Notes et commentaires au sujet des conférences de S. Mallat du Collège de France (2025)

Génération de données en IA par transport et débruitage

J.E Campagne *

Janv. 2025; rév. 11 mars 2025

*Si vous avez des remarques/suggestions veuillez les adresser à `jeaneric DOT campagne AT gmail DOT com`

Table des matières

1. Avant-propos

***Avertissement**: Dans la suite vous trouverez mes notes au style libre prises au fil de l'eau et remises en forme avec quelques commentaires ("ndje" ou bien sections dédiées). Il est clair que des erreurs peuvent s'être glissées et je m'en excuse par avance. Vous pouvez utiliser l'adresse mail donnée en page de garde pour me les adresser. Je vous souhaite une bonne lecture.*

Veuillez noter que sur le site web du Collège de France vous trouverez toutes les vidéos des cours, des séminaires ainsi que les notes de cours non seulement de cette année mais aussi des années précédentes [1].

Je tiens à remercier l'ensemble de l'équipe du Collège de France qui réalise et monte les vidéos sans lesquelles l'édition de ces notes serait rendue moins confortable.

Notez également que S. Mallat [2] donne en libre accès des chapitres de son livre *"A Wavelet Tour of Signal Processing"*, 3ème edition. ainsi que d'autres matériels sur son site de l'ENS.

Cette année 2025 c'est la huitième du cycle de la chaire de la Science des Données de S. Mallat, le thème en est: **Génération de données en IA par transport et débruitage**.

Enfin, dans le repository GitHub [3], j'ai mis des notebooks d'applications numériques illustrant le cours depuis 2022. Autant que faire se peut, les notebooks peuvent être exécutés sur Google Colab.

2. Séance du 15 Janv.

Durant cette séance Stéphane Mallat va nous brosser le programme du cours de l'année qui concerne les modèles génératifs qui sont au coeur de bien des applications que le public a appris à manipuler pour générer du texte, des images, des vidéos etc, pour le meilleurs comme pour le pire. Le point de vue du cours est le même que celui pris dès les

1. https://www.college-de-france.fr/chaire/stephane-mallat-sciences-des-donnees-chaire-statutaire/events
2. https://www.di.ens.fr/~mallat/CoursCollege.html
3. https://colab.research.google.com/github/jecampagne/cours_mallat_cdf

débuts du cycle des conférences: le *pourquoi*, et non le *comment*. C'est-à-dire que le cours n'est pas dédié à la mise en œuvre pratique de tel ou tel code d'architecture de réseaux de neurones, il y a d'autres endroits où vous trouverez cela, mais il s'attache à faire découvrir les fondements mathématiques de ces architectures, et par certains cotés quelles sont les limites de notre compréhension des performances de plus en plus étonnantes des modèles.

2.1 Quelques faits marquants de l'année écoulée

Pour commencer, S. Mallat nous fait un petit panorama de ce qu'il l'a marqué durant cette année dans le champ qui nous intéresse ici. Il est très impressionné par la rapidité, l'accumulation des recherches menées dans le domaine notamment grâce à l'accumulation des financements et l'attrait pour des chercheuses et chercheurs de qualité. Le domaine avance donc à grands pas.

Tout d'abord, il est remarquable que les prix Nobel de Chimie et de Physique 2024 aient couronné des réalisations reliées aux réseaux de neurones. Celui de Chimie récompense Demis Hassabis et John Jumper qui ont réussi grâce au modèle `Alphafold` de DeepMind [4] à prédire la structure tridimensionnelle de presque toutes les protéines connues. C'est un problème que l'on pensait irréalisable à partir de base données de séquences et surtout avec une telle précision. Bien entendu, cela a un impact considérable dans le domaine de la Chimie, de la Pharmacologie et aussi de la Médecine pour expliquer des maladies causées par des repliements anormaux de structures aminée, etc. Celui de Physique qui a plutôt surpris dans la communauté des physiciens, a été décerné à John J. Hopfield et Geoffrey E. Hinton pour leurs travaux pionniers dans les années 1980 à la base du développement marquant qu'il s'en est suivi dans le domaine de l'apprentissage automatique artificiel. Leurs deux contributions mentionnées par le comité Nobel sont *les réseaux de Hopfield et les Machines de Boltzmann*. S. Mallat fait remarquer que G. Hinton a eu un impact dans le domaine de l'apprentissage notamment concernant l'algorithme de base de la back-propagation [5], mais aussi sur toutes les architectures profondes. Quant à J. Hopfield, l'impact a été surtout dans le pont qu'il a tissé entre apprentissage neuronal et Physique Statistique [6]. Ce travail a été fondamental, et d'ailleurs on l'a déjà

4. Jumper et al., 2021, https://www.nature.com/articles/s41586-021-03819-2
5. Voir Cours 2019 Sec. 8
6. NDJE: Si vous voulez en savoir plus vous pouvez consulter une petite note que j'ai mise en ligne sur le

appréhendé dans les cours précédents [7], mais aussi, nous le verrons dans ce cours où la Physique Statistique est en arrière-plan.

Ensuite, concernant les grands modèles de langages (LLM) qui datent de 3 ans (déjà!) le développement récent qui retient l'attention concerne *les systèmes de preuves* en Mathématique. S. Mallat nous dit que l'avancée récente est la rencontre entre deux approches: la première par *induction* qui est utilisée classiquement en Machine Learning (ML) en partant des données comme dans le cas des LLM, et la seconde par *déduction* qui est celle utilisée par les codes de vérifications de preuves informatiques (`Lean` [8] ou `Coq` [9]). Donc, dans les codes "mixtes", une production de preuves est effectuée par les LLM suivi d'un apprentissage par renforcement qui est garanti par le système de preuves. Cette double approche est mise en oeuvre par exemple dans les codes `AlphaProof` et `AlphaGeometry` de DeepMind [10]. Ces modèles ont remporté la médaille d'argent des Olympiades de Mathématiques, et nul doute que la médaille d'or sera dans leur escarcelle bientôt si ce n'est déjà fait à l'heure où sont écrites ces lignes. Au-delàÒ de ces challenges, S. Mallat nous dit que les stratégies de preuve se sont considérablement améliorées et le domaine est très actif.

Un autre domaine qui évolue très rapidement est celui de la *robotique* et selon S. Mallat c'est la révolution à venir après les LLM. La raison en est que dès que l'on va vouloir acquérir l'ensemble des données des capteurs communicants, là on va explorer un champ considérable qui va bien au-delà d'Internet actuel. Et donc, les systèmes d'interprétations vont devoir agréger des données très hétérogènes par nature. Une grosse difficulté de la robotique pour réaliser des tâches qui nous paraissent simples comme la préhension, marcher dans des terrains chaotiques, et qu'il faut beaucoup d'information temps réel, et d'une certaine façon, il faut des données labellisées. Or, on ne peut pas simplement envoyer le robot dans un environnement inconnu, car la quantité de données qu'il faudrait pour l'apprentissage est beaucoup trop grande à l'heure actuelle. On a alors recours à des *simulations* dans lesquels on plonge ces systèmes neuronaux. Et donc, tout repose sur la possibilité de simuler des environnements complexes de plus en plus réalistes.

---

repository Github de ces courshttps://github.com/jecampagne/cours_mallat_cdf/tree/main/Cours2025

7. Voir par ex. Cours de 2023 dédié à l'Entropie
8. https://lean-lang.org/
9. https://coq.inria.fr/
10. Voir article https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/

Le dernier domaine qui a marqué cette année n'est pas sans rapport avec le précédent, car il s'agit de la possibilité des simulations par IA en Physique. Notamment, la météorologie et climatologie, ont vu l'apparition de modèles tels que `GraphCast` et `GenCast` de DeepMind [11]. `GraphCast` effectue des prédictions à 10 jours et est tout à fait compétitif par rapport aux modèles classiques de résolution des équations différentielles. `GenCast` est plutôt probabiliste basé sur des modèles par *diffusion* que nous étudierons cette année, et génère des prédictions sur une quinzaine de jours. Il y d'autres approches qui allient les contraintes physiques et ML comme `NeuralGCM` de Google [12]: ce code fait des prédictions à 2-15 jours et reproduit les températures sur une période de 40 ans passée de manière plus précise que les modèles atmosphériques traditionnels.

Bien entendu, il y a d'autres codes disponibles, mais on remarque que tous ces systèmes sont produits par des sociétés privées avec des capitaux conséquents pour pouvoir disposer de capacité de calcul non moins conséquentes. NDJE: Notons au passage la généralisation de JAX [13] comme librairie de développement, et vous noterez qu'un certain nombre des notebooks que j'ai mis à votre disposition sont écrits dans ce langage [14].

2.2 Panorama de la génération par IA

L'IA générative fait partie du champ de l'apprentissage statistique avec à la base la construction de modèles à partir des données. Au cours des années précédentes, nous avons vu que dans ce domaine, on distingue:

— Les **problèmes supervisés** où l'on a une donnée $x \in \mathbb{R}^d$ (d très grand en pratique comme des images, des sons, des champs, etc), et l'enjeu est de répondre à une question y (entière ou continue). La stratégie est d'utiliser une base de données d'entrainement constituée de couple $\{x_i, y_i\}_{i \leq n}$. La difficulté réside tout d'abord dans la constitution de cette base de données labellisées qui doit être suffisamment peuplée et non biaisée.

11. Voir article https://deepmind.google/discover/blog/graphcast-ai-model-for-faster-and-more-accurate-global-weather-fo et les modèles https://github.com/google-deepmind/graphcast

12. Voir article https://research.google/blog/fast-accurate-climate-modeling-with-neuralgcm/

13. https://jax.readthedocs.io/en/latest/

14. Pour plus d'information, je mets à votre disposition des notebooks dédiés https://github.com/jecampagne/JaxTutos dont celui-ci pour commencer `JAX_Cophy_tuto.ipynb`

— Les **problèmes non supervisés** où l'enjeu cette fois est de construire un modèle des données x sous forme d'une distribution de probabilité (*pdf*) $p(x)$ qui nous renseigne où sont concentrées les données dans le grand espace $\mathbb{R}^d$. La stratégie ici est également d'utiliser une base d'entrainement, mais non labellisée, constituée d'échantillons $\{x_i\}_{i\leq n}$ que l'on suppose **indépendants identiquement distribués** (*iid*) (hypothèse fondamentale). À l'intérieur de cette catégorie de problèmes, on distingue la sous-catégorie suivante.

— Les **problèmes auto-supervisés** dont les LLM font partie, où l'on définit des labels y_i à partir des échantillons de la base de données $\{x_i\}_{i\leq n}$. Typiquement, pour un modèle de langage, à partir d'une phrase dont tous les mots sont connus, on en cache un pour le faire deviner au modèle, puis à partir d'un paragraphe on cache une phrase, etc. On peut bien entendu étendre ce type de masquage sur des images et cela se généralise à d'autres types de données.

Ce qui va plutôt nous intéresser durant le cours concerne les catégories des problèmes non supervisés et auto-supervisés. Pourquoi le problème de découvrir la *pdf* $p(x)$ sous-jacente à des données est-il difficile? et Pourquoi a-t'il été révolutionné par l'IA?

2.3 Retour sur la malédiction de la dimension

Ce sujet a été maintes fois abordé dans les cours, et cela dès le premier datant de 2017-18, mais il est toujours bon de prendre conscience pourquoi déterminer la pdf $p(x)$ est un problème difficile *a priori*[15]. Ce que l'on cherche à déterminer c'est la localisation des données qui s'agrègent sur une surface de dimension d' plongée dans l'espace de dimension d (Fig. 1).

Le problème est facile si la dimension d' est très petite même si d est grand par ailleurs. On peut penser par exemple à un système robotique à base de rotules où potentiellement d peut être très grand selon la digitalisation de l'espace exploré, mais en sous-jacent le nombre de degrés de liberté qui fixe d' est quant à lui de l'ordre de la vingtaine. Le problème devient alors abordable. Le point est que les problèmes auxquels nous allons nous confronter ne sont pas du tout de cette nature: remarquons par exemple une

15. NDJE. A *posteriori*, on se demande s'il n'y a pas des formes de régularités sous-jacentes qui permettent que les réseaux de neurones arrivent à obtenir $p(x)$. C'est par exemple traité dans le cours de 2021

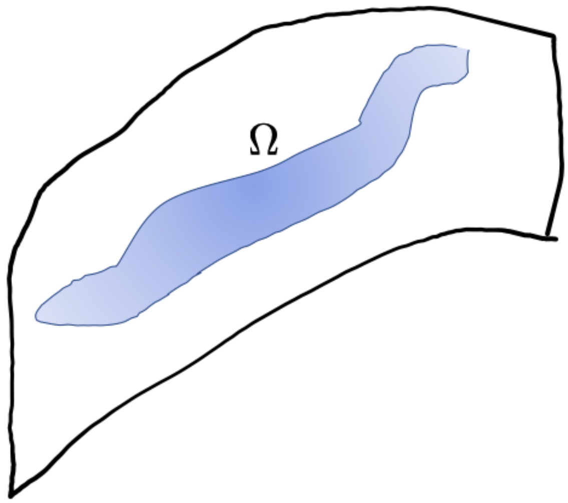


FIGURE 1 – Schématisation de la localisation des données sur une surface de dimension d' plongée dans l'espace de dimension d.

image d'un intérieur de maison, ne serait-ce que la texture des matériaux introduit une complexité qui fait exploser le nombre de degrés de liberté et donc d' est sans doute très grand même si plus petit que d. Typiquement, S. Mallat nous dit que d' est de l'ordre de $O(10^{3-4})$ dans une image où $d = 10^6$. Donc, **la variété sur laquelle se concentre les données est en pratique de grande dimension**. Cependant, si d' est grand ce qui nous mettrait en difficulté, il se peut que $p(x)$ comporte des régularités. C'est ce coin qui va nous servir pour construire des modèles et atteindre notre but.

Prenons par exemple le cas d'un bruit blanc gaussien (bbg), c'est une probabilité de d variables indécentes, dont chacune d'elles est distribuée selon une loi gaussienne, par exemple[16] $x_i \sim \mathcal{N}(0, \sigma^2/d)$. Ainsi

$$p(x_1, x_2, \ldots, x_d) = \prod_{i=1}^{d} p_i(x_i) \propto \prod_{i=1}^{d} \exp\left\{-\frac{x_i^2 d}{2\sigma^2}\right\} = \exp\left\{-\frac{\sum_{i=1}^{d} x_i^2 d}{2\sigma^2}\right\} = \exp\left\{-\frac{\|x\|^2 d}{2\sigma^2}\right\} \tag{1}$$

Or, $z = \sum_{i=1}^{d} x_i^2$ est une variable aléatoire. Si les x_i sont iid selon $\mathcal{N}(0,1)$, on obtiendrait

16. NDJE. je reprends un argument développé dans le cours de 2024 où la variance est distribuée uniformément sur les d variables.

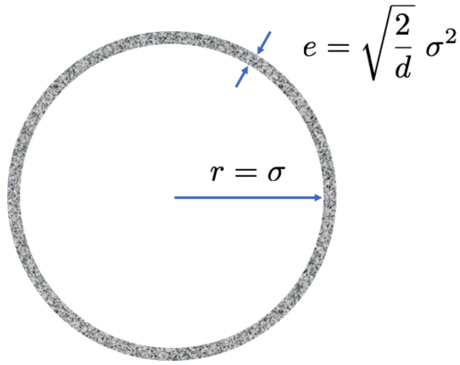


FIGURE 2 – Localisation de $\|x\|^2$ si toutes les composantes sont *iid* $x_i \sim \mathcal{N}(0, \sigma^2/d)$ en dimension d.

que $p(z)$ est la distribution du χ^2 à d degrés de liberté [17] dont l'espérance est $\mathbb{E}[z] = d$. Par changement de variables, on obtient alors

$$\|x\|^2 = \sum_{i=1}^{d} x_i^2 \xrightarrow[d\to\infty]{} \sigma^2 \tag{2}$$

et de même la variance du χ^2 à d degrés est $2d$, donc

$$Var(\|x\|^2) \xrightarrow[d\to\infty]{} \frac{2\sigma^4}{d} \tag{3}$$

Ainsi la localisation des échantillons d'un bruit blanc gaussien se fait sur une couche sphérique dont l'épaisseur tend vers 0 quand d augmente. On peut donc voir en grande dimension cette localisation comme une variété sphérique (Fig. 2) de dimension égale à celle de l'espace moins 1 unité donc grande, mais le seul degré de liberté est la variance σ^2.

Donc, la stratégie va être de chercher des paramétrisations à petit nombre de degrés de liberté et qui pour autant permettront de décrire des surfaces de grande dimension afin de correspondre au mieux aux cas pratiques rencontrés. Bien entendu, un *bbg* n'est pas très structurant si l'on peut dire, mais on peut penser à paramétriser des matrices de covariances, et à utiliser des modèles plus sophistiqués comme on va le voir. Le cœur

17. NDJE. la loi $\chi^2(d)$ est $(1/2)^{d/2}/\Gamma(d/2) z^{d/2-1} e^{-z/2}$.

du problème sera de choisir une paramétrisation adéquate et de l'apprendre à partir des échantillons. Quelle est la difficulté dans ce contexte?

Imaginons que pour choisir notre paramétrisation, nous prenions une hypothèse assez simple donnée par une régularité de type Lipschitzienne à savoir qu'elle est dérivable presque partout, soit $\forall x, y$ dans le domaine de définition, il existe une constante K telle que

$$|p(x) - p(y)| \leq K\|x - y\| \tag{4}$$

Est-ce une hypothèse suffisante pour pouvoir estimer $p(x)$? Si tel est le cas, la *pdf* est régulière, et si $x \in [0,1]^d$, il suffit d'échantillonner l'espace avec un intervalle ε assez fin pour pouvoir effectuer une sorte d'interpolation pour connaître $p(x)$ dans tout l'espace. Or, si l'on veut garantir que x soit à une distance inférieure à ε de n'importe quel x_i, alors on peut se convaincre facilement qu'il faut au moins n échantillons tels que (nb. on utilisera la dimension d comme celle de la variété/surface des x pour simplifier l'écriture)

$$n \geq \varepsilon^{-d} \tag{5}$$

Remarquons que si l'on procède à un histogramme en découpant l'espace en petite boite de taille ε^d, il faut également plusieurs échantillons par boite pour au moins faire une moyenne ayant un sens. Ainsi, pour $\varepsilon = 1/10$ et d de l'ordre de quelques milliers, on dépasse largement le nombre d'atomes de l'Univers. Il s'agit de la malédiction de la dimensionnalité.

Cela nous orienterait vers l'usage d'hypothèses bien plus fortes que Lipschitz, donc des régularités bien plus fortes pour $p(x)$. Or, jusqu'à "récemment", nul pensait que qu'il était possible de concevoir des distributions de probabilités de visages, de chambres à coucher, de protéines, etc. Ce que nous dit S. Mallat, c'est qu'avant les réseaux de neurones essentiellement ce sont les modèles gaussiens, voire des versions plus sophistiquées qui étaient utilisés, mais malgré tout, ils restaient très confinés à des classes de processus assez simples (Voir Cours 2023). **Ce domaine a réellement changé du tout au tout avec les modèles génératifs basés sur des réseaux de neurones.**

Par la suite S. Mallat effectue un tour d'horizon introductif aux modèles génératifs.

2.4 Les modèles auto-régressifs LLM

Une stratégie de modélisation est au cœur des LLM: ce sont des modèles auto-régressifs. Les données sont des *séquences*, donc il y a un ordre total, et chacune des données est *quantifiée* (*token*) c'est-à-dire que la donnée ne prend qu'une valeur dans un ensemble fini. Ainsi, si la valeur d'un *token* est représentée par x_i, on peut alors considérer la série ordonnée $\{x_1, x_2, \ldots, x_{t-1}\}$ et tenter de prédire x_t où t peut matérialiser un temps discret. Si par la suite l'on dispose de la *pdf* conditionnelle $p(x_t|x_1, x_2, \ldots, x_{t-1})$, alors on peut produire des échantillons x_t connaissant les *tokens* passés. En sous-jacent, on utilise la décomposition suivante en utilisant les probabilités conditionnelles de complexités croissantes:

$$p(x_1, \ldots, x_T) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots = p(x_1) \prod_{t=2}^{T} p(x_t|x_1, \ldots, x_{t-1}) \tag{6}$$

Notons au passage que l'hypothèse de Markov simplifie le problème (Cours 2023) en requérant que $p(x_t|x_1, \ldots, x_{t-1}) = p(x_t|x_{t-1})$, c'est-à-dire que si $t-1$ est le présent, il suffit pour prédire l'avenir. Mais d'une manière générale, il faut pouvoir modéliser toutes les probabilités conditionnelles $p(x_t|x_1, \ldots, x_{t-1})$. C'est là où les **Transformer**, introduits en 2017[18], ont pu montrer leur capacité à modéliser des dépendances à longue portée. Notez que la prise en compte ou non de la négation en début de phrase peut influer grandement le sens de celle-ci: *"Ne pas prendre en compte la dépendance à grande échelle est fondamentale"*. Et ne parlons pas des virgules qui font passer des nuits blanches aux diplomates écrivant des traités de toutes sortes. Or, le problème de ces dépendances à longue portée aurait pu être insoluble face la malédiction de la dimensionnalité, mais c'est aussi l'astuce du mécanisme d'*attention* qui permet de résoudre le problème.

Nous n'étudierons pas cette année ce type de problèmes, car les données que nous allons rencontrer sont *multi-dimensionnelles*.

18. Ashish Vaswani et al. 2017, *Attention Is All You Need*, Advances in neural information processing systems, page 5998–6008. https://arxiv.org/abs/1706.03762

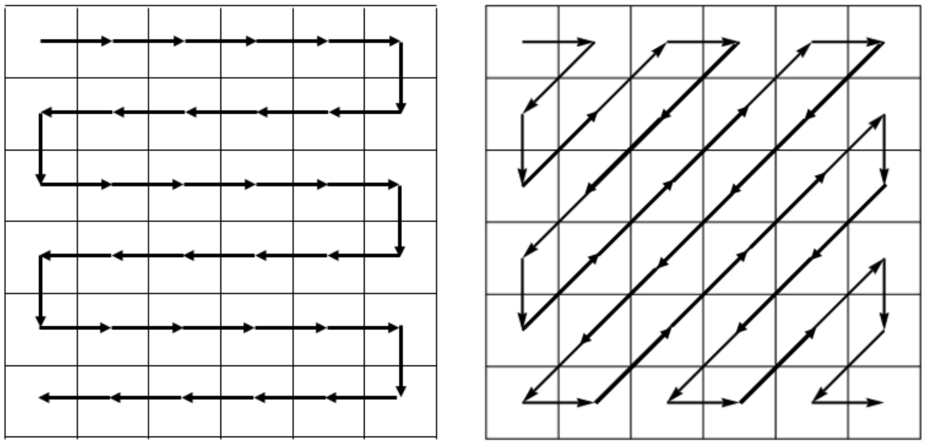


Figure 3 – Deux façons d'indexer les pixels d'une image.

2.5 Données multi-diemnsionnelles

Le cadre des LLM présenté ci-dessus n'est pas bien adapté aux cas de données multi-diemnsionnelles présentent dans des problèmes de Physique, d'analyses d'images, etc. Prenons une image pour illustrer le propos: il n'y a pas de manière naturelle pour organiser les pixels de manière séquentielle. Voyons les deux exemples d'indexage de la figure 3. Si l'on veut rendre compte d'interaction entre un pixel i et ses 8 proches voisins sur la grille 2D, il y a clairement un problème, car certains des pixels seront indexés dans la séquence selon $i-1$ et $i+1$ ce qui rendrait compte d'un lien causal direct, mais d'autres pixels auront des index très délocalisés. On introduit des liens de causalité à grande échelle d'une manière artificielle. Finalement, lors de la modélisation des probabilités, on va introduire une dépendance selon le schéma de cheminement unidimensionnel utilisé sans pour autant user de la topologie naturelle de la grille 2D. Or, cette topologie est très importante. Ceci dit des codes existent qui exploitent les descriptions unidimensionnelles, mais ils n'atteignent pas les meilleurs résultats.

Notre approche sera d'utiliser au mieux toute l'information multi-diemnsionnelle du problème.

2.6 Modèles probabilistes avec énergie de Gibbs

Ce type de modélisation s'inscrit dans le schéma de R. Fisher élaboré au début des années 1920, et nous l'avons abordé lors des Cours de 2022 à 2024. Ce sont des modèles à la base de la Physique Statistique, où l'idée pour apprendre $p(x)$ est de se donner une famille paramétrée de *pdf* $\{p_\theta(x)\}_\theta$ suffisamment "large" pour pouvoir trouver une valeur θ^* du paramètre qui permet d'ajuster au mieux $p(x)$. Le choix de la famille est bien entendu crucial. Si l'on considère que $p(x)$ n'est pas nulle, le log est alors défini et la forme exponentielle est un choix classique

$$p_\theta(x) = Z_\theta^{-1} e^{-U_\theta(x)} \tag{7}$$

La modélisation revient alors à trouver **la paramétrisation de l'énergie** $U_\theta(x)$ (par analogie à la Physique) adaptée au problème. Les événements x les plus probables sont ceux qui minimisent l'énergie (soit U_θ). C'est alors que les *a priori* (ex. symétries, invariants, etc) ou l'information en général sur le problème, rentrent en jeu pour introduire de la structure dans la paramétrisation $U_\theta(x)$. Cependant, en ML qui traite par exemple de la génération de visages, l'on a pas vraiment d'information *a priori* contrairement à la Physique qui est guidée par des Principes. Donc, en ML, on part avec un handicap.

Ceci dit une fois le choix fait d'une paramétrisation, on va chercher à **minimiser la métrique de Kullback-Leibler** [19] entre $p_\theta(x)$ et $p(x)$. Cette minimisation est équivalente dans le schéma de Fisher, à utiliser le **Maximum de Vraisemblance** afin de trouver θ^*. Les x sont bien localisés là où il y a une plus grande probabilité, et l'ensemble typique où se localise x est une notion de Cl. Shannon a introduit dans les années 1940 (Voir Cours 2022). Cette phase est un problème d'**optimisation**.

Dans le cas gaussien $U_\theta(x)$ est convexe, on a aucun problème, mais cela **devient très compliqué quand $U_\theta(x)$ n'est pas convexe**. Le point délicat par exemple dans un modèle à deux minima de même probabilité, est que la génération d'échantillons x doit rendre compte que x peut se trouver dans l'un ou l'autre des minima d'énergie, car ils rendent compte différentes configurations. Certes, si on se contentait de générer que des x correspondant à un seul minimum, on obtiendrait de bons échantillons, mais on ne rendrait pas compte de la diversité. Pensez par exemple à la génération de visages où l'on

19. Voir par ex. Cours 2022 Sec. 6.3, Cours 2024 Sec. 5.2

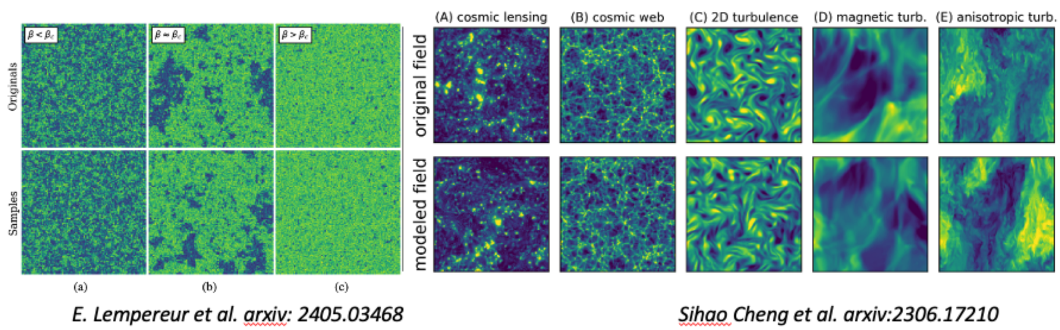


FIGURE 4 – Exemple de générations d'images de champs de Physiques.

ne produirait que ceux d'un seul genre. Ce problème de rendre compte de la diversité, et donc de **la généralisation, est le problème clé que l'on rencontre dans le domaine**.

