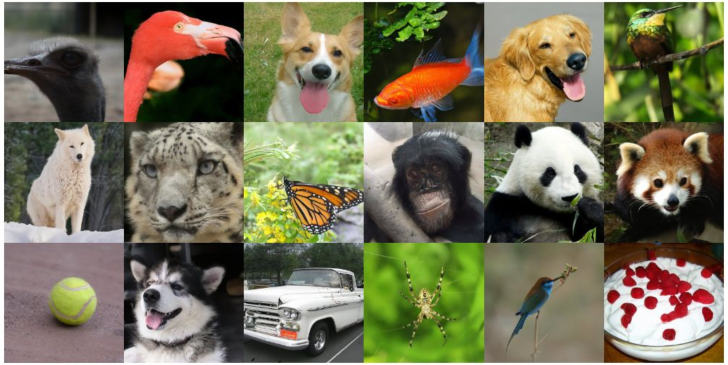
Deep Learning Architectures

Diffusion model

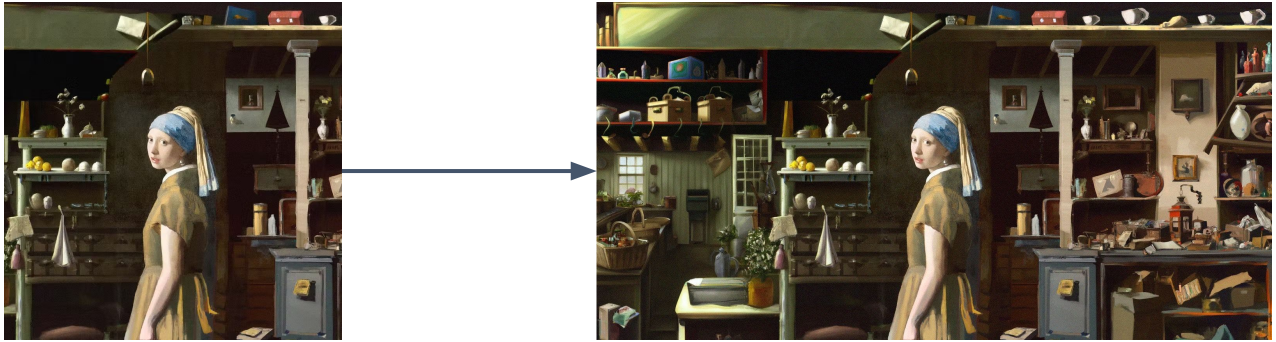
Institut du Développement et des Ressources en Informatique Scientifique

DPM Landscape



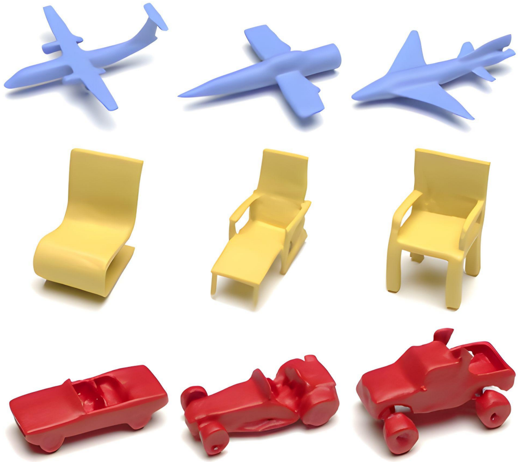
Dhariwal & Nichol, 2021

Source : [***https://github.com/bentoml/stable-diffusion-bentoml***](https://github.com/bentoml/stable-diffusion-bentoml)

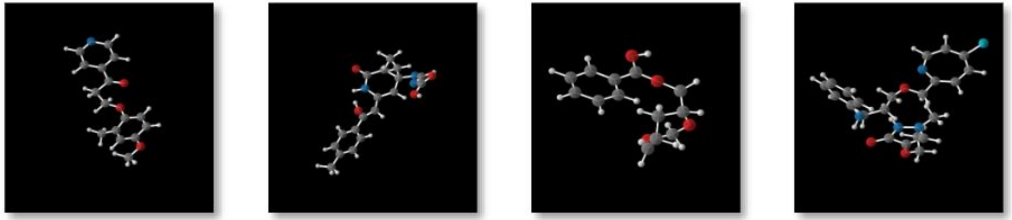


Source : Dall-E 3

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



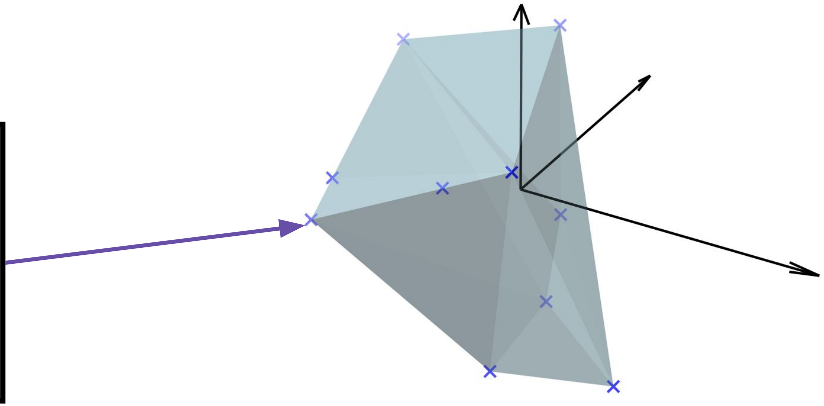
Source : https://arxiv.org/pdf/2210.06978.pdf



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

Generation of videos, of molecules, of …

$$= \begin{bmatrix} 0.1 \\ 0.1 \\ 1.0 \end{bmatrix}$$

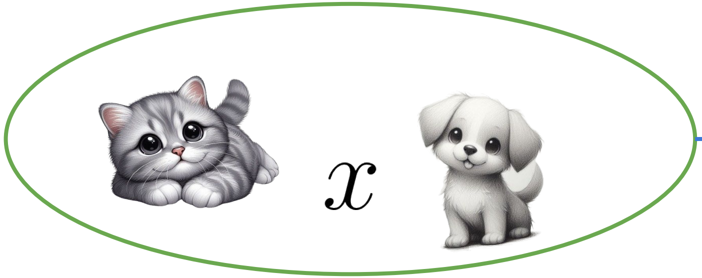


Source : Dall-E 3

$p(x)$ **= Probabilité que le point x appartient à notre base de données**

Distribution de données P(x)

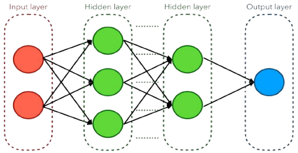
Generation vs Classification



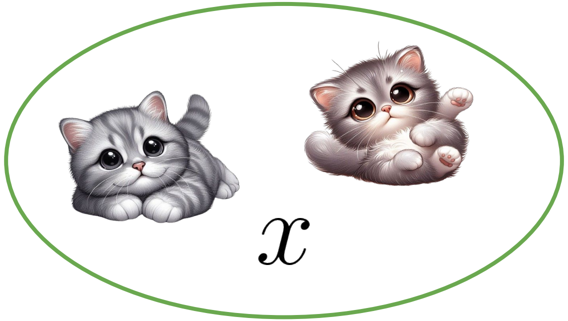
y

$$p(y|x) \sim q(y|x)$$

Classification



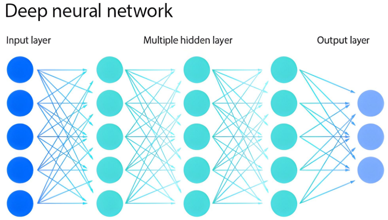
Generation



$$p(x) \sim q(x)$$

Output of models

Classification with 3 classes

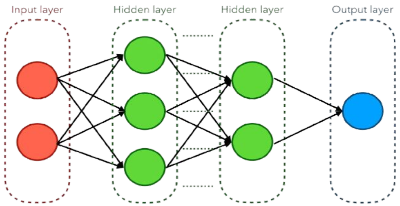


a discrete probability

$$= \begin{bmatrix} 0.1 \\ 0.1 \\ 0.8 \end{bmatrix} \begin{matrix} \text{: chien} \\ \text{: chat} \\ \text{: humain} \end{matrix} \rightarrow = \begin{bmatrix} 0.0 \\ 0.0 \\ 1.0 \end{bmatrix}$$

$$q(y|x)$$

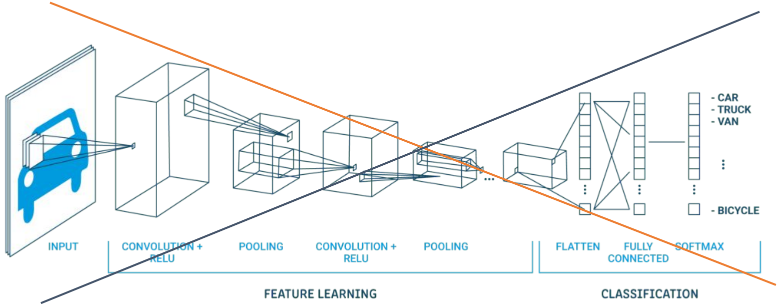
???

