Lecture d'article

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

NeurIPS 2020

Lecture de [J. Ho, A. Jain, P. Abbeel ; NeurIPS 2020]

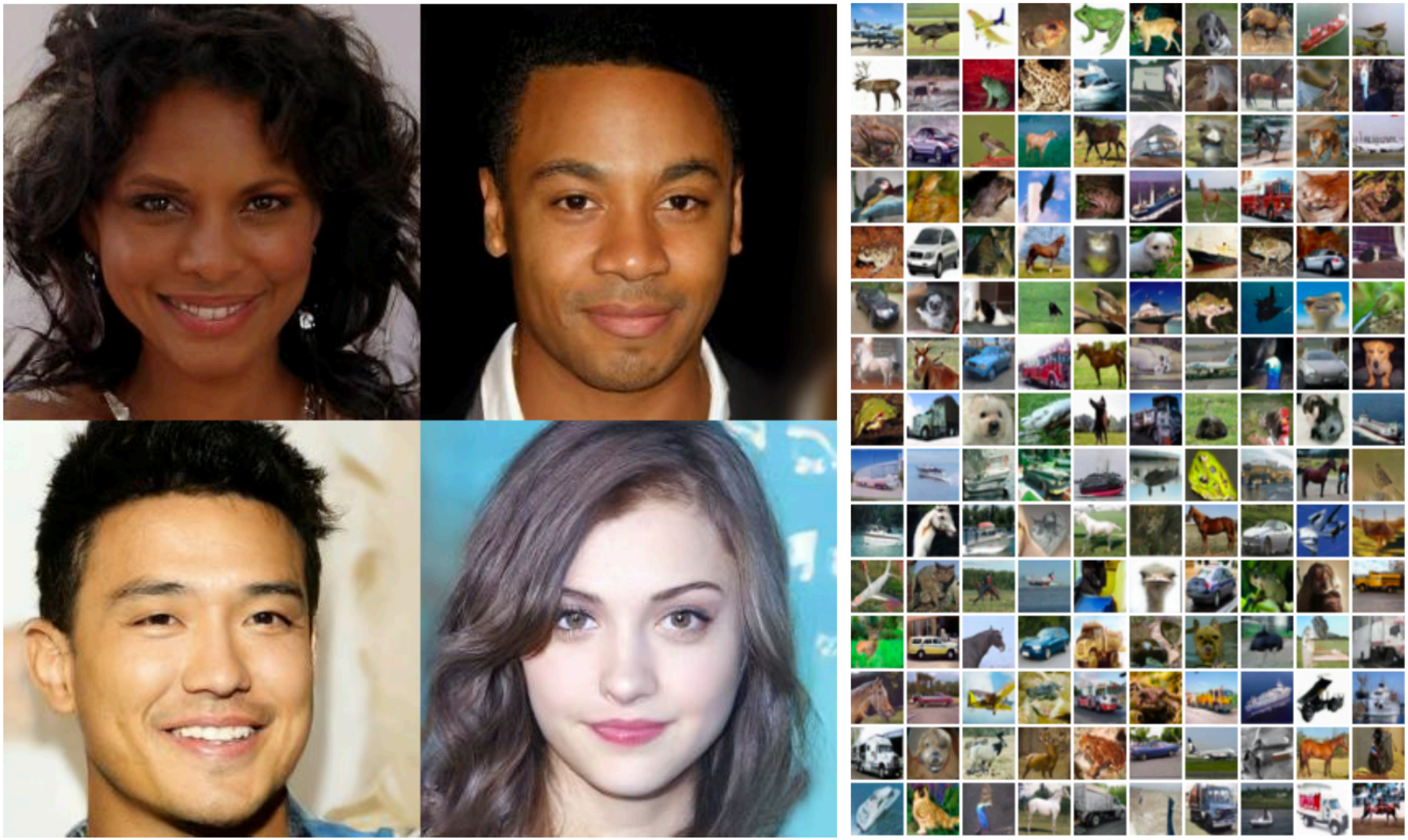


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

Lecture de [J. Ho, A. Jain, P. Abbeel ; NeurIPS 2020]

Partie 1 : Principe des modèles de diffusion [Sohl-Dickstein et al, 2015] (Section 2 de [Ho et al, 2020])

Partie 2 : Contributions de [Ho et al, 2020] pour générer des images complexes

Partie 3 : Aspects ingénierie et résultats de [Ho et al, 2020]

Partie 4 : Extensions à la génération *conditionnelle* de données

2 Background → Résumé de :

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

ICML & JMLR 2015

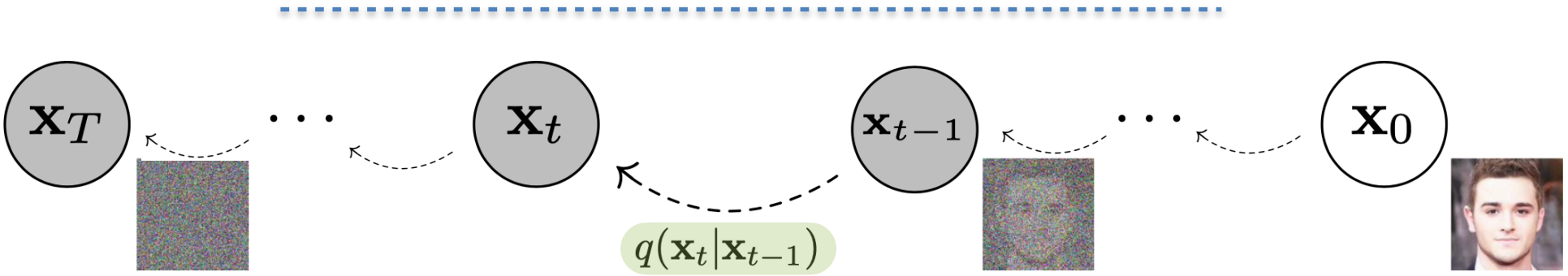
Soient

- Une liste de scalaires $\beta_1 \leq \beta_2 \leq \ldots \leq \beta_T$ ayant des valeurs $\in]0,1[$
- Une observation x_0 issue d'une distribution q *(ex : image, embedding, p scalaires, …)*

Pour $t = 1,2,\ldots,T$, on défini alors le **processus de diffusion *forward*** tel que :

