

Deep Learning for Data Science

DS 542

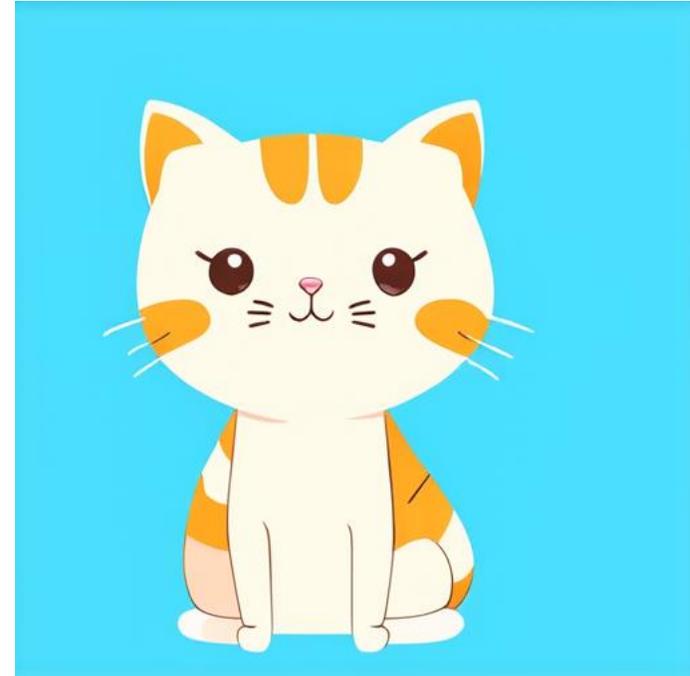
<https://dl4ds.github.io/sp2026/>

Diffusion Models

Plan for Today

- Project 4
- Denoising autoencoders
- Error diffusion process
- Diffusion models

Project 4 Inputs



<https://github.com/DL4DS/synth-cute>

Project 4

Build a model

- Synthetic data set
 - Built with Stable Diffusion 3.
- Build your own image diffusion model based on this data set.

Grading criteria

Auto-grader (**not live yet**)

- Inception score
- Fréchet Inception Distance

Manual-grading (**subjective**)

- 5% our chosen seeds
- 5% your choice of best output

Will share our favorites in Piazza

Quantifying Performance - Inception Score

Grading via another model

- Usually the Inception model for ImageNet
- Want generated images to have a single very likely classification.
- But average flat classification across generated images.
- Formal formula checking KL-divergence between those on a per-generated image basis...

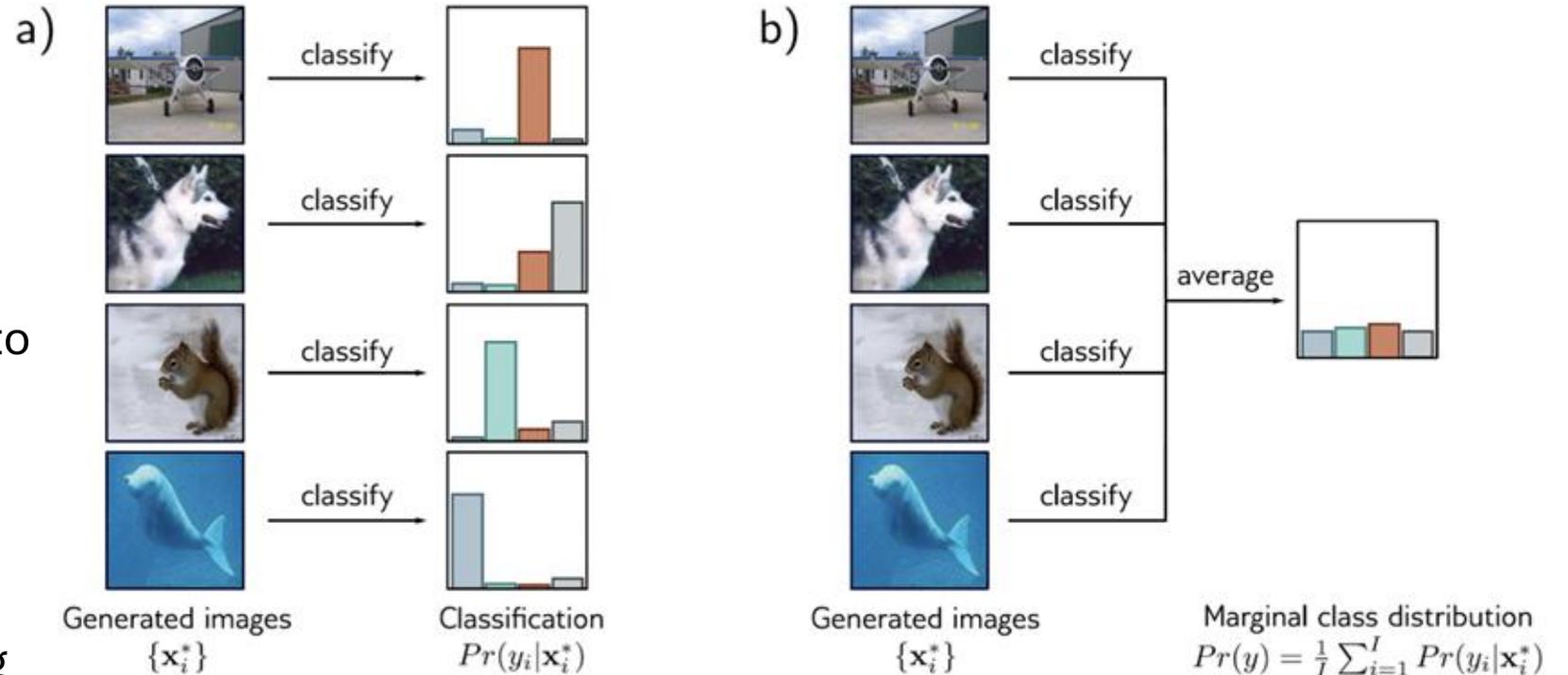


Figure 14.4 Inception score. a) A pretrained network classifies the generated images. If the images are realistic, the resulting class probabilities $Pr(y_i|\mathbf{x}_i^*)$ should be peaked at the correct class. b) If the model generates all classes equally frequently, the marginal (average) class probabilities should be flat. The inception score measures the average distance between the distributions in (a) and the distribution in (b). Images from Deng et al. (2009).

Inception Score Math

Inception score

$$e^{\mathbb{E}_x[KL(p(y|x)||p(y))]}$$

Why exponentiate the whole thing?

Decoder ring

- $p(y)$ = probability of class y
- $p(y|x)$ = conditional probability of class y
- $KL(P(x)||Q(x)) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)} =$
Kullback-Leibler divergence.
 - Interpretation is comparing approximate $Q(x)$ to true $P(x)$.

Interpretation of Inception Score

$$e^{\mathbb{E}_x[KL(p(y|x)||p(y))]}$$

- Range of values is $[1, n]$ for n classes.
- Value of 1 implies $\mathbb{E}_x[KL(p(y|x)||p(y))] = 0$
 - So, average KL divergence is 0.
 - But KL divergences is always ≥ 0 .
 - So, this implies that the classifier just predicts original class distribution.
classifier failed!
- High values correspond to classifier being able to distinguish classes of generated images.

Quantifying Performance – Fréchet Inception Distance

Another visual similarity metric based on Inception model (others can be used).

- Map generated images to distribution of Inception features.
- Model the distribution of Inception features as a multivariate normal distribution.
- Compare two such distributions with the Wasserstein distance.
 - Also called “earth mover’s distance”
 - Smaller is better.
 - Closed form solution from multivariate normal assumption.

Fréchet Inception Distance Math

- Let f map images to the **features** (last hidden outputs) of a classification model.
 - Originally used Inception model. Project 4 will use a new classifier.
- Apply f to real data set and generated data set.
 - Compute mean μ and covariance Σ of f outputs for each data set.
 - Use these statistics to specify **multivariate normal distribution approximating the distribution** of the features of each data set.
- Compute Fréchet distance d between these distributions.

$$d^2 = |\mu_G - \mu_R|^2 + \text{tr}(\Sigma_G + \Sigma_R - 2(\Sigma_G \Sigma_R)^{1/2})$$

Any Questions?

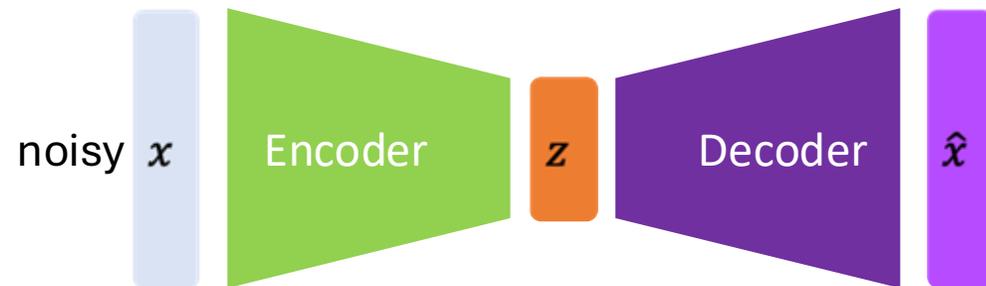


Moving on

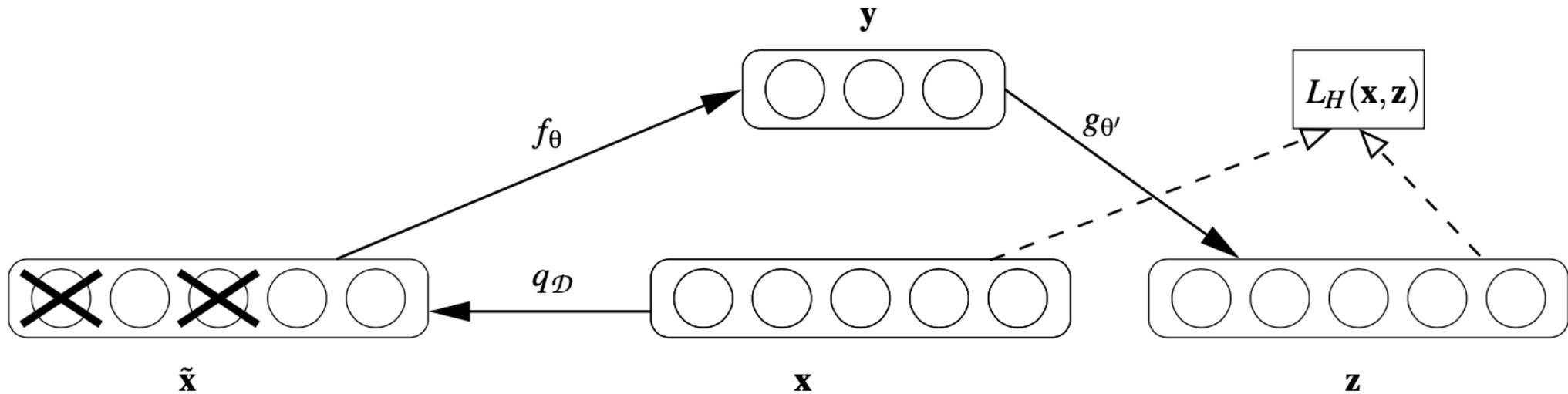
- Project 4
- Denoising autoencoders
- Error diffusion process
- Diffusion models

Denoising Autoencoder Idea

- Train an autoencoder but feed noisy data into the encoder.
- Still try to recover the clean data.



