Mémoire: Modèles de diffusion

Alexis Solar

25/08/2024

Résumé

Les modèles de diffusion par débruitage sont des modèles génératifs profonds qui sont aujourd'hui très populaires et qui ont montré des résultats remarquables, notamment par la qualité des échantillons qu'ils produisent. Ce mémoire fait la revue de la méthode initiale ayant rendu populaire ce type de modèle et l'illustre par un cas pratique : l'entrainement d'un réseau de neurone U-net pour la génération d'image. Il a pour but de contribuer à une meilleure compréhension des modèles de diffusion par débruitage et leur application à la génération d'image. L'implémentation est disponible sur mon github.

1 Introduction

Ces dernières années, plusieurs types de modèles génératifs profonds comme les GANs (Generative Adversarial Networks)[14] et VAEs (Variational Autoencoder) [15] ont montré des avancées impressionnantes au niveau visuel (génération d'image, défloutage, upscaling d'image), audio, de la détection d'anomalie, réduction de dimension, etc.

En 2015, l'article [3] a introduit un nouveau type de modèle génératif profond : les modèles de diffusion. Aujourd'hui ces modèles sont utilisés en vision par ordinateur, reconstruction d'image médicale, traitement du langage naturel, etc. Dans un domaine comme la génération d'image, ces modèles se sont imposés face aux GANs ou VAEs comme étant à la fois performants et flexibles. En 2020, [1] a contribué à cette tendance en présentant la génération d'image de qualité grâce à ce type de modèle. C'est cette méthode qui sera principalement présentée.

La méthode consiste dans un premier temps à bruiter des images petit à petit avec du bruit gaussien jusqu'à obtenir des images totalement bruitées et puis dans un second temps à entrainer un réseau de neurone à faire le chemin inverse et débruiter ces images pour finalement pouvoir générer des images seulement à partir de bruit gaussien.

Plus formellement, le modèle de diffusion est une chaine de Markov avec un paramètre entrainé grâce à l'inférence variationnelle afin de produire des échantillons ressemblants aux données de départ. Ce modèle apprend à faire le chemin inverse du processus de diffusion, qui est une chaine de Markov qui ajoute un peu de bruit gaussien à chaque transition.

2 Notations

Définition 2.1. Soit T, le nombre d'étapes de bruitage fixé. (Ici fixé à 1000)
On note x_0 l'image de départ. On a donc x_t avec $0 \leq t \leq T$ représentant l'image x_0 bruitée avec t étapes d'ajout de bruit gaussien. (Figure 1)

FIGURE 1 – Représentation des étapes de bruitage entre x_0 (tout à gauche) et x_T (tout à droite)

Définition 2.2. Soit β_t la variance du bruit gaussien à l'étape t. Elle est fixée et croit linéairement de $\beta_1 = 0.0001$ à $\beta_T = 0.02$. (Figure 2)

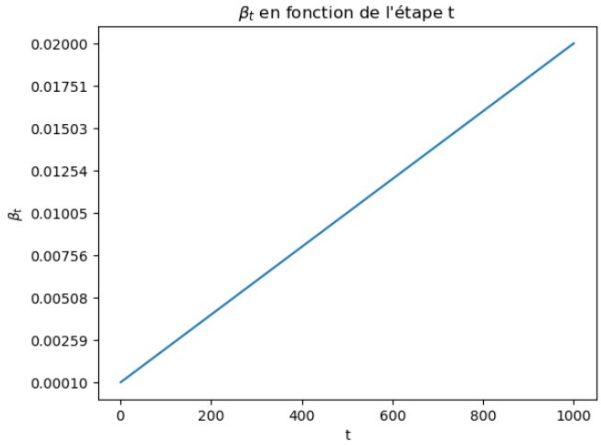

FIGURE 2 – Croissance de β_t en fonction de t avec T=1000

Remarque 2.3. Ici le choix de fixer la variance est fait comme dans l'article DDPM [1] mais elle pourrait également être apprise par le modèle comme c'est le cas dans d'autres méthodes.

Définition 2.4. Le processus de bruitage q, appelé forward process ou diffusion process, est donné par :

$$q(x_1, ..., x_T \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

et

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

où les paramètres de la loi normale sont :
- le résultat x_t
- la moyenne $\sqrt{1-\beta_t}x_{t-1}$
- la variance $\beta_t I$

avec l'image de départ $x_0 \sim q(x_0)$

Définition 2.5. Le processus inverse p est une chaine de Markov définie par la distribution jointe :

$$p_\theta(x_0, ..., x_T) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)$$

et

$$p_\theta(x_{t-1} \mid x_t) := \mathcal{N}(x_{t-1}, \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

où les paramètres de la loi normale sont :
- le résultat x_{t-1}
- la moyenne $\mu_\theta(x_t, t)$
- la variance $\Sigma_\theta(x_t, t)$

avec $p(x_T) \sim \mathcal{N}(0, 1)$

Remarque 2.6. Le paramètre θ sera optimisé durant l'apprentissage afin que $\mu_\theta(x_t, t)$ permette de retrouver l'image moins bruitée x_{t-1} à partir de x_t. La variance $\Sigma_\theta(x_t, t)$ par contre est fixée à chaque étape comme vue précedemment.

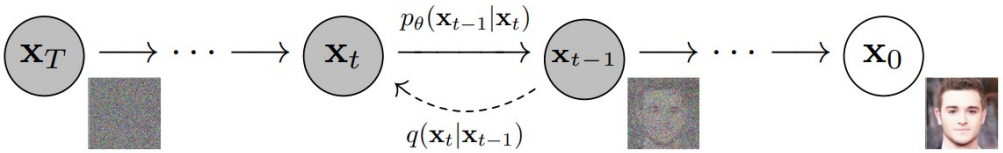

FIGURE 3 – Illustration du principe de bruitage/débruitage provenant du papier DDPM [1].