- $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$ (densité de probabilité de x_t sachant x_{t-1})
- $q(x_{1:T}|x_0) = \prod_{i=1}^{T} q(x_t|x_{t-1})$ (densité de probabilité des $\{x_1, \ldots, x_T\}$ sachant x_0)

C'est ainsi une chaine de Markov qui ajoute progressivement du bruit à x_O



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

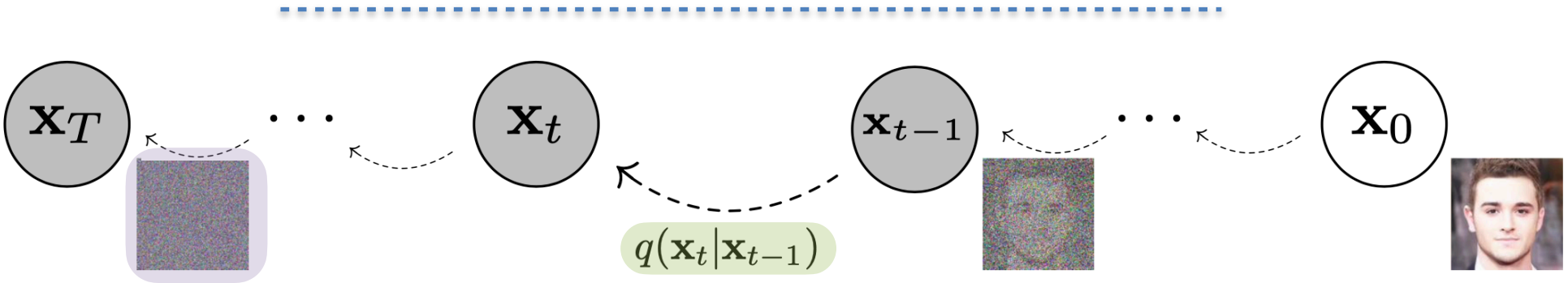
Si T est suffisamment grand, on peut raisonnablement considérer que :

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

On a en effet la propriété (très utile par la suite) :

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

Avec $\alpha_t = 1 - \beta_t$ et $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$



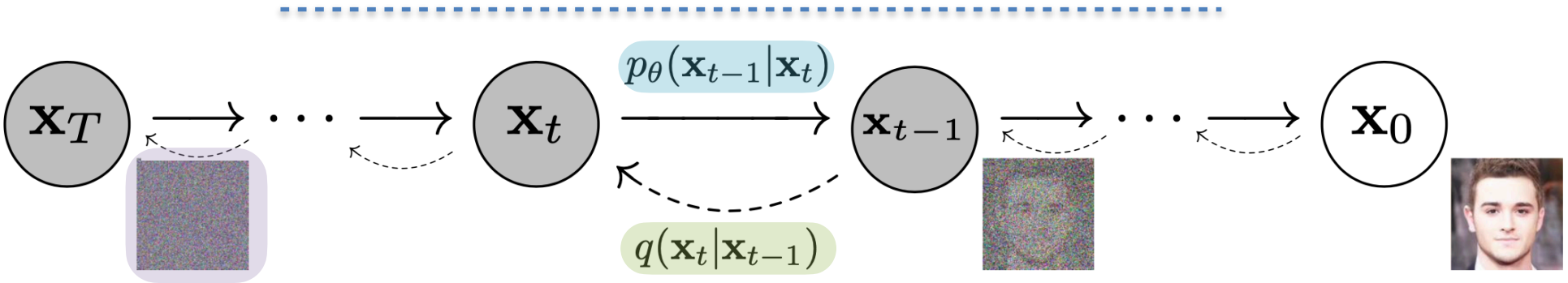
$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t I)$ $\quad$ $p(x_T) = \mathcal{N}(x_T; 0, I)$

On s'intéresse maintenant au processus de diffusion inverse (*reverse process*) qui est défini à partir de

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

avec des transitions dépendant de paramètres θ

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



$q(x_t\,|\,x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$ $\quad$ $p(x_T) = \mathcal{N}(x_T; 0, I)$ $\quad$ $p_\theta(x_{t-1}\,|\,x_t) = \mathcal{N}\left(x_{t-1}\,;\,\mu_\theta(x_t,t)\,,\Sigma_\theta(x_t,t)\right)$

On s'intéresse maintenant au processus de diffusion inverse (*reverse process*) qui est défini à partir de

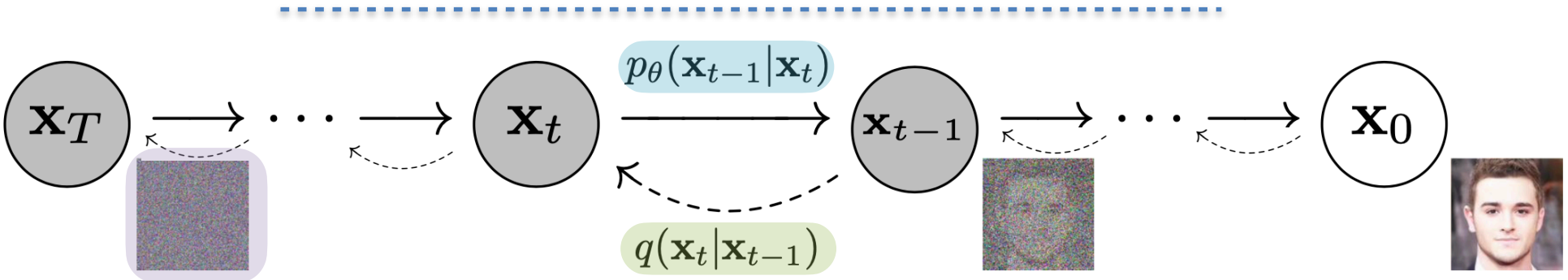
$$p(x_T) = \mathcal{N}(x_T; 0, I)\,,$$

avec des transitions dépendant de paramètres θ

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

On va plus tard optimiser une fonctionnelle qui dépend de $q(x_{1:T}\,|\,x_0)$, et de la densité de distribution jointe $p_\theta(x_{0:T})$:

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



$q(x_t\,|\,x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$ $\quad$ $p(x_T) = \mathcal{N}(x_T; 0, I)$ $\quad$ $p_\theta(x_{t-1}\,|\,x_t) = \mathcal{N}\left(x_{t-1}\,;\,\mu_\theta(x_t,t)\,,\Sigma_\theta(x_t,t)\right)$