Denoising Autoencoder Training



“Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion” by Vincent et al (2010)

Denoising Autoencoders

Early predecessors, but
could not handle nearly as
much noise.

“Stacked Denoising
Autoencoders: Learning
Useful Representations in a
Deep Network with a Local
Denoising Criterion” by
Vincent et al (2010)



(a) SAE

(b) SDAE

Any Questions?

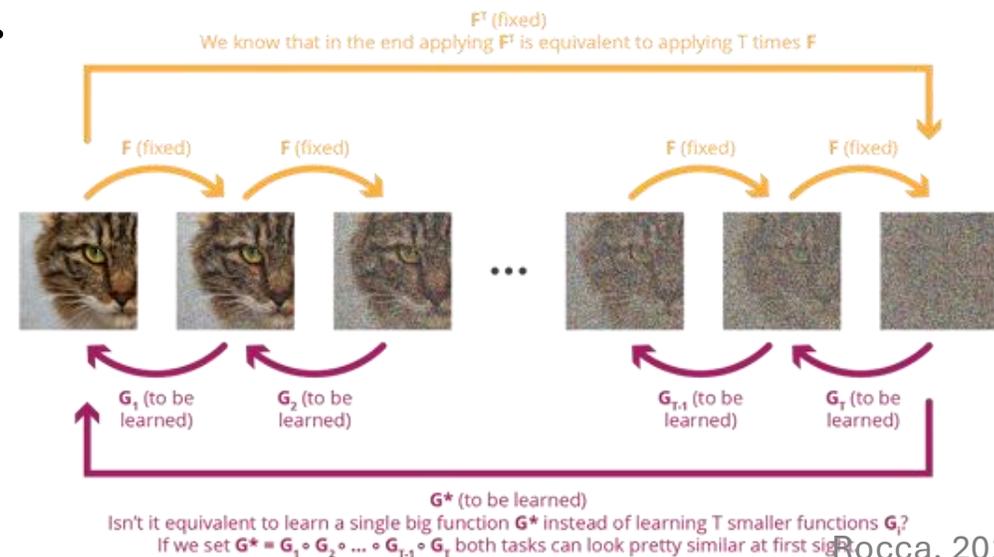


Moving on

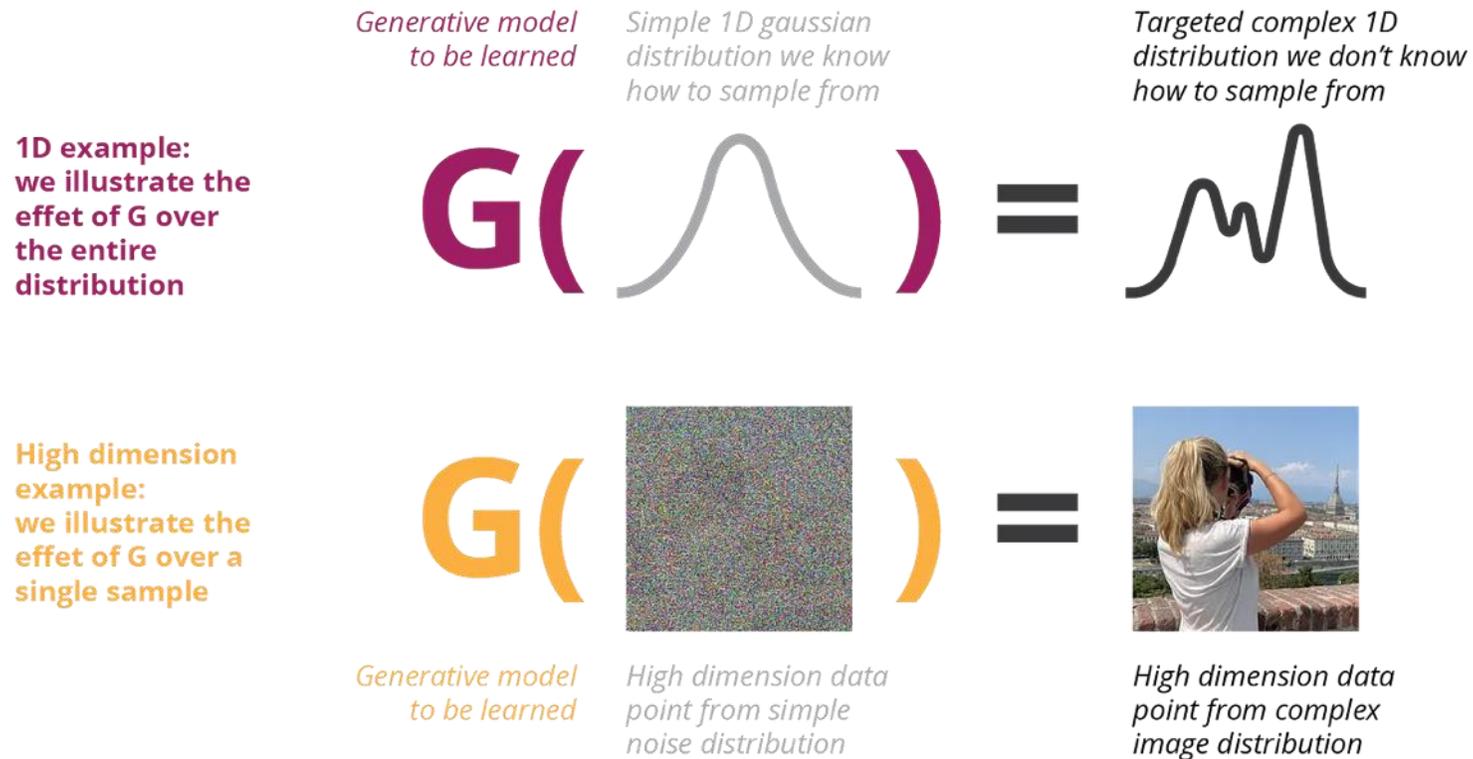
- Project 4
- Denoising autoencoders
- **Error diffusion process**
- Diffusion models

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.



Simple Latent Distribution to Complex Output Distribution



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

Stochastic process

Stochastic Processes

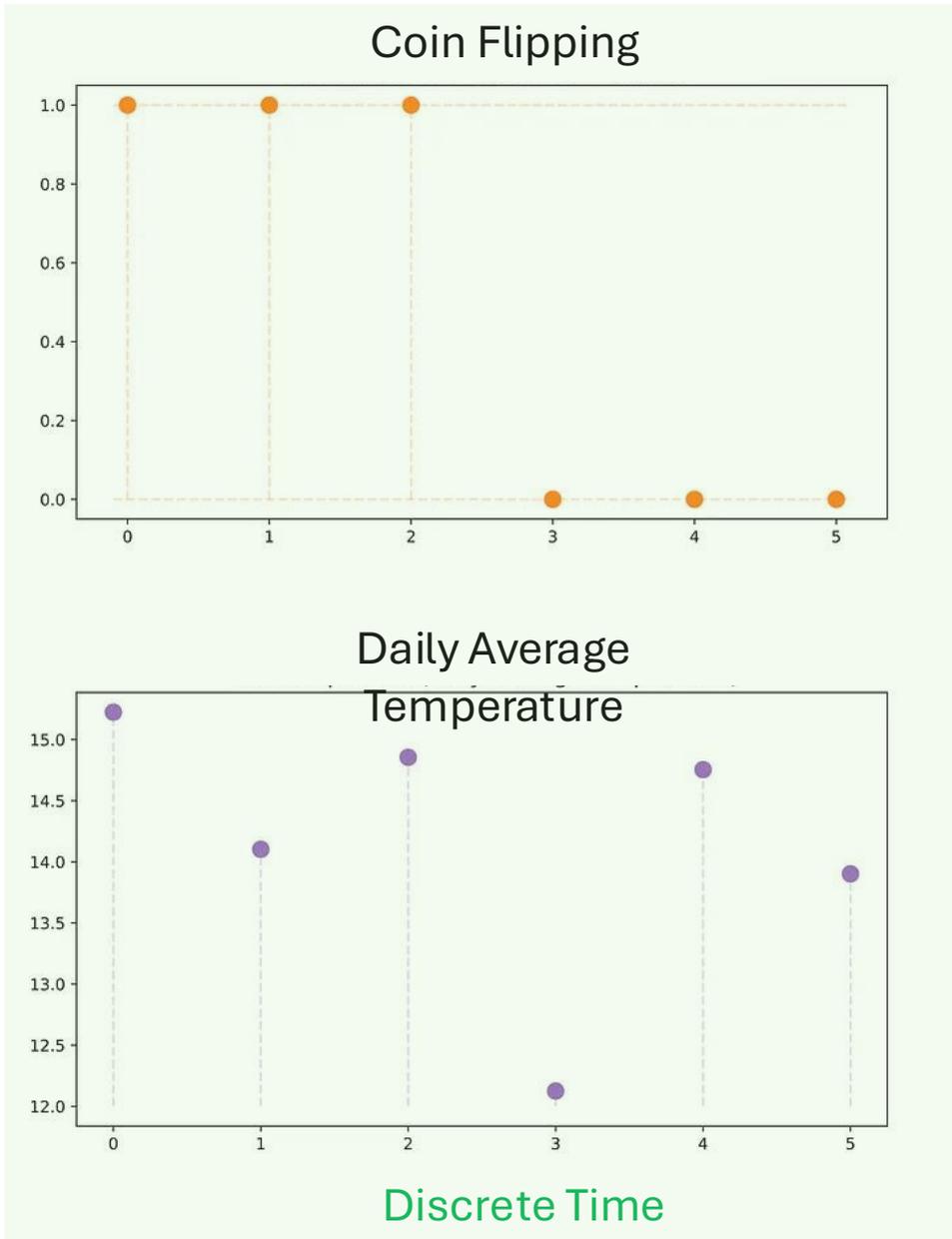
- Discrete: $X_n, \quad \forall n \in \mathbb{N}$
- Continuous: $X_t, \quad \forall t \geq 0$

