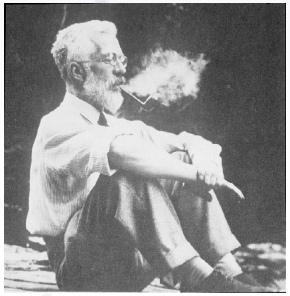
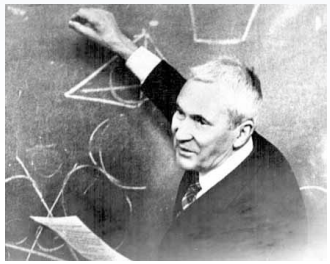
Applied ML

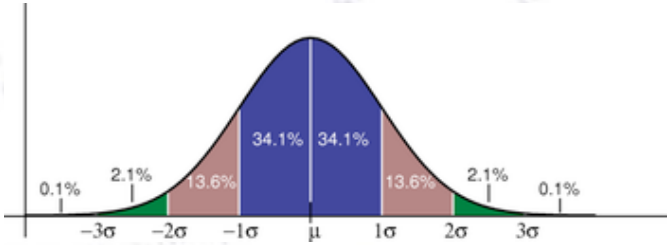
Diffusion Models





Troels C. Petersen (NBI)

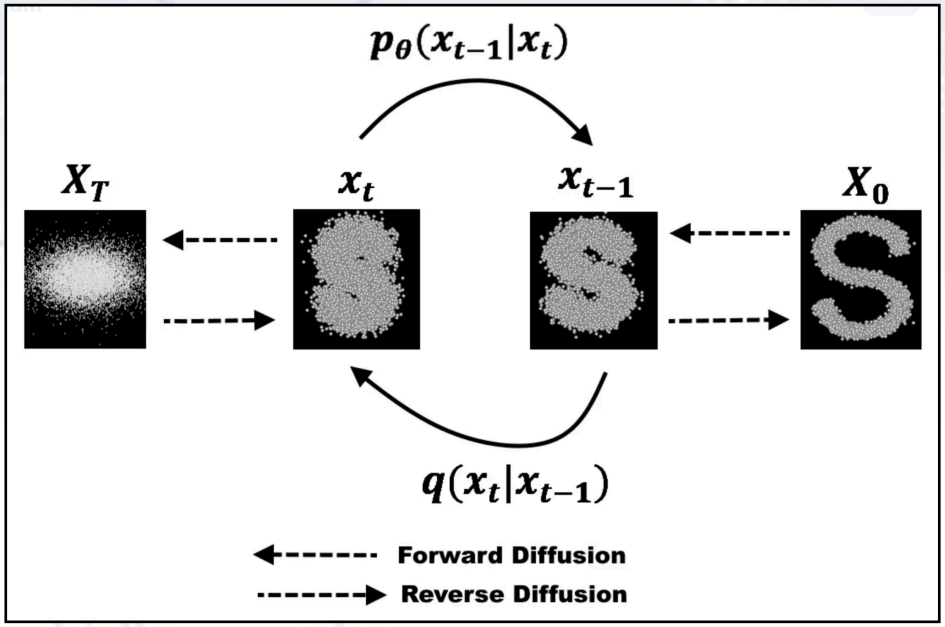


"Statistics is merely a quantisation of common sense - Machine Learning is a sharpening of it!"

Diffusion

Diffusion models is a class of models for computer vision that a capable of

- Image generation
- Image de-noising
- Image in-painting
- Image super-resolution



Diffusion models consist of three parts:

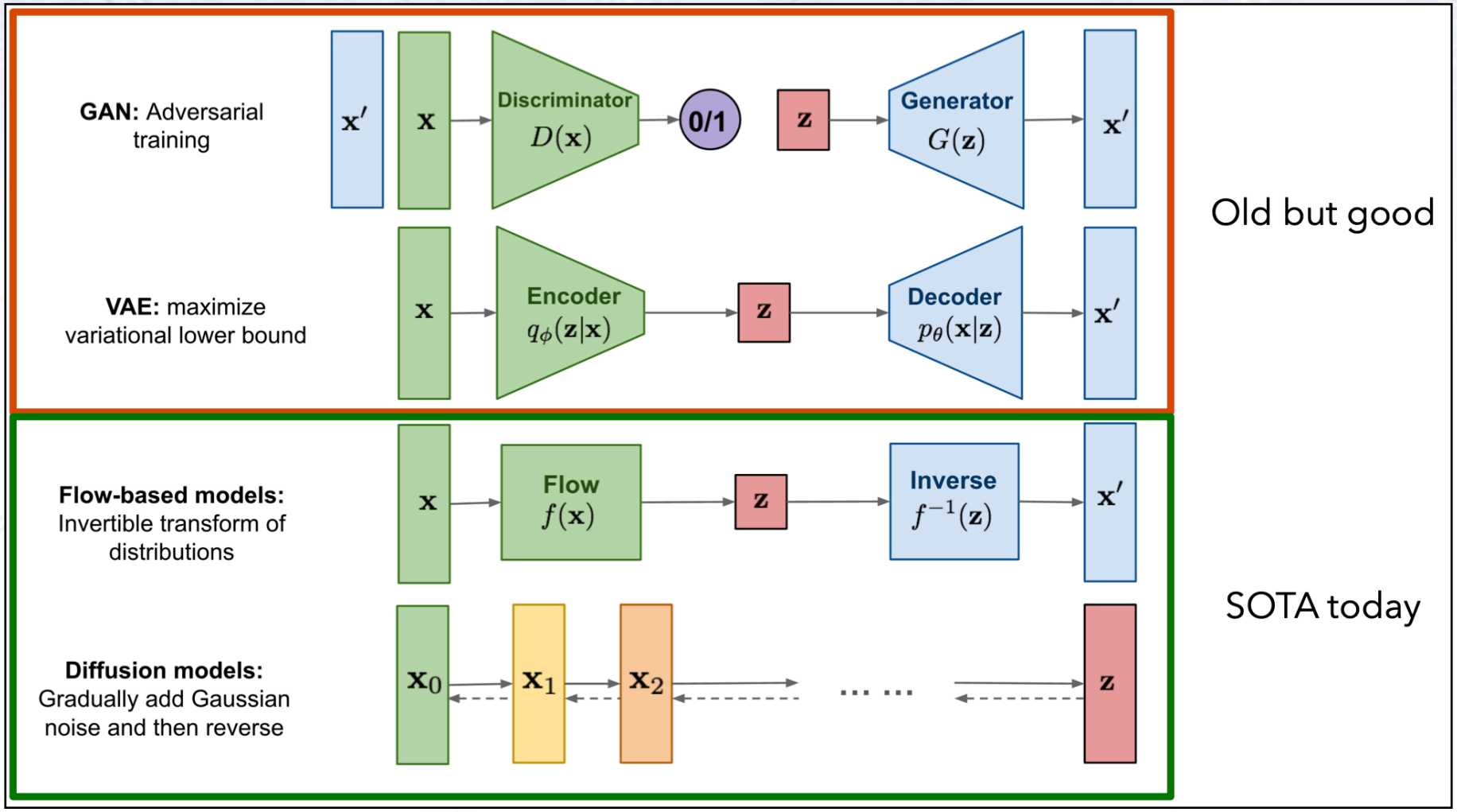
1. Forward diffusion
2. Reverse diffusion
3. Sampling procedure