On va alors considérer la *moins* log-vraisemblance de x_0 en fonction du modèle reverse p_θ et de l'hypothèse de bruit gaussien au temps T :

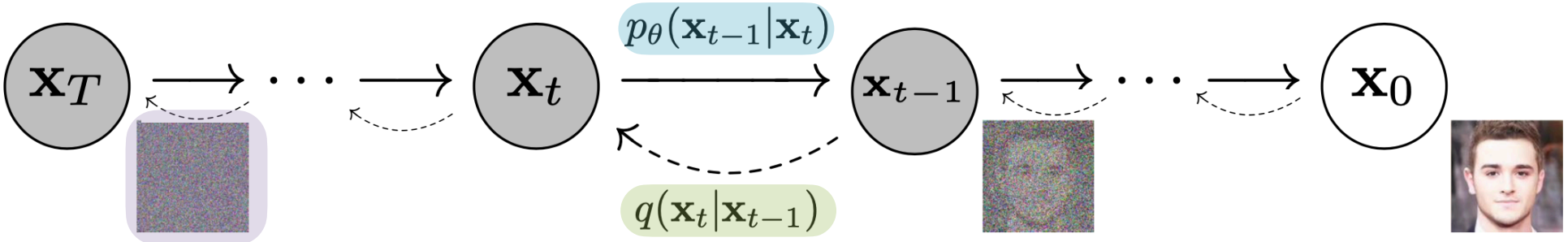
$$\mathbb{E}[-\log p_\theta(x_0)]$$

On peut donner une borne variationelle à ce terme qui sera minimisée pour l'apprentissage des θ

$$\mathbb{E}[-\log p_\theta(x_0)] \leq L$$

$$L = \mathbb{E}_q\left[-\log \frac{p_\theta\left(x_{0:T}\right)}{q\left(x_{1:T}|x_0\right)}\right] = \mathbb{E}_q\left[-\log p(x_T) - \sum_{t\geq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right]$$

… estimation à l'aide de technique de Monte-Carlo de forte variance (cf article)



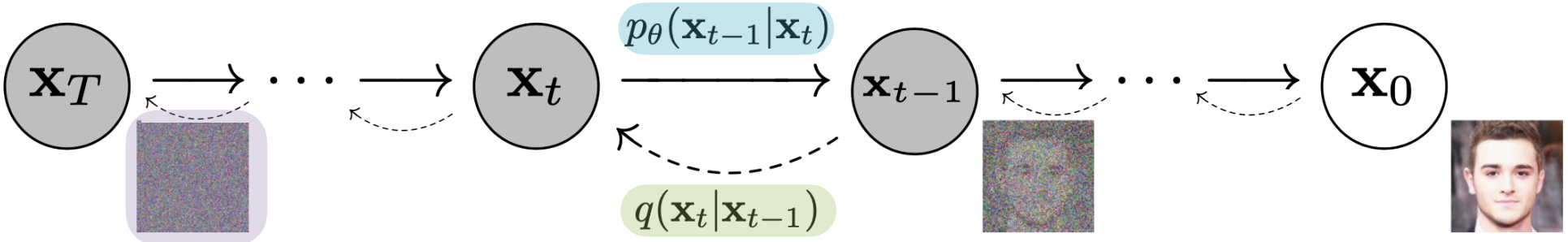
$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$ $\quad$ $p(x_T) = \mathcal{N}(x_T; 0, I)$ $\quad$ $p_\theta(x_{t-1}|x_t) = \mathcal{N}\left(x_{t-1}\,;\, \mu_\theta(x_t,t)\,, \Sigma_\theta(x_t,t)\right)$

Réécriture de la log-vraisemblance (cf Annexe A) : « all KL divergences are comparisons between Gaussians, so they can be calculated in a Rao-Blackwellized fashion with closed form expressions »

$$
\begin{aligned}
L &= \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \\
&\vdots \\
&= \mathbb{E}_q\left[-\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] \\
&\mathbb{E}_q\Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\,\|\,p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\,\|\,p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\Big] \qquad (5)
\end{aligned}
$$

$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I))$ avec $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$

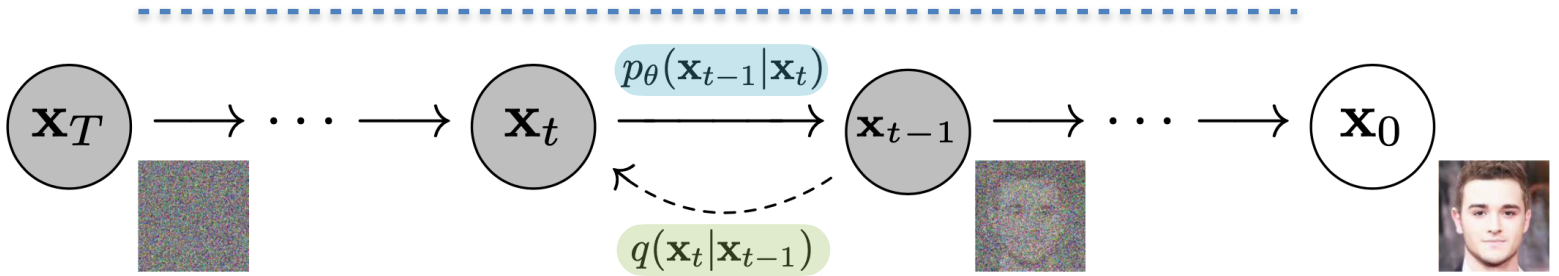
$$
\begin{aligned}
&q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0), \tilde{\beta}_t\mathbf{I}), \\
&\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \\
&\text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t
\end{aligned}
$$



$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$ $\quad$ $p(x_T) = \mathcal{N}(x_T; 0, I)$ $\quad$ $p_\theta(x_{t-1}|x_t) = \mathcal{N}\left(x_{t-1}\,;\,\mu_\theta(x_t,t)\,,\,\Sigma_\theta(x_t,t)\right)$

3 Diffusion models and denoising autoencoders

Diffusion models might appear to be a restricted class of latent variable models, but they allow a large number of degrees of freedom in implementation. One must choose the variances β_t of the forward process and the model architecture and Gaussian distribution parameterization of the reverse process. To guide our choices, we establish a new explicit connection between diffusion models and denoising score matching (Section 3.2) that leads to a simplified, weighted variational bound objective for diffusion models (Section 3.4). Ultimately, our model design is justified by simplicity and empirical results (Section 4). Our discussion is categorized by the terms of Eq. (5).

