FACULTÉ DES SCIENCES
UNIVERSITÉ D'ANGERS

Université d'Angers

Master Mathématiques et Applications

M2 DATA SCIENCE

Année académique 2024-2025

Travail Encadré de Recherche

Modèles génératifs de diffusion

Étudiants :
Thibaut Gipteau
Arnaud Gonin
Mahamat Mahamat Nour Bachar

Tuteur enseignant :
Fabien Panloup

Engagement de non plagiat

Nous, soussignés, déclarons être pleinement conscients que le plagiat de documents ou d'une partie d'un document publiés sur toutes formes de support, y compris internet, constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée. En conséquence, nous nous engageons à citer toutes les sources que nous avons utilisées pour écrire ce rapport.

Résumé

Ces dernières années, les modèles génératifs ont connu un essor considérable dans le domaine de l'apprentissage automatique, en particulier pour générer des données synthétiques reproduisant fidèlement aux données d'apprentissage. Si les approches traditionnelles, telles que les GAN et les auto-encodeurs variationnels, ont contribué à cet objectif, les méthodes basées sur la diffusion, comme le Denoising Diffusion Probabilistic Model (DDPM), se révèlent prometteuses grâce à leur capacité à modéliser le bruit et à inverser le temps.

Dans ce travail, nous nous intéressons principalement aux modèles de diffusion, en étudiant leur implémentation pratique et leurs fondements théoriques, notamment la notion d'inversion du temps et leur lien avec des processus stochastiques tels que l'Ornstein-Uhlenbeck. Par ailleurs, nous explorons l'évaluation de la qualité des images générées via des métriques comme la Fréchet Inception Distance (FID). Des exemples appliqués sur des jeux de données standards, notamment MNIST, illustrent les performances et les limites de ces approches. L'analyse comparative avec d'autres techniques génératives met en lumière les atouts des modèles de diffusion dans le cadre de la génération et du débruitage d'images. Enfin, nous présentons une alternative, le Sliced Score Matching, en analysant ses avantages et inconvénients.

Mots-clés : modèles génératifs, diffusion, DDPM, inversion du temps, processus Ornstein-Uhlenbeck, Fréchet Inception Distance, Denoising Score Matching, Sliced Score Matching, MNIST.

Remerciements

Nous tenons à exprimer notre profonde gratitude à notre encadrant, **Fabien Panloup**, pour ses conseils tout au long de ce travail.

Nous remercions également Mohamed Alfaki Ag Aboubacrine Assadeck(PhD), dont les indications sur des points connexes à ses travaux nous ont aidés à mieux comprendre certains aspects de notre sujet.

Nous remercions aussi OpenAI, dont le modèle `GPT-4o` nous a permis de trouver de nombreux tutoriels et références enrichissants pour nos recherches.

Enfin, nous adressons notre reconnaissance à toutes celles et ceux qui nous ont aidés, d'une manière ou d'une autre, à mener à bien ce projet.

Merci !

« Les mathématiques ne sont pas une science dure, c'est juste une science qui demande du temps. »

Stefan Banach

Table des matières

Motivation

Les modèles génératifs ont connu un essor considérable ces dernières années. Parmi les approches les plus connues figurent les **Generative Adversarial Networks (GAN)** et les **Variational Autoencoders (VAE)**.

Les GAN, introduits par [5], reposent sur une dynamique compétitive entre un générateur et un discriminateur. Malgré leur capacité à produire des images de qualité, ils ont certains défauts importants, comme des entraînements parfois instables et un problème appelé *mode collapse*. Cela signifie que le modèle peut se focaliser sur quelques échantillons et ignorer le reste des données, ce qui limite sa capacité à bien représenter l'ensemble des données.

Les VAE, popularisés par [10], reposent sur une approche probabiliste à travers un schéma encodeur-décodeur. Cependant, utiliser une divergence Kullback-Leibler dans l'espace latent peut rendre les résultats plus flous. Cela s'explique par le fait que la contrainte imposée empêche le modèle de capturer tous les détails et le pousse à produire des images trop « générales ».

En réponse à ces limitations, des approches basées sur le **score matching** et le renversement temporel des processus de diffusion ont récemment émergé comme des alternatives prometteuses. Dans [17], l'apprentissage du score, c'est-à-dire du gradient du logarithme de la densité $\nabla \log p(.,.)$, permet de reconstruire de manière efficace la distribution cible via une équation différentielle stochastique. Cette méthode présente l'avantage de produire des échantillons plus diversifiés que les précédents et fidèles à l'ensemble des données.

Chapitre 1

Principe général

Nous avons déjà mentionné que le principe général des modèles génératifs est d'apprendre à produire, à partir de données initiales, de nouvelles données similaires. Les modèles de diffusion sont un cas particulier de ce principe : en définissant un processus de "bruitage pas-à-pas" des données initiales dont on connaît la nature "inversible" ainsi que la "limite", on peut, à partir de cette distribution limite simulable, "débruiter pas-à-pas" jusqu'à obtenir une nouvelle donnée vraisemblable dans l'espace des données initiales. Dans la plus grande partie de cet article, on considérera le cadre facilitant de la génération d'images, et particulièrement du dataset MNIST. Commençons par définir les processus de diffusion, et leur utilité dans notre cadre pratique.

1.1 Processus de diffusion

Un processus de diffusion est un processus stochastique, défini sous sa forme générale *forward* par [15] :

$$\begin{cases} dX_t := f(X_t, t)dt + g(t)dW_t \\ X_0 \sim p_0 \end{cases}$$

où dW_t est une marche aléatoire de loi normale centrée et réduite (mouvement brownien, que nous détaillerons dans 2.2.1). On notera toujours dans la suite p_t pour désigner la loi de X_t à un instant t du processus donné.

Un tel processus doit être compris, dans notre cas "MNIST", comme le modèle théorique (*i.e.* continu) d'un bruitage des images de départ, où p_0 désigne la distribution des images de départ pixel par pixel. Une version discrétisée (par le schéma d'Euler-Maruyama, nous y reviendrons sur cette méthode dans 2.4) est alors donnée par :

$$\begin{cases} X_{k+1} = X_k + \Delta_t f(X_k, k\Delta_t) + g(k\Delta_t)\sqrt{k\Delta_t}Z_k \\ Z_k \sim \mathcal{N}(0,1) \\ X_0 \sim p_0 \end{cases}$$

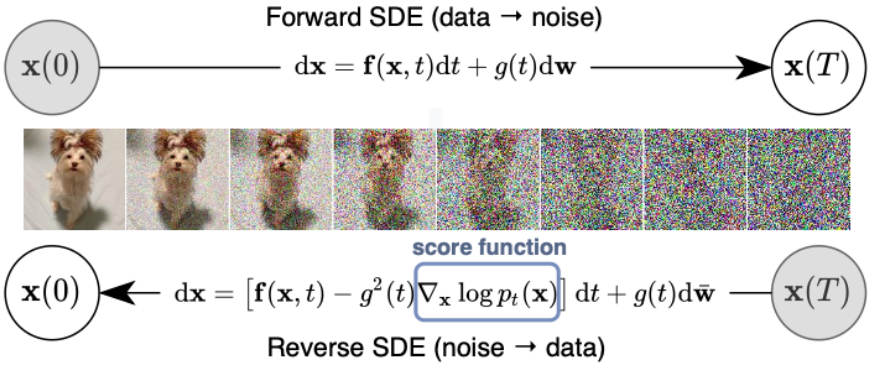
avec Δ_t suffisamment petit.

D'après le théorème d'Anderson [1], on sait qu'un tel processus est inversible, au sens où on a le processus de diffusion à temps inversé *reverse* suivant :

$$\begin{cases} d\hat{X}_t = [f(\hat{X}_t, t) - g(t)^2 \nabla_x \log p_t(\hat{X}_t)]d_t + g(t)d\hat{W}_t \\ \hat{X}_t := X_{T-t} \\ X_T \sim p_T \end{cases}$$

Ici, $\hat{X}$ correspond à $\overline{X}$, défini dans 2.3.
Par un schéma discrétisé similaire à celui du processus *forward*, on devrait alors pouvoir, à partir d'une image bruitée (X_T), générer une image restaurée (X_0) correspondant à l'état initial, *i.e.* avant bruitage par le processus de diffusion *forward*. En admettant qu'on prenne un T suffisamment grand, on peut supposer que X_T sera entièrement constitué de bruit, et donc potentiellement facilement simulable.

FIGURE 1.1 – Le modèle forward-reverse sur une image [17]



Deux problèmes apparaissent pour le moment :
— Bien que simuler le processus *forward* soit facile à partir d'un échantillon de p_0, ce n'est pas le cas pour le processus *reverse* : la loi p_t n'est pas connue à priori, et à plus forte raison le terme $\nabla_x \log p_t(\hat{X}_t)$
— La simulation de X_T requiert la connaissance immédiate de la loi p_T, ce qui doit motiver le choix du processus utilisé.
Ces deux problèmes seront résolus de manières différentes, que nous verrons par la suite. On se concentrera d'abord sur l'idée générale permettant l'approximation du terme $\nabla_x \log p_t(\hat{X}_t)$.

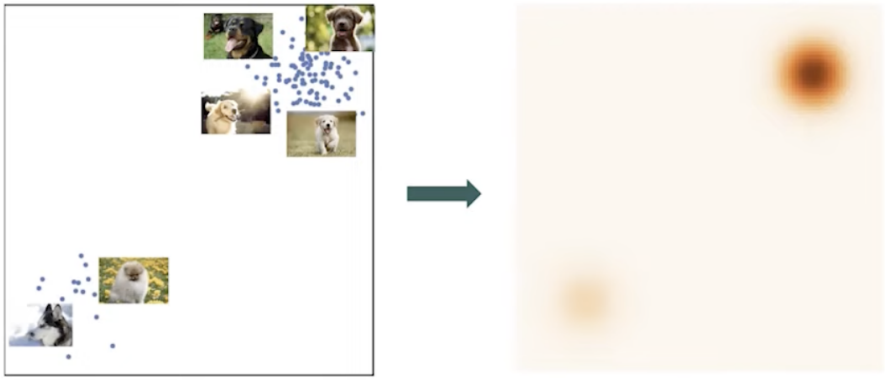
1.2 Approximation et fonction de score

Une méthode adaptée pour déterminer le terme $\nabla_x \log p_t(\hat{X}_t)$ est de construire un réseau de neurones destiné à l'approximer. Considérons dans un premier temps l'interprétation de ce terme. Nous nous dégageons de la notion "itérative" des modèles de diffusion dans cette section, et y reviendrons par la suite.

Nous cherchons à estimer une distribution de probabilité, à une étape donnée du processus. Dans ce cadre, la probabilité recherchée sera notée $p(x)$. Lorsqu'une telle distribution est estimée par un réseau de neurones, c'est usuellement au prix de transformations. Si l'on note $f(x)$ la sortie du réseau, on rend positive cette sortie en passant à l'exponentielle, puis on normalise le résultat pour obtenir une distribution. On a donc :

$$\xrightarrow{réseau} f(x) \xrightarrow{positiv.} \exp f(x) \xrightarrow{normal.} \frac{\exp f(x)}{Z} \approx p(x)$$

FIGURE 1.2 – Distribution des données intiales [16]



$$Z = \int e^{f(x)} dx$$

Dans les faits, la constante Z est particulièrement compliquée à calculer (problème P-complet même dans des cas discrets). Les approches précédant les modèles à diffusion contournaient ce problème soit en approximant la constante (Energy Based Models), soit en restreignant le modèle utilisé pour rendre cette constante calculable (modèles autorégressifs, VAE).