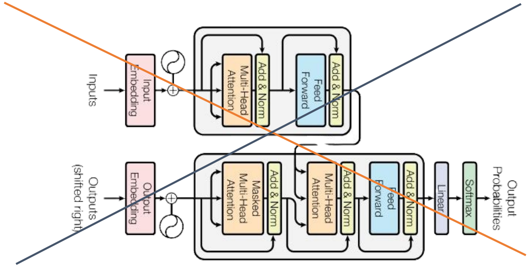


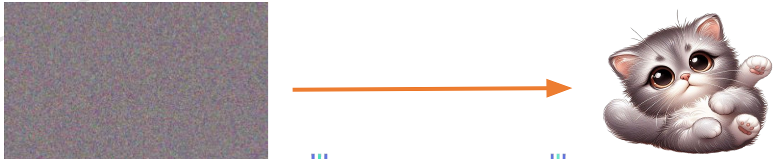
$$= \mathcal{N}(output, \sigma^2)$$

$$= output + z * \sigma, \text{where } z \sim \mathcal{N}(0, 1)$$

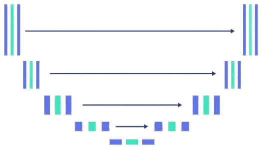
Generation of one continuous value







$\mathcal{N}(0,1)$



$p(x)$

$q(x|\mathcal{N}(0,1))$

$\sim$

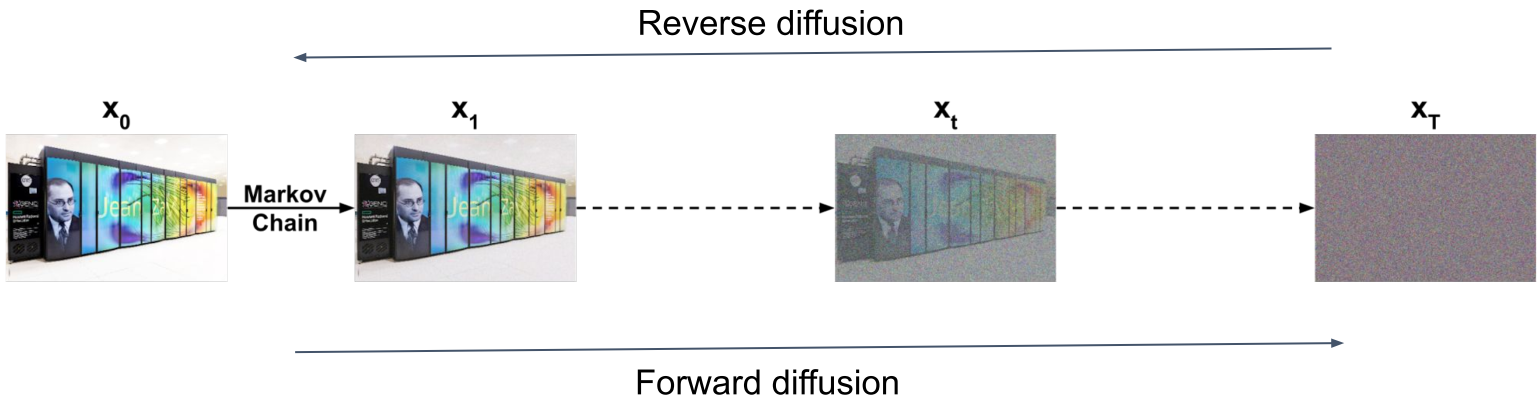
$$q(x|\mathcal{N}(0,1)) * \mathcal{N}(0,1) = q(x)$$

CNN/Transformer/GNN vs Diffusion Model

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



DDPM Principle

Forward Diffusion Process

Denoising Diffusion Probabilistic Models (DDPM)

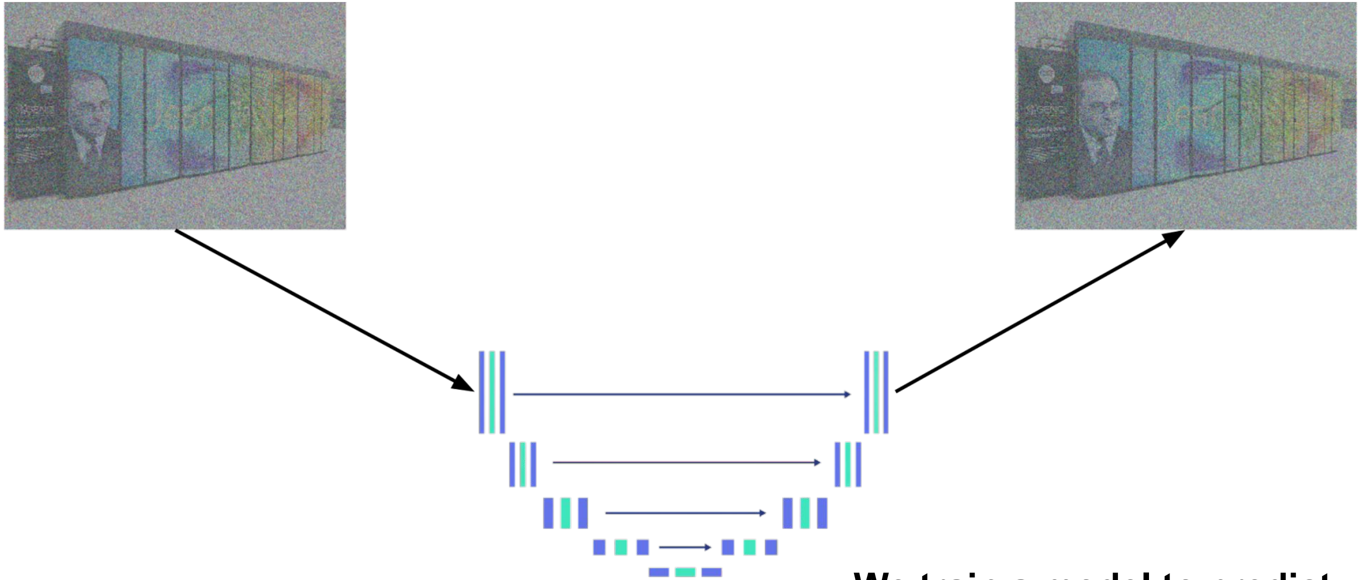
Here we choose T=1000, but it can be different values (it's an hyperparameter)