They are build on the idea, that if you apply a known process (diffusion) on e.g. images, then by finding the reverse process, one can generate new images.

It is thus a **generative** model type. OpenAI's Dall-E 2 model - capable of generating images from text - is based on diffusion.

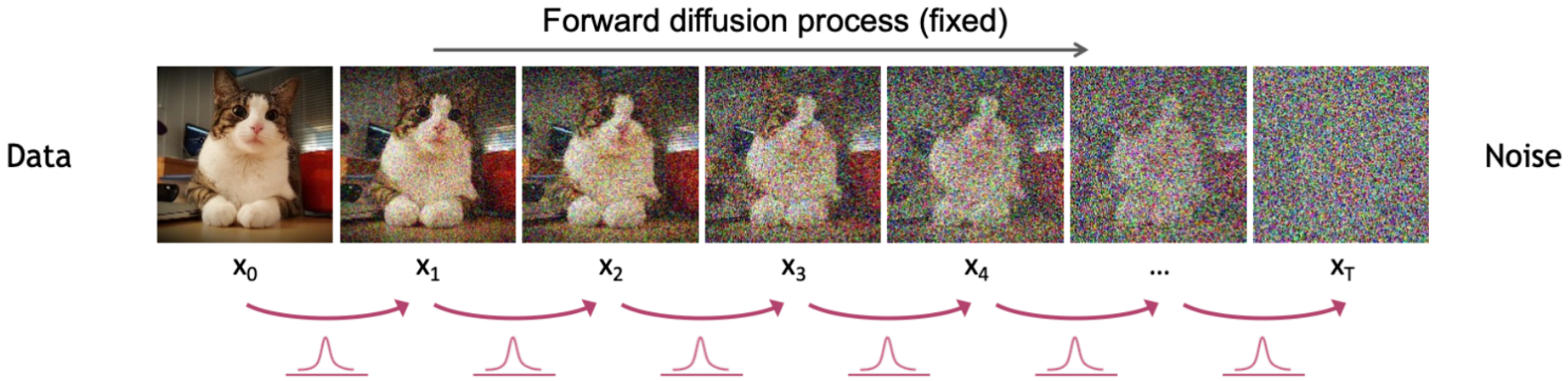
Diffusion

Diffusion has a different ML architecture from the other "classic" methods of generative models. In diffusion one uses a repeated alteration (not NN layers).



From https://lilianweng.github.io/posts/2021-07-11-diffusion-models

Forward Process

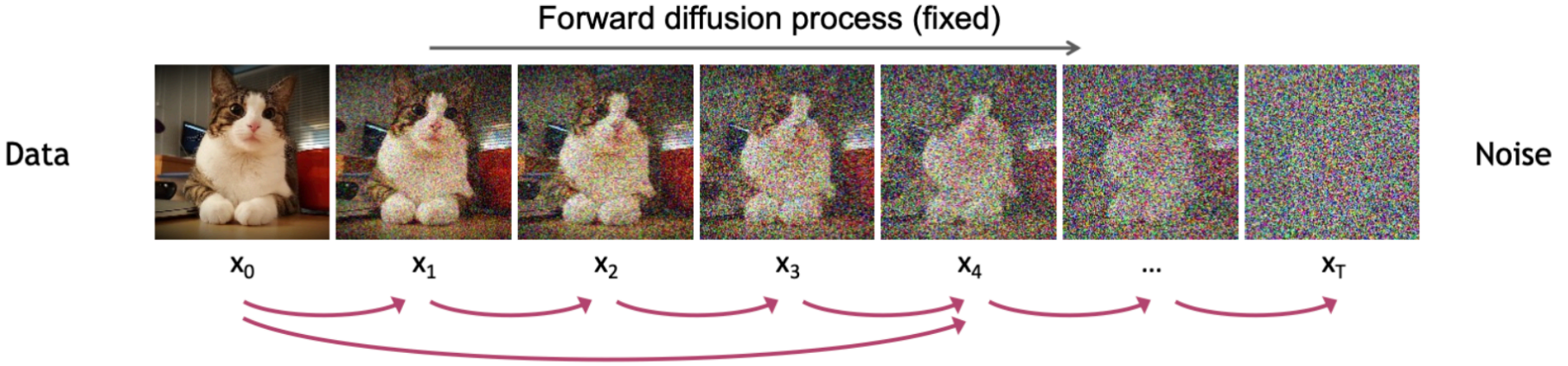


$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

- No learning
  - Sample random numbers from a gaussian
  - Add them with some scaling to all of the pixels
  - Repeat T times
- End product — pure noise $\mathcal{N}(0,\mathbf{I})$

Forward Process

- Noising is a Markov process and noise is Gaussian
- Can jump to any stage directly



Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

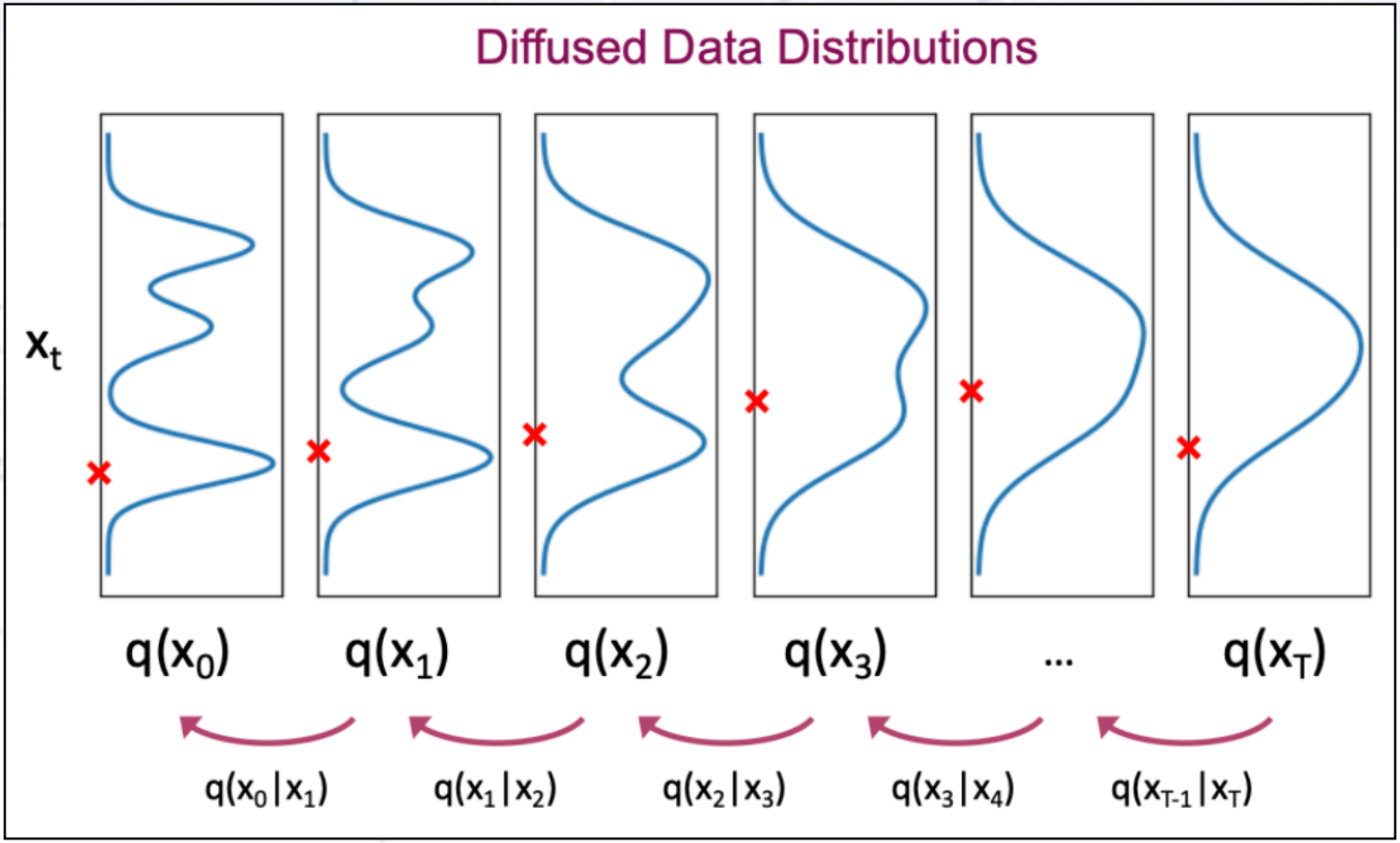
For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

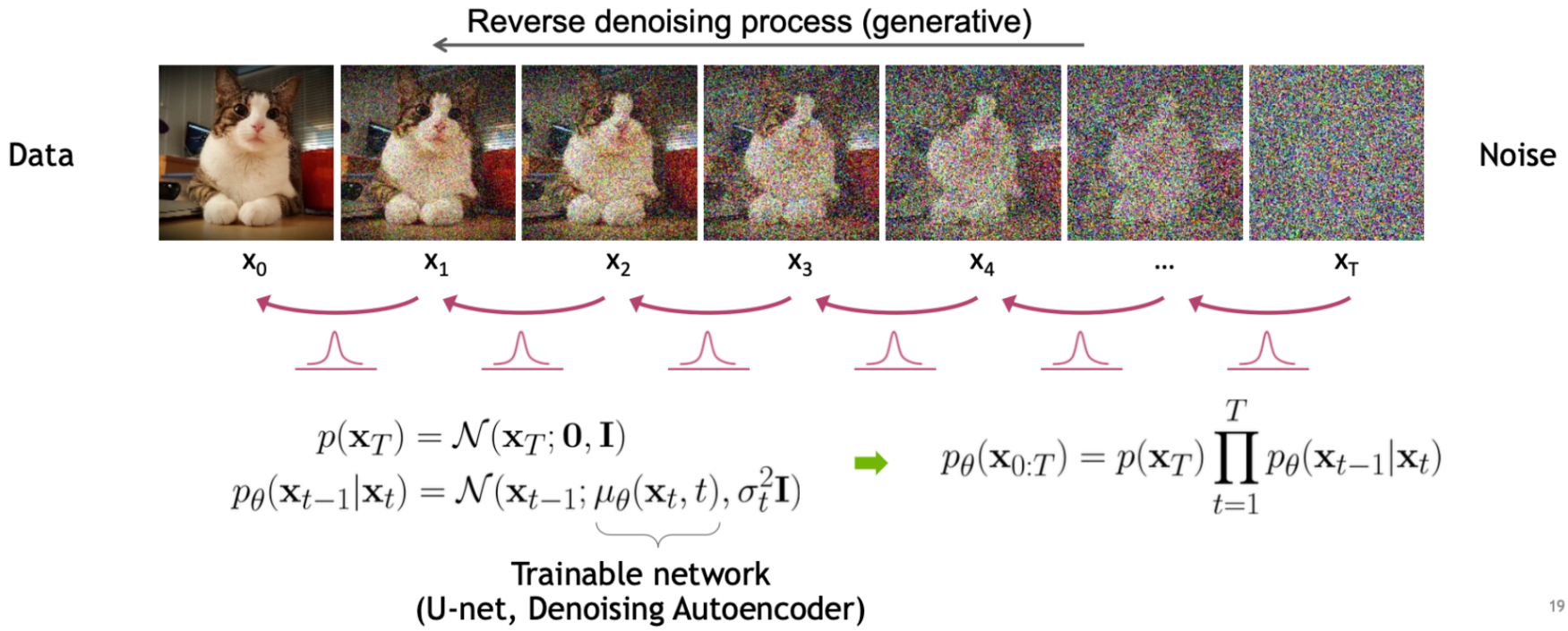
See https://cvpr2023-tutorial-diffusion-models.github.io/