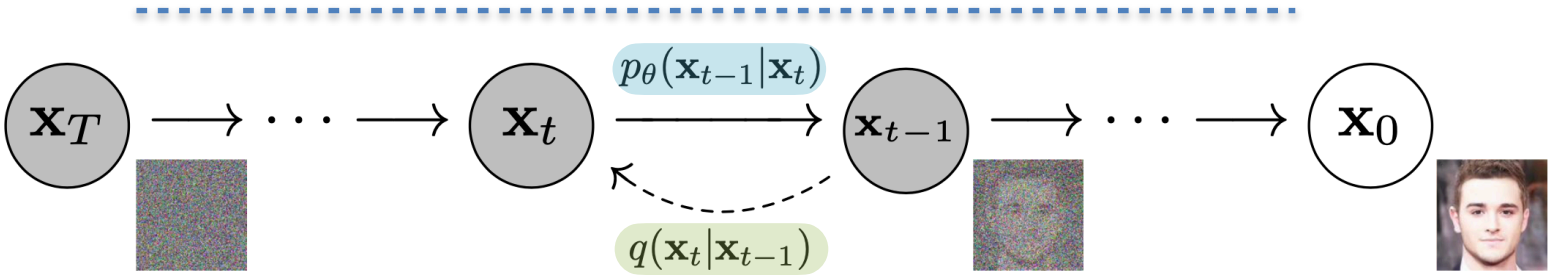


$$L = \mathbb{E}_q \Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Big] \quad (5)$$

3.2 Reverse process and $L_{1:T-1}$

Now we discuss our choices in $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$ for $1 < t \leq T$.

First we set $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$ to untrained time dependent constants.



$$L = \mathbb{E}_q \Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Big]$$

3.2 Reverse process and $L_{1:T-1}$

Now we discuss our choices in $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$ for $1 < t \leq T$.

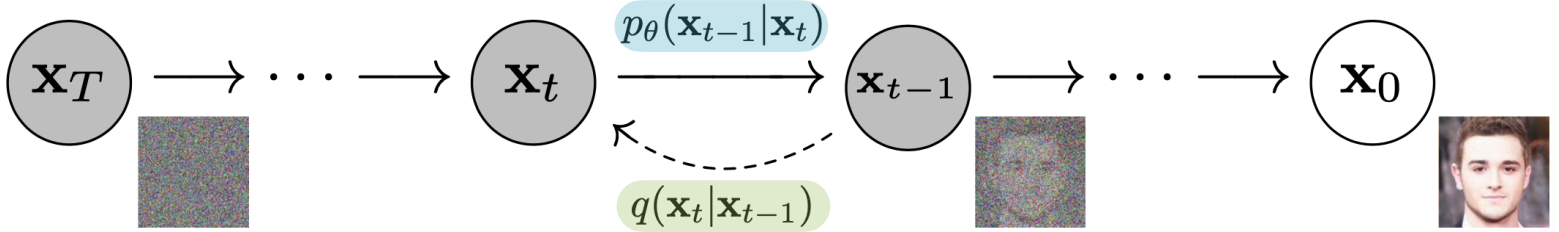
First, we set $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$ to untrained time dependent constants.

Second, to represent the mean $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$, we propose a specific parameterization motivated by the following analysis of L_t. With $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$, we can write:

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C \tag{8}$$

where C is a constant that does not depend on θ. So, we see that the most straightforward parameterization of $\boldsymbol{\mu}_\theta$ is a model that predicts $\tilde{\boldsymbol{\mu}}_t$, the forward process posterior mean.

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



$$L = \mathbb{E}_q \Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Big]$$

3.2 Reverse process and $L_{1:T-1}$

Now we discuss our choices in $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$ for $1 < t \leq T$.

First we set $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$ to untrained time dependent constants.

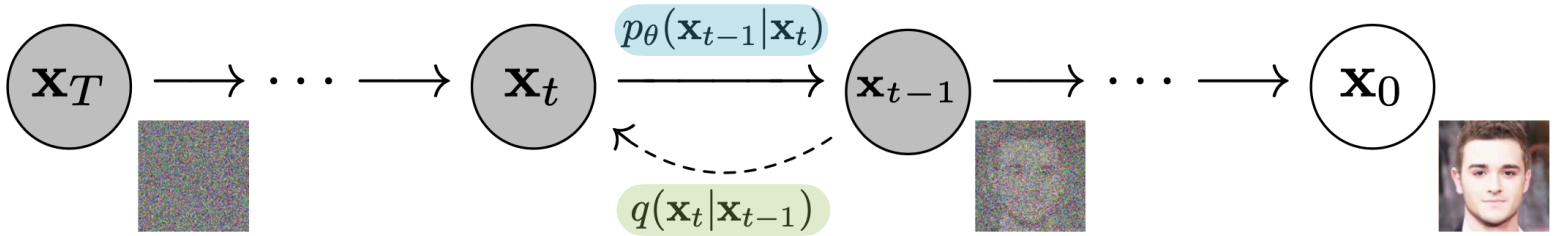
Second, to represent the mean $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$, we propose a specific parameterization motivated by the following analysis of L_t. With $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$, we can write:

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C \tag{8}$$

where C is a constant that does not depend on θ. So, we see that the most straightforward parameterization of $\boldsymbol{\mu}_\theta$ is a model that predicts $\tilde{\boldsymbol{\mu}}_t$, the forward process posterior mean.