DDPM Principle 3

Reverse Diffusion Process

x_t — a bit more noisy than x_{t-1}

x_{t-1} — a bit less noisy than x_t

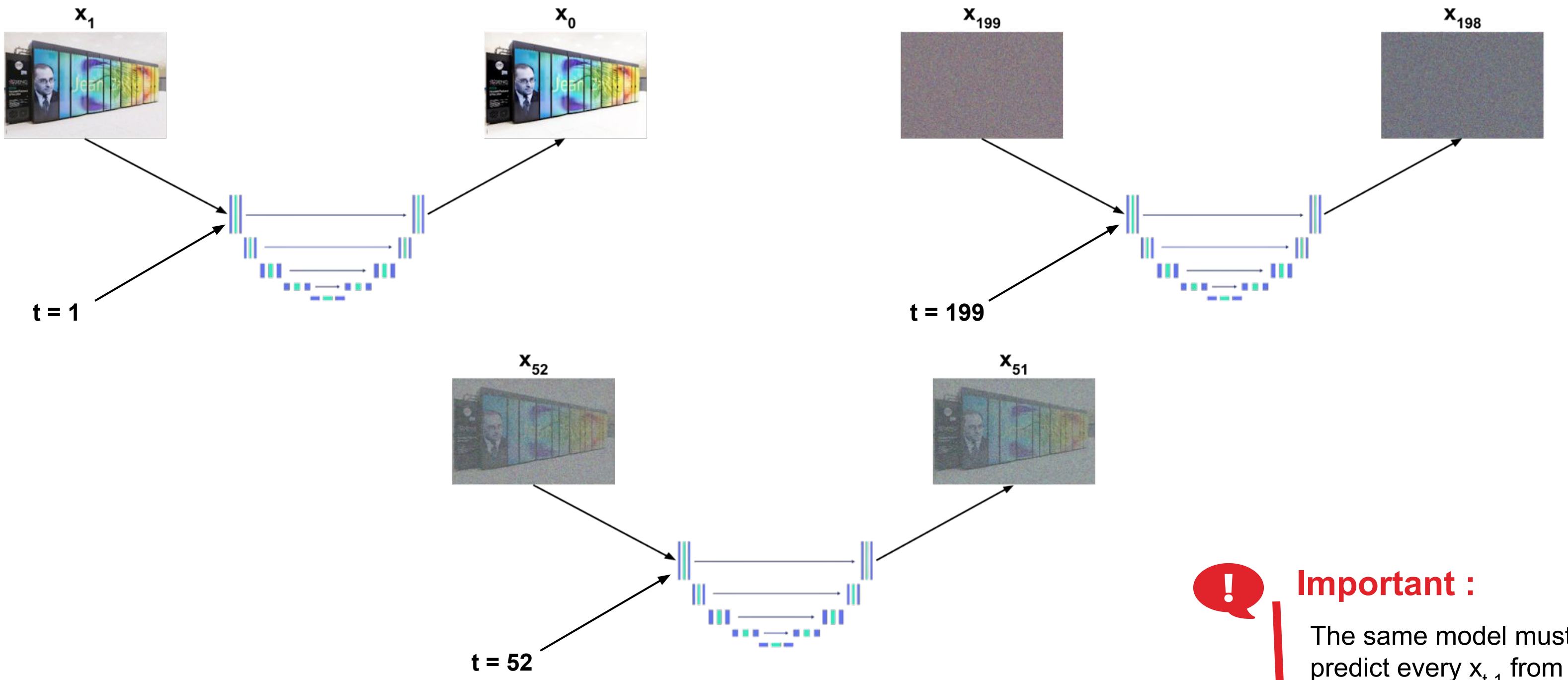


We train a model to predict x_{t-1} from x_t

x_0 is any image from the dataset

DDPM Principle 4

Reverse Diffusion Process

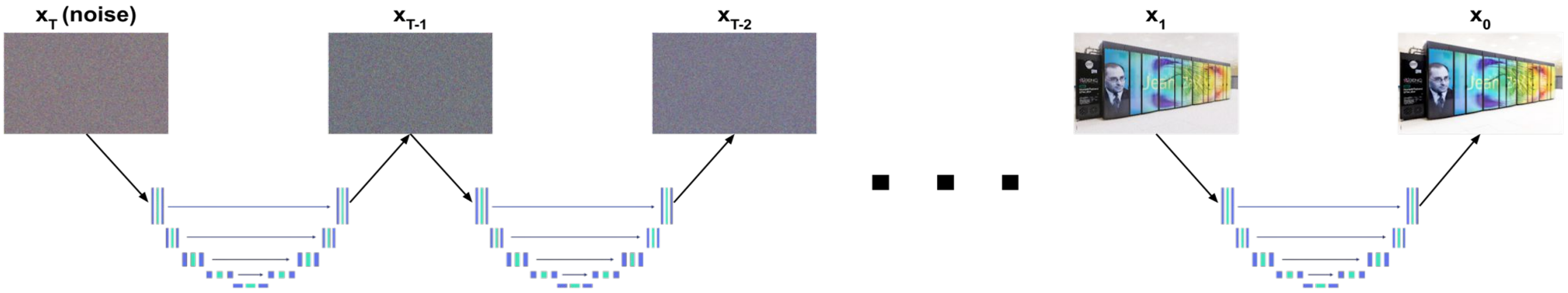


Important :

The same model must predict every x_{t-1} from x_t

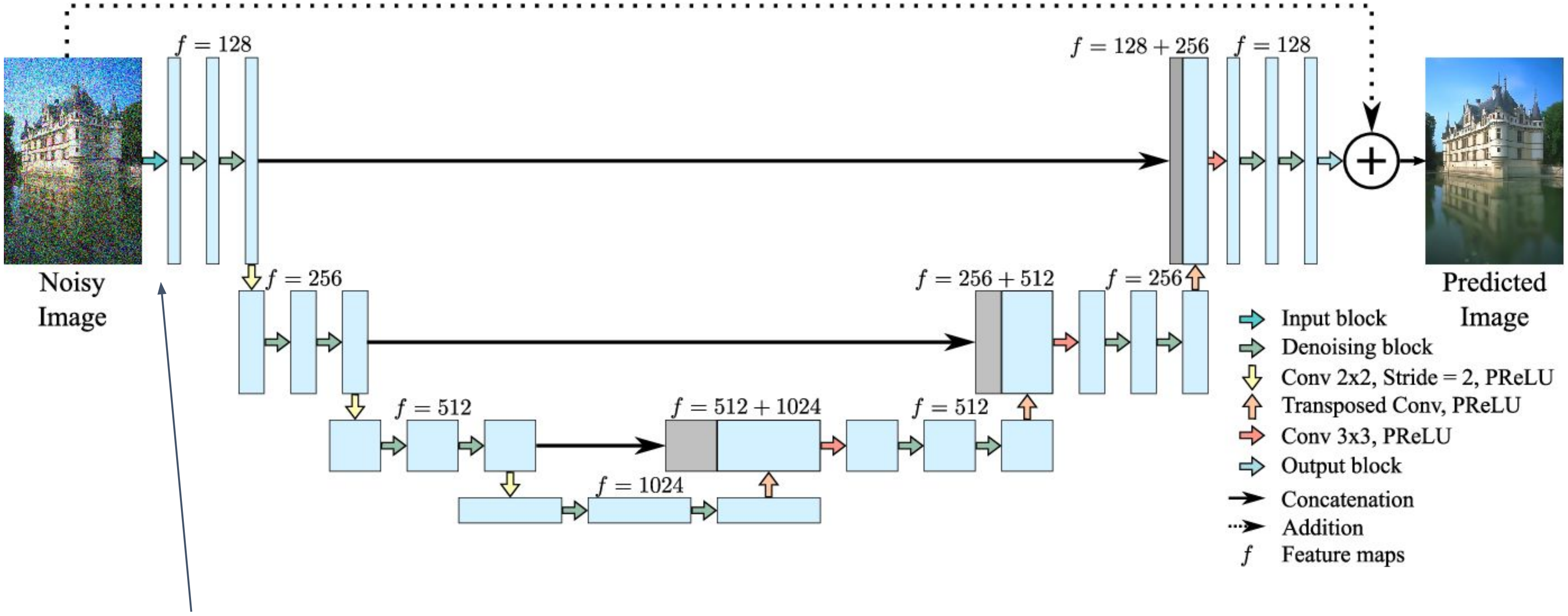
DDPM Principle 5

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



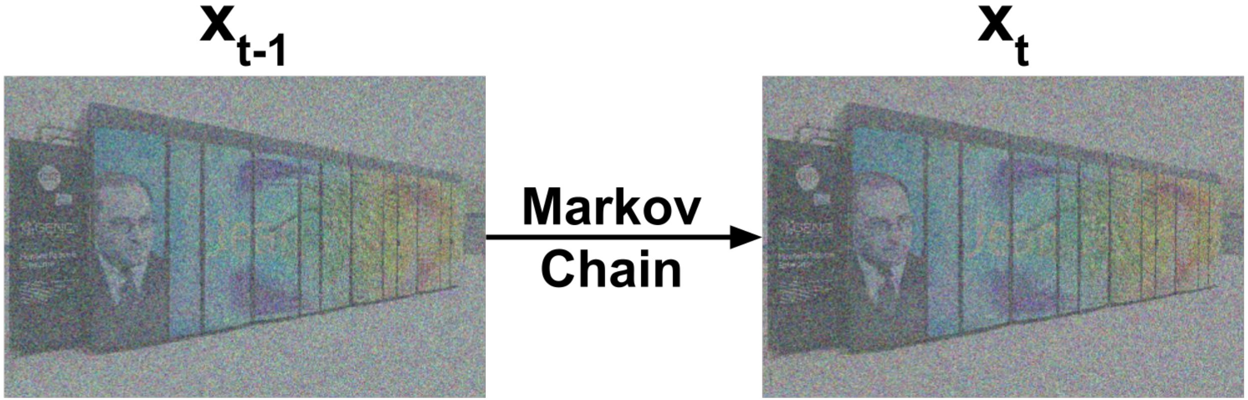

```
torch.randn(4)
```

DDPM Principle 6



$$\mathrm{PE}(t) = \left[\sin\left(\frac{t}{10000^{\frac{2i}{d}}}\right), \cos\left(\frac{t}{10000^{\frac{2i}{d}}}\right)\right]_{i=0}^{d/2}$$

Unet: a good denoiser + t-embedding



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

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

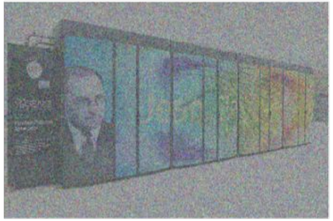
$$where \quad z_{t-1} \sim \mathcal{N}(0, I)$$