Nous résumerons à la séance suivante ces techniques à base de chaines de Markov et Monte Carlo. Ces techniques permettent de modéliser des champs tels que (Fig. 4): des champs en cosmologie, de turbulence et ϕ^4 (ou Ising). Dans ce cas x est l'état du système à l'équilibre et la probabilité est donnée par la forme exponentielle citée, il s'agit de la distribution de Gibbs. Ces problèmes sont "relativement classiques" bien compris, car ils relèvent de la catégorie de **champs ergodiques** (ex. Cours 2023).

Cependant, cette hypothèse d'ergodicité n'est pas adaptée aux cas de génération d'images de visages, vues d'intérieur de maison, etc (rappel: l'érgodicité implique une invariance par translation). Et pour pouvoir mettre en œuvre des modèles génératifs capables de réaliser des images comme sur la figure 5, il va falloir opter pour autre chose.

2.7 Generative Adversarial Networks (GAN)

Les GAN ont été introduits par Goodfellow et al. [20] en 2014. Nous les avons évoqués durant le Cours de 2019 Sec. 2.8., et le schéma à l'origine est donné sur la figure 6. Deux réseaux agissent en adversaires: le réseau génératif (G) qui par exemple à partir d'un bruit

20. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Advances in Neural Information Processing Systems, pages 2672–2680. https://arxiv.org/pdf/1406.2661

Figure 5 – Exemple de générations de 2x6 images de visages: à gauche par le modèle de type Normalizing Flows (source: `Glow` de Diederik P. Kingma et Prafulla Dhariwal *arxiv:1807.03039*), à droite par un modèle de type Generative Adversarial Networks (source: Xin Wang et al. *arxiv:2202.07145*).

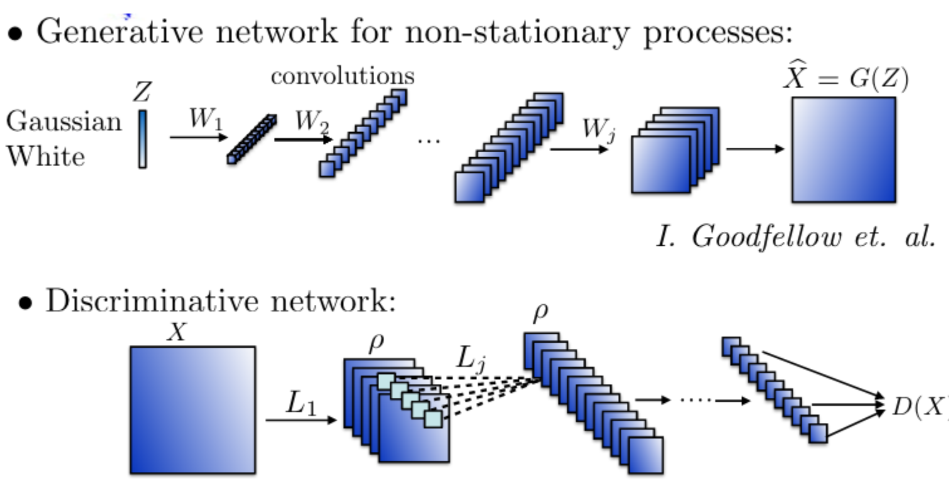


Figure 6 – Schéma d'un GAN de Goodfellow et al. avec deux réseaux adversaires: un générateur et un discriminant (classificateur).

blanc gaussien ($z \sim \pi(z) = \mathcal{N}(\mathbf{0}, \mathbf{1})$) produit de nouvelles images x, et le discriminant (D) fournit la probabilité que l'image qu'on lui présente est issue de la base de données d'entrainement (*real image*), plutôt qu'issue d'une génération $G(z)$ (*fake image*). Afin, d'optimiser les deux réseaux, à partir des images d'entraînement (de *pdf* p_{data}), l'idée est de résoudre un problème de `minmax` qui se présente comme suit:

$$\min_{\boldsymbol{G}} \max_{\boldsymbol{D}} \left\{ \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\log \boldsymbol{D}(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim \pi(\boldsymbol{z})}[\log(1 - \boldsymbol{D}(\boldsymbol{G}(\boldsymbol{z})))] \right\}, \tag{8}$$

Malgré les problèmes de mise en œuvre que nous allons mentionner ci-après, l'évolution des GANs a permis l'éclosion de générateurs dans bien des domaines: en Physique, dans le médical, la météo, etc. Une technique a par exemple utilisé non pas des images de bruit blanc, mais des images comme conditionnement. S. Mallat nous montre un exemple à l'écran d'images cérébrales issues d'un scanner "CT" et la prédiction à partir d'un GAN. De même, il nous présente un exemple en météorologie à partir d'images radar de précipitations [21] qui en 2021 était capable de faire des prédictions à 90 minutes sur des zones typiques de 1.3 km de rayon.

Le point délicat est que la minimisation n'est pas facile, car il s'agit d'un point selle. Et beaucoup de recherche vont élaborer des améliorations. Que se passe-t'il? En fait, dans un GAN, il n'y a rien qui force le générateur à explorer tout l'espace des possibles pour x. Il s'agit d'un défaut d'échantillonnage dit de "**mode collapse**" qui à avoir avec la non prise en compte de tous les minima évoqués plus avant. On obtient via $G(z)$ des échantillons tout à fait corrects, mais l'ensemble des x produits peut très bien ne représenter qu'une partie des modes possibles (manque de diversité, **problème de généralisation**). Pour palier ces problèmes, diverses techniques de régularisation sont mises à l'œuvre [22], mais finalement S. Mallat nous dit que le problème d'optimisation est assez mal maitrisé. **Le danger principal est de biaiser les générations et d'entrainer des formes de mémorisation de la base d'entrainement**. Tant que l'on se contente de "belles images" il n'y a rien de choquant, mais dès que l'on veut utiliser les échantillons pour des applications (ex. médicales, météorologiques) cela est très problématique, car il se produit des artefacts dont l'origine sont les biais de la base de données d'entrainement reproduits

21. NDJE. Sans doute tiré de https://www.nature.com/articles/s41586-021-03854-z.
22. Voir par exemple le modèle `light-weight-GAN de Liu B., Zhu Y., Song K., Elgammal A., 2021, in International Conference on Learning Representations. https://openreview.net/forum?id=1Fqg133qRaI`. Voir la Sec. 2 de l'article.

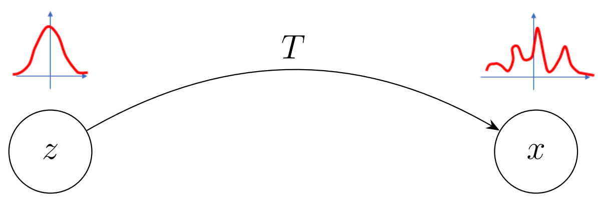


FIGURE 7 – Transport de la *pdf* des variables latentes z vers la *pdf* des données x.

par le générateur.

2.8 Modélisation par transport

En fait, ce que l'on a cherché à faire dans la section précédente, c'est de transformer une distribution $\pi(z)$ des variables (latentes) z ayant par exemple une pdf d'un bruit blanc, en une distribution des données x ($p(x)$). On parle alors de **transport** T tel que $p = T_{\#}\pi$ (Fig. 7). Donc, on peut se poser la question plus générale de l'estimation des transports: quelle classe de transports nous faut-il chercher? et comment guider notre choix?

Qui dit transport de probabilités, nous dit que l'on veut chercher des *transports optimaux* introduits par G. Monge en 1781 puis abordés par Léonid Kantorovich dans les années 1940 par la Théorie de la mesure. Le point est que **le transport T n'est pas unique** en général, on peut permuter n'importe quel couple (z_i, x_j) tout en préservant les *pdf* de départ et d'arrivée. Donc, il nous faut un principe qui nous fixe le choix. Si le domaine de x et z est le même (ex. une image de même dimension), on peut utiliser la fonction de coût $d(x, T(x)) \propto \|x - T(x))\|$ ou bien $\|x - T(x))\|^2$ qu'il faut alors minimiser.

Ceci dit, S. Mallat nous indique que les modèles basés sur ce type de transports ne sont pas performants à l'heure actuelle pour les problèmes en grande dimension. D'autres algorithmes ont été utilisés.

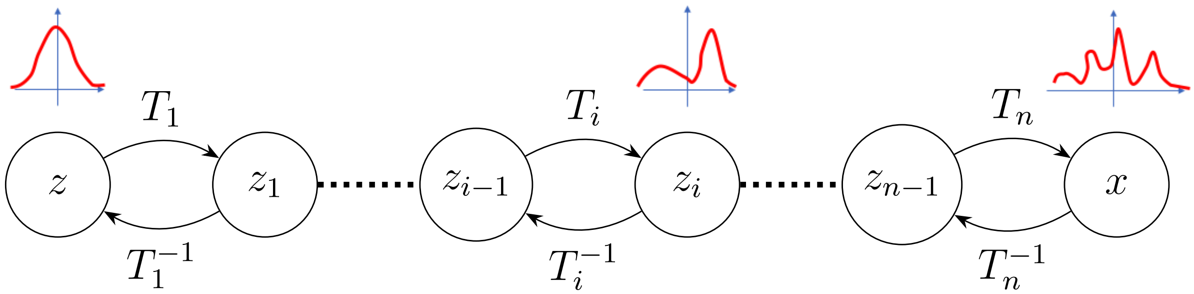


FIGURE 8 – Schematic Normalizing Flow process. On the top the *forward* or *generative* direction from a simple $\boldsymbol{z}$ distribution to a more complex one for $\boldsymbol{x}$. On the bottom the *backward* or *training* direction from complex to simple distributions.

2.8.1 Normalizing Flows (NF)

Un transport comme Normalizing Flow (ou Flow) T est défini comme un difféomorphisme (c'est-à-dire, un bijecteur) entre l'espace des données et un espace latent, tel que:

$$\boldsymbol{x} = T(\boldsymbol{z}) \quad \Leftrightarrow \quad \boldsymbol{z} = T^{-1}(\boldsymbol{x}). \tag{9}$$

Si la distribution de $\boldsymbol{z}$ est $\pi(\boldsymbol{z})$ (par exemple, $\mathcal{N}(\mathbf{0}, \mathbf{1})$), alors la relation suivante s'applique:

$$p(\boldsymbol{x}) = \pi(\boldsymbol{z})|\det J_T(\boldsymbol{z})|^{-1} = \pi(T^{-1}(\boldsymbol{x}))|\det J_{T^{-1}}(\boldsymbol{x})|, \tag{10}$$

où J_T et $J_{T^{-1}}$ sont les jacobiens des transformations T et T^{-1}, respectivement. Contrairement aux GANs, les dimensions des espaces latents et des données sont les mêmes par définition dans les modèles basés sur les flux.

Le flux T est généralement construit comme une composition de plusieurs flux individuels $\{T_i\}_{i<n}$ tels que:

$$T = T_1 \circ T_2 \circ \cdots \circ T_n \quad \Leftrightarrow \quad T^{-1} = T_n^{-1} \circ T_{n-1}^{-1} \circ \cdots \circ T_1^{-1}. \tag{11}$$

La figure 8 illustre une vue schématique de la direction *forward* ou *générative* et de la direction *backward/reverse* ou *training*. Si nous notons $\boldsymbol{z}_i = T_i(\boldsymbol{z}_{i-1})$ pour tout $i < n$ (en utilisant $\boldsymbol{z}_0 = \boldsymbol{z}$ et $\boldsymbol{z}_n = \boldsymbol{x}$), alors:

$$\log|\det J_T| = \sum_{i=0}^{n-1} \log|\det J_{T_i}(\boldsymbol{z}_{i-1})|. \tag{12}$$

Chaque bijecteur T_i est paramétré et on cherche la structure qui permet l'optimisation, où la pdf $p(x)$ maximise la vraisemblance des données qui sont dans la base de données. C'est une approche intéressante d'un point de vue mathématique qui fait un lien entre le transport optimal et les techniques de la section 2.6. Un modèle qui marche sur ce principe de flows est `Glow` dont des exemples de générations de visages sont présentés sur la figure 5. Mais S. Mallat nous dit qu'on a du mal à structurer ces bijecteurs, et il faut donc injecter beaucoup d'information *a priori*, et ces modèles restent limités pour des problèmes en très grandes dimensions.

2.8.2 Transport par Score-diffusion

C'est un sujet abordé lors de la dernière séance du Cours de 2024. En se reportant à l'équation 11, on se demande comment décomposer l'opérateur T? En fait, au lieu de prendre une décomposition discrète, on introduit une variable continue de "temps" et T peut alors être décomposé en grand nombre de transformations élémentaires "plus simples" où l'on va injecter de l'information. On se trouve alors dans le domaine des **équations différentielles ordinaires** (ODE) où x_t va être modifié selon une équation de type

$$\frac{dx_t}{dt} = v_t(x_t) \tag{13}$$

La pdf $p_t(x_t)$ est alors donnée par l'équation de Joseph Liouville qu'il a élaboré dans le cadre de la Physique hamiltonienne.

S. Mallat nous dit que ce type de transport est malgré tout un peu trop restreint, en particulier pour qu'il existe une solution à l'équation ci-dessus, il faut que la vitesse v_t soit Lipschitz or c'est une contrainte qu'il est difficile à obtenir numériquement. On va alors se tourner vers les **équations différentielles stochastiques** (SDE) où le transport est localement plus irrégulier. Dans ce cas, x_t est solution de

$$d\boldsymbol{x}_t = -\boldsymbol{x}_t\, dt + \sqrt{2}\, dW_t \quad t \in [0, T_{max}], \tag{14}$$

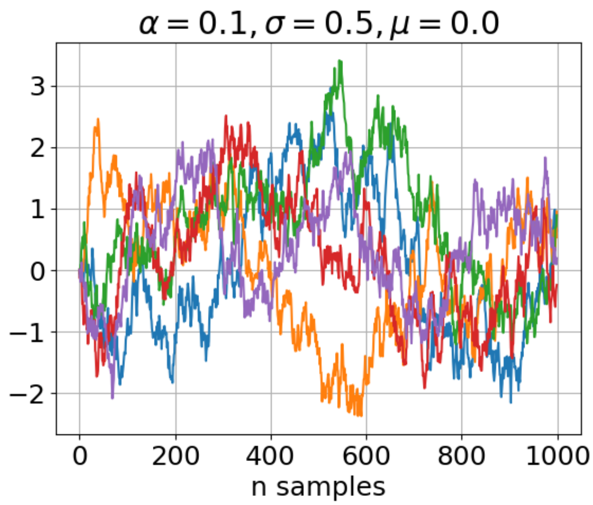


FIGURE 9 – Exemples de réalisations de transports selon l'équation différentielle stochastiques $dX = -\alpha X\ dt + \sigma\ dW$ avec $dW = \mathcal{N}(0, dt)$ ($\alpha = 0.1, \sigma = 0.2$).

où dW_t représente un processus de Wiener (mouvement brownien) et T_{max} représente la durée maximale du transport supposée suffisamment grande. À chaque étape (t_i, t_{i+1}) le transport n'est plus déterministe comme pour les Normalizing Flows, mais probabiliste [23] (Fig. 9). La *pdf* $p_t(x_t)$ est régie comme pour les ODE, par une équation différentielle, qui est une équation [24] de **Fokker-Planck** obtenue dans les années 1914-17 lors de l'étude du mouvement brownien.

En fait, ce point de vue d'un processus régit par une équation différentielle, va nous donner une technique très classique d'échantillonnage de probabilité. Faisons remarquer que si nous voulons que partant d'une *pdf* p_0 nous aboutissions à $p(x)$, c'est qu'il y a une convergence qui s'opère et d'une certaine manière $p(x)$ **doit être un point fixe de l'équation**. Nous verrons que si l'on paramétrise $p(x)$ selon une distribution de Gibbs (Eq. 7) alors la condition de stationnarité nous dit que la vitesse $v_t(x_t)$ (Eq. 13) doit satisfaire:

$$v_t(x) = \nabla_x \log p_t(x) = -\nabla U_t(x) \tag{15}$$

23. NDJE. Vous pouvez modifier le notebook `Ornstein_Uhlenbeck.ipynb` disponible sur Github https://github.com/jecampagne/cours_mallat_cdf dans le répertoire "cours2024".
24. Adriaan Fokker (1887-1972) et Max Planck (1858-1947)

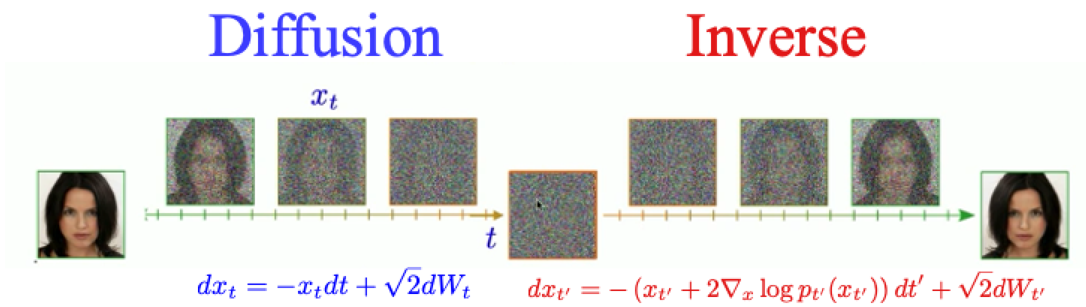


FIGURE 10 – Processus de Diffusion et Inverse permettant de déterminer le transport entre un bruit blanc et la *pdf* des données.

où $\nabla_x \log p(x)$ s'appelle le **score** [25], ce qui nous amène aux algorithmes de **Score Diffusion**. Tout cela nous donne une base de connaissance qui nous vient de la Physique Statistique.

Pour aller plus loin, nous allons remarquer que l'objectif est le suivant. Il nous faut découvrir un transport T qui à partir d'un bruit blanc nous donne la distribution sous-jacente des données. Mais en regardant le schéma 8, il nous parait alors qu'il serait plus simple d'identifier T comme $(T^{-1})^{-1}$. En fait, il est plus simple de passer de $p(x)$ à une distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$ définissant ainsi T^{-1}. C'est ce qui a motivé le travail de Yang Song et al. (2021, *arXiv:2011.13456*).

Ainsi, en partant d'un échantillon de la base de données qui est une réalisation de $p(x)$ par hypothèse, pour aboutir à une réalisation d'un pur bruit blanc gaussien, il suffit d'ajouter progressivement du bruit (transport "Diffusion" de la figure 10) (il y a une renormalisation à chaque étape pour que l'image d'origine disparaisse). Il s'agit de la réalisation d'un processus stochastique régit par l'**équation de diffusion d'Ornstein-Uhlenbeck** [26] qui est la plus simple et s'écrit

$$dx_t = -x_t dt + \sqrt{2}\, dW_t \tag{16}$$

Une fois que l'on a le chemin T^{-1}, il nous faut l'inverser. C'est tout à fait possible en

25. NDJE. Voir Cours 2024 Sec. 5.2 la remarque que je fais sur cette appellation, car il faut bien remarquer qu'il s'agit ici de considérer le gradient par rapport à x et non par rapport aux éventuels paramètres θ_t.
26. Leonard Salomon Ornstein (1880-1941) et George Eugene Uhlenbeck (1900-88) tous deux physiciens.

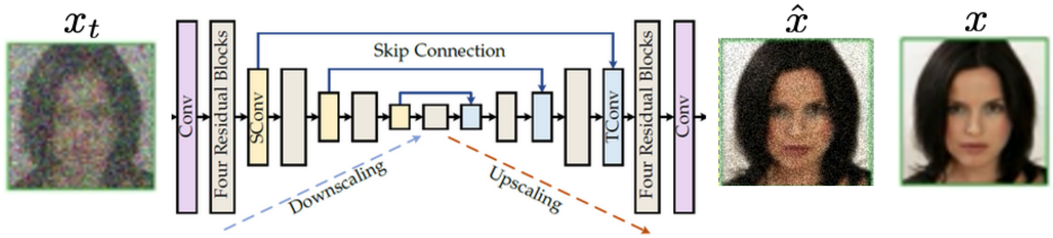


Figure 11 – Usage d'un réseau profond pour effectuer une tâche de débruitage. Voir par exemple Kai Zhang et al. (2021) *arXiv:2008.13751* (Sec. 3.1). A gauche: l'image x_t à débruiter; à droite: $\hat{x}$ résultat du réseau de neurones et x l'image telle que le réseau doit minimiser $\|x-\hat{x}\|^2$ et cela pour tous les échantillons de la base de données et les valeurs de la variance du bruit contenu dans les images bruitées.

utilisant une **équation de Langevin**[27] **amortie** qui s'écrit

$$dx_{t'} = -\left(x_{t'} + 2\nabla_x \log p_{t'}(x_{t'})\right) dt' + \sqrt{2}dW_{t'} \tag{17}$$

où apparait le score de la *pdf* à l'étape t'.

Si le schéma est mathématiquement clair, le point qui nous reste à résoudre est de pouvoir disposer du score à toutes les étapes du transport "Inverse" (Fig. 10). C'est *a priori* un problème difficile, car il nous faut apprendre d fonctions (pour le calcul du gradient) en grande dimension. C'est là où il y a eu une grosse surprise: les réseaux de neurones profonds permettent d'apprendre ces scores. Quelle est l'idée sous-jacente?

En définitive, il faut aller piocher dans un problème connexe qui traite du **débruitage d'un signal** x_t tel que

$$x_t = x + z \qquad z \sim \mathcal{N}(0, \sigma_t^2) \tag{18}$$

car en fait le transport Inverse (Fig. 10) effectue bien un débruitage pour retrouver un échantillon x. Or, qui dit débruitage dit trouver un estimateur $\hat{x}$ de x. Il y a alors un résultat standard des années 1950-60 qui stipule que chercher $\hat{x}$ tel qu'il réalise le problème de minimisation

$$\min \left[\mathbb{E}_{x\sim p_t}(\|\hat{x}-x\|^2)\right] \tag{19}$$

27. Paul Langevin (1872-1946)

Figure 12 – Exemple d'une génération d'image en utilisant l'interface de `ChatGPT` qui utilise `DALL-E`.

est équivalent à effectuer l'ascension de gradient suivante (en 1 step)

$$\hat{x} = x_t + \sigma^2 \nabla_x \log p_t(x_t). \tag{20}$$

Donc, si le problème de minimisation est équivalent à trouver $\hat{x}$, utilisons alors un réseau de neurones profond (Fig. 11) par exemple de type `U-Net`[28] , entrainons le en minimisant la loss quadratique (apprentissage supervisé), alors on aura $\hat{x}$ pour x_t, et donc on aura accès à $\nabla_x \log p_t(x_t)$ en inversant l'équation ci-dessus.

Maintenant, la question (fondamentale) qui se pose est de savoir si le réseau de neurones arrive à obtenir le débruiteur optimal? En fait, il semble y arriver et même très bien et cette technique est par exemple mise en œuvre dans les applications grand public telles que `DALL-E`, `Midjourney`, `StableDiffusion`, etc (Fig. 12). Comme on peut en juger la qualité des images est impressionnante.

Cependant, S. Mallat attire notre attention sur le fait qu'il existe toujours une question fondamentale sous-jacente: **les résultats sont-ils vraiment des échantillons de**

28. Olaf Ronneberger et al. (2015) *arXiv:1505.04597*.

$p(x)$, ou bien est-ce que par un jeu d'assemblage astucieux sont-ils des mélanges des images d'origine? Dans le second cas de figure, les images produites dépendent totalement de la base de données d'entrainement, on serait alors face à un problème de généralisation. Le but du cours va être de comprendre les concepts et propriétés mathématiques de ces algorithmes de génération, et cette **question de généralisation est omniprésente en apprentissage**.

Dans le cas des modèles génératifs, on obtient un estimateur de $p(x)$ obtenu à partir de la base d'entrainement, mais on considère que ce n'est pas un bon estimateur si "par malheur" il venait à varier si l'on changeait la base de données, fussent-elles toujours d'images de chambres à coucher ou de visages si on s'intéresse à l'un ou l'autre cas. Dans ce problème si l'estimateur est bon sa variance doit tendre vers 0 quand la taille de la base de données tend vers l'infini. Ce point a été testé dans l'article[29] de Kadkhodaie et al. (2024) que S. Mallat avait évoqué dans le Cours de 2024 (Sec. 9.4). Le point qui les a étonnés est qu'au fur et à mesure que la taille de la base d'entrainement augmente, **vers** $N = O(10^5)$ **(pour des images 80x80), une transition s'opère entre un comportement de type "mémorisation" et un comportement de "généralisation" qui semble donc indiqué que le modèle a appris** $p(x)$ (Fig. 13). Cependant, S. Mallat nous indique qu'il faut réaliser que pour des petites images, pour atteindre la généralisation, il a fallu beaucoup de données!

Maintenant, le résultat précédent montre que l'on peut atteindre une **petite variance** de l'estimateur, mais il faut également garantir que l'estimateur soit **non biaisé**. C'est-à-dire il faut s'assurer que l'on a bien convergé vers $p(x)$, or cela pose la question de l'obtention du "bon score", ou d'une manière équivalente il s'agit du problème **l'optimalité du débruitage**. Or, cela nous fait passer dans un chapitre classique des mathématiques à savoir l'**Analyse Harmonique**. NDJE. Ce sujet a été abordé durant le Cours de 2021. Dans le cas du débruitage qui a été étudié durant les années 1950 à 2000, on peut citer:

— Estimation linéaire: filtre de Wiener;

— Estimation non-linéaire[30] parcimonieuses: bases orthonormales;

— Bases d'ondelettes, curvelettes, bandelettes: optimalité?