Realization of a random variable \rightarrow sample

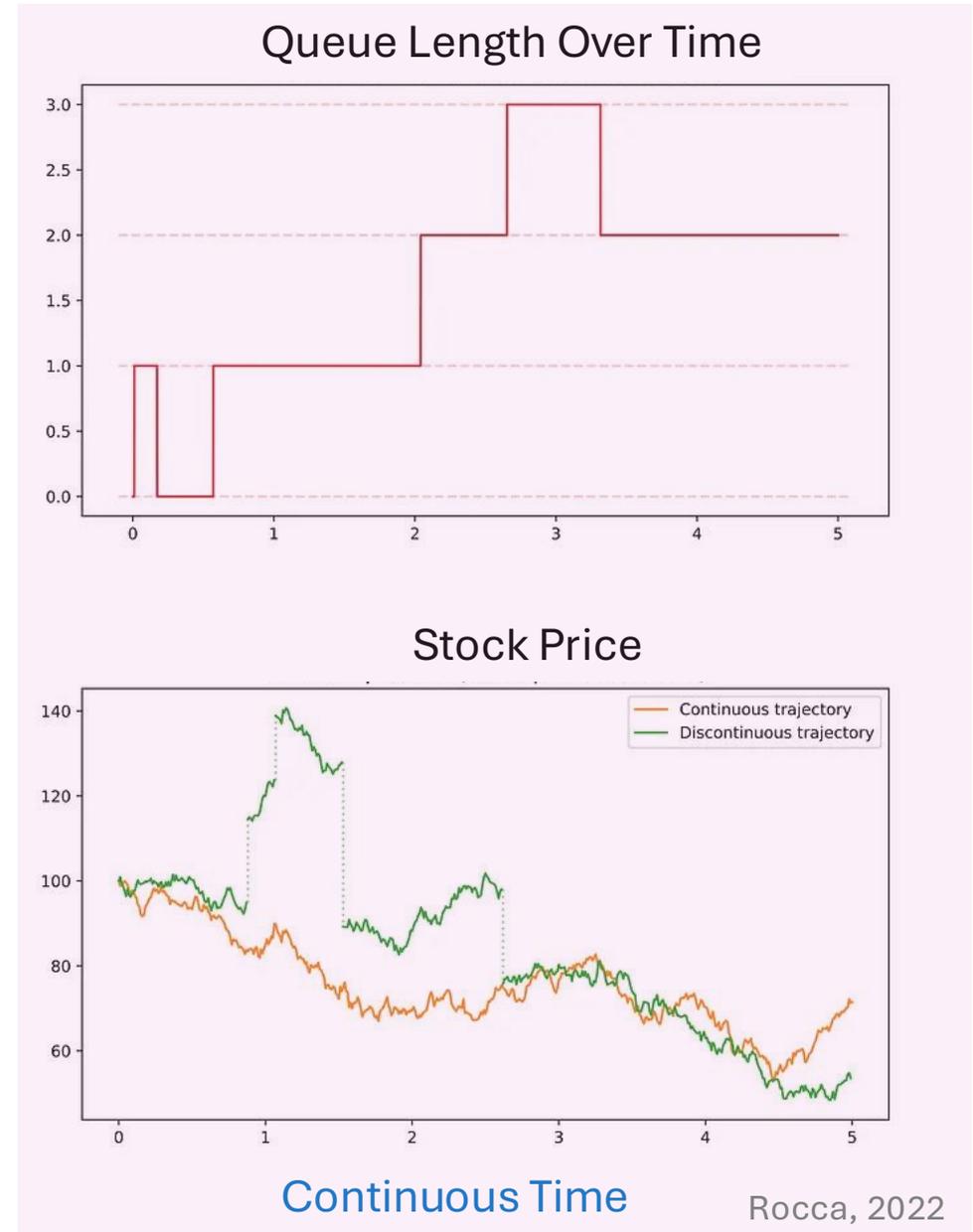
Realization of a stochastic process \rightarrow sample path or trajectory

Different types of stochastic processes

Discrete Value



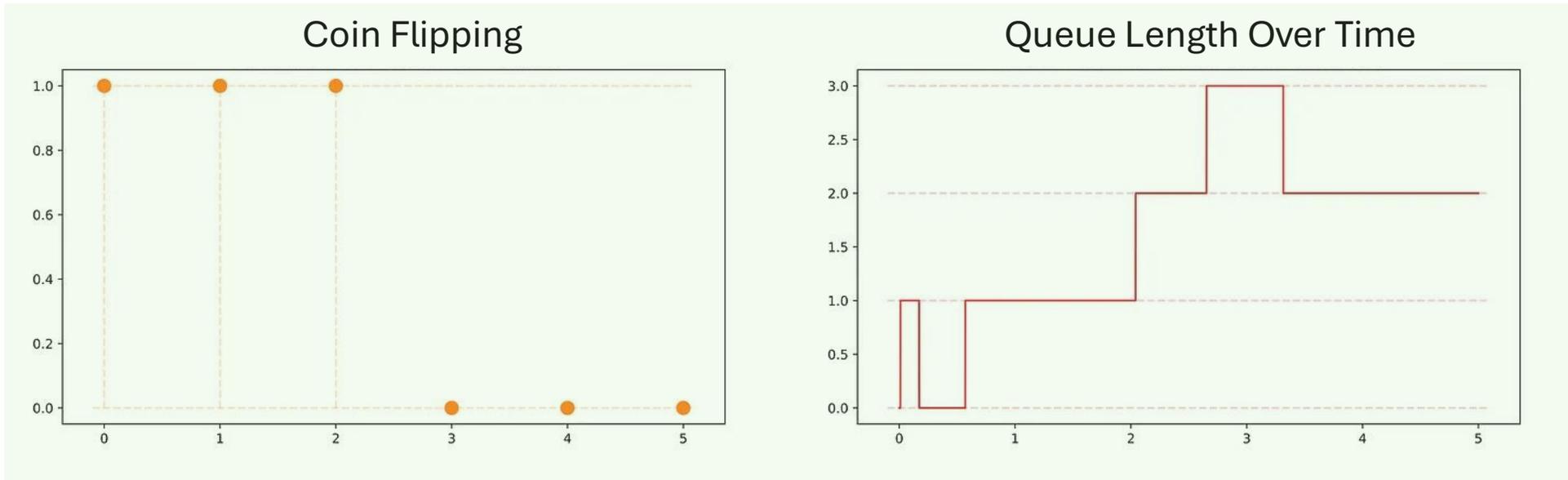
Continuous Value



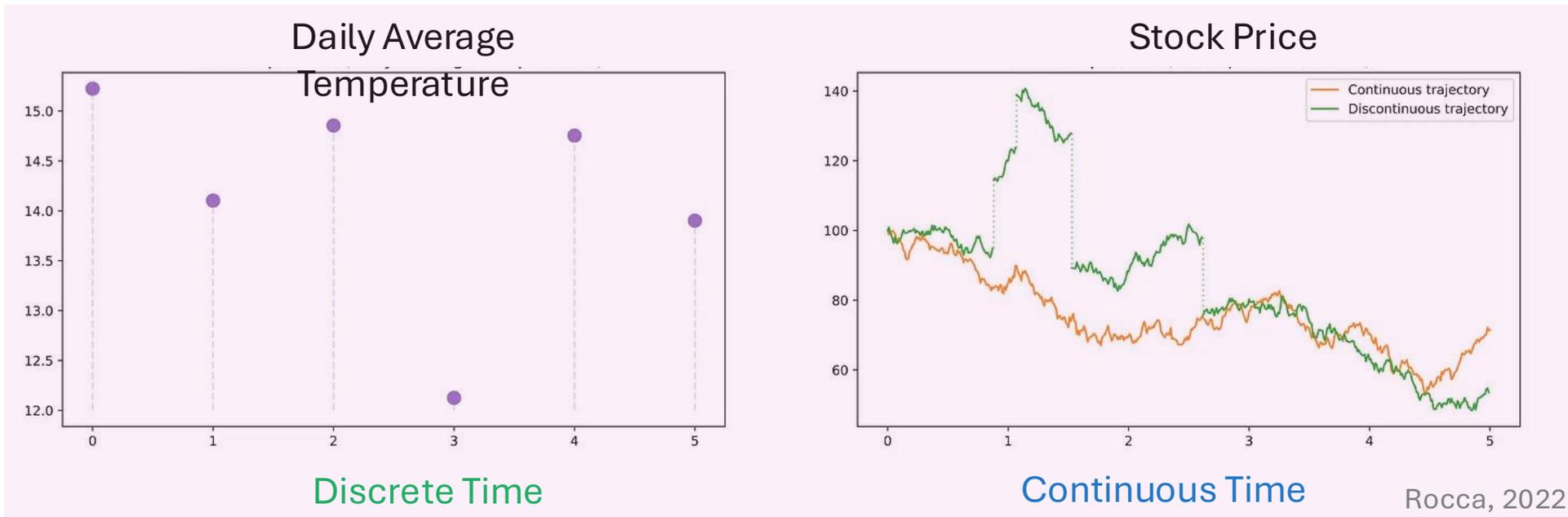
Continuous Time

Different types of stochastic processes

Discrete Value



Continuous Value



Markov (Stochastic) Process

A *Markov process* is a *stochastic process* with no memory; only the current state matters.

