

# **Bsc Data Science for Responsible Business**

## **Deep Learning Course**

---

### **Diffusion Models**

**Emmanuel Dellandrea** - [emmanuel.dellandrea@ec-lyon.fr](mailto:emmanuel.dellandrea@ec-lyon.fr)

# Denoising Diffusion Probabilistic Models (DDPM)

## Overview



Dhariwal & Nichol, 2021



Source : <https://github.com/bentoml/stable-diffusion-bentoml>



Source : DALL-E 2

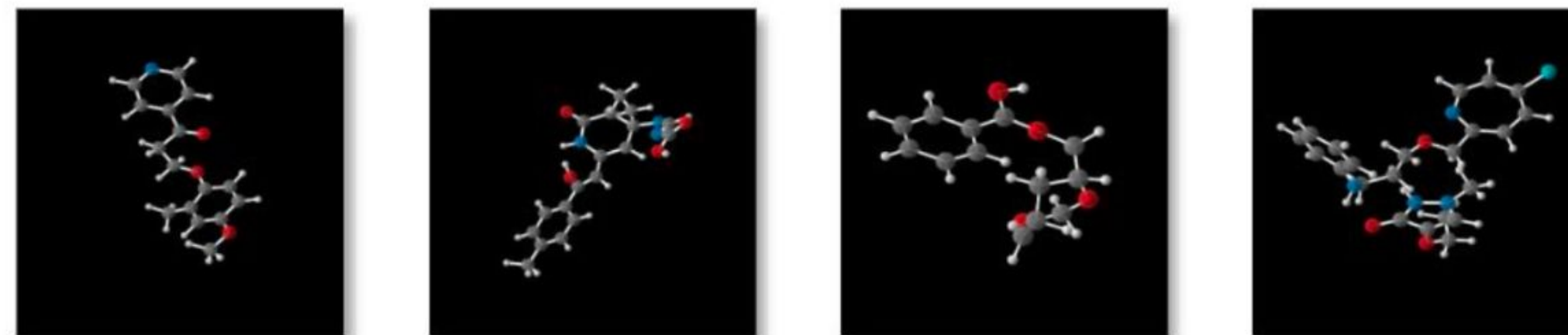
# DDPM overview



Source : <https://ai.meta.com/blog/emu-text-to-video-generation-image-editing-research/>

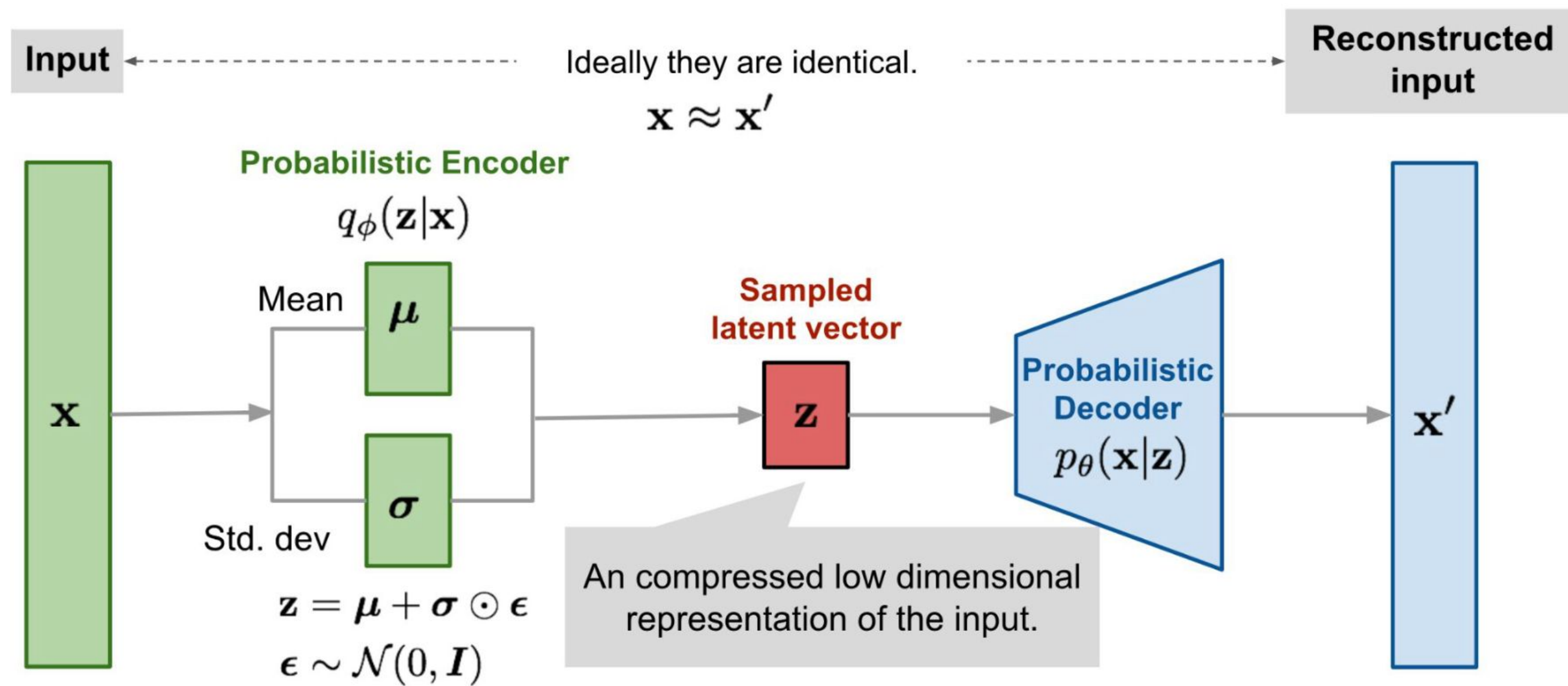


Source : <https://arxiv.org/pdf/2210.06978.pdf>

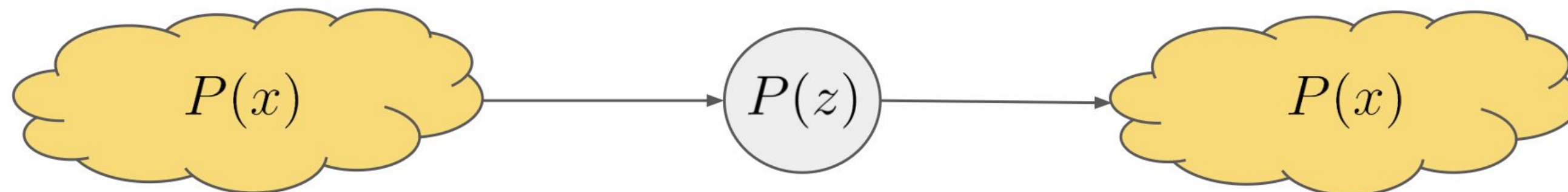


Source : <https://arxiv.org/pdf/2305.01140.pdf>

# Recap: VAE



Source <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>



# Recap: VAE

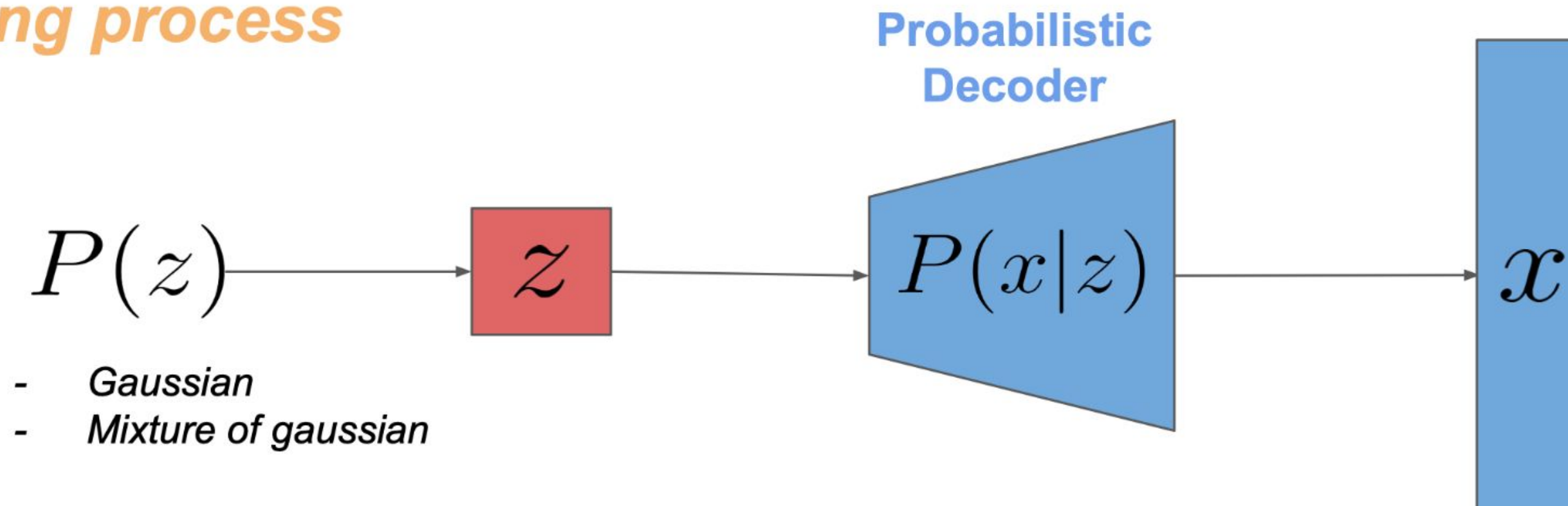
## Training process

metric

VAE COST FUNCTION

$$\log p_{\theta}(\mathbf{x}) - D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\theta}(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} \log p_{\theta}(\mathbf{x}|\mathbf{z}) - D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\theta}(\mathbf{z}))$$