Dans le cadre des modèles de diffusion, cette constante est contournée par l'utilisation du *score de Stein* $\nabla_x \log p(x)$:

$$\nabla_x f(x) = \nabla_x \log p(x) + \nabla_x Z = \nabla_x \log p(x)$$

Le terme dépendant de Z disparaît, puisque Z est constante, cependant que par des calculs d'intégrales, la fonction $\nabla_x \log p(x)$ toute l'information sur la distribution de la loi. En effet, on a

$$\log p(x) = \log p(x_0) + \int_{x_0}^{x} \nabla_t \log p(t) dt \tag{1.1}$$

$$p(x) = p(x_0) \exp \int_{x_0}^{x} \nabla_t \log p(t) dt \tag{1.2}$$

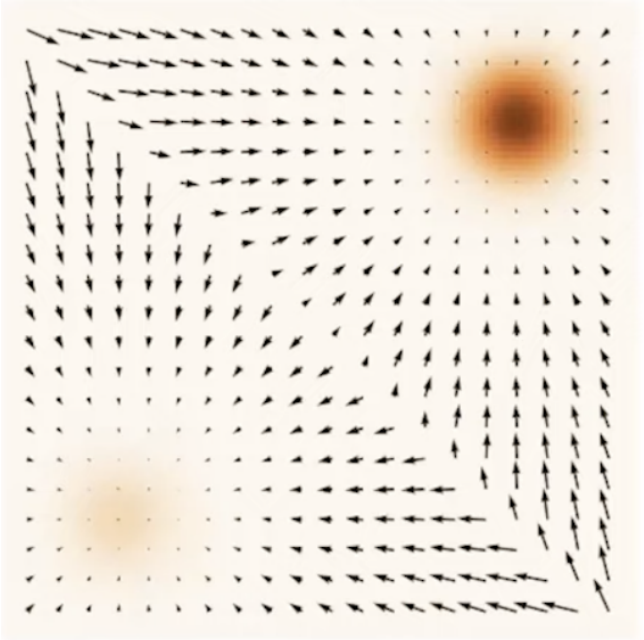
où $p(x_0)$ est à nouveau une constante de normalisation, mais qu'il ne sera cette fois pas nécessaire de calculer (le processus de génération à venir utilisant directement le score). L'équivalence en information est visuellement expliquée par la figure 1.3, où un champ de vecteurs vient remplacer l'information sur la distribution vue dans l'exemple précédent.

La disparition du besoin de normalisation permet une estimation efficace par réseau de neurones. Notre réseau sera ainsi chargé d'estimer un *champs de vecteurs*, le plus proche possible du score de Stein. Cette proximité se traduit par la divergence de Fisher, définie par :

$$\frac{1}{2} \mathbf{E}_p[\|s_\theta(x) - \nabla_x \log p(x)\|^2]$$

Maintenant que nous avons à notre disposition une fonction désignant la distance entre notre réseau et le terme problématique, deux chemins s'offrent à nous. Le premier, nommé "Denoising Score Matching", est l'objet du chapitre suivant. Une autre section s'intéressera à l'approche "Sliced Score Matching".

FIGURE 1.3 – Distribution et score de Stein [16]



Chapitre 2

Processus de diffusion inverse

2.1 Introduction

Dans le cadre des modèles de diffusion, il est crucial de comprendre comment, par inversion temporelle, on peut obtenir une dynamique inverse permettant de générer des échantillons à partir d'une distribution de données. Nous considérons ici une chaîne de Markov X_t définie pour $t \in [0, T]$, $T > 0$ et nous démontrons que le processus inversé

$$\overline{X}_t = X_{T-t}$$

satisfait une équation différentielle stochastique(SDE) dont la fonction drift comporte un terme "correctif" $\nabla \log p(x, t)$, le **score** déterminé par le réseau.

2.2 Processus Forward et son équation de Fokker-Planck

2.2.1 Définition du processus *Forward*

Soit $(X_t)_{t\in[0,T]}$ où $T > 0$ un processus stochastique satisfaisant une équation différentielle stochastique, aussi appelée processus de diffusion. L'EDS est de la forme suivante :

$$dX_t = b(X_t, t)\, dt + \sigma(t)\, dW_t, \quad X_0 \sim p_0, \tag{2.1}$$

avec :

— $b : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ une fonction lipschitzienne dite aussi fonction de *drift* et $\mathcal{C}^{2,1}(\mathbb{R}^d \times [0, T], \mathbb{R}^d)$,

— $\sigma : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d \times \mathbb{R}_+$ matrice de diffusion, symétrique($\sigma(t) = \sigma(T - t)$) et $\sigma(t) > 0$ pour tout t, $\mathcal{C}^{2,1}$ et indépendant de x c'est-à-dire $\sigma(x, t) = \sigma(t)$ pour tout $x \in \mathbb{R}^d$.

Définition 2.2.1 (Mouvement brownien)**.** *Un mouvement brownien est un processus stochastique $(W_t)_{t\geq 0}$ défini sur un espace de probabilité, vérifiant :*

1. *Les accroissements $W_{t+s} - W_t$ sont indépendants, pour tout $s, t \geq 0$.*
2. *Pour tout $t, s \geq 0$, $W_{t+s} - W_t$ suit une loi normale de moyenne 0 et de variance s (c'est-à-dire $\mathcal{N}(0, s)$).*
3. *Les trajectoires de W_t sont continues presque sûrement, c'est-à-dire pour presque toute réalisation ω, la fonction $t \mapsto W_t(\omega)$ est continue.*
4. *Il est souvent supposé que $W_0 = 0$. On dit alors que le mouvement brownien est standard[9].*

Définition 2.2.2 (Fonction test)**.** *Comme la définition de fonction test n'est pas universelle, on appelle ici une fonction test une fonction $\varphi : \mathbb{R}^d \to \mathbb{R}$ appartenant à l'espace $\mathcal{C}^\infty(\mathbb{R}^d)$, c'est-à-dire :*

1. *φ est infiniment différentiable sur $\mathbb{R}^d$,*
2. *φ possède un support compact, c'est-à-dire qu'il existe un ensemble compact $K \subset \mathbb{R}^d$ tel que $\varphi(x) = 0$ pour tout $x \notin K$.*

Cette propriété de support compact garantit que, lors de l'intégration sur $\mathbb{R}^d$, les termes de bord disparaissent, ce qui est crucial lorsqu'on effectue des intégrations par parties. Cette annulation des termes du bord est importante dans l'étude de l'inversion du temps.

Remarque 1. *Les fonctions tests sont considérées comme des fonctions régulières car elles sont infiniment différentiables et ont un support compact. C'est ce qui explique l'usage du terme* ***régulières*** *dans la littérature des modèles de diffusion.*

Dans la suite de ce chapitre, nous noterons $\nabla = \nabla_X$ pour le gradient par rapport à X, $\nabla \cdot f = \mathrm{div}(f)$ pour la divergence, Δ le Laplacien et $u \cdot v = \langle u, v \rangle$ pour le produit vectoriel.

Théorème 1 (Équation de Fokker–Planck)**.** *Soit la densité $p(x,t)$ associée au processus X_t satisfaisant l'équation*

$$\partial_t p(x,t) = -\nabla \cdot \Big(b(x,t)p(x,t)\Big) + \frac{1}{2}\sigma(t)\sigma^T(t)\,\Delta p(x,t), \tag{2.2}$$

Démonstration. Soit $f \in C^\infty(\mathbb{R}^d)$ une fonction test. En appliquant la formule d'Itô à $f(X_t,t)$, nous avons :

$$d\Big(f(X_t,t)\Big) = \frac{\partial f}{\partial t}(X_t,t)\,dt + \nabla f(X_t,t)\cdot dX_t + \frac{1}{2}\,\mathrm{tr}\Big(\sigma(t)\sigma(t)^T\nabla^2 f(X_t,t)\Big)dt.$$

En réinjectant l'expression de dX_t (2.1), on obtient

$$\begin{aligned} df(X_t,t) &= \frac{\partial f}{\partial t}(X_t,t)\,dt + \nabla f(X_t,t)\cdot\Big(b(X_t,t)\,dt + \sigma(t)\,dW_t\Big) \\ &\quad + \frac{1}{2}\,\mathrm{tr}\Big(\sigma(t)\sigma(t)^T\nabla^2 f(X_t,t)\Big)dt \\ &= \frac{\partial f}{\partial t}(X_t,t) + b(X_t,t)\cdot\nabla f(X_t,t)dt + \frac{1}{2}\,\mathrm{tr}\Big(\sigma(t)\sigma(t)^T\nabla^2 f(X_t,t)\Big)dt \\ &\quad + \nabla f(X_t,t)\cdot\sigma(t)\,dW_t. \end{aligned}$$

En prenant l'espérance, le dernier terme s'annule (car $\mathbb{E}[\nabla f(X_t,t)\cdot\sigma(t)\,dW_t] = 0$) et nous obtenons :

$$\mathbb{E}\Big[df(X_t,t)\Big] = \mathbb{E}\Big[\frac{\partial f}{\partial t}(X_t,t) + b(X_t,t)\cdot\nabla f(X_t,t)dt + \frac{1}{2}\,\mathrm{tr}\Big(\sigma(t)\sigma(t)^T\nabla^2 f(X_t,t)\Big)dt\Big],$$

$$\Leftrightarrow \frac{d}{dt}\mathbb{E}\Big[f(X_t,t)\Big] = \mathbb{E}\Big[\frac{\partial f}{\partial t}(X_t,t) + b(X_t,t)\cdot\nabla f(X_t,t) + \frac{1}{2}\,\mathrm{tr}\Big(\sigma(t)\sigma(t)^T\nabla^2 f(X_t,t)\Big)\Big].$$

Puisque $\mathbb{E}\Big[f(X_t,t)\Big] = \int_{\mathbb{R}^d} f(x,t)p(x,t)\,dx$(formule de transfert), on a

$$\frac{d}{dt}\int_{\mathbb{R}^d} f(x,t)p(x,t)\,dx = \int_{\mathbb{R}^d}\Big[\frac{\partial f}{\partial t}(x,t) + \mathcal{L}f(x,t)\Big]p(x,t)\,dx,$$

où $\mathcal{L}$ le générateur infinitésimal est défini par

$$\mathcal{L}f(x,t) = b(x,t)\cdot\nabla f(x,t) + \frac{1}{2}\operatorname{tr}\Big(\sigma(t)\sigma(t)^T\nabla^2 f(x,t)\Big).$$