Reverse (de-noising) process

We want to obtain $q(x_{t-1} \mid x_t)$.
If β_t is small, $q(x_{t-1} \mid x_t)$ will be Gaussian.
This means that we can approximate $q(x_{t-1} \mid x_t)$ with a neural network.



Reverse (de-noising) process



- In practice, people predict amount of the noise added on the previous step directly
- $\mathscr{L} = \mathrm{MSE}(\epsilon_{t-1} - \epsilon_\theta(x_t, t))$, where ϵ_t is added noise

Diffusion Summary

To some (simple) degree, the diffusion model setup can be summarised by the below two algorithms.

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
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: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

The training part is to train the algorithm to estimate the amount of noise added at each step.

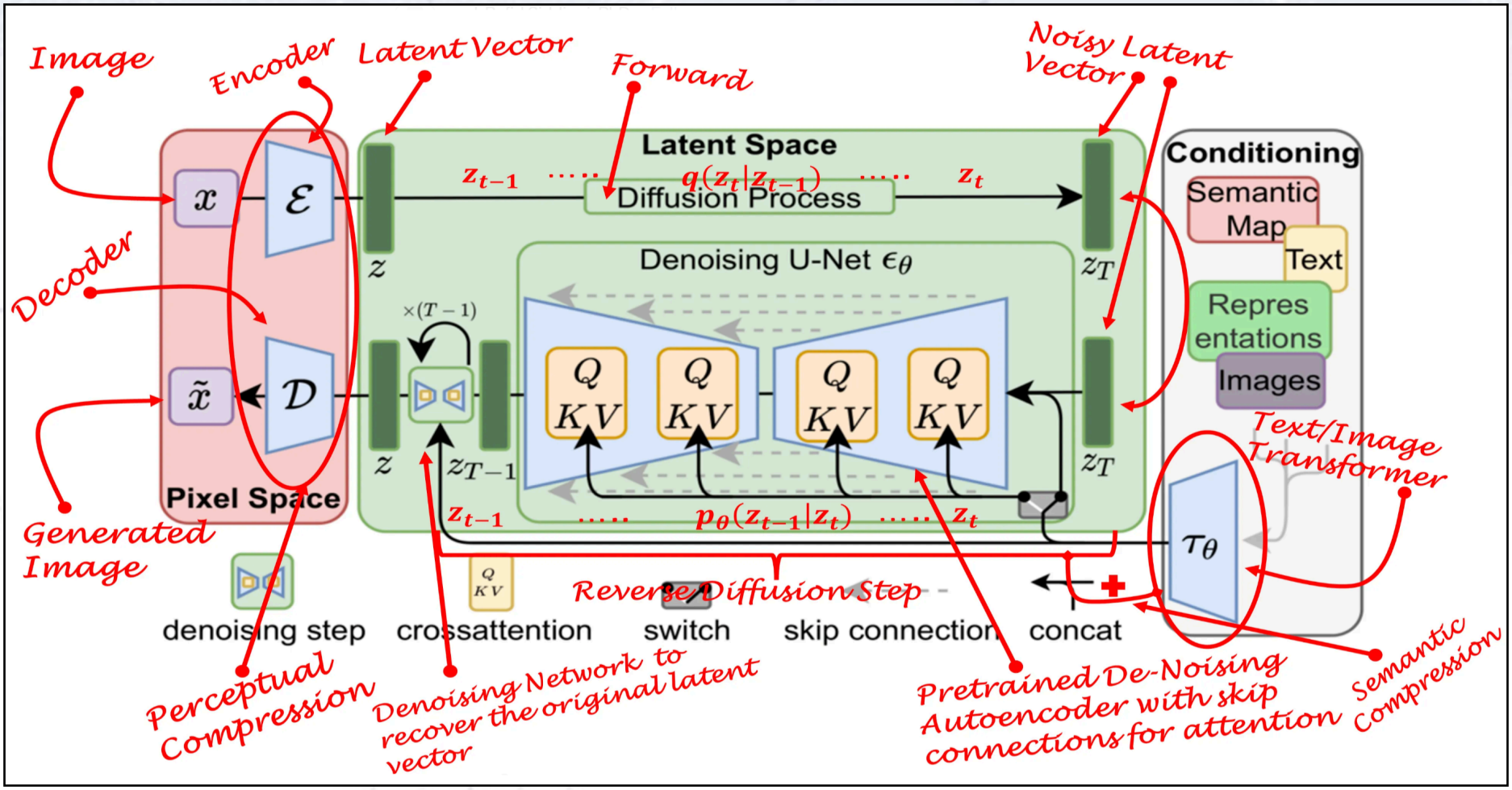
The sampling part is to produce new images from sampling the noisy space and then using the trained part.

Diffusion example problem

Take images, and apply the diffusion forward and and then backwards in pixel space… i.e. NOT LATENT diffusion.