## Sampling process



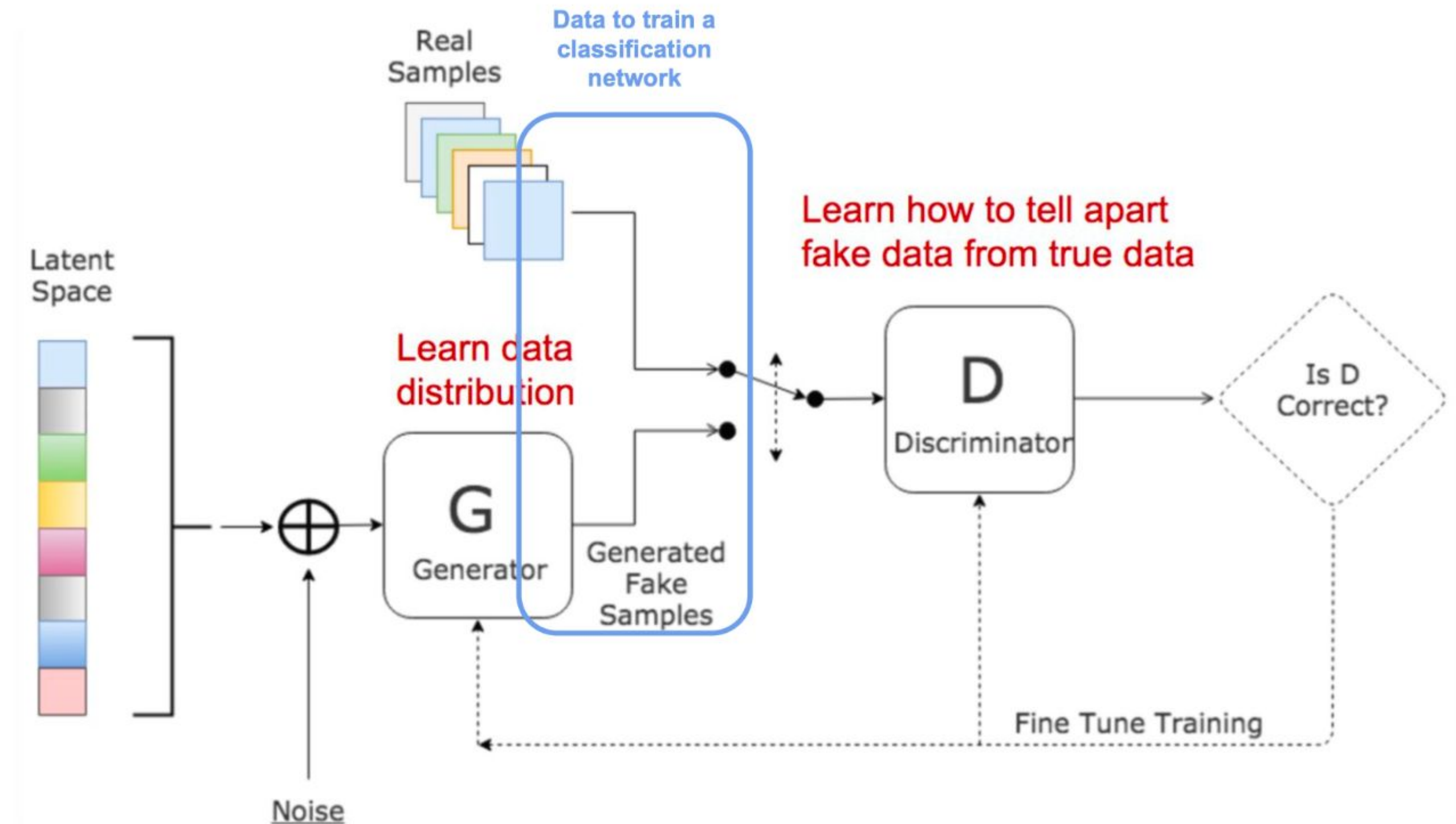
# Recap: GAN

## 2 networks in opposition :

- Generator
- Discriminator

## Ideal solution :

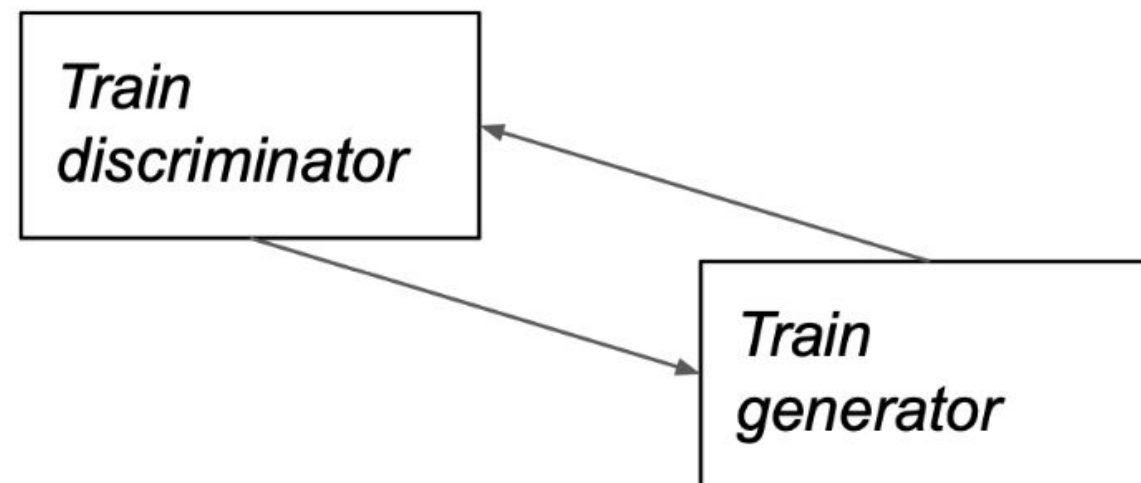
- Generator  $\sim P(x|z)$
- Discriminator =  $\frac{1}{2}$



Source  
<https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html>

# Recap: GAN

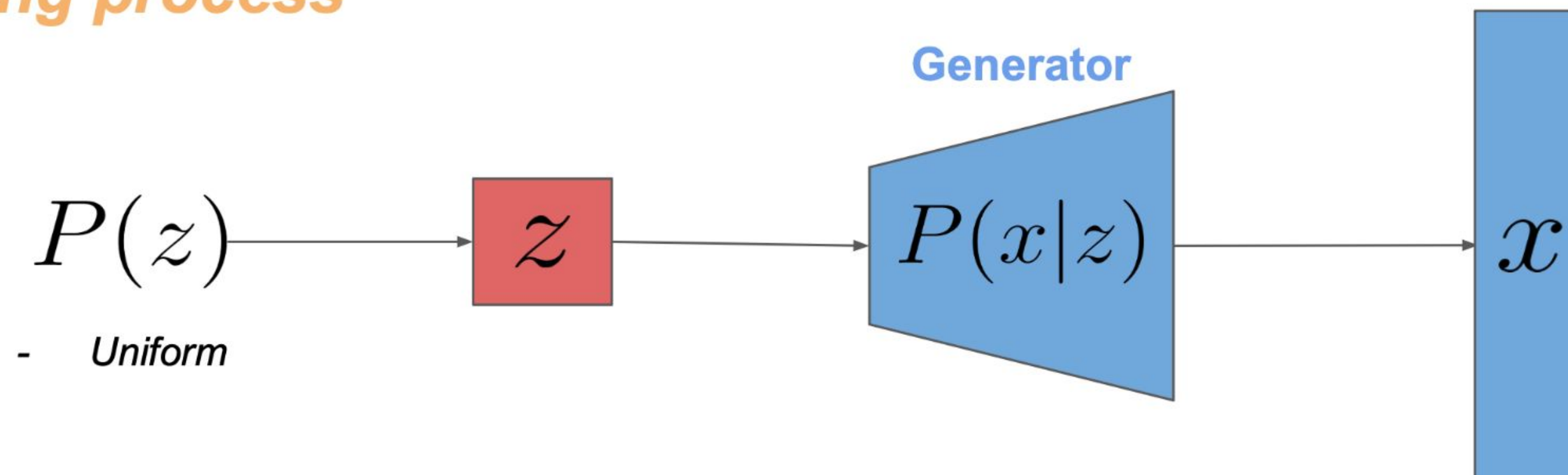
## Training process



### GAN COST FUNCTION

$$\min_G \max_D L(D, G) = \mathbb{E}_{x \sim p_r(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \\ = \mathbb{E}_{x \sim p_r(x)} [\log D(x)] + \mathbb{E}_{x \sim p_g(x)} [\log(1 - D(x))]$$

## Sampling process



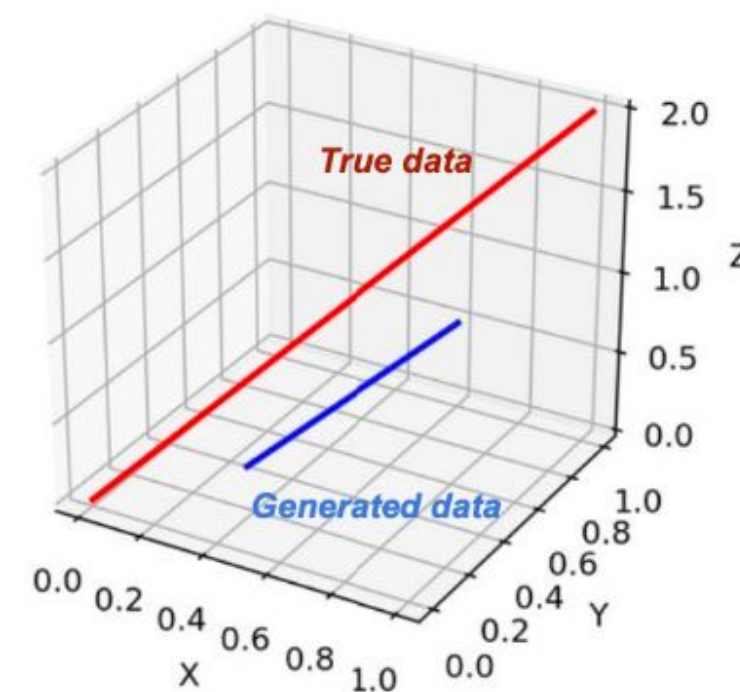
# Recap: GAN

GAN convergence problems !

**No convergence** due to the nature of the problem (MinMax)

?

