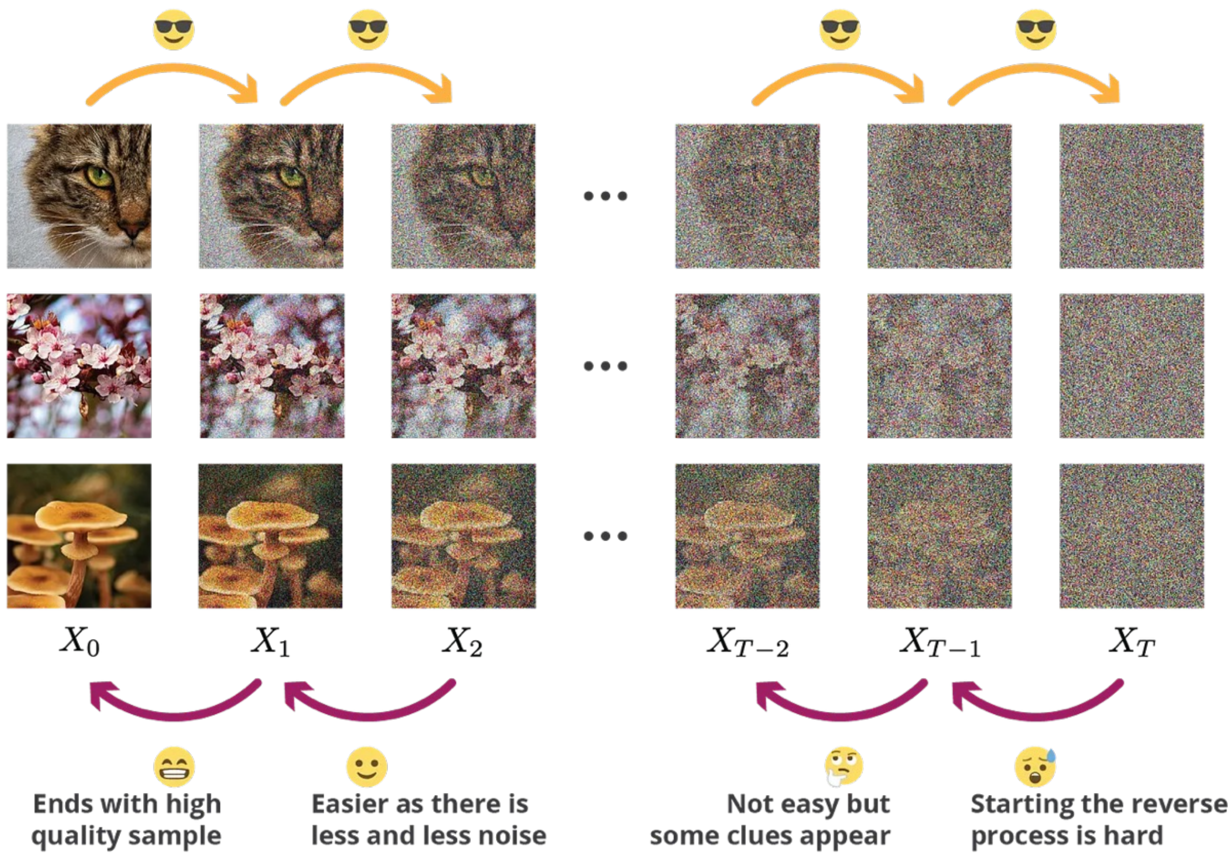
that will enable the learning of the reverse process