29. Kadkhodaie Z., Guth F., Simoncelli E. P., Mallat S., 2024, in The Twelfth International Conference on Learning Representations. https://openreview.net/forum?id=ANvmVS2Yr0

30. NDJE. Voir les Sec. 3.1 et 3.2 du Cours de 2021 pour la distinction linéaire/non-linéaire.

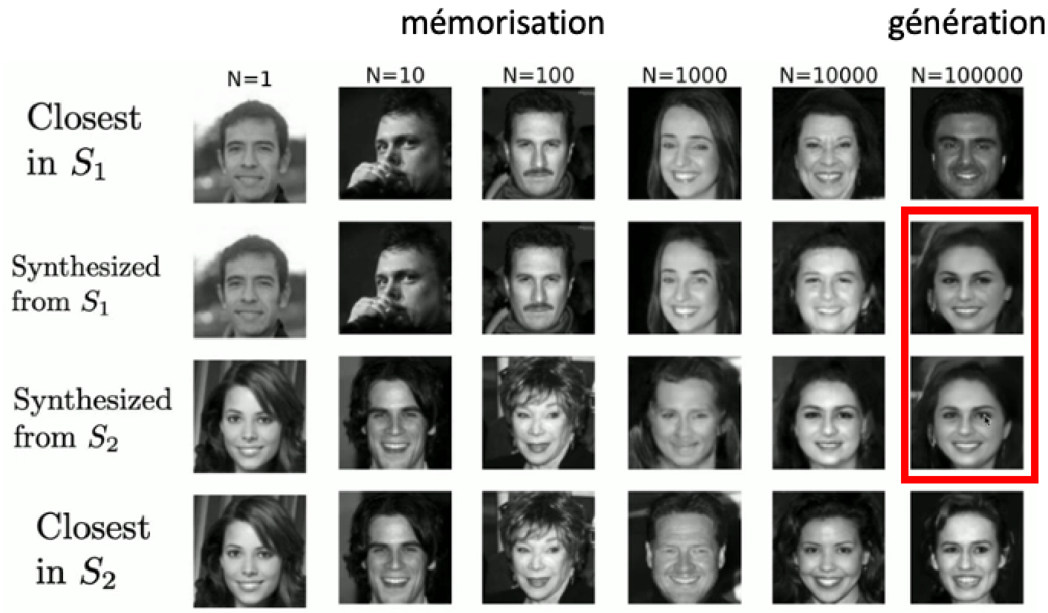


FIGURE 13 – Deux modèles génératifs S_1 et S_2 sont entrainés avec deux sous-lots issus de la même base de données d'images de visages (80×80 pixels) sans recouvrement et chacune de taille N. On donne ensuite la même image de bruit aux deux modèles afin de synthétiser une nouvelle image. On peut trouver l'image de la base de donnée qui approcherait au mieux cette image synthétisée. Tant que $N < O(10,000)$, S_1 et S_2 produisent des images différentes, et ressemblant d'autant plus à une image de la base de données que N est petit. Dès que $N = 100,000$, les images synthétisées par S_1 et S_2 sont d'une part (quasi-)identiques et d'autre par différentes des images les plus proches des deux bases de données. On assiste bien à une transition vers un modèle qui généralise bien: il a appris la distribution de probabilité $p(x)$.

Au passage, notez la diversité des sujets de math qui s'agrègent autour des modèles génératifs: transport optimal, équation différentielle stochastique, optimisation, et tiens l'Analyse Harmonique. Pourquoi? en gros quand on a un signal à débruiter la première idée qui vient et d'effectuer une séparation entre basses fréquences (plutôt composées par le signal) et hautes fréquences (plutôt composées par le bruit). Cette technique est celle de Wiener (filtrage dit linéaire) qui s'implémente en Fourier. Dans les années 1980, on a réalisé que l'on pouvait faire mieux en utilisant des opérateurs non-linéaires, avec au cœur le problème de la représentation des données, et la mise au point de bases parcimonieuses (l'énergie de la fonction à traiter est concentrée sur un petit nombre de coefficients). En fait, **les réseaux de neurones arrivent à faire beaucoup mieux, d'une certaine manière, ils s'adaptent mieux à la topologie du signal**, et un enjeu de la recherche et de comprendre pourquoi.

En regardant un peu de haut la recherche dans ce domaine, tout ce qui est en dehors du réseau de neurones, c'est-à-dire dès qu'il nous donne le score, les maths sont bien comprises. Donc, la chose à comprendre c'est ce que fait le réseau quand il apprend le score, et c'est que l'on va tenter de faire dans ce cours qui se trouve à la frontière de la recherche.

Si le temps le permet, nous irons jusqu'à voir comment on peut conditionner l'apprentissage par exemple en donnant une prescription c au générateur. Il s'agit d'un problème que l'on peut voir sous l'angle de Bayes:

$$p(x|c) \propto p(c|x)p(x) \tag{21}$$

où $p(c|x)$ c'est un classificateur et $p(x)$ est le générateur inconditionnel. Les séminaires de cette année porteront sur les sujets de générations par IA.

3. Séance du 22 Janv.

Durant cette séance, nous allons donc explorer des modèles génératifs qui pour rappel sont du ressort de l'apprentissage non supervisé, et qui à partir d'échantillons supposés *iid* $\{x_i\}_{i\leq n}$ veulent estimer directement ou indirectement la distribution de probabilité supposée sous-jacente $p(x)$, pour pouvoir l'échantillonner à nouveau. Dans ce cadre, nous avons vu à la dernière séance qu'il y a diverses approches pour concevoir de tels modèles génératifs. Ceux que nous allons traiter durant cette séance sont les modèles classiques à énergie de Gibbs (Sec.2.6) importants notamment en Physique, et les GANs (Sec.2.7) où à émerger la puissance des réseaux de neurones profonds dans ce contexte. Dans le cas des modèles "énergétiques" l'échantillonnage se fait à l'aide de chaines de Markov (Cours 2024) par des techniques de Monte Carlo (MCMC), c'est donc une méthode stochastique; dans le cas des GANs, il s'agit d'une méthode déterministe d'échantillonnage avec un réseau de neurones qui a été entrainé à minimiser une fonction de coût. Rappelons que les GANs par nature doivent réaliser une sorte d'équilibre entre le générateur et le discriminateur (équilibre de Nash) qui l'est difficile à maitriser, et donc sont apparues les méthodes que nous avons mentionnées à la précédente séance, à savoir les Normalizing Flows (Sec. 2.8.1) qui réalisent un transport discret de probabilité, et les modèles à base de score-diffusion (Sec. 2.8.2) qui opèrent quant à eux un transport continu avec au cœur un réseau débruiteur.

3.1 Modèles d'énergie de Gibbs

3.1.1 Typologie des paramétrisations

Comme déjà mentionné ces modèles ont déjà été introduits dans les cours de 2023 et 2024; mais nous y revenons cette année, car ils sont la base en Physique Statistique et on peut sans servir comme guide en ML. On définit une famille paramétrée de distributions $\{p_\theta\}_\theta$ telles que

$$p_\theta(x) = Z_\theta^{-1} e^{-U_\theta(x)}, \qquad Z_\theta = \int e^{-U_\theta(x)}\ dx \tag{22}$$

Pour mémoire la constante de normalisation Z_θ est la fonction de partitions de Gibbs à partir de laquelle l'on retrouve toutes les fonctions de la thermodynamique classique. La principale difficulté dans ce contexte est le choix du modèle, c'est-à-dire que choisir comme représentation de "l'énergie" $U_\theta(x)$?

Les modèles classiques bien étudiés en Math et Physique sont les **modèles gaussiens**, c'est-à-dire quadratique en x de la forme

$$U_\theta^{Gauss}(x) = \frac{1}{2}(x-\mu)^T\ \Sigma_\theta^{-1}\ (x-\mu) \qquad \text{(modèle gaussien)} \tag{23}$$

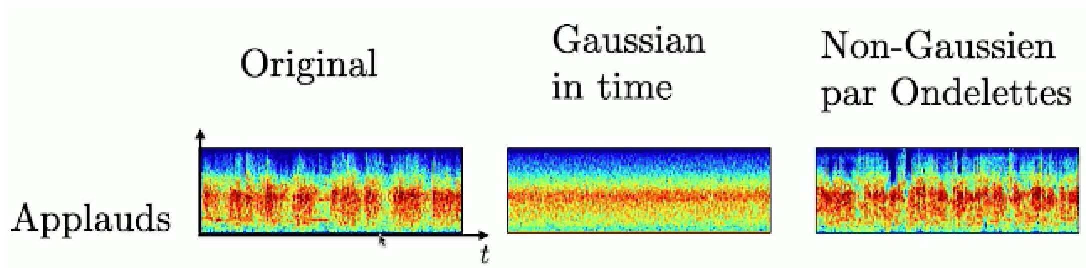


FIGURE 14 – Représentations Temps-Fréquence d'un applaudissement: le son original à gauche; au milieu la génération d'un modèle gaussien qui ne conserve que l'énergie originale à toutes les échelles/fréquences; à droite la génération par un modèle plus réaliste qui tient compte des corrélations entre échelles.

où Σ_θ est la matrice de covariance du système. La génération d'échantillons x est assez simple, car on peut diagonaliser Σ_θ et faire apparaitre des composantes indépendantes. Les ensembles typiques (Cours 2023) sont des ellipsoïdes de révolutions, dont les axes principaux centrés sur μ ont des directions données par les vecteurs propres de Σ_θ. Notons que dans ce cadre $Z_\theta = (2\pi)^{d/2}|\det(\Sigma_\theta)|^{1/2}$. Ces modèles gaussiens sont simples et le sont trop dans le contexte de modélisation de phénomènes intermittents. La figure 14 reprend un exemple du Cours de 2023 montrant l'**absence de structures** quand on utilise un modèle gaussien pour décrire une trame sonore présentant des intermittences. Pourtant, ces phénomènes transitoires sont omniprésents non seulement dans le domaine vocal, mais aussi dans le traitement d'image où les contours des objets sont des transitions de contraste. Ces modèles donc peuvent être utilisés en première intention, car ils sont aisés à mettre en œuvre, mais ils s'avèrent rapidement très limités dans les cas pratiques où il y a de la structuration.

Pour concevoir des modèles plus complexes, l'idée est venue naturellement de rajouter un **potentiel**, et dans ce cas les modèles les plus simples sont à potentiel scalaire:

$$U_\theta(x) = U_\theta^{Gauss}(x) + V_\theta(x) \tag{24}$$

Dans l'échelle de la complexité ont alors émergé les **modèles exponentiels** pour lesquels

$$U_\theta(x) = \Theta^T \Phi(x) = \sum_k \theta_k \phi_k(x) \tag{25}$$

où Θ est un vecteur de paramètres et $\Phi(x)$ est une famille de fonctions $\{\phi_k(x)\}_k$ qui nous faut choisir. Remarquons que les modèles gaussiens et à potentiels scalaires sont des exemples particuliers de ce type de modélisation. Il y a eu tout un travail à la fois en Physique et en Math pour comprendre quels sont les fonctions $\phi_k(x)$ les plus adaptées. Une idée est que si les modèles gaussiens sont des polynômes d'ordre 2, pourquoi ne pas introduire des polynômes d'ordres supérieurs. Si l'idée paraît séduisante, on peut montrer que plus le degré devient élevé plus l'estimation des paramètres Θ devient instable, et il faut énormément d'échantillons pour combattre la variance. Bien entendu, on peut choisir d'autres types de fonctions et dans ce cadre il y a une longue tradition en traitement du signal. Cependant, ce qui est apparu "récemment" c'est l'usage des **réseaux de neurones** où $U_\theta(x) = \mathrm{NN}_\theta(x)$, avec là aussi un florilège de choix d'architectures spécifiques. Comment estimer les paramètres du réseau est assez compliqué dans ce schéma, et c'est pour cette raison que sont apparus les techniques de GANs, etc. Étudions néanmoins comment s'opère l'optimisation.

3.1.2 Optimisation des paramètres

Ce que l'on veut, c'est obtenir θ^* tel que p_{θ^*} estime au mieux $p(x)$. Pour cela, il nous faut une métrique qui contraint le choix de p_θ. L'approche la plus classique introduite à R. Fisher en 1922 est celle du **Maximum de Vraisemblance**[31] $\log p_\theta(x)$. Si l'on dispose d'échantillons typiques[32] x, c'est que leurs probabilités $p(x)$ sont grandes, et il en est de même de $\log p(x)$. Donc, ce que l'on va considérer c'est $\mathbb{E}_{x\sim p}[\log p_\theta(x)]$, soit l'espérance de la vraisemblance de $p_\theta(x)$ quand x est distribué selon la *pdf* $p(x)$, et l'on va demander à maximiser cette quantité pour obtenir θ^*:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\ \mathbb{E}_{x\sim p}[\log p_\theta(x)] = \underset{\theta}{\operatorname{argmin}}\ \mathbb{E}_{x\sim p}[-\log p_\theta(x)] = \underset{\theta}{\operatorname{argmin}}\ \ell(\theta) \tag{26}$$

31. NDJE. Cours 2022 Sec. 3.5
32. nb. vocable de Cl. Shannon

la dernière égalité introduit la fonction de coût $\ell(\theta)$ à minimiser. Or, cette quantité est très naturelle en **Théorie de l'Information**. Explicitons l'espérance:

$$\begin{aligned}\ell(\theta) &= -\int \log p_\theta(x)\ p(x)dx \\ &= \int p(x) \log \frac{p(x)}{p_\theta(x)}\ dx - \int p(x) \log p(x)\ dx \\ &= D_{KL}(p\|p_\theta) - \mathbb{H}[p] \end{aligned} \tag{27}$$

où l'on fait apparaître la **divergence de Kullback-Leibler** et **l'entropie** de p. Et donc minimiser $\ell(\theta)$ (selon θ) est équivalent à minimiser $D_{KL}(p\|p_\theta)$. Or, nous savons [33] que

$$D_{KL}(p\|p_\theta) \geq 0; \qquad D_{KL}(p\|p_\theta) = 0 \Leftrightarrow p_\theta = p \tag{28}$$

Donc, minimiser $D_{KL}(p\|p_\theta)$ a tendance à modifier p_θ vers la direction de p, et *in fine* cela produit un bon modèle des données.

Dans le cas de modèles d'énergie de Gibbs $\ell(\theta)$ s'écrit:

$$\ell(\theta) = \mathbb{E}_{x\sim p}[U_\theta(x)] + \log Z_\theta \tag{29}$$

Lorsque l'on dispose d'échantillons $\{x_i\}_{i\leq n}$, on peut calculer une estimation de $\ell(\theta)$, en utilisant une moyenne de Monte Carlo pour estimer l'espérance, et l'on a

$$\hat{\ell}(\theta) = \frac{1}{n}\sum_{i=1}^{n} U_\theta(x_i) + \log Z_\theta \tag{30}$$

Ensuite, pour estimer le meilleur θ, on procède par **descente de gradient**: étant donnée une valeur initiale θ_0, le passage de l'étape t à l'étape $t+1$ se fait selon

$$\theta_{t+1} - \theta_t = -\varepsilon \nabla_\theta \ell(\theta_t) \tag{31}$$

Quelles sont les difficultés de cette optimisation?

33. ex. Cours 2023 Sec. 5.3 pour la définition et la propriété de positivité de la divergence de Kullback-Leibler. hint: usage de l'inégalité de Jensen.

Il nous faut calculer:

$$
\begin{aligned}
\nabla_\theta \ell(\theta) &= \mathbb{E}_{x\sim p}[\nabla_\theta U_\theta(x)] + \nabla_\theta \log Z_\theta \\
&= \mathbb{E}_{x\sim p}[\nabla_\theta U_\theta(x)] + Z_\theta^{-1} \int (-\nabla_\theta U_\theta(x)) e^{-U_\theta(x)}\, dx \\
&= \mathbb{E}_{x\sim p}[\nabla_\theta U_\theta(x)] - \mathbb{E}_{x\sim p_\theta}[\nabla_\theta U_\theta(x)] \qquad (32)
\end{aligned}
$$

On constate que le gradient de $\ell(\theta)$ est nul quand $p = p_\theta$. Dans le cas de modèles exponentiels

$$\nabla_\theta U_\theta(x) = \Phi(x) \qquad (33)$$

qui est manifestement indépendant des θ.

Maintenant, la question de la convergence se pose, ce qui revient à se demander si $\ell(\theta)$ est convexe ou pas. Il nous faut alors calculer les dérivées secondes à savoir le Hessien $H(\ell(\theta))$. Dans le cas de modèles exponentiels, le premier terme de $\nabla_\theta \ell(\theta)$ (Eq. 32) ne dépend pas de θ donc n'intervient pas pour le calcul du Hessien, et l'on montre que si $A(\Theta) = \log Z_\theta$ alors [34]

$$\nabla^2_\theta A(\Theta) = Cov_{x\sim p_\theta}(\Phi(x)) \geq 0 \quad \text{cad.} \quad \forall (k,k') \leq K, \frac{\partial^2 A}{\partial\theta_k \partial\theta_{k'}} = Cov_{x\sim p_\theta}(\phi_k(x), \phi_{k'}(x)) \geq 0 \qquad (34)$$

Ce qui nous dit que **dans le cas de modèles exponentiels, la fonction de coût est convexe et donc on a une garantie de convergence vers une unique solution**, même s'il peut y avoir des cas d'optimisations où la fonction de coût est plate aux environs du minimum. Toujours est-il que ces modèles exponentiels servent de référence.

La situation est totalement différente quand $U_\theta(x)$ est modélisée par un réseau de neurones où l'on a beaucoup de minima locaux. Ceci dit dans les cas où l'on utilise les NN pour des problèmes de classification ou de régression, même si on a des minima locaux, en général les propriétés des modèles qui convergent différemment sont identiques statistiquement, même si l'on ne maitrise pas bien pourquoi la plupart du temps. Donc, bon an mal an, les minima locaux ne vont pas être le point bloquant, quelle est alors la difficulté? En regardant l'équation 32, le premier terme est une espérance par rapport aux données, donc on le calcule sans difficulté en prenant la moyenne $\frac{1}{n}\sum_i$ sur les échantillons

34. NDJE. voir Cours 2023 Sec. 8.2, Th. 18

$\{x_i\}_i$, mais le problème vient du second terme où l'espérance est selon p_θ que l'on est en train d'optimiser. On serait tenté d'effectuer une estimation à l'étape t selon la méthode suivante

$$\mathbb{E}_{x\sim p_\theta}[\nabla_\theta U_\theta(x)] \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta U_\theta(y_i;\theta_t) \quad y_i \sim p_{\theta_t} \tag{35}$$

On se rend compte alors que pour effectuer *un seul pas* de descente de gradient, **il nous faut échantillonner p_{θ_t} et cela un grand nombre de fois (N)** pour estimer la moyenne empirique. Et en soit l'échantillonnage est un problème complexe.

Au bilan, même si le cadre mathématique est clair et que l'on peut *in fine* avoir des interprétations des formes d'énergie optimisées, **la mise en pratique est quasiment rédhibitoire**, et il nous faudra trouver une autre approche si l'on veut utiliser beaucoup de paramètres, notamment dans le cas d'usage des réseaux de neurones. Et pourtant nous n'aurons pas le choix: il nous faudra utiliser des réseaux profonds, car les modèles exponentiels ne sont pas suffisamment expressifs pour modéliser des distributions de probabilités des cas les plus complexes.

Cependant, comme le remontre S. Mallat, **les modèles exponentiels sont capables de capturer les distributions de champs étudiés en Physique** comme ceux de la figure 4, que cela soit en cosmologie, dans le domaine des turbulences et en Physique Statistique des modèles de spins d'Ising ou champ ϕ^4. Ce n'est pas rien. En fait, ce type de problèmes peut être attaqué sans réseaux de neurones. Les structures qui apparaissent à toutes les échelles peuvent être capturées par un choix judicieux de $\{\phi_k(x)\}_k$, comme les bases d'ondelettes, et d'une manière générale il y a de **l'information hiérarchique de l'information à capturer**. Ceci dit les champs en question sont assez "simples" car **ils sont stationnaires et ergodiques**, c'est-à-dire que si l'on regarde un de ces champs localement à différents endroits, on note en sous-jacent **une invariance par translation** des propriétés statistiques. Le cas des **visages** de la figure 5 ou des **chambres à coucher** de la figure 14 appartiennent à une tout autre catégorie de problèmes: **les problèmes non ergodiques**.

Néanmoins, avant de passer à la génération des cas difficiles, voyons comment est effectué l'échantillonnage mentionné dans le contexte de l'équation 35, pourquoi il peut être difficile, et comment envisager des méthodes pour résoudre les problèmes.

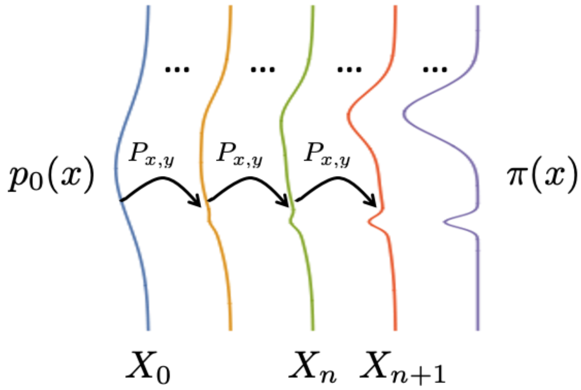


FIGURE 15 – Évolution de la distribution initiale $p_0(x)$ vers la distribution cible (ici notée $\pi(x)$) par application de la matrice de transition $P_{x,y}$ de la chaîne de Markov.

3.2 Échantillonnage d'une *pdf* $\pi(x)$: chaines de Markov

On veut échantillonner une distribution cible, par exemple $\pi = p_\theta$, dont on connait l'expression analytique (via celle des U_θ) et il nous faut tirer un x typique, c'est-à-dire probable $x \sim \pi(x)$. Quelle est la démarche [35]? On va retrouver la notion de transport. L'idée est que l'on va partir d'une *pdf* p_0 connue qui n'est pas celle que l'on veut finalement, mais pour laquelle on sait tirer un échantillon x_i (ex. loi uniforme, gaussienne), et l'on va le transporter de telle façon que l'ensemble des $\{x_i\}_i$ aient la *pdf* cible. Donc, **le problème est de trouver le transport** T. L'approche classique pour réaliser cela est d'utiliser une **chaine de Markov** [36]. C'est une méthode très puissante, mais nous dit S. Mallat, elle est d'une certaine manière "trop flexible".

Si l'on part de $x_0 \sim p_0$, on va progressivement le transformer par application itérative d'une matrice de transition $P_{x,y}$ bien choisie pour l'amener à un échantillon $x \sim \pi$ (Fig. 15). Donc, l'on considère la chaine de variables aléatoires $(v.a)$ $(X_0, X_1, X_2, \ldots, X_n)$ et la chaine correspondante des *pdf* $(p_0, p_1, p_2, \ldots, p_n)$. Si l'on regarde la probabilité conditionnelle du futur connaissant tout le passé, la propriété de Markov nous dit que seul

35. NDJE. Voir aussi Cours 2024 Sec. 7
36. Cours 2024 Sec. 7.8

compte le présent pour prédire le futur. Autrement dit

$$p(X_n = x_n|X_0 = x_0, X_1 = x_1, \ldots, X_{n-1} = x_{n-1}) = p(X_n = x_n|X_{n-1} = x_{n-1}) \tag{36}$$

où le membre de droite est une probabilité de transition de l'état $X_{n-1} = x_{n-1}$ vers l'état $X_n = x_n$ que l'on représente $P_{x,y} = p(y|x)$ avec x l'état initial et y l'état final (attention à l'ordre). On retrouve cette propriété de Markov dans bien de systèmes physiques [37]. Cela a pour conséquence de simplifier la relation générale de l'équation 6:

$$p(x_0, x_1, \ldots, x_n) = p(x_0)\prod_{i=1}^{n} p(x_t|x_{t-1}) \tag{37}$$

Ainsi, on peut avoir accès simplement à toutes les statistiques jointes.

À ce schéma général, pour obtenir T, on adjoint des simplifications. En principe $p(X_n = x_n|X_{n-1} = x_{n-1}) = P_{x_{n-1},x_n}$ dépend de n, une simplification consiste alors à imposer une **indépendance vis-à-vis de n (chaine stationnaire)**, et on utilise alors la notation générique $P_{x,y} = P$ (matrice ligne, colonne). Donc, à l'étape $n+1$ de l'évolution de la chaine de Markov

$$p_{k+1}(x) = \sum_{x_k \in \mathcal{X}} p(x|x_k)p_k(x_k) \tag{38}$$

En termes matriciels, si $\mu_k = [p(X_k = x)]_{x\in\mathcal{X}}$ (vecteur colonne), alors [38]

$$\mu_k = P^T\mu_{k-1} = (P^T)^k\mu_0 \tag{39}$$

On veut que lorsque quand n tend vers l'infini μ_n converge vers $\mu_\pi = [\pi(x)]_{x\in\mathcal{X}}$ qui dans le cas paramétré s'écrit $\mu_\theta = [p_\theta(x)]_{x\in\mathcal{X}}$, et la convergence doit s'opérer quelle que soit la distribution initiale. Cela nous dit alors que μ_π doit être un **point fixe de la transformation**. Remarquons que si la notation vectorielle de μ fait référence à des états discrets pour simplifier le propos, l'idée se généralise au cas continu. Maintenant, il faut se poser la question: à quelle(s) condition(s) sur la matrice P ce schéma de convergence est-il possible?

37. NDJE. Quelques exemples sont présentés dans le Cours de 2023 Sec. 6.4.2 et dans le Cours 2024 Sec. 8.1.
38. NDJE. par cohérence, je reprends les notations de 2023 et 2024

Les propriétés qui garantissent la convergence sont les suivantes [39]: l'**irréductibilité**, la **récurrence positive** (ou l'absence de cycle), et **unicité du point fixe** (de la mesure invariante). Donc, on a un cadre assez large qui nous permet de créer des chaines de Markov (c'est-à-dire déterminer la matrice de transition) ayant μ_π (et *pdf* cible) comme point fixe.

Un algorithme particulier permet cela, il s'agit de celui de **Metropolis-Hasting** [40]. Il nous faut définir la matrice $P_{x,y} = p(y|x)$ et l'idée est de partir d'une densité $Q(x,y)$ connue [41], par exemple une gaussienne ($Q(x,y) \propto \exp(-\|x-y\|^2/(2\sigma^2))$) et de la modifier selon

$$p(y|x) = Q(x,y) \times \rho(x,y) \tag{40}$$

Si on ne garde que le noyau gaussien $\rho(x,y) = 1$, pour $x = x_0$ fixé, on échantillonne $y \sim Q(x_0, y)$, et y à son tour devient la valeur de x_1, le nouvel échantillon de la chaine. Mais dans ce cas, il n'y a aucune chance que π (p_θ) soit le point de convergence. Il nous faut donc modifier la forme de $\rho(x,y)$ qui n'est autre que **la probabilité d'acceptation** de l'échantillon y. C'est une procédure de "rejet" [42] sur la probabilité de transition proposée.

Il faut en fait que l'on ait une propriété de **balance détaillée** liée à la **propriété des chaines de Markov réversibles**, laquelle nous dit que la distribution invariante π et les probabilités de transitions sont reliées par la relation:

$$\pi(x)p(y|x) = \pi(y)p(x|y) \tag{41}$$

C'est-à-dire le nombre d'entités qui vont de l'état x vers l'état y est le même que celui qui vont de y à x. Donc, cela donne une condition sur $P_{x,y}$ pour que π soit un point fixe, et donne alors l'expression de ρ.

Dans le cas de Nicholas C. Metropolis envisageait, $Q(x,y) = Q(y,x)$ (comme le cas

39. NDJE. Voir Cours 2024 Sec. 8.4 avec notamment le théorème d'ergodicité (Th. 13).
40. NDJE. Cours 2024 Sec. 8.6, et voir dans le repository Github en 2023 le notebook `Monte_Carlo_Sampling.ipynb` par exemple.
41. NDJE. je reprends la notation de 2024
42. NDJE. Cours 2024 Sec. 7.5

gaussien), ce qui donne [43]

$$\rho(x,y) = \min\left(1, \frac{\pi(y)}{\pi(x)}\right) \qquad \text{(Metropolis)} \tag{42}$$

Dans le cas où $Q(x,y)$ n'est pas symétrique, l'algorithme 1 donne l'expression générale.