Sous les hypothèses adéquates, on peut intervertir la dérivation et l'intégration :

$$\int_{\mathbb{R}^d}\frac{d}{dt}\Big(f(x,t)p(x,t)\Big)\,dx = \int_{\mathbb{R}^d}\Big[\frac{\partial f}{\partial t}(x,t) + \mathcal{L}f(x,t)\Big]p(x,t)\,dx,$$

$$\Leftrightarrow \int_{\mathbb{R}^d} f(x,t)\partial_t p(x,t)\,dx = \int_{\mathbb{R}^d}\Big[b(x,t)\cdot\nabla f(x,t) + \frac{1}{2}\operatorname{tr}\Big(\sigma(t)\sigma(t)^T\nabla^2 f(x,t)\Big)\Big]p(x,t)\,dx.$$

Effectuons maintenant une intégration par parties dans le terme contenant le drift, on a :

$$\int_{\mathbb{R}^d}\nabla f(x,t)\cdot b(x,t)p(x,t)\,dx = -\int_{\mathbb{R}^d} f(x,t)\nabla\cdot\Big(b(x,t)p(x,t)\Big)\,dx,$$

et pour le terme diffusif, par une double intégrale par parties, on a :

$$\int_{\mathbb{R}^d}\operatorname{tr}\Big(\sigma(t)\sigma(t)^T\nabla^2 f(x,t)\Big)p(x,t)\,dx = \int_{\mathbb{R}^d} f(x,t)\operatorname{tr}\Big(\sigma(t)\sigma(t)^T\nabla^2 p(x,t)\Big)\,dx.$$

Ainsi, l'identité devient

$$\int_{\mathbb{R}^d} f(x,t)\partial_t p(x,t)\,dx = \int_{\mathbb{R}^d} f(x,t)\left[-\nabla\cdot\Big(b(x,t)p(x,t)\Big) + \frac{1}{2}\operatorname{tr}\Big(\sigma(t)\sigma(t)^T\nabla^2 p(x,t)\Big)\right]dx.$$

Comme cette relation est vraie pour tout $f\in C^\infty(\mathbb{R}^d)$, on en déduit l'égalité

$$\partial_t p(x,t) = -\nabla\cdot\Big(b(x,t)p(x,t)\Big) + \frac{1}{2}\operatorname{tr}\Big(\sigma(t)\sigma(t)^T\nabla^2 p(x,t)\Big).$$

On suppose que $\sigma(t)\sigma(t)^T$ est proportionnel à l'identité [12](page 103), on peut écrire

$$\frac{1}{2}\operatorname{tr}\Big(\sigma(t)\sigma(t)^T\nabla^2 p(x,t)\Big) = \frac{1}{2}\sigma(t)\sigma(t)^T\,\Delta p(x,t),$$

ce qui complète la preuve [13]. □

Remarque 2. *L'équation de Fokker-Planck caractérise l'évolution dans le temps de la densité $p(x,t|x_0,0)$ du processus markovien.*

Définition 2.2.3 (Adjoint du générateur et équation de Fokker–Planck)**.** *Soit $\mathcal{L}$ le générateur infinitésimal d'un processus de Markov défini par*

$$\mathcal{L}\varphi(x,t) = b(x,t)\cdot\nabla\varphi(x,t) + \frac{1}{2}\nabla\cdot\Big(a(t)\nabla\varphi(x,t)\Big),$$

pour toute fonction test φ , où $a(x,t) = \sigma(t)\sigma^T(t)$. L'adjoint de $\mathcal{L}$, noté $\mathcal{L}^*$, est défini par la relation suivante :

$$\int_{\mathbb{R}^d}\varphi(x)\,(\mathcal{L}\psi)(x)\,dx = \int_{\mathbb{R}^d}(\mathcal{L}^*\varphi)(x)\,\psi(x)\,dx,$$

pour toutes fonctions tests φ, ψ.

L'équation de Fokker-Planck associée au processus X_t est alors donnée par

$$\partial_t p(t,x) = \mathcal{L}^* p(t,x).$$

2.3 Retournement temporel et processus inversé

Nous introduisons le processus inversé par :

$$\overline{X}_t = X_{T-t}, \quad t \in [0,T]. \tag{2.3}$$

Nous notons alors $\overline{p}(x,t) = p(x,T-t)$ la densité associée à la chaîne $\overline{X}_t$

Lemme 2.3.1 (EDS du processus inversé). *Si $(X_t)_{t\in[0,T]}$ satisfait (2.1), alors, sous des hypothèses adéquates, le processus inversé $(\overline{X}_t)_{t\in[0,T]}$ est un processus de Markov inhomogène qui satisfait une équation de la forme suivante :*

$$d\overline{X}_t = \overline{b}(X_t,t)\,dt + \overline{\sigma}(t)\,d\overline{W}_t,$$

où $\overline{b}(x,t)$ et $\overline{\sigma}(t)$ sont à déterminer.

Proposition 2.3.1 (Équation de Fokker-Planck associée au processus inversé). *La densité $\overline{p}(x,t)$ associée au processus inversé $\overline{X}_t$ satisfait l'équation de Fokker-Planck suivante :*

$$\partial_t \overline{p}(x,t) = -\nabla\cdot\Big(\overline{b}(x,t)\overline{p}(x,t)\Big) + \frac{1}{2}\overline{\sigma}(t)\overline{\sigma}^T(t)\,\Delta\overline{p}(x,t), \tag{2.4}$$

Nous allons, par la suite, déterminer explicitement $\overline{b}(x,t)$ et $\overline{\sigma}$, en particulier, nous montrerons que :

$$\overline{b}(x,t) = -b(x,T-t) + \sigma(t)\sigma^T(t)\nabla\log p(x,T-t).$$

Démonstration. Cette équation de Fokker-Planck est déterminée de la même manière que X_t. □

Théorème 2 (Formule de retournement du temps). *[6][18]Sous des hypothèses de régularité suffisantes sur b,σ et $(p(.,t))_{0\le t\le T}$ alors :*

$$\overline{\sigma}(t) = \sigma(t)$$

et

$$\overline{b}(x,t) = -b(x,t) + a(t)\,\nabla\log p(x,t), \tag{2.5}$$

où $a(t,x) = \sigma(t)\sigma(t)^T$.

Démonstration. Posons $s = T-t$. Par définition, la densité du processus inversé est

$$\overline{p}(x,t) = p(x,T-t) = p(x,s).$$

En appliquant la règle de dérivées de fonctions composées, on a

$$\frac{\partial}{\partial t}\overline{p}(x,t) = \frac{\partial}{\partial t}p(x,T-t) = -\frac{\partial}{\partial s}p(x,s)\Big|_{s=T-t}.$$

D'une part, la densité $p(x,s)$ satisfait l'équation (2.2) :

$$\partial_s p(x,s) = -\nabla\cdot\Big(b(x,s)p(x,s)\Big) + \frac{1}{2}a(s)\,\Delta p(x,s)$$

En remplaçant s par $T-t$, on obtient :

$$\frac{\partial}{\partial t}\overline{p}(x,t) = \nabla\cdot\Big(b(x,T-t)\overline{p}(x,t))\Big) - \frac{1}{2}a(T-t)\,\Delta\overline{p}(x,t).$$

D'autre part, on a montré que la densité $\overline{p}(x,t)$ satisfaisait l'équation de Fokker–Planck 2.4

$$\partial_t \overline{p}(x,t) = -\nabla \cdot \Big(\overline{b}(x,t)\,\overline{p}(x,t)\Big) + \frac{1}{2}\overline{a}(t)\,\Delta \overline{p}(x,t).$$

Par symétrie de σ, on a

$$\overline{a}(t) = a(T-t) = a(t),$$

et par identification des deux équations précédentes, nous obtenons l'équation suivante :

$$\nabla \cdot \Big(b(x,T-t)p(x,T-t)\Big) - \frac{1}{2}a(t)\,\Delta p(x,T-t) = -\nabla \cdot \Big(\overline{b}(x,t)p(x,T-t)\Big) + \frac{1}{2}a(t)\,\Delta p(x,T-t)$$

En réarrangeant cette équation, nous obtenons :

$$\Leftrightarrow \nabla \cdot \Big(\overline{b}(x,t)p(x,T-t)\Big) = -\nabla \cdot \Big(b(x,T-t)p(x,T-t)\Big) + a(t)\,\Delta p(x,T-t)$$

Comme σ ne dépend pas de x, on a cette relation :

$$a(t)\,\Delta p(x,T-t) = \nabla \cdot \Big(a(T-t,x)\nabla p(x,T-t)\Big).$$

Cela nous donne la relation suivante :

$$\Leftrightarrow \nabla \cdot \Big(\overline{b}(x,t)p(x,T-t)\Big) = -\nabla \cdot \Big(b(x,T-t)p(x,T-t)\Big) + \nabla \cdot \Big(a(T-t,x)\nabla p(x,T-t)\Big).$$

Enfin, par linéarité du gradient et en supposant la stricte positivité de la densité $p(x,T-t)$, on obtient :

$$\overline{b}(x,t) = -b(x,T-t) + a(t,x)\,\nabla \log p(x,T-t).$$

Notons que $\nabla \log p = \frac{\nabla p}{p}$ pour $p > 0$. □

Remarque 3. *Cette formule montre que le drift du processus inversé est obtenu en renversant temporellement le drift associé à X_t et en ajoutant une correction qui est le gradient logarithmique de la densité (le* score*). Ce résultat est fondamental dans les modèles génératifs de diffusion, où le processus inverse joue un rôle central dans la génération de nouvelles données en suivant un chemin temporel inversé, depuis une distribution de bruit vers la distribution des données. Posons $\overline{X}_t = Y_t$. Ainsi, l'EDS de X_{T-t} s'écrit comme suit :*

$$dY_t = \Big(-b(Y_t,T-t) + \sigma(Y_t,T-t)\sigma(Y_t,T-t)^T\,\nabla \log p(Y_t,T-t)\Big)\,dt + \sigma(T-t)\,dW_t, \quad (2.6)$$

avec $W_t \overset{\mathcal{L}}{\sim} \overline{W}_t$.

Notons que dans certaines publications comme Song et al. [17], le signe du coefficient de drift change suite à un changement de variable ($s = T - t \Rightarrow d_t s = -dt$).

2.4 Discrétisation d'Euler-Maruyama

La discrétisation d'Euler-Maruyama est une méthode numérique permettant d'approximer la solution d'une équation différentielle stochastique (EDS) sous la forme :

$$dX_t = b(X_t,t)dt + \sigma(X_t,t)dW_t,$$

où W_t est un mouvement brownien, b est le drift, et σ est la coefficient de diffusion.

Proposition 2.4.1 (Schéma de discrétisation et convergence)**.** *Soit une partition de l'intervalle* $[0, T]$ *donnée par* $\{t_k = k\Delta t\}_{k=0}^{N}$ *avec* $\Delta t = \frac{T}{N}$ *et* $N > 0$*. Sous des hypothèses adéquates sur les coefficients de l'équation différentielle stochastique, le schéma d'Euler-Maruyama s'écrit :*

$$X_{t_{k+1}} = X_{t_k} + b(X_{t_k}, t_k)\Delta t + \sigma(X_{t_k}, t_k)\Delta W_k, \tag{2.7}$$

où $\Delta W_k = W_{t_{k+1}} - W_{t_k} \sim \mathcal{N}(0, \Delta t)$.