Future behavior only depends on the present, or

Present only depends on the previous sample.

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

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)

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

where:

$a(\cdot)$ is called the **drift coefficient**

$\sigma(\cdot)$ is called the **diffusion coefficient**

W is the **Wiener process**

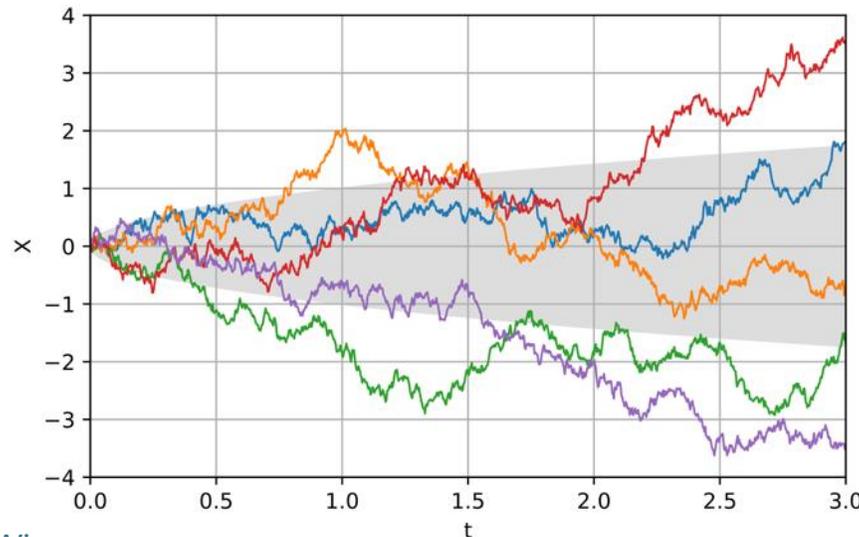
Wiener Process (Brownian Motion)

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 ,
→ $W_{t+u} - W_t \sim \mathcal{N}(0, u)$.
4. W has almost surely continuous paths: W_t is almost surely continuous in t .

Five sampled processes



Expected standard deviation

Norbert Wiener

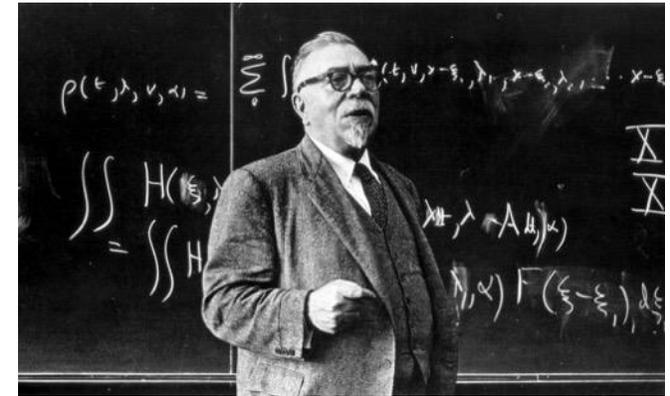
Norbert Wiener (November 26, 1894 – March 18, 1964) was an American computer scientist, mathematician and philosopher. He became a professor of mathematics at the Massachusetts Institute of Technology (MIT).

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.

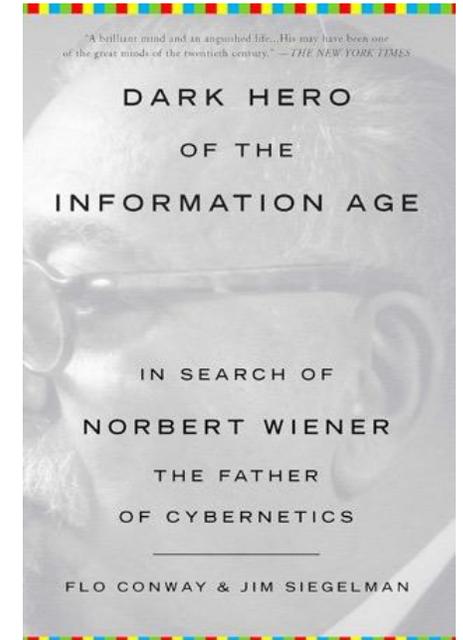
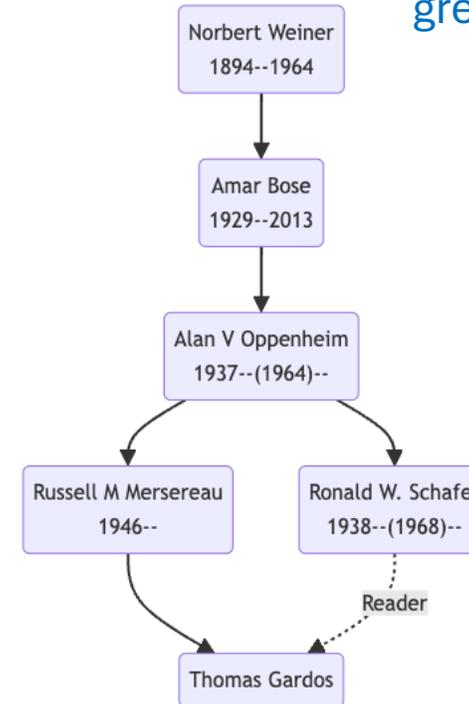
Wiener is considered the originator of cybernetics, the science of communication as it relates to living things and machines.

Heavily influenced John von Neumann, Claude Shannon, etc...

Wrote “The Machine Age” in 1949 anticipating robots, etc.