3 Fonction de coût

Définition 3.1. La fonction de coût $L(\theta)$ est définie par :

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

Remarque 3.2. Cette fonction de perte est utilisée dans le but de maximiser la vraisemblance de l'image prédite par le modèle (appelé "Likelyhood Based") ce qui revient à minimiser $\mathbb{E}[-logp_\theta(x_0)]$.

La fonction de coût n'étant pas calculable directement, on utilise la méthode "Variational Lower Bund" afin de la majorer pour ensuite minimiser cette borne supérieure. [3]

Proposition 3.3. Pour tout θ,

$$\mathbb{E}[-\log p_\theta(x_0)] \leq \mathbb{E}_q[-\log \frac{p_\theta(x_0,...,x_T)}{q(x_1,...,x_T \mid x_0)}] = \mathbb{E}_q[-\log p(x_T) - \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})}] =: L_{VLB}(\theta)$$

4 Divergence de Kullback-Leibler

Définition 4.1. On définit la divergence de Kullback-Leibler afin de mesurer la différence entre deux distributions de probabilités. Soient P et Q deux lois de probabilité continues de densités respectives p et q :

$$D_{KL}(P \,||\, Q) := \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$$

Remarque 4.2. Cela nous permet de simplifier la notation de la fonction de coût $L_{VLB}(\theta)$ afin de minimiser chaque terme de celle-ci.

Proposition 4.3.

$$L_{VLB}(\theta) = \mathbb{E}_q[\underbrace{D_{KL}(q(x_T \mid x_0) \,||\, p(x_T))}_{L_T} + \sum_{t=2}^{T} \underbrace{D_{KL}(q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t))}_{L_{t-1}} - \underbrace{\log p_\theta(x_0 \mid x_1)}_{L_0}]$$

Remarques 4.4. Le terme L_T ne dépendant pas du paramètre θ, il est constant et on peut donc l'ignorer (à priori, avec T assez grand, ce terme devrait être faible. Cette idée sera développée en annexe).

L_t pour $1 \leq t \leq T-1$ est donc la divergence de Kullback entre deux lois normales. Les variances de celles-ci étant constantes, on cherche à évaluer la capacité du modèle à prédire la bonne moyenne.

Le cas de L_0 sera étudié ensuite.

Proposition 4.5. Pour tout $1 \leq t \leq T$,

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

avec $\tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\tilde{\alpha}_t}}\epsilon_t)$ et $\tilde{\beta}_t = \frac{1-\tilde{\alpha}_{t-1}}{1-\tilde{\alpha}_t}\beta_t$

Définition 4.6. Afin de minimiser L_t pour $1 \leq t \leq T-1$, on cherche donc à entrainer notre modèle $p_\theta(x_{t-1} \mid x_t) := \mathcal{N}(x_{t-1}, \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$ à trouver $q(x_{t-1} \mid x_t, x_0) \sim \mathcal{N}(x_{t-1}, \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)$.

Pour cela on paramétrise $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\tilde{\alpha}_t}}\epsilon_\theta(x_t, t))$ où $\epsilon_\theta(x_t, t)$ est le réseau de neuronne à entrainer et $\Sigma_\theta(x_t, t)) = \sigma_t^2 I$ avec $\sigma_t^2 = \beta_t$ ou $\tilde{\beta}_t$ (Dans le papier DDPM, il est précisé que les résultats expérimentaux sont similaires en utilisant β_t ou $\tilde{\beta}_t$)

Proposition 4.7. pour $1 \leq t \leq T-1$ on peut simplifier L_t :

$$L_t = \mathbb{E}_{x_0, \epsilon_t}[\frac{(1-\alpha_t)^2}{2\sigma_t^2 \alpha_t (1-\tilde{\alpha}_t)} \left\| \epsilon_t - \epsilon_\theta(\sqrt{\tilde{\alpha}_t}x_0 + \sqrt{1-\tilde{\alpha}_t}\epsilon_t, t) \right\|^2] + C$$

Définition 4.8. Afin de calculer L_0, on suppose que les données sont des vecteurs de D pixels où chaque pixel est lui même un vecteur de dimension 3 (pour les 3 couleurs primaires RGB) à valeurs entre 0 et 255 normalisés linéairement entre -1 et 1 pour le réseau de neuronne.
Ensuite on calcule la vraisemblance discrète :

$$p_\theta(x_0 \mid \ x_1) = \prod_{i=1}^{D} \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} \mathcal{N}(x, \mu_\theta^i(x_1, 1), \sigma_1^2) dx$$

où $\quad \delta_+(x_0^i) = \begin{cases} \infty & \text{si } x = 1 \\ x + \frac{1}{255} & \text{si } x < 1 \end{cases} \quad \delta_-(x_0^i) = \begin{cases} \infty & \text{si } x = -1 \\ x - \frac{1}{255} & \text{si } x > -1 \end{cases}$

Remarque 4.9. Après plusieurs simplifications de la fonction de coût $L_{VLB}(\theta)$ dus à la paramétrisation du modèle, on observe finalement que l'on évalue le modèle sur sa capacité à prédire le bruit $\epsilon_t \sim \mathcal{N}(0, 1)$.

Définition 4.10. Le terme $\frac{(1-\alpha_t)^2}{2\sigma_t^2 \alpha_t (1-\tilde{\alpha}_t)}$ de la loss est enlevé dans le papier DDPM car améliorant les performances empiriquement, ce qui nous permet de définir la fonction de coût finale :

$$L_{simple}(\theta) := \mathbb{E}_{t, x_0, \epsilon_t}[\left\| \epsilon_t - \epsilon_\theta(\sqrt{\tilde{\alpha}_t}x_0 + \sqrt{1-\tilde{\alpha}_t}\epsilon_t, t) \right\|^2] + C$$

où $t \sim \mathcal{U}(1, ..., T)$
$x_0 \sim q(x_0)$
$\epsilon_t \sim \mathcal{N}(0, 1)$ $\forall t$

$\forall t > 1$ cela correspond à L_{t-1} comme défini précedemment mais pour $t = 1$ cela correspond donc à L_0.