Ce schéma fournit une approximation du processus X *avec un ordre de convergence de* $1/2$*. Autrement dit, il existe une constante* $C > 0$ *telle que*

$$\sup_{0 \le t \le T} \mathbb{E}\left[|X_t - X_t^{\Delta t}|\right] \le C\,\Delta t^{1/2}.$$

Démonstration. La démonstration suit les arguments présentés dans le cours de Vaes U. [19]. Nous renvoyons le lecteur à ce cours pour les détails complets de la preuve. □

Remarque 4. *La solution analytique est comme suit :*

$$X_{t_{k+1}} = X_{t_k} + \int_{t_k}^{t_{k+1}} b(X_s, s)ds + \int_{t_k}^{t_{k+1}} \sigma(X_s, s)\, dW_s.$$

2.4.1 Application au processus inverse

Dans le cadre des modèles génératifs de diffusion, la discrétisation d'Euler-Maruyama est utilisée pour simuler le processus inversé (2.6) ; elle est comme suit :

$$Y_{t_{k+1}} = Y_{t_k} - \Big(b(Y_{t_k}, t_N - t_k) - \sigma\sigma^T \nabla \log p(Y_{t_k}, t_N - t_k)\Big)\,\Delta t + \sigma(Y_{t_k}, t_N - t_k)\Delta W_k, \tag{2.8}$$

où $t_k \in [\![0, t_N]\!]$ et $\sigma = \sigma(Y_{t_k}, t_N - t_k)$.

Après cette introduction de la discrétisation d'Euler-Maruyama pour simuler le processus inverse, une question centrale se pose : comment obtenir une estimation précise du terme de correction $\nabla \log p(.,.)$ intervenant dans le drift inversé ?

2.5 Vers le Denoising Score Matching

Dans la pratique, l'estimation du score, c'est-à-dire le terme $\nabla \log p(.,.)$, représente un défi majeur puisque la densité $p(.,.)$ n'est pas connue de manière explicite. Pour surmonter cette difficulté, on recourt à l'entraînement d'un réseau de neurones (UNET [4.1] pour des images, ou des perceptrons multi-couches pour des tables) qui apprend à approximer ce gradient via des techniques telles que le **denoising score matching**. Ces méthodes consistent à ajouter un bruit contrôlé aux données, puis à ajuster le réseau pour que sa sortie converge vers le gradient du logarithme de la densité. Le **score** ainsi obtenu joue un rôle central dans la simulation du processus inverse, en guidant les trajectoires générées pour transformer progressivement une distribution de bruit en une distribution proche de celle des données initiales.

2.5.1 Expérimentation sur une distribution en anneau

Pour tester l'algorithme, nous considérons une distribution en anneau, où les points sont générés autour d'un cercle de rayon fixe avec une légère dispersion. Le bruit est ajouté progressivement pour simuler un processus de diffusion, qui conduit à une distribution standard.

Le processus, défini dans 3.1, satisfait l'EDS suivante :

$$dX_t = -\beta X_t dt + \sqrt{2\beta} dW_t, \tag{2.9}$$

où $\beta > 0$, et W_t est un mouvement brownien standard.
Et son inverse défini par l'EDS suivante :

$$dX_t = \beta(\nabla \log p(X_t, t) - X_t)dt + \sqrt{2\beta} dW_t, \tag{2.10}$$

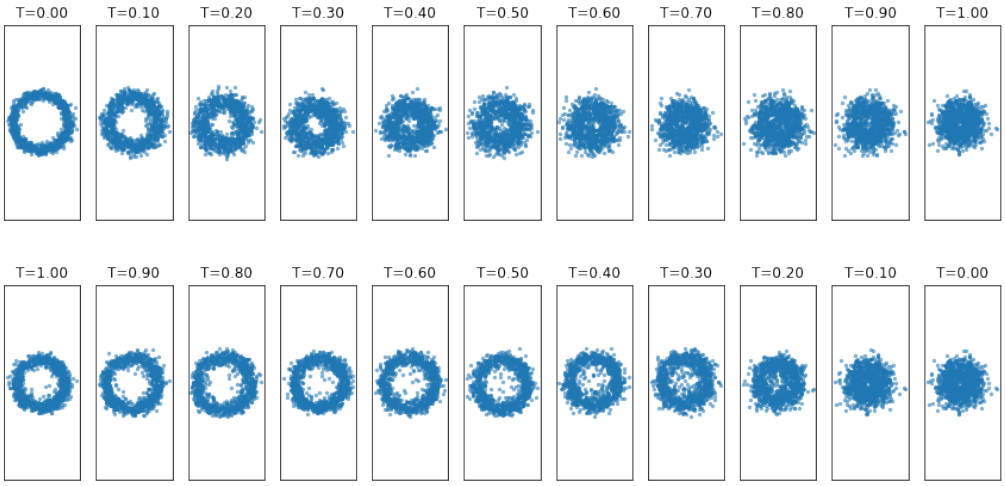


FIGURE 2.1 – Simulation d'un bruitage et inversion du temps sur un anneau, **FID** = 0.0003

Doit-on se contenter de comparaisons visuelles ? Non, il existe une métrique pour évaluer la qualité des images générées : la **FID**.

FIGURE 2.2 – Comparaison visuelle

2.5.2 Métrique d'évaluation des données générées

Définition 2.5.1 (Distance de Wasserstein)**.** *La distance de Wasserstein d'ordre 2 entre deux mesures de probabilités μ et ν sur $\mathbb{R}^n$ est définie par :*

$$W_2(\mu,\nu) = \inf_{\gamma\in\Gamma(\mu,\nu)} \mathbb{E}_{(X,Y)\sim\gamma}\left[\|X-Y\|_2^2\right]^{1/2}, \tag{2.11}$$

où $\Gamma(\mu,\nu)$ désigne l'ensemble des lois jointes de μ et ν [3].

Lorsqu'on considère deux lois gaussiennes $\mathcal{N}(m_1,\Sigma_1)$ et $\mathcal{N}(m_2,\Sigma_2)$, la distance de Wasserstein admet une forme explicite :

Proposition 2.5.1. *Si $\mu = \mathcal{N}(m_1,\Sigma_1)$ et $\nu = \mathcal{N}(m_2,\Sigma_2)$, alors la distance de Wasserstein s'exprime comme :*

$$W_2^2(\mu,\nu) = \|m_1-m_2\|^2 + \mathrm{Tr}\left(\Sigma_1+\Sigma_2-2(\Sigma_1^{1/2}\Sigma_2\Sigma_1^{1/2})^{1/2}\right). \tag{2.12}$$

Démonstration. La démonstration suit les arguments présentés dans [3]. Nous renvoyons le lecteur à cet article pour les détails de la preuve. □

Définition 2.5.2. *La Fréchet Inception Distance est une métrique couramment utilisée pour évaluer la qualité des images générées. Elle repose sur* ***l'hypothèse que les "représentations" des images suivent une loi gaussienne dans l'espace latent****. Elle est définie par :*

$$\mathrm{FID} = \|\mu_r-\mu_g\|^2 + \mathrm{Tr}\left(\Sigma_r+\Sigma_g-2(\Sigma_r\Sigma_g)^{1/2}\right), \tag{2.13}$$

où (μ_r,Σ_r) et (μ_g,Σ_g) sont respectivement les moyennes et matrices de covariance des images réelles et générées.

Remarque 5. *On remarque que la FID repose directement sur la formule de la distance de Wasserstein pour des lois normales, en supposant notamment la normalité des distributions et la commutativité des matrices de covariance ($\Sigma_r\Sigma_g = \Sigma_g\Sigma_r$.)*

Implémentation pratique :

Dans une application typique, nous importons un modèle pré-entraîné tel qu'Inception-V3, entraîné sur ImageNet [4], et que nous pouvons charger directement depuis des plateformes comme HuggingFace ou TensorFlow.
Inception-V3, bien qu'initialement conçu pour la classification, est ici utilisé pour extraire des *features* (ou activations) des images.
Dans le cas du jeu de données MNIST, nous utilisons Inception-V3 pour extraire des caractéristiques "abstraites" des images réelles et générées. Ensuite, nous calculons la FID en comparant les distributions de ces caractéristiques. Concrètement :

- Pour les images réelles, nous extrayons les activations d'Inception-V3 et calculons la moyenne et la covariance des activations.
- Pour les images générées par un modèle génératif, nous effectuons le même processus pour obtenir les moyennes et covariances des activations.

Bien que cette approche soit couramment utilisée, dans ce travail, la FID ne sera pas appliquée directement à un jeu de données comme MNIST. Pour plus de détails sur l'utilisation de la FID, vous pouvez consulter ce lien [8].

Chapitre 3

Denoising Score Matching

Retournons au cadre itératif du modèle de diffusion, dans lequel il s'agit d'estimer $\nabla_x \log p_t(\hat{X}_t)$. À partir du score de Fisher, nous pouvons définir le score d'un réseau qui chercherait à estimer ce terme en fonction du temps t du processus, comme une intégrale sur l'intervalle de temps des scores de Fisher. Nous définissons donc la perte suivante :

$$\mathcal{L}_{DSM} = \int_0^T \lambda(t)\mathbf{E}_{X_t}[\|s_\theta(X_t, t) - \nabla_{X_t} \log p_t(X_t)\|^2]$$

où $\lambda(t)$ est un facteur de poids positif, optionnel.
Quoi qu'il en soit, les deux problèmes soulevés dans la première section ne sont toujours pas résolus, la difficulté se résumant dans le fait de ne pas connaître la loi p_t pour t donné.

Dans le cadre DSM, le problème est contourné en deux temps. Premièrement, nous choisirons un processus de diffusion précis qui, sans être de loi de transition p_t connue, possède une expression explicite des lois de transitions conditionnellement à X_0. Deuxièmement, une réécriture de la fonction de perte $\mathcal{L}_{DSM}$ permettra d'employer ces lois conditionnelles, jusqu'à un cadre applicable en pratique.

3.1 Le processus Ornstein-Uhlenbeck

Le processus de Ornstein-Uhlenbeck est celui historiquement utilisé dans l'approche DSM. On verra plus tard qu'un autre processus très proche sera utilisé dans la pratique : les résulats de cette section se font cependant à partir de Ornstein-Uhlenbeck, par souci de cohérence.
Un processus d'Ornstein-Uhlenbeck est un processus de la forme :

$$dX_t = -\beta X_t dt + \sigma dW_t$$

Ce processus vérifie :

$$\begin{cases} p_{t|0} = \mathcal{N}(e^{-\beta t}X_0, \sigma_t^2 I) \\ \sigma_t^2 = \dfrac{\sigma^2}{2\beta}(1 - e^{-2\beta t}) \end{cases}$$

On peut déduire de ces propriétés une résolution du problème concernant l'ignorance de la loi limite du processus de diffusion. En effet, on a :

$$p_{t|0} \xrightarrow{t\to\infty} \mathcal{N}(0, \frac{\sigma^2}{2\beta})$$

loi limite indépendante de l'état initial. On a donc une loi ($\sim X_T$ pour T grand) qui ne pose pas de problème de simulation, permettant de donner un point de départ au processus *reverse* visant à générer des images.