Intuition behind learning the reverse process



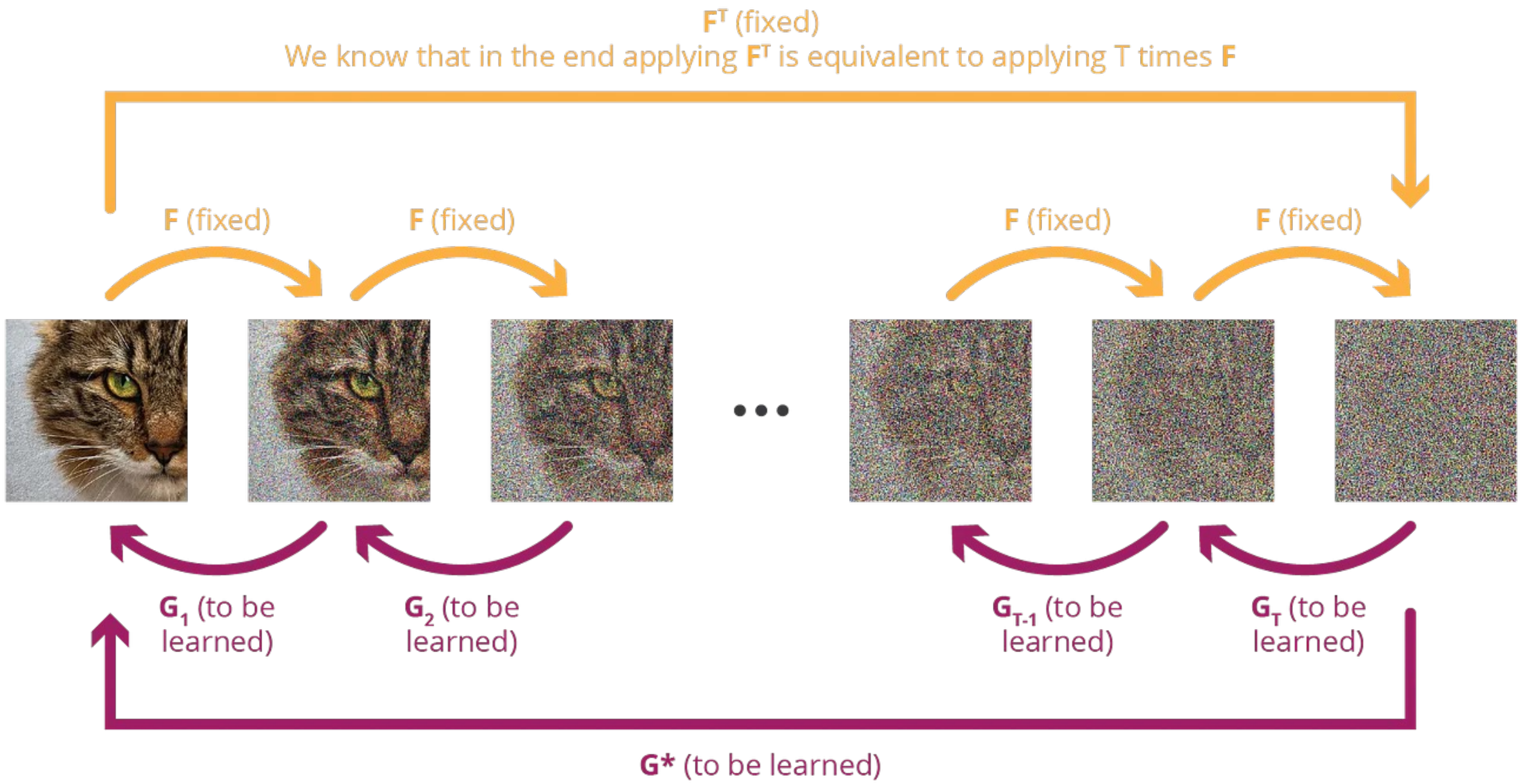
Reversing process in one step is extremely difficult

Doing it in steps gives us some clues



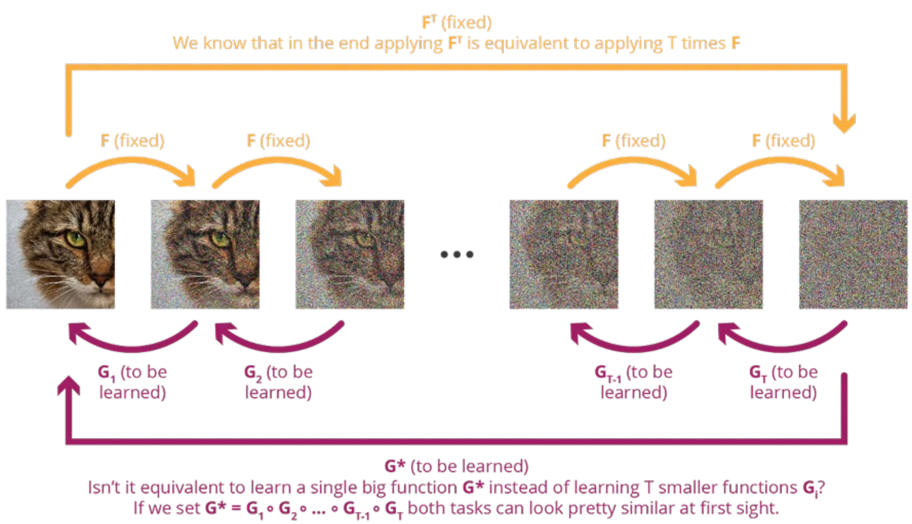
Rocca, 2022

One step versus multi-step



Isn't it equivalent to learn a single big function **G*** instead of learning T smaller functions $\mathbf{G_i}$?
If we set $\mathbf{G^* = G_1 \circ G_2 \circ \ldots \circ G_{T-1} \circ G_T}$ both tasks can look pretty similar at first sight.