great, great grand advisor 😊



Discretizing

So

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

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

Discretizing the SDE

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

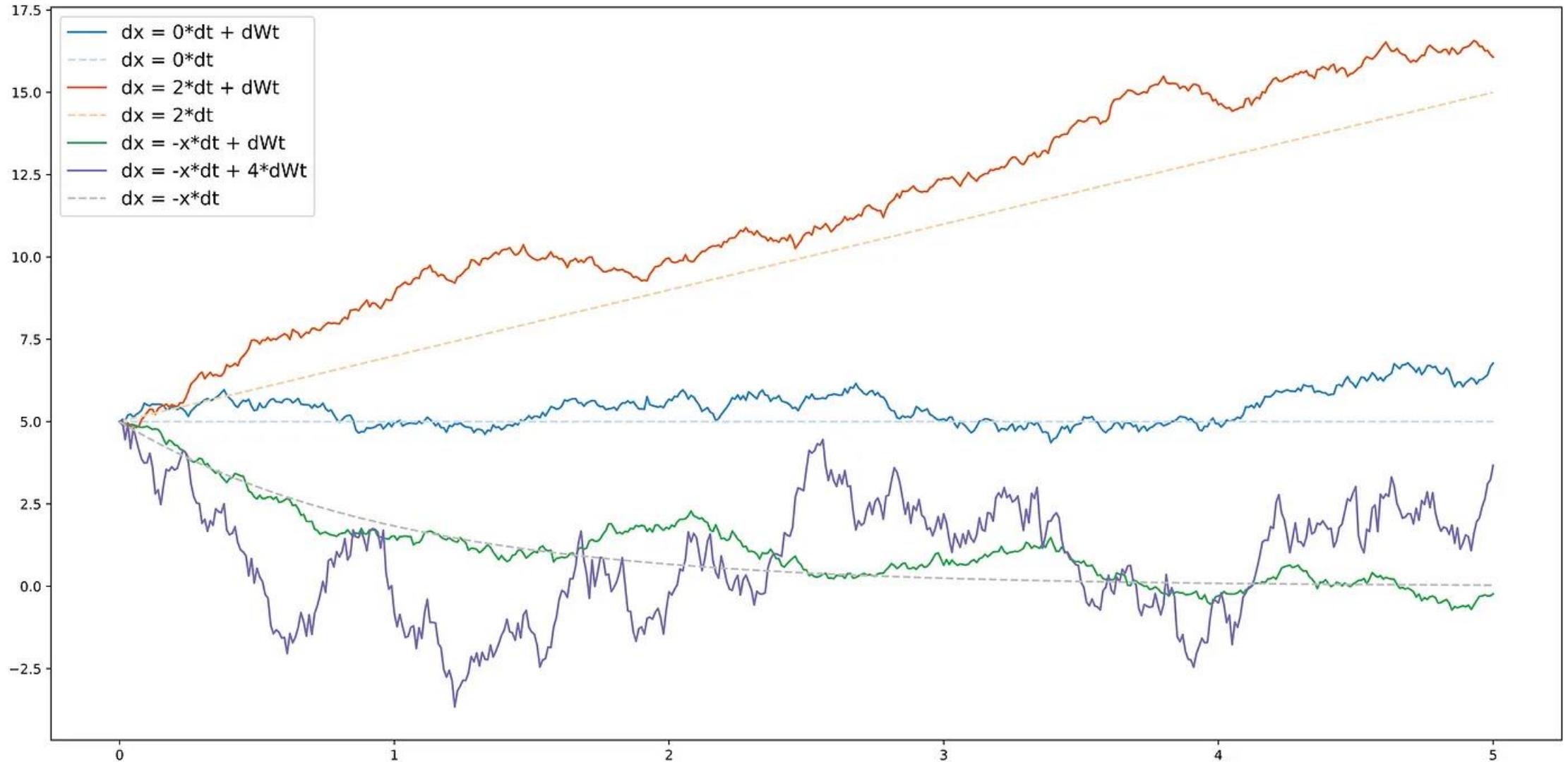
Which can also be rewritten

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

Deterministic drift term

Normal RV with std proportional to diffusion term

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

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

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

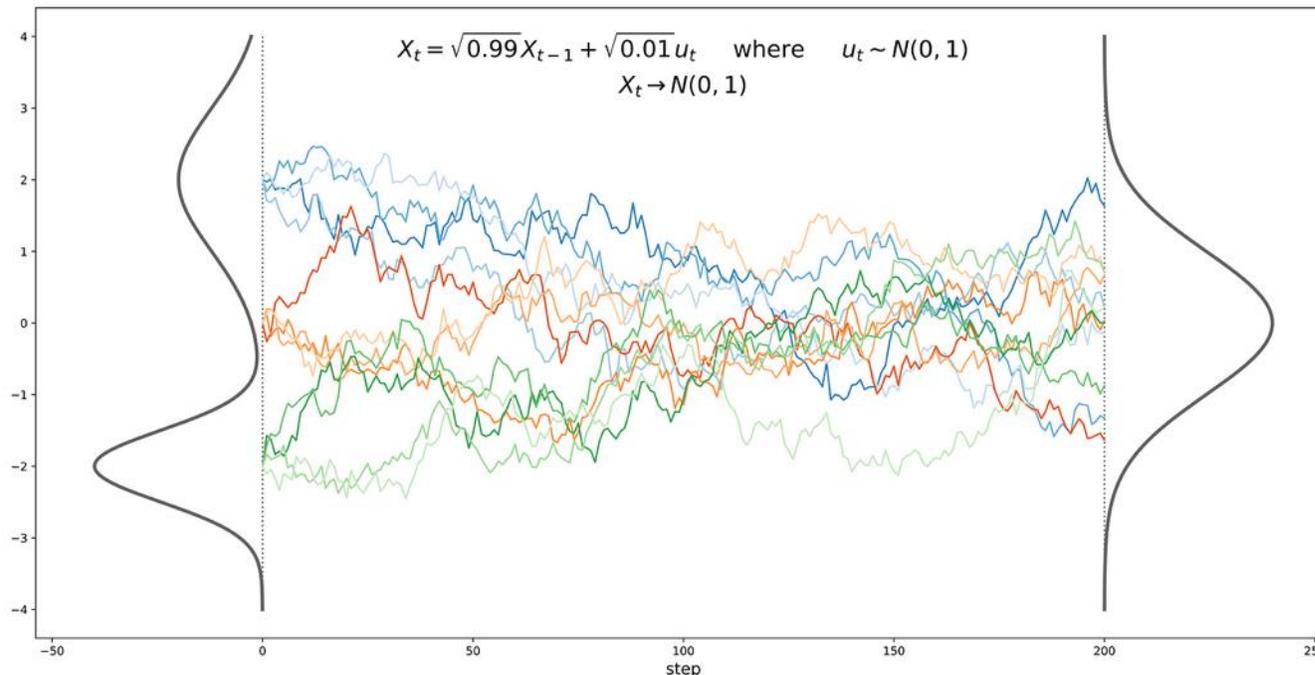
and $p(X_t)$ is the *marginal probability* of X_t

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

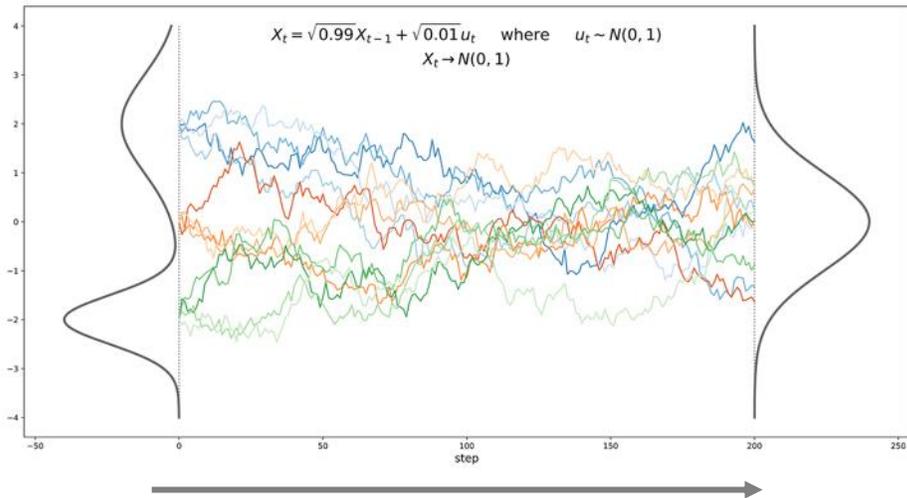
$$X_t = \sqrt{1-p}X_{t-1} + \sqrt{p}u_t \quad \text{where} \quad u_t \sim \mathcal{N}(0,1) \quad \text{and} \quad p = 0.01$$



Intuition behind diffusion processes

For the diffusion process

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



Towards Gaussian

After a given number of steps T we can write

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

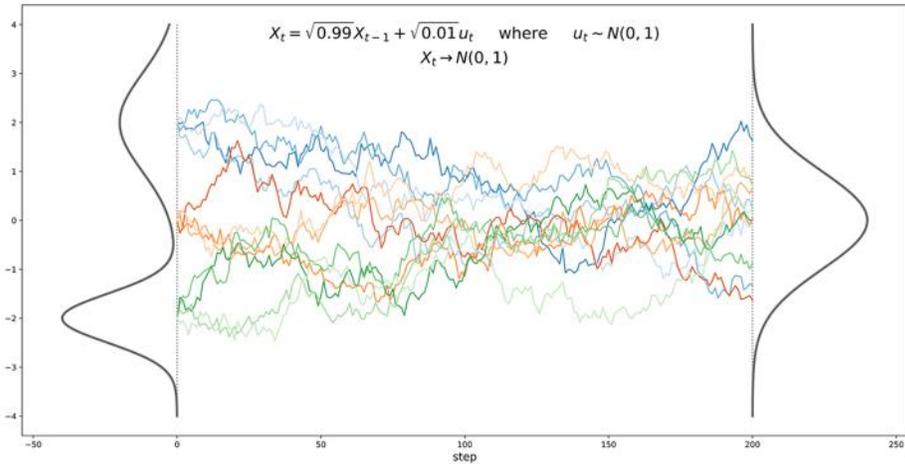
This is a sum of independent gaussians, so can express as single gaussian with variance the sum of the variances.

Intuition behind diffusion processes

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



Towards Gaussian

$$(\sqrt{1-p})^T \xrightarrow{T \rightarrow \infty} 0$$

and

$$\sum_{i=0}^{T-1} \underbrace{\left(\sqrt{p}(\sqrt{1-p})^i \right)^2}_{\text{Geometric series}} = p \frac{1 - (1-p)^T}{1 - (1-p)} \xrightarrow{T \rightarrow \infty} 1$$

Variance of the gaussian

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

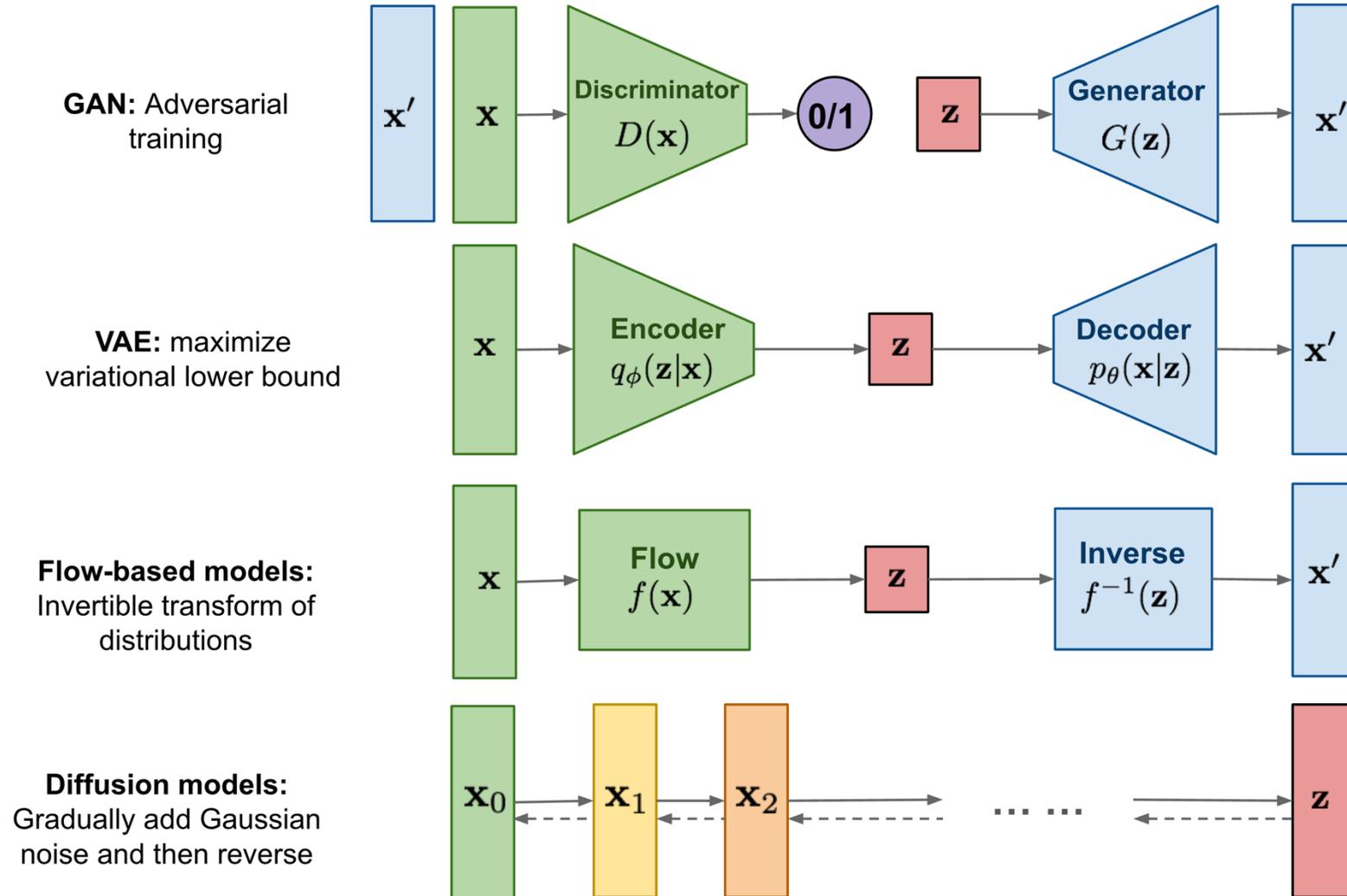
Any Questions?