Maintenant que la fonction de coût est définie, on peut passer à l'algorithme qui optimise le paramètre θ.

5 Algorithmes

On en déduit l'algorithme 1 d'entrainement du modèle qui consiste à minimiser $L_{simple}(\theta)$ par descente de gradient. Le 2ème est l'algorithme d'échantillonnage expliqué en introduction, permettant de générer des images à partir de bruit gaussien. Il obtenu par la paramétrisation du modèle : $p_\theta(x_{t-1} \mid x_t) := \mathcal{N}(x_{t-1}, \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)), \sigma_t^2 I)$

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** $\mathbf{x}_0$

FIGURE 4 – Algorithmes d'entrainement 1 et d'échantillonage 2 issues du papier DDPM

6 U-net

Le modèle entrainé ϵ_θ est un réseau de neuronne convolutif (CNN) d'architecture appelée U-net. Initialement introduit pour la segmentation d'image médicale, cette architecture de réseau de neurone permet d'améliorer les performances d'un CNN classique en ajoutant des connexions entre les couches de l'encodeur et celles du décodeur.[12, 13]

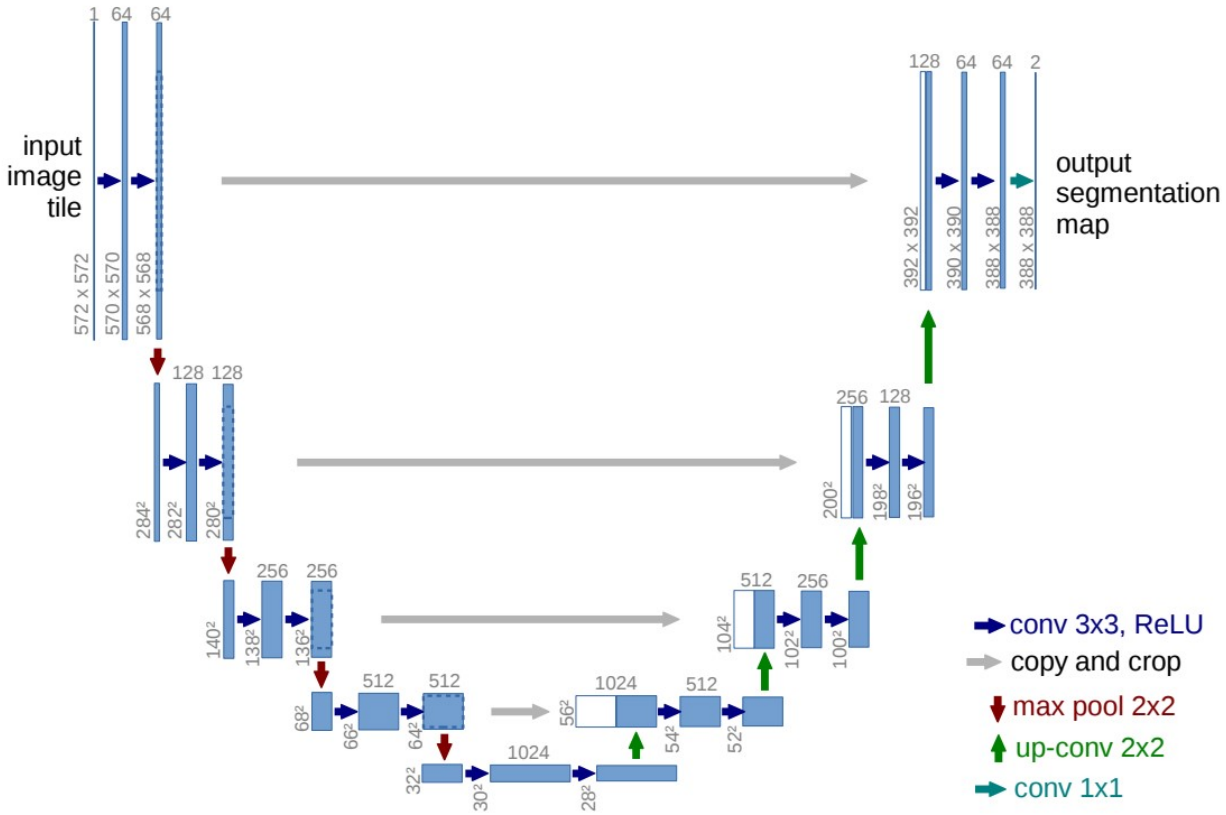

FIGURE 5 – Schéma de la structure du U-net issu du papier de recherche d'origine [12]

Le U-net se décompose en 3 parties :

La première partie est l'encodeur qui prend en entrée une image (dans ce cas l'image bruitée), la passe par un filtre convolutif et une fonction d'activation ReLU pour enfin terminer par un max pooling, ce qui va permettre de réduire la dimension de l'image. Cette opération est répétée plusieurs fois pour réduire la dimension de l'image jusqu'à l'étaler complètement.