Nous notons alors que le processus Ornstein-Uhlenbeck peut se réécrire (notations conservées par la suite) :

$$\begin{cases} X_t = \gamma_t X_0 + \sigma_t \varepsilon \\ \gamma_t = e^{-\beta t} \\ \varepsilon \sim \mathcal{N}(0, I) \end{cases}$$

3.2 Réécriture de $\mathcal{L}_{DSM}$

3.2.1 Écriture conditionnelle

Nous nous donnons dans cette section une autre écriture de la fonction de perte, qui comme annoncé plus tôt permet d'utiliser les transitions conditionnelles (connues dans le cadre O.U.). La réécriture suivante demande une justification.

$$\mathcal{L}(\theta) = \int_0^T \lambda(t) \mathbb{E}_{X_0} \left[\mathbb{E}_{X_t|X_0} \left[\| s_\theta(X_t, t) - \nabla_{X_t} \log p_{t|0}(X_t|X_0) \|^2 \mid X_0 \right] \right] dt + C \tag{3.1}$$

preuve :

$$\begin{aligned} \nabla_{X_t} \log p_t(X_t) &= \frac{\nabla_{X_t} p_t(X_t)}{p_t(X_t)} \\ &= \frac{1}{p_t(X_t)} \nabla_{X_t} \int_{\mathbb{R}^d} p_{t|0}(X_t|X_0) p_0(X_0) dX_0 \\ &= \int_{\mathbb{R}^d} \left(\nabla_{X_t} p_{t|0}(X_t|X_0) \right) \frac{p_0(X_0)}{p_t(X_t)} dX_0 \\ &= \int_{\mathbb{R}^d} \left(\nabla_{X_t} \log p_{t|0}(X_t|X_0) \right) \frac{p_0(X_0) p_{t|0}(X_t|X_0)}{p_t(X_t)} dX_0 \\ &= \int_{\mathbb{R}^d} \left(\nabla_{X_t} \log p_{t|0}(X_t|X_0) \right) p_{0|t}(X_0|X_t) dX_0 \\ &= \mathbb{E}_{X_0|X_t} \left[\nabla_{X_t} \log p_{t|0}(X_t|X_0) \mid X_t \right] \end{aligned}$$

$$\begin{aligned} \mathcal{L}(\theta) &= \int_0^T \lambda(t) \mathbb{E}_{X_t} \left[\| s_\theta(X_t, t) - \nabla_{X_t} \log p_t(X_t) \|^2 \right] dt \\ &= \int_0^T \lambda(t) \mathbb{E}_{X_t} \left[\| s_\theta(X_t, t) \|^2 - 2 \langle s_\theta(X_t, t), \nabla_{X_t} \log p_t(X_t) \rangle \right] dt + C \\ &= \int_0^T \lambda(t) \mathbb{E}_{X_t} \left[\| s_\theta(X_t, t) \|^2 - 2 \langle s_\theta(X_t, t), \mathbb{E}_{X_0|X_t} [\nabla_{X_t} \log p_{t|0}(X_t|X_0) \mid X_t] \rangle \right] dt + C \\ &= \int_0^T \lambda(t) \mathbb{E}_{X_t} \left[\mathbb{E}_{X_0|X_t} [\| s_\theta(X_t, t) \|^2 - 2 \langle s_\theta(X_t, t), \nabla_{X_t} \log p_{t|0}(X_t|X_0) \rangle \mid X_t] \right] dt + C \\ &= \int_0^T \lambda(t) \mathbb{E}_{X_t, X_0} \left[\| s_\theta(X_t, t) - \nabla_{X_t} \log p_{t|0}(X_t|X_0) \|^2 \right] dt + C \end{aligned}$$

3.2.2 Réécriture avec Ornstein Uhlenbeck

Nous avons à présent fait le nécessaire pour donner une expression de la fonction de perte adaptée à un processus de type Ornstein-Uhlenbeck, qui prendra une forme particulièrement propice à la mise en place d'un algorithme d'optimisation.

En utilisant $X_t := \gamma_t X_0 + \sigma_t \varepsilon$, la fonction de score se simplifie comme

$$\nabla_{X_t} \log p_t(X_t|X_0) = \frac{\gamma_t X_0 - X_t}{\sigma_t^2} := -\frac{\varepsilon}{\sigma_t}$$

On définit le *scaled score network* (prédiction du bruit) par :

$$\varepsilon_\theta(X_t, t) := -\sigma_t s_\theta(X_t, t)$$

Alors, notre loss devient :

$$\mathcal{L}(\theta) = \int_0^T \lambda(t)\mathbb{E}_{X_0}\left[\mathbb{E}_{X_t|X_0}\left[\|s_\theta(X_t,t) - \nabla_{X_t}\log p_{t|0}(X_t|X_0)\|^2 \mid X_0\right]\right] dt$$

$$= \int_0^T \frac{\lambda(t)}{\sigma_t^2}\mathbb{E}_{X_0}\left[\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,I)}\left[\|\varepsilon_\theta(\gamma_t X_0 + \sigma_t\varepsilon, t) - \varepsilon\|^2 \mid X_0\right]\right] dt$$

$$= T\,\mathbb{E}_{\substack{X_0\sim p_0 \\ t\sim\mathcal{U}([0,T]) \\ \varepsilon\sim\mathcal{N}(0,I)}}\left[\frac{\lambda(t)}{\sigma_t^2}\|\varepsilon_\theta(\gamma_t X_0 + \sigma_t\varepsilon, t) - \varepsilon\|^2\right]$$

Laquelle formule donne une routine d'entraînement avec tirage du temps selon une loi uniforme continue.

while (not converged)
$X_0 \sim p_0 = p_{\text{data}}$
$t \sim \text{Uniform}([0,T])$
$\varepsilon \sim \mathcal{N}(0,I)$
$X_t = \gamma_t X_0 + \sigma_t \varepsilon$
Call optimizer with $\frac{\lambda(t)}{\sigma_t^2}\nabla_\theta\|\varepsilon_\theta(X_t,t) - \varepsilon\|^2$
end

FIGURE 3.1 – Routine d'entraînement du DSM avec O.U. [15]

3.3 Vers la pratique : DDPM et choix de réseau

On remarquera qu'un dernier frein à l'application de nos résultats est la nature continue du processus employé. En effet, l'apparition du temps à l'entrée du réseau de neurones pose un problème de taille sur deux "dimensions" :

— *horizontalement*, le temps apparaissant continu dans le tirage de la loi uniforme, chaque image originelle implique une infinité de versions bruitées. La multiplication de ces cas complique l'apprentissage du réseau.

— *verticalement*, l'insertion du temps dans le réseau est, à priori, l'insertion d'un seul paramètre ; paramètre qui devient négligé devant la dimension d'une image : par exemple avec MNIST, le temps représentera un $785^{ème}$ de la donnée.

Pour contourner le premier problème, on introduira un modèle issu du DSM qui discrétise le modèle : le DDPM [7]. C'est un modèle légèrement différent du DSM sur d'autres points, mais qui reste très proche.
Pour le second problème, c'est le choix de l'architecture du réseau qui sera à explorer. Historiquement, ce sont les modèles UNET avec time embeddings que l'on retient. Pour éviter de dévier du propos, et puisque ces réseaux sont plutôt connus, nous discutons ce sujet en annexe [14].

3.3.1 DDPM (*Denoising Diffusion Probabilistic Models*)

Le DDPM, un processus de bruitage, dont la loi conditionnelle à l'état initial est exprimée par une gaussienne connue, ce qui nous permet de revenir très rapidement à l'écriture de la fonction de perte vu précédemment.

On définit le processus associé au DDPM par :

$$X_0 \sim p_0 = p_{\text{data}}$$

$$X_t | X_{t-1} \sim \mathcal{N}\left(\sqrt{1-\beta_t} X_{t-1}, \beta_t I\right) \quad \text{for} \quad t = 1, \dots, T \quad (0 < \beta_t < 1)$$

$$\implies X_t := \sqrt{1-\beta_t} X_{t-1} + \sqrt{\beta_t} Z_t, \quad Z_t \sim \mathcal{N}(0, I), \quad \text{for} \quad t = 1, \dots, T$$

Après quelques calculs, on en déduit :

$$X_t | X_0 \sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t} X_0, (1-\bar{\alpha}_t) I\right)$$

où

$$\bar{\alpha}_t = \prod_{s=1}^{t} (1-\beta_s)$$

La fonction de perte peut alors être recalculée d'une manière en tout point identique à celle du cadre DSM.

En posant : $\begin{cases} X_t := \sqrt{\bar{\alpha}_t} X_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, I) \\ \varepsilon_\theta := -\sqrt{1-\bar{\alpha}_t} s_\theta \\ \tilde{\lambda}_t = \frac{\lambda_t \beta_t^2}{(1-\beta_t)(1-\bar{\alpha}_t)} \end{cases}$

On calcule :

$$
\begin{aligned}
\mathcal{L}(\theta) &= \sum_{t=1}^{T} \lambda_t \mathbb{E}_{X_t}\left[\|\mu(X_t,t) - \mu_\theta(X_t,t)\|^2\right] \\
&= \sum_{t=1}^{T} \frac{\lambda_t \beta_t^2}{1-\beta_t} \mathbb{E}_{X_t}\left[\|\nabla_{X_t} \log p_t(X_t) - s_\theta(X_t,t)\|^2\right] \\
&= \sum_{t=1}^{T} \frac{\lambda_t \beta_t^2}{1-\beta_t} \mathbb{E}_{X_0,X_t}\left[\|\nabla_{X_t} \log p_{t|0}(X_t|X_0) - s_\theta(X_t,t)\|^2\right] + C \\
&= \sum_{t=1}^{T} \frac{\lambda_t \beta_t^2}{1-\beta_t} \mathbb{E}_{X_0,X_t}\left[\left\|-\frac{1}{1-\bar{\alpha}_t}(X_t - \sqrt{\bar{\alpha}_t}X_0) - s_\theta(X_t,t)\right\|^2\right] + C \\
&= \sum_{t=1}^{T} \frac{\lambda_t \beta_t^2}{(1-\beta_t)(1-\bar{\alpha}_t)} \mathbb{E}_{\substack{X_0 \sim p_{\text{data}} \\ \varepsilon \sim \mathcal{N}(0,I)}}\left[\|\varepsilon - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}X_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon, t)\|^2\right] + C \\
&= \sum_{t=1}^{T} \tilde{\lambda}_t \mathbb{E}_{\substack{X_0 \sim p_{\text{data}} \\ \varepsilon \sim \mathcal{N}(0,I)}}\left[\|\varepsilon - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}X_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon, t)\|^2\right] + C \\
&= T\,\mathbb{E}_{\substack{X_0 \sim p_{\text{data}} \\ \varepsilon \sim \mathcal{N}(0,I) \\ t \sim \mathcal{U}(\{1,\dots,T\})}}\left[\tilde{\lambda}_t \|\varepsilon - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}X_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon, t)\|^2\right] + C
\end{aligned}
$$

On arrive donc de nouveau à une formule facilement interprétable dans le cadre algorithmique, et la routine devient :

while (not converged)
$X_0 \sim p_0 = p_{\text{data}}$
$t \sim \text{Uniform}(\{1, \dots, T\})$
$\varepsilon \sim \mathcal{N}(0, I)$
$X_t = \sqrt{\bar{\alpha}_t} X_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon$
Call optimizer with $\tilde{\lambda}_t \nabla_\theta \|\varepsilon_\theta(X_t, t) - \varepsilon\|^2$
end

FIGURE 3.2 – routine d'entraînement DDPM [15]

3.4 Mise en pratique sur MNIST

Les codes sources concernant cette section sont tous accessibles sur les dépôts github, dont les liens sont renseignés en fin de rapport.
Cette section expose la mise en pratique d'un algorithme DDPM à la base de donnée MNIST. Dans un premier temps, nous exposons les paramètres utilisés. Puis, on montrera des résultats de génération issus du modèle. Enfin, on essaie de comparer notre génération avec celle de deux autres modèles (un VAE et un GAN).