Moving on

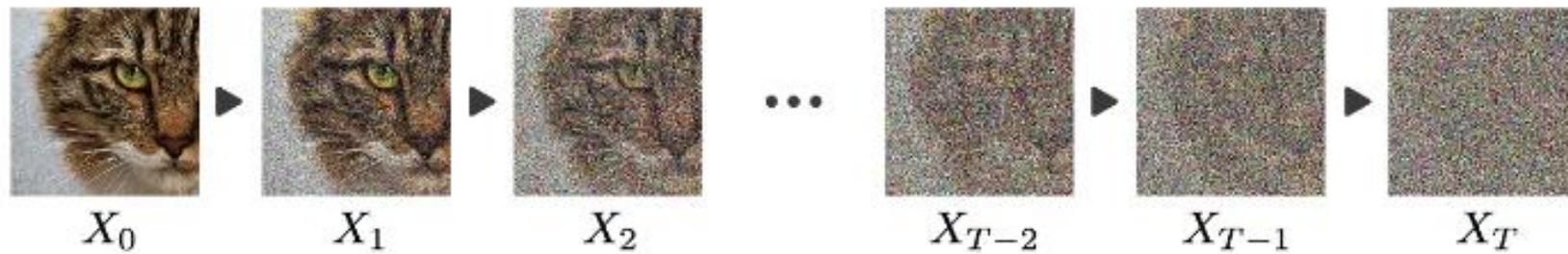
- Project 4
- VAE math wrap up
- Denoising autoencoders
- Error diffusion process
- **Diffusion models**

Different Generative Models



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



$$X_1 = \sqrt{1-p} X_0 + \sqrt{p} u_1 \sim \mathcal{N}(0, I)$$

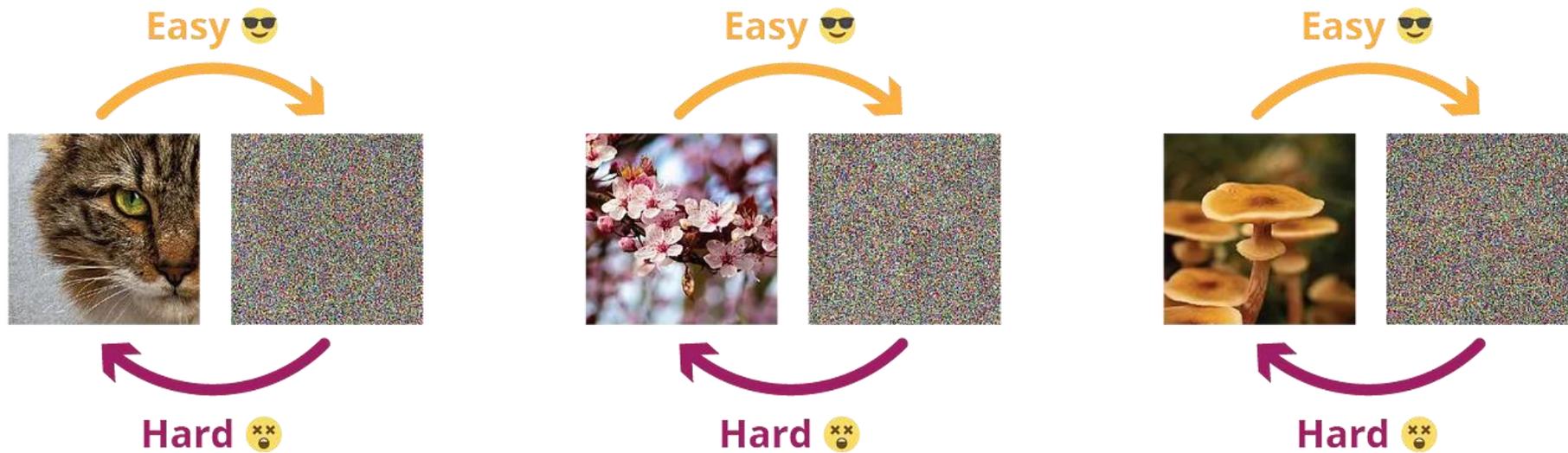
The equation shows the generation of X_1 from X_0 and a noise vector u_1 . X_1 is equal to $\sqrt{1-p}$ times X_0 plus \sqrt{p} times u_1 . The noise vector u_1 is sampled from a standard normal distribution, $\mathcal{N}(0, I)$.

Why use diffusion?

Answer:

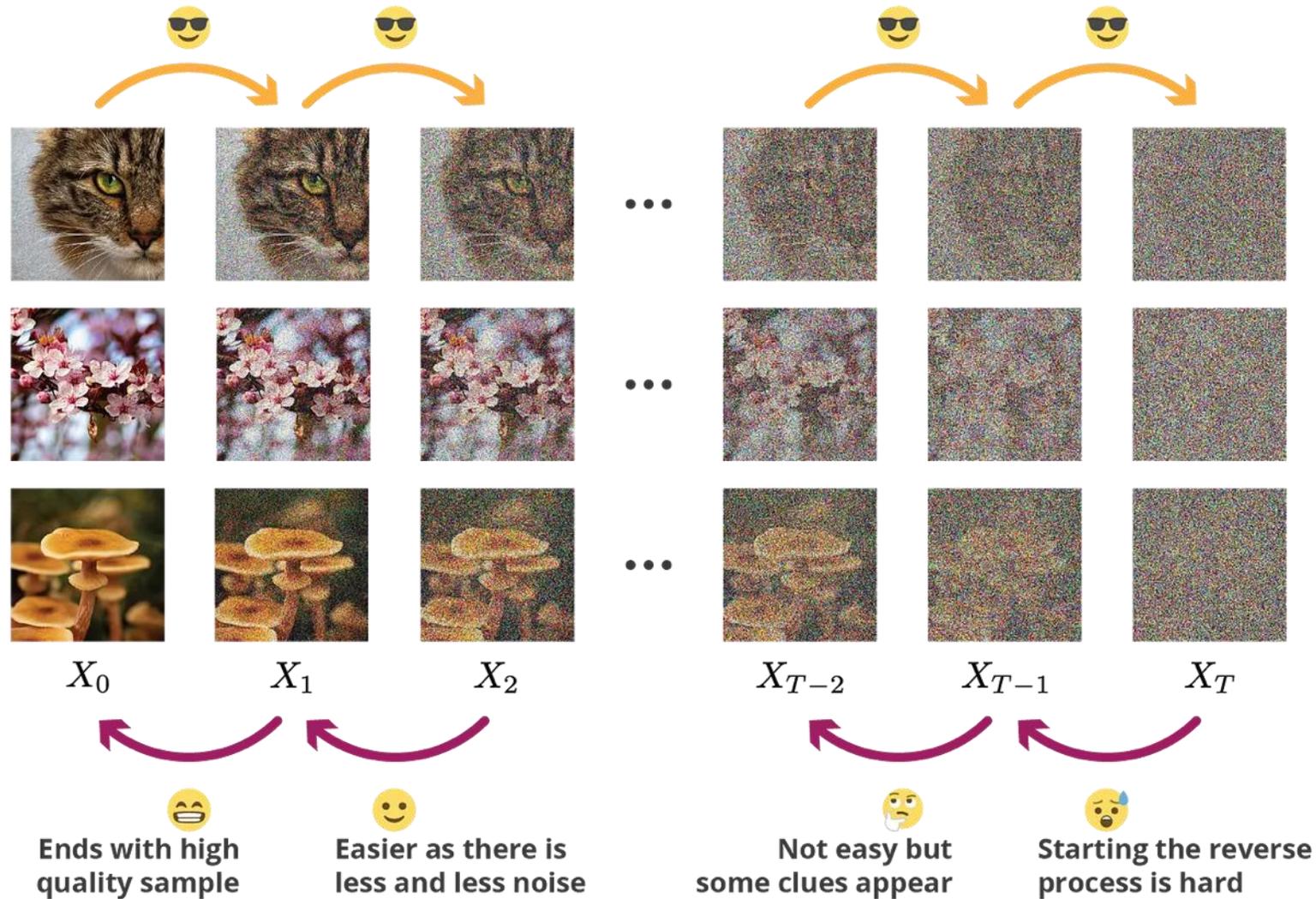
Gives us a progressive and structured way to go from a complex distribution to an isotropic gaussian noise that will enable the learning of the reverse process