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

Forward diffusion 2 - Sampling a gaussian distribution

Forward diffusion 3

$\mathbf{x}_t$



$=$

$\sqrt{1-\beta_t}$

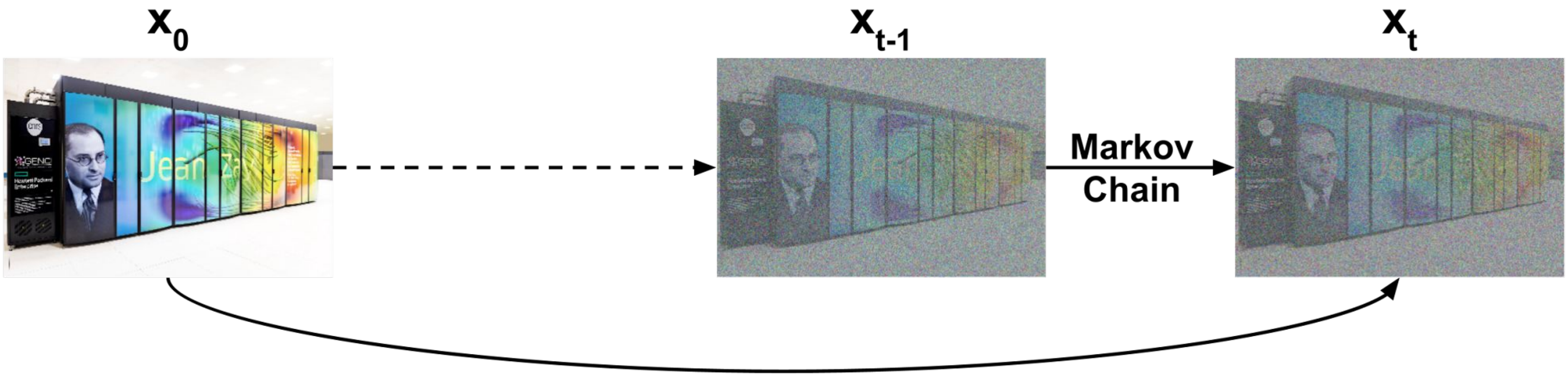
$\mathbf{x}_{t-1}$

$+$

$\sqrt{\beta_t}$

Gaussian noise

$$\mathbf{x}_t = \sqrt{1-\beta_t}\,\mathbf{x}_{t-1} + \sqrt{\beta_t}\,\text{(Gaussian noise)}$$



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

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

Forward diffusion 4

β_t (the noise schedule) such that $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

$\beta_t \epsilon (0,1),\ \beta_t\ following\ a\ schedule\ \beta_1 < \beta_2 < \ldots < \beta_T$

Linear scheduling

0.02 0.0001

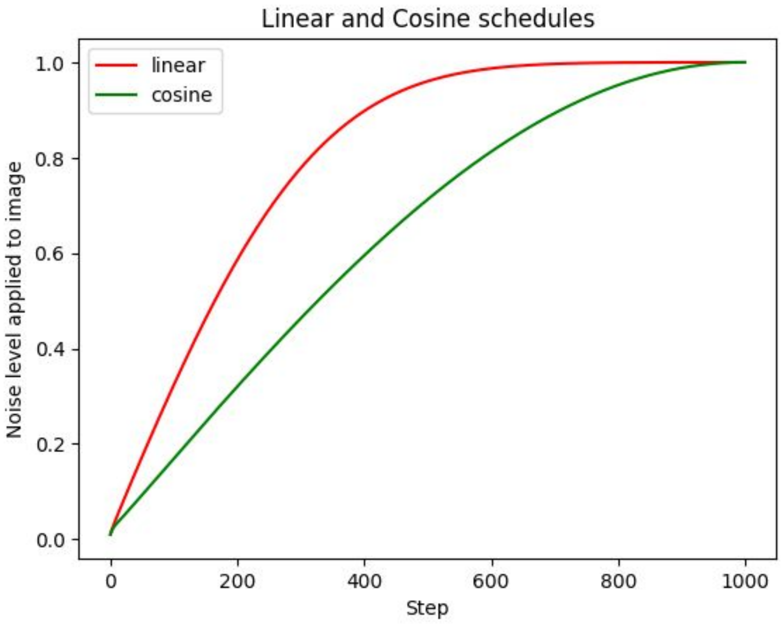
$$(\beta_1, \ldots, \beta_T) = (\beta_1 + (t-1) * \frac{\beta_T - \beta_1}{T-1})_{t \in \{1 \ldots T\}}$$

1000

Cosinus scheduling

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2$$

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



Beta scheduling

$$q(x_0)$$

Données

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

Données bruitées par un petit bruit gaussien

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

Données bruitées à l'étape t

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

Computation 0

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

Chaîne de Markov

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

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

Computation 1 - Gaussian reverse diffusion

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

$$\tilde{\mu}_t(\mathbf{x_t}, \mathbf{x_0}) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x_0} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x_t}$$

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

$$q(x_t|x_0)$$

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

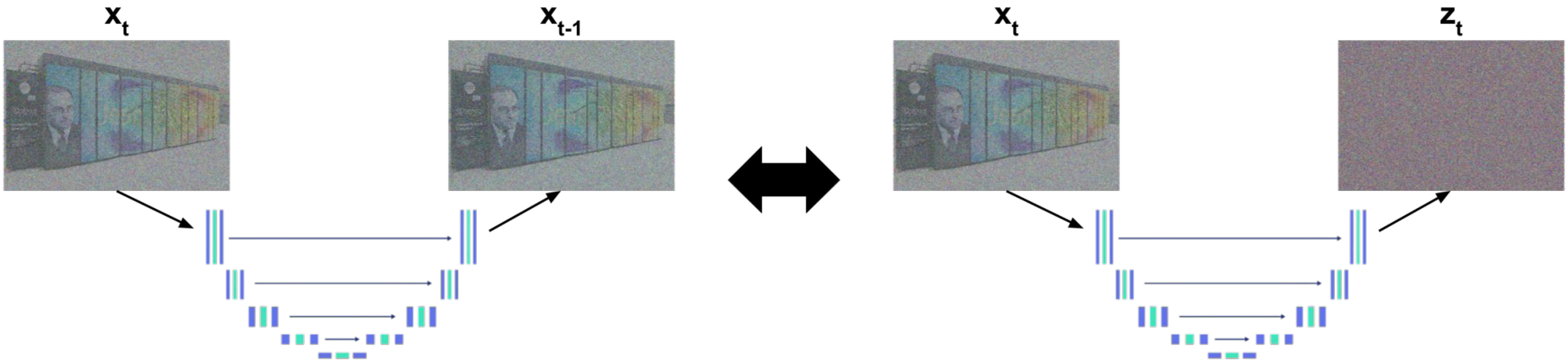
→

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

$$\tilde{\mu}_t(\mathbf{x_t}, \mathbf{x_0}) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x_t} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z_t}\right)$$

$\mathbf{z_t}$???

Computation 2 - Mean and Variance of the reverse diffusion



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

Reverse diffusion 1

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

Our network

$\mathbf{x}_{t-1}$

$$= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x_t} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z_t}\right) + \sqrt{\tilde{\beta}_t}\,\mathbf{z}$$

z (noise)

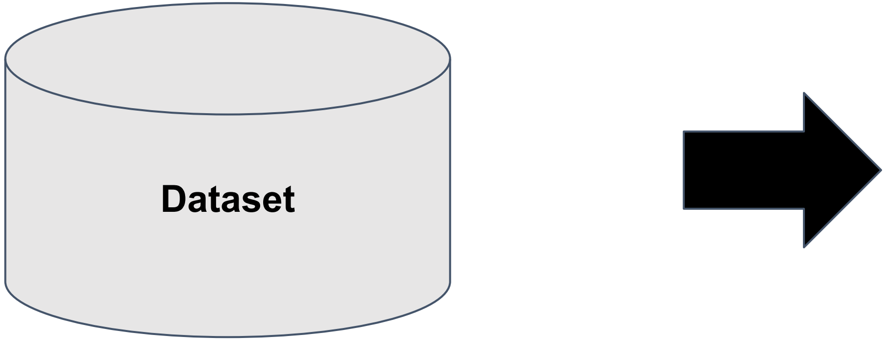
Reverse diffusion 2

Algorithm 1 Training

```
repeat
  x_0 ~ q(x_0)
  t ~ Uniform({1, ..., T})
  ε ~ N(0, I)
  Take gradient descent step on
      ∇_θ ‖ε − ε_θ(√ᾱ_t x_0 + √(1 − ᾱ_t) ε, t)‖²
until converged
```