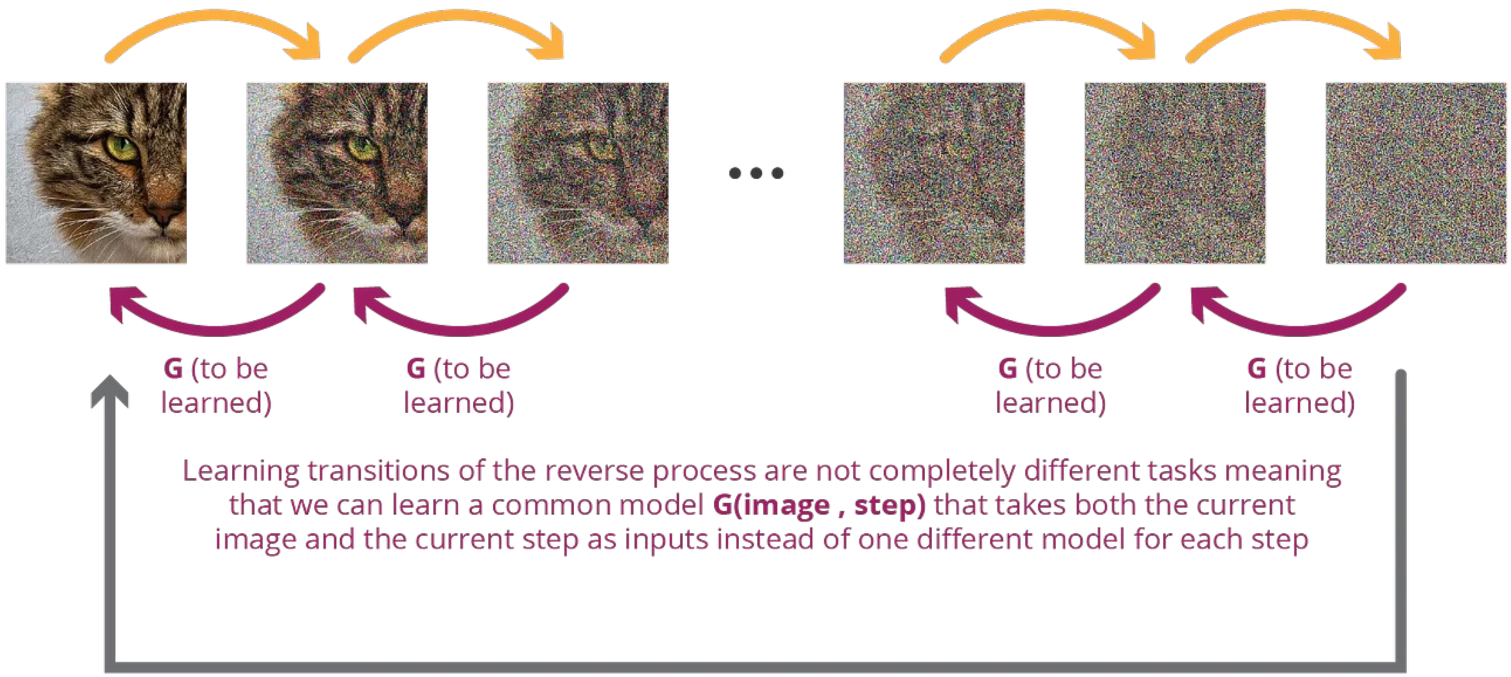
Advantage of multi-step reverse process



1. Don't have to learn a unique transform G_i for each step, but rather a single transform that is a function of the index step. Drastically reduces size of the model.

2. Gradient descent is much more difficult in one step and can exploit coarse to fine adjustments in multiple steps,.

Iterative versus one step



G* (to be learned)

G* can't rely on the same nice iterative structure than G, meaning that this unrolled version supposed to be equivalent to $\mathbf{G_1 \circ G_2 \circ \dots \circ G_{T-1} \circ G_T}$ will have more parameters and will be harder to train

Rocca, 2022

Diffusion Model from Scratch

- HuggingFace Notebook
- https://github.com/huggingface/diffusion-models-class

02_diffusion_models_from_scratch.ipynb

Similarities and differences to VAEs

Similarities:

- An encoder transforms a complex distribution into a simple distribution in a structured way to learn a decoder that produces a similar sample

Differences:

- DPM is multi-step, versus one step for VAE
- DPM encoder is fixed and does not get trained
- DPM will be trained based on the structure of the diffusion process
- DPM latent space is exactly same as input, as opposed to VAE which reduces dimensionality

Rocca, 2022

Dall-E 3

Mathematics of Diffusion Models

Assume the forward and reverse process operate in T steps.

Both forward and reverse process are discrete so becomes a *Markov chain* with *gaussian transition probability*.

Diffusion Process

Any ***diffusion process*** can be described by a ***stochastic differential equation*** (SDE)

$$dX_t = a(X_t, t)dt + \sigma(X_t, t)dW_t$$

where:

- $a(\cdot)$ is called the drift coefficient
- $\sigma(\cdot)$ is called the diffusion coefficient
- W is the Wiener process

Both a and σ are a function of the value and time

16

$\boxed{\mathcal{N}(\mu, \sigma^2)}$

Mathematics of Diffusion Models

Denote x_0 as a sample from a distribution $q(x_0)$.

Forward process: gaussian transition probability

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{(1-\beta_t)}\, x_{t-1}, \beta_t I) \quad \text{where} \quad t \in \mathbb{N}$$

and where β_t indicates trade-off between info to be kept from previous step and new noise added.

$\mathcal{N}(\mu, \sigma^2)$

Mathematics of Diffusion Models

Denote x_0 as a sample from a distribution $q(x_0)$.

Forward process: gaussian transition probability

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{(1-\beta_t)}\, x_{t-1}, \beta_t I) \quad \text{where} \quad t \in \mathbb{N}$$

and where β_t indicates trade-off between info to be kept from previous step and new noise added.

We can equivalently write

$$x_t = \sqrt{(1-\beta_t)}\, x_{t-1} + \sqrt{\beta_t}\, \epsilon_t \qquad \epsilon_t \sim \mathcal{N}(0, I)$$

Discretized diffusion process

Rocca, 2022

$\mathcal{N}(\mu, \sigma^2)$

Mathematics of Diffusion Models

Through recurrence, we can represent any step in the chain as directly represented from x_0:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}\, x_0,\ (1 - \bar{\alpha}_t)I)$$

where

$$\alpha_t = (1 - \beta_t) \quad \text{and} \quad \bar{\alpha}_t = \textstyle\prod_{i=1}^{t} \alpha_i = \prod_{i=1}^{t}(1 - \beta_i)$$

and from the Markov property, the entire forward trajectory is

$$q(x_{0:T}) = q(x_0) \prod_{t=1}^{T} q(x_t|x_{t-1})$$

The reverse process

With the assumption on the drift and diffusion coefficients, the reverse of the diffusion process takes the same form.