Algorithm 1 Metropolis-Hastings

Require: $Q(x,y)$ une distribution facile à échantillonner pour obtenir x

1: Shoot $x_0 \sim \mu_0$ (ex. $Q(x,0)$)
2: **for** $i : 1, \ldots, n$ **do**
3: Shoot $y \sim Q(x_{i-1}, .)$ and $u \sim \mathcal{U}(0,1)$
4: Compute $r = \rho(x_{i-1}, x_{prop}) = \min\left(1, \frac{Q(y,x_{i-1})\pi(x_{prop})}{Q(x_{i-1},y)\pi(x_{i-1})}\right)$
5: **if** $r = 1$ OR $u \leq r$ **then** $x_i = y$
6: **else** $x_i = x_{i-1}$
7: Keep x_i
8: return $(x_i)_{i \leq n}$

Au bilan, on a une théorie bien construite autour des chaines de Markov, avec une garantie de convergence avec des algorithmes MCMC. Quel grain de sable peut faire bloquer cette mécanique bien huilée? Il s'agit des problèmes où $U_\theta(x)$ est **non-convexes**, qui donnent une distribution p_θ multi-modale. La figure 16 illustre le propos: à chaque itération de la chaine de Markov, si le noyau gaussien $Q(x,y)$ a une largeur (σ) trop petite, on va selon toute vraisemblance finir par avoir un échantillon de p_θ qui correspond à un mode (forte probabilité), mais on n'aura quasiment pas la possibilité d'explorer les autres modes. En effet, pour cela, il faudrait qu'un nouvel échantillon y du noyau Q "tombe" dans la région d'attraction d'un autre mode (donc à forte probabilité), pour qu'il ait une chance de passer l'étape 5 qui requière une valeur de ρ assez grande pour que y soit garder. Or, si σ est trop petit un tel "saut" est impossible, **la conséquence est que l'on génère qu'un seul type d'échantillon**. Pour remédier à cela, on pourrait penser à augmenter largement la valeur de σ. Dans ce cas extrême, on peut se retrouver avec à chaque étape une proposition y telle que $p_\theta(y) \ll p_\theta(x_{n-1})$, on l'élimine alors systématiquement à l'étape 5 de l'algorithme, et **la chaine stagne**. Donc, on a deux cas

43. NDJE. Voir Cours 2023 Sec. 8.9

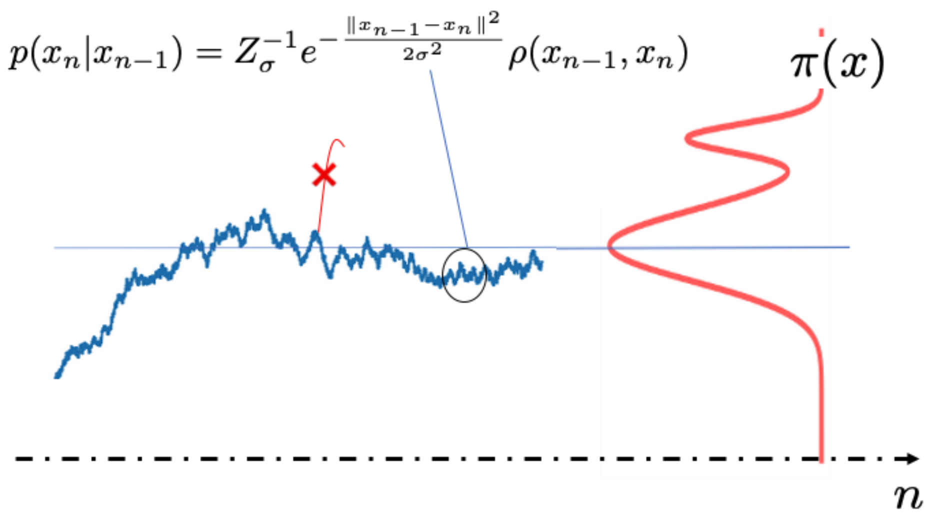


FIGURE 16 – Cas problématique d'une distribution cible $\pi = p_\theta$ multi-modale ($U_\theta(x)$ non convexe et un noyau $Q(x, y)$ gaussien avec un σ trop petit.

extrêmes où la chaine de Markov donne soit un échantillon d'un seul mode, soit elle tombe dans une trappe où elle n'avance pas. **La chaine n'échantillonne pas correctement la densité de probabilité p_θ, et donc la collection des échantillons $(x_i)_i$ ne peut servir ni pour calculer des propriétés statistiques de p_θ, ni pour calculer des intégrales comme celle rencontrée ($\mathbb{E}_{x\sim p_\theta}[\nabla_\theta U_\theta(x)]$) dans le calcul du gradient de la fonction de coût pour déterminer les paramètres optimaux θ^*. C'est le gros problème de ces techniques Monte Carlo en grande dimension où l'on utilise des réseaux de neurones pour modéliser U_θ.**

Bien entendu, il a été tenté de modifier le schéma ci-dessus, notamment en essayant la technique de *repliqua-exchange* (RepEx). Primo, au lieu d'essayer d'obtenir des échantillons de $p_\theta = \pi$ uniquement, avec ses problèmes de multi-modes, on va tenter d'obtenir des échantillons de la famille $\{\pi_\beta(x) = \pi(x)^\beta\}_{\beta\in[0,1]}$. Notons que $\beta = 1$ nous donne la distribution cible, et pour $\beta < 1$, nous obtenons des répliques plus "plates" où passer d'un mode à l'autre est facilité. Le paramètre β est identifié à l'inverse d'une température, réminiscence des modèles en thermodynamique où $p \propto e^{-\beta U(x)}$. Secundo, imaginons que l'on fasse évoluer plusieurs chaines, chacune avec des valeurs de β différentes. À l'étape k, les chaines i et j sont dans les états (x_k^i, x_k^j), et l'idée est alors qu'à l'étape $k+1$ l'on ait $x_{k+1}^i = x_k^j$ et $x_{k+1}^j = x_k^i$. Bon, il est clair que si cet échange est effectué brutalement ni l'une ni l'autre chaine ne seront des échantillons de π^{β_i} et π^{β_j}. Il nous faut donc une probabilité d'acceptation de cet échange. En fait, c'est le critère de Metropolis qu'il convient, à savoir

$$\rho^{RepEx}(x^i_{k+1} = x^j_k, x^j_{k+1} = x^i_k) = \min\left(1, \frac{\pi_{\beta_i}(x^j_k)}{\pi_{\beta_i}(x^i_k)} \times \frac{\pi_{\beta_j}(x^i_k)}{\pi_{\beta_j}(x^j_k)}\right) \tag{43}$$

où l'on reconnait les probabilités de transition pour la chaine i et la chaine j de l'équation 42. Maintenant, on choisit des paires de chaines proches en valeur de β et les autres chaines utilisent le schéma MCMC "classique". Il y a beaucoup de raffinements dans ce genre d'algorithme. Le fait est que l'on peut potentiellement échantillonner des probabilités multi-modales, mais pour ce faire, remarquons que pour obtenir $x \sim p_\theta$ (à chaque étape de la minimisation de $\ell(\theta)$) il nous faut échantillonner les distributions $\beta < 1$, ce qui en fin de compte ne nous intéressent pas. De plus, plus la dimensionnalité du problème augmente plus le nombre de $(\beta_i)_i$ à considérer augmente aussi, ce qui aggrave le constat précédent. Donc, quand on utilise des réseaux de neurones pour paramétrer U_θ, on ne s'en sort pas de cette manière. La solution va être de **changer la métrique**.

3.3 La métrique de Fisher

La métrique qui nous paraissait naturelle pour obtenir le fait que p_{θ^*} soit aussi proche de p que l'on veuille était celle de Kullback-Leibler tirée du Maximum de Vraisemblance (Eq. 27). Or, si la métrique $D_{KL}(p||p_\theta)$ est certes bien motivée mathématiquement, elle est d'une certaine manière "trop forte" nous dit S. Mallat: soit on a affaire à un problème "simple", et la métrique ne pose pas de problème, soit on a à traiter des problèmes complexes et alors, on doit faire face aux problèmes cités à la section précédente, en particulier les algorithmes peuvent être très lents. **D'où vient cette lenteur de l'optimisation?**

Elle vient de l'estimation du terme $\mathbb{E}_{x\sim p_\theta}[\nabla_\theta U_\theta(x)]$ dans le calcul du gradient de la fonction de coût (Eq. 32). Or, ce terme vient de la constante de normalisation Z_θ, lui-même venant du calcul de $\log p_\theta(x)$. Peut-on s'en affranchir? Que se passe-t'il si on dérive par rapport à x en non θ (attention!):

$$-\nabla_x \log p_\theta(x) = \nabla_x U_\theta(x) \tag{44}$$

Il s'agit du **score** mentionné à la séance précédente. Si l'on veut calculer les espérances, on peut le faire directement à partir des échantillons d'entrainement:

$$\mathbb{E}_{x\sim p}[\nabla_x U_\theta(x)] = \frac{1}{n}\sum_i \nabla_x U_\theta(x_i) \tag{45}$$

et le gradient par rapport à θ est également simple à calculer.

Donc, au lieu de maximiser $\mathbb{E}_{x\sim p}[\log p_\theta(x)]$ (maximum de vraisemblance, Eq. 26) ou minimiser avec le signe moins, on va minimiser les gradients, et la fonction de coût s'écrit alors

$$\ell(\theta) = \mathbb{E}_{x\sim p}\left[\|\nabla_x \log p_\theta(x) - \nabla_x \log p(x)\|^2\right] \tag{46}$$

Il s'agit de **la métrique de Fisher**. C'est à la base des algorithmes efficaces d'échantillonnage mentionnés dans la séance précédente. En s'affranchissent de l'étape d'échantillonnage selon p_θ à chaque étape de la descente de gradient, les algorithmes sont infiniment plus rapides. Mais, il y a un hic cependant, en procédant ainsi, quid de la constante de normalisation? En fait, on perd de l'information, et cela pose des difficultés qui viennent de la faiblesse de la métrique utilisée. Mais, en passant aux équations différentielles d'évolution (algo de score diffusion), nous allons les éviter. Mais avant cela, voyons l'étape des GANs.

3.4 Modèles à base de réseaux: les GAN

Des méthodes "classiques" décrites dans les sections précédentes, il est bon de retenir que les problèmes complexes sont difficiles, c'est-à-dire que si l'on trouve un algorithme qui marche "miraculeusement", nous dit S. Mallat: il faut toujours avoir un regard critique et se demander jusqu'à quel point est-il si prometteur que cela? n'y-a-t'il pas une chose qui a été laissée de coté? C'est le cas des GANs évoqués (Sec. 2.7) lors de la séance précédente. Ils permettent de générer de belles images (Fig. 5 à droite) mais le problème de rester coincé dans un mode peut persister dans les cas complexes. Ceci dit, les GANs ont montré à la grande surprise, 1) que définir une "*distribution de probabilité de visage*" peut être crédible, et 2) l'expressivité est donnée par les réseaux de neurones. Il n'en reste pas moins que la question de la généralisation se pose: peut-on distinguer le GAN d'un logiciel d'arrangement astucieux de briques de base et de modification de la couleur, de la forme de telle ou telle partie du visage, etc. Dans la suite, on considère en arrière-plan le cas de génération d'images, mais il se généralise bien entendu [44].

L'idée de base est de partir d'un bruit blanc (on parle de **variables latentes**) et il nous faut trouver le transport idoine pour obtenir x un échantillon de la distribution cible voulue. Voir la figure 6 pour illustrer le propos. Si $x \in \mathbb{R}^d \sim p_{data}$, la variable aléatoire latente $z \sim p_z$ quant à elle peut être dans un espace de dimension plus réduite (c'est le cas en général). On définit alors une architecture neuronale, le **générateur** paramétré G_θ, tel que $x = G_\theta(z)$. Comment tester que x est bien un nouvel échantillon de la *pdf* sous-jacente aux données? Comme on l'a vu, si on se lance dans la voie d'un maximum de vraisemblance, on sait que l'on va au devant de difficultés d'optimisation qui peuvent être rédhibitoires. Donc, Goodfellow et al. se sont demandés comment savoir si x est une image de la base de données (*true sample*) ou bien si elle est issue de G (*fake sample*)? Il suffirait de faire appel à une sorte d'*oracle* qui nous dirait "oui" ou "non". Faute d'oracle, on peut élaborer un classificateur (ou *discriminateur*) de nouveau avec une architecture neuronale, $D_{\theta'}(x)$**, donnant la probabilité que** x **soit issu de la base de données**.

Pour optimiser les deux architectures, l'idée est que G_θ et $D_{\theta'}$ agissent en adversaires: G_θ va tenter de faire des "vrais" exemplaires issus de p_{data} aussi ressemblant que possible, et $D_{\theta'}$ va essayer de les débusquer comme "faux". Quand le discriminateur n'arrive plus à différentier les échantillons issus de G_θ, de ceux issus de la base de données, alors on pense que G_θ donne de bons échantillons de p_{data}. Ainsi, on se donne la fonction de coût suivante:

$$\min_G \max_D V(D,G) = \min_G \max_D \{\mathbb{E}_{x\sim p_{data}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1-D(G(z)))]\} \tag{47}$$

Le premier terme donne une maximisation de la vraisemblance de la probabilité du classificateur? Le second terme concerne le générateur et le discriminateur, en le minimisant on va faire en sorte que $D(G(z)) \approx 1$, c'est-à-dire que $G(z)$ produise un échantillon qui ressemble à ceux de la base de données, mais le discriminateur va vouloir maximiser ce terme et s'il n'est pas leurré, il trouvera que $G(z)$ est une image fausse et tendra à donner la réponse $D(G(z)) \approx 0$. La situation d'équilibre (D^*, G^*) est un **équilibre de Nash** établi en Théorie des jeux. Cependant, est-on garantie que l'on a bien une convergence, et en

44. NDJE. Dans le notebook de 2025 `JAX_blob_GAN_vanilla.ipynb`, je vous donne un exemple de GANs "classiques" afin de générer des distributions 2D constituées par des gaussiennes centrées sur des sommets de polygones réguliers.

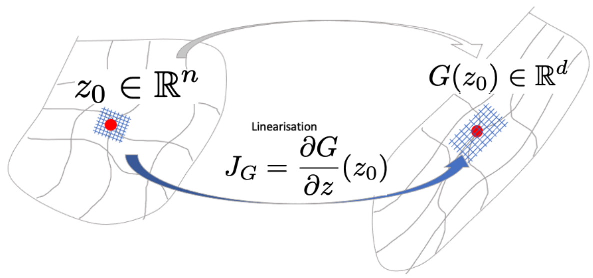


FIGURE 17 – Illustration de la fonction du Jacobien du générateur G entre l'espace des variables latentes (z) et l'espace des données (x).

l'occurrence que les $x = G^*(z)$ avec $z \sim p_z$ sont bien des échantillons de p_{data}? La réponse est oui si l'on est capable de trouver (D^*, G^*).

Prenons $\max_D V(D, G)$, il s'agit de la fonction de coût que le générateur doit minimiser.

Lemme 1 *Donc, on se place dans le cas où l'on fixe G dans un état, et l'on cherche le D^* qui maximise $V(D, G)$, alors*

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} \tag{48}$$

avec $p_G(x)$ la probabilité de x comme résultat de $G(z)$ avec $z \sim p_z$. Il s'agit de la push-forward de p_z qui s'écrit $p_G(x) = p_z(z) \times |J_G|^{-1}$, avec J_G le jacobien de la transformation G.

Démonstration 1.
Ecrivons ce que vaut $V(D, G)$:

$$V(D, G) = \int p_{data}(x) \log D(x)\ dx + \int p_z(z) \log(1 - D(G(z)))\ dz \tag{49}$$

Or, on peut réécrire $p_z(z)dz$ en considérant le transport G en utilisant le Jacobien de la transformation (Fig. 17). Ainsi,

$$p_z(z)dz = p_z(z)|J_G|^{-1}dx = p_G(x)dx \tag{50}$$

où la dernière égalité n'est que la définition de $p_G(x)$. Donc,

$$V(D,G) = \int \Big(p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)) \Big)\, dx \tag{51}$$

Maintenant pour $(a,b,y) \in [0,1]$ et $(a,b) \neq (0,0)$, la quantité $a \log y + b \log(1-y)$ atteint son maximum pour $y_0 = a/(a+b)$. Ainsi

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} \tag{52}$$

■

Ayant ce résultat, comment optimiser le générateur? En fait, il nous faut montrer que p_G converge vers p_{data}, ce qui au passage donne que pour tout x, $D^*(x) = 1/2$. La métrique à minimiser pour G devient après optimisation de D en D^*,

$$\begin{aligned} \ell(G) = V(D^*, G) &= \int \left(p_{data} \log \left(\frac{p_{data}(x)}{p_{data}(x) + p_G(x)} \right) + p_G(x) \log \left(\frac{p_G(x)}{p_{data}(x) + p_G(x)} \right) \right) dx \\ &= \underbrace{D_{KL}(p_{data} \| p_m) + D_{KL}(p_G \| p_m)}_{2 D_{JS}(p_{data} \| p_G)} - 2 \log 2 \end{aligned} \tag{53}$$

où $p_m(x) = (p_{data} + p_G)/2$ et D_{JS} est la **métrique de Jensen-Shannon** [45] qui est symétrique contrairement à la divergence de Kullback-Leibler.

Maintenant, le minimum de $D_{JS}(p_{data} \| p_G)$ d'après les propriétés de divergence de Kullback-Leibler, est obtenu *ssi* $p_G = p_{data}$. Nous avons donc notre résultat voulu qui se traduit par le théorème suivant:

> **Théorème 1**
>
> $$\min_G \max_D V(D,G) = -2 \log 2 \Leftrightarrow p_G = p_{data} \tag{54}$$

Donc, **l'optimisation du GAN en théorie nous garantie d'apprendre la *pdf* sous-jacente des données**. Le point délicat est qu'il faut trouver (D^*, G^*). À la séance prochaine, nous

45. NDJE. il y a peut-être des définitions qui diffèrent du facteur multiplicatif.

verrons que c'est plus compliqué qu'il n'y paraît, et nous verrons où se cache le problème, et comment des stratégies ont été mises en œuvre pour l'éviter.

4. Séance du 29 Janv.

4.1 Retour sur les GANs: pourquoi ça coince?

Comme déjà évoqué, le premier article sur les GANs date de 2014 [46] dont le point marquant fut que l'on s'est rendu compte que l'on pouvait **apprendre des transports de probabilités à l'aide de réseaux de neurones profonds**. Et ce faisant, il y a eu toute une évolution d'idées que nous allons explorer. S. Mallat nous dit que c'est un archétype de "la recherche qui est en train de se faire", moteur des cours au Collège de France. Ce qui fait la spécificité du domaine, c'est que la recherche va très vite, bien que les concepts mathématiques sous-jacents ne soient pas nouveaux. Par exemple, l'idée de "Transport" comme déjà évoqué (Sec. 2.8) vient de travaux de G. Monge de la fin du XVIIIe siècle. Concernant les GANs, c'est le générateur qui joue le rôle du transporteur. Nous avons vu à la séance précédente que pour optimiser le générateur, nous avons introduit un discriminateur agissant en adversaire. Le discriminaeur optimal quant à lui est donné par la relation du lemme 1, et le théorème 1 nous donne le couple générateur-discriminateur optimal (D^*, G^*).

La démonstration du théorème 1 est simple. En effet, pour obtenir G^*, il nous faut effectuer la minimisation de la métrique de Jensen-Shannon (Eq. 53) qui a les propriétés de la divergence de Kullback-Leibler, avec en plus la symétrie, ce qui en fait effectivement une métrique. Elle est nulle *ssi* $p_{data} = p_G$, donc le générateur apprend bien en théorie la distribution des données. Donc, le schéma théorique est clair, il nous faut maintenant passer à la pratique. L'Algorithme 2 est celui utilisé par Goodfellow et al (dit "GAN vanilla") [47].

À la suite des expériences numériques qui s'en sont suivies, voici une liste des problèmes rencontrés:

46. NDJE. voir note de bas de page 20.
47. NDJE. j'ai mis à disposition un exemple simple dans le notebook de 2025 *JAX_blob_GAN_vanilla.ipynb* pour vous faire la main.

Algorithm 2 GAN (Goodfellow et al., 2014)
Il faut préciser le nombre d'itérations total de mise à jour du générateur, le nombre de fois k que le discriminateur est mis à jour par boucle d'optimisation du générateur; le nombre m qui donne la taille des échantillons, et le type d'optimiseur pour les descentes de gradients.

1: **for** nomber of iterations **do**
2: **for** k steps **do**
3: • Shoot m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ selon p_z.
4: • Get m data $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from the dataset.
5: • Discriminator weights update (*gradient ascent*):

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

6: • Shoot m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ selon p_z.
7: • Generator weights update (*gradient descent*):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

— C'est difficile à entrainer en raison du caractère "adversaires" des deux réseaux qui entraine des phénomènes d'oscillations;

— Comme déjà évoqué, le problème principal est que dans le cadre d'optimisation de réseaux de neurones, il y a toutes les chances que l'on se trouve bloqué dans **un minimum local** du discriminateur. Cela entraine le phénomène des **mode collapse**, c'est-à-dire des modes de p_{data} que la génération $G(z)$ ne produit pas *in fine*, pour la raison que le discriminateur les aura rejetés comme images "*fake*" lors de l'entrainement. C'est un cas similaire au problème de la figure 16 vu dans le cadre des chaines de Markov. Par exemple, dans le cas de génération de visages, l'effet du *mode collapse* a pour conséquence par exemple de ne générer qu'un type de visage (ou du moins en oublie certains). Dans le cas de scènes de chambre à coucher, certaines peuvent comporter des personnages, le *mode collapse* aurait pour effet de ne produire que des scènes sans aucune personne.

— Concernant les phénomènes d'oscillations, certains peuvent concerner des minima locaux se caractérisant par exemple comme suit: pendant un temps la génération se fixe sur un type de visages, puis passe soudainement à la génération d'un autre

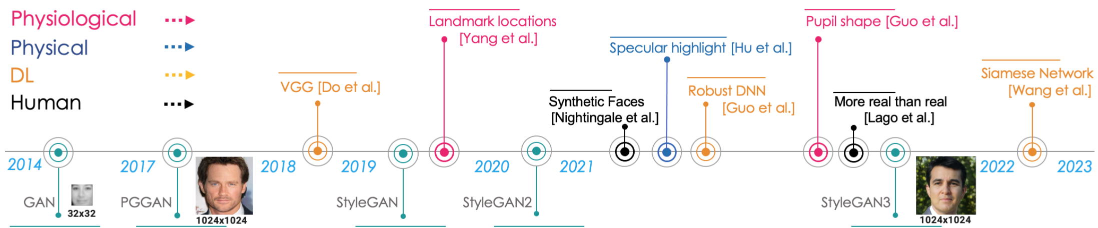


FIGURE 18 – Petite frise chronologique des GANs donnée dans l'article de Xin Wang et al. *arxiv:2202.07145* (v6 datée de Nov. 2023). Après 2017, la série des modèles comme StyleGAN permet de générer des visages hautement réalistes, difficiles à distinguer à l'œil humain. Il y a en parallèle des recherches sur la détection des images générées par de tels modèles pour les distinguer de vraies images par exemple pour aider au *fact-cheking*.

type et ainsi de suite.

Tout ceci nous dit qu'il y a des pièges, car on peut produire de très belles images qui appartiennent bien à un mode de la densité de probabilité (p_{data}), mais on a aucune garantie que l'on reproduise l'ensemble de la *pdf*. Au final, il y a généralement un **manque de diversité des échantillons**. Et malheureusement, le manque de diversité est très difficile à juger.

Durant grosso modo 6 ans, l'effort a porté sur la production de belles images avec des réseaux de plus en plus gros pour augmenter la résolution. La figure 18 illustre l'évolution des GANs. Cependant, a-t'on la bonne distribution de probabilité?

Le point critique est l'usage (même sous forme symétrique Jensen-Shannon) de la divergence de Kullback-Leibler qui pose problème, car elle est difficile à optimiser [48] comme

48. NDJE. Martin Arjovsky et al. fournissent dans *arxiv:1701.07875* un exemple simple. Imaginons une *v.a* Z de *pdf* uniforme sur $[0,1]$. Notons alors p_0 la distribution des points $(0,z)$ (il s'agit du segment de droite unité sur l'axe vertical). Soit maintenant p_θ la *pdf* des points (θ,z) avec θ un unique paramètre réel. En considérant $x \in \mathbb{R}\times[0,1]$

$$D_{JS}(p_0\|p_\theta) = \frac{1}{2}\int dx \Big(p_0(x)\log\left(\frac{p_0(x)}{p_0(x)+p\theta(x)}\right) + p_\theta(x)\log\left(\frac{p_\theta(x)}{p_0(x)+p\theta(x)}\right)\Big) + \log 2$$

il vient:
- si $\theta = 0$, $D_{JS}(p_0\|p_\theta) = 0$, de même $D_{KL}(p_0\|p_\theta)$ et $D_{KL}(p_\theta\|p_0)$ sont nulles;
- par contre si $\theta \neq 0$, $x = (x_1, z)$ si $p_0(x) = 0$ si $x_1 \neq 0$ tandis que $p_\theta(x) = 0$ si $x_1 \neq \theta$; donc on en conclut que $D_{JS}(p_0\|p_\theta) = \log 2$, et $D_{KL}(p_0\|p_\theta) = D_{KL}(p_\theta\|p_0) = +\infty$.

On voit bien un problème de convergence quand $\theta \to 0$.

on l'a vu lors de l'optimisation des problèmes d'énergie paramétrée, et que l'on retrouve ici d'une certaine manière. La question qui vient alors est: peut-on changer de métrique qui nous renseignerait que p_G converge bien vers p_{data}, sans avoir l'inconvénient de la métrique de Jensen-Shannon et de Kullback-Leibler? C'est durant cette quête que l'on est venu à repenser le problème.

4.2 Transport optimal

Concernant le problème des GANs, on aimerait en fait que p_G qui est une *pdf* paramétrée par un jeu de paramètres θ_g, soit régulière pour que la descente de gradient se passe bien (NDJE. voir note de bas de page 48 pour un exemple simple de problème). D'une certaine façon, on aimerait que si l'on considère une divergence/métrique $D(p_{\theta_g} \| p_{data})$ nous ayons la propriété suivante

$$D(p_{\theta_g} \| p_{data}) \xrightarrow[\theta_g \to \theta_g^*]{} 0 \tag{55}$$

Or, D_{KL} est très sensible quand on s'approche de l'optimum. Il nous faut trouver quelque chose de "plus faible", tout en garantissant d'obtenir $p_{\theta_g^*} = p_{data}$. On est donc amené à repenser le problème, et en particulier à utiliser des métriques du *transport optimal* que l'on associe à Gaspard Monge (1746-1818) et Léonid Kantorovich (1912-86).