Latent Diffusion "Overview"

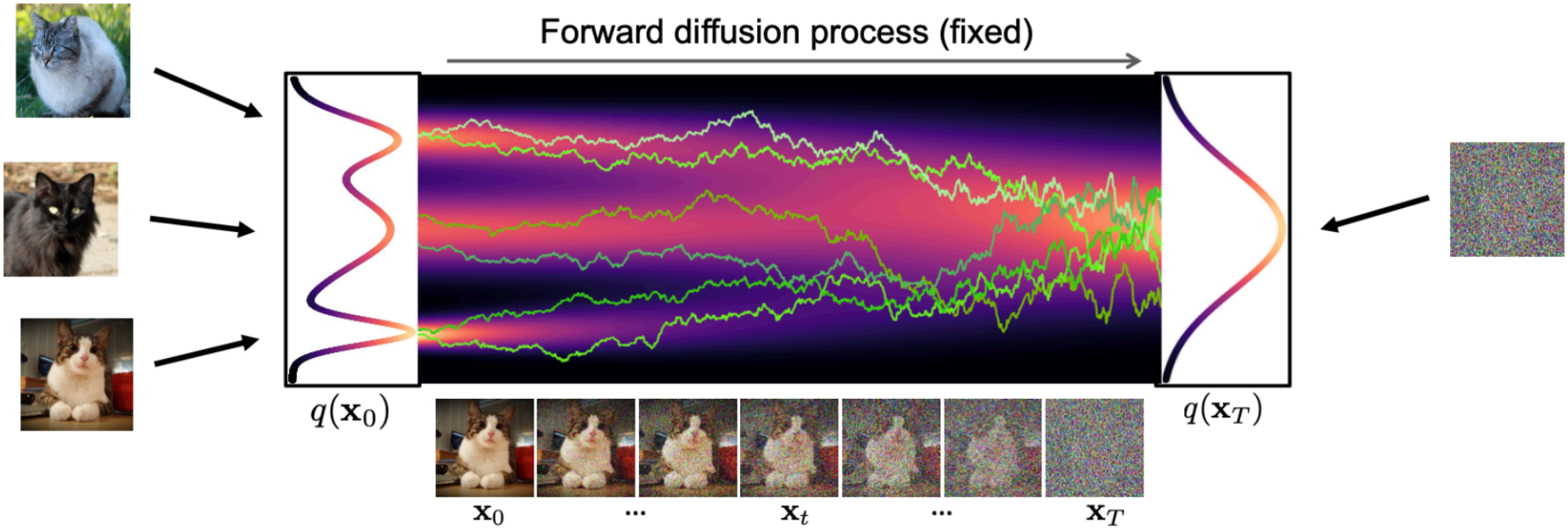
Here is an attempt at labelling the whole process (& Latent) and all the parts…



From https://towardsdatascience.com/what-are-stable-diffusion-models-and-why-are-they-a-step-forward-for-image-generation-aa1182801d46

Denoising Diffusion Implicit Model

This specific type of model can be considered a Stochastic Differential Equation (SDE).

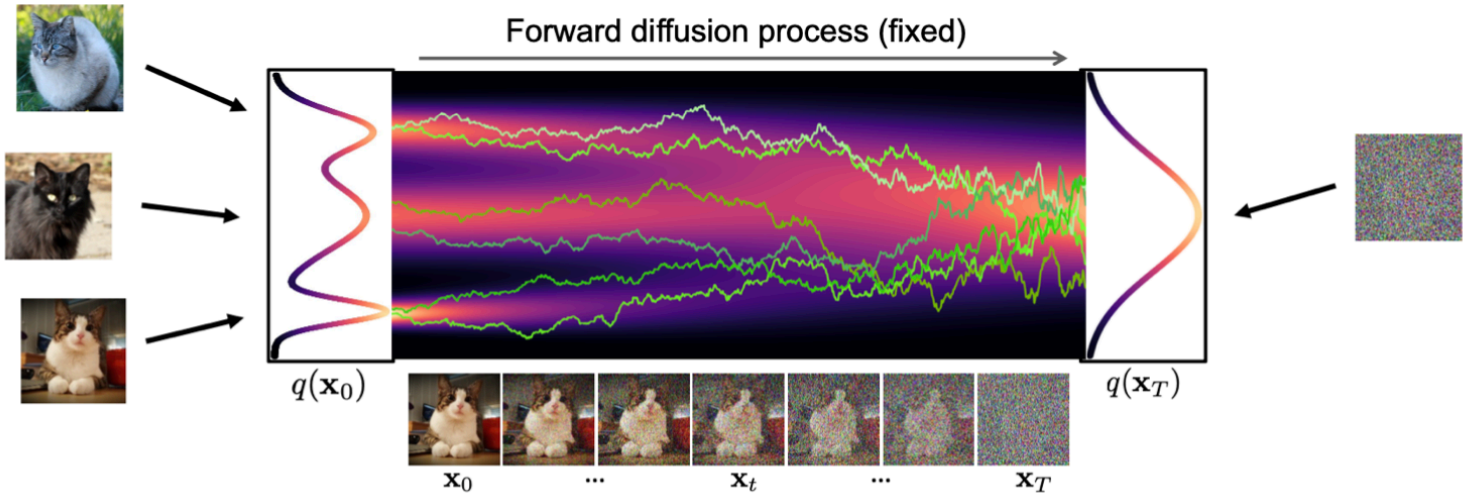


Forward diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t}_{\text{drift term (pulls towards mode)}} + \underbrace{\sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t}_{\text{diffusion term (injects noise)}}$$

Denoising Diffusion Implicit Model

This specific type of model can be considered a Stochastic Differential Equation (SDE).