Reverse gaussian transition probability

$$q(x_{t-1}|x_t)$$

can then be approximated by

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

where μ_θ and Σ_θ are two functions parameterized by θ and learned.

The reverse process

Using the Markov property, the probability of a given backward trajectory can be approximated by

$$\mathrm{p}_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

where $p(x_T)$ is an isotropic gaussian distribution that does not depend on θ

$$p(x_T) = \mathcal{N}(x_T; 0, I)$$

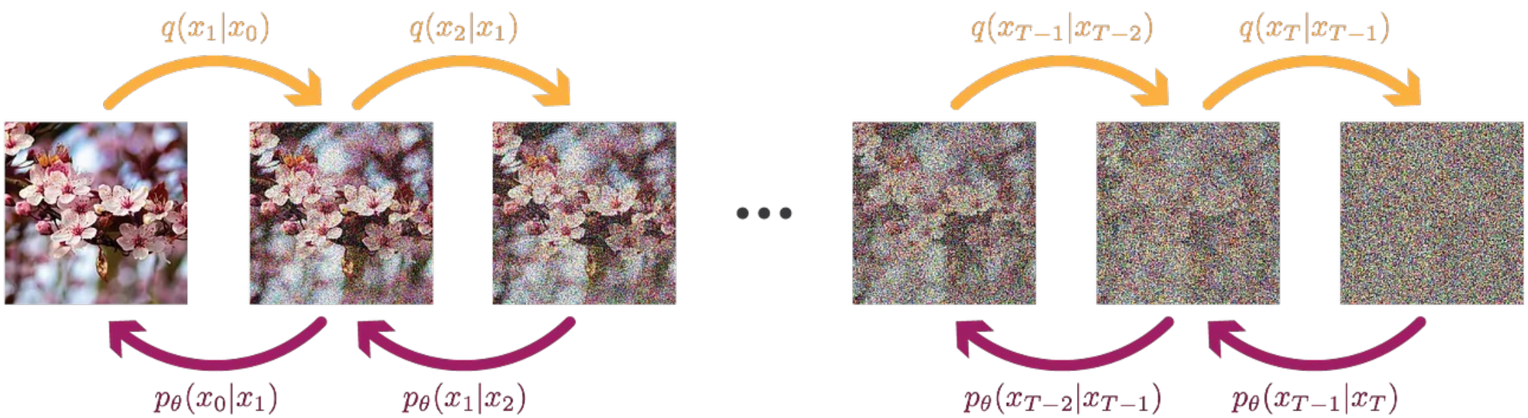
FIXED FORWARD PROCESS

Initial distribution

$q(x_0)$

Gaussian transition kernel

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



Approximation of

$q(x_{t-1}|x_t)$

Gaussian transition kernel with parameters to be learned

$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$

Initial distribution

$p(x_T) = \mathcal{N}(x_t; 0, I)$

LEARNED BACKWARD PROCESS

Questions

How do we learn the parameters θ for μ_θ and Σ_θ?

What is the loss to be optimized?

- We hope that $p_\theta(x_0)$, the distribution of the last step of the reverse process, will be close to $q(x_0)$

Optimization Objective

$$
\begin{aligned}
\mu_\theta^*, \Sigma_\theta^* &= \arg \min_{\mu_\theta, \Sigma_\theta} \left(D_{KL}(q(x_0) || p_\theta(x_0)) \right) \\
&= \arg \min_{\mu_\theta, \Sigma_\theta} \left(- \int q(x_0) \log \left(\frac{p_\theta(x_0)}{q(x_0)} \right) dx_0 \right) \\
&= \arg \min_{\mu_\theta, \Sigma_\theta} \left(- \int q(x_0) \log(p_\theta(x_0)) dx_0 \right)
\end{aligned}
$$

Skipping a lot more math

- Expand p-theta as marginalization integral
- Use Jensen’s inequality to define a slightly simpler upper bound to the loss
- Some manipulations with Bayes’ Theorem
- Properties of KL divergence of two gaussian distributions
- An additional simplification suggested by [Ho et al 2020]

Diffusion models in practice

We have the forward process

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \qquad q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