4.2.1 Le problème de G. Monge

G. Monge a une œuvre considérable en géométrie, en analyse, il a cocréé l'Ecole des Arts et Métiers et l'Ecole Polytechnique (dont il a écrit tous les programmes à sa création), il était Pair de France, Ministre de la Marine, membre d'Académies, etc. Le problème de Monge consigné dans un rapport à l'Académie de Sciences était très pratique: il s'agissait du problème *des Remblais et des Déblais*[49]. L'idée est de pouvoir transporter, avec un minimum d'effort, des tas de sables pour combler des trous. Si z est la position de départ d'un grain de sable et $T(z)$ la position transposée, le coût de transport est une fonction

49. G. Monge 1781, *Mémoire sur la théorie des déblais et de remblais*, https://gallica.bnf.fr/ark:/12148/bpt6k35800/f796

c qui va dépendre de la distance entre ces deux positions, comme $c(x, y) = \|x - y\|^n$ (ex. $n = 1, 2$). Ensuite, il faut trouver la solution au problème suivant:

$$\inf_T \left[\int c(z, T(z)) \; p_z(z) dz; \quad \text{avec la contrainte : } T_\# p_z = p_d \right] \tag{56}$$

avec p_d la distribution des trous (p_{data} dans notre cas). Le problème de G. Monge est de savoir s'il existe une solution, quelles sont les conditions d'existence, et d'en trouver la solution.

Existence d'une solution? En général la réponse est non. Un contre-exemple peut être construit de la façon suivante:

$$p_z(z) = \delta(z - z_0) \qquad\qquad p_{data}(x) = \frac{1}{2}\delta(x - x_0) + \frac{1}{2}\delta(x - x_1) \tag{57}$$

T est une fonction et $T(z_0)$ ne peut être attribué en même temps à x_0 et à x_1. Il faut des conditions de régularités "raisonnables" pour qu'il y ait des solutions.

4.2.2 Relaxation de L. Kantorovich

Concernant la construction de la solution du problème de Monge, c'est en fait un problème non convexe complexe. Il faudra attendre les travaux de Léonid Kantorovich en 1942 [50] pour avoir un autre éclairage. L. Kantorovich eu le prix de la Banque de Suède (Nobel d'Economie) en 1973. Quelle est son idée: au lieu de considérer le transport déterministe de G. Monge (déplacement de grains de sable 1-à-1), il considère un **transport stochastique**. C'est la même idée que lorsque l'on considère les chaines de Markov, entre chaque étape de la chaine, il y a une probabilité de transition.

Pour effectuer le transport de p_z vers p_{data}, soit alors la densité de probabilité $\gamma(z, x)$ sur l'espace $\chi_z \times \chi_x$ qui satisfait les contraintes suivantes

$$\int \gamma(z, x) p_{data}(x) dx = p_z(z), \qquad\qquad \int \gamma(z, x) p_z(x) dz = p_x(x) \tag{58}$$

qui sont représentées schématiquement sur la figure 19. La première concerne la conservation à z fixé de la densité de probabilité $p_z(z)$, et la seconde symétriquement décrit la

50. L. Kantorovich (1942 en russe) *On the transfer of masses*, https://www.math.toronto.edu/mccann/assignments/477/Kantorovich42.pdf.

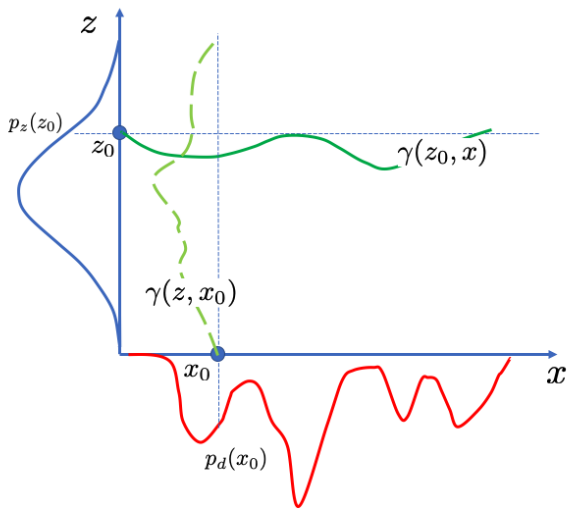


FIGURE 19 – Schématisation des contraintes du problème de L. Kantorovich.

conservation à x fixé de la densité de probabilité $p_x(x)$. Ces deux propriétés garantissent si γ existe, la propriété équivalente du transport de Monge $T_{\#}p_z = p_{data}$ mais cette fois d'une manière probabiliste (ou stochastique). L'ensemble des distributions satisfaisant les contraintes 58 forment un ensemble noté $\Pi(p_z, p_d)$, c'est l'ensemble des distributions sur $\chi_z \times \chi_x$ qui ont pour marginales p_z et p_d.

Il nous faut maintenant avoir l'équivalent du transport à moindre coût. L. Kantorovich utilise une métrique qui sera appelée dans les années 70 celle de **Wasserstein** [51] qui s'écrit [52]

$$W_m(p_z, p_d) = \inf_{\gamma \in \Pi(p_z, p_d)} \mathbb{E}_{(z,x)\sim\gamma(z,x)} \left[\|z - x\|^m\right] \tag{59}$$

où le coût de **transport a été symétrisé**. La forme intégrale du coût s'écrit

$$\mathbb{E}_{(z,x)\sim\gamma(z,x)} \left[\|z - x\|^m\right] = \int_{\chi_z} \int_{\chi_x} \|z - x\|^m \; \gamma(z, x) \; dx dz \tag{60}$$

Cette métrique a des propriétés d'une distance et notamment, elle est **convexe** [53].

Pourquoi a-t-on introduit cette distance de Wasserstein? Elle est plus faible que celle

51. En l'honneur de Leonid Vaseršteĭn (1944-) même si c'est L.Kantorovich qui l'a introduite.
52. NDJE. il peut se trouver des formulations légèrement différentes.
53. NDJE. Concernant le problème de la note de page 48, $W_1(p_0, p_\theta) = |\theta|$ donc converge bien quand θ tend vers 0.

de Kullback-Leibler, c'est-à-dire que si p et q soient supportées par un compact

$$W_1(p,q) \leq C \times \sqrt{D_{KL}(p\|q)} \tag{61}$$

La métrique de Kullback-Leibler est plus sensible, mais $W_1(p,q)$ est suffisante, car si $W_1(p,q) = 0$ alors $p = q$. Donc, étant plus faible W_1 doit être plus facile à optimiser. Il s'en suit que l'on peut mettre au point des algorithmes efficaces tant que l'on n'est pas en trop grande dimension.

L'idée d'utiliser W_1 a été introduite par Martin Arjovsky et al. (voir note de bas de page 48.) en 2017 pour concevoir "*Wasserstein GAN*"[54]. Les auteurs démontrent le théorème suivant que nous admettrons[55]:

Théorème 2 *Si T_θ est un transport de la pdf p vers la pdf de $T_\theta(z)$ avec $z \sim p$, notée p_θ,*

1. *Si T_θ est continu en θ, alors $W_1(p,p_\theta)$ est presque partout continue en θ.*
2. *Si en plus T_θ est localement Lipschitz, alors $W_1(p,p_\theta)$ est presque partout différentiable en θ.*
3. *Les deux propriétés ci-dessus sont fausses pour la divergence de Kullback-Meibler et la distance de Jensen-Shannon.*

La première propriété nous dit que la distance W_1 apporte de la **régularité à la densité de probabilité paramétrée du générateur** (p_G), et la seconde propriété permet d'effectuer **une descente de gradient** pour l'optimisation.

Donc *a priori*, nous avons un cadre bien défini mathématiquement avec un espoir de réaliser un algorithme efficace. Rappelons que notre but est de modifier la fonction de coût du discriminateur (Eq. 53) basée sur la métrique de Jensen-Shannon (composée elle-même de divergences de Kullbach-Leibler). Il y a un résultat de L. Kantorovich et G. Rubinstein (1958) qui stipule que W_1 peut se calculer de la manière suivante:

---

54. NDJE. J'ai mis sur le repository le notebook `JAX_blob_GAN_Wasserstein_regul.ipynb` qui est une adaptation de ce GAN avec une régularisation des gradients.
55. nb. $f : A \to B$ (espaces munis d'une distance) est localement Lipschitz signifie que pour tout $(x,y) \in A$, il existe une constante K, notée $\|f\|_L$, telle que $d_B(f(x), f(y)) \leq K d_A(x,y)$.

Théorème 3 ***(Dualité de Kantorovich-Rubinstein)***

$$W_1(p,q) = \sup_{\|f\|_L \leq 1} \Big[\mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{x\sim q}[f(x)]\Big] \tag{62}$$

En pratique, pour calculer $W_1(p,q)$, on peut utiliser une fonction d'évaluation (pensez à $D(x)$) pour calculer la différence d'espérances, et quand on arrive au maximum, on obtient une évaluation de la métrique. Cela peut se traduire pour notre couple d'adversaires paramétrés $(D_{\theta_d}, G_{\theta_g})$, à rechercher dans l'ensemble des $\{D_{\theta_d}\}_{\theta_d}$ localement Lipschitz ($\|D_{\theta_d}\|_L \leq 1$) et à calculer [56]

$$V(D^*,G) = W_1(p_{data}, p_G) = \sup_{\|D\|_L \leq 1)} \Big[\mathbb{E}_{x\sim p_{data}}[D(x)] - \mathbb{E}_{z\sim p_z}[D(G(z))]\Big] \tag{63}$$

Rappelons qu'ensuite il nous faut minimiser $V(D^*,G) = W_1(p_{data}, p_G)$ pour obtenir G^*. Or, la convergence de p_G vers p_{data} va être facilitée par la différentiabilité de W_1. Notons, au passage que D_{θ_d} ne donne pas *a priori* une probabilité, on n'utilise plus la vraisemblance, d'où la disparition du log qui était présent dans l'Algo. 2. Dans l'article de Martin Arjovsky et al., les auteurs emploient le vocable de "*critic*" pour parler de D, mais nous garderons le terme de discriminateur néanmoins.

En analysant ce qui précède, on se rend compte qu'en changeant la fonction de coût, on adapte la métrique (ou vice-versa). **L'optimisation du "discriminateur" permet de définir une métrique entre la distribution des données p_{data} et p_G la distribution transportée par le générateur.** Et finalement, ce que l'on veut, c'est une métrique qui soit facile à optimiser pour obtenir le bon générateur, c'est-à-dire celui pour lequel $p_G = p_{data}$.

Après la sortie de l'article et le GAN mis en œuvre (*Wasserstein GAN*), d'autres expérimentations ont été effectuées [57]. Certes cela marche un peu mieux, mais pas d'une manière fulgurante par rapport à la solution initiale de Goodfellow et al. Au bilan, **on constate que cela ne résout pas les problèmes** mentionnés à la fin de la section 4.1. Nous sommes toujours dans le cadre d'un équilibre de Nash, il y a toujours les phénomènes

56. rappel p_G est la *pdf* des $G(z)$ avec $z \sim p_z$
57. NDJE. Voir par exemple Ishaan Gulrajani et al. (*arxiv:1704.00028*) publié après celui de Martin Arjovsky et al.

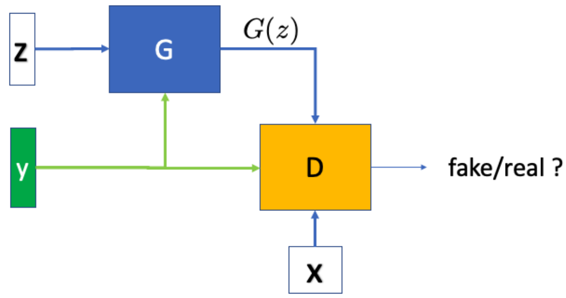


FIGURE 20 – Schématisation d'un GAN conditionnel: par rapport au schéma de la figure 6, on ajoute l'information y à l'entrée du générateur et du discriminateur.

d'oscillations entre les deux adversaires. Le problème reste très complexe avec des minima locaux qui peuvent bloquer la maximisation de D qui empêche de bien calculer la distance de Wasserstein (Eq. 63), donc finalement l'optimisation de G en pâtit. Le point est que nous sommes en **très grande dimension** et que les algorithmes deviennent assez lourd [58].

4.3 GANs conditionnels (cGAN)

L'idée [59] des GANs conditionnels (Fig. 20) est que l'on va "guider" le générateur et le discriminateur en leur donnant des informations supplémentaires y qui peuvent être de toute sorte (prompt, labels de classes, autre image, etc). On considère dans ce contexte une fonction de coût $V(D,G)$ conditionnée comme par exemple

$$V(D,G) = \mathbb{E}_{x\sim p_{data}}[\log D(x|y)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z|y)))] \tag{64}$$

et l'on opère le problème minmax des adversaires comme pour le cas des GANs classiques (Eq.47). Notons que l'on pourrait tout aussi bien utiliser la métrique de Wasserstein. D'une manière pratique, y peut être ajouté au tableau z (resp. x) à l'entrée du générateur (resp. discriminateur).

58. NDJE. Il faut par exemple ajouter des contraintes sur les gradients à chaque pas d'optimisation du discriminateur qui ralentissent considérablement les algorithmes.
59. NDJE. Mehdi Mirza, Simon Osindero (2014) *arxiv:1411.1784* paru peut de temps après celui de Goodfellow et al.

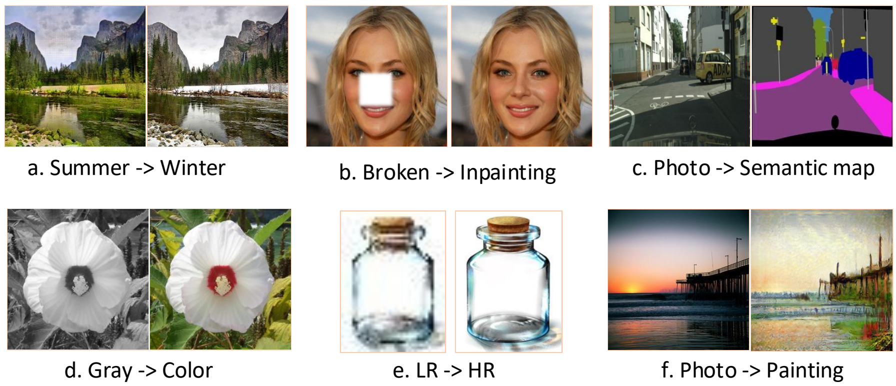


Figure 21 – Quelques illustrations de la transformation d'image-à-image avec un GAN conditionnel (source *arxiv:2101.08629*).

S. Mallat nous présente des illustrations d'images issues de tels "cGAN"[60]: par exemple en entrainant le GAN avec des images prises en été et en hiver, tout en donnant cette information, on peut transformer une photo d'un paysage prise en été pour la faire apparaitre en hiver. Sur ce schéma, on peut effectuer des tâches d'*inpainting*, de segmentation, et bien d'autres encore (Fig. 21). Le problème est plus facile car plus contraint. L'article montre également des comparaisons de plusieurs cGANs (différents selon leurs architectures, les fonctions de coûts, etc) dans la tâche de modifier un visage selon 5 critères (2 couleurs de cheveux, 2 sexes, l'age) (Fig. 22).

4.4 Évaluation des GANs: critères de fidélité

Les expériences numériques réalisées sur les GANs et cGANs au fil du temps paraissent convaincantes, mais comme toujours il faut se poser la question: est-ce que cela marche vraiment bien? C'est-à-dire, a-t-on $p_G = p_{data}$? Le problème sous-jacent est qu'**il n'y a pas l'optimisation explicite d'une métrique**. Concernant W_1 on passe par un argument de dualité (Th. 3), et pour avoir accès à la bonne métrique sur G, il faut obtenir le D optimal or, on n'en est pas certain finalement.

60. NDJE. Il s'agit d'images extraites de l'article de revue de Yingxue Pang et al (2021) *arxiv:2101.08629*. Vous pouvez lire l'article plus ancien de Ph. Isola et al. (2018) *Image-to-Image Translation with Conditional Adversarial Networks*, *arxiv:1611.07004* associé au software `pix2pix`

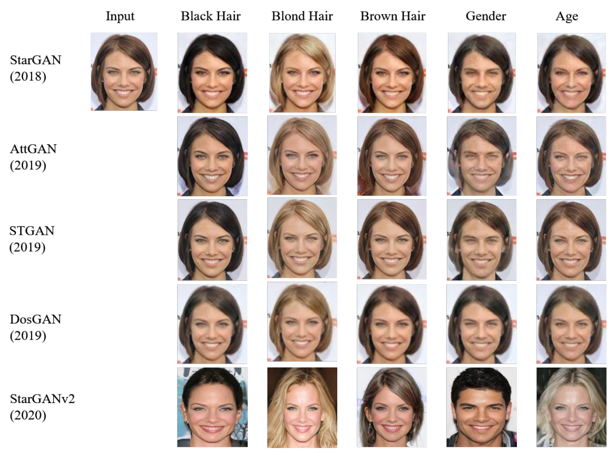


FIGURE 22 – Comparaison de différents cGANs dans la génération d'images de visages conditionnées par 5 attributs (source *arxiv:2101.08629*).

Les "expérimentateurs" se sont tournés vers des **évaluations qualitatives** mettant en jeu[61]:

— un **critère de fidélité** qui répond à la question: l'image est-elle de "bonne qualité"? C'est-à-dire, l'image générée peut-elle être considérée comme fidèle à ce que l'on perçoit d'une image de la base de données. En termes mathématiques, grossièrement cela se traduit sur le fait que le support de p_G (cf. les ensembles typiques) est tel que

$$\text{Support}(p_G) \subset \text{Support}(p_{data}) \tag{65}$$

— auquel il nous faut ajouter de la **diversité**, car on a besoin de l'inclusion inverse

$$\text{Support}(p_G) \supset \text{Support}(p_{data}) \tag{66}$$

Or, c'est le gros problème des GANs.

En pratique, nous avons des exemples (*real*) du dataset $\{x_i^{data}\}_i$ et des exemples (*fake*) issus du générateur $\{x_j^g\}_j$, en guise de critère de fidélité, on peut envisager de calculer une **métrique sur les moments**. C'est-à-dire qu'à partir d'un x, on définit un ensemble de descripteurs (*features*) ($\Phi(x)$) qui peuvent être des choses simples comme des moments de différents ordres, tout comme des sorties de réseaux de neurones, etc. On peut alors

61. nb. le cas d'image est pour illustrer le propos, cela se généralise.

calculer

$$\mathbb{E}_{x\sim p_{data}}[\Phi(x)] = \frac{1}{N}\sum_{i=1}^{N}\Phi(x_i) \tag{67}$$

idem pour $\mathbb{E}_{x\sim p_G}[\Phi(x)]$. La question qui se pose alors est de savoir

$$\mathbb{E}_{x\sim p_{data}}[\Phi(x)] \stackrel{?}{=} \mathbb{E}_{x\sim p_G}[\Phi(x)] \tag{68}$$

La difficulté est de choisir [62] les $\Phi(x)$. Les descripteurs utilisés en traitement d'images sont des descripteurs `FID` (*Fréchet Inception Distance*) introduits en 2017. Concernant "Fréchet" (Maurice Fréchet) cela vient de la distance entre deux variables aléatoires (moyenne μ, déviation standard σ, ou matrice de covariance)

$$\begin{aligned} d^2(X,Y) &= (\mu_X - \mu_Y)^2 + (\sigma_X - \sigma_Y)^2 \\ \text{ou } d^2(X,Y) &= \|\mu_X - \mu_Y\|^2 + \mathrm{Tr}(\Sigma_X + \Sigma_Y - 2(\Sigma_X\Sigma_Y)^{1/2}) \end{aligned} \tag{69}$$

On va utiliser les *v.a* $X = \Phi(X^{data})$ et $Y = \Phi(X^g)$. Concernant Φ, on entraine sur la base de données `ImageNet` un réseau de neurones d'architecture particulière, l'`Inception` de Google (v3, 2016), pour qu'il soit un bon classificateur, et comme $\Phi(x)$ on choisit la couche située au milieu de l'architecture.

Maintenant, **rien ne nous garantit que les descripteurs soient suffisamment "fins"** pour évaluer la réponse à la question posée précédemment (Eq. 68), ni même si cela rend compte de la qualité/diversité de la génération. Les choix de l'Inception, de la base ImageNet, et de son entrainement introduisent inévitablement un biais [63]. NDJE: Concrètement, on peut dans le cadre d'analyse astro, se poser la question si des images de galaxies peuvent traiter de la sorte (test FID), alors qu'ImageNet n'en contient aucune? Mais plus dramatiquement, S. Mallat nous indique que la méthode FID détecte assez mal les problèmes de **mode collapse**.

Finalement, la communauté est arrivée à un point où les images étaient certes de plus en plus belles, mais il n'y avait pas de démonstration que $p_G = p_{data}$. **Le problème de la diversité n'est vraiment pas un détail**, si l'on simule des environnements pour optimiser

62. NDJE. dans le cas de production d'images de galaxies, on utilise par exemple des variables liés à la "morphologie" comme celles produites par le software python `statmorph` https://github.com/vrodgom/statmorph.
63. NDJE. on peut même se demander si un GAN peut être conditionné à passer le test "FID".

un système, mais que la simulation "néglige" des cas de figures, le système en question ne sera pas robuste par rapport à l'occurrence d'événements peut être un peu plus rares, mais pas forcément. D**onc, *in fine* ces générateurs ne sont pas suffisamment fiables dans des cas où cette notion est importante.** On a donc fini par proposer d'autres approches pour circonscrire ces problèmes.

4.5 Normalizing Flows (NF)

S. Mallat nous montrera un algorithme mis en œuvre par Kingma et Dhariwal[64] (`Glow`) qui est un aboutissement dans ce domaine[65]. Remarquons au passage que l'article de Goodfellow et al. datait seulement de 4 ans. Le cadre mathématique est identique, à savoir, on définit un transport T d'une distribution simple $p_z(z)$ vers une distribution $p_{data}(x)$ plus complexe à échantillonner de prime abord.

La première idée supplémentaire évoquée à la section 2.8.1 est d'imposer non seulement que **le transport soit réversible** (Fig. 8) mais que T et T^{-1} soient **différentiables**, ce sont des **difféomorphismes**. Cela conditionne que la **dimension de l'espace latent** (espace des z) soit **la même que celle de l'espace des données** (espace des x). Notons au passage que dans le cas des GANs, il n'y a pas cette contrainte. La seconde idée supplémentaire également déjà évoquée, est que le transport global de p_z à p_{data} est découpé en une collection discrète de **transports élémentaires** plus faciles à optimiser:

$$T = T_k \circ T_{k-1} \circ \cdots \circ T_1 \quad \Leftrightarrow \quad T^{-1} = T_1^{-1} \circ T_2^{-1} \circ \cdots \circ T_k^{-1} \tag{70}$$

Il y a trois avantages qui *a priori* vont être donnés par cette approche:

— z est une **variable latente** que l'on peut calculer à partir de x (ce n'est pas le cas du GAN). Donc par exemple, une fois que l'on a optimisé le générateur, on peut lui faire générer des échantillons $\{x_i^g\}$, les faire transporter en sens inverse vers une collection de $\{T^{-1}(x_i^g) = z_i^g\}_i$ et tester s'ils sont bien issus de p_z bien plus

64. NDJE. Diederik P. Kingma, Prafulla Dhariwal (2018), *Glow: Generative Flow with Invertible 1x1 Convolutions*, *arxiv:1807.03039*

65. NDJE. concernant les Normalizing Flows on peut citer antérieurement par exemple l'article de Tabak E. G., Turner C. V., 2013, *A family of non-parametric density estimation algorithms*, Communications on Pure and Applied Mathematics, 66, 145, https://math.nyu.edu/~tabak/publications/Tabak-Turner.pdf

facile à manier. Aussi, on peut à partir d'un z^g, lui faire une translation euclidienne (métrique dans l'espace d'un *bbg*), $z' = z^g + \delta_z$, et de nouveau générer un nouvel échantillon $x^{g'} = T(z')$, ce qui permet par exemple des déformations continues d'une image, et aussi ajouter des lunettes (ou les enlever) à un visage, etc.

— on va effectuer un **calcul explicite des distances**, en l'occurrence celle de **la vraisemblance**, donc celle de **Kullback-Leibler**. On va pouvoir directement comparer des algorithmes en calculant la valeur de cette divergence.

— enfin, **on va éviter l'équilibre de Nash** sous-tendu par le minmax qui est à l'origine des phénomènes d'oscillations des GANs.

Ce qui reste des expériences effectuées avec les GANs c'est que **le transport est calculé à l'aide d'un réseau de neurones profond**.

4.5.1 L'effet d'un transport

Pour comprendre comment s'optimise un NF, nous allons voir l'effet d'un transport sur une distribution de probabilité. On note certes $p_g = T_\# \, p_z$, mais comment s'obtient p_g? On peut se rapprocher de la figure 23: l'idée maitresse est que le transport étant déterministe, il y a **une conservation de la probabilité** d'un ensemble $A \subset \Omega_x$ (espace des x générés) lors du transport des $z \in \Omega_z$ qui l'ont généré. On peut utiliser n'importe quelle fonction h définie sur A, et la contrainte de conservation s'écrit alors:

$$\int_A h(x) p_g(x) dx = \int_{T^{-1}(A)} h(T(z)) p_z(z) dz \tag{71}$$

Si $h = \mathbf{1}_A$, l'indicatrice de A, alors l'égalité s'écrit

$$\mathbb{P}_g(A) = \mathbb{P}_z(T^{-1}(A)) \tag{72}$$

Maintenant quel est le lien entre dx et dz, avec $x = T(z)$? Il s'agit d'un changement de variables où apparait le déterminant [66] de la matrice carrée Jacobienne [67] de la transformation (Fig. 17):

$$dx = dz \times |J_T(z)| \tag{73}$$

66. NDJE. soit le déterminant est noté det A soit $|A|$, mais en tous les cas, on prend la valeur absolue.
67. NDJE. Voir Cours 2019 Sec. 8.1.3

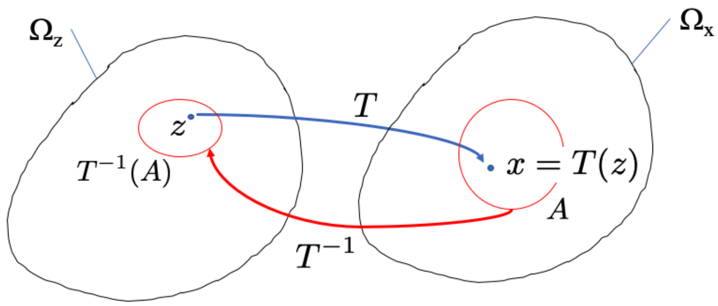


FIGURE 23 – Schématisation de la loi de conservation des probabilités (Eq. 71)

Donc, l'équation 71 devient

$$\int_A h(x)\ p_g(x)\ dx = \int_A h(x)\ p_z(T^{-1}(x))\ |J_T(z)|^{-1}\ dx \tag{74}$$

et cela quelque soit h, d'où l'on retrouve la relation mentionnée à la section 2.8.1, à savoir que p_g se calcule à partir de p_z et du transport T^{-1} selon

$$p_g(x) = p_z(T^{-1}(x))\ |J_T(T^{-1}(x))|^{-1} = p_z(T^{-1}(x))\ |J_{T^{-1}}(x)| \tag{75}$$

Le terme du Jacobien est là pour tenir compte du rapport du volume de $T^{-1}(A)$ à celui de A.