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right) \right\|^2$$

Training 1



$\mathbf{x}_0$

Training 2

Uniform distribution

Between 1 and T

$t = 50$

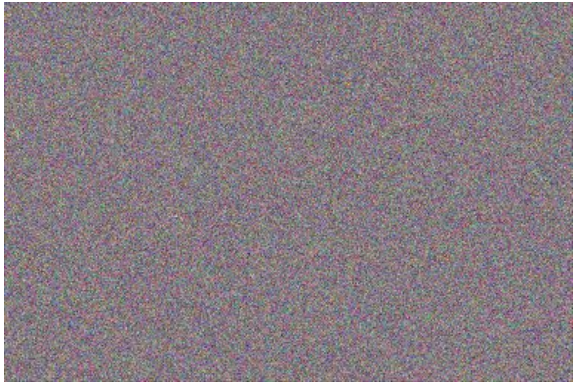
x_0

Training 3

Gaussian distribution

Same shape than x_0

$\mathbf{z_t}$ (= ϵ)

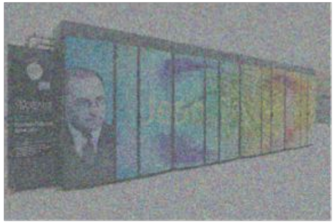


$\mathbf{x_0}$

$t = 50$

Training 4

x_t



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

x_0

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

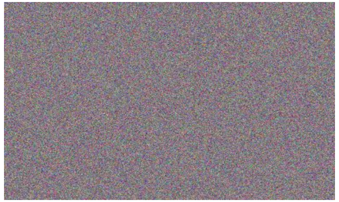
z_t

---

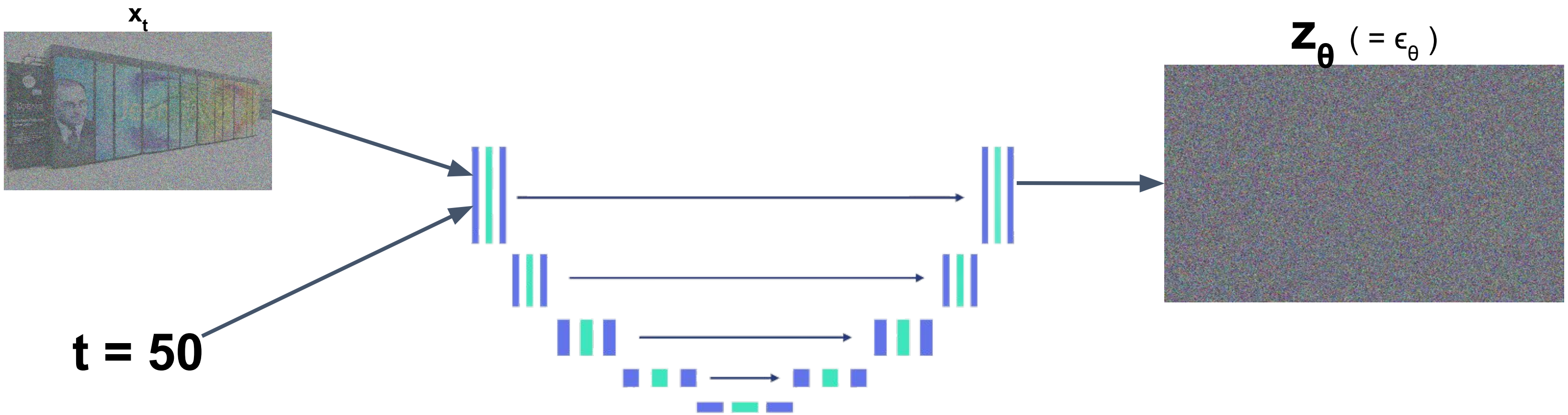
x_0

t = 50

z_t



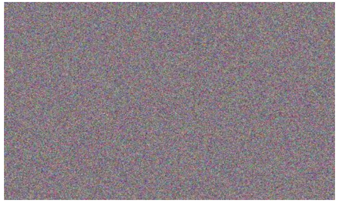
Training 5



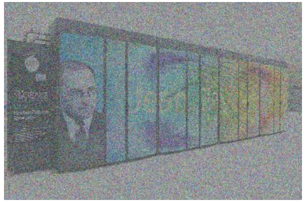
x_0

$t = 50$

z_t



x_t

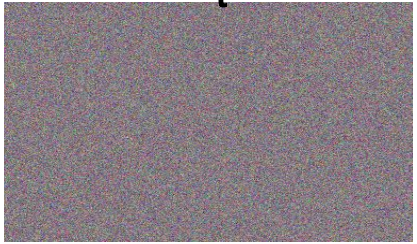


Training 6

Training 7

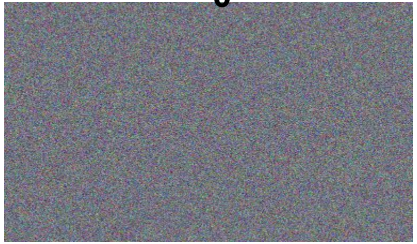
Loss = ‖

z_t



−

z_θ

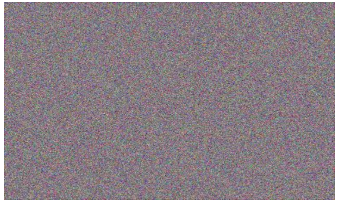


‖^2

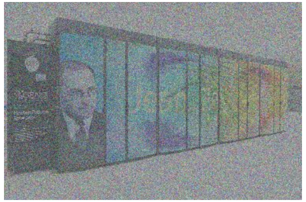
x_0

t = 50

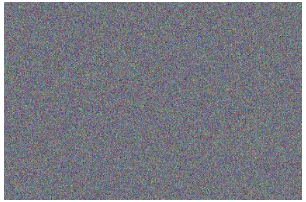
z_t

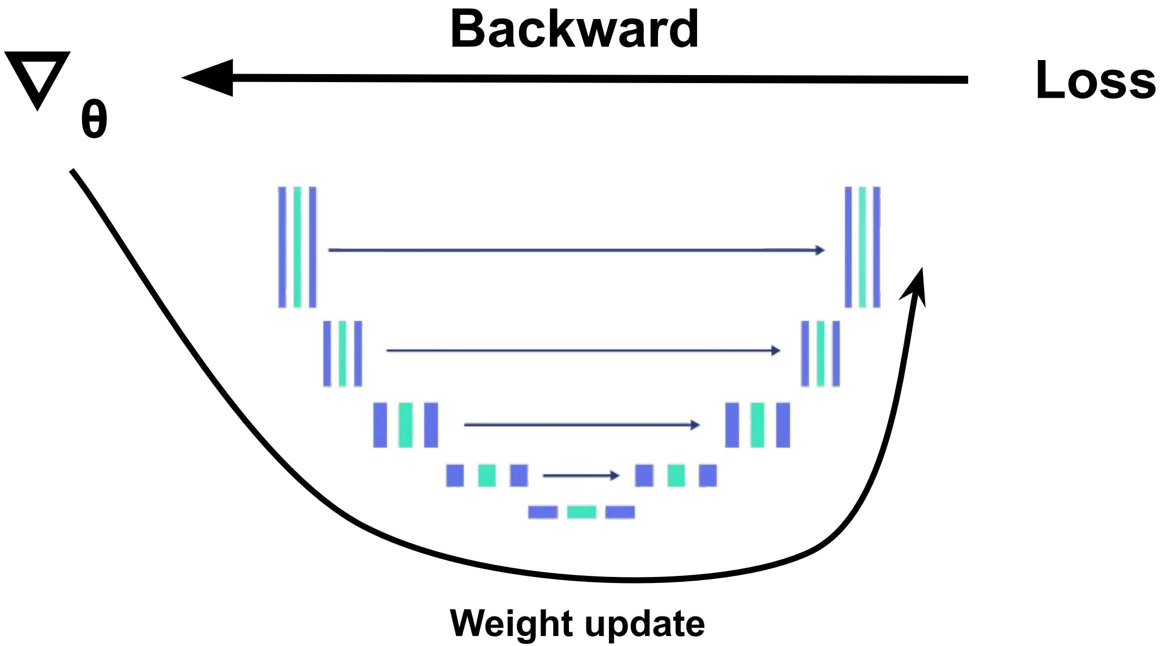


x_t



z_θ





Training 8

$$L_{\text{VLB}} = \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0)$$

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

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

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

A more complicated loss …

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

for $t = T, \ldots, 1$ **do**

  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$

  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$

end for

return $\mathbf{x}_0$

https://arxiv.org/abs/2006.11239

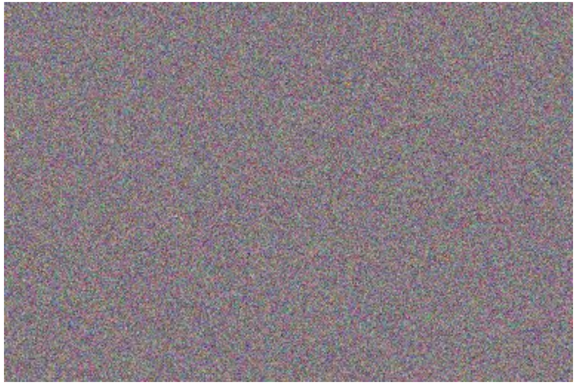
Sampling 1

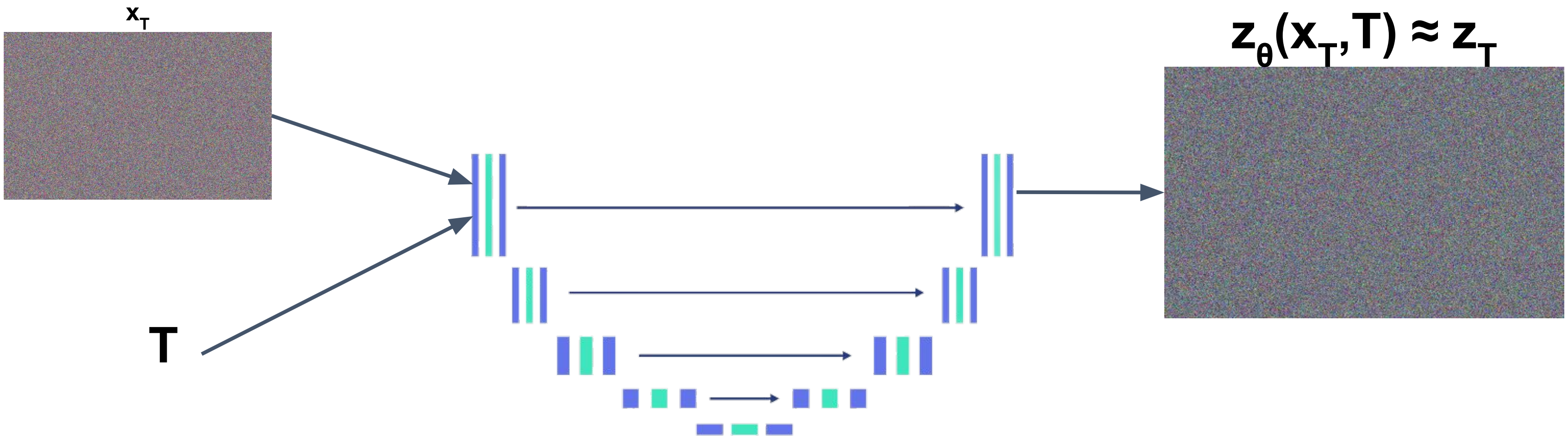
Sampling 2

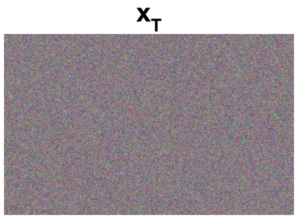
Gaussian distribution

Same shape than training dataset images

$\mathbf{x}_T$