and our reverse process

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

and we want to train to minimize this simplified upper bound

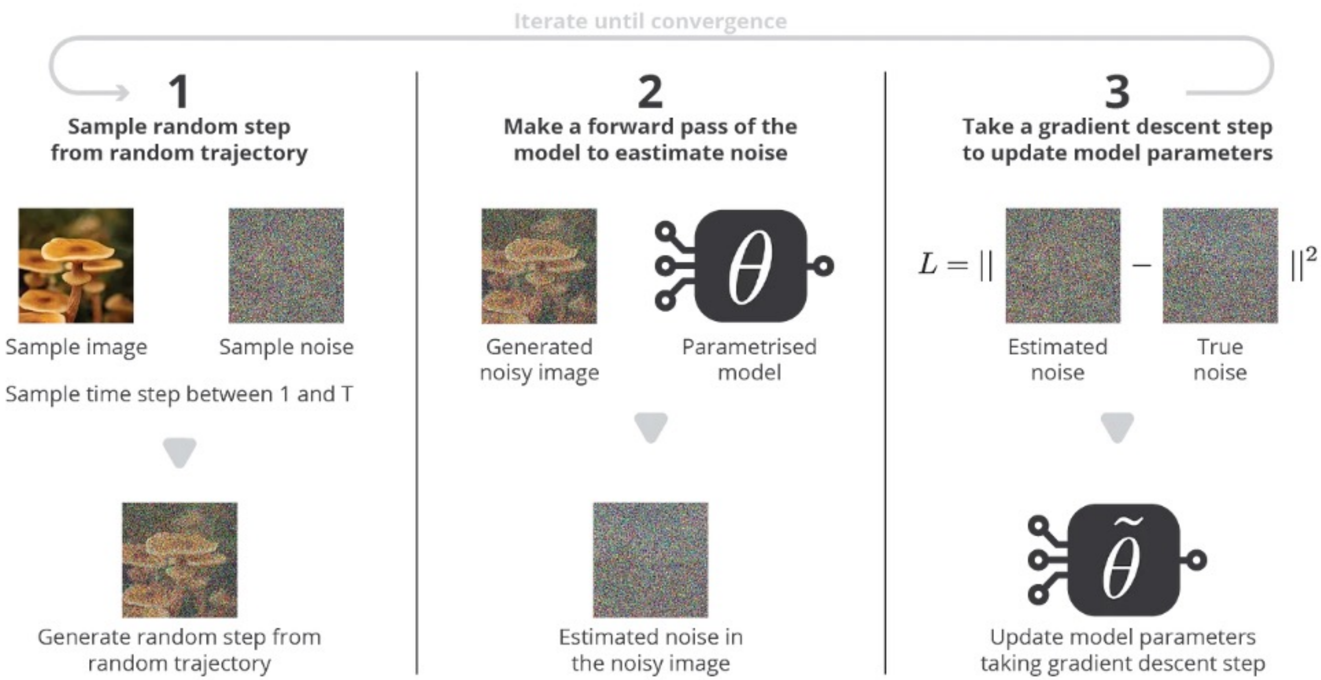
$$\mathbb{E}_{x_0,t,\epsilon}\left(||\epsilon - \epsilon_\theta(x_t, t)||^2\right) = \mathbb{E}_{x_0,t,\epsilon}\left(||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2\right)$$

Dall-E 3

Training Process

Algorithm 1 Training

repeat
  $x_0 \sim q(x_0)$ ▷*Sample random initial data*
  $t \sim \text{Uniform}(\{1, ..., T\})$ ▷*Sample random step*
  $\epsilon \sim \mathcal{N}(0, I)$ ▷*Sample random noise*
  $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$ ▷*Rand. step of rand. trajectory*
  Take gradient descent step on $\nabla_\theta ||\epsilon - \epsilon_\theta(x_t, t)||^2$ ▷*Optimisation*
until converged



To sample/generate

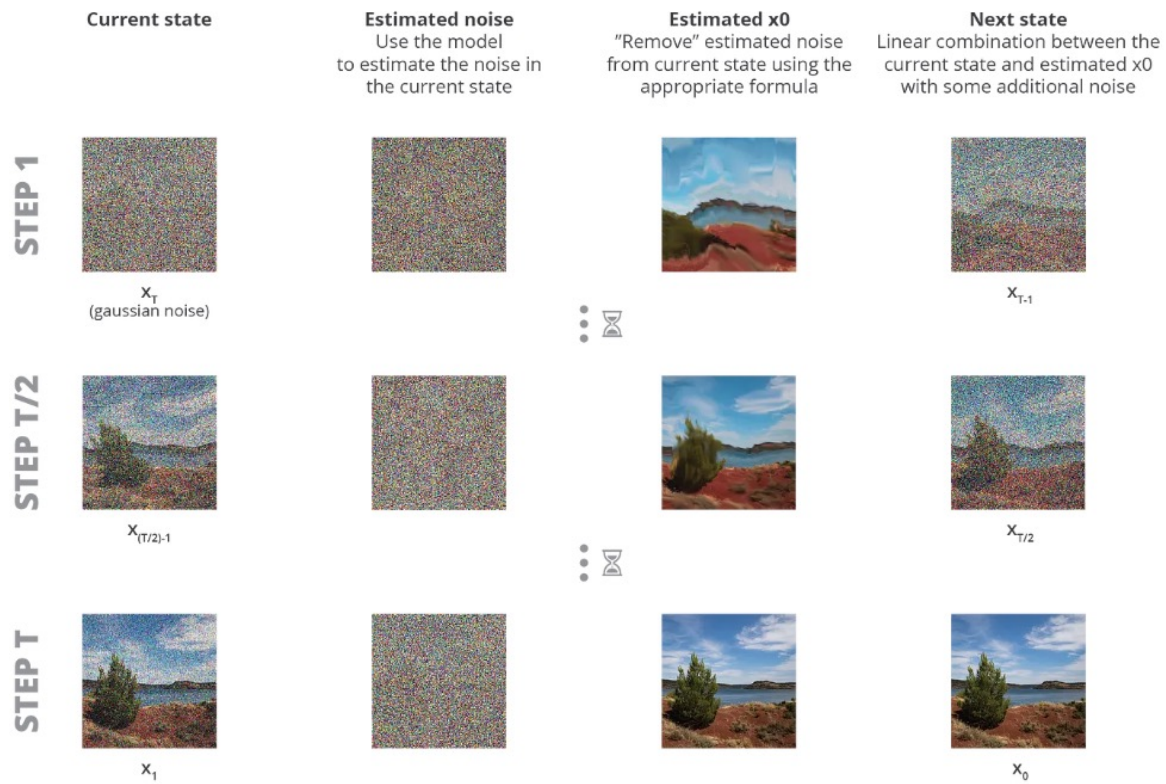


Illustration of the sampling process of a denoising diffusion probabilistic model.