Intuition behind learning the reverse process

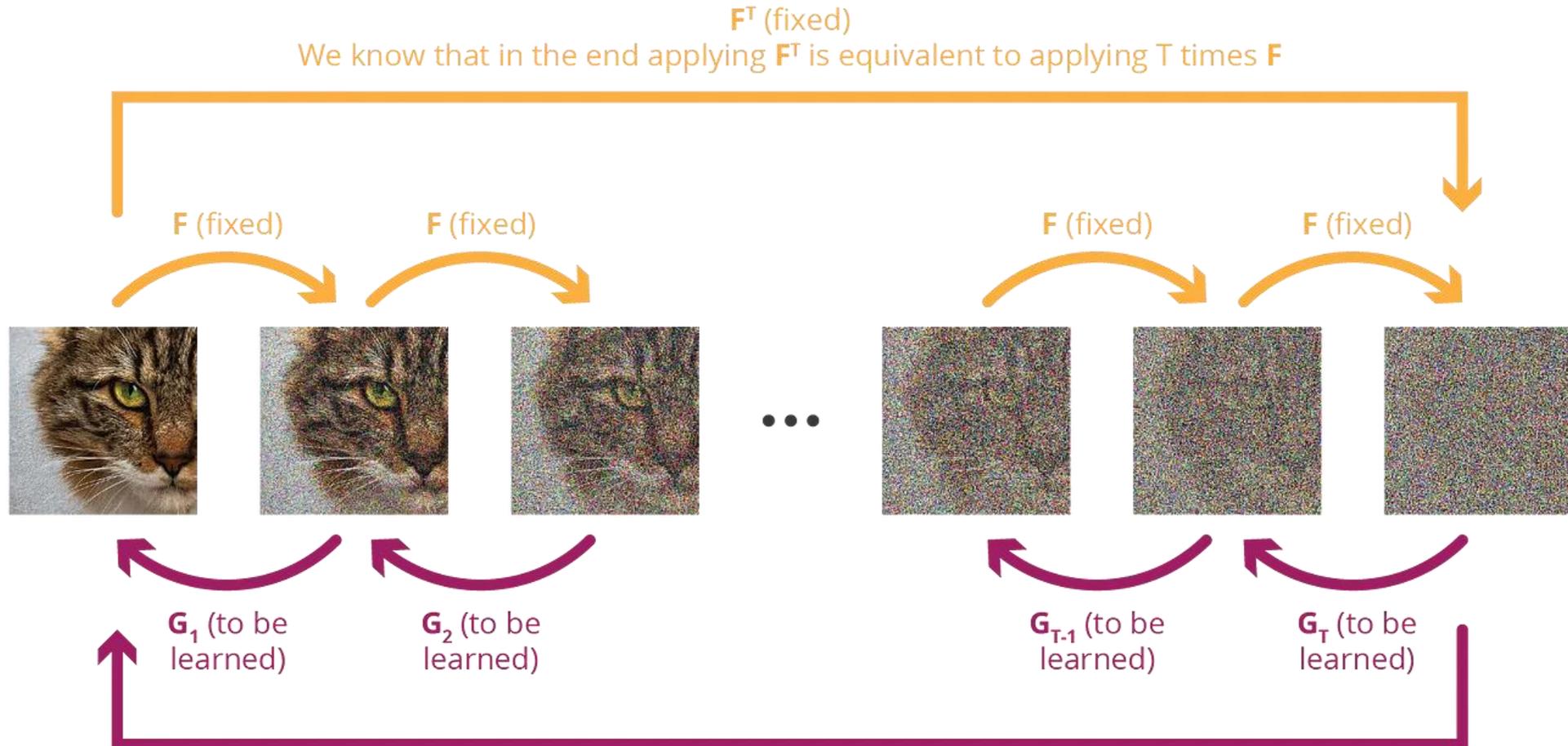


Reversing process in one step is extremely difficult.

Doing it in steps gives us some clues

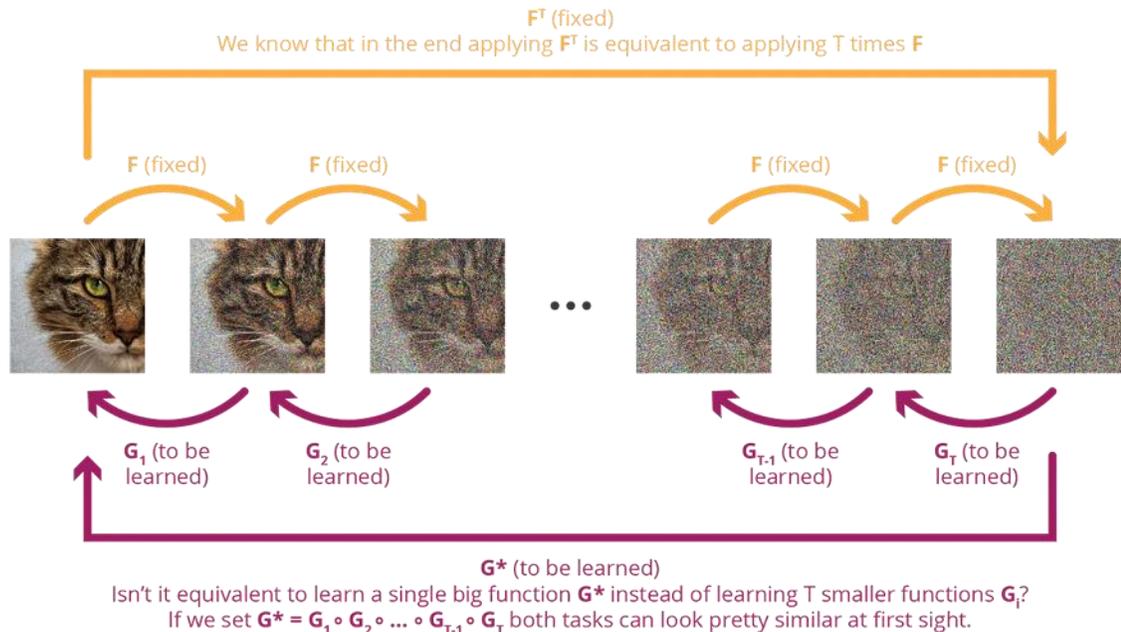


One step versus multi-step



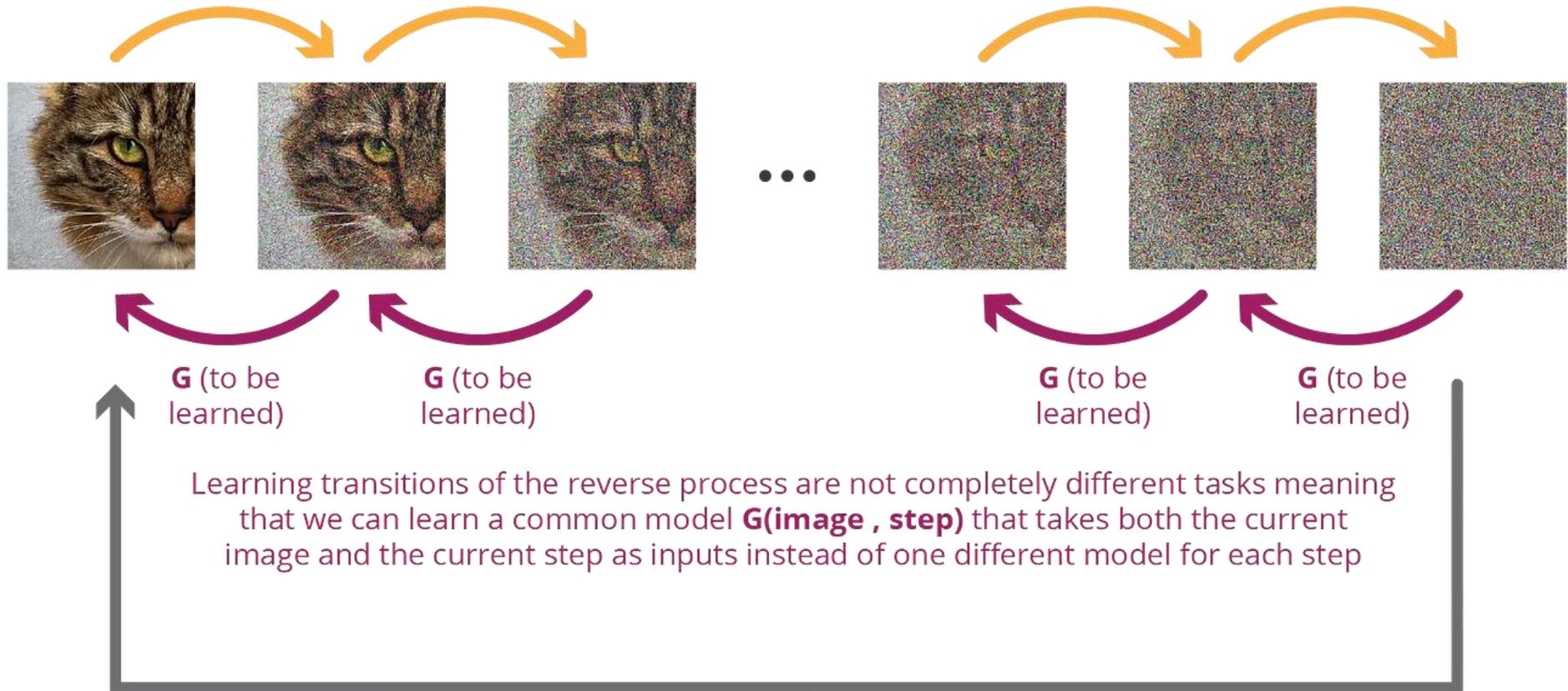
G^* (to be learned)
Isn't it equivalent to learn a single big function G^* instead of learning T smaller functions G_i ?
If we set $G^* = G_1 \circ G_2 \circ \dots \circ G_{T-1} \circ G_T$ both tasks can look pretty similar at first sight.

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



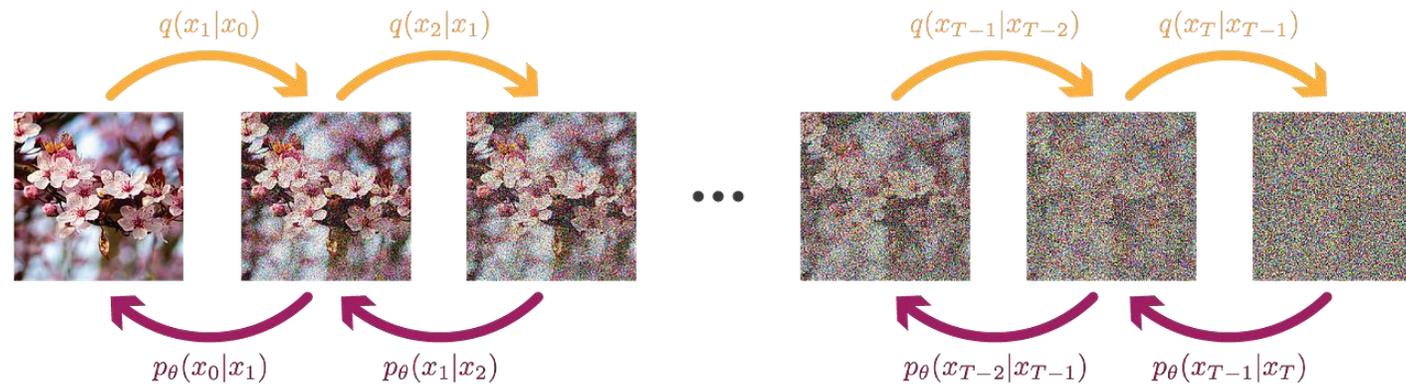
G* (to be learned)
G* can't rely on the same nice iterative structure than **G**, meaning that this unrolled version supposed to be equivalent to $\mathbf{G}_1 \circ \mathbf{G}_2 \circ \dots \circ \mathbf{G}_{T-1} \circ \mathbf{G}_T$ will have more parameters and will be harder to train