**Vanishing gradient** due to discriminant being too perfect, generator can't train anymore



Source  
<https://lilianweng.github.io/posts/2017-08-20-gan/>

**Mode collapse** due to generator learning only some good examples instead of the whole data distribution

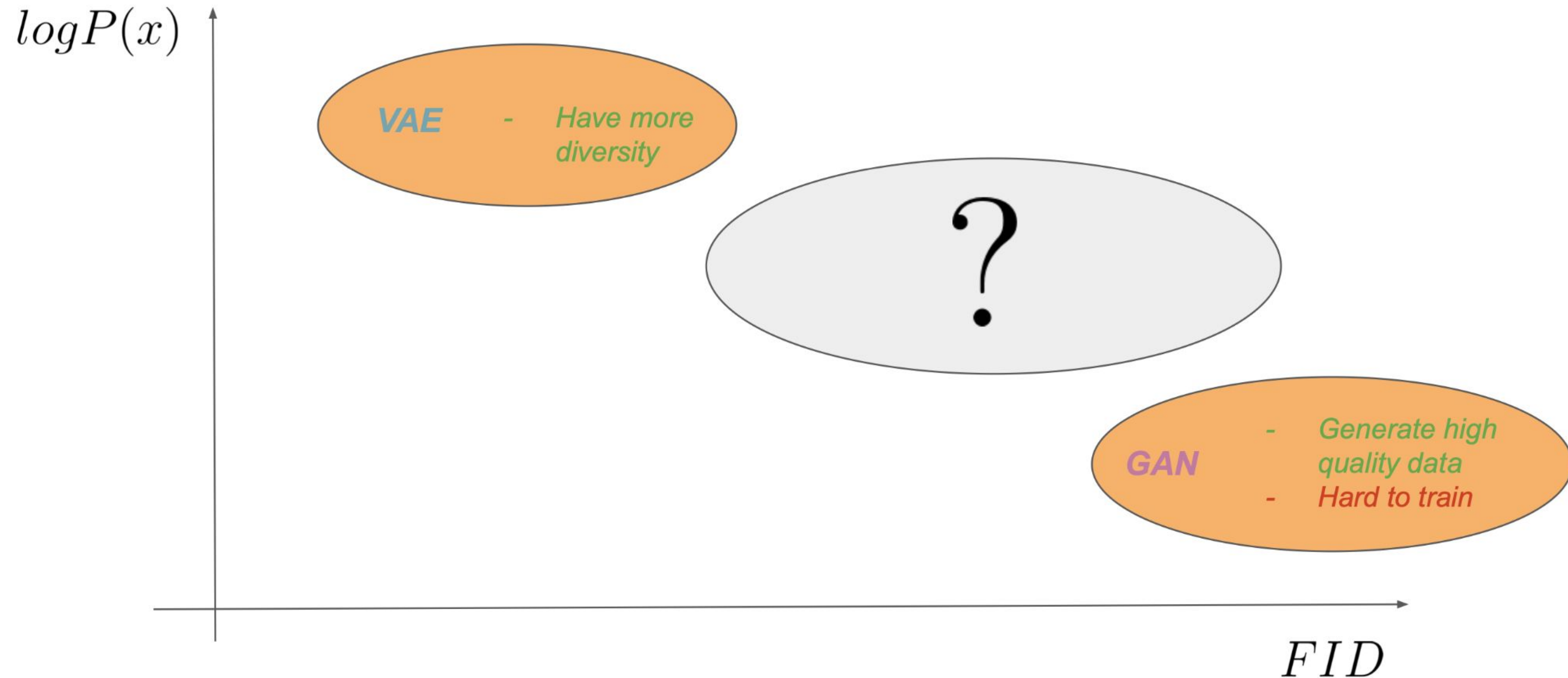
Generated Data



True Data



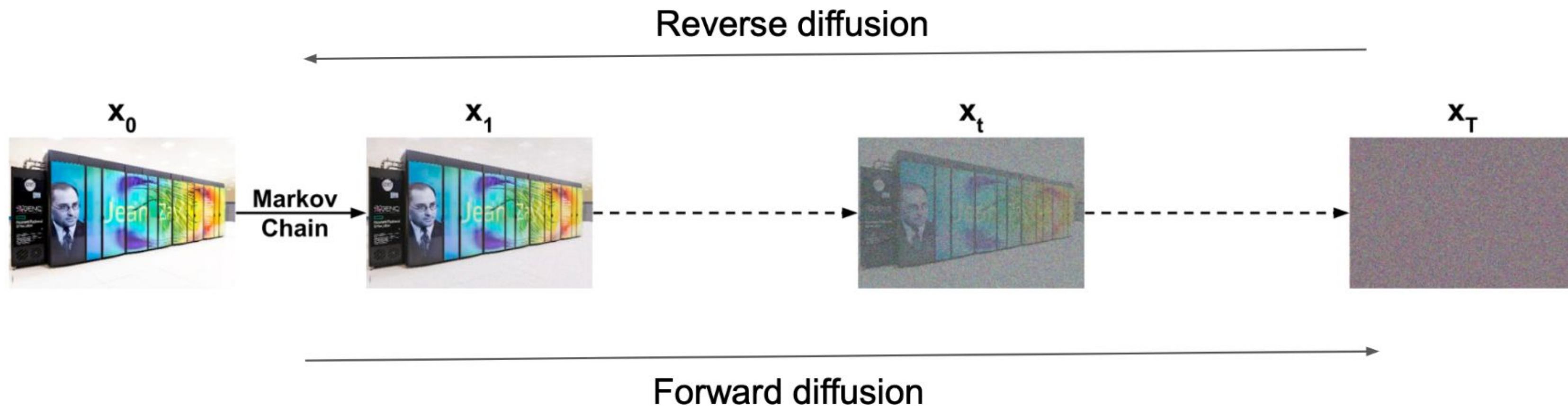
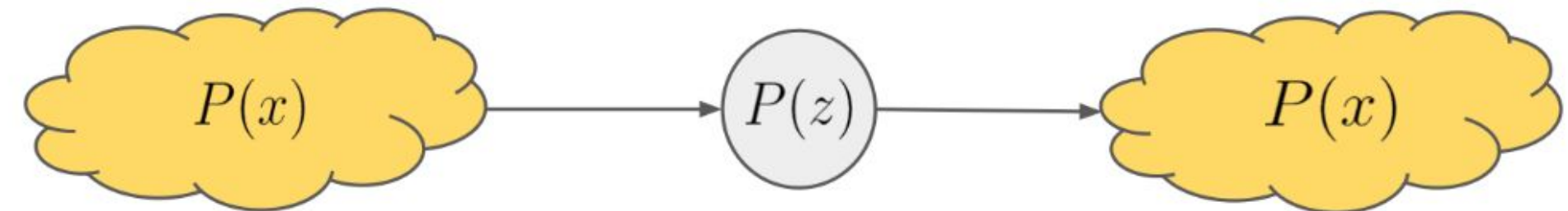
# GAN vs VAE



# DDPM principle

**Denoising diffusion probabilistic models (DDPM) consist of two processes:**

- Forward diffusion process that **gradually adds noise** to input
- Reverse denoising process that learns to generate data by denoising



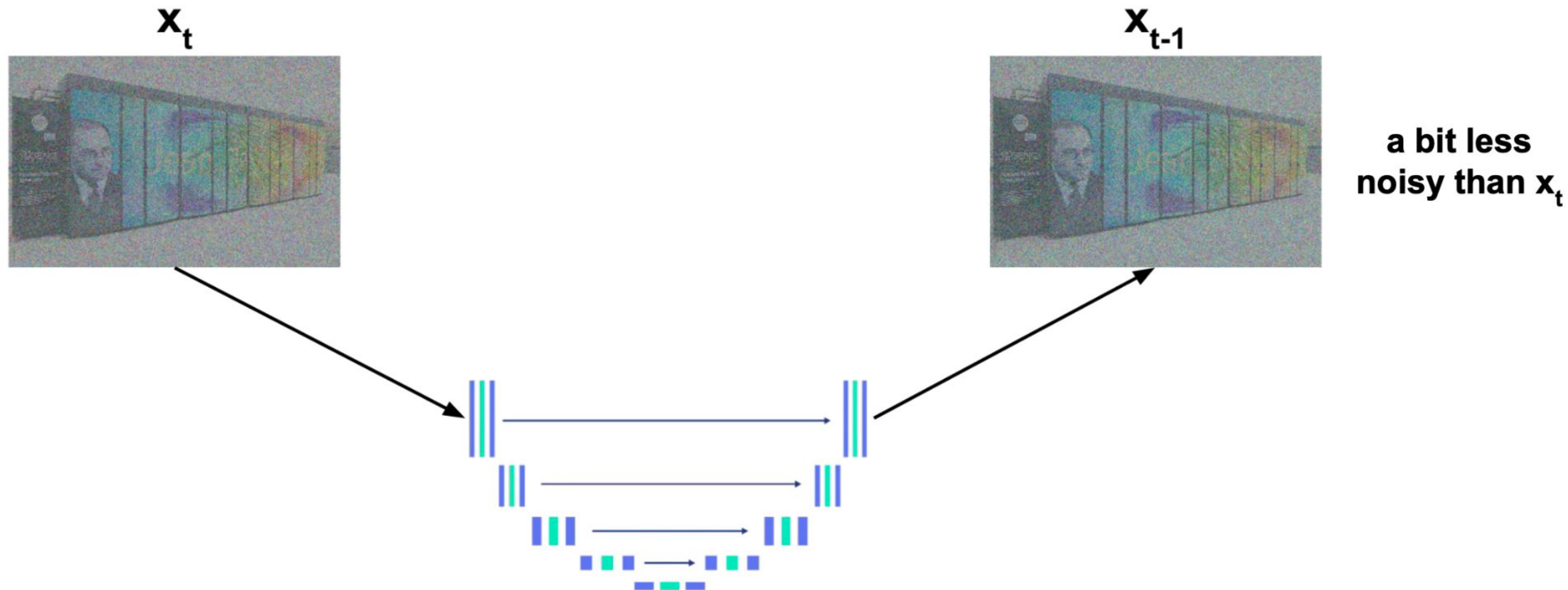
## Forward Diffusion Process



Denoising Diffusion Probabilistic  
Models (DDPM)

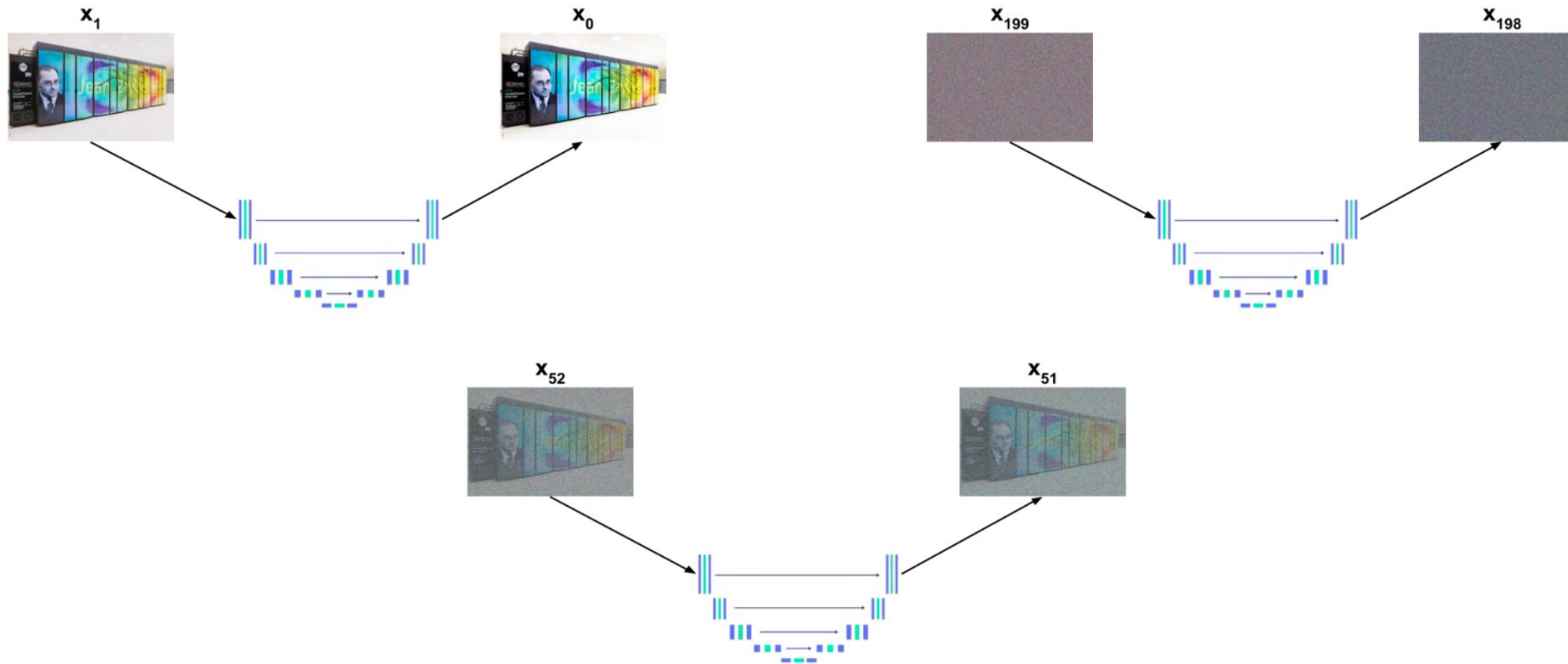
**Here we choose  $T=1000$ , but it can be different values (it's an hyperparameter)**

## Reverse Diffusion Process



We train a model to predict  $x_{t-1}$  from  $x_t$

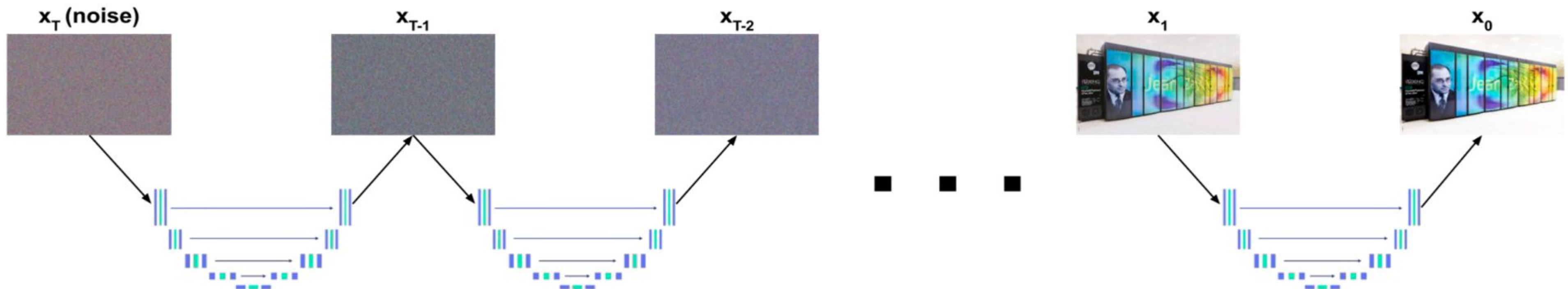
## Reverse Diffusion Process



The same model must predict every  $x_{t-1}$  from  $x_t$   $Model(x_t, t)$

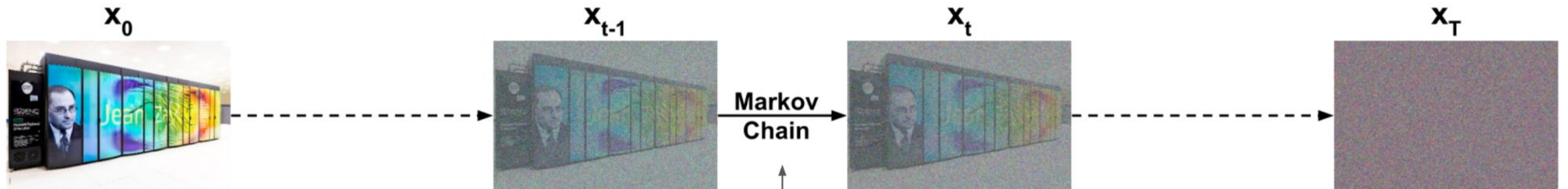
# DDPM principle

**After the training, the model will generate images from Gaussian noise by following a sampling process :**



```
torch.randn(4)
```