Rocca, 2022

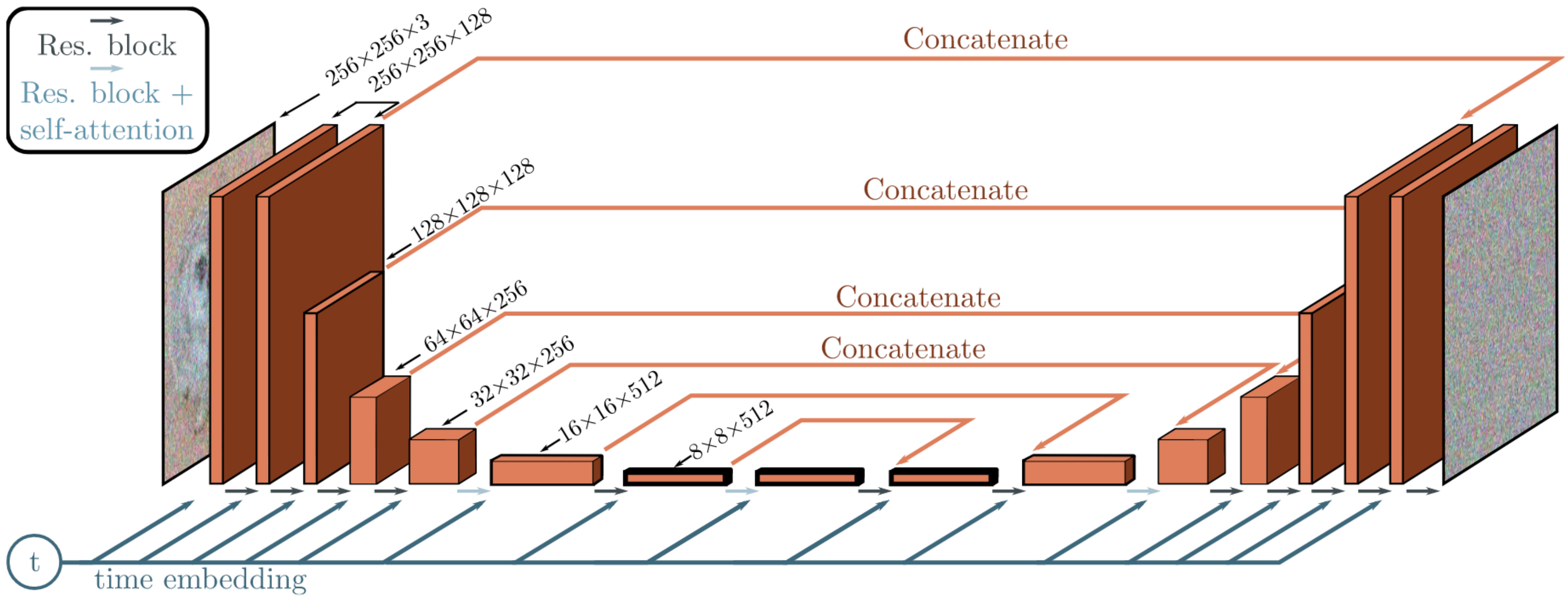
To sample/generate

Algorithm 2 Sampling

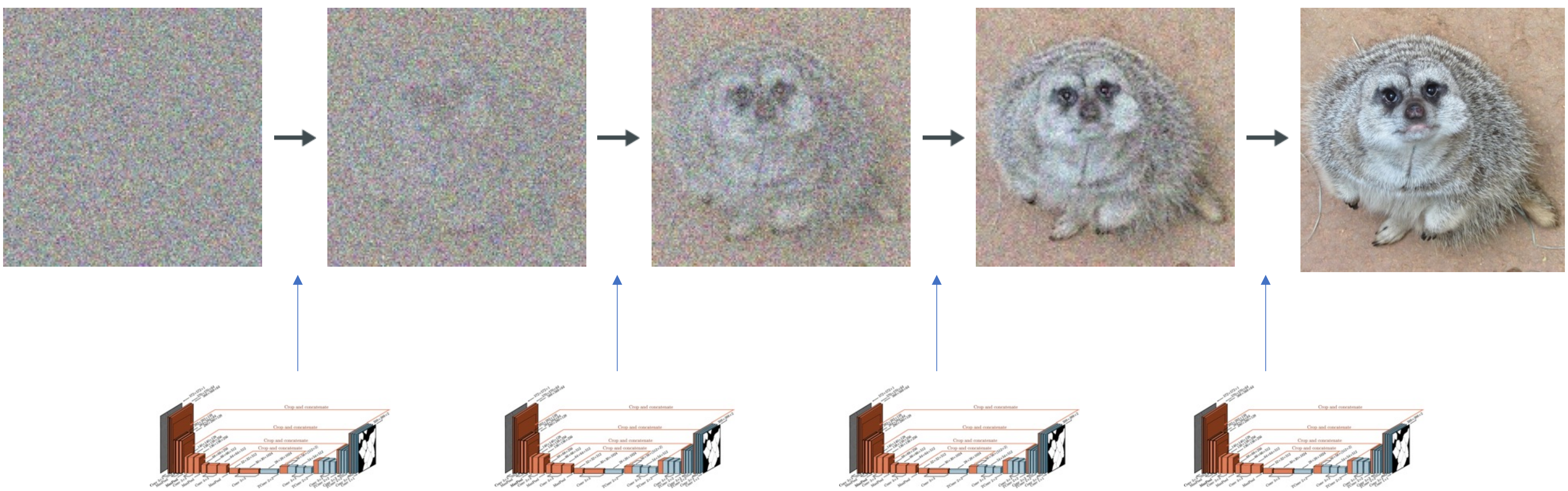
$x_T \sim \mathcal{N}(0, I)$ ▷*Initial isotropic gaussian noise sampling*
for t = T, ..., 1 **do**
  $z \sim \mathcal{N}(0, I)$ if $t > 1$ else $z = 0$ ▷*Sample random noise (if not last step)*
  $\tilde{\epsilon} = \epsilon_\theta(x_t, t)$ ▷*Estimated noise in current noisy data*
  $\tilde{x_0} = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1 - \bar{\alpha}_t}\tilde{\epsilon})$ ▷*Estimated x_0 from estimated noise*
  $\tilde{\mu} = \mu_t(x_t, \tilde{x_0}) \left(= \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)\right)$ ▷*Mean for previous step sampling*
  $x_{t-1} = \tilde{\mu} + \sigma_t z$ ▷*Previous step sampling*
end for
return x_0