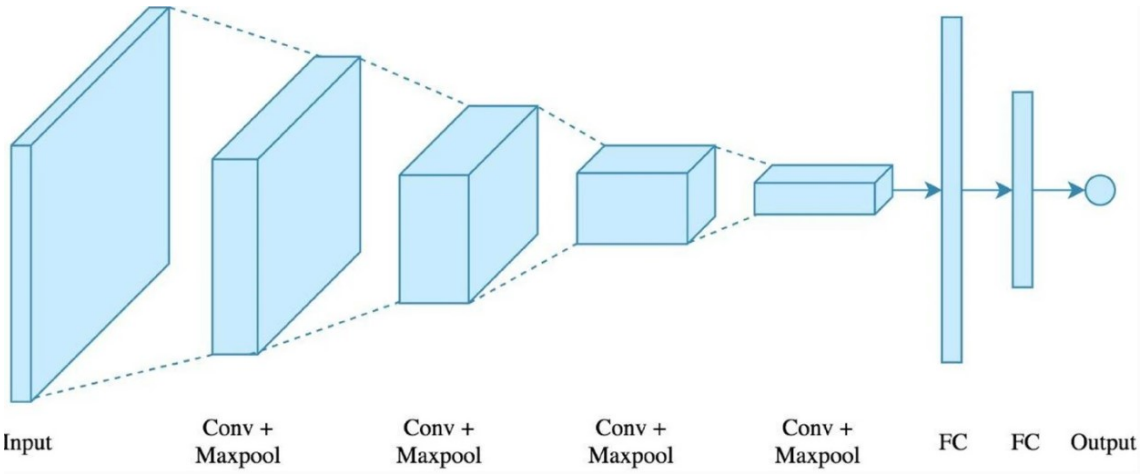

FIGURE 6 – illustration d'un encodeur CNN, Source

La deuxième partie est la partie basse de la structure en forme de U appelée Bottleneck (Goulot d'étranglement) où des filtres de convolution avec fonction d'activation ReLU sont encore appliqués mais sans max pooling cette fois car l'information est déjà compressée.

Enfin la troisième partie est le décodeur qui fait le chemin inverse de l'encodeur en partant du Bottleneck et en augmentant la résolution de l'image jusqu'à retrouver la dimension de l'image d'origine par une opération appelée convolution transposée. Chaque couche reçoit également les caractéristiques de la couche correspondante issue de l'encodeur, ce qui fait la particularité du U-net et permet de transmettre des informations spatiales. Pour finir, il y a une couche de sortie permettant la segmentation de l'image.

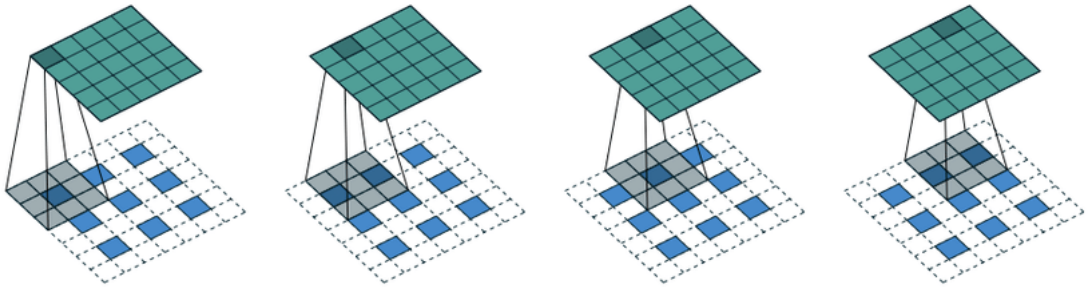

FIGURE 7 – illustration de la convolution transposée CNN, Source

7 Résultats

L'entrainement à été réalisé sur le dataset Landscape Pictures avec une carte graphique RTX 3050 et est détaillé dans la partie suivante.

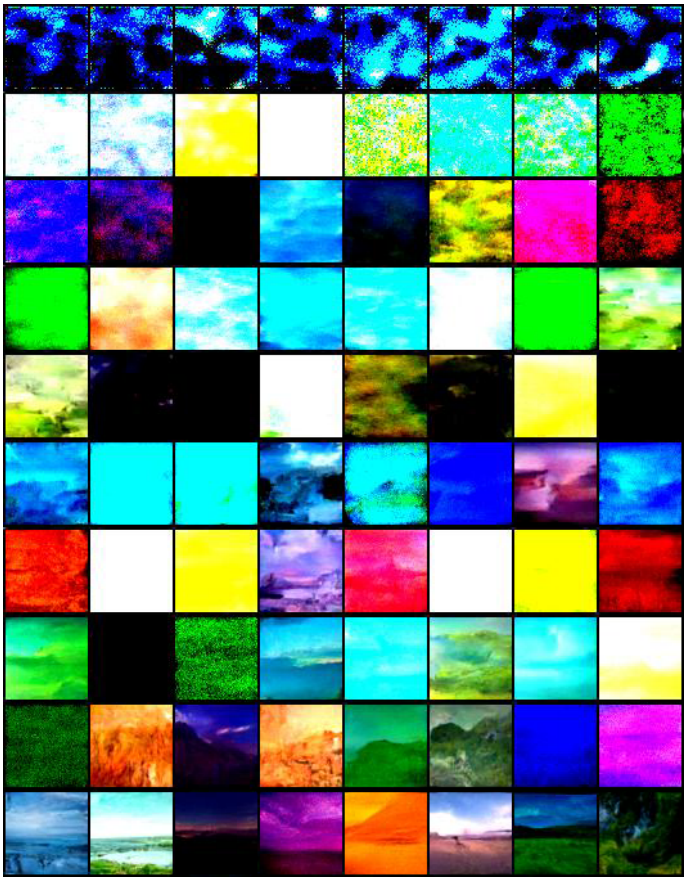

FIGURE 8 – Evolution de la génération d'image au cours de l'entrainement du modèle (de haut en bas)