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

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

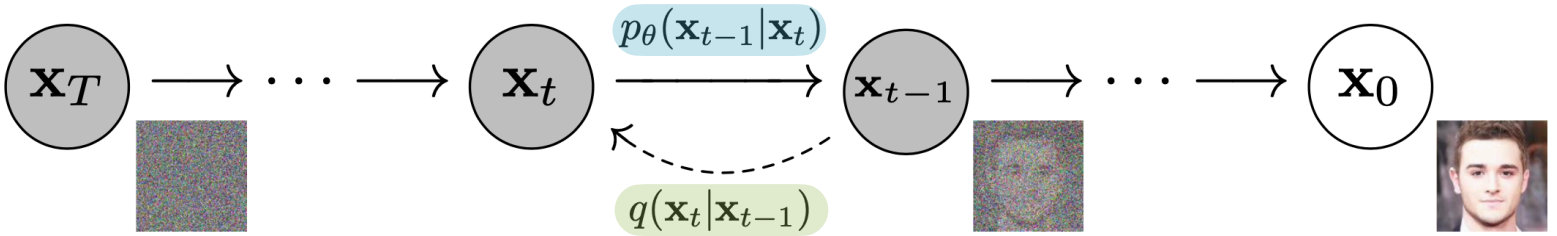


$$L = \mathbb{E}_q \Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Big]$$

3.2 Reverse process and $L_{1:T-1}$

On utilise alors $\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ et $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}), t)\right\|^2\right] \tag{10}$$



$$L = \mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\right]$$

3.2 Reverse process and $L_{1:T-1}$

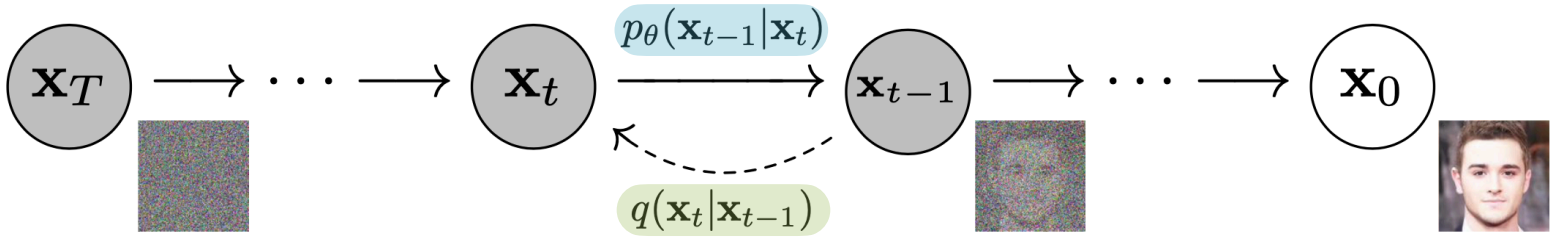
On utilise alors $\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ et $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}), t)\right\|^2\right] \tag{10}$$

Since $\mathbf{x}_t$ is available as input to the model, we may choose the parameterization

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) \tag{11}$$

where $\boldsymbol{\epsilon}_\theta$ is a function approximator intended to predict $\boldsymbol{\epsilon}$ from $\mathbf{x}_t$.



$$L = \mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\right]$$

3.2 Reverse process and $L_{1:T-1}$

On utilise alors $\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ et $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$:

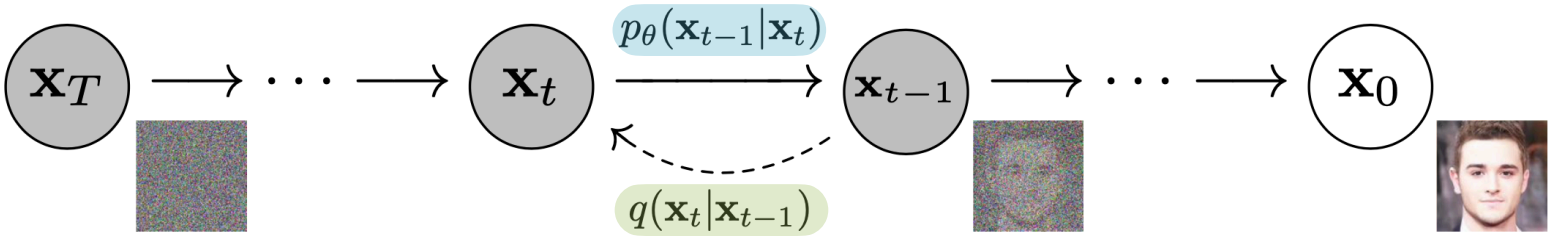
$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}), t)\right\|^2\right] \tag{10}$$

Since $\mathbf{x}_t$ is available as input to the model, we may choose the parameterization

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) \tag{11}$$

where $\boldsymbol{\epsilon}_\theta$ is a function approximator intended to predict $\boldsymbol{\epsilon}$ from $\mathbf{x}_t$.

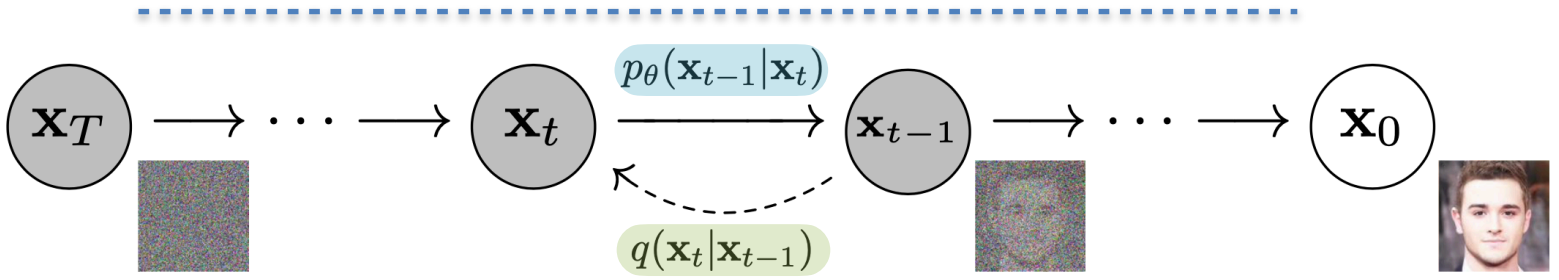
… with (11), Eq. (10) simplifies to: $\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$ (12)



$$L = \mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\right]$$

3.1 Forward process and L_T

We ignore the fact that the forward process variances β_t are learnable by reparameterization and instead fix them to constants (see Section 4 for details). Thus, in our implementation, the approximate posterior q has no learnable parameters, so L_T is a constant during training and can be ignored.