U-Net (2016) as the Model

Output is same size as the input.



U-Net for reverse diffusion



Conditional generation

Classifier Guidance

- Modify the denoising update from z_t to z_{t-1} to incorporate class information, c.

Guidance from text

- Condition on a sentence embedding computed from a language model

Conditional generation using classifier guidance

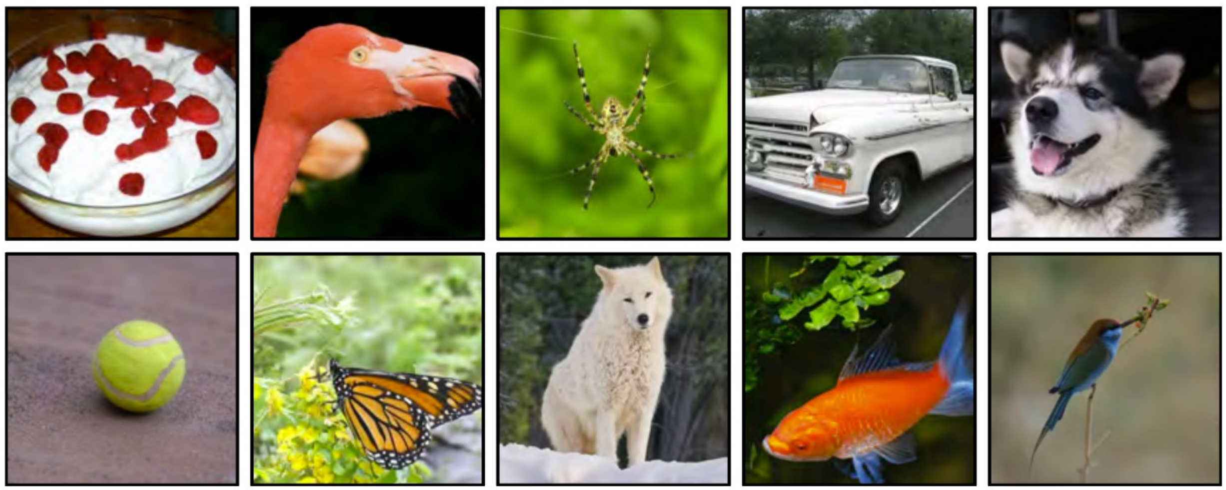


Figure 18.12 Conditional generation using classifier guidance. Image samples conditioned on different ImageNet classes. The same model produces high quality samples of highly varied image classes. Adapted from Dhariwal & Nichol (2021).

Dhariwal and Nichol, "Diffusion Models Beat GANs on Image Synthesis." 2021.