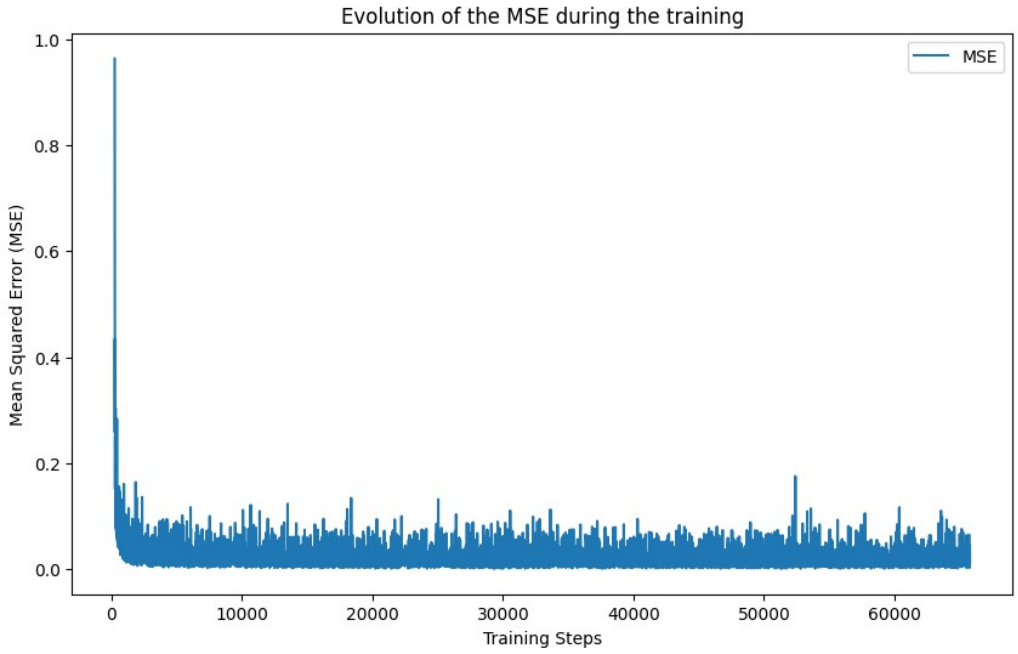

FIGURE 9 – Evolution de la métrique d'évaluation du modèle MSE pendant l'entrainement

8 Axes d'amélioration

Plusieurs hyperparamètres sont ici modulables et peuvent impacter, de façon plus ou moins importante, les performances finales du modèle. La plupart d'entre eux ne sont probablement pas optimaux dans le cas de cet entrainement et doivent donc être remis dans leur contexte :

Choix pratiques :

Dataset : L'entrainement à été réalisé avec environ la moitié du dataset Landscape Pictures (2091 images) contenant des images de paysages en résolution 64x64 afin d'obtenir des résultats corrects avec peu de puissance de calcul. Pour obtenir de meilleures performances, il serait plus judicieux de choisir un dataset plus grand pour entrainer ce réseau de neurone comme le dataset CIFAR10, souvent utilisé pour les comparaisons de performance contenant 60000 images en résolution 32x32.

Optimiseur : L'optimiseur utilisé est l'AdamW, variante de l'optimiseur Adam avec une régularisation L2, une méthode de descente de grandient courante pour entrainer un réseau de neuronne (pseudo code de l'algorithme en annexe). [7, 8] Le learning rate a été fixé à 0.0003. L'utilisation d'un autre optimiseur ou d'un autre learning rate pourrait également conduire à des performances différentes.

Batch : L'entrainement a été fait par batch de 8 images. Cela est dû à la contrainte de VRAM sur le GPU, des batch de 16 ou 32 images auraient permis un entrainement plus rapide et une fonction de coût plus stable.

Epoch : Le nombre d'epoch pour l'entrainement a été fixé à 250. Au vu des résultats, étant donné que le modèle génère encore beaucoup d'images unicolores, il semble qu'il soit plus proche du sous-apprentissage que du sur-apprentissage. Il faudrait donc augmenter le nombre d'epoch d'entrainement pour améliorer les performances.

Choix de méthode :

Evaluation du modèle : L'évaluation du modèle a seulement été réalisé avec la métrique MSE pour comparer l'erreur entre le bruit gaussien et celui prédit par le modèle. Comme dit précedemment, le modèle n'est probablement pas sur-entrainé mais l'usage d'un test set aurait permis de s'en assurer pour quelques calculs supplémentaires. Une autre métrique couramment utilisée est le FID (Fréchet Inception Distance) qui permet de mesurer la qualité d'une image d'une façon qui correspond mieux à la perception humaine qu'une simple MSE. [9, 10]

Paramétrisation : Les performances peuvent être améliorées en paramétrant la variance autrement que linéairement ou en la rendant apprenable par le modèle. [11]

Architecture : L'architecture de réseau de neurone utilisée ici est le U-net comme dans la méthode d'origine. [1] L'utilisation d'une architecture différente conduirait à d'autres résultats. A noter que le U-net permet en théorie d'obtenir de bons résultats même avec assez peu de données, ce qui est utile dans ce cas pratique.

Méthode : Enfin, d'autres méthodes existent aujourd'hui comme l'approche "Score Based" qui pourrait permettre l'amélioration des résultats. [4, 5, 6]

Références

[1] Jonathan Ho, Ajay Jain, Pieter Abbeel, Denoising Diffusion Probabilistic Models, 2020

[2] Alex Nichol, Prafulla Dhariwal, Improved denoising diffusion probabilistic models, 2021

[3] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, Surya Ganguli, Deep Unsupervised Learning using Nonequilibrium Thermodynamics, 2015

[4] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, Ming-Hsuan Yang, Diffusion Models: A Comprehensive Survey of Methods and Applications, 2022

[5] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, Ben Poole, Score-Based Generative Modeling through Stochastic Differential Equations, 2021

[6] Stanislas Strasman, Antonio Ocello, Claire Boyer, Sylvain Le Corff, Vincent Lemaire, An analysis of the noise schedule for score-based generative models, 2024