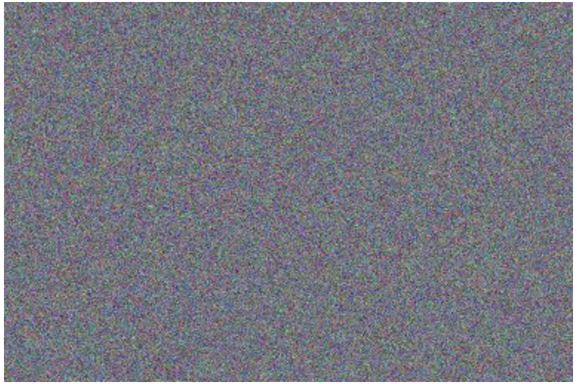
Sampling 3

Sampling 3

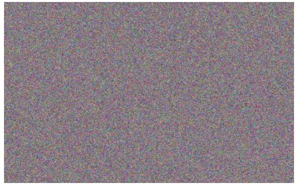
Gaussian distribution

Same shape than training dataset images

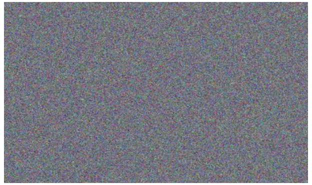
z (noise)



x_T



z_T



x_{T-1}

$\approx \frac{1}{\sqrt{\alpha_T}}($

x_T

$- \frac{\beta_T}{\sqrt{1-\bar{\alpha}_T}}$

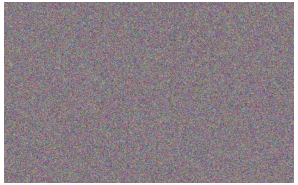
$z_\theta(x_T,T)$

$) + \tilde{\beta}_T$

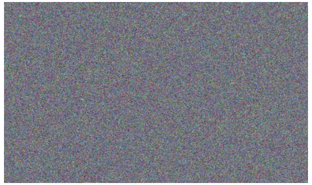
z (noise)

---

x_T



$z_\theta(x_T, T)$

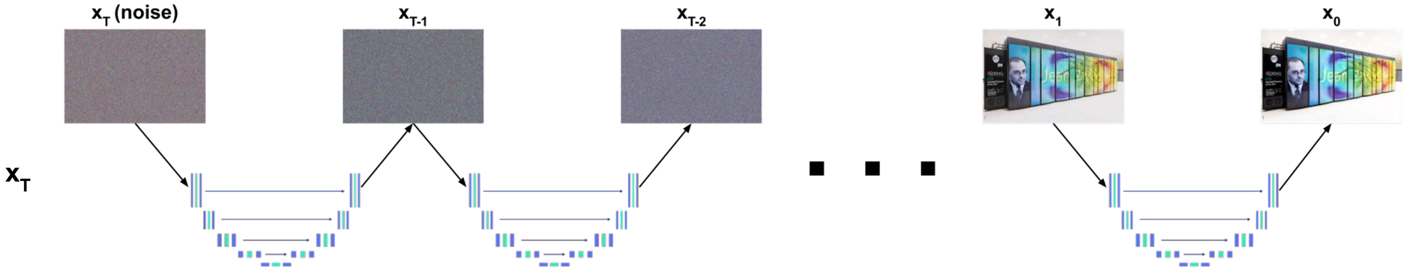


z (random noise)

Sampling 4

and repeat !

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



Sampling 5

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

New mathematical view :

$$\nabla_{\mathbf{x_t}} \mathbf{log(q(x_t))} = -\frac{\mathbf{z_\theta(x_t, t)}}{\sqrt{1-\bar{\alpha}_t}}$$

$$\tilde{\mu}_t(x_t, x_0) \approx \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z_\theta(x_t, t)}) = \frac{1}{\sqrt{\alpha_t}}(x_t + \beta_t \times \nabla_{\mathbf{x_t}} \mathbf{log(q(x_t))})$$

$$\nabla_{\mathbf{x_t}} \mathbf{log(q(x_t))} \longrightarrow \nabla_{\mathbf{x_t}} \mathbf{log(q(x_t|y))}$$

Unconditional to Conditional Diffusion Model

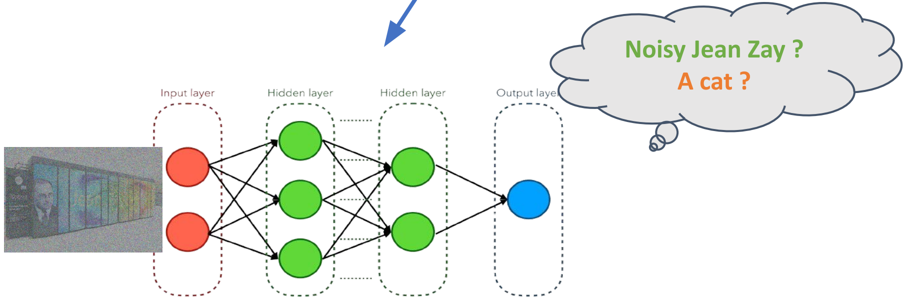
$$\nabla_{\mathbf{x_t}} \log(\mathbf{q}(\mathbf{x_t}|\mathbf{y})) = \nabla_{\mathbf{x_t}} \log(\mathbf{q}(\mathbf{x_t})) + \nabla_{\mathbf{x_t}} \log(\mathbf{q}(\mathbf{y}|\mathbf{x_t}))$$

Strength of guidance

$$\tilde{\mu}_t(x_t, x_0, y) \approx \frac{1}{\sqrt{\alpha_t}}(x_t + \beta_t \times [\nabla_{\mathbf{x_t}} \log(\mathbf{q}(\mathbf{x_t})) + \lambda \times \nabla_{\mathbf{x_t}} \log(\mathbf{q}(\mathbf{y}|\mathbf{x_t}))])$$

Unconditional pretrained model

Classifier output



Classifier Guidance

Principle of the Classifier Guidance method

Can be very large and long to train

Which guidance strength ?