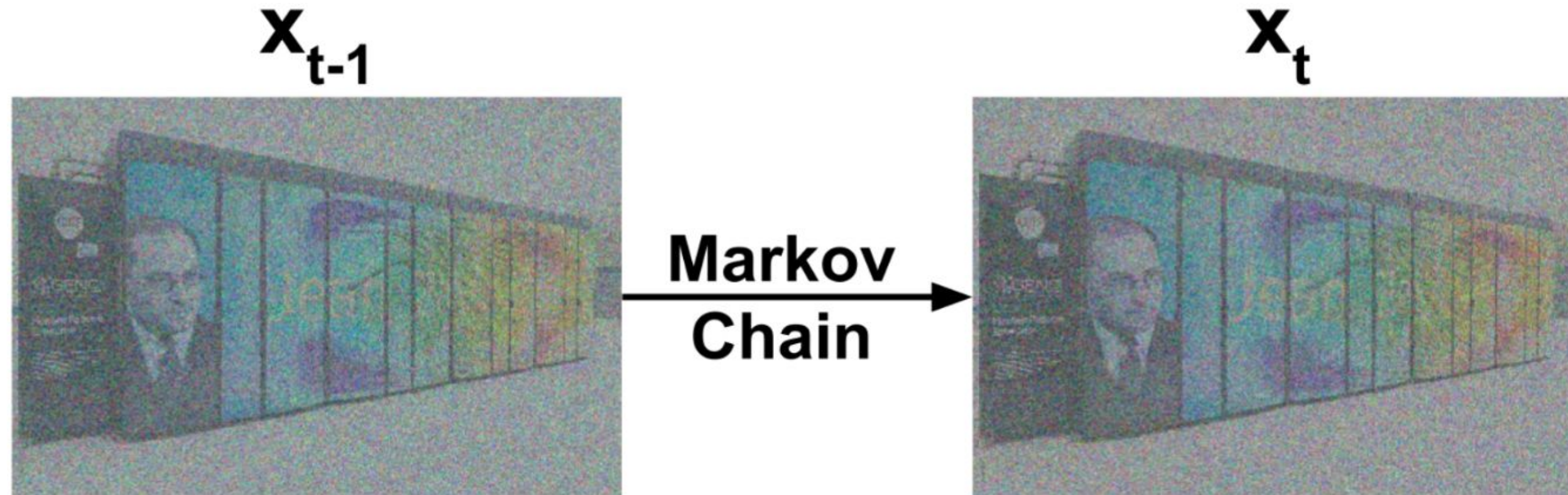
# DDPM forward diffusion



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

$(\beta_t)$  is a hyperparameter

# DDPM forward diffusion



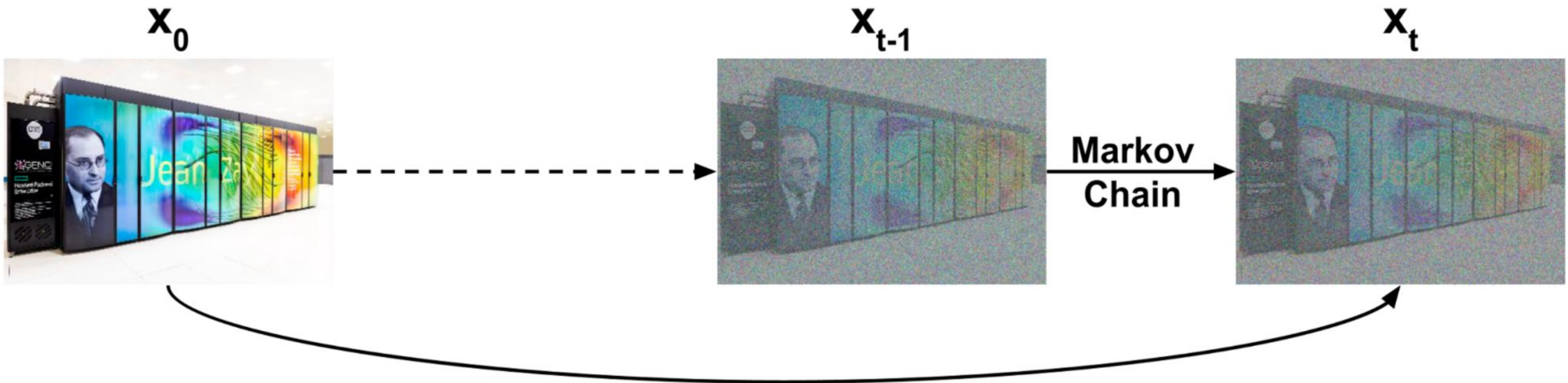
$$\mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t I)$$

$$x_t = \sqrt{1 - \beta_t}x_{t-1} + \sqrt{\beta_t}z_{t-1}$$

$$\text{where } z_{t-1} \sim \mathcal{N}(0, I)$$

```
torch.randn(4)
```

# DDPM forward diffusion



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

$$\text{where } \bar{\alpha}_t = \prod_{i=1}^t (1 - \beta_i)$$

# DDPM forward diffusion

$\mathbf{x}_t$



$= \sqrt{\bar{\alpha}_t}$

$\mathbf{x}_0$



$+$

$\sqrt{1 - \bar{\alpha}_t}$

$\mathbf{Z}$  (Gaussian noise)



$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} z$$

**So we can sample a noised image at any time step directly from the original image**

# DDPM computation

$$q(x_0)$$

Data distribution

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

Noisy data by a small  
gaussian noise



$$q(x_t|x_0)$$

$$q(x_{t-1}|x_t, x_0)$$



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

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)$$

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}\mathbf{x}_t$$

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

$$q(x_t|x_0)$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}z_t$$

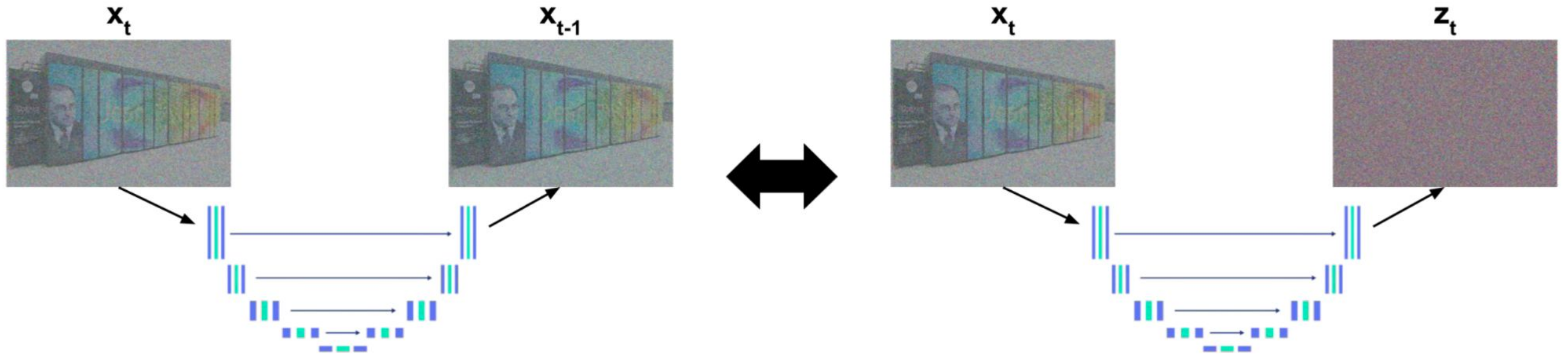


$$q(x_{t-1}|x_t, z_t)$$

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\mathbf{z}_t)$$

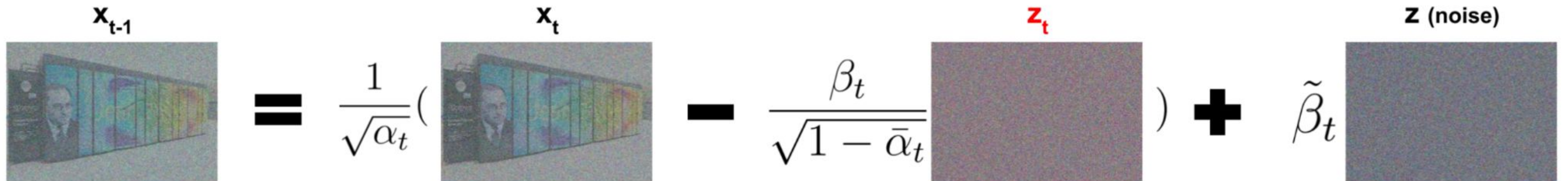
$\mathbf{z}_t$  ???

# DDPM reverse diffusion



We can predict  $x_{t-1}$  by predicting  $z_t$

# DDPM reverse diffusion

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \mathbf{z}_t \right) + \tilde{\beta}_t \mathbf{z} \text{ (noise)}$$


$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \mathbf{z}_t \right) \approx \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \mathbf{z}_{\theta}(\mathbf{x}_t, \mathbf{t}) \right)$$

*Our network*

# DDPM reverse diffusion

$$\frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \mathbf{z}_\theta(\mathbf{x}_t, t) \right) \quad x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} z_t$$

$$L_{\text{VLB}} = L_T + L_{T-1} + \dots + L_0$$

$$L_t = \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[ \frac{(1 - \alpha_t)^2}{2\alpha_t(1 - \bar{\alpha}_t) \|\boldsymbol{\Sigma}_\theta\|_2^2} \|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_t, t)\|^2 \right]$$

$$L_t^{\text{simple}} = \mathbb{E}_{t \sim [1, T], \mathbf{x}_0, \epsilon_t} \left[ \|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2 \right]$$