[7] Ilya Loshchilov, Frank Hutter, Decoupled Weight Decay Regularization, 2017

[8] Diederik P. Kingma, Jimmy Ba, Adam: A Method for Stochastic Optimization, 2014

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter, GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, 2017

[10] Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, Sanjiv Kumar, Rethinking FID: Towards a Better Evaluation Metric for Image Generation, 2023

[11] Prafulla Dhariwal, Alex Nichol, Diffusion Models Beat GANs on Image Synthesis, 2021

[12] Olaf Ronneberger, Philipp Fischer, Thomas Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015

[13] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R. Zaiane, Martin Jagersand, U2-Net: Going Deeper with Nested U-Structure for Salient Object Detection, 2020

[14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative Adversarial Networks, 2014

[15] Diederik P Kingma, Max Welling, Auto-Encoding Variational Bayes, 2013

9 Simplifications

Définition 8.1. Soit $\alpha_t = 1 - \beta_t$ et $\tilde{\alpha_t} = \prod_{s=1}^{t} \alpha_s$

Remarque 8.2. On a que $q(x_t \mid x_{t-1}) := \mathcal{N}(x_t, \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$, ce qui revient à écrire que :

$$q(x_t \mid x_{t-1}) = \sqrt{1-\beta_t}x_{t-1} + \beta_t\epsilon \qquad \text{où } \epsilon \sim \mathcal{N}(0,1)$$

Proposition 8.4. A partir de la remarque 8.2. on obtient que :

$$q(x_t \mid x_0) = \sqrt{\tilde{\alpha_t}}x_0 + \sqrt{1-\tilde{\alpha_t}}\epsilon \qquad \text{où } \epsilon \sim \mathcal{N}(0,1)$$

Preuve : Par définition, on a :

$$q(x_t \mid x_{t-1}) = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon \qquad \text{où } \epsilon \sim \mathcal{N}(0,1)$$

et aussi

$$q(x_{t-1} \mid x_{t-2}) = \sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon' \qquad \text{où } \epsilon' \sim \mathcal{N}(0,1)$$

ce qui nous donne donc

$$\begin{aligned} q(x_t \mid x_{t-2}) &= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon') + \sqrt{1-\alpha_t}\epsilon \quad (1)\\ &= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\epsilon' + \sqrt{1-\alpha_t}\epsilon \quad (2)\end{aligned}$$

Or ϵ et ϵ' suivant des lois normales indépendantes de moyennes nulles, on obtient que :

$$\sqrt{\alpha_t(1-\alpha_{t-1})}\epsilon' + \sqrt{1-\alpha_t}\epsilon \sim \mathcal{N}(0, \sigma^2)$$

avec $\sigma^2 = \alpha_t(1-\alpha_{t-1}) + 1 - \alpha_t = 1 - \alpha_t\alpha_{t-1}$

Ce qui nous donne finalement :

$$q(x_t \mid x_{t-2}) = \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon \qquad \text{où } \epsilon \sim \mathcal{N}(0,1)$$

En itérant de la même manière jusqu'à x_0, on obtient bien que :

$$q(x_t \mid x_0) = \sqrt{\tilde{\alpha_t}}x_0 + \sqrt{1-\tilde{\alpha_t}}\epsilon \qquad \text{où } \epsilon \sim \mathcal{N}(0,1)$$

Remarque 8.5. trouver x_T directement à partir de x_0 permet de gagner du temps et de passer de T à 1 étape pour obtenir l'image bruitée.

Proposition 8.6. La loi du dernier échantillon bruité est donnée par

$$q(x_T \mid x_0) = \sqrt{\tilde{\alpha_T}}x_0 + \sqrt{1-\tilde{\alpha_T}}\epsilon \cong \mathcal{N}(0,1)$$

Pour cela, on doit s'assurer que $\sqrt{\tilde{\alpha_t}}$ décroit vers 0 et donc que $\sqrt{1-\tilde{\alpha_t}}$ croit vers 1 quand t augmente car ces valeurs sont fixées au départ. (Figure 3)

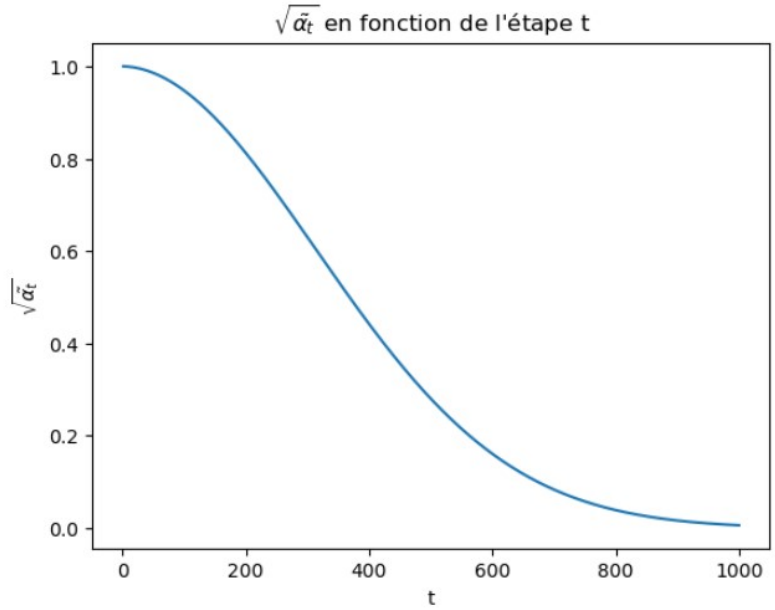

FIGURE 10 – Décroissance de $\sqrt{\tilde{\alpha}_t}$ en fonction de t avec T=1000