$$L = \mathbb{E}_q \Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Big]$$

3.3 Data scaling, reverse process decoder, and L_0

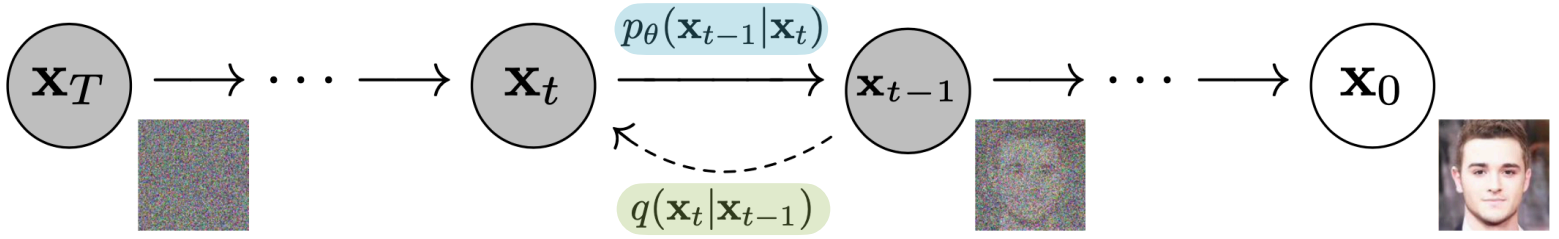
We assume that image data consists of integers in $\{0, 1, \dots, 255\}$ scaled linearly to $[-1, 1]$. This ensures that the neural network reverse process operates on consistently scaled inputs starting from the standard normal prior $p(\mathbf{x}_T)$. To obtain discrete log likelihoods, we set the last term of the reverse process to an independent discrete decoder derived from the Gaussian $\mathcal{N}(\mathbf{x}_0; \boldsymbol{\mu}_\theta(\mathbf{x}_1, 1), \sigma_1^2 \mathbf{I})$:

$$p_\theta(\mathbf{x}_0|\mathbf{x}_1) = \prod_{i=1}^{D} \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} \mathcal{N}(x; \mu_\theta^i(\mathbf{x}_1, 1), \sigma_1^2)\, dx \tag{13}$$

$$\delta_+(x) = \begin{cases} \infty & \text{if } x = 1 \\ x + \frac{1}{255} & \text{if } x < 1 \end{cases} \qquad \delta_-(x) = \begin{cases} -\infty & \text{if } x = -1 \\ x - \frac{1}{255} & \text{if } x > -1 \end{cases}$$

where D is the data dimensionality and the i superscript indicates extraction of one coordinate.

Similar to the discretized continuous distributions used in VAE decoders and autoregressive models [34, 52], our choice here ensures that the variational bound is a lossless codelength of discrete data, without need of adding noise to the data or incorporating the Jacobian of the scaling operation into the log likelihood.



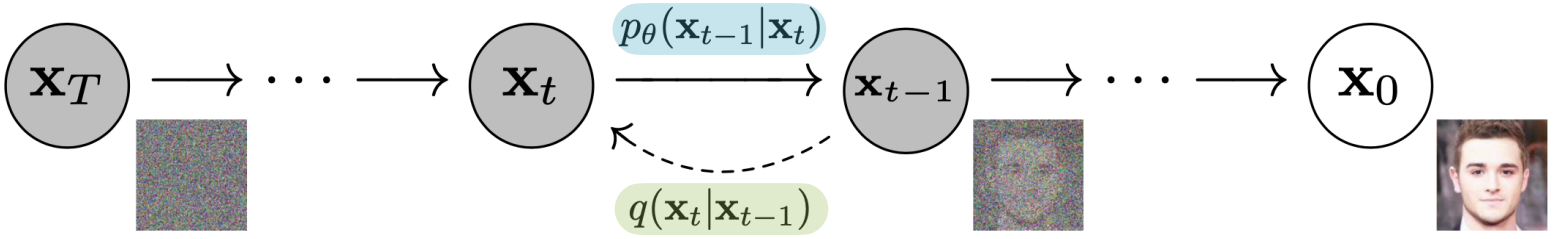
$$L = \mathbb{E}_q \Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Big]$$

3.4 Simplified training objective

… we found it beneficial to sample quality (and simpler to implement) to train on the following variant of the variational bound:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right] \tag{14}$$

where t is uniform between 1 and T. …



$$L = \mathbb{E}_q\left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}\underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\right]$$

3.2 Reverse process and $L_{1:T-1}$

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

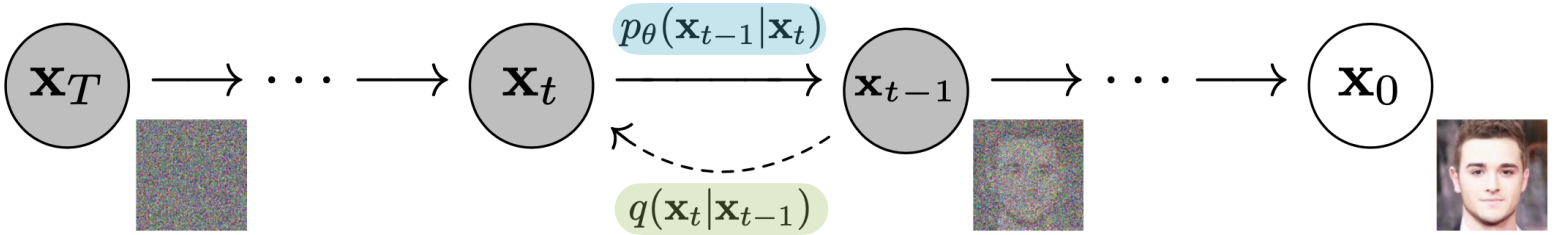
Algorithm 2 Sampling

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** $\mathbf{x}_0$

Estimation de ϵ à partir de x_t et t

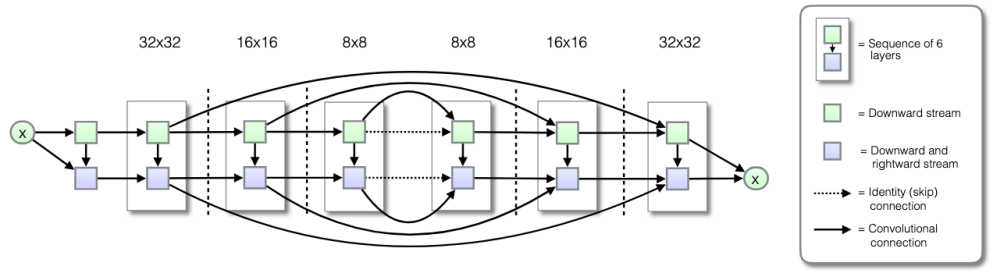
Echantillonnage d'un x_t à partir de x_0 suivant Eq. (4)

$\mu_\theta(x_t, t)$ utilisé dans
$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\left(x_{t-1}\,;\, \mu_\theta(x_t, t)\,, \sigma_t I\right)$



$$L = \mathbb{E}_q \left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \right]$$

This structure resembles the VAE model with top down inference used by Kingma et al. (2016), as well as the U-net used by Ronneberger et al. (2015) for image segmentation. Figure 2 shows our model structure graphically.