4.5.2 Optimisation de la vraisemblance

Considérons un transport paramétré T_θ où les θ en particulier sont les poids d'un réseau de neurones profonds. Ce que l'on veut c'est que la distribution p_z soit transportée en p_θ (ex p_G) de telle façon que p_θ approxime au mieux p_{data}. Comme déjà vu précédemment, on utilise la vraisemblance comme dans le cas des modèles paramétrés (Sec. 3.1.2), où la fonction de coût à minimiser s'écrit

$$\ell(\theta) = -\mathbb{E}_{x\sim p_{data}}[\log p_\theta(x)] = D_{KL}(p_{data}\|p_\theta) - \mathbb{H}[p_{data}] \tag{76}$$

L'avantage est que l'on peut estimer la valeur de la vraisemblance une fois l'optimisation effectuée. Or,

$$\log p_\theta(x) = \log p_z(T_\theta^{-1}(x)) + \log |J_{T_\theta^{-1}}(x)| \tag{77}$$

Donc, p_z étant connue comme un bruit blanc, le $\log p_z$ correspond à une norme au carré, il nous faut maintenant un transport inversible dont il faut calculer le Jacobien. Et quand on considère la chaine de transports, il nous faut calculer la somme de tous les log des déterminants des jacobiens. Le point délicat techniquement est donc le calcul de ces jacobiens.

Nous verrons que l'on peut rendre ce calcul simple pour des choix particuliers de types de transports[68]. Il va y avoir de nouveau une évolution, et à un certain point, la communauté s'est rendue compte que discrétiser le transport devient trop compliqué, et qu'il vaudrait mieux tenter la voie des *transports continus*.

5. Séance du 5 Fév.

NDJE. Petit point sur les notations. Par la suite les notations des pdf p_{data} et p peuvent être confondues, p_g est la pdf d'un générateur, p_z celle des variables latentes. De plus, p_0 est la pdf initiale dans un processus où p_t est celle des échantillons à un instant t d'évolution. Suivant le cas, p_0 est identique à p_z ou à p_{data}. Il faut donc faire attention au contexte.

Nous continuons notre incursion dans le domaine des Normalizing Flows de la séance précédente. Le point nouveau par rapport aux GANs est que l'on a une métrique, la vraisemblance, pour juger de la qualité du modèle (Eqs. 76, 77). Si on dispose de données $\{x_i\}_{i\leq n}$ *iid* selon p_{data}, alors la vraisemblance peut être approximée par la moyenne Monte Carlo

$$\ell(\theta) \approx -\frac{1}{n}\sum_{i=1}^{n}\Big(\log p_z(T_\theta^{-1}(x_i)) + \log |J_{T_\theta^{-1}}(x_i)|\Big) \tag{78}$$

Pour trouver θ^* optimum, on opère une descente de gradient afin de minimiser $\ell(\theta)$, tout en sachant que le problème n'est sans doute pas convexe. Cependant, on espère qu'il y a

68. NDJE. sur le repository Github je vous ai mis 1) une note (en français) que j'ai écrit en 2022 montrant quelques détails d'architectures, 2) deux notebooks `TensorFlow_bijector_1D_simple.ipynb` et `JAX_FLows_MAF_NVP_simple.ipynb` qui mettent en œuvre simplement des NFs.

une convergence possible vers un minimum local assez profond (comparable au minimum global). Ainsi, $\ell(\theta^*)$ nous renseigne sur la qualité du modèle optimisé. On a le cadre, comment cela se met-il en place concrètement?

5.1 En pratique: le modèle `Glow`

La première chose déjà évoquée est que le transport global T (de p_z à p_g) est factorisé en de multiples transports élémentaires $(T_i)_{i\leq k}$ (Fig. 8, Eq. 11), nous avons alors si $z_{i-1} = T_i^{-1}(z_i)$ ($z_0 = z$, $z_k = x$)

$$\log |J_{T^{-1}}(x)| = \sum_{i=1}^{k} \log |J_{T_i^{-1}}(z_i)| \tag{79}$$

Donc, le déterminant se calcule facilement à condition de pouvoir calculer le déterminant de chacun des transports T_i. C'est le point qui peut être bloquant si l'on ne choisit pas judicieusement le type de transport. Il doit être **suffisamment flexible pour ne pas nuire à l'expressivité du modèle, tout en ayant un déterminant simple à calculer**. Il faut se rendre compte que les déterminants en question sont de très grandes matrices, typiquement d^2 avec $d = O(10^6)$. C'est pour cette raison que dans la littérature ont été étudiés des opérateurs à matrices triangulaires. Le déterminant est alors le produit des d éléments de la diagonale.

S. Mallat nous décrit l'algorithme menant à l'architecture du modèle `Glow` de Kingma et Dhariwal [69]. Le modèle est construit à partir de transports de type:

— **couplage affine**: la variable x de dimension d est scindée en deux vecteurs (x_0, x_1) de dimensions respectives (d_0, d_1) ($d = d_0 + d_1$), et la transformation (inverse) est donnée par

$$T_\theta^{-1}(x) = \left(x_0, e^{s_\theta(x_0)} \odot x_1 + t_\theta(x_0)\right) \tag{80}$$

avec une partie égale à l'identité et l'autre qui emploie le produit d'Hadamard $\odot$ pour que chaque composante de x_1 soit multipliée par un coefficient non-nul

69. Kingma D. P., Dhariwal P., 2018, "*Glow: Generative Flow with Invertible 1x1 Convolutions*", in Bengio S., Wallach H., Larochelle H., Grauman K., Cesa-Bianchi N., Garnett R., eds, NIPS '18 Vol. 31, Advances in Neural Information Processing Systems. Curran Associates, *arxiv:1807.03039*. NDJE. Pour les personnes intéressées par des structures employées dans des modèles à base de Normalizing Flows, voici une revue par Papamakarios G., Nalisnick E., Rezende D. J., Mohamed S., Lakshminarayanan B., 2021, *Normalizing Flows for Probabilistic Modeling and Inference*, Journal of Machine Learning Research, 22, 1, *arxiv:1912.02762*

(l'exponentielle) ne dépendant que de x_0, à laquelle l'on ajoute un biais. Les fonctions s_θ et t_θ dépendent de paramètres, et peuvent être assez compliquées, notamment dans le cas **des réseaux de neurones**. Cet opérateur est simple à inverser ($z = (z_0, z_1)$):

$$T_\theta(z) = \Big(z_0, (z_1 - t_\theta(z_0)) \odot e^{-s_\theta(z_0)}\Big) \tag{81}$$

De plus le déterminant de $J_{T^{-1}}$ est donné par les d_1 composantes du vecteur $e^{s_\theta(x_0)}$, en prenant le logarithme, nous avons

$$\log |J_{T^{-1}}| = \sum_{i=1}^{d_1} (s_\theta(x_0))_i \tag{82}$$

ce qui est simple à calculer.

— **convolution (linéaire)** 1×1. Remarquons que lors d'une transformation affine, une seule partie de x, à savoir x_1, est affectée. Donc, l'idée est d'effectuer des **cascades d'opérations affines** (on parle de couche) en intercalant entre chaque couche une **opération qui mélange les composantes**. Pour ce faire, on utilise une opération linéaire dite de "convolution 1×1" (Fig. 24). L'opérateur agit dans la direction des canaux (le même pour tous les pixels) tel que si (i, j) repère un pixel d'une "imagette" de taille $W \times H \times C$ (*width, height, channels*) alors

$$T_\theta^{-1}(x)_{i,j} = w_\theta \; x_{i,j} = z_{i,j} \tag{83}$$

avec w_θ un opérateur linéaire de mélange le long des canaux que l'on impose inversible tel que

$$T_\theta(x)_{i,j} = w_\theta^{-1} \; z_{i,j} \tag{84}$$

Le déterminant du Jocobien de T_θ^{-1} est égal à [70]

$$\log |J_{T_\theta^{-1}}(x)| = H \times W \times \log |w_\theta| \tag{85}$$

(nb. si C est le nombre de canaux, la complexité du calcul de $\log |w_\theta|$ est $O(C^3)$ que l'on peut ramener si besoin à $O(C)$ par une décomposition LU.)

— **activation normalisation**. Si l'on reprend la notation de l'imagette ci-dessus, on

70. rappel: pour un opérateur linéaire A, $|A|$ signifie $|det(A)|$.

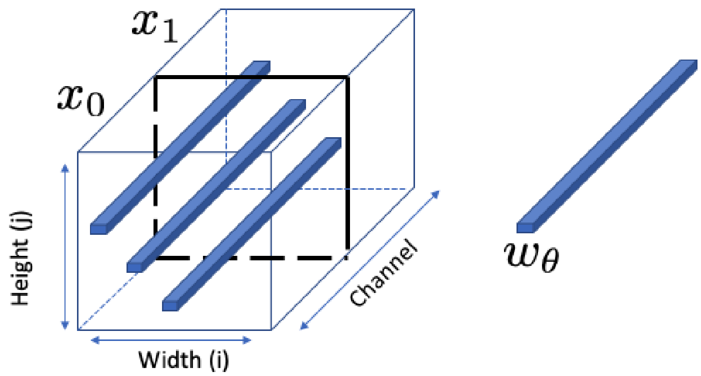


FIGURE 24 – Schématisation de l'opération de convolution 1×1 (vecteur w_θ de dimension C) qui agit sur les canaux identiquement sur les pixels $H \times W$.

effectue sur chaque pixel une normalisation le long des canaux (similaire à la Batch-Norm).

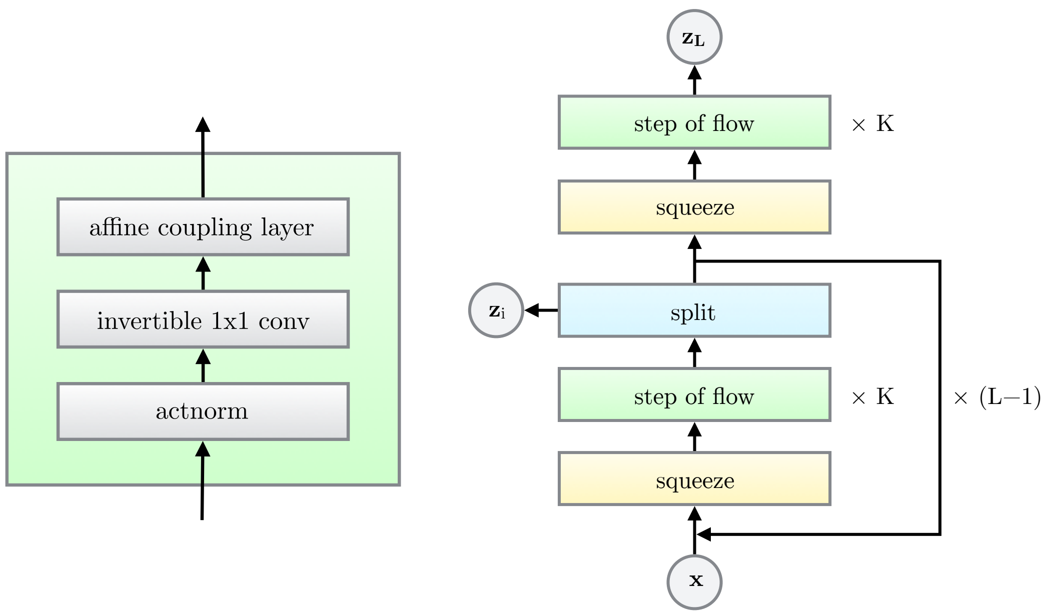
$$z_{i,j} = s_\theta \odot x_{i,j} + b_\theta \tag{86}$$

Le déterminant du Jacobien qui nous intéresse est $H \times W \times \log|s_\theta|$

Donc, l'algorithme `Glow` procède par cascades (couches) que la figure extraite de l'article original retranscrit: on y voit les différentes actions mentionnées ci-dessus constituant un ensemble qu'essentiellement, on répète plusieurs fois donnant une architecture multi-échelle [71]. Le nombre de paramètres est de l'ordre de 44 millions.

S. Mallat nous montre des exemples d'images similaires à celles de la figure 5. NDJE. J'en donne des exemples sur la figure 26. C'est assez remarquable, mais cela n'atteint pas l'état-de-l'art. On sent bien qu'il y a des défauts dans ces images notamment concernant le fond, les cheveux. Si l'on utilise des images d'entrainement de scènes d'intérieur ou d'églises, le constat général est que **les images générées sont de moins bonne qualité** que celles issues de GANs, bien que nous avons vu que la qualité est une chose, mais la diversité en est une autre. Pourquoi la qualité n'est pas optimale? Cela réside dans le fait que les contraintes sur **l'opérateur** T ont pour conséquence qu'il **n'est pas assez riche** (ou

71. NDJE. Certaines opérations sont issues de l'article de Dinh L., Sohl-Dickstein J., Bengio S., 2017, *Density estimation using Real NVP*, in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, https://openreview.net/forum?id=HkpbnH9lx (updated 2023)



(a) One step of flow. (b) Multi-scale architecture.

FIGURE 25 – Architecture du modèle `Glow` (source: voir note de bas de page 69)

Figure 26 – Quelques images issues de `Glow`.

expressif).

Malgré tout, il y a un aspect intéressant de ces modèles, consistant à jouer sur le fait que z **est une variable l'attente qui nous donne la possibilité de manipuler les images** [72]. À partir de deux images (x_0, x_1), nous pouvons trouver les variables correspondantes (z_0, z_1) de l'espace latent (action de $z_i = T^{-1}(x_i)$). Cet espace pour lequel $p(z) \propto \exp(-\|z\|^2/2\sigma^2)$ est muni d'une métrique naturelle: la métrique euclidienne. On peut l'utiliser pour regarder des distances entre des images. Or, si l'on veut obtenir une série d'images "interpolant" entre x_0 et x_1 (Fig. 27) et qui restent des échantillons de la pdf $p_g(x)$, il est facile de les obtenir à partir $z_\varepsilon = z_0\varepsilon + z_1(1-\varepsilon)$ ($\varepsilon \in [0,1]$) par $T(z_\varepsilon) = x_\varepsilon$ (Fig. 28). Toute interpolation linéaire de type $x_0\varepsilon + x_1(1-\varepsilon)$ seront des images, mais selon toute vraisemblance auront des défauts, et ne seront donc pas des échantillons de p_g. Pour bien le faire, il faudrait connaître la géométrie dans cet espace, afin de suivre des géodésiques, ce qui est très compliqué. Notons que les géodésiques dans l'espace latent sont des arcs de cercles (en grande dimension) que l'on a approximé par des droites.

On peut également ajouter des "structures": ex. visage plus ou moins souriant, visage dont la couleur de peau ou des cheveux varient, des visages de personnes plus ou moins jeunes, etc (Fig. 29). Pour cela il nous faut des exemples de la base de données où l'on a mis en forme un critère (label) qui donne par exemple une idée de l'intensité du "sourire" (pas forcément simple à mettre en œuvre). Mettons que l'on ait un label binaire $\ell \in \{0, 1\}$ qui identifie 2 lots d'images $X_0 = \{x \ / \ \ell = 0\}$ et $X_1 = \{x \ / \ \ell = 1\}$. Pour chaque image de ces deux lots, on peut appliquer T^{-1} pour obtenir les variables latentes correspondantes

72. NDJE. vous pouvez aller sur le site https://openai.com/index/glow/ où vous pourrez faire quelques manipulations.

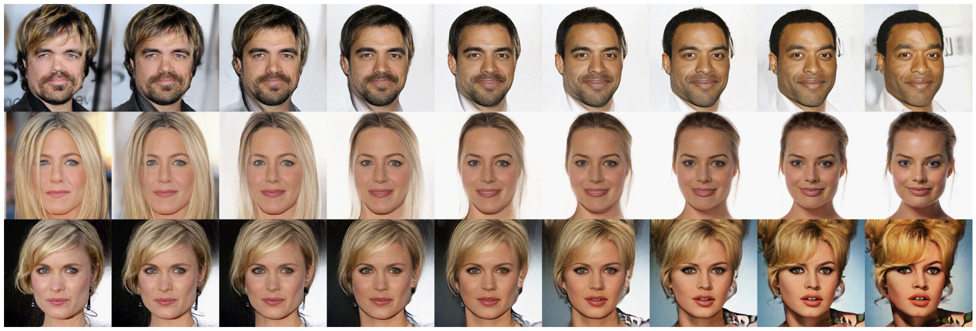


Figure 27 – Quelques images issues de `Glow` par interpolation dans l'espace latent.

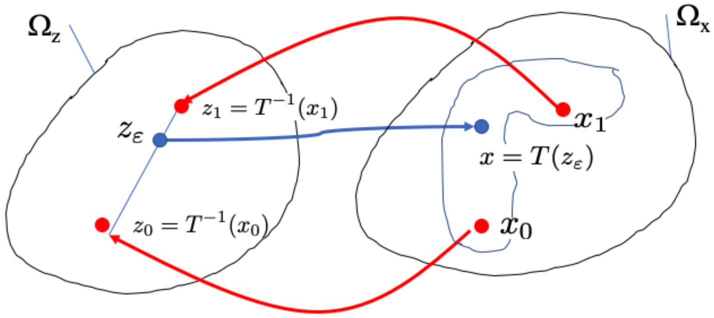


Figure 28 – Schématisation de l'interpolation d'images (voir texte).

Figure 29 – Exemple de transformations (Glow) concernant le sourire d'une personne.

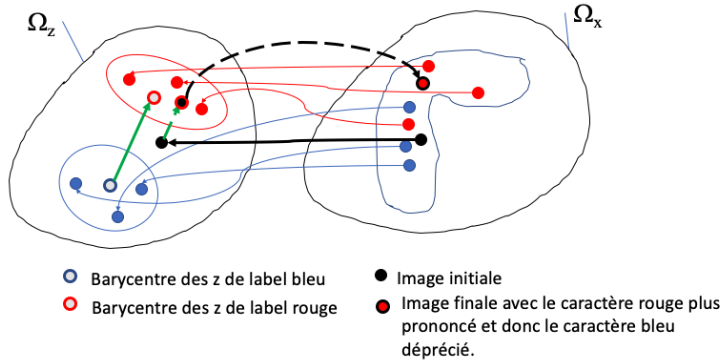


FIGURE 30 – Schématisation de l'addition d'un caractère (d'une structure) tout en dépréciant son contraire. Ici, on distingue le caractère par la couleur rouge et, son contraire par la couleur bleue. En identifiant dans la base d'images celles qui ont le label rouge ou bleu, on peut construire dans l'espace latent deux ensembles via l'action de T^{-1}, dont on peut calculer les barycentres (points gris cerclés de bleu ou rouge) et la direction "bleu vers rouge" (flèche pleine verte). Puis considérant une image (point noir), par l'action de T^{-1} on construit la variable latente correspondante que l'on peut propager dans la direction de l'espace de label rouge (flèche trait-tiré verte, point noir cerclé de rouge). On obtient alors une image avec le label rouge par action de T (point rouge cerclé de noir).

Z_0 et Z_1, avec $Z_i = \{T^{-1}(x) \ / \ x \in X_i\}$. On espère que le label soit tel que les ensembles Z_0 et Z_1 soient séparés. Considérons alors le barycentre z_i de chaque ensemble Z_i, on a donc z_0 qui est représentatif des variables latentes de type $\ell = 0$ (et idem pour z_1). Soit alors une image x de la base de données, on peut alors connaitre $z = T^{-1}(x)$. Si en retour, on aimerait une version de x qui a plutôt le caractère $\ell = 0$, alors on construit la direction ($e_{1,0}$, vecteur unitaire) supportée par le segment (z_1, z_0), et l'on translate z dans cette direction pour obtenir $z_\varepsilon = z + \varepsilon e_{1,0}$. Puis par transport $T(z_\varepsilon) = x_\varepsilon$, on obtient le résultat escompté (Fig. 30).

Ici, nous avons exploré un modèle particulier [73], mais les idées sont assez génériques pour ce type d'approche par Normalizing Flow. À ce stade, **on a un support mathématique qui est mieux maitrisé** par rapport aux GANs, mais comme déjà exprimé, **les transports ne sont pas assez riches** pour modéliser des problèmes complexes. C'est dû aux restrictions sur les déterminants nécessaires pour la mise en pratique. Il nous faut donc revoir la méthode, en particulier, nous allons explorer **les transports continus**.

73. NDJE. voir la note de bas de page 68 pour des notebooks de codes de NF simples.

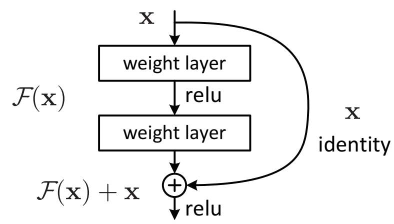


FIGURE 31 – Illustration d'une *skip connection* (source *arXiv:1512.03385*).

5.2 Équation différentielle ordinaire (ODE)

L'approche des Normalizing Flow a introduit une discrétisation du transport T, l'idée est de faire tendre la discrétisation vers un continuum de transformations qui nous mène vers une équation différentielle (ODE pour *ordinary differential equation*). Donc, le passage de z de l'espace latent vers x de l'espace des images, se fait par incrément infinitésimal $dx(t)$ où t est une variable de "temps" (nb. pour les NF, on considère des Δt macroscopiques si l'on veut une analogie). Ainsi, on écrit l'ODE sous la forme

$$\frac{dx_t}{dt} = V_{\theta,t}(x_t) \tag{87}$$

où $V_{\theta,t}$ joue le rôle d'une "vitesse" paramétrée. Une version discrétisée eulérienne donne

$$x_{t+\Delta t} = x_t + V_{\theta,t}(x_t)\ \Delta t \tag{88}$$

En termes de réseaux de neurones où $V_{\theta,t}$ peut être considéré comme une couche, le schéma ressemble à celui d'un ResNet [74], c'est-à-dire un réseau avec *skip connection* (Fig. 31). Inversement, si chaque couche d'un ResNet permet d'obtenir une transformation comme celle de l'équation 88, alors on peut concevoir qu'**un réseau ResNet complet peut être dans la limite le schéma d'un transport associé à une ODE**. C'est l'idée des *Neural ODE* étudiée dans les années 2018-19 [75]. Le champ de la recherche s'est alors penché sur la question de savoir *si tous les réseaux de neurones que l'on entraine peuvent être des*

74. NDJE. Kaiming He et al. (2016), *Deep Residual Learning for Image Recognition*, Conference on Computer Vision and Pattern Recognition. *arXiv:1512.03385*.
75. NDJE. Voir par ex. RTQ Chen et al. (2018) *Neural ordinary differential equations*, Advances in neural information processing systems. *arxiv:1806.07366*

approximations d'équations différentielles? La réponse est *non*, mais l'idée est intéressante et nous allons l'explorer.

Une fois que l'on dispose de $V_{\theta,t}$, la distribution de probabilité de x_t, notée p_t, suit aussi une équation d'évolution. C'est un sujet fondamental en Physique: quelle est l'équation dans l'espace des phases (position-moment) d'une collection de particules? Ce problème donne lieu à l'**équation de Liouville**[76] fondamentale pour comprendre des systèmes dynamiques. Notamment concernant l'évolution de systèmes chaotiques où la prédiction de la trajectoire de chaque particule est illusoire, car la moindre instabilité aussi petite qu'elle soit fait diverger dans le futur des particules proches à un instant t. Cependant, l'approche statistique permet de calculer l'évolution de la densité de probabilité de l'ensemble de particules considérées. C'est le cas typique en météorologie où l'on bruite les données initiales d'une équation de Navier-Stockes afin d'obtenir un ensemble de prédictions sur le temps qu'il va faire dans le futur et sur lesquelles on peut calculer des statistiques.

Comment est reliée p_t à $V_{\theta,t}$? Dans ce contexte comment se calcule "les déterminants" qui nous gênaient dans le cas des Normalizing Flows? Dans un premier temps, existe-t-il une solution à l'ODE (Eq. 87)? La réponse est donnée par le théorème classique suivant:

Théorème 4 ***(Cauchy-Lipchitz)***
Soit le problème suivant où $V(t,x)$ est défini sur un ouvert Ω de $\mathbb{R}\times X$ (X espace vectoriel normé des x) continue de Ω dans X:

$$\frac{dx(t)}{dt} = V(t, x(t)) \qquad \text{avec} \quad x(t_0) = x_0 \quad \text{(condition de Cauchy)}$$

où $(t_0, x_0) \in \Omega$ et sous la condition que V soit localement lipchitzienne sur la variable x, c'est-à-dire que si on note $A \subset \Omega$, alors il existe une constante k telle que

$$\forall (t,x),(t,x') \in A \Rightarrow \|V(t,x) - V(t,x')\| \leq k\|x - x'\|$$

(les champs de vitesses sont réguliers, p.p différentiables avec une dérivée bornée par

76. Joseph Liouville (1809-32) mathématicien (analyse complexe, topologie différentielle, physique mathématique, etc), professeur au Collège de France

> *k).*
>
> *Alors, il existe une solution $x(t)$ dans un intervalle de temps qui contient t_0.*

Démonstration 4. A priori pour calculer $x(t)$, on procèderait par l'intégration suivante:

$$x(t) = x_0 + \int_{t_0}^{t} V(s, x(s))\ ds \tag{89}$$

La question est de savoir s'il existe un x qui satisfait cette équation? Soit l'opérateur T

$$T[x(t)] = x_0 + \int_{t_0}^{t} V(s, x(s))\ ds \tag{90}$$

il nous faudrait obtenir la solution $T[x(t)] = x(t)$ (**point fixe**), ce que l'on pourrait faire à l'aide de l'algorithme itératif simple suivant:

$$x_0(t) = x_0; \qquad x_{k+1}(t) = T[x_k(t)] \tag{91}$$

Pour cela, il faut s'assurer de la propriété de contraction de l'opérateur T, qui se traduit par

$$d(T[x], T[x']) \leq \gamma\ d(x, x') \qquad \text{avec} \quad \gamma < 1 \tag{92}$$

Or, $\forall u \in [t_0, t]$

$$T[x(u)] - T[x'(u)] = \int_{t_0}^{u} \Big(V(s, x(s)) - V(s, x'(s)) \Big)\ ds \tag{93}$$

donc [77]

$$\begin{aligned} \|T[x(u)] - T[x'(u)]\| &\leq \int_{t_0}^{u} \Big\| V(s, x(s)) - V(s, x'(s)) \Big\|\ ds \\ &\leq k \int_{t_0}^{u} \|x(s) - x'(s)\|\ ds \\ &\leq k\ |u - t_0| \sup_{s \in [t_0, u]} \|x(s) - x'(s)\| \\ &\leq k\ |t - t_0| \sup_{s \in [t_0, t]} \|x(s) - x'(s)\| \end{aligned} \tag{94}$$

où on a utilisé la condition Lipschitz de V sur la variable x à s fixé. Comme l'inégalité

77. NDJE. Supposons que X soit notée $\|.\|$.

est vraie pour tout u, on en conclut que

$$\sup_{s\in[t_0,t]} \|T[x(s)] - T[x'(s)]\| \leq k\ |t - t_0| \sup_{s\in[t_0,t]} \|x(s) - x'(s)\| \tag{95}$$

Maintenant, pour obtenir la contractance de l'opérateur T, on doit se fixer t tel que $k|t-t_0| = \gamma < 1$. Sur l'intervalle $[t_0, t]$, on aura bien une solution $x(s)$ obtenue à l'aide de l'algorithme décrit ci-dessus. Le théorème de Banach nous garantit que le point fixe est unique. ∎

Ce que l'on en déduit, c'est que l'on peut construire la solution à l'ODE, mais au prix d'une contrainte assez forte sur le champ de vitesses qui doit être assez régulier pour pouvoir effectuer un transport. On retombe en quelque sorte sur la propriété de difféomorphisme des Normalizing Flows. La question est de savoir si l'on a gagné en expressivité pour faire mieux que ces derniers.