---

## Algorithm 1 Training

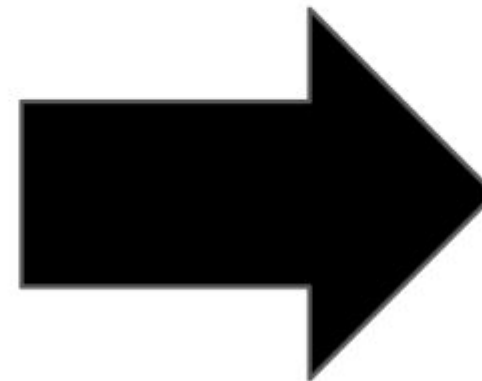
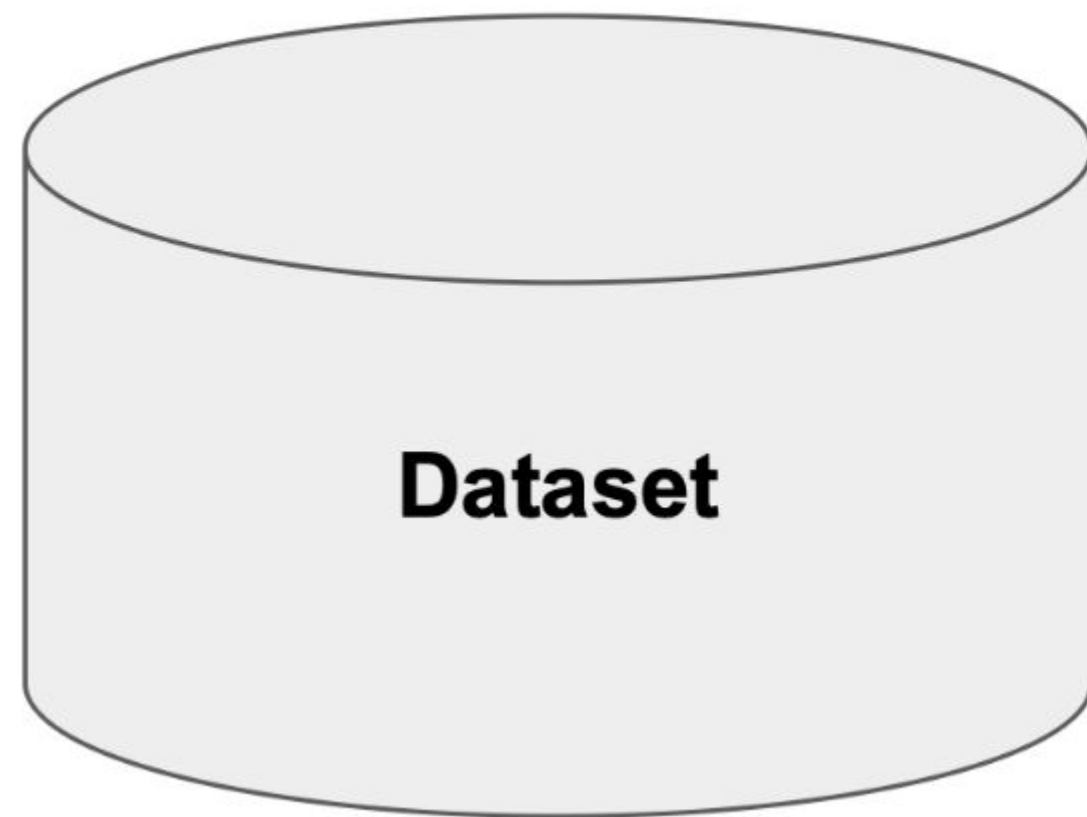
---

- 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
$$\nabla_{\theta} \left\| \epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t) \right\|^2$$
- 6: **until** converged

---

<https://arxiv.org/abs/2006.11239>

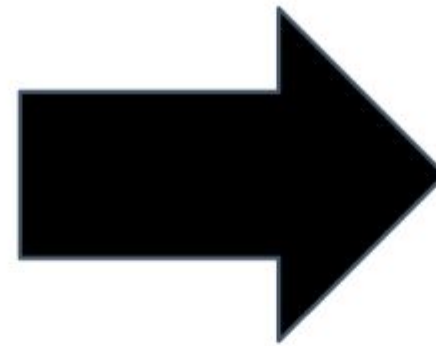
# DDPM training



# DDPM training

**Uniform  
distribution**

Between 1 and T



**$t = 50$**

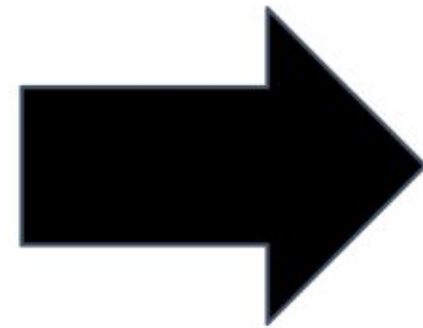
$x_0$



# DDPM training

**Gaussian  
distribution**

Same shape than  $x_0$



$\mathbf{z}_t (= \epsilon)$





$\mathbf{x}_0$



**$t = 50$**

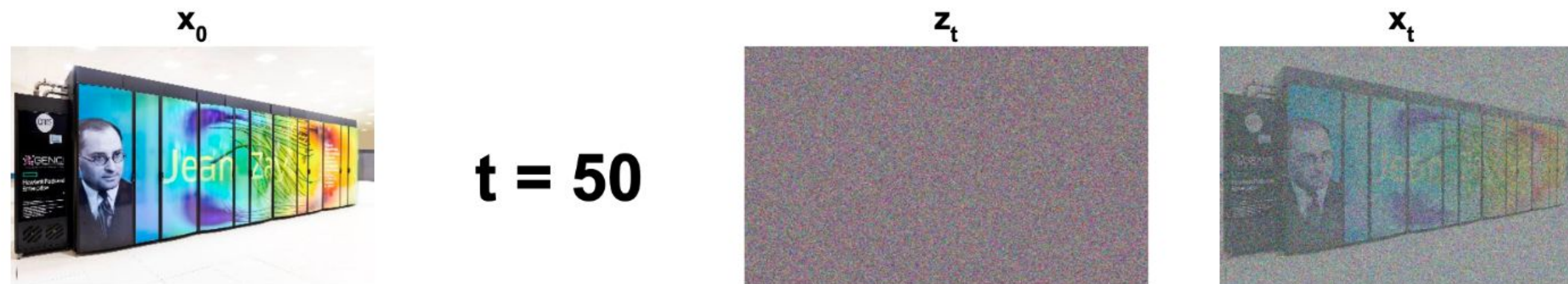
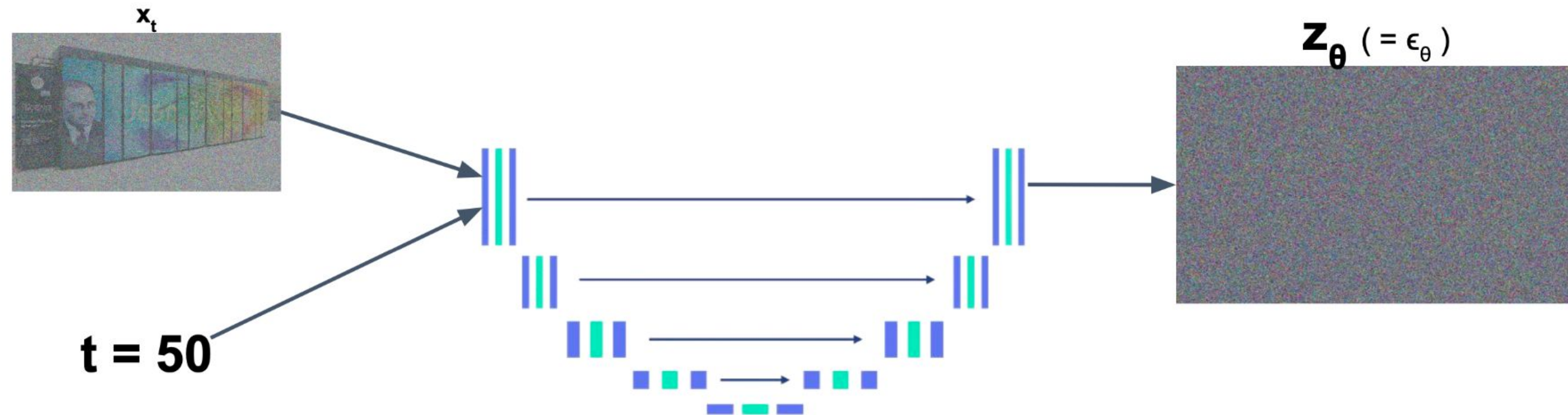
# DDPM training

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \mathbf{z}_t$$


$$\mathbf{x}_0 \quad \mathbf{z}_t$$


**t = 50**

# DDPM training

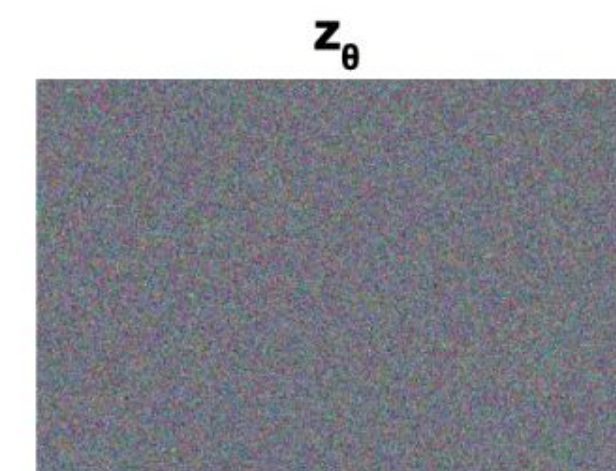
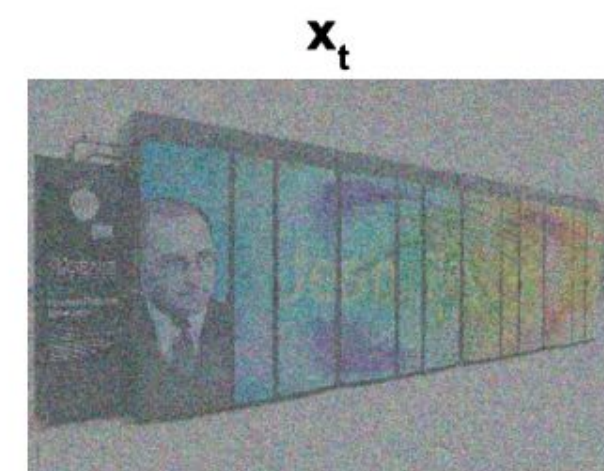
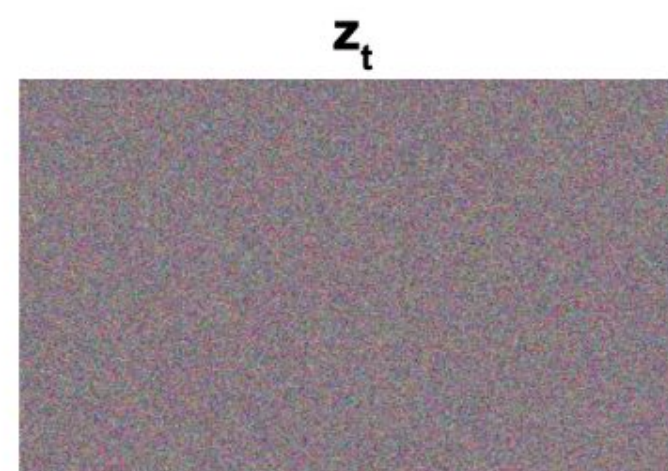