Forward diffusion SDE:

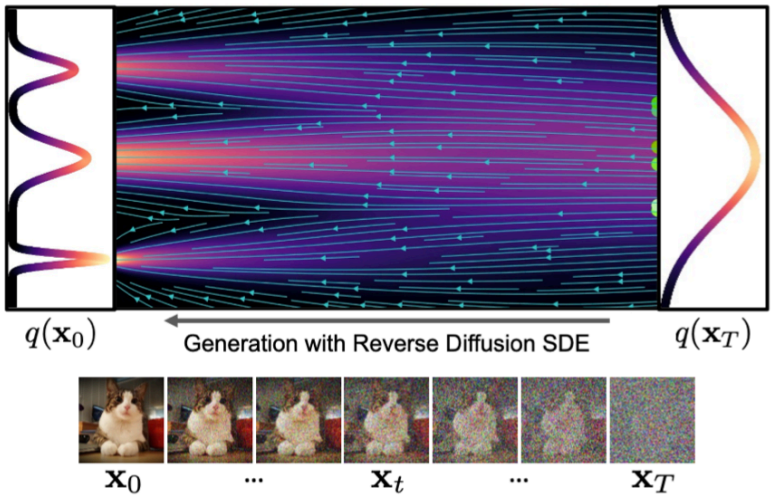
$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t}_{\text{drift term (pulls towards mode)}} + \underbrace{\sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t}_{\text{diffusion term (injects noise)}}$$

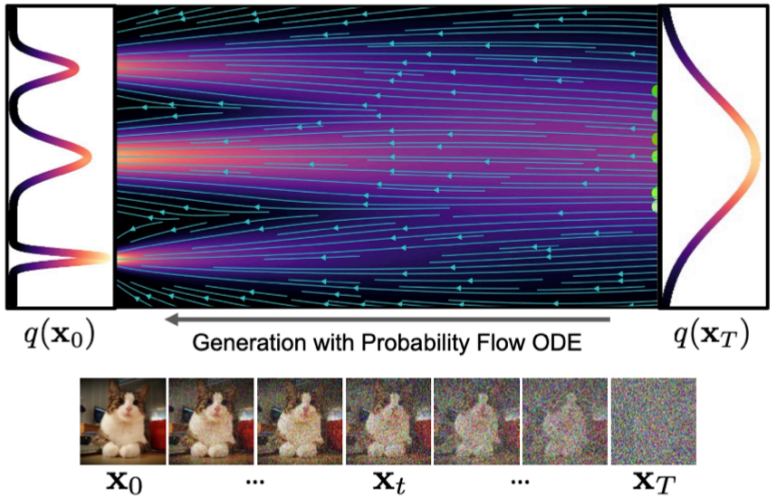
Reverse diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = \overbrace{\left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\underbrace{\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)}_{\text{“Score Function”}}\right]}^{\text{drift term}}\mathrm{d}t + \overbrace{\sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t}^{\text{diffusion term}}$$

Denoising Diffusion Implicit Model

Putting the two parts together as ODE/SDE, one makes it possible to use advanced (existing) algorithms for solving these.





- **Generative Reverse Diffusion SDE (stochastic):**

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

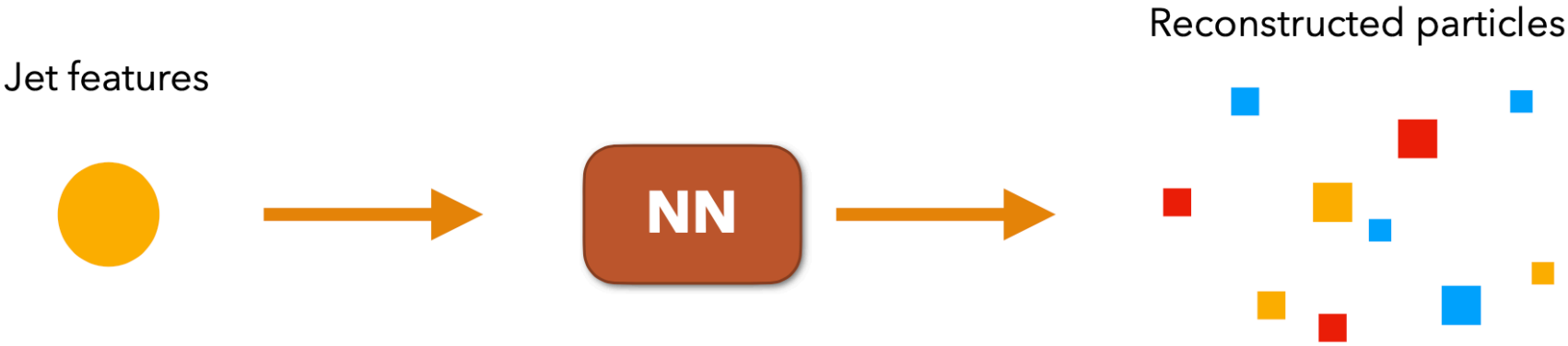
- **Generative Probability Flow ODE (deterministic):**

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

- Accelerate generation
- Enables use of advanced ODE/SDE solvers

Example uses of DDIM

Apart from images, it can be used in many other places for generation…

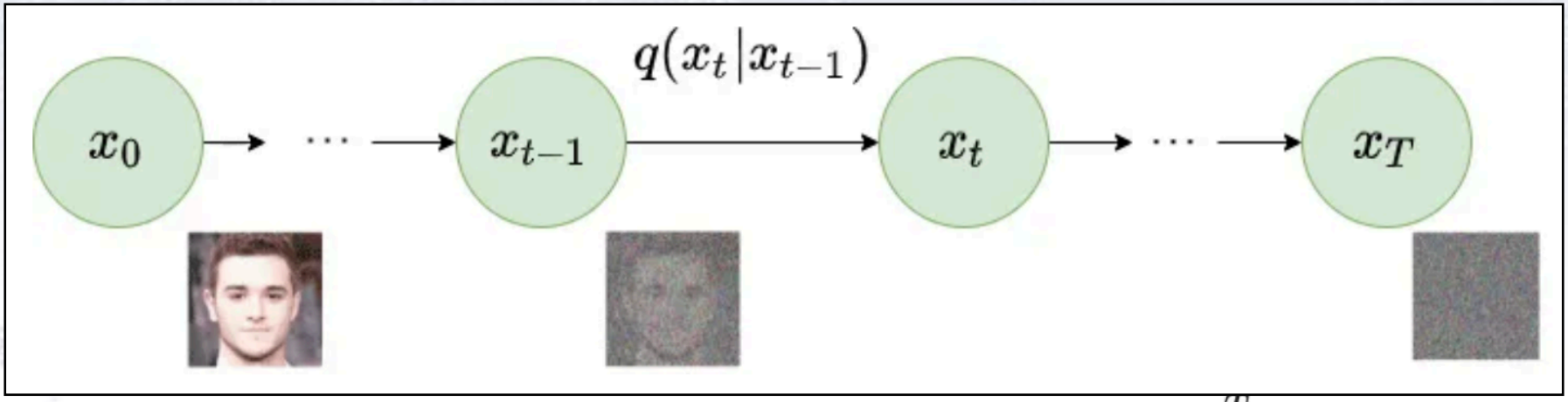


- JetNet dataset — gluon jets, up to 30 constituents
- Number of constituents known in advance
- DDIM formulation — solve SDE/ODE to generate data