5.3 Équation de Liouville

La question qui demeure est la relation entre la densité de probabilité et le champ de vitesses? On reprend l'image des particules qui se déplacent dans l'espace des phases de la mécanique classique et on écrit une **équation de continuité** sur les probabilités.

Théorème 5 *(**Liouville**)*
Sous les hypothèses du théorème de Cauchy-Lipschitz, l'évolution de $p_t(x)$ est donnée par[a]

$$\frac{\partial p_t(x(t))}{\partial t} + \nabla_x \bullet \Big(p_t(x(t))\ V_t(x(t))\Big) = 0 \tag{96}$$

a. NDJE. p et V sont des fonctions de (t,x) avec x lui-même une fonction de t, la dérivée partielle ∂t agit sur la première variable de (t,x) (et pas sur le t de $x(t)$); et l'opérateur $\nabla_x\bullet$ (divergence) appliqué à un vecteur $A(x)$ se calcule à partir des dérivées partielles sur les composantes du vecteur $x = (x_1,\ldots,x_d)$ selon $\nabla_x \bullet A(x) = \sum_{i=1}^d \frac{\partial A_i}{\partial x_i}(x)$, on reconnait un produit scalaire de $\nabla_x = (\partial/\partial x_1,\ldots,\partial/\partial x_d)$ et de $A = (A_1,\ldots,A_d)$.

Avant d'élaborer une preuve du théorème, voyons ce que cela donne:

$$\nabla_x \bullet \Big(p_t(x(t))\ V_t(x(t))\Big) = p_t(x(t))\Big(\nabla_x \bullet V_t(x(t)\Big) + V_t(x(t) \bullet \nabla_x p_t(x(t)) \tag{97}$$

Le second terme fait intervenir le gradient de la probabilité $p_t(x)$, c'est donc un vecteur de composantes $(\partial p_t(x)/\partial x_1, \dots, \partial p_t(x)/\partial x_d)$ calculé en $x(t)$. Mis dans l'équation dévolution, si l'on ne considère que ce terme uniquement, on a alors affaire à une **équation d'advection** de la *pdf*, comme celle qui décrit la quantité de chaleur ($\propto T$) transportée par un fluide en mouvement: $\partial T \partial t + \vec{v} \bullet \vec{\nabla}_x T = 0$. Donc, ce terme nous dit que **la densité de probabilité suit le champ de vitesse**. Le premier terme, faisant intervenir la divergence de V_t n'est autre que le **terme de Jacobien** des Normalizing Flows. Si l'on a un gradient de la vitesse, cela produit des **phénomènes de compression/décompression du volume dans lequel se trouvent les particules**. Ainsi, une fois que l'on a $V_t(x(t)$ le calcul du "Jacobien" se fait naturellement à travers la divergence.

Démonstration 5. L'idée de la démonstration se fait en suivant la logique de l'évolution d'un lot de particules $(x^{(i)})$ dans un volume $\Omega \subset \Omega_x$, et l'on considère les échanges à sa surface. La conservation de la probabilité va nous fournir une équation comme dans le contexte de fluide incompressible.

Soit $P_t(\Omega)$ la probabilité associée aux $x \in \Omega$:

$$P_t(\Omega) = \int_\Omega p_t(x)\ dx \tag{98}$$

La variation temporelle totale de cette probabilité est donnée non seulement par la variation locale temporelle en chaque point qui se traduit par

$$\frac{\partial P_t(\Omega)}{\partial t} = \int_\Omega \frac{\partial p_t(x)}{\partial t}\ dx \tag{99}$$

et aussi par une contribution de flux. Soit la quantité

$$J_t(x(t)) = p_t(x(t))\ V_t(x(t)) \tag{100}$$

il s'agit de la **densité de courant de probabilité**. Le flux de J à travers l'enveloppe Σ de

Ω s'écrit:

$$Flux = \oint_{\Sigma} J_t(x(t)) \bullet n(x)\ dS \tag{101}$$

avec dS l'élément de la surface Σ et $n(x)$ le vecteur normal à la surface "sortant" au point x. Le théorème de la divergence nous permet alors d'écrire

$$Flux = \int_{\Omega} \nabla_x \bullet J_t(x(t))\ dx \tag{102}$$

Le bilan de la variation totale de probabilité est alors donné par la somme de ces deux contributions

$$\frac{dP(\Omega)}{dt} = \int_{\Omega} \left(\frac{\partial p_t(x)}{\partial t} + \nabla_x \bullet J_t(x(t)) \right) dx \tag{103}$$

La conservation de la probabilité nous dit que $\frac{dP(\Omega)}{dt} = 0$, et cela quelque soit le volume Ω, ce qui implique alors la version locale de l'équation de conservation

$$\frac{\partial p_t(x)}{\partial t} + \nabla_x \bullet J_t(x(t)) = 0 \tag{104}$$

■

5.4 Équation différentielle stochastique (SDE)

Le problème de l'approche par ODE (Eq. 87), c'est qu'elle impose par nature de connaître parfaitement le champ de vitesse $V_t(x)$. Or, en pratique, il y a des erreurs, ne serait-ce que les modèles paramétrés $V_{\theta,t}$ ne sont pas parfaits. De plus, le problème étudié peut avoir un réel phénomène aléatoire intrinsèque. On va donc étudier des équations différentielles stochastiques de type [78]

$$dx_t = V_t(x_t)\ dt + \sigma(x_t)\ dW_t \tag{105}$$

où W_t est un terme de **mouvement brownien**, c'est-à-dire que l'on rajoute au terme de vitesse de l'ODE un **terme de bruit blanc** (toutes les composantes sont des $v.a$ gaussiennes

78. nb. a_t est un raccourci pour $a(t)$ et l'on utilise les notations du théorème de Liouville

indépendantes.). Le modèle physique que l'on peut avoir en tête et celui de la **diffusion** [79]. Donc, si on ne retient que le terme brownien, les particules au cours du temps explorent un domaine de plus en plus grand. Si l'on ajoute le terme de vitesse, les trajectoires vont certes suivre le flot, mais elles **ne sont plus inversibles**. La connaissance individuelle des trajectoires n'a plus de sens.

Par contre, nous pouvons avoir une description probabiliste du problème (solution faible) en trouvant l'équation d'évolution de la densité de probabilité qui va remplacer l'équation de Liouville, et nous verrons qu'elle est inversible. En fait, **le terme de la mesure de Wiener va apporter de la flexibilité à la modélisation de transports inversibles sur les distributions de probabilités**. Voici donc l'équation d'évolution (1913):

Théorème 6 ***(Fokker-Planck)***

$$\frac{\partial p_t(x_t)}{\partial t} + \nabla_x \bullet \Big(p_t(x_t)\ V_t(x_t)\Big) - \Delta_x\Big(\frac{1}{2}\sigma^2(x_t)p_t(x_t)\Big) = 0 \tag{106}$$

Par rapport à l'équation de Liouville, nous avons un terme supplémentaire donné par le Laplacien $\Delta_x = \sum_{i=1}^d \partial^2/\partial x_i^2$. Le courant de probabilité $J_t(x_t)$ (Eq. 100) qui ne comprenait qu'un **terme d'advection**, va en être modifié par la **diffusion des particules**. Les lois de la diffusion dans la matière ont été établies par Adolf Fick (1855) et sont les analogues des lois de la chaleur de Joseph Fourier établies en 1822. Le courant de densité de matière est proportionnel au gradient de cette densité (signe négatif). Les particules vont des zones de grandes densités vers les zones de faibles densités, c'est un phénomène d'homogénéisation pour lequel *in fine*, on atteint un état stationnaire.

79. NDJE. L'équation de diffusion est du type

$$D\Delta_x\phi(x,t) = \frac{\partial\phi(x,t)}{\partial t}$$

avec D un coefficient exprimé en $m^2.s^{-1}$. Typiquement, cela donne des solutions gaussiennes $\phi(x,t) \propto e^{-x^2/(4Dt)}$, où il y a un un phénomène d'étalement de la densité de particules au cours du temps. Un exemple: pour la vapeur d'eau dans l'air (25^o C, 1 atm), le coefficient de diffusion vaut $D \approx 2\ 10^{-5}\ m^2.s^{-1}$, donc il faut à peu près 3 minutes pour qu'une molécule de vapeur se déplace dans l'air de 1 mètre. En réalité, il faut tenir compte des termes de convections. Mais cela est une autre histoire...

Théorème 7 ***(Fick)***

$$J_t^{diffusion}(x_t) = -\frac{1}{2}\sigma^2(x_t)\ \nabla_x p_t(x_t) \tag{107}$$

Donc, comme d'une manière générale, notant $J^{tot} = J^{advection} + J^{diffusion}$, nous avons le bilan suivant

$$\frac{\partial p_t(x_t)}{\partial t} + \nabla_x \bullet J_t^{tot}(x_t) = 0 \tag{108}$$

quand on injecte l'expression de $J^{diffusion}$, et notant que $\nabla_x \bullet \nabla_x = \Delta_x$, nous obtenons l'équation de Fokker-Planck.

Fokker-Flanck en 1D

Procédons à une justification en 1D de l'équation de Fokker-Planck qui permet de rendre accessible les différentes contributions. Nous allons nous focaliser ($V_t = 0$) sur le mouvement brownien se caractérise par des accroissements stationnaires gaussiens et la variance est proportionnelle au temps. La première chose est que l'on discrétise le problème, c'est-à-dire que l'on approxime le mouvement brownien par une marche aléatoire, ce qui n'est pas sans rappeler les études sur les chaînes de Markov [80]. Sur un pas de temps Δt, la particule située originellement en x, peut effectuer un saut ε soit dans le sens positif ou négatif (eg. $\varepsilon = \pm\Delta x$) avec la même probabilité 1/2 (pas de vitesse globale). Au bout de N pas de temps, soit au bout d'une durée $T = N\Delta t$, si la position initiale est $x_0 = 0$ alors la particule se trouve à une position x_T telle que

$$x_T = \sum_{k=1}^{N} \varepsilon_k \tag{109}$$

Les $(\varepsilon_k)_{k\leq N}$ sont des $v.a$ (de Rademacher) indépendantes. Donc, quand N tend vers l'infini, nous avons le théorème central limite [81] qui nous dit que la distribution de x_T va devenir gaussienne. La variance est simplement

$$Var(x_T) = N \times \Delta^2 x = T \times \frac{\Delta^2 x}{\Delta t} \tag{110}$$

80. NDJE. Voir Cours 2024 Sec. 8.1, Cours 2023 Sec. 6.4.2.2.
81. NDJE. Cours 2022 Sec. 5.5 Th 10.

Est-ce que cela converge vers un mouvement brownien? On aimerait que

$$Var(x_T) \xrightarrow[(\Delta x,\Delta t)\to(0,0)]{} T \times \sigma^2 \qquad \text{soit} \quad \frac{\Delta^2 x}{\Delta t} \to \sigma^2 \tag{111}$$

Donc, il nous faut ajuster la marche aléatoire en ce sens.

Concernant la probabilité que la particule se trouve au point $x(t+\Delta t)$, $p(x = x(t+\Delta t))$ notée $p(x, t+\Delta t)$, elle se décompose de la façon suivante en considérant le saut des particules venant de la gauche et de la droite de x:

$$p(x, t+\Delta t) = \frac{1}{2}\, p(x+\Delta x, t) + \frac{1}{2}\, p(x-\Delta x, t) \tag{112}$$

Un développement de Taylor au premier ordre en t du membre de gauche, et au second ordre en x pour le membre de droite donne (termes dominants, $\Delta x, \Delta t$ tendant vers 0):

$$\Delta t \times \frac{\partial p(x,t)}{\partial t} = \frac{1}{2}\Delta^2 x \times \frac{\partial^2 p(x,t)}{\partial x^2} + O(\Delta^3 x) \Longrightarrow \frac{\partial p(x,t)}{\partial t} = \frac{1}{2}\sigma^2\ \Delta_x p(x,t) + \underbrace{O(\sigma^2 \Delta x)}_{\to 0} \tag{113}$$

Ce qui justifie bien le terme de Fritz de l'équation de Fokker-Planck.

6. Séance du 12 Fév.

L'équation de Fokker-Planck, partiellement établie [82] dans la séance précédente est une équation différentielle ordinaire (ODE) de l'évolution d'une densité de probabilité, associée à une équation différentielle stochastique (SDE) du mouvement d'une particule. Cette ODE sur $p_t(x)$ a pour conséquence importante son échantillonnage par une diffusion de Langevin. Nous allons aborder cette méthode, avant de passer aux algorithmes de score-diffusion qui introduisent une modification du niveau de bruit (parfois associé à une modification de la température) et donnent des résultats les plus performants du moment.

82. NDJE. S. Mallat revient en début de cette séance sur la démonstration en changeant les notations, je l'ai rassemblée à la fin de la séance précédente.

6.1 Échantillonnage par l'équation de Langevin

6.1.1 Trouver la vitesse et la variance

On se place dans le cadre d'une probabilité de Gibbs (Sec. 3.1) que l'on écrit sous la forme

$$p(x) = Z^{-1} e^{-\beta U(x)} \tag{114}$$

où β est selon une réminiscence du facteur de Boltzmann l'inverse d'une sorte de température ($\beta = 1/T$), diminuer β correspond au cas où l'on augmenterait la température, c'est-à-dire, on augmenterait l'agitation thermique. On aimerait obtenir des échantillons typiques[83] de cette *pdf*. Ces échantillons vont avoir tendance à correspondre au minimum d'énergie (U).

Soit un point initial $z = x_0 \sim p_0$ où p_0 peut être n'importe quelle distribution *a priori* (ex. une loi normale). Il nous faut définir un transport pour obtenir un échantillon de $p(x)$. Ce transport est continu, indexé par le temps t. On cherche une SDE qui permet de réaliser ce transport (σ est une constante ici):

$$dx = V_t(x_t)\ dt + \sigma\ dW_t \tag{115}$$

La question est comment ajuster V_t et σ pour que lorsque t évolue on finisse par évoluer dans le support de $p(x)$ (à la façon d'une chaîne de Markov). En fait, on aimerait que $p(x)$ **soit un invariant du transport**. Pour ce faire, nous allons imposer la **stationnarité** de l'équation de Fokker-Planck pour p:

$$\frac{\partial p(x)}{\partial t} = -\nabla_x \bullet \Big(p(x)\ V_t(x) \Big) + \frac{\sigma^2}{2}\ \Delta_x p(x) = 0 \tag{116}$$

Rappelons que concernant les échantillons $x \sim p$, $U(x)$ doit être petit, donc si l'on procède d'une évolution par descente de gradient, la direction d'évolution des x_t est fixée (opposée du gradient), donc un bon candidat pour $V_t(x)$ est

$$V_t(x) = -\nabla_x U(x) = \beta^{-1}\ \underbrace{\nabla_x \log p(x)}_{score} \tag{117}$$

83. NDJE. ex. Cours 2022 Sec. 6.4

Ainsi, comme $\nabla_x \log p(x) = (\nabla_x p(x))/p(x)$ et $\nabla_x \bullet \nabla_x = \Delta_x$, il vient

$$\left(-\beta^{-1} + \frac{\sigma^2}{2}\right)\Delta_x p(x) = 0 \Rightarrow \sigma = \sqrt{2\beta^{-1}} \tag{118}$$

Donc, le problème d'échantillonner $p(x)$ peut se décliner suivant le système SDE/ODE d'évolution de $x_t = x(t)$ et de $p_t(x_t) = p(t, x(t))$ au cours du temps ($t = 0$, $x(0) = x_0$ et $p(0, x) = p_0(x)$):

$$\begin{cases} dx_t & = -\nabla_x U(x_t)\ dt + \sqrt{2\beta^{-1}}\ dW_t \\ \dfrac{\partial p_t(x_t)}{\partial t} & = \nabla_x \bullet \Big(p_t(x_t)\ \nabla_x U(x_t)\Big) + \beta^{-1}\Delta_x p_t(x_t) \end{cases} \tag{119}$$

avec [84] le score de p s'écrivant $s(x) = \nabla_x \log p(x) = -\beta \nabla_x U(x)$.

6.1.2 Convergence

Les questions sont de savoir si partant d'une condition initiale (p_0), on converge bien vers $p(x)$ (invariant du transport) et a-a-t-on unicité de la convergence? Cela se démontre avec des conditions relativement faibles, notamment lorsque $U(x)$ **tend vers l'infini quand** $\|x\|$ **tend vers l'infini**, c'est-à-dire la probabilité devenant très petite sur les bords, il y a une sorte de confinement dans les zones de basses énergies. On montre d'une manière générale que [85]

$$\frac{dD_{KL}(p\|p_t)}{dt} < 0 \tag{120}$$

et si on ajoute la contrainte sur $U(x)$, alors que **il y a convergence de p_0 vers p, mais elle peut se faire à un rythme qui peut être très lent**.

Pourquoi? On peut se représenter le cas de figure suivant (Fig. 32) où $U(x)$ présente 2 modes avec un plus profond que l'autre: si p_0 est concentrée autour du mode le plus profond, les x_t sont confinés dans ce mode, car le terme de vitesse de l'ODE est quasi nul; et ne compte que le terme brownien. Pour pouvoir explorer l'ensemble du profil de $U(x)$ et notamment le second minimum, il faut pouvoir faire un saut pour passer la barrière de

84. NDJE. attention, il s'agit du score de la *pdf* p, donc pas d'indice t.
85. NDJE. par exemple A. R. Plastino et al. (1997), *Minimum Kullback entropy approach to the Fokker-Planck equation*, Phys. Rev. E 56, 3927, https://www.researchgate.net/publication/235474425_Minimum_Kullback_entropy_approach_to_the_Fokker-Planck_equation

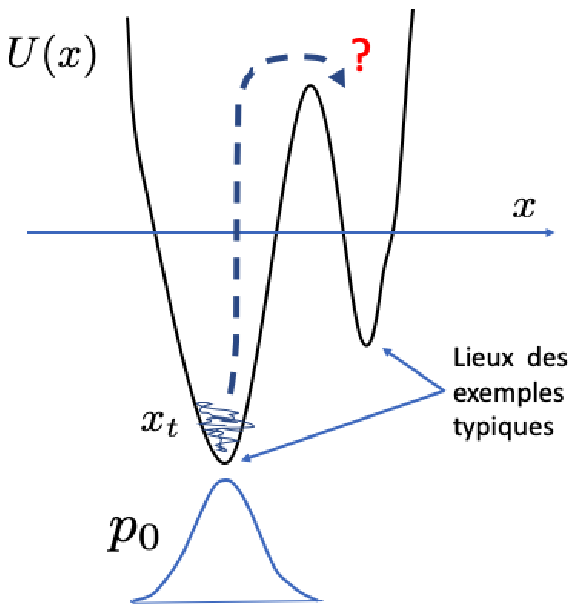


FIGURE 32 – Cas d'une énergie $U(x)$ avec deux modes et une distribution initiale p_0 centrée sur le mode le plus profond. Comment sauter la barrière de potentielle?

potentiel. Or, cela ne peut s'envisager que si le terme brownien à une grande variance (β^{-1} grand), c'est-à-dire à haute température. Ce qui induit un fort bruitage de la descente de gradient, et donc des trajectoires x_t de grandes variances, ce qui ne permet pas de les concentrer dans les lieux typiques de la $p(x)$. Finalement, **il n'y a pas de bons choix pour le facteur β (ou la température) du mouvement brownien**.

Ce qui a été fait expérimentalement, c'est d'effectuer ce que l'on appelle un recuit simulé (*simulated annealing*). C'est-à-dire de changer la température (ou β), donc le terme σ du mouvement brownien, en fonction du temps:

$$dx_t = -\nabla_x U(x_t)\ dt + \sqrt{2\beta_t^{-1}}\ dW_t \tag{121}$$

Au départ, on prend une température assez grande pour pouvoir élargir l'exploration des trajectoires, puis progressivement, on abaisse la température. On utilise aussi l'échange de répliques (voir discussion de la section 3.2) de chaines de Langevin obtenues avec différentes valeurs de β_0. Ce sont des algorithmes assez complexes qui nécessitent **un ajustement de la température pour à la fois bien échantillonner les lieux typiques tout**

en permettant l'exploration les différents modes.

6.1.3 Discrétisation d'Euler-Maruyama

Dernier point, en pratique, il nous faut discrétiser les équations différentielles. Notamment, en utilisant la **discrétisation d'Euler-Maruyama**[86] (σ est indépendant du temps ici):

$$x_{t+\Delta t} = x_t + V_t(x_t)\Delta t + \sigma\sqrt{\Delta t}\ \xi_t \qquad \xi_t \sim \mathcal{N}(0,1) \tag{122}$$

La discrétisation introduit une erreur et donc le transport de p_0 ne correspond pas parfaitement à p, ou autrement dit les échantillons x_t explorent un peu autour des lieux typiques de $p(x)$. Ceci dit, il y a des algorithmes correcteurs comme celui de Langevin *Metropolis-adjusted* (connu sous le nom `MALA`), mais ils sont assez lents dès que l'énergie n'est pas convexe. Malgré tout, ces algorithmes ont été utilisés pour échantillonner des probabilités de Gibbs.

6.2 Génération par Score Diffusion

Une chose que l'on retient de la méthode de **diffusion de Langevin**, c'est que dans le transport qui mène à p, **la condition initiale p_0 est oubliée** grâce au mouvement brownien. Mais cela se fait avec une **convergence lente**, et il faut pouvoir calculer (Eq. 117) $\nabla_x \log p(x)$ c'est-à-dire pouvoir **disposer du score** de $p(x)$. Ce score pose deux problèmes: 1) il faut connaître $U(x)$, et 2) plus critique, les lieux typiques de $p(x)$ forment une variété de $\mathbb{R}^d$, donc le gradient a des singularités, ce qui rend son **calcul très instable**. Ce cas de figures est notamment rencontré dans le cas des images.

Les algorithmes de score-diffusion vont aborder ces problèmes par une **régularisation** de p. D'une manière générale, une façon simple d'effectuer une telle régularisation d'une fonction, consiste à la moyenner par **convolution avec une gaussienne**. La convolution agit en l'occurrence dans l'espace des probabilités, qu'en est-il du côté des x_t? En fait, cela revient à **ajouter du bruit blanc**. Ainsi, ces algorithmes sont centrés sur l'ajout de bruit

86. Gisiro Maruyama (1916-86). NDJE. Un exemple de résolution de l'équation d'Ornstein–Uhlenbeck par cette technique de discrétisation est donné dans un notebook élaboré pour le cours de l'année 2024 et mis à jour[87].

pour régulariser les probabilités. Cela nous permet de calculer le transport de données noyées par le bruit, vers des échantillons de p.

6.2.1 Équation de Ornstein-Uhlenbeck

NDJE. L'étude a également été faire dans le cours de 2024 Sec. 9.1, voir également la note de bas de page 23.

Nous allons aborder la méthodologie de la diffusion en ayant à l'esprit les Normaiizing Flows (Sec. 4.5) pour lesquels nous avons vu que les transports T sont inversibles (voir la figure 8). Et plus précisément, nous avons vu (Sec. 4.5) qu'il est naturel de construire T à partir de T^{-1}, c'est-à-dire de partir des données de p pour les transformer en échantillons de p_0 beaucoup plus simple (ex. bruit blanc). Il s'agit de la partie du processus notée **forward diffusion** et, on peut se référer à la figure 10 pour se faire une idée. En partant d'une distribution p_0 (ici, il s'agit de p_{data} ou p) dont x_0 est un échantillon, ajouter du bruit se fait à l'aide de l'équation **Ornstein-Uhlenbeck**

Théorème 8
Soit l'équation Ornstein-Uhlenbeck suivante:

$$dx_t = -\kappa\, x_t\; dt + \sigma\, dW_t \qquad x(t=0) = x_0 \tag{123}$$

La solution s'exprime comme suit

$$x_t = e^{-\kappa t}\, x_0 + \sigma \left(\frac{1-e^{-2\kappa t}}{2\kappa}\right)^{1/2} z, \qquad z \overset{iid}{\sim} \mathcal{N}(0, Id) \tag{124}$$

Démonstration 8.
La démonstration procède par un changement de variable $y_t = e^{\kappa t} x_t$ qui satisfait à l'équation de pur bruit blanc

$$dy_t = e^{\kappa t}\sigma\, dW_t \Rightarrow y_t = y_0 + \sigma \int_0^t e^{\kappa u}\, dW(u) \tag{125}$$

La solution y_t fait donc intervenir une intégrale (somme) sur des incréments $dW(u)$ du brownien qui sont des $v.a$ indépendantes gaussiennes ($dW(u) \sim \mathcal{N}(0,1)$). Le résultat (cf. l'intégrale) est donc une $v.a$ gaussienne. Quelle est sa variance [88]?

$$Var(y_t) = \sigma^2 \int_0^t e^{2\kappa u} du = \sigma^2 \left(\frac{e^{2\kappa t} - 1}{2\kappa} \right) \tag{126}$$

Donc,

$$y_t = y_0 + \sigma \left(\frac{e^{2\kappa t} - 1}{2\kappa} \right)^{1/2} z, \qquad z \overset{iid}{\sim} \mathcal{N}(0, Id) \tag{127}$$

ce qui nous donne bien la solution quand on fait le changement de variable inverse $x_t = e^{-\kappa t} y_t$. **On remarque "l'oubli de la distribution initiale" et la convergence se fait de manière exponentielle vers le bruit blanc gaussien de variance** $\sigma/\sqrt{2\kappa}$.

Dans la suite, on prend le jeu de paramètres suivant:

$$\kappa = 1, \quad \sigma = \sqrt{2} \qquad \text{(paramètres normalisés)} \tag{128}$$

afin d'obtenir la formulation de Eq 14. ■

Concernant l'expression de $p_t(x_t)$ qu'elle est-elle? Pour cela, il faut se souvenir de deux résultats: soit les $v.a$ (X, Z, X_1, X_2) avec leurs *pdf* correspondantes et α un réel, nous avons

$$\begin{aligned} Z = \alpha X &\Rightarrow p_z(z) = \alpha^{-1} p_x(\alpha^{-1} z) \\ X = X_1 + X_2 &\Rightarrow p_x(x) = \int_{-\infty}^{\infty} p_{x_1}(u) p_{x_2}(x-u) du = (p_{x_1} * p_{x_2})(x) \end{aligned} \tag{129}$$

Ainsi à partir de l'équation 8, l'expression de p_t est donnée pas [89]

$$p_t = \left[e^t \, p_0(e^t \, .) \right] * g_{\sigma_t} \tag{130}$$

avec $g_{\sigma_t} = \mathcal{N}(0, \sigma_t^2)$ et $\sigma_t^2 = 1 - e^{-2t}$. **Ce qui se traduit par une compression de** p_0 **suivit d'un lissage gaussien**. On aurait pu aussi partir de p_0, et la convoluer avec une gaussienne

88. NDJE. pensez à $z = \sum_i a_i z_i$ avec $z_i \sim \mathcal{N}(0,1)$: la moyenne est nulle et la variance est $\sum_i a_i^2$.
89. NDJE. $p_t(x) = \int_{-\infty}^{\infty} e^t \, p_0(e^t \, u) g_{\sigma_t}(x-u) du = \int_{-\infty}^{\infty} p_0(v) g_{\sigma_t}(x - e^{-t} v) dv$.

dilatée, ce qui traduit bien la dilution des images initiales en un bruit blanc. Ce processus efface tous les détails de la distribution initiale. À l'inverse, on retrouvera la distribution initiale p_0 en partant d'une distribution à un instant t, soit p_t.