3.4.1 Paramètres

Le choix des paramètres est dû à [7].
Quelques explications suivent sur certains de ces paramètres.

	Valeur
Seed	0
Paramètres de Diffusion	
Nombre de Timesteps (n_steps)	1000
β_{min}	0.0001
β_{max}	0.02
Paramètres d'Entraînement	
Taille de Batch (batch_size)	128
Nombre d'Epochs (n_epochs)	82
Taux d'Apprentissage (lr)	0.0001
Paramètres UNet	
Profondeur	4 blocs
Dimension de l'embedding temporel (time_emb_dim)	100
Nombre de paramètres de l'embedding	100 000
Nombre total de paramètres	1 435 486

TABLE 3.1 – Tableau des Paramètres du Modèle de Diffusion

— *seed* : nous renseignons la graine de génération dans un soucis de reproductibilité. Les résultats de cette section proviennent tous de la graine "0".

— β_{min} *et* β_{max} : les valeurs $\beta_{t=1}$ et $\beta_{t=T}$ respectivement. Les valeurs intermédiaires sont déterminées par *numpy.linspace* d'après le nombre de timesteps renseigné.

— *Profondeur* : le "format" du UNET. On a utilisé 4 blocs descendant, suivis par un bloc central (*botleneck*), puis par 4 blocs montant. Cette architecture est la même que celle visible sur la figure 4.1 en annexe.

3.4.2 Résultats

Après entraînement, nous obtenons les résultats de la figure 3.3.

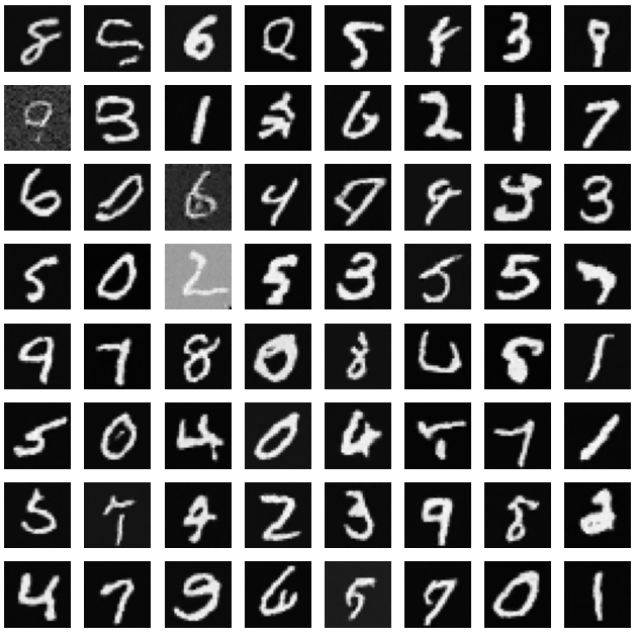


FIGURE 3.3 – 64 résultats de génération du DDPM

3.4.3 Comparaison avec d'autres modèles

Méthodologie

Dans le cadre MNIST, on pourrait être tenté, soit de se contenter d'une appréciation visuelle des résultats, soit d'utiliser une métrique connue, comme FID mentionnée précédemment.
La première façon de faire a pour tort de confondre la qualité d'un nombre et sa similarité avec MNIST à proprement parler. C'est bien cette similarité qui doit primer. La seconde façon de faire donne un score clair, mais qui reste une "boîte noire".
On s'est donc permis pour les besoins de l'expérience de proposer une approche plus manuelle pour la comparaison et l'évaluation.

Puisqu'il est rapide et facile de produire un modèle de classification sur MNIST à partir d'un réseau de neurone, nous entraînons dans un premier temps un CNN (*convolutional neural network*) à déterminer le label d'un digit MNIST. Après quelques epochs, sa précision est de 98,91%.
Ce modèle donne en sortie, non pas une prédiction à proprement parler, mais dix logits, qui représentent chacun la certitude du classifieur que le digit auquel il est confronté apartienne à telle ou telle classe. La prédiction se fait ensuite en prenant l'argmax de ces logits.
On a choisi de considérer ce "logit maximum" comme un score sur l'image générée, puisqu'il n'importe pas de savoir a priori quel nombre est généré, mais plutôt de savoir dans quelle mesure ce nombre est à considérer comme un MNIST reconnaissable.

Cette approche permet à la fois d'obtenir, comme pour FID, un résultat global (moyenne des logits maximums pour chaque modèle), mais aussi d'extraire les meilleurs résultats de chaque modèle, ou encore de tester la variété, en utilisant le classifieur pour déterminer quels nombres ont été générés.

Les tests sont réalisés sur 100 images pour chaque modèle. Les modèles de VAE et de GAN ont été entraînés sur la même machine que le DDPM, avec un peu plus de 100 epochs. Le choix du nombre d'epochs a été fait sur la base du temps d'entraînement : on a essayé de laisser environ autant de temps d'entraînement à chaque modèle.

Précision de la génération

Modèle	MML
Témoin (MNIST réel)	18.9480
VAE	11.1808
GAN	11.0247
DDPM (ours)	12.7437

TABLE 3.2 – Résultats des modèles avec le score MML (mean max logit)

On constate que, selon ce score, le DDPM donne les meilleurs résultats, suivi de près par les deux autres modèles. Le calcul du MML pour les données réelles permet de mettre en perspective ces résultats.

Variété de la génération

On a ici utilisé le même classifieur que précédemment pour prédire les labels des nombres générés. On regarde ensuite la distribution des labels ainsi prédits (figure 3.4).

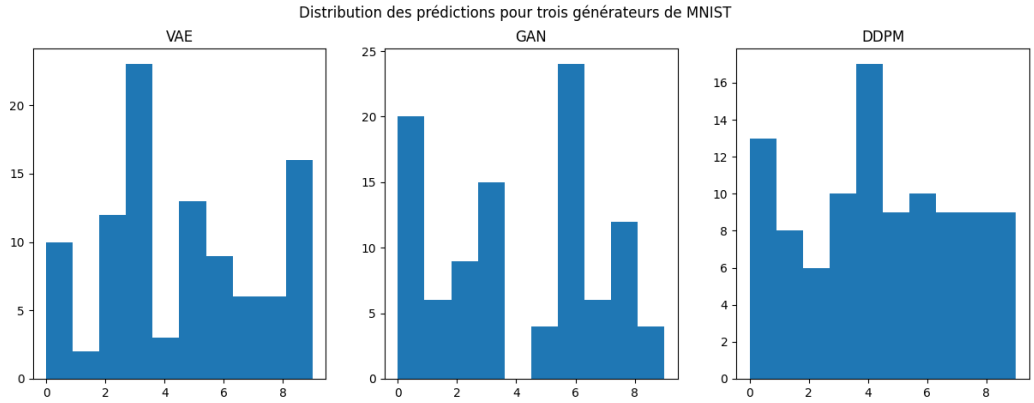


FIGURE 3.4 – Distribution des labels prédits sur les générations des modèles comparés

Ces résultats illustrent bien un des avantages annoncés de la diffusion, qui est de proposer des générations plus variées que les modèles précédents.

Comparaison des meilleurs résultats pour chaque label

On conclue en exposant ici les meilleures générations pour chaque label, pour chaque modèle. On notera d'ailleurs l'absence de "4" prédit pour le générateur GAN, ce qui vient appuyer l'avantage de DDPM en terme de diversité.

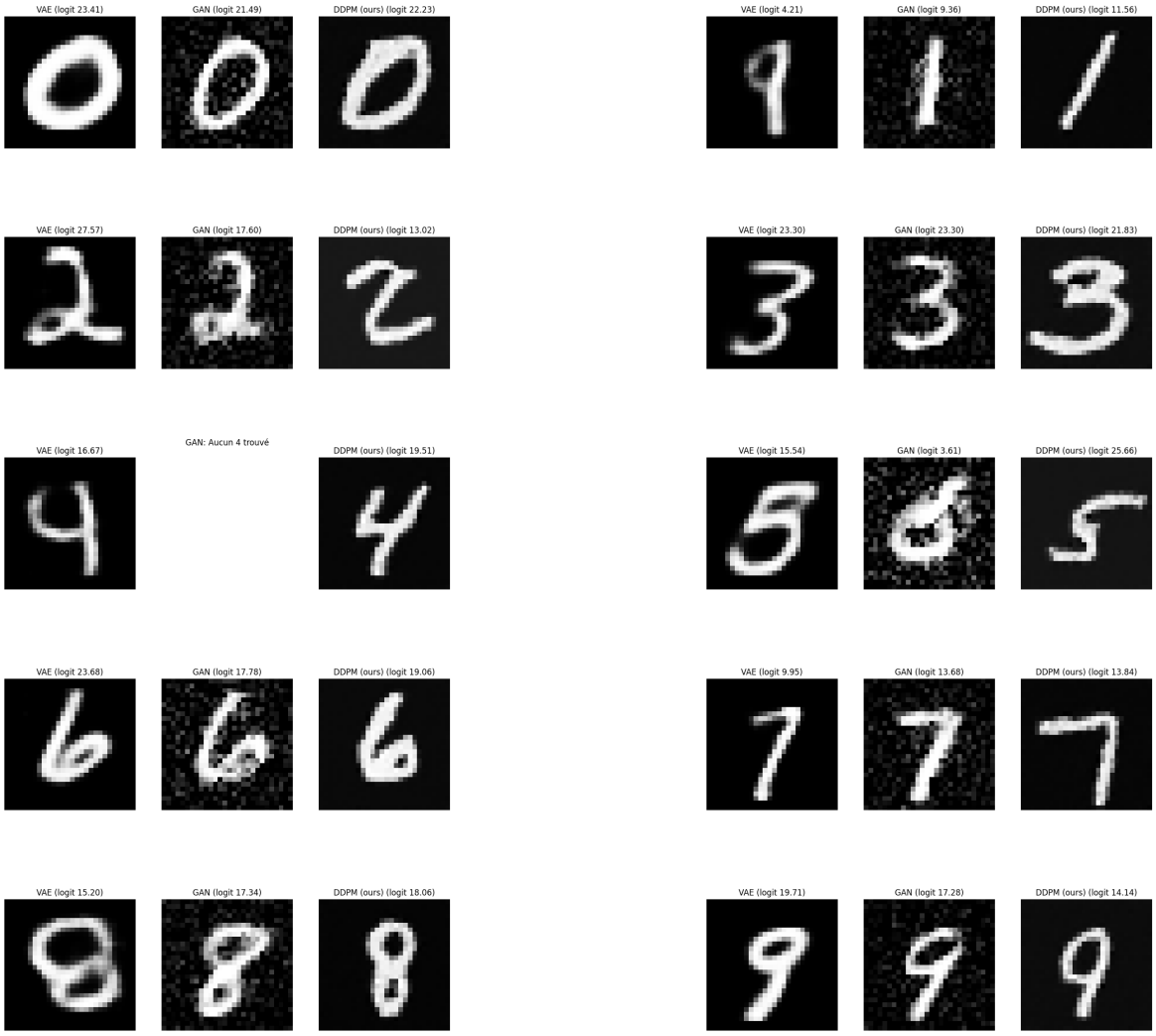


FIGURE 3.5 – Meilleurs digits générées par chaque modèle (score MML)