Bonus Slides

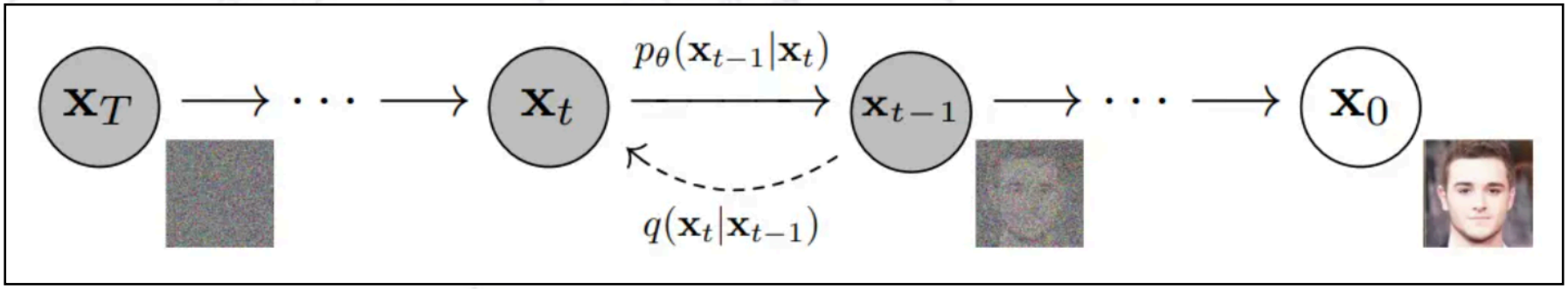
Diffusion

Forward (noising) diffusion process (simple!):



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

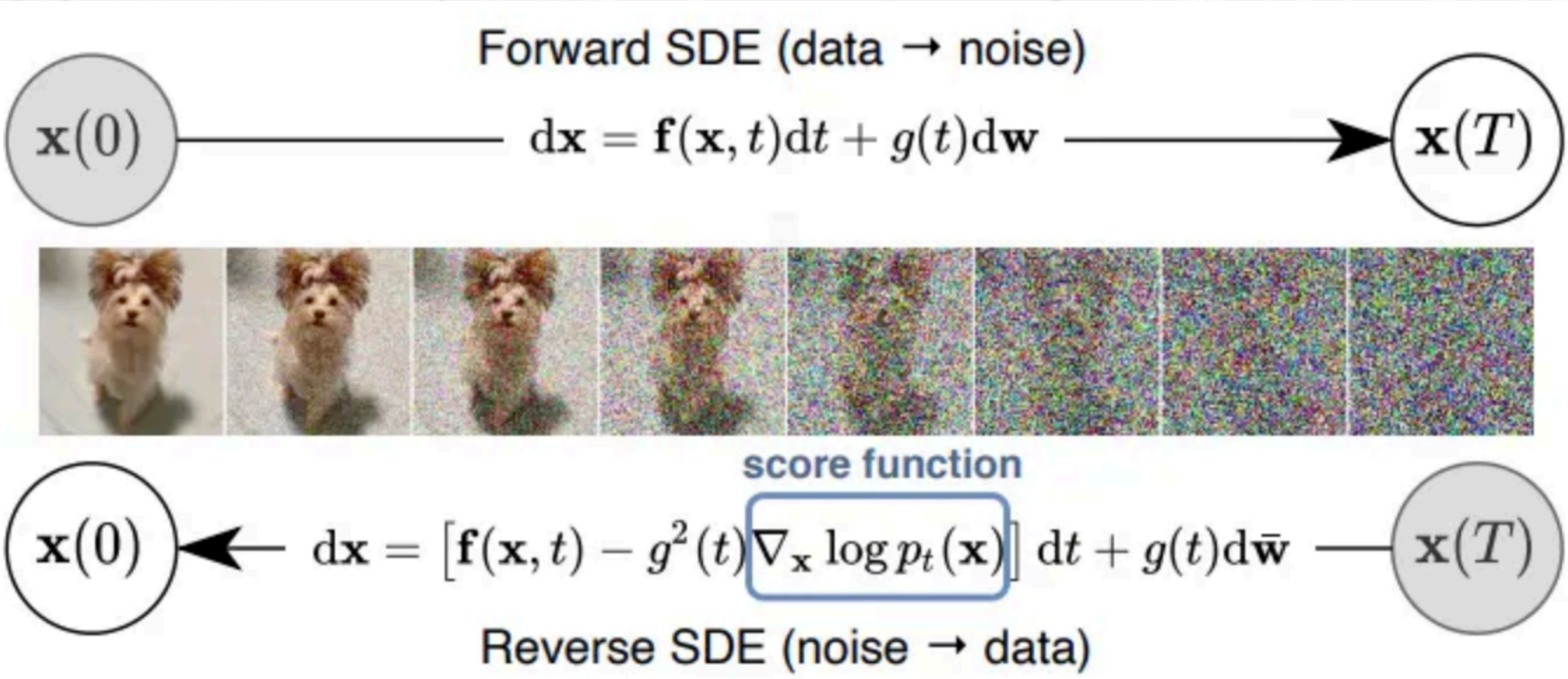
Reverse (de-noising) diffusion process (not simple!):