# DDPM training

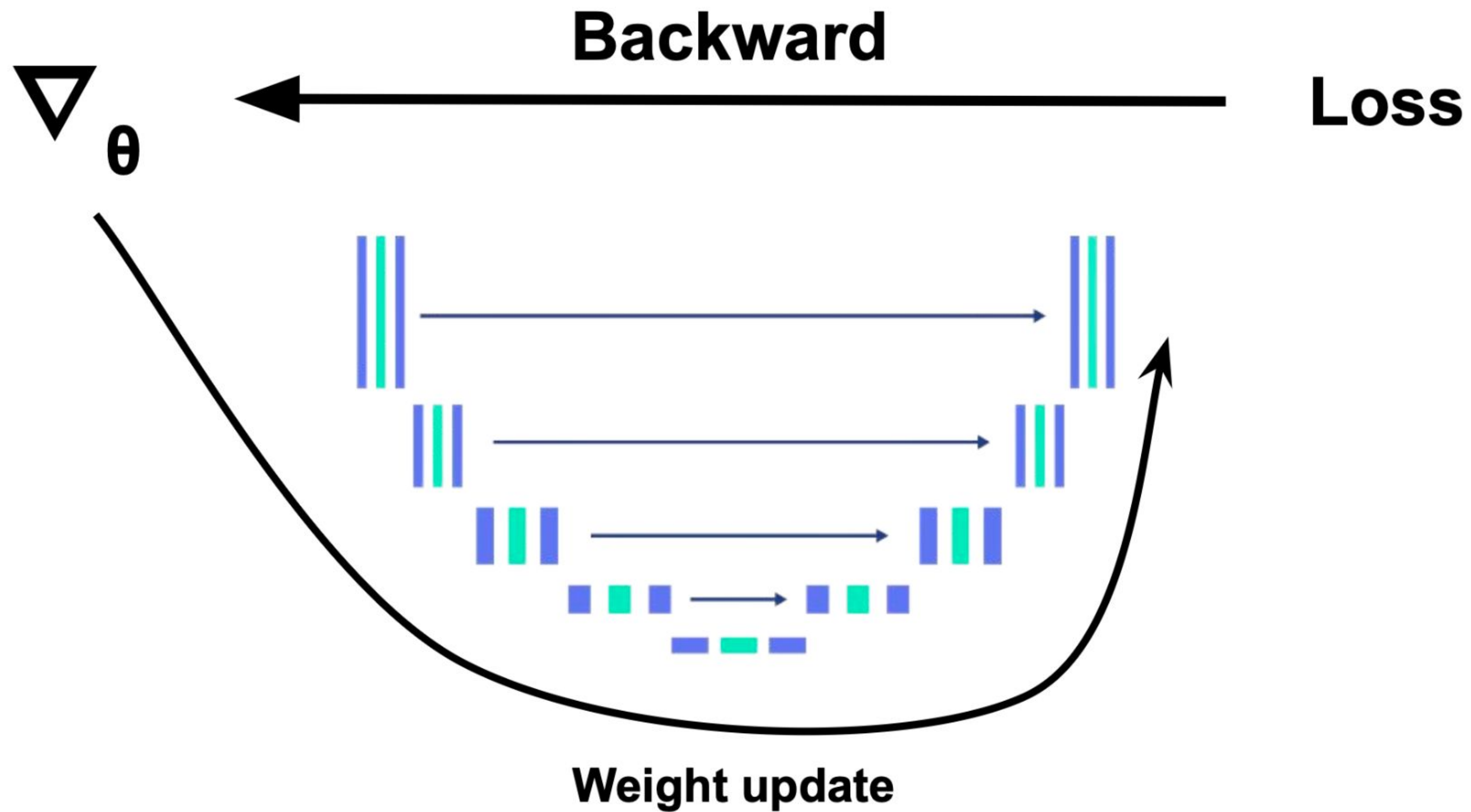
$$\text{Loss} = \left\| z_t - z_\theta \right\|^2$$



$t = 50$



# DDPM training



**and repeat !**

**It was just 1 iteration.**

---

## 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**  $\mathbf{x}_0$

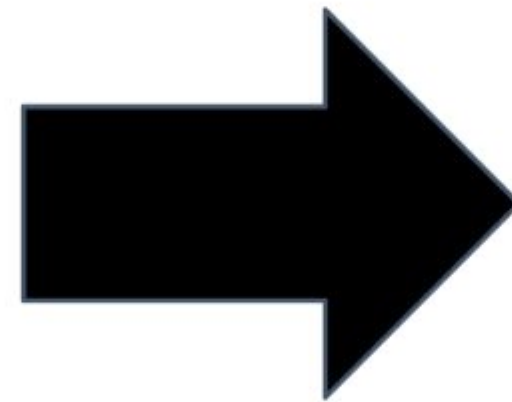
---

<https://arxiv.org/abs/2006.11239>

# DDPM sampling

**Gaussian  
distribution**

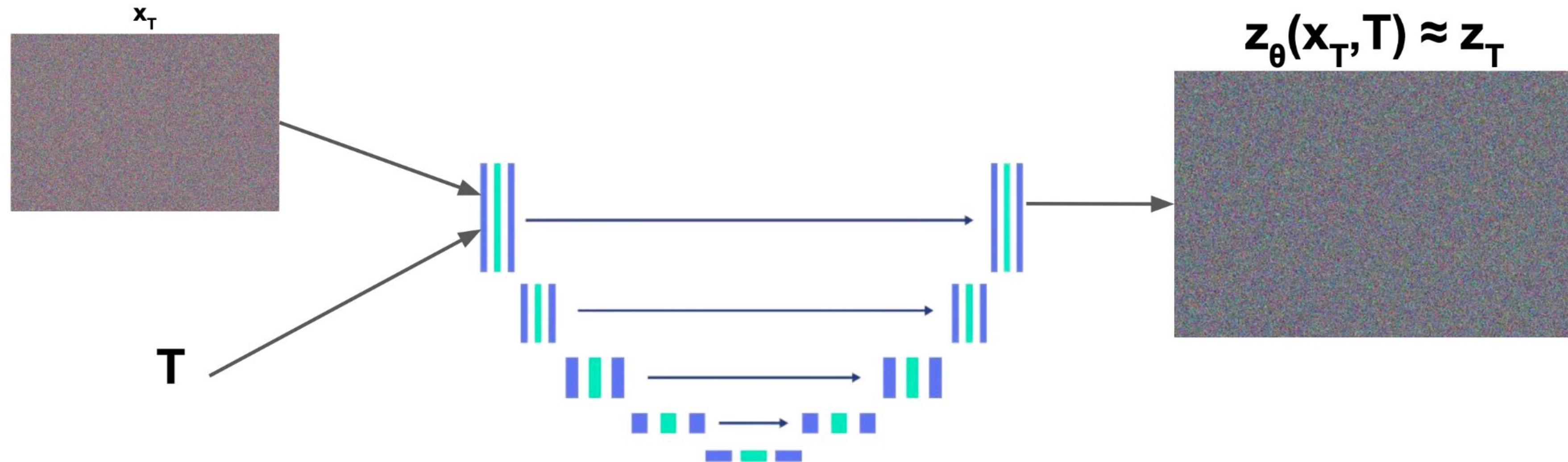
Same shape than training  
dataset images



$\mathbf{x}_T$



# DDPM sampling



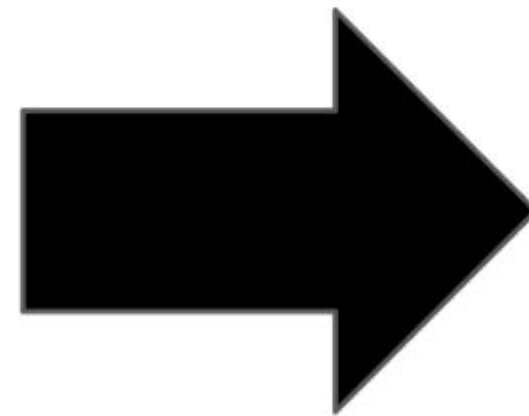
**Reminder:**

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t} z_t$$


# DDPM sampling

**Gaussian  
distribution**

Same shape than training  
dataset images



**z (noise)**

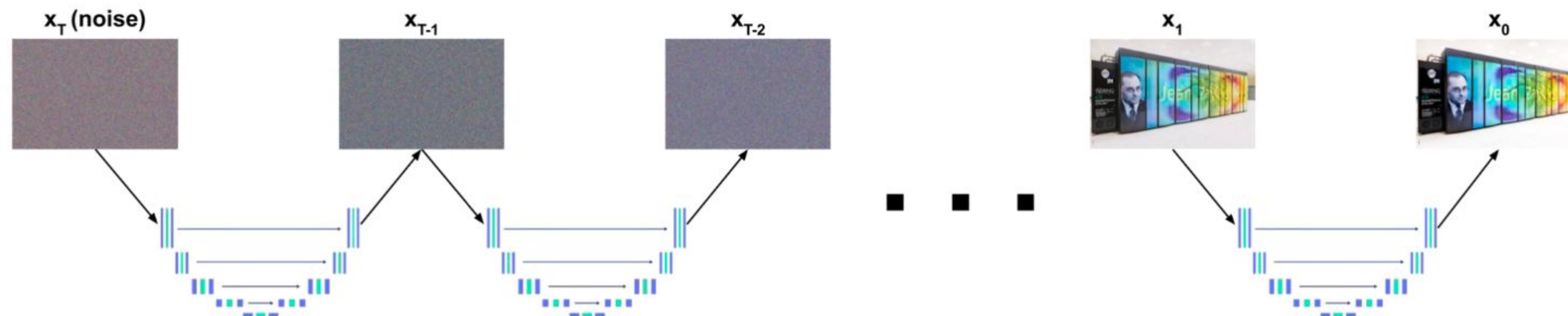


# DDPM sampling

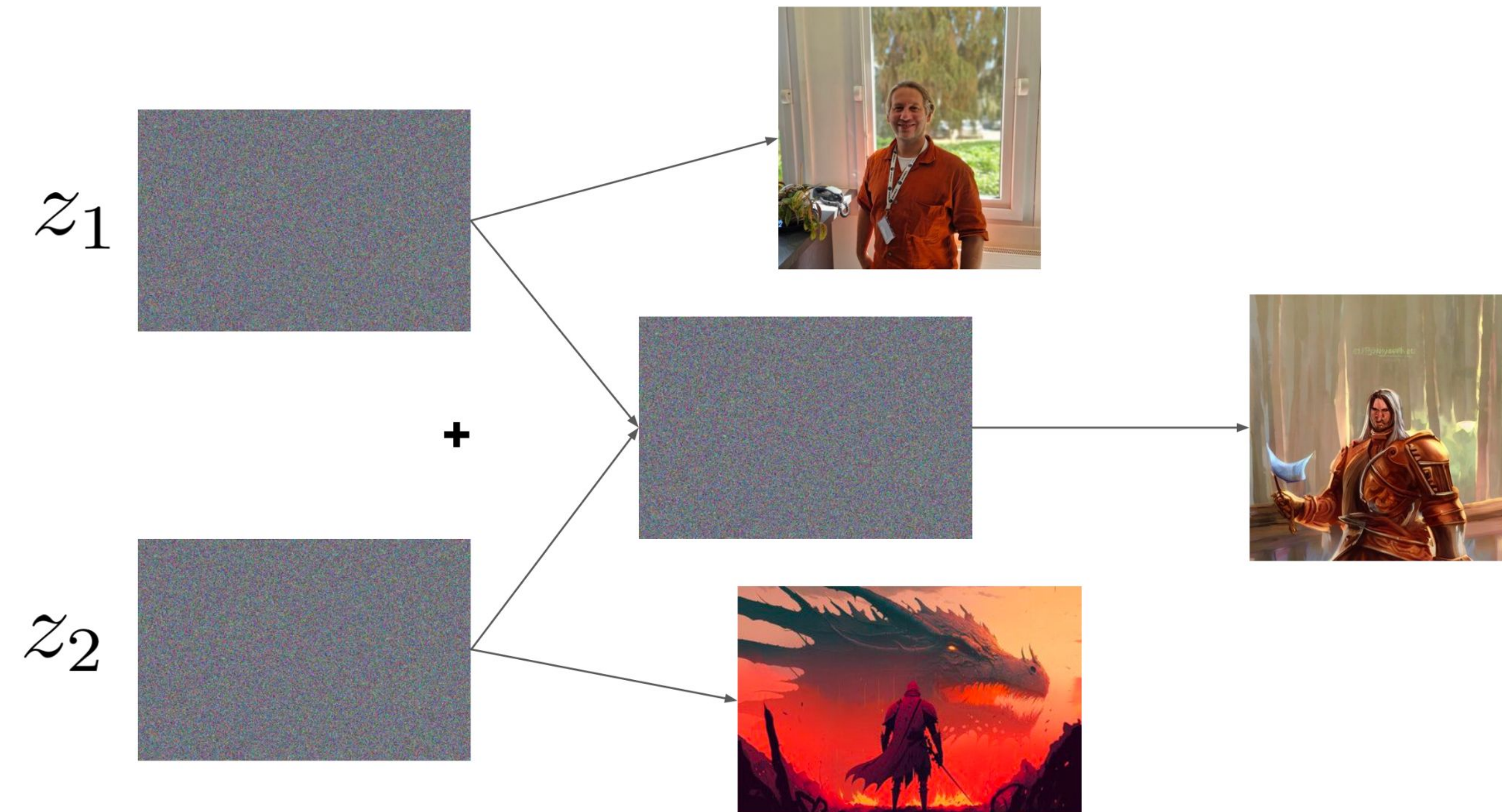
$$\mathbf{x}_{T-1} \approx \frac{1}{\sqrt{\alpha_T}} \left( \mathbf{x}_T - \frac{\beta_T}{\sqrt{1 - \bar{\alpha}_T}} \mathbf{z}_\theta(\mathbf{x}_T, T) \right) + \tilde{\beta}_T \mathbf{z} \text{ (noise)}$$