Chapitre 4

Sliced Score matching

4.1 Présentation

Pour entraîner le réseau de neurones du score, nous pouvons utiliser une autre erreur qu'on appelle erreur de Sliced Score Matching. L'expression de cette nouvelle fonction de perte est issue d'un développement de la formule initiale (celle donnée à la page 14) qui donne $\mathrm{Tr}(J_{X_t} s_\theta(X_t, t))$ (la trace de la matrice Jacobienne de $X_t \mapsto s_\theta(X_t, t)$) que l'on peut calculer numériquement. Néanmoins, pour réduire le coût de calcul, cette méthode utilise un vecteur aléatoire ν qui permet de remplacer $\mathrm{Tr}(J_{X_t} s_\theta(X_t, t))$ par le terme $\frac{d}{dh}\nu^T s_\theta(X_t + h\nu, t)|_{h=0}$. En effet, la rétropropagation sur ce dernier terme est moins coûteuse que sur la trace.

4.2 Formule

La proposition suivante montre l'égalité entre la fonction de perte initiale présentée au début de chapitre 3 et l'erreur de Sliced Score Matching, qui correspond à la deuxième ligne de la formule.

Proposition 4.2.1. *Soit ν un vecteur aléatoire à valeurs dans $\mathbb{R}^n$ tel que $\mathbb{E}_\nu[\nu_i \nu_j] = \delta_{ij}$ ($= 1$ si $i = j$ et $= 0$ sinon). Nous avons le résultat suivant :*

$$\begin{aligned}\mathcal{L}(\theta) &= \int_0^T \lambda(t)\mathbb{E}_{X_t}[||s_\theta(X_t, t) - \nabla_{X_t} \log p_t(X_t)||^2]dt \\ &= \int_0^T \lambda(t)\mathbb{E}_{X_t}[||s_\theta(X_t, t)||^2 + 2\mathbb{E}_\nu[\frac{d}{dh}\nu^T s_\theta(X_t + h\nu, t)|_{h=0}]]dt + C\end{aligned}$$

Où C est une constante par rapport à θ.

Démonstration. Nous montrerons ce résultat en trois étapes :

1. $\forall A \in \mathcal{M}_n(\mathbb{R}), \mathbb{E}_\nu[\nu^T A \nu] = \mathrm{Tr}(A)$
2. $\mathcal{L}(\theta) = \int_0^T \lambda(t)\mathbb{E}_{X_t}[||s_\theta(X_t, t)||^2 - 2\langle s_\theta(X_t, t), \nabla_{X_t} \log p_t(X_t)\rangle]dt + C$
3. $-\mathbb{E}_{X_t}[\langle s_\theta(X_t, t), \nabla_{X_t} \log p_t(X_t)\rangle] = \mathbb{E}_{X_t}\mathbb{E}_\nu[\frac{d}{dh}\nu^T s_\theta(X_t + h\nu, t)|_{h=0}]$

1. On note $A = (a_{ij})$.

$$\mathbb{E}_\nu[\nu^T A \nu] = \mathbb{E}_\nu[\sum_{i,j} a_{ij}\nu_i\nu_j] = \sum_{i,j} a_{ij}\mathbb{E}_\nu[\nu_i\nu_j] = \sum_i a_{ii} = \mathrm{Tr}(A)$$

Le terme $\nu^T A \nu$ est appelé estimateur de trace de Hutchinson.

2.

$$||s_\theta(X_t, t) - \nabla_{X_t} \log p_t(X_t)||^2 = ||s_\theta(X_t, t)||^2 - 2\langle s_\theta(X_t, t), \nabla_{X_t} \log p_t(X_t)\rangle + ||\nabla_{X_t} \log p_t(X_t)||^2$$

Comme $\int_0^T \lambda(t)\mathbb{E}_{X_t}[||\nabla_{X_t} \log p_t(X_t)||^2]dt$ est constant par rapport à θ, on peut noter ce terme par C.
3.

$$\begin{aligned}-\mathbb{E}_{X_t}[\langle s_\theta(X_t,t), \nabla_{X_t} \log p_t(X_t)\rangle] &= -\int_{\mathbb{R}^n} \langle s_\theta(x,t), \nabla_x \log p_t(x)\rangle p_t(x)dx \\ &= -\int_{\mathbb{R}^n} \langle s_\theta(x,t), \nabla_x p_t(x)\rangle dx\end{aligned}$$

Car $\nabla_x \log p_t(x) = \nabla_x p_t(x)/p_t(x)$.
On note $s_\theta(x,t) = (s_\theta(x,t)_1, s_\theta(x,t)_2, ..., s_\theta(x,t)_n)$. Pour i compris entre 1 et n, on a par intégration par partie :

$$\int_{x_i\in\mathbb{R}} s_\theta(x,t)_i \frac{\partial p_t(x)}{\partial x_i} dx_i = [s_\theta(x,t)_i p_t(x)]_{x_i=-\infty}^{x_i=+\infty} - \int_{x_i\in\mathbb{R}} \frac{\partial s_\theta(x,t)_i}{\partial x_i} p_t(x)dx_i$$

Pour la suite, on admettra que $x_i \mapsto s_\theta(x,t)_i p_t(x)$ tend vers 0 lorsque $|x_i|$ tend vers l'infini. Ainsi, en appliquant aussi le théorème de Fubini, on a :

$$\begin{aligned}\int_{\mathbb{R}^n} s_\theta(x,t)_i \frac{\partial p_t(x)}{\partial x_i} dx &= \int_{x_1\in\mathbb{R}} \int_{x_2\in\mathbb{R}} ... \int_{x_n\in\mathbb{R}} s_\theta(x,t)_i \frac{\partial p_t(x)}{\partial x_i} dx_n ... dx_2 dx_1 \\ &= \int_{x_1\in\mathbb{R}} ... \int_{x_n\in\mathbb{R}} \int_{x_i\in\mathbb{R}} s_\theta(x,t)_i \frac{\partial p_t(x)}{\partial x_i} dx_i dx_n ... dx_1 \\ &= -\int_{x_1\in\mathbb{R}} ... \int_{x_n\in\mathbb{R}} \int_{x_i\in\mathbb{R}} \frac{\partial s_\theta(x,t)_i}{\partial x_i} p_t(x) dx_i dx_n ... dx_1 \\ &= -\int_{\mathbb{R}^n} \frac{\partial s_\theta(x,t)_i}{\partial x_i} p_t(x)dx\end{aligned}$$

Par linéarité de l'intégrale et en utilisant la propriété montrée en 1., on obtient alors :

$$\begin{aligned}-\mathbb{E}_{X_t}[\langle s_\theta(X_t,t), \nabla_{X_t} \log p_t(X_t)\rangle] &= \int_{\mathbb{R}^n} \sum_i \frac{\partial s_\theta(x,t)_i}{\partial x_i} p_t(x)dx \\ &= \mathbb{E}_{X_t}[\mathrm{Tr}(J_{X_t} s_\theta(X_t,t))] \\ &= \mathbb{E}_{X_t}\mathbb{E}_\nu[\nu^T J_{X_t} s_\theta(X_t,t)\nu]\end{aligned}$$

Où $J_{X_t} s_\theta(X_t,t)$ est la matrice Jacobienne de $x \mapsto s_\theta(x,t)$ au point X_t. Pour la dernière étape, on pose :

$$\eta : \left\{ \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{R}^n \\ h & \longmapsto & s_\theta(X_t + h\nu, t) \end{array} \right.$$

On a :

$$\frac{d}{dh} s_\theta(X_t + h\nu, t) = \eta'(h) = J_{X_t+h\nu} s_\theta(X_t + h\nu, t)\nu$$

Et ainsi :

$$\nu^T J_{X_t} s_\theta(X_t,t)\nu = \nu^T J_{X_t+h\nu} s_\theta(X_t + h\nu, t)\nu|_{h=0} = \frac{d}{dh} \nu^T s_\theta(X_t + h\nu, t)|_{h=0}$$

Ce qui donne le résultat de l'étape 3.
Pour déduire la formule du Sliced Score Matching, il suffit d'injecter l'égalité de l'étape 3. dans l'étape 2. ce qui donne le résultat de la proposition. □

4.3 Algorithme d'entraînement

Pour entraîner le réseau de neurones du score en considérant l'erreur SSM, pour chaque image X_0 tirée de notre base de données on applique les opérations suivantes :

Input : Une image initiale X_0 issue de la distribution de départ
Output : Mise à jour des paramètres θ du score s_θ

1 On tire t selon la loi $\mathcal{U}(0, T)$;
2 On bruite X_0 selon l'équation de diffusion jusqu'à obtenir une image X_t bruitée au temps t ;
3 On tire un vecteur aléatoire ν dans $\mathbb{R}^n$ dont la loi implique la propriété $\mathbb{E}[\nu\nu^T] = I$ (on peut prendre $\nu \sim \mathcal{N}(0, I)$ ou $\nu = (\nu_1, ...\nu_n)$ tel que $\nu_1, ...\nu_n$ est une suite i.i.d de variables de loi de Rademacher);
4 On calcule $\frac{d}{dh}\nu^T s_\theta(X_t + h\nu, t)|_{h=0}$;
5 Enfin, on fait un pas de descente de gradient avec

$$\lambda(t)\nabla_\theta(||s_\theta(X_t, t)||^2 + 2\frac{d}{dh}\nu^T s_\theta(X_t + h\nu, t)|_{h=0})$$

Algorithme 1 : Algorithme d'entraînement du SSM

4.4 Avantages et inconvénients

Contrairement au DSM, nous n'avons pas besoin d'évaluer la densité $p_t(X_t|X_0)$. Cependant, malgré l'utilisation de l'estimateur de Hutchinson ($\nu^T A\nu$), cette méthode reste très coûteuse à cause de la nécessité de calculer la dérivée $\frac{d}{dh}\nu^T s_\theta(X_t+h\nu, t)|_{h=0}$ pour chaque X_t, ce qui devient extrémement lourd sur des grandes bases de données. Enfin, selon [15], le DSM performerait mieux que le SSM lorsqu'il est possible de l'utiliser. Néanmoins, la méthode DDPM reste la plus utilisée dans l'industrie parmi les autres méthodes de discrétisation.