Learning the Backward Process

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

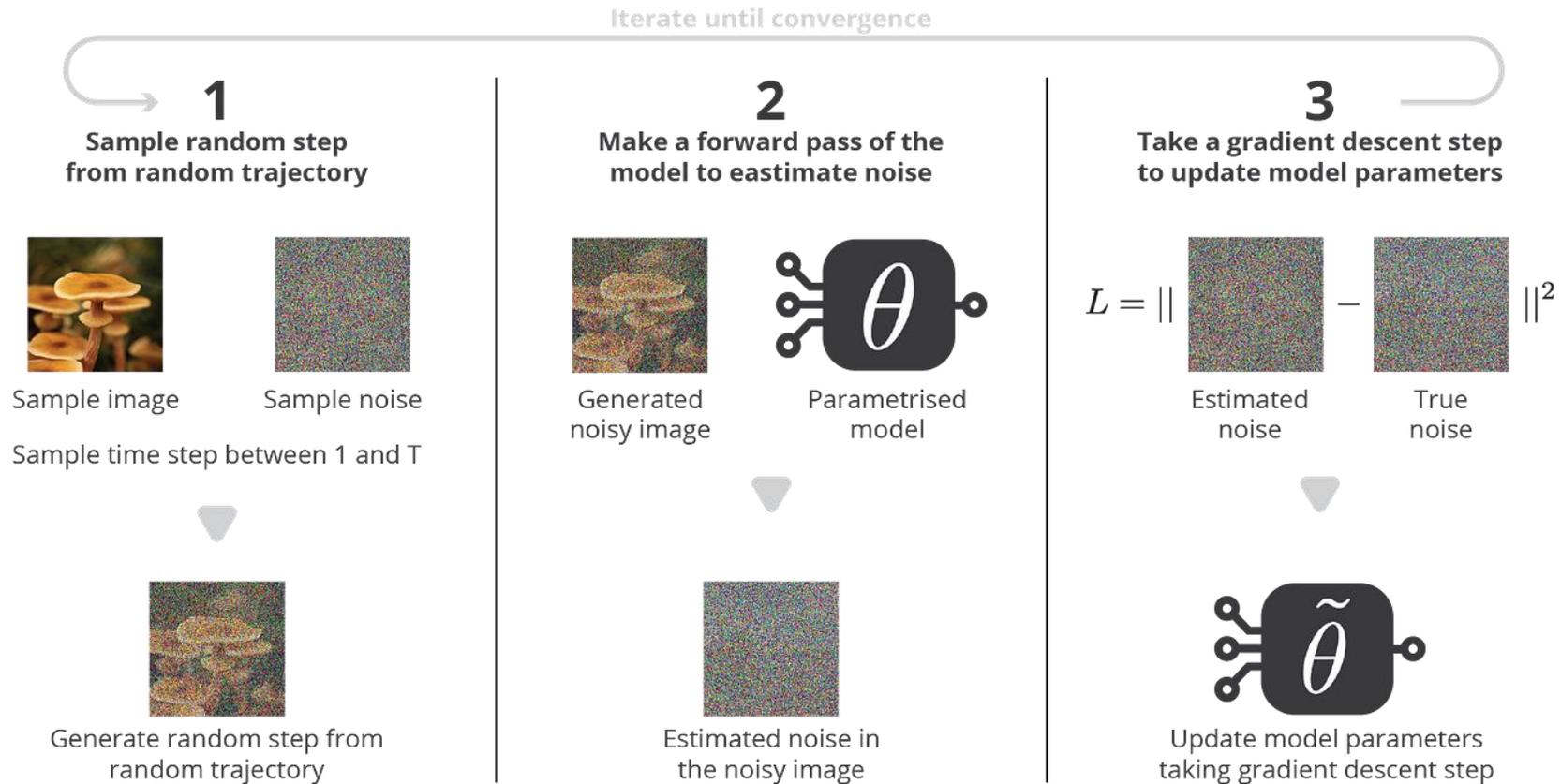
LEARNED BACKWARD PROCESS

Skipping a Lot of Math...



Will sketch it next time.

Focus on Estimating the Noise



Training

Algorithm 1 Training

1: **repeat**

2: $x_0 \sim q(x_0)$

▷ *Sample random initial data*

3: $t \sim \text{Uniform}(\{1, \dots, T\})$

▷ *Sample random step*

4: $\epsilon \sim \mathcal{N}(0, I)$

▷ *Sample random noise*

5: $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$

▷ *Rand. step of rand. trajectory*

6: Take gradient descent step on $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(x_t, t)\|^2$

▷ *Optimisation*

7: **until** converged

Sampling

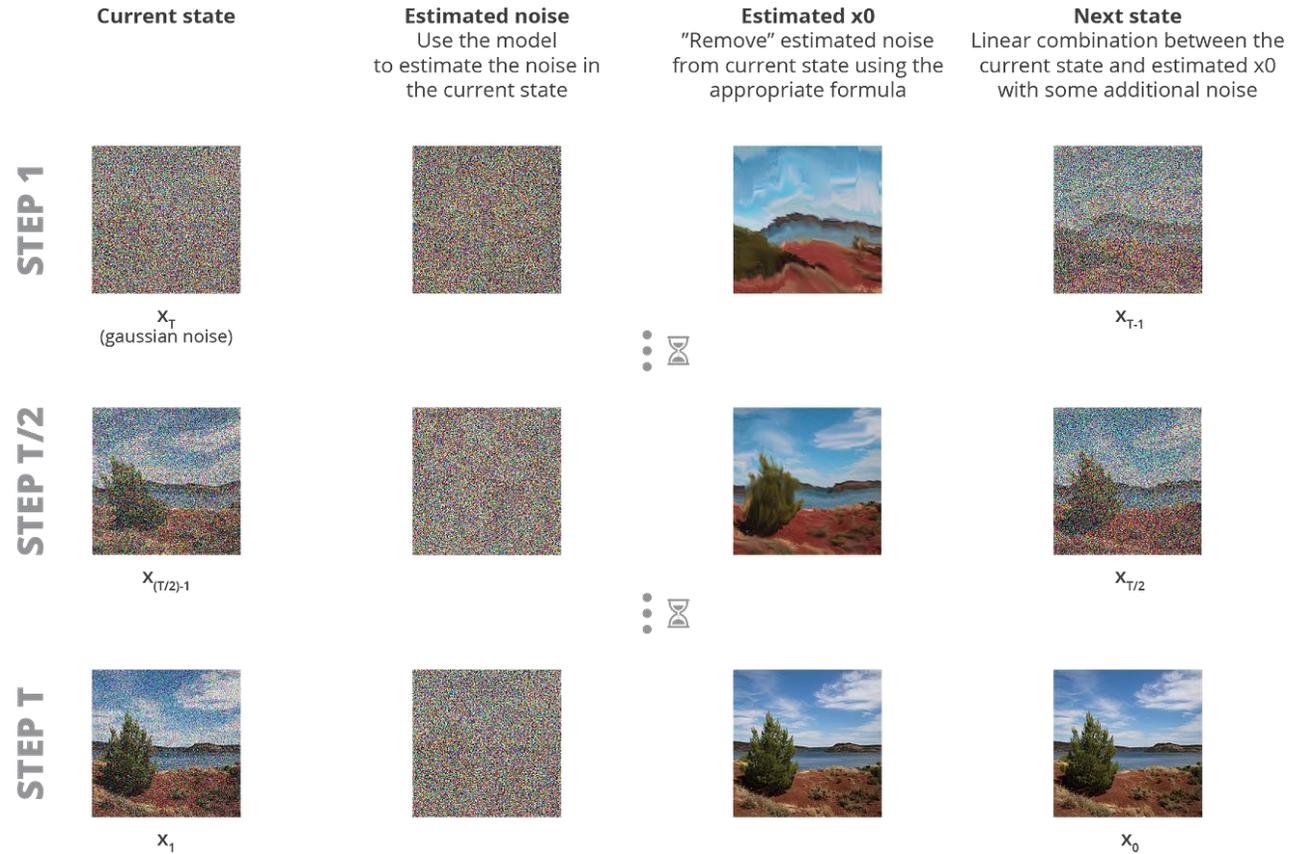
Algorithm 2 Sampling

- 1: $x_T \sim \mathcal{N}(0, I)$ *▷ Initial isotropic gaussian noise sampling*
 - 2: **for** $t = T, \dots, 1$ **do**
 - 3: $z \sim \mathcal{N}(0, I)$ if $t > 1$ else $z = 0$ *▷ Sample random noise (if not last step)*
 - 4: $\tilde{\epsilon} = \epsilon_\theta(x_t, t)$ *▷ Estimated noise in current noisy data*
 - 5: $\tilde{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1 - \bar{\alpha}_t}\tilde{\epsilon})$ *▷ Estimated x_0 from estimated noise*
 - 6: $\tilde{\mu} = \mu_t(x_t, \tilde{x}_0) \left(= \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) \right)$ *▷ Mean for previous step sampling*
 - 7: $x_{t-1} = \tilde{\mu} + \sigma_t z$ *▷ Previous step sampling*
 - 8: **end for**
 - 9: **return** x_0
-

Why Not One Step Reversal?

- Because we don't think that our model was **actually good enough** to reverse the whole thing at once.
- Remember, the noise terms are larger than the signal terms when we start this process!

Sampling Example



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 size as input**, as opposed to VAE which reduces dimensionality

Diffusion Model from Scratch

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

Any Questions?



Moving on

- Project 4
- VAE math wrap up
- Denoising autoencoders
- Error diffusion process
- Diffusion models