1 — 2 — 3 — 4

DDPM

Train our Unconditional diffusion model

Classifier

Train a classifier model on noisy images

Guided noise estimation

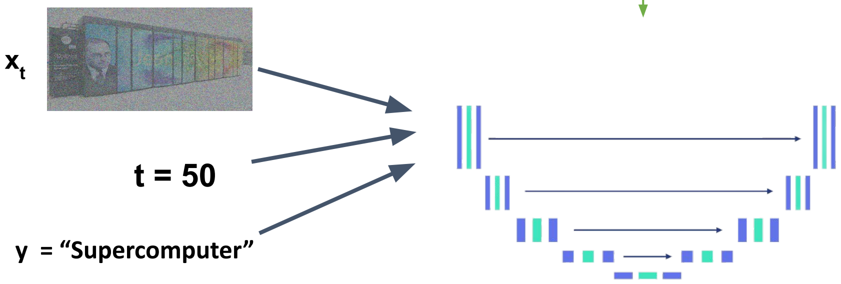
Compute the conditional part from the output of the classifier and the unconditional part

Guided denoising

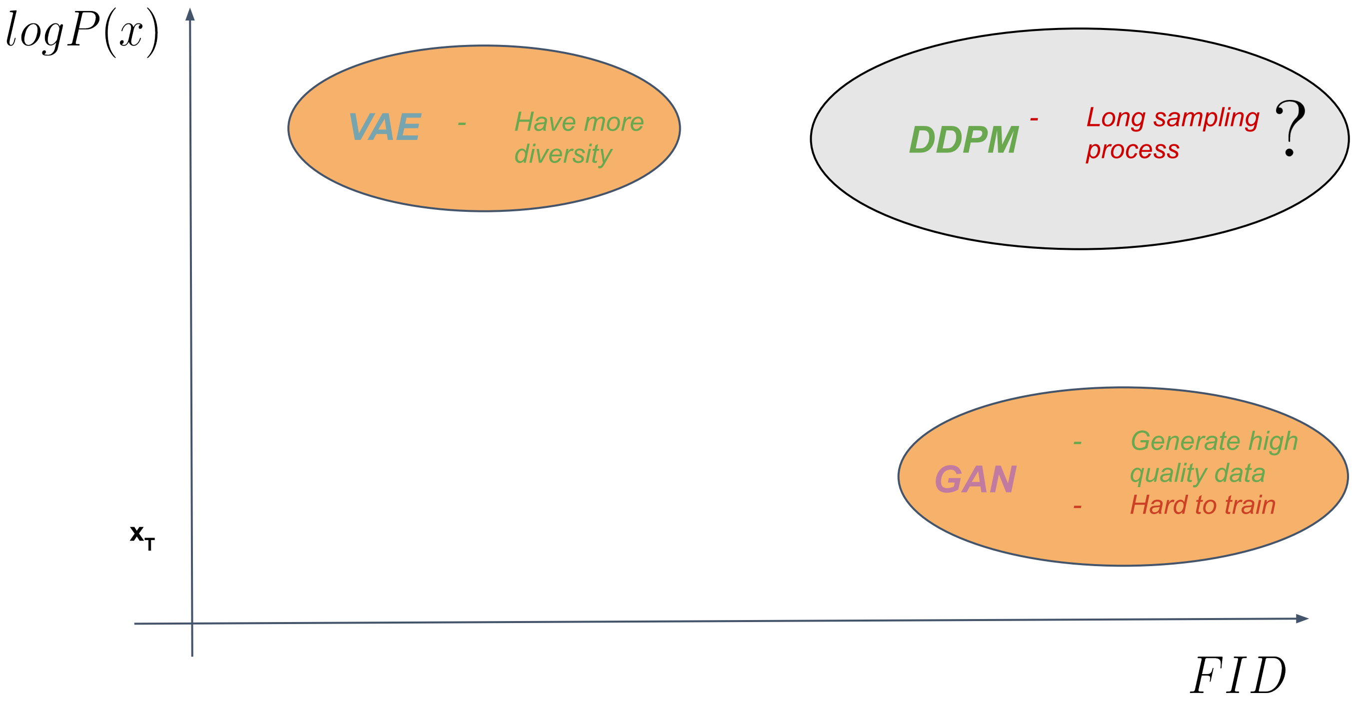
Denoise using the guided noise estimated

$$\frac{1}{\sqrt{\alpha_t}}(x_t + \beta_t \times [\nabla_{\mathbf{x_t}} \mathbf{log}(\mathbf{q}(\mathbf{x_t})) + \lambda \times \nabla_{\mathbf{x_t}} \mathbf{log}(\mathbf{q}(\mathbf{y}|\mathbf{x_t}))])$$

$$\nabla_{\mathbf{x_t}} \mathbf{log}(\mathbf{q}(\mathbf{y}|\mathbf{x_t})) = \nabla_{\mathbf{x_t}} \mathbf{log}(\mathbf{q}(\mathbf{x_t}|\mathbf{y})) - \nabla_{\mathbf{x_t}} \mathbf{log}(\mathbf{q}(\mathbf{x_t}))$$



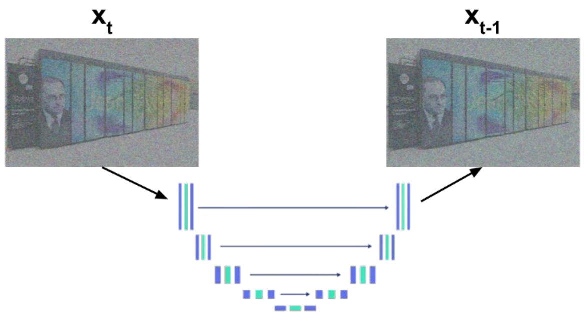
Classifier-Free Guidance

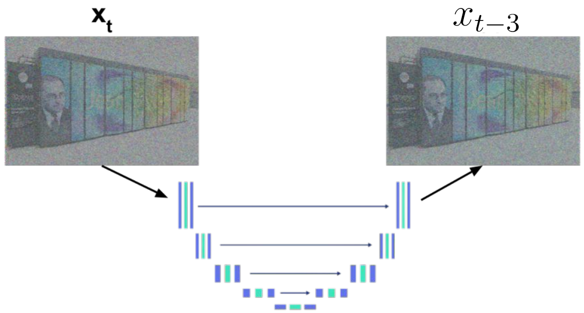


Limitations :

"For example, it takes around 20 hours to sample 50k images of size 32 x 32 from a DDPM, but less than a minute to do so from a GAN on a Nvidia 2080 Ti GPU." (DDIM, 2021)

DDPM vs GAN vs VAE





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

$$x_{t-3} = \sqrt{\bar{\alpha}_{t-3}} x_0 + \sqrt{1 - \bar{\alpha}_{t-3}} z$$

$\longrightarrow$

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

$$x_{t-3} = \sqrt{\bar{\alpha}_{t-3}} * \frac{x_t - \sqrt{1 - \bar{\alpha}_t} z_t}{\sqrt{\bar{\alpha}_t}} + \sqrt{1 - \bar{\alpha}_{t-3}} z$$

Reduce the sampling step

Generalisation to a bigger class of inverse process (non-Markovian)

$$q_\sigma(x_{t-1}|x_t, x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2}.\frac{x_t - \sqrt{\bar{\alpha}_t}x_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2 I)$$

$$\sigma_t^2 = \tilde{\beta}_t \quad \Longrightarrow \quad DDPM$$

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

Important :
Same network and training as a DDPM

DDIM 1

$$\sigma_t^2 = \tilde{\beta}_t \quad \Rightarrow \quad DDPM$$

$$\sigma_t^2 = \eta . \tilde{\beta}_t \qquad \eta \in [0, 1]$$

$\eta = 0$ => DDIM

	S	CIFAR10 (32 × 32) 10	20	50	100	1000	CelebA (64 × 64) 10	20	50	100	1000
η	0.0	**13.36**	**6.84**	**4.67**	**4.16**	4.04	**17.33**	**13.73**	**9.17**	**6.53**	3.51
	0.2	14.04	7.11	4.77	4.25	4.09	17.66	14.11	9.51	6.79	3.64
	0.5	16.66	8.35	5.25	4.46	4.29	19.86	16.06	11.01	8.09	4.28
	1.0	41.07	18.36	8.01	5.78	4.73	33.12	26.03	18.48	13.93	5.98
$\hat{\sigma}$		367.43	133.37	32.72	9.99	**3.17**	299.71	183.83	71.71	45.20	**3.26**

FID

DDIM 2

Source *Lugmayr, Andreas, et al. "Repaint: Inpainting using denoising diffusion probabilistic models." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.*