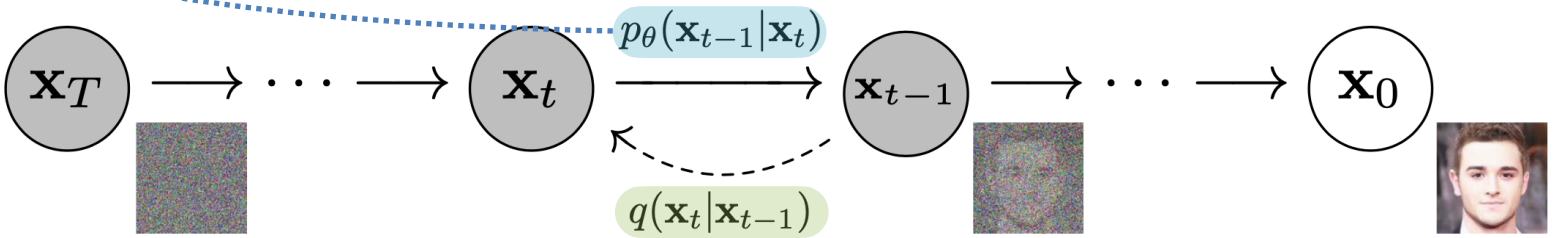


Salimans et al, ICLR 2017

To represent the reverse process, we use a U-Net backbone similar to an unmasked PixelCNN++ [52, 48] with group normalization throughout [66]. Parameters are shared across time, which is specified to the network using the Transformer sinusoidal position embedding [60]. We use self-attention at the 16×16 feature map resolution [63, 60]. Details are in Appendix B.

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks.

Vaswani et al et al, NeurIPS 2017



Recommendations pratiques de [Nichol et Dhariwal, PMLR 2021] :

The setup from Ho et al. (2020) (optimizing L_{simple} while setting $\sigma_t^2 = \beta_t$ and $T = 1000$) achieves a log-likelihood of 3.99 (bits/dim) on ImageNet 64×64 after 200K training iterations. We found in early experiments that we could get a boost in log-likelihood by increasing T from 1000 to 4000; with this change, the log-likelihood improves to 3.77.

In Ho et al. (2020), the authors set $\Sigma_\theta(x_t, t) = \sigma_t^2 \mathbf{I}$, where σ_t is not learned. Oddly, they found that fixing σ_t^2 to β_t yielded roughly the same sample quality as fixing it to $\tilde{\beta}_t$.

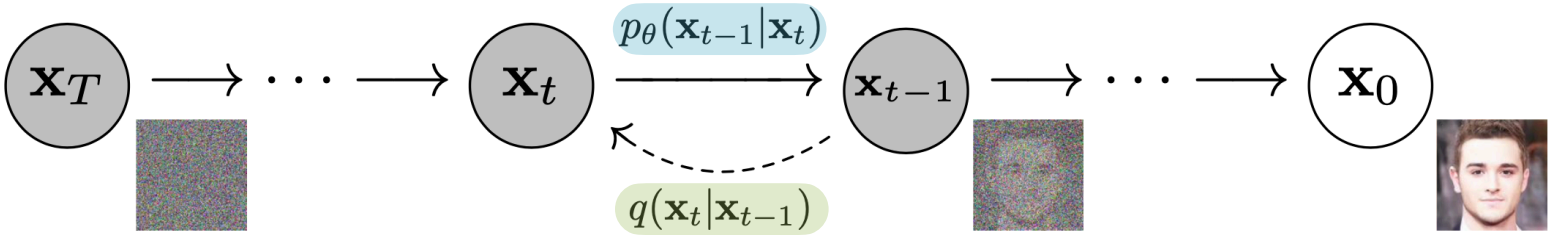
...

we construct a different noise schedule in terms of $\bar{\alpha}_t$:

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2 \tag{17}$$

For all of our experiments, we use a UNet model architecture[4] similar to that used by Ho et al. (2020). We changed the attention layers to use multi-head attention (Vaswani et al., 2017), and opted to use four attention heads rather than one (while keeping the same total number of channels). We employ attention not only at the 16x16 resolution, but also at the 8x8 resolution. Additionally, we changed the way the model conditions on t. In particular, instead of computing a conditioning vector v and injecting it into hidden state h as $\text{GroupNorm}(h + v)$, we compute conditioning vectors w and b and inject them into the hidden state as $\text{GroupNorm}(h)(w + 1) + b$.

We have shown that, with a few modifications, DDPMs can sample much faster and achieve better log-likelihoods with little impact on sample quality. The likelihood is improved by learning Σ_θ using our parameterization and L_{hybrid} objective. This brings the likelihood of these models much closer to other likelihood-based models. We surprisingly discover that this change also allows sampling from these models with many fewer steps.



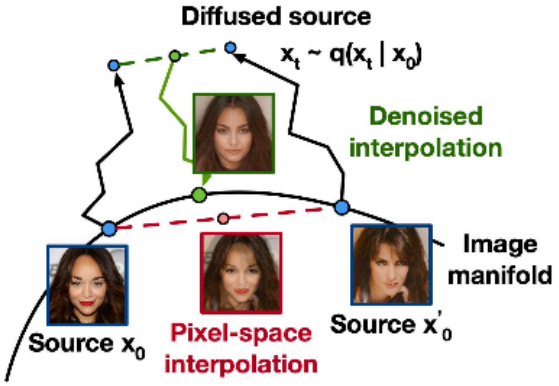
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Figure 6: Unconditional CIFAR10 progressive generation ($\hat{\mathbf{x}}_0$ over time, from left to right). Extended samples and sample quality metrics over time in the appendix (Figs. 10 and 14).