Discussion

Dans cette étude, nous avons exploré une approche générative basée sur le renversement temporel des processus de diffusion, combinée à une discrétisation par Euler-Maruyama. Cette méthode repose sur l'estimation du score, c'est-à-dire du gradient du logarithme de la densité $\nabla \log p(x,t)$, et présente plusieurs avantages par rapport aux approches génératives plus classiques, telles que les **Generative Adversarial Networks (GAN)** et les **Variational Autoencoders (VAE)**.

Pour pallier les limites de ces dernières, des techniques basées sur le **score matching** ont été proposées. Notamment, le *Denoising Score Matching* (DSM) et ses variantes, telles que le *Sliced Score Matching* (SSM), offrent des solutions prometteuses en estimant directement le score, qui "capture" les variations des données. Ces approches, combinées au renversement temporel — comme démontré dans [17] — il devient possible de générer des échantillons de qualité.

Dans le cadre de notre projet, nous avons choisi d'appliquer ces méthodes pour générer en $\mathbb{R}^2$, telles qu'un anneau construit à partir de distributions gaussiennes et d'autres données plus complexes(MNIST). L'utilisation d'une mesure d'évaluation telle que la Fréchet Inception Distance (FID) nous permet de quantifier la qualité des échantillons générés, et de comparer objectivement notre approche à d'autres modèles existants.

Les résultats préliminaires indiquent que l'approche basée sur le score matching, combinée au renversement temporel, permet de générer des structures géométriques précises. Néanmoins, quelques pistes d'amélioration subsistent.

Limites mathématiques

Sur le plan théorique, notre approche repose sur des hypothèses fortes pour assurer l'existence, l'unicité et la régularité de la solution de l'équation différentielle stochastique.

L'estimation du score nécessite que la densité $p(x,t)$ soit *suffisamment lisse et strictement positive* pour tout X. Ces hypothèses sont cruciales pour garantir la convergence du schéma de renversement temporel et l'exactitude de l'estimation du score, comme détaillé dans [17]. Toutefois, elles limitent l'applicabilité de la méthode à des contextes où ces conditions sont remplies, ce qui peut ne pas être le cas pour certaines données.

Limites numériques

D'un point de vue numérique, plusieurs difficultés se posent :

— **Discrétisation et Erreurs Numériques :** La discrétisation de l'EDS par le schéma d'Euler-Maruyama introduit une erreur de discrétisation qui dépend du pas de temps Δt. Afin d'obtenir une bonne approximation, il est souvent nécessaire d'utiliser un pas de temps très petit, ce qui augmente la complexité computationnelle et le temps de calcul [11].

— **Estimation du Score :** Les méthodes telles que le Denoising Score Matching (DSM) et le Sliced Score Matching (SSM) requièrent l'optimisation de réseaux de neurones

complexes pour estimer le score. Cette optimisation est coûteuse en grande dimension, et peut être sensible aux choix d'hyperparamètres.

— **Stabilité Numérique :** Le renversement temporel d'un processus de diffusion peut être numériquement instable, surtout si la densité $p(x, t)$ varie fortement. Des schémas d'intégration plus sophistiqués, comme des méthodes de Runge-Kutta stochastiques, pourraient être envisagés.

Annexes

Architecture Unet avec time embedding

Les réseaux de neurones que nous avons utilisés pour prédire le score ou le bruit (en fonction de la méthode utilisée) sont des U-net. Le U-net est un réseau qui, contrairement aux CNN classiques, renvoie un tenseur de même hauteur et largeur que l'image initiale. Il a été initialement développé pour la segmentation d'images biomédicales au département d'informatique de l'université de Fribourg [14]. Nous rappelons rapidement son architecture à l'aide du schéma ci-dessous :

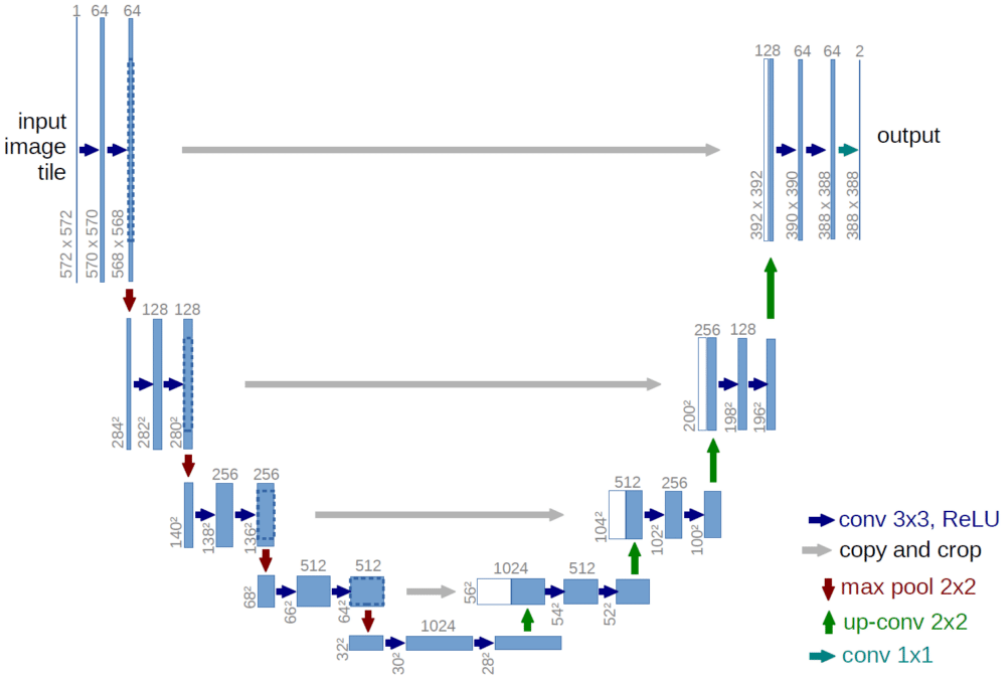


FIGURE 4.1 – Architecture d'un réseau U-net

Comme vous pouvez le voir, il s'agit d'un réseau possédant une partie "encodeur" et "décodeur". Dans la partie "encodeur", on applique des convolutions successives puis on fait grandement baisser la dimension de l'image à l'aide de max poolings. On réitère ces étapes jusqu'au bottleneck (là où la dimension de l'image est minimale, en bas du schéma). Puis, dans la partie "décodeur", on applique des up-convolutions pour faire augmenter la dimension, on concatène les canaux de l'image avec ceux qu'on avait dans la partie "encodage" de dimension similaire, on applique des convolutions puis on réitère jusqu'à obtenir une image de même taille que l'image initiale.

Cependant, le réseau U-net que nous utilisons dans notre projet diffère de celui ci-dessus car il prend aussi en entrée une valeur de temps t. Pour traiter cette valeur, nous la transformons en time-embedding de la même manière que dans les modèles de transformers [2]. C'est-à-dire que nous associons à chaque position temporelle un vecteur dans un espace de dimension arbitraire que nous noterons d ici. Pour i allant de 0 à $d/2-1$, l'expression du vecteur d'embedding de la position t est le suivant :

$$\begin{cases} P[t,2i] &= \sin(\frac{t}{10000^{\frac{2i}{d}}}) \\ P[t,2i+1] &= \cos(\frac{t}{10000^{\frac{2i}{d}}}) \end{cases}$$

Ou de manière équivalente, pour k allant de 0 à $d-1$, on a :

$$\begin{cases} P[t,k] &= \sin(\frac{t}{10000^{\frac{k}{d}}}) & \text{si k paire} \\ P[t,k] &= \cos(\frac{t}{10000^{\frac{k-1}{d}}}) & \text{si k impaire} \end{cases}$$

Le choix de la valeur 10000 est une recommandation de l'article *Attention Is All You Need*. [2]. Pour une image et son temps de bruitage t donnés, dans chaque bloc du U-net (c'est à dire chaque ensemble de couches successives délimité par 2 max poolings, 2 up-convolutions ou ceux se trouvant aux extrémités) on applique à l'embedding de t plusieurs couches fully-connected et d'activation SiLU. Après ces transformations (propres à chaque bloc du U-net), le vecteur donné est de dimension égale aux nombres de canaux de l'image à l'entrée bloc. Ce vecteur est ensuite sommé avec l'image puis cette somme correspond à l'image qui sera soumise à toutes les transformations suivantes du U-net. Pour faire une telle somme avec une image ayant par définition une hauteur et une largeur de tailles supérieures à 1, pour un canal donné on retient la valeur unique de l'embedding qui sera la même sur chaque pixel. En d'autres termes, le time-embedding ne dépend pas de la hauteur et de la largeur.

Enfin, une autre différence avec les U-net classiques est que le notre applique au début de chaque bloc une couche de "GroupNorm", c'est-à-dire une normalisation sur les pixels et des "groupes" de canaux [20]. Par exemple, pour une image à 20 canaux, on calcule la moyenne et l'écart-type sur tous les pixels des 5 premiers canaux, ensuite sur les pixels du $6^{\text{ième}}$ au $10^{\text{ième}}$ canal, etc. Puis sur ces 5 premiers canaux on normalise par la moyenne et l'écart-type correspondant, et on fait la même chose pour les autres groupes de canaux.

Liens vers les codes utilisés dans le rapport

— **chapitre 2** : https://github.com/mahamat9/diffusion_model_M2DSUA/tree/main/mahamat/script

— **chapitre 3** : https://github.com/tgipteau/ddpm/tree/main

— **chapitre 4** : https://github.com/mahamat9/diffusion_model_M2DSUA/tree/main/arnaud

Bibliographie

[1] Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3) :313–326, 1982.

[2] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. Attention is all you need, 2017.

[3] Djalil Chafaï. Wasserstein distance between two gaussians. 2010.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. 2009.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[6] Haussmann, U. G. and Pardoux, É. Time reversal of diffusion processes. *The Annals of Probability*, 14(4) :1188–1205, 1986.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[8] Bioinformatics JKU. Fid. https://github.com/bioinf-jku/TTUR/tree/master/FID_vs_Inception_Score.

[9] Ioannis Karatzas and Steven E. Shreve. *Brownian Motion and Stochastic Calculus*. Springer, New York, 2nd edition, 1991.

[10] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2013.

[11] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer, 1992.

[12] Grigorios A. Pavliotis. *Stochastic Processes and Applications : Diffusion Processes, the Fokker-Planck and Langevin Equations*. Springer, 2014.

[13] QuantPie. Fokker planck equation derivation : Local volatility, ornstein uhlenbeck, and geometric brownian, 2019. YouTube video.

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Springer International Publishing*, pages 234–241, 2015.

[15] Ernest Ryu. Generative ai and foundation models, spring 2024.

[16] Yang Song. Diffusion and score-based generative models. YouTube video.

[17] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021.

[18] Wenpin Tang and Hanyang Zhao. Score-based diffusion models via stochastic differential equations – a technical tutorial, June 2024. Department of Industrial Engineering and Operations Research, Columbia University.

[19] Urbain Vaes. Strong convergence of the euler–maruyama method. Lectures W6.

[20] Yuxin Wu and Kaiming He. Group normalization. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.