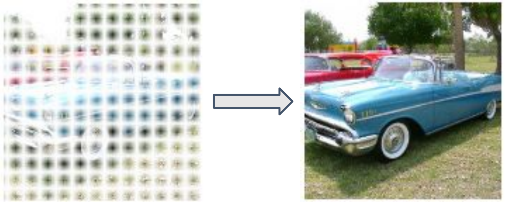
Other task 1

Diffusion models can solve a variety of tasks. We already know about image generation, as well as conditional image generation (for instance with a short paragraph describing the picture)

Other tasks:

- ➔ Inpainting

- ➔ Super-resolution



- ➔ Outpainting

Other task 2

We can solve many of these tasks through the usage of a mask

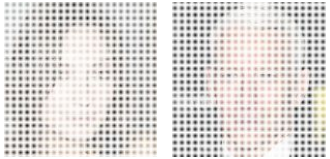
- ➔ Wide mask

- ➔ Thin mask

- ➔ Outer mask for expanding the image

- ➔ Right side mask for halving the image

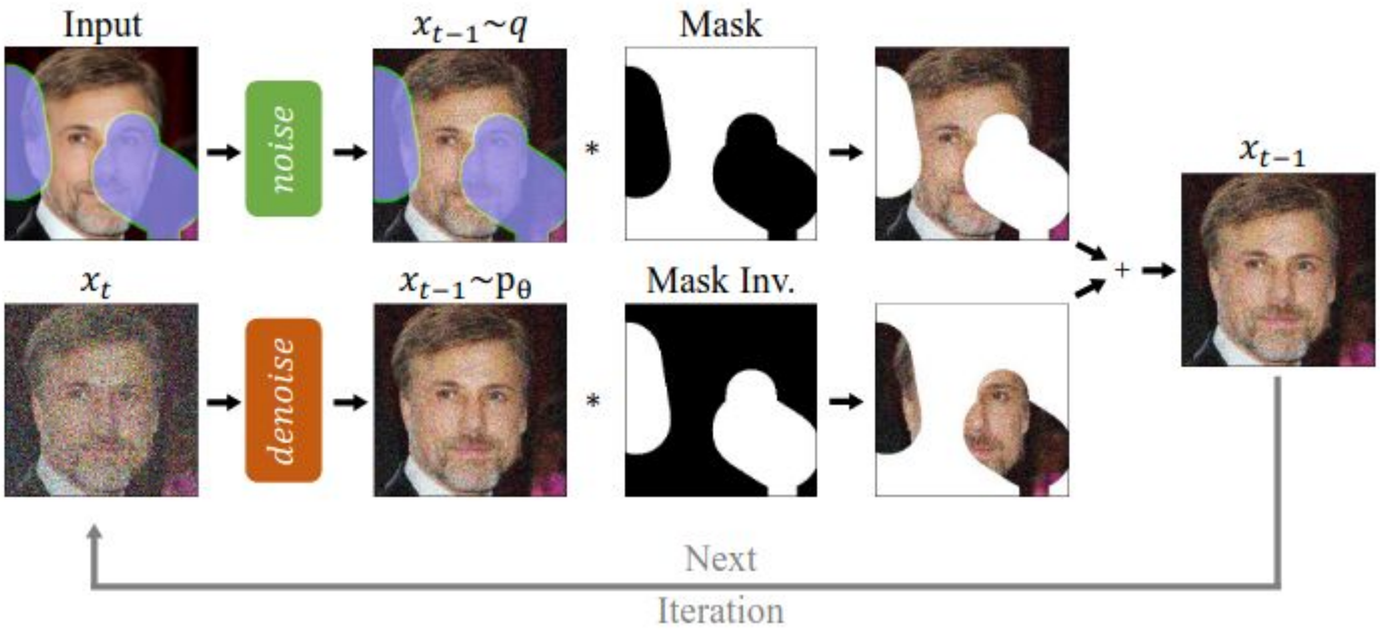
- ➔ Every second pixel in both directions for super-resolution



- ➔ Every second row of pixels for alternating lines

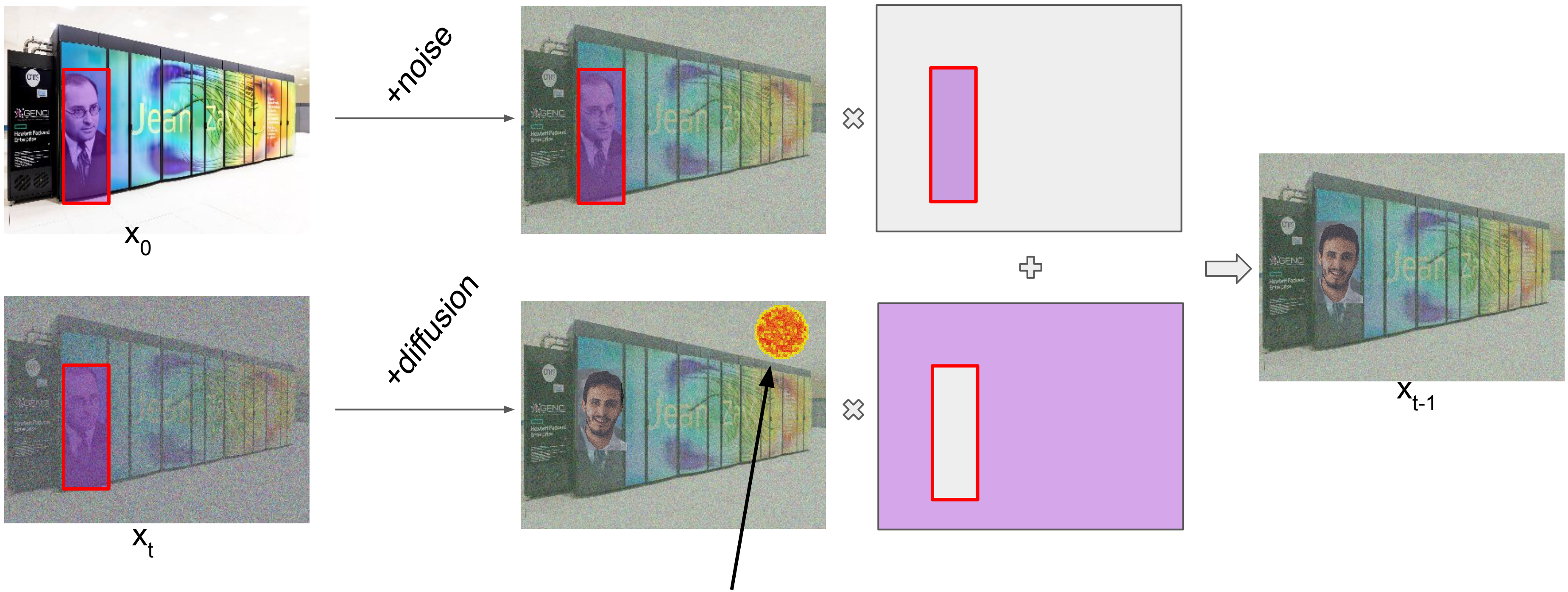
Source *Lugmayr, Andreas, et al. "Repaint: Inpainting using denoising diffusion probabilistic models." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.*

Step t :



Source *Lugmayr, Andreas, et al. "Repaint: Inpainting using denoising diffusion probabilistic models." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.*

Other task 3



New artifacts added (in this coarse example, our diffusion model drew a sun), so we force the known background again!

Other task 4

Papers:

- Deep Unsupervised Learning using Nonequilibrium Thermodynamics (DPM) (https://arxiv.org/abs/1503.03585)
- Denoising Diffusion Probabilistic Models (DDPM) (https://arxiv.org/abs/2006.11239)
- Improved Denoising Diffusion Probabilistic Models (IDDPM) (https://arxiv.org/abs/2102.09672)
- Denoising Diffusion Implicit Models (DDIM) (https://arxiv.org/abs/2010.02502)
- Diffusion Models Beat GANs on Image Synthesis (https://arxiv.org/abs/2105.05233)
- Classifier-free diffusion guidance (https://arxiv.org/pdf/2207.12598)
- Score-Based Generative Modeling through Stochastic Differential Equation (https://openreview.net/pdf?id=PxTIG12RRHS)
- Repaint: Inpainting using denoising diffusion probabilistic models (https://arxiv.org/pdf/2201.09865)
- Diffusion Models in Vision: A Survey (https://arxiv.org/abs/2209.04747)
- Diffusion Models: A Comprehensive Survey of Methods and Applications(https://arxiv.org/abs/2209.00796)

Other resources:

- Lilian Weng's article (https://lilianweng.github.io/posts/2021-07-11-diffusion-models)
- Yang Song's article (https://yang-song.net/blog/2021/score)
- Outlier video (https://www.youtube.com/watch?v=HoKDTa5jHvg)

Online Resources