# and repeat !

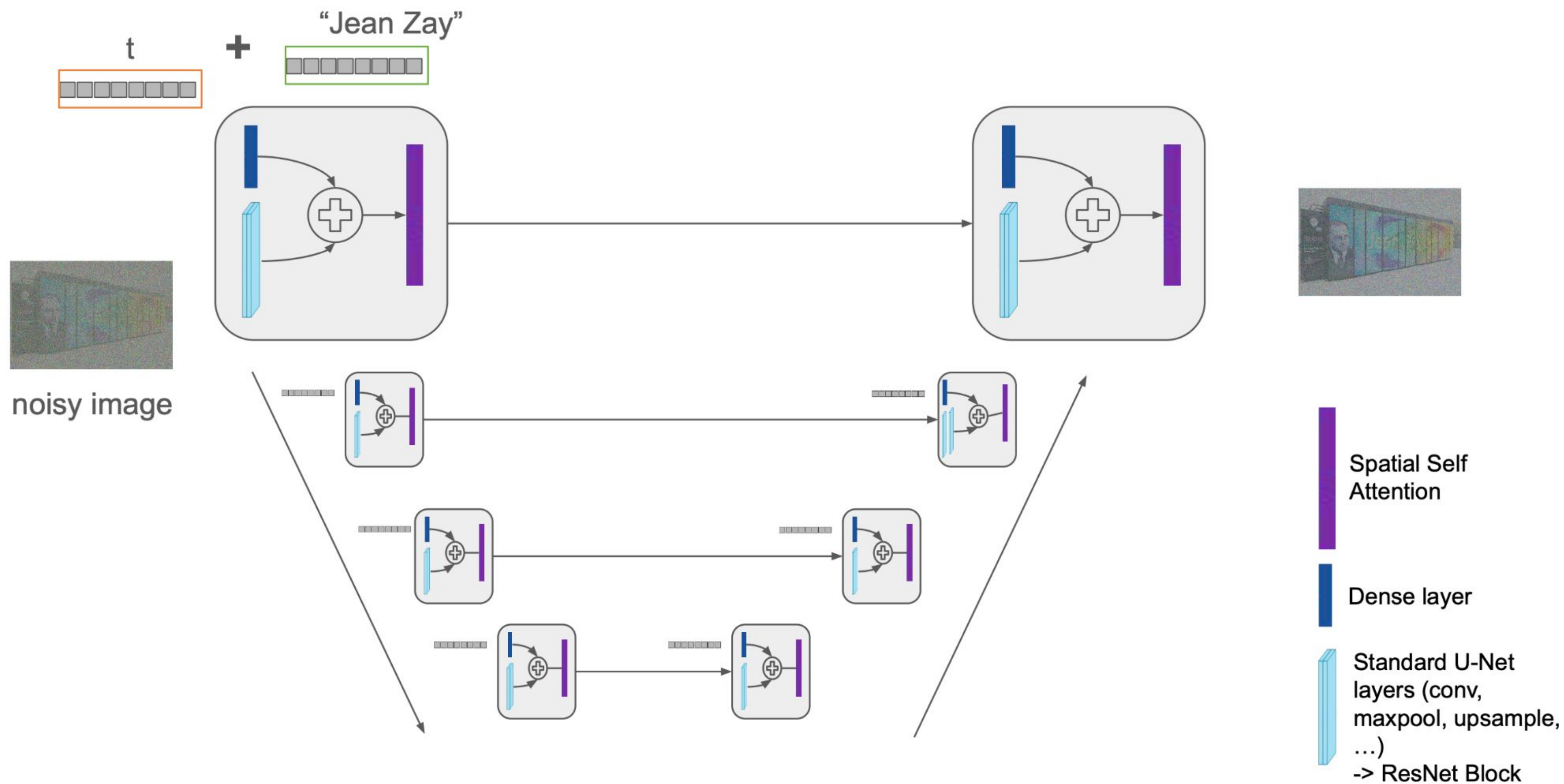
Don't generate  $x_T$ , replace it by  $x_{T-1}$  and  $T$  by  $T-1$ ... and do it again  $T$  time.