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$



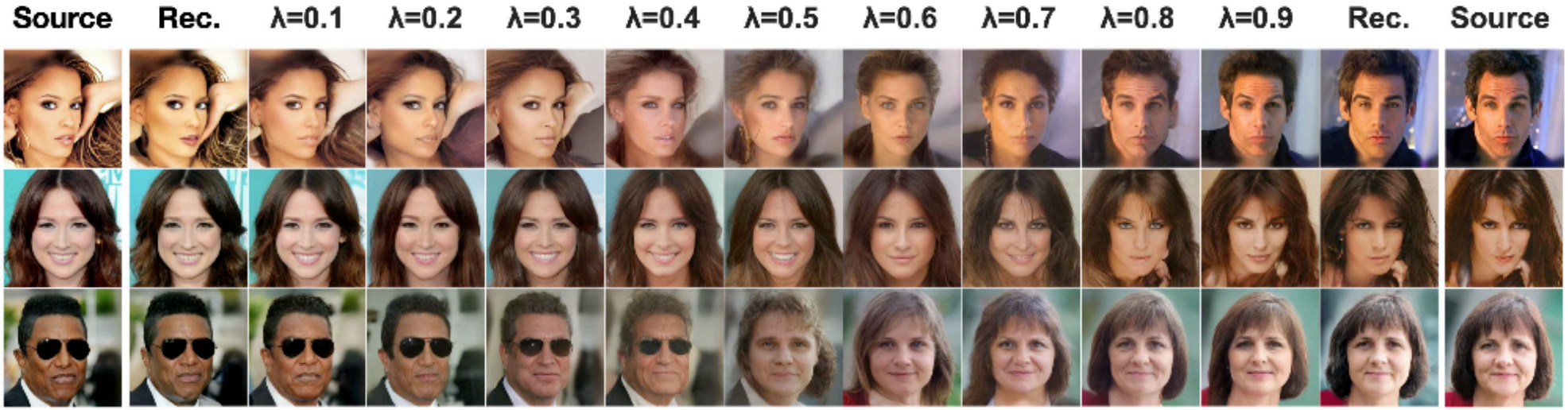


Figure 8: Interpolations of CelebA-HQ 256x256 images with 500 timesteps of diffusion.

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Figure 7: When conditioned on the same latent, CelebA-HQ 256×256 samples share high-level attributes. Bottom-right quadrants are $\mathbf{x}_t$, and other quadrants are samples from $p_\theta(\mathbf{x}_0|\mathbf{x}_t)$.

Extrait de [Ho et al. JMLR 2021] :

2.2 Conditional Diffusion Models

In the conditional generation setting, the data $\mathbf{x}_0$ has an associated conditioning signal $\mathbf{c}$, for example a label in the case of class-conditional generation, or a low resolution image in the case of super-resolution (Saharia et al., 2021; Nichol and Dhariwal, 2021). The goal is then to learn a conditional model $p_\theta(\mathbf{x}_0|\mathbf{c})$. To do so, we modify the diffusion model to include $\mathbf{c}$ as input to the reverse process:

$$p_\theta(\mathbf{x}_{0:T}|\mathbf{c}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{c}), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{c}) = \mathcal{N}(\mathbf{x}_{t-1};\boldsymbol{\mu}_\theta(\mathbf{x}_t,t,\mathbf{c}),\boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t,\mathbf{c}))$$

$$L_\theta(\mathbf{x}_0|\mathbf{c}) = \mathbb{E}_q\left[L_T(\mathbf{x}_0) + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{c})) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1,\mathbf{c})\right].$$

The data and conditioning signal $(\mathbf{x}_0,\mathbf{c})$ are sampled jointly from the data distribution, now called $q(\mathbf{x}_0,\mathbf{c})$, and the forward process $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ remains unchanged. The only modification that needs to be made is to inject $\mathbf{c}$ as a extra input to the neural network function approximators: instead of $\boldsymbol{\mu}_\theta(\mathbf{x}_t,t)$ we now have $\boldsymbol{\mu}_\theta(\mathbf{x}_t,t,\mathbf{c})$, and likewise for $\boldsymbol{\Sigma}_\theta$. The particular architectural choices for injecting these extra inputs depends on the type of the conditioning $\mathbf{c}$, as described next.

Lecture de [Ho et al. ; NeurIPS 2020] → Partie 4 : génération conditionnelle de données

High-Resolution Image Synthesis with Latent Diffusion Models

Robin Rombach[1] * Andreas Blattmann[1] * Dominik Lorenz[1] Patrick Esser[R] Björn Ommer[1]
[1]Ludwig Maximilian University of Munich & IWR, Heidelberg University, Germany [R]Runway ML
https://github.com/CompVis/latent-diffusion

SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song* Stanford University yangsong@cs.stanford.edu
Jascha Sohl-Dickstein Google Brain jaschasd@google.com
Diederik P. Kingma Google Brain durk@google.com
Abhishek Kumar Google Brain abhishk@google.com
Stefano Ermon Stanford University ermon@cs.stanford.edu
Ben Poole Google Brain pooleb@google.com

Hierarchical Text-Conditional Image Generation with CLIP Latents

Aditya Ramesh* OpenAI aramesh@openai.com
Prafulla Dhariwal* OpenAI prafulla@openai.com
Alex Nichol* OpenAI alex@openai.com
Casey Chu* OpenAI casey@openai.com
Mark Chen OpenAI mark@openai.com

CLASSIFIER-FREE DIFFUSION GUIDANCE

Jonathan Ho & Tim Salimans
Google Research, Brain team
{jonathanho,salimans}@google.com

Cascaded Diffusion Models for High Fidelity Image Generation

Jonathan Ho* JONATHANHO@GOOGLE.COM
Chitwan Saharia* SAHARIAC@GOOGLE.COM
William Chan WILLIAMCHAN@GOOGLE.COM
David J. Fleet DAVIDFLEET@GOOGLE.COM
Mohammad Norouzi MNOROUZI@GOOGLE.COM
Tim Salimans SALIMANS@GOOGLE.COM

DiffusionCLIP: Text-Guided Diffusion Models for Robust Image Manipulation

Gwanghyun Kim[1] Taesung Kwon[1] Jong Chul Ye[2,1]
Dept. of Bio and Brain Engineering[1], Kim Jaechul Graduate School of AI[2]
Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea

… and many many others since 2020 …

An Overview of Diffusion Models: Applications, Guided Generation, Statistical Rates and Optimization*

Minshuo Chen[1] Song Mei[2] Jianqing Fan[1] Mengdi Wang[1]
[1]Princeton University [2]UC Berkeley

Cours 6 et 7 du MVA sur les modèles génératifs (Bruno Galerne) : https://generativemodelingmva.github.io/