From https://www.superannotate.com/blog/diffusion-models

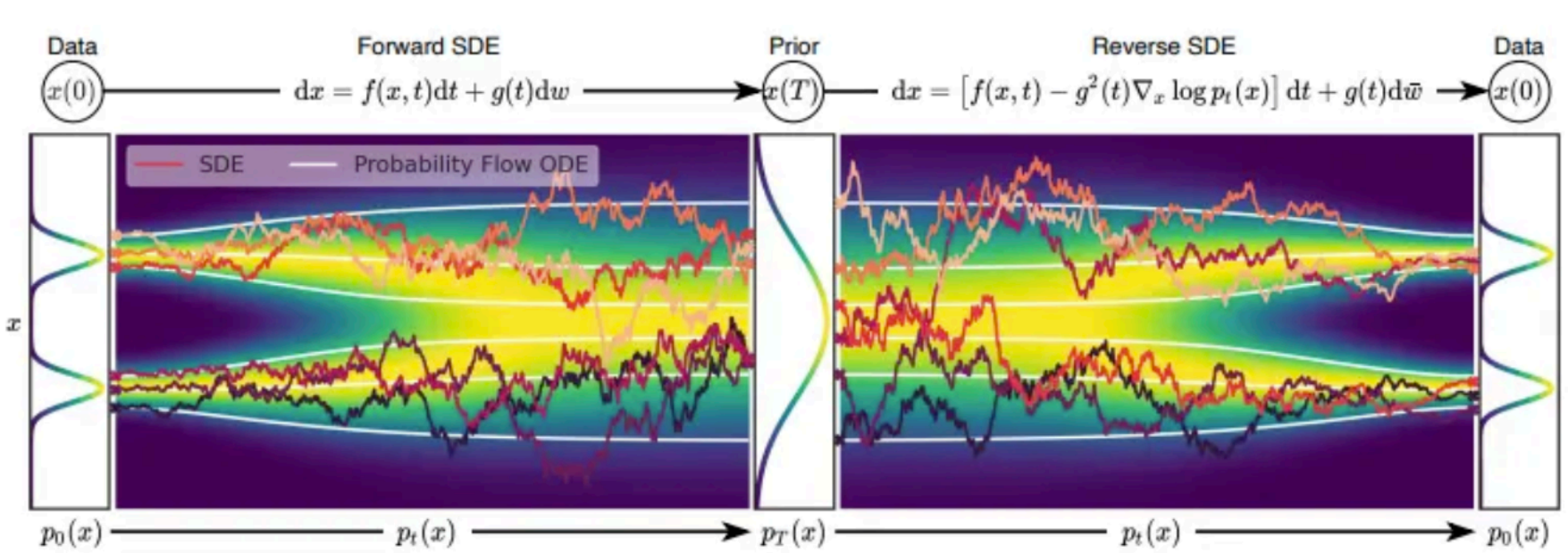
Stochastic Differential Equation

The process can be considered a Stochastic Differential Equation, which in small steps transforms the image to noise.



Score-based Generative Models

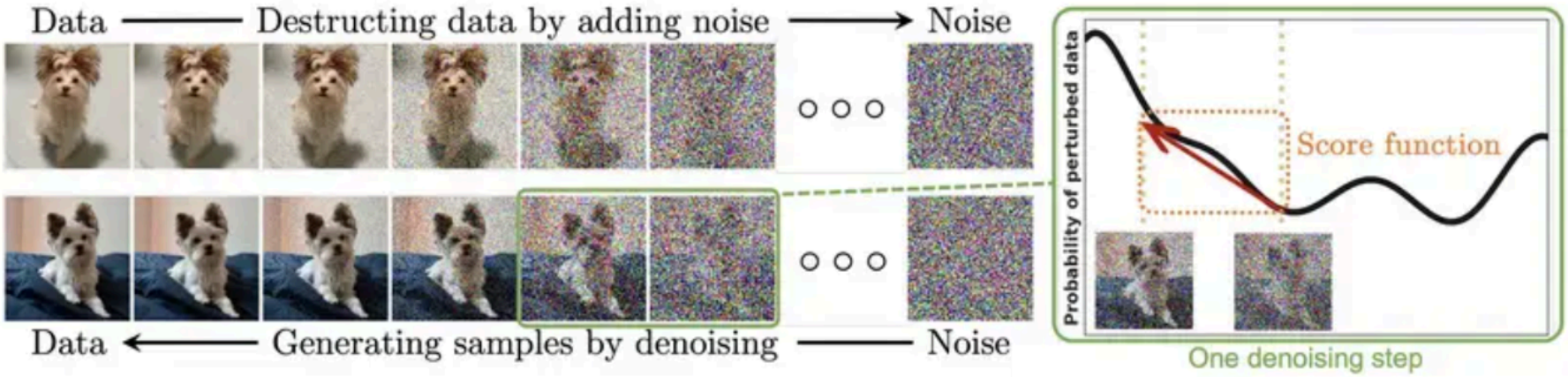
In the reverse step, the model tries to reverse the process of noise addition. SGMs teaches the model to start from noisy data and progressively remove noise to reveal a more clear and detailed image.



Thus, the SDE maps data to a noise distribution (the prior), and reverse this SDE for generative modelling.

Denoising diffusion probabilistic models

DDPMs are a specific type of Diffusion Model, that focuses on removing noise from data in a probabilistic way.



During training, they learn how noise is added to data over time and how to reverse this process to recover the original data. This involves using probabilities to make educated guesses about what the data looked like before noise was added.

This approach is essential for the model's capability to accurately reconstruct data, ensuring the outputs aren't just noise-free but also closely resemble the original data.

Denoising Diffusion Implicit Model

The scoring is slightly complicated…

Reverse diffusion SDE: $$\mathrm{d}\mathbf{x}_t = \left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\underbrace{\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)}_{\text{“Score Function”}}\right]\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\bar{\omega}_t$$

We can learn $\nabla_{x_t}\log q_t(x_t)$ with NN, however direct regression is not possible.

Instead, we can diffuse individual data points x_0. Diffused $q_t(x_t\,|\,x_0)$ is tractable.

Denoising score matching:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)}||\underbrace{\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}_{\text{neural network}} - \underbrace{\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t|\mathbf{x}_0)}_{\text{score of diffused data sample}}||_2^2$$

diffusion time t | data sample $\mathbf{x}_0$ | diffused data sample $\mathbf{x}_t$

“Variance Preserving” SDE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

$$q_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t;\gamma_t\mathbf{x}_0,\sigma_t^2\mathbf{I})$$

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\frac{1}{\sigma_t^2}||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)||_2^2$$

Same loss as before!