10 Développement de la fonction de coût

Proposition 9.1. Pour tout θ,

$$\begin{aligned}\mathbb{E}[-\log p_\theta(x_0)] &\leq \mathbb{E}_q[-\log \frac{p_\theta(x_0,...,x_T)}{q(x_1,...,x_T \mid x_0)}] \\ &= \mathbb{E}_q[-\log p(x_T) - \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})}] =: L_{VLB}(\theta)\end{aligned}$$

Preuve : On rappelle la loi marginale $p_\theta(x_0) = \int p_\theta(x_0,...,x_T)dx_1,..,x_T$

On a donc

$$-\log p_\theta(x_0) = -\log \int p_\theta(x_0,...,x_T)dx_1,..,x_T \tag{3}$$

$$= -\log \int \frac{p_\theta(x_0,...,x_T)}{q(x_1,...,x_T \mid x_0)} q(x_1,...,x_T \mid x_0)dx_1,..,x_T \tag{4}$$

$$\leq \mathbb{E}_{q(x_1,...,x_T|x_0)}[-\log \frac{p_\theta(x_0,...,x_T)}{q(x_1,...,x_T \mid x_0)}] \text{ par l'inégalité de Jensen} \tag{5}$$

Ce qui donne :

$$\mathbb{E}_{q(x_0)}[-\log p_\theta(x_0)] \leq \mathbb{E}_{q(x_0,...,x_T)}[-\log \frac{p_\theta(x_0,...,x_T)}{q(x_1,...,x_T \mid x_0)}]$$

Enfin en utilisant $p_\theta(x_0,...,x_T) := p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)$ au numérateur et $q(x_1,...,x_T \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$ au dénominateur, on a bien :

$$\mathbb{E}_q[-\log \frac{p_\theta(x_0,...,x_T)}{q(x_1,...,x_T \mid x_0)}] = \mathbb{E}_q[-\log p(x_T) - \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})}]$$

Proposition 9.2. Pout tout θ,

$$L_{VLB}(\theta) = \mathbb{E}_q[\underbrace{D_{KL}(q(x_T \mid x_0) \mid\mid p(x_T))}_{L_T} + \sum_{t=2}^{T} \underbrace{D_{KL}(q(x_{t-1} \mid x_t, x_0) \mid\mid p_\theta(x_{t-1} \mid x_t))}_{L_{t-1}} - \underbrace{logp_\theta(x_0 \mid x_1)}_{L_0}]$$

Preuve :

$$L_{VLB}(\theta) = \mathbb{E}_q[-\log p(x_T) - \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})}] \tag{6}$$

$$= \mathbb{E}_q[-\log p(x_T) - \sum_{t=2}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})} - \log \frac{p_\theta(x_0 \mid x_1)}{q(x_1 \mid x_0)}] \tag{7}$$

$$= \mathbb{E}_q[-\log p(x_T) + \sum_{t=2}^{T} \log \frac{q(x_t \mid x_{t-1})}{p_\theta(x_{t-1} \mid x_t)} + \log \frac{q(x_1 \mid x_0)}{p_\theta(x_0 \mid x_1)}] \tag{8}$$

$$= \mathbb{E}_q[-\log p(x_T) + \sum_{t=2}^{T} \log \frac{q(x_{t-1} \mid x_t)q(x_t)}{p_\theta(x_{t-1} \mid x_t)q(x_{t-1})} + \log \frac{q(x_1 \mid x_0)}{p_\theta(x_0 \mid x_1)}] \text{ par la formule de bayes} \tag{9}$$

$$= \mathbb{E}_q[-\log p(x_T) + \sum_{t=2}^{T} \log \frac{q(x_{t-1} \mid x_t, x_0)q(x_t \mid x_0)}{p_\theta(x_{t-1} \mid x_t)q(x_{t-1} \mid x_0)} + \log \frac{q(x_1 \mid x_0)}{p_\theta(x_0 \mid x_1)}] \tag{10}$$

$$= \mathbb{E}_q[-\log p(x_T) + \sum_{t=2}^{T} \log \frac{q(x_{t-1} \mid x_t, x_0)}{p_\theta(x_{t-1} \mid x_t)} + \sum_{t=2}^{T} \log \frac{q(x_t \mid x_0)}{q(x_{t-1} \mid x_0)} + \log \frac{q(x_1 \mid x_0)}{p_\theta(x_0 \mid x_1)}] \tag{11}$$

$$= \mathbb{E}_q[-\log p(x_T) + \sum_{t=2}^{T} \log \frac{q(x_{t-1} \mid x_t, x_0)}{p_\theta(x_{t-1} \mid x_t)} + \log \frac{q(x_T \mid x_0)}{q(x_1 \mid x_0)} + \log \frac{q(x_1 \mid x_0)}{p_\theta(x_0 \mid x_1)}] \text{ par télescopage} \tag{12}$$

$$= \mathbb{E}_q[\log \frac{q(x_T \mid x_0)}{p(x_T)} + \sum_{t=2}^{T} \log \frac{q(x_{t-1} \mid x_t, x_0)}{p_\theta(x_{t-1} \mid x_t)} - \log p_\theta(x_0 \mid x_1)] \text{ en réarrangeant les termes} \tag{13}$$

$$= \mathbb{E}_q[D_{KL}(q(x_T \mid x_0) \mid\mid p(x_T)) + \sum_{t=2}^{T} D_{KL}(q(x_{t-1} \mid x_t, x_0) \mid\mid p_\theta(x_{t-1} \mid x_t)) - log p_\theta(x_0 \mid x_1)] \tag{14}$$

Proposition 9.3. Pour tout $1 \leq t \leq T$,

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

avec $\tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\tilde{\alpha}_{t-1}}\beta_t}{1-\tilde{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\tilde{\alpha}_{t-1})}{1-\tilde{\alpha}_t} x_t$ et $\tilde{\beta}_t := \frac{1-\tilde{\alpha}_{t-1}}{1-\tilde{\alpha}_t}\beta_t$