Cascaded conditional generation based on a text prompt

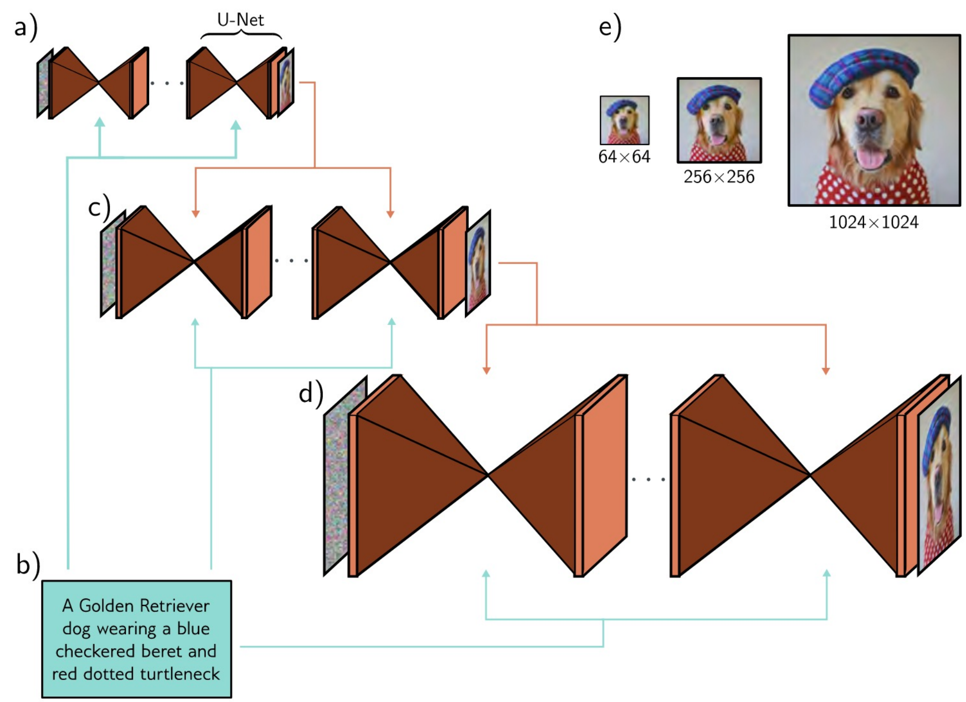


Figure 18.11 Cascaded conditional generation based on a text prompt. a) A diffusion model consisting of a series of U-Nets is used to generate a 64×64 image. b) This generation is conditioned on a sentence embedding computed by a language model. c) A higher resolution 256×256 image is generated and conditioned on the smaller image *and* the text encoding. d) This is repeated to create a 1024×1024 image. e) Final image sequence. Adapted from Saharia et al. (2022b).

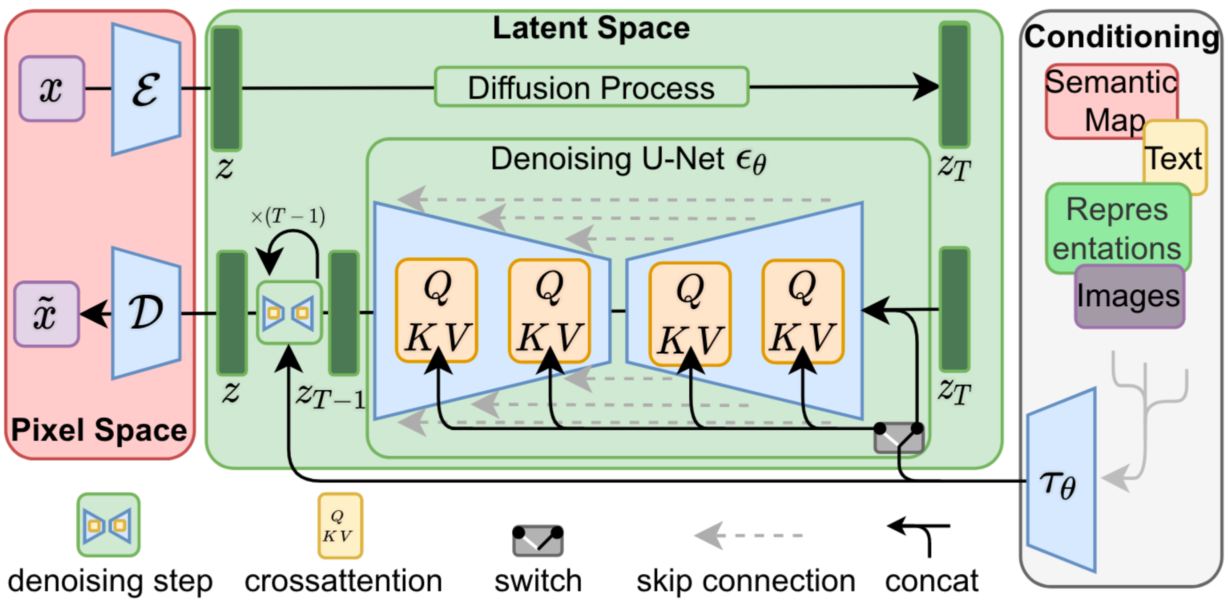
Conditional generation using text prompts

Figure 18.13 Conditional generation using text prompts. Synthesized images from a cascaded generation framework, conditioned on a text prompt encoded by a large language model. The stochastic model can produce many different images compatible with the prompt. The model can count objects and incorporate text into images. Adapted from Saharia et al. (2022b).

C. Saharia *et al.*, "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding," 2022.

Stable Diffusion – Latent Diffusion Models

Project the original data to a smaller latent space using a conventional autoencoder and then run the diffusion process in the smaller space.



Rombach et al, "High-Resolution Image Synthesis with Latent Diffusion Models," 2022

For practice

- Notebook 18.1 – Diffusion Encoder in 1D Open in Colab
- Notebook 18.2 – Training Decoder Open in Colab
- Notebook 18.3 – 1D Reparameterized Model (more robust) Open in Colab
- Notebook 18.4 – Families of Diffusion Models (DDIM) Open in Colab

Resources

- UDL, Chapter 18, Diffusion Models
- Rocca, "Understanding Diffusion Probabilistic Models (DPMs)", Medium.com, 2022
- HuggingFace Diffusion Models Class
- CVPR 2023 Tutorial: Denoising Diffusion Models: A Generative Learning Big Bang, https://cvpr.thecvf.com/virtual/2023/tutorial/18546

Link

For discussion sections:

- Recap the theory
- Show example code with MNIST
- Review Diffusion Transformer (time permitting)