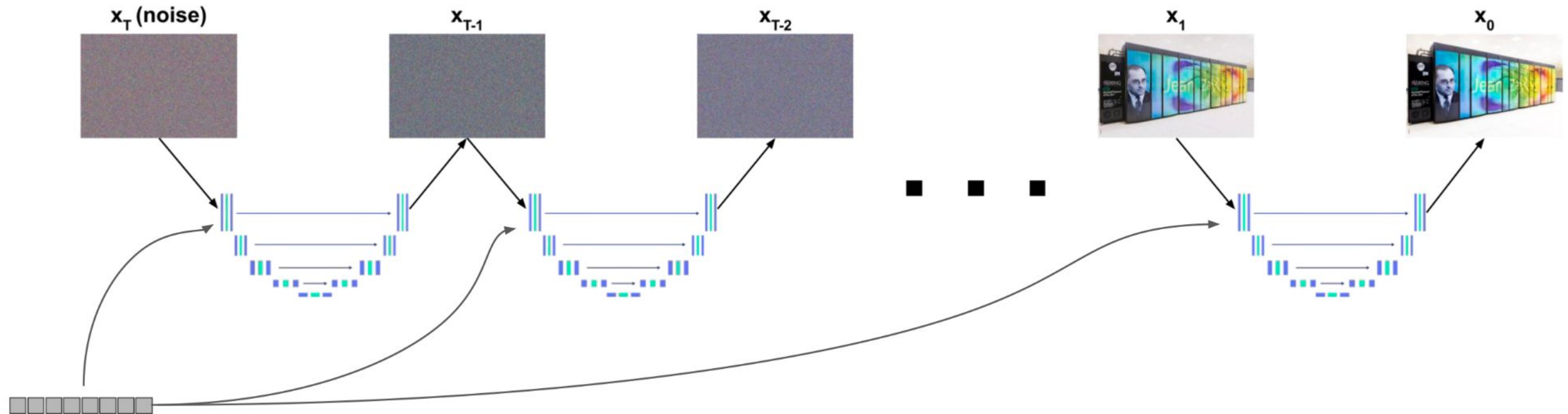
# Noise interpolation



# Architecture for Conditional Image Generation

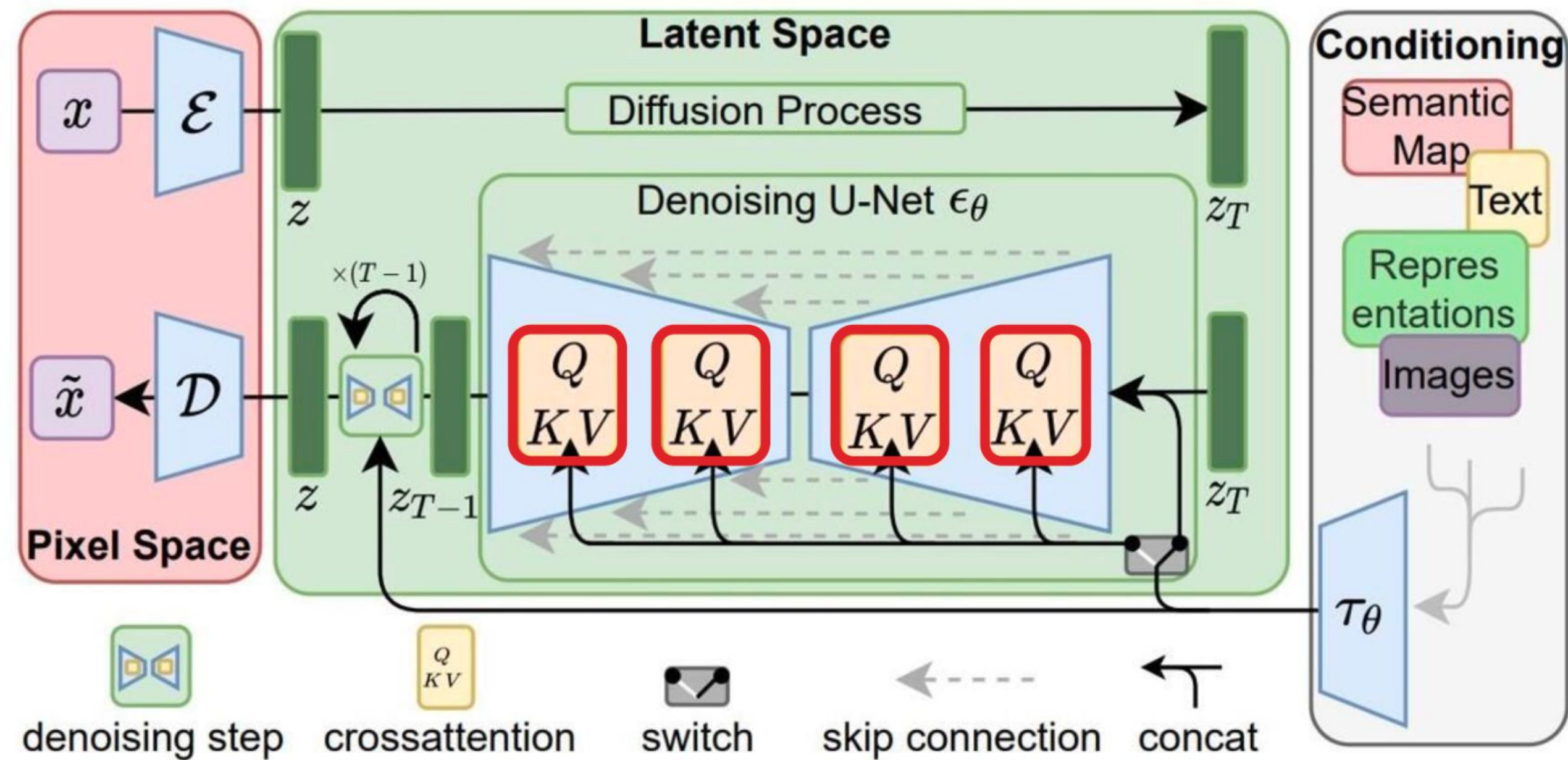


# Conditional Diffusion: text to image



# Conditional Diffusion: cross attention

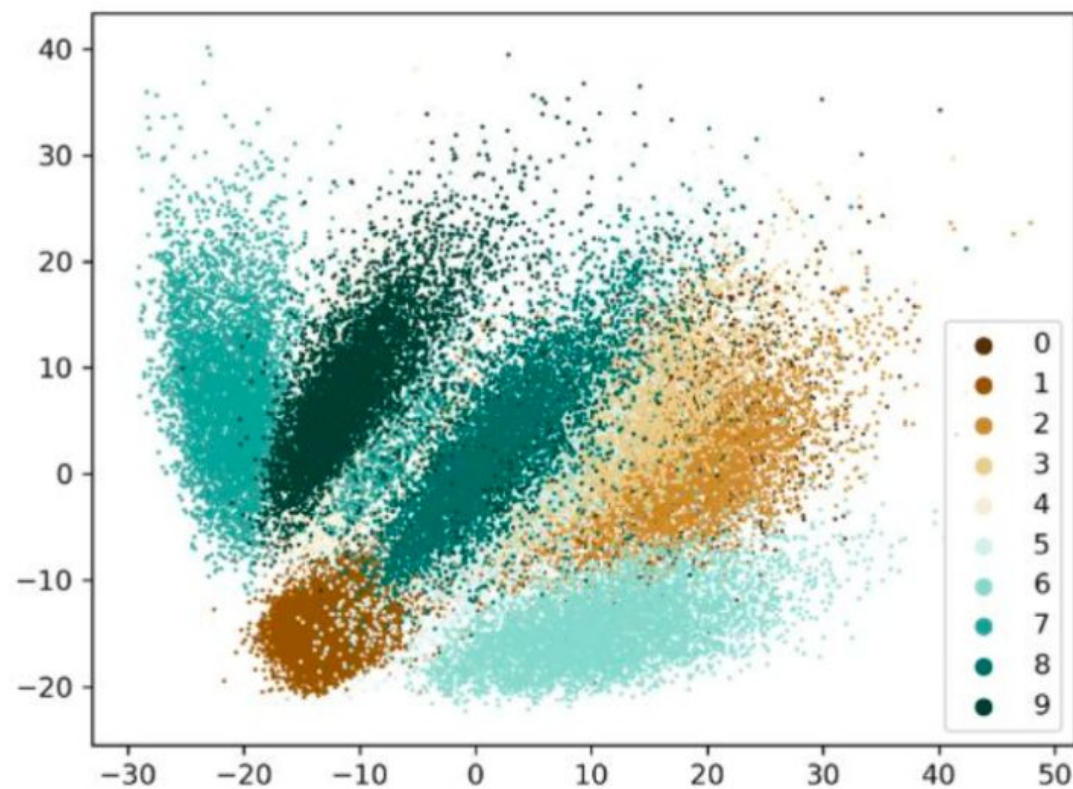
Stable Diffusion uses cross-attention to make the denoising process consistent with the provided sentence embedding



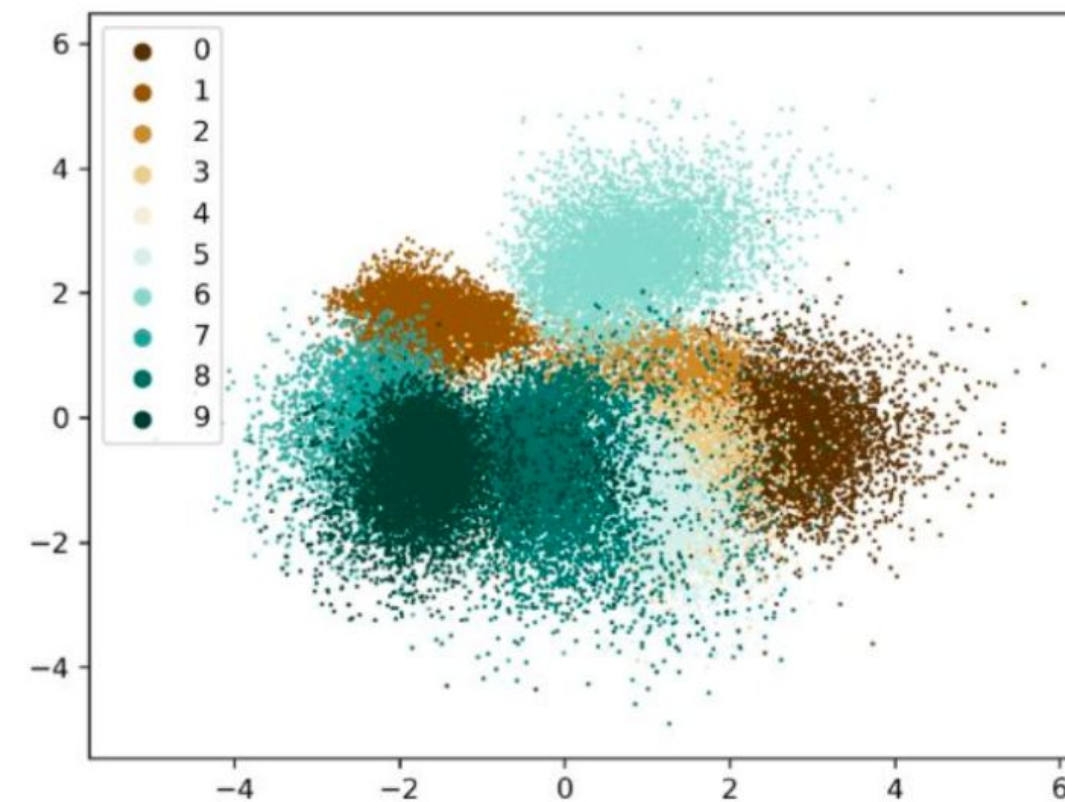
**Source** Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

# Latent Diffusion: concept of Latent Space

## Latent space of MNIST database for AE and VAE



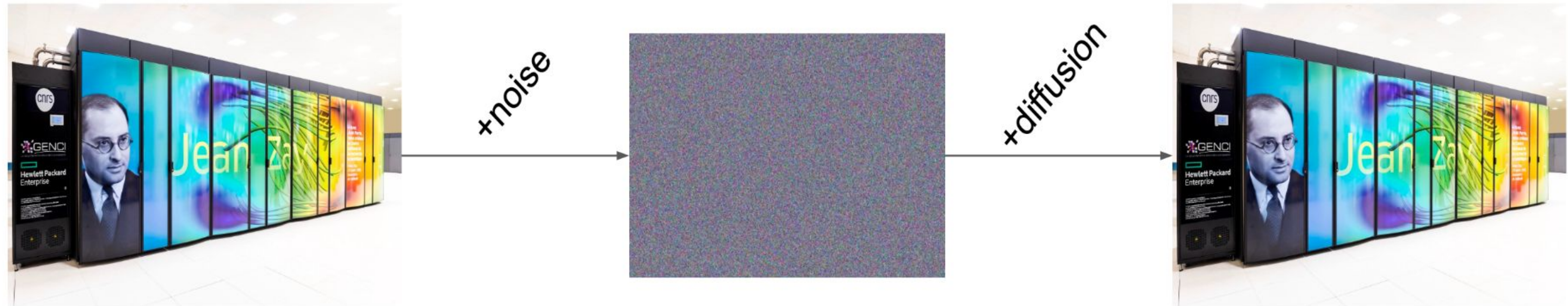
(a) Latent Distribution by Label for AE



(b) Latent Distribution by Label for VAE

Source <https://thilospinner.com/towards-an-interpretable-latent-space/>

- Similar objects are close to one another in the latent space
- Usually lower dimension than original data (therefore does compression as well)
- Usually impossible to visualize by a human



# Latent Diffusion Model

Latent space

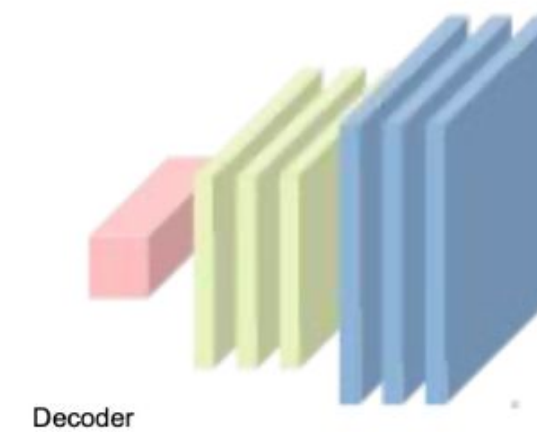
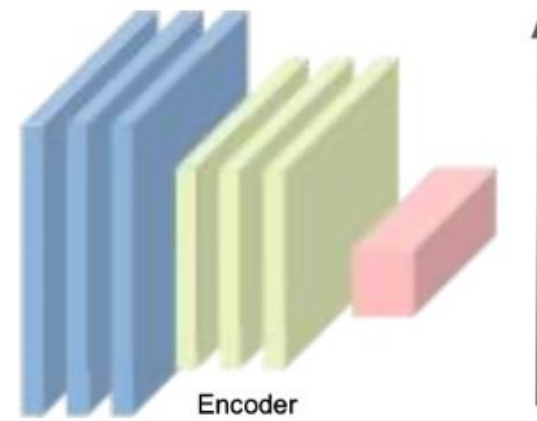
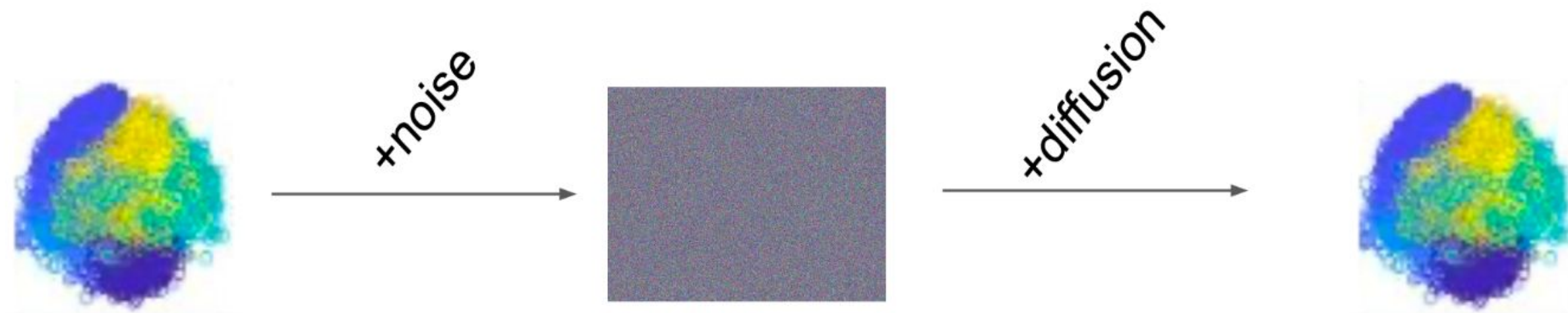


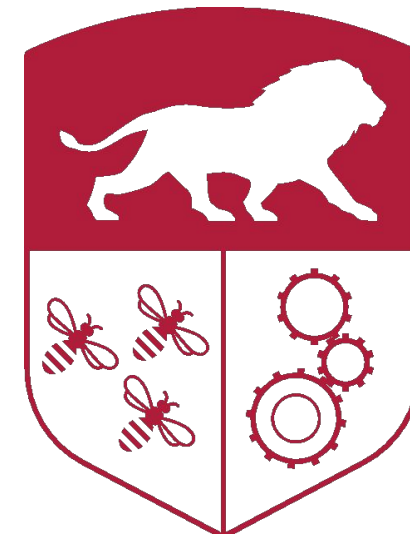
Image space



# Some useful references

---

- Fidle - Deep Learning Introduction (<https://www.fidle.cnrs.fr/w3/>)
- CS231n: Convolutional Neural Networks for Visual Recognition (<http://cs231n.stanford.edu>)
- Neural Networks and Deep Learning (<http://neuralnetworksanddeeplearning.com>)
- Deep Learning (<http://www.deeplearningbook.org>)
- PyTorch (<http://pytorch.org>)
- Weights & Biases (<https://wandb.ai/site/>)
- Hugging Face (<https://huggingface.co/>)



**CENTRALE  
LYON**

---

36, avenue Guy de Collongue 69130 Écully  
[www.ec-lyon.fr](http://www.ec-lyon.fr) | [@centralelyon](https://twitter.com/centralelyon)