Proposition 9.4. Pour tout $1 \leq t \leq T$,

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\tilde{\alpha}_t}}\epsilon_t)$$

Preuve : On utilise que $x_t = \sqrt{\tilde{\alpha}_t}x_0 + \sqrt{1-\tilde{\alpha}_t}\epsilon_t$ et donc $x_0 = \frac{1}{\sqrt{\tilde{\alpha}_t}}(x_t - \sqrt{1-\tilde{\alpha}_t}\epsilon_t)$
En remplaçant x_0 dans $\tilde{\mu}_t$, on obtient finalement :

$$\tilde{\mu}_t = \frac{\sqrt{\tilde{\alpha}_{t-1}}\beta_t}{1-\tilde{\alpha}_t}\frac{1}{\sqrt{\tilde{\alpha}_t}}(x_t - \sqrt{1-\tilde{\alpha}_t}\epsilon_t) + \frac{\sqrt{\alpha_t}(1-\tilde{\alpha}_{t-1})}{1-\tilde{\alpha}_t}x_t \tag{15}$$

$$= (\frac{\beta_t}{1-\tilde{\alpha}_t}\frac{1}{\sqrt{\alpha_t}} + \frac{\sqrt{\alpha_t}}{1-\tilde{\alpha}_t} - \frac{\sqrt{\alpha_t}\tilde{\alpha}_{t-1}}{1-\tilde{\alpha}_t})x_t - \frac{\beta_t}{\sqrt{1-\tilde{\alpha}_t}}\frac{1}{\sqrt{\alpha_t}}\epsilon_t \tag{16}$$

$$= \frac{1}{\sqrt{\alpha_t}}(\frac{\beta_t}{1-\tilde{\alpha}_t} + \frac{\alpha_t}{1-\tilde{\alpha}_t} - \frac{\alpha_t\tilde{\alpha}_{t-1}}{1-\tilde{\alpha}_t})x_t - \frac{\beta_t}{\sqrt{1-\tilde{\alpha}_t}}\frac{1}{\sqrt{\alpha_t}}\epsilon_t \tag{17}$$

$$= \frac{1}{\sqrt{\alpha_t}}(\frac{1-\alpha_t}{1-\tilde{\alpha}_t} + \frac{\alpha_t}{1-\tilde{\alpha}_t} - \frac{\tilde{\alpha}_t}{1-\tilde{\alpha}_t})x_t - \frac{1-\alpha_t}{\sqrt{1-\tilde{\alpha}_t}}\frac{1}{\sqrt{\alpha_t}}\epsilon_t \tag{18}$$

$$= \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\tilde{\alpha}_t}}\epsilon_t) \tag{19}$$

Proposition 9.5. pour $1 \leq t \leq T-1$ on a

$$L_t = \mathbb{E}_q[\frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\|^2] + C$$

avec C ne dépendant pas de θ.
On l'obtient à partir de la divergence de Kullback entre les deux lois normales.

Proposition 9.6. pour $1 \leq t \leq T-1$ on peut simplifier L_t :

$$L_t = \mathbb{E}_{x_0,\epsilon_t}[\frac{(1-\alpha_t)^2}{2\sigma_t^2\alpha_t(1-\tilde{\alpha}_t)} \left\|\epsilon_t - \epsilon_\theta(\sqrt{\tilde{\alpha}_t}x_0 + \sqrt{1-\tilde{\alpha}_t}\epsilon_t, t)\right\|^2] + C$$

Preuve :

$$L_t = \mathbb{E}_q[\frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\|^2] + C \tag{20}$$

$$= \mathbb{E}_{x_0,\epsilon_t}[\frac{1}{2\sigma_t^2} \left\| \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\tilde{\alpha}_t}}\epsilon_t) - \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\tilde{\alpha}_t}}\epsilon_\theta(x_t, t)) \right\|^2] + C \tag{21}$$

où x_t ne dépend que de x_0 et ϵ_t (22)

$$= \mathbb{E}_{x_0,\epsilon_t}[\frac{(1-\alpha_t)^2}{2\sigma_t^2\alpha_t(1-\tilde{\alpha}_t)} \left\|\epsilon_t - \epsilon_\theta(\sqrt{\tilde{\alpha}_t}x_0 + \sqrt{1-\tilde{\alpha}_t}\epsilon_t, t)\right\|^2] + C \tag{23}$$

11 Algorithme AdamW

input : γ(lr), β_1, β_2(betas), θ_0(params), $f(\theta)$(objective), ϵ (epsilon)
λ(weight decay), *amsgrad*, *maximize*
initialize : $m_0 \leftarrow 0$ (first moment), $v_0 \leftarrow 0$ (second moment), $\widehat{v_0}^{max} \leftarrow 0$

---

for $t = 1$ **to** ... **do**
  if *maximize* :
    $g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1})$
  else
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  $\theta_t \leftarrow \theta_{t-1} - \gamma\lambda\theta_{t-1}$
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1)g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2)g_t^2$
  $\widehat{m_t} \leftarrow m_t / (1-\beta_1^t)$
  $\widehat{v_t} \leftarrow v_t / (1-\beta_2^t)$
  if *amsgrad*
    $\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$
    $\theta_t \leftarrow \theta_t - \gamma\widehat{m_t} / (\sqrt{\widehat{v_t}^{max}} + \epsilon)$
  else
    $\theta_t \leftarrow \theta_t - \gamma\widehat{m_t} / (\sqrt{\widehat{v_t}} + \epsilon)$

---

return θ_t

FIGURE 11 – Pseudo code de l'algorithme AdamW provenant de la documentation Pytorch