6.2.2 Équation de diffusion inverse

Quelle est l'expression du transport T qui agit lors de la génération d'échantillons x de $p(x)$ en partant d'échantillons de la distribution $p_z(z) = p_0(z)$ ($p_z = \mathcal{N}(0, Id)$)? Pour ce faire, il nous faut inverser T^{-1}. On peut se référer à la figure 10 (backward/inverse). En partant [90] de $x_0 \sim p_0$ par diffusion (T^{-1}), au bout d'un temps T suffisamment grand, on obtient $x_T \approx x_\infty$ que l'on considère totalement dominé par le terme de bruit, c'est-à-dire que c'est un échantillon de $\mathcal{N}(0, Id)$. L'opération T quant à elle, part d'un échantillon de bruit blanc pour obtenir un échantillon de $p(x)$.

Soit $y_t = x_{T-t}$ ($t \in [0, T]$), on aimerait que y_0 soit un échantillon de $p(x)$. Rappelons que l'on ne peut demander que $y_0 = x_0$, car le processus brownien est irréversible. Comme on raisonne sur les distributions de probabilités, c'est l'équation de Fokker-Planck (Th. 6) qu'il nous faut. Dans notre cas de coefficients normalisés, l'équation d'évolution de x_t s'écrit (nb. dans ce cas p_0 est la *pdf* la distribution des données p_{data} ou p, tandis que p_T est celle du bruit blanc):

$$\frac{\partial p_t(x_t)}{\partial t} = -\nabla_x \bullet \Big(-x_t \, p_t(x_t)\Big) + \Delta_x p_t(x_t) \tag{131}$$

elle est associée à

$$dx_t = -x_t \, dt + \sqrt{2} \, dW_t \tag{132}$$

Si l'on note q_t la distribution de probabilité associée à y_t, on aimerait que $q_t = p_{T-t}$, alors l'équation d'évolution s'écrit (nb. dans ce cas q_0 est la *pdf* du bruit blanc, et q_T est la distribution des données p_{data} ou p):

$$\frac{\partial q_t(y_t)}{\partial t} = -\nabla_y \bullet \Big(y_t \, q_t(y_t)\Big) - \Delta_y q_t(y_t) \tag{133}$$

Si on veut traduire cette équation de Fokker-Planck en termes d'évolution de y_t, le premier

90. NDJE. attention notez que $p_0 = p_{data}$.

terme (*drift*) a le signe opposé, ce qui ne pose pas de soucis, **le seul hic vient du changement de signe du second terme ($-\Delta$)**. Ce terme aurait tendance à inverser le processus brownien, mais comment à partir du bruit peut-on reconstruire une image nette (ex. celle d'un visage)? Utilisons un exemple en traitement du signal: si l'on convolue une fonction f par une gaussienne g, on peut voir le résultat dans le domaine de Fourier:

$$h(u) = (f * g)(u) \to \hat{h}(\omega) = \hat{f}(\omega)\hat{g}(\omega) \tag{134}$$

Or, $\hat{g}(\omega)$ est une gaussienne, donc la convolution atténue fortement toutes les hautes fréquences de f, et même si $\hat{g}(\omega)$ n'est pas nulle, l'inversion devient très instable à haute fréquence, donc on se dit que retrouver f est quasi-impossible. C'est notre cas *a priori* pour obtenir le second terme de l'équation d'évolution de y_t, serait-on dans une impasse? Ce qui va nous sauvez, c'est que l'on veut une inversion en moyenne particule par particule. L'étude de la diffusion-inverse a été faire par Brian D.O. Anderson en 1982 [91]

Théorème 9 ***(reverse diffusion, Anderson 1982)***
L'équation qui régit $y_t = x_{T-t}$ est donnée par

$$dy_t = \Big[y_t + (1+\alpha)\underbrace{\Delta_y \log q_t(y_t)}_{score}\Big]\, dt + \sqrt{2\alpha}\, dW_t \tag{135}$$

Démonstration 9.
Essayons la chose suivante: remplaçons le terme $-\Delta_y q_t(y_t)$ selon

$$\begin{aligned} -\Delta_y q_t(y_t) &= -(1+\alpha)\Delta_y q_t(y_t) + \alpha\Delta_y q_t(y_t) \\ &= -\nabla_y \bullet \Big(q_t(y_t)\,(1+\alpha)\nabla_y \log q_t(y_t)\Big) + \alpha\Delta_y q_t(y_t) \end{aligned} \tag{136}$$

Donc, l'équation de Fokker-Planck peut s'écrire

$$\frac{\partial q_t(y_t)}{\partial t} = -\nabla_y \bullet \Big(q_t(y_t)\,[y_t + (1+\alpha)\nabla_y \log q_t(y_t)]\Big) + \alpha\Delta_y q_t(y_t) \tag{137}$$

91. NDJE. Brian D.O. Anderson (1982), *Reverse-time diffusion equation models* Stochastic Processes and their Applications, Volume 12, Issue 3, Pages 313-326.

avec un terme de *drift* modifié, et une nouvelle variance

$$V_t = y_t + (1+\alpha)\Delta_y \log q_t(y_t) \qquad \frac{\sigma^2}{2} = \alpha \tag{138}$$

Maintenant que l'on a un terme $+\Delta_y$, l'équation d'évolution de y_t peut être écrite par analogie au cas de dx_t selon

$$dy_t = \left[y_t + (1+\alpha)\underbrace{\Delta_y \log q_t(y_t)}_{score}\right] dt + \sqrt{2\alpha}\, dW_t \tag{139}$$

Notons que la figure 10 montre le cas où $\alpha = 1$ avec également une autre définition du sens de la flèche du temps qui peut varier d'une publication à l'autre [92].

Notons que **le terme de drift (V_t) modifié permet de guider les trajectoires des particules** pour qu'elles adoptent la bonne distribution p_{data} quand $t = T$. C'est l'idée de Langevin de **donner à la vitesse la composante du score** (Sec. 6.1.1). ■

Donc depuis 1982, la communauté savait que l'on pouvait inverser une équation de diffusion, tout le problème résidait dans le calcul du **score**. Notons qu'il s'agit du score de $q_t = p_{T-t}$ et non celui de $p = p_{data}$, et de ce fait, il est un peu plus agréable, car il a été **régularisé par une gaussienne**. Mais, il reste que l'on ne connait pas la distribution p. C'est dans ce contexte que Yang Song [93] et al. en 2021 ont donné la pièce manquante en permettant de **calculer le score à l'aide d'un réseau de neurones profond**. Il faut cependant **beaucoup de données**. S. Mallat nous dit que ce résultat fut une très grande surprise, car le score est une fonction en très grande dimension. La technique mise en œuvre est celle d'un **débruitage**.

92. NDJE. Sur le slide présenté de S.Mallat, la définition est celle prise dans ce paragraphe, avec B remplaçant W et y_t apparait sous sa forme x_{T-t}.

93. NDJE. Yang Song et al. (2021), *Score-Based Generative Modeling through Stochastic Differential Equations* ICLR 2021 *arxiv:2011.13456*, et David McAllester (2023) *On the Mathematics of Diffusion Models arxiv:2301.11108*

6.2.3 L'usage d'un débruitage

Une première chose que l'on effectue, concerner une renormalisation des variables x_t de l'équation d'Ornstein-Uhlenbeck (Th. 8) pour faire disparaître le facteur $e^{-\kappa t}$. *NDJE. on reprend la définition de y_t de la démonstration du théorème, et l'on la note x_{σ_t}.* Donc, en reprenant l'équation 127, on peut écrire (x_0 est l'image d'origine que l'on note x sans dépendance en t):

$$x_{\sigma_t} = x + z \qquad z \overset{iid}{\sim} \mathcal{N}(0, \sigma_t^2), \qquad \sigma_t = \sigma \left(\frac{e^{2\kappa t} - 1}{2\kappa} \right)^{1/2} \tag{140}$$

On utilise par la suite les paramètres normalisés (Eq. 128) qui donnent l'expression de l'écart standard du bruit z, $\sigma_t = \sqrt{e^{2t} - 1}$. La variable aléatoire x_{σ_t} a une densité de probabilité notée $p_{\sigma_t}(x_{\sigma_t})$.

On se retrouve alors avec un problème classique [94] d'une variable x bruitée par un bruit blanc. Le problème que l'on se pose est de trouver un **estimateur** $\hat{x}(x_\sigma)$ **de** x qui enlève au mieux le bruit. Une des façons de trouver cet estimateur, c'est d'utiliser l'**erreur quadratique**

$$\min \left\{ \mathbb{E}_{x,z} \left[\| \hat{x}(x_{\sigma_t}) - x \|^2 \right] \right\} \tag{141}$$

Dans ce contexte, il y a un résultat, à plusieurs reprises redécouvert et reformulé, de M. C. K. Tweedie de 1947, H. Robbins en 1956 qui reconnu le travail de Tweedie, K. Miyasawa en 1961 et plus récemment par M. Raphan et E. P. Simoncelli en 2011 [95]. Ce résultat est le suivant

> **Théorème 10**
> *L'estimateur optimal est donné par*
>
> $$\hat{x}(x_{\sigma_t}) = x_{\sigma_t} + \sigma_t^2 \nabla_{x_{\sigma_t}} \log p_{\sigma_t}(x_{\sigma_t}) \tag{142}$$

Ce qui signifie que le débruitage optimal est obtenu en dirigeant la descente de gradient par le score de la distribution de probabilité du signal bruité.

94. NDJE. Voir aussi Cours 2024 Sec. 9.2, Cours 2021 Sec 2.2.1 (Figure 3).
95. NDJE. Voir par ex. https://www.cns.nyu.edu/pub/lcv/raphan07b.pdf.

Démonstration 10.
Pour alléger, on enlève l'indice du temps t. Comme $x_\sigma = x + z$ est une somme de deux $v.a$ indépendantes, sa densité de probabilité est la convolution de p_x et p_z

$$p_\sigma(x_\sigma) = (p_x * p_z)(x_\sigma) = \int p_x(x) p_z(x_\sigma - x)\, dx \tag{143}$$

Comme p_z est une gaussienne de moyenne nulle et de variance σ^2, il vient

$$\begin{aligned}\nabla_{x_\sigma} p_\sigma(x_\sigma) &= \int p_x(x) p_z(x_\sigma - x) du = \int p_x(x) \left(-\frac{x_\sigma - x}{\sigma^2}\right) p_z(x_\sigma -)\, dx \\ &= \frac{1}{\sigma^2} \int (x - x_\sigma) p_x(x) p_z(x_\sigma - x)\, dx \end{aligned}\tag{144}$$

Or, remarquons que $p_z(x_\sigma - x)$ n'est autre que la probabilité conditionnelle $p(x_\sigma | x)$. Il vient alors en notant $p(x, x_\sigma)$ la probabilité jointe de x et x_σ

$$\sigma^2 \nabla_{x_\sigma} p_\sigma(x_\sigma) = \int (x - x_\sigma) p_x(x) p(x_\sigma | x)\, dx = \int (x - x_\sigma) p(x, x_\sigma)\, dx \tag{145}$$

Or $p(x, x_\sigma) = p(x | x_\sigma) p(x_\sigma)$, il vient alors

$$\sigma^2 \nabla_{x_\sigma} \log p_\sigma(x_\sigma) = -x_\sigma + \int x\, p(x | x_\sigma)\, dx = -x_\sigma + \mathbb{E}_x[x | x_\sigma] \tag{146}$$

De plus, si l'on cherche une constante μ qui estime au mieux x en erreur quadratique alors

$$\min \left\{ \mathbb{E}_x \left[\|x - \mu\|^2 \right] \right\} \Rightarrow \mu = \mathbb{E}_x[x] \tag{147}$$

On peut donc identifier à x_σ fixé $\mathbb{E}_x[x | x_\sigma]$ à $\hat{x}(x_\sigma)$. ■

Ainsi, **pour calculer le score, il nous faut obtenir un débruiteur optimal**. Pour cela, on utilise un réseau de neurones profond comme celui (U-Net) de la figure 11. Considérant ce réseau, les données en entrée sont des images x_σ bruitées dont on connait les originales x, et en utilisant la fonction de coût $\|x - x_\sigma\|^2$ à minimiser, nous obtenons l'estimateur $\hat{x}$ optimal. Et finalement, $(\hat{x} - x)/\sigma$ fournit le score de p_σ. **Il nous faut cependant beaucoup d'images x et de réalisations de bruit** [96] **et de plus nous n'avons aucune garantie que**

96. NDJE. Dans la version exposée par S. Mallat, le réseau ne connait pas le niveau de bruit dans l'image

l’optimisation du réseau aboutisse au débruiteur optimal.

Et bien ça marche [97]! La procédure est donc: 1) obtenir un réseau débruiteur d’images à partir de données de visages ou chambres-à-coucher, etc, 2) puis à partir d’une réalisation de bruit faire l’évolution de reverse-diffusion (Th. 9). On obtient un échantillon d’une "probabilité" d’images de visages, chambres-à-coucher, etc. même si on ne sait pas exactement ce que cela veut dire.

Cependant, il y a un mystère, car on essaye d’estimer le **score qui est une fonction en très grande dimension**, et rappelons-le avait des singularités que l’on a régularisé par convolution d’une gaussienne. Pour que cela marche, il faut **battre la malédiction de la dimension** qui n’est possible que si cette fonction a des **régularités**. Le constat est que cela semble être le cas. Maintenant, ce que l’on peut tester c’est de savoir si les échantillons générés sont des mélanges astucieux d’images de la base de données, ou bien, ce sont des échantillons indépendants de la base de données? Dit autrement, y’a-t’il **mémorisation ou généralisation?** Dans les séances prochaines, nous verrons ce qu’il en est, et nous nous pencherons sur l’architecture de l’U-Net pour comprendre pourquoi il peut constituer un bon débruiteur.

7. Séance du 26 Fév.

A partir de cette séance, nous allons nous pencher sur la notion de **débruitage optimal**, et sur les mathématiques qui ont été développées dans ce domaine. Revenons tout d’abord sur l’algorithme de score-diffusion pour situer où intervient la notion de débruitage.

---

x_σ. Dans l’étude de Y. Song le réseau est conditionné par σ_t, car il y a un schéma d’évolution du bruit en fonction du temps. Vous pouvez voir cette dépendance dans le notebook `JAX_blob_diffusion.ipynb` du cours de cette année.

97. NDJE. note de bas de page 93

7.1 Transport par Score-diffusion

7.1.1 Diffusion et Inversion

Pour rappel, nous avons d'abord défini [98] le transport $\boldsymbol{T}^{-1}$ qui pour un échantillon $x \in \Omega_x$ des données l'envoie sur un échantillon $z \in \Omega_z$ de bruit (Fig. 23). Ce transport s'effectue à l'aide de l'équation d'Ornstein-Uhlenbeck (Th. 8) qui pour mémoire s'écrit (paramètres normalisés Eq.128):

$$dx_t = -x_t \, dt + \sqrt{2} \, dW_t \qquad x(t=0) = x_0 \tag{148}$$

qui signifie que l'on ajoute progressivement du bruit au signal. La solution à tout temps t s'exprime selon

$$x_t = e^{-t} x_0 + \left(1 - e^{-2t}\right)^{1/2} z, \qquad z \overset{iid}{\sim} \mathcal{N}(0, Id) \tag{149}$$

où la composante initial x_0 est atténuée progressivement au profit d'une composante de pur bruit (partie "Diffusion" de la figure 10).

La question à laquelle nous avons répondue par la suite est: peut-on inverser $\boldsymbol{T}^{-1}$ afin d'obtenir le transport $\boldsymbol{T}$ qui permet la génération de nouveaux échantillons issus de la *pdf* des données? La réponse est oui grâce au théorème d'Anderson (Th. 9). Si l'on note T le temps au bout duquel le processus de diffusion est considéré comme représentatif de la solution à temps infini, et si l'on considère l'évolution de x_{T-t} pour le processus inverse [99]

$$dx_{T-t} = \left[x_{T-t} + \nabla_{x_{T-t}} \log p_{T-t}(x_{T-t})\right] \; dt + \sqrt{2} \, dW_t \tag{150}$$

Pour rappel, le score $\nabla_{x_{T-t}} \log p_{T-t}(x_{T-t})$ apparait quand on essaye d'inverser la diffusion dans l'équation de Fokker-Planck, afin de changer le signe du laplacien (Eq. 133). Une fois le score connu, la génération de nouveaux échantillons x est possible (partie "Inversion" de la figure 10).

98. NDJE. je prends la liberté d'écrire ici en gras les opérateurs de transports sachant que va apparaitre un temps T par la suite. D'ailleurs cette confusion de notation est déjà présente dans les sections précédentes.

99. NDJE. dans la séance précédente nous avion pris les notations y_t et q_t pour x_{T-t} et p_{T-t}. De plus $\alpha = 1$ dans le cas considéré ici.

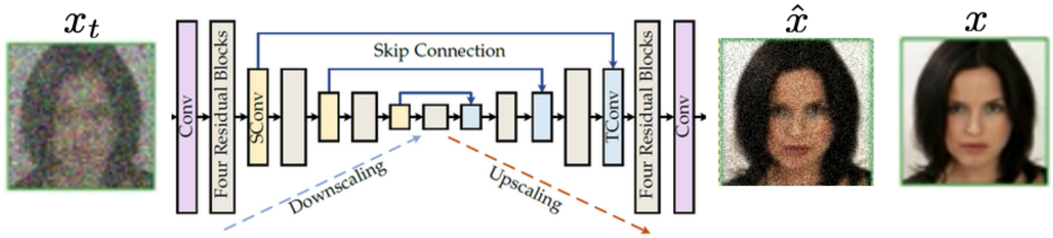


FIGURE 33 – Réseau U-Net utilisé comme débruiteur.

7.1.2 Obtenir le score: débruitage

Le point central est donc d'obtenir le **score**, et le fait qu'on puisse le faire à l'aide de réseaux profonds a constitué un tournant majeur. Cela ce fait par une procédure de débruitage que nous avons abordée à la section 6.2.3. Elle se sert du théorème 10 que nous récrivons d'une manière renormalisée [100]. Si un signal x est bruité par z (indép. de x) selon $x_\sigma = x + z$ avec $z \overset{iid}{\sim} \mathcal{N}(0, \sigma^2\, Id)$, son estimateur optimal au sens de l'erreur quadratique, noté $\hat{x}(x_\sigma)$, est relié au score de la façon suivante:

$$\min\left\{\mathbb{E}_{x,z}\left[\|\hat{x}(x_{\sigma_t}) - x\|^2\right]\right\} \Rightarrow \hat{x}(x_\sigma) = x_\sigma + \sigma^2\, \nabla_x \log p_\sigma(x_\sigma) \tag{151}$$

Donc, il nous faut par exemple un réseau de neurones comme celui [101] de la figure 33. Dans un premier schéma, on fournit au réseau des données de type (x_σ, σ), c'est-à-dire des données bruitées et le niveau de bruit qui les affecte individuellement, et la sortie est $f_\theta(x_\sigma)$. La fonction de coût [102] n'est autre que la MSE

$$\ell(\theta) = \mathbb{E}[\|x_\sigma - f_\theta(x_\sigma)\|^2] \tag{152}$$

La minimisation se fait à l'aide d'une descente de gradient classique (SGD, Adam...). Si, l'optimisation obtient le meilleur θ possible, noté θ^*, alors on est en droit d'identifier

100. $x_{\sigma_t} \leftarrow x_t e^t$, $x \leftarrow x_0$, $\sigma \leftarrow \sqrt{e^{2t} - 1}$.
101. NDJE. Il s'agit de la figure 11 reproduite ici par commodité.
102. NDJE. On peut également faire apprendre au réseau "le bruit" et non "le signal", c'est-à-dire que $f_\theta(x_\sigma)$ doit retrouver le z, il s'agit d'une technique de résidu.

$f_{\theta^*}(x_\sigma)$ à $\hat{x}(x_\sigma)$ et donc

$$\nabla_x \log p_\sigma(x_\sigma) = \frac{f_{\theta^*}(x_\sigma) - x_\sigma}{\sigma^2} \tag{153}$$

7.1.3 Le cas d'un processus gaussien bruité

Prenons le cas où x est gaussien de covariance $\boldsymbol{C}$, c'est-à-dire

$$p(x) = Z^{-1} \exp\left\{-\frac{1}{2} x^T \boldsymbol{C}^{-1} x\right\} \tag{154}$$

alors x_σ est aussi gaussien de covariance $\boldsymbol{C}_\sigma = \boldsymbol{C} + \sigma^2\, Id$ (rappel: x et z sont indépendants). Donc

$$p(x_\sigma) = Z_\sigma^{-1} \exp\left\{-\frac{1}{2} x_\sigma^T \boldsymbol{C}_\sigma^{-1} x_\sigma\right\} \tag{155}$$

Dans ce cas, le score associé à $p(x_\sigma)$ est donné par

$$\nabla_x \log p_\sigma(x_\sigma) = -\boldsymbol{C}_\sigma^{-1}\, x_\sigma \tag{156}$$

Et *in fine* **l'estimateur optimal d'un processus gaussien contaminé par un bruit gaussien** est donné par l'expression suivante

$$\hat{x}(x_\sigma) = (Id - \sigma^2 \boldsymbol{C}_\sigma^{-1})\, x_\sigma \qquad \boldsymbol{C}_\sigma = \boldsymbol{C} + \sigma^2\, Id \tag{157}$$

Nous allons retrouvé cette formule par une autre approche.

7.1.4 Erreur d'estimation du score

Maintenant, si l'optimisation n'est pas parfaite, et que donc l'estimation du score ne l'est pas non plus, la question qui se pose est: est-ce que cette **erreur sur le score** va induire une grosse erreur sur la distribution de probabilité obtenue par l'algorithme de diffusion inverse ou pas? Si l'on note $s_\theta(x_\sigma)$ le score obtenu à partir du réseau débruiteur, c'est-à-dire

$$s_\theta(x_\sigma) = \frac{f_\theta(x_\sigma) - x_\sigma}{\sigma^2}, \tag{158}$$

ce que l'on voudrait contrôler, c'est la divergence $D_{KL}(p\|p_\theta)$ entre la distribution exacte des données p, et celle obtenue par diffusion inverse notée ici p_θ. On sait que si $s_\theta(x_\sigma)$

est exactement le bon score, alors l'opérateur $\boldsymbol{T}^{-1}$ est bien inversé et la divergence est nulle. Dans ce cadre, il y a un théorème de Y. Song et al.[103] qui stipule que moyennant certaines hypothèses

$$D_{KL}(p\|p_\theta) \leq \int_0^\infty \mathbb{E}_{x_\sigma \sim p_\sigma} \left[\|s_\theta(x_\sigma) - \nabla_{x_\sigma} \log p_\sigma(x_\sigma)\|_2^2 \right] \sigma d\sigma \tag{159}$$

Ce que cela nous dit, c'est que si on a une petite erreur sur le score, et qu'elle s'intégre, alors **on peut apprécier l'erreur sur la distribution obtenue par diffusion inverse**. D'autres bornes sont disponibles avec la distance de Wasserstein (W_2). C'est une différence par rapport à ce que donne un GAN, pour lequel on a aucune façon de contrôler les erreurs sur les *pdf*s.

Ceci étant dit, le problème reste de pouvoir estimer le score avec une erreur qui n'est pas trop grande. En fait, on peut se convaincre qu'il y deux types d'erreur qui se dégagent et qui s'apparentent aux **erreurs traditionnelles d'un estimateur**[104].

1. **Variance/généralisation**: Le premier type d'erreur concerne un **problème de variance**, ou de **généralisation** versus **mémorisation**. La question qui se pose alors est: quelle est la dépendance de θ^* (paramètres du réseau) par rapport à la base d'entrainement? Si l'on change le lot d'entrainement (tout en préservant la *pdf* sous-jacente) a-t-on le même θ^* et donc le même score s_θ, ou pas? Si l'on s'attend a une différence numérique, se pose la question de la variance de l'estimateur s_{θ^*}. Il est assez naturel de considérer que si la variance est trop importante, c'est donc que l'estimateur dépend très étroitement de la base d'entrainement. On sent bien intuitivement alors que la génération d'images sera subordonnée par une *mémorisation* de la base d'entrainement. Au contraire, si l'estimateur a une immunité vis-à-vis de la base d'entraînement, on est conforté dans l'idée qu'il y a une *généralisation*, et les échantillons obtenus sont bien de nouveaux représentants de p_{data}.
2. **Biais/erreur de modèle**: Le second type d'erreur se place dans le cadre d'une variance nulle (invariant par changement de lot d'entraînement), et la question est alors de savoir si l'estimateur s_{θ^*} correspond bien à la valeur du score. C'est-à-dire

103. NDJE. Yang Song et al. (2021), *Maximum Likelihood Training of Score-Based Diffusion Models, arxiv:2101.09258. Il faut adapter au cas* $dx_t = g(t)dw_t$ *alors la solution s'exprime* $x_t = x_0 + \sigma_t z$ *avec* $z \sim \mathcal{N}(0,1)$ *où* $\sigma_t^2 = \int_0^t g(u)^2 du$, *d'où le facteur* $1/2g(t)^2 dt$ *du théorème de Y. Song se transforme en* $\sigma_t d\sigma_t$ *et l'on a fait disparaitre l'indice t qui n'est pas pertinent ici.*
104. NDJE. Voir Cours 2018 Sec. 2.3

que le modèle peut certes être optimisé et donner le même θ^* pour n'importe quel lot d'entraînement, et pourtant donner une valeur erronée du score. Ce cas de figure est sous-tendu par *primo* l'**architecture du réseau** qui doit être suffisamment flexible pour pouvoir modéliser des transports complexes, et *secundo* l'**optimisation** qui doit aboutir au bon θ^* (la fonction de coût doit bien être au minimum).

Nous allons aborder ces questions avec un réseau de neurones particulier: le **U-Net** [105] que nous représentons plus en détails sur la figure 34. Originellement, il était destiné pour effectuer des tâches de segmentation dans le monde biomédical (ex. reconnaissance de structures neuronales dans des images obtenues par un microscope électronique). Par la suite, il a été utilisé dans bien d'autres fonctions grâce à ses caractéristiques. Premièrement, les images en entrée et en sortie sont de même dimension ($H \times W \times C$) bien adapté pour un débruiteur, deuxièmement l'architecture est composée en deux phases. La première phase s'apparente à un *encoder* qui comprime l'information en appliquant une cascade d'opérateurs (convolution 3×3, ReLu, BatchNorm) où l'on intercale un sous-enchantillonnage. La seconde phase opère comme un *decoder* en décomprimant l'information par une cascade d'operateurs "inverses" (sur-échantillonnage, déconvolution, ReLu,...) et intègre à chaque étape les sorties correspondantes de l'*encoder* (flèches horizontales dans la figure), ce qui permet une **modélisation des interactions multi-échelles**.

Ceci dit, pourquoi l'usage des différents opérateurs (convolution, non-linéarité, normalisation) est important, et pourquoi la notion d'échelle (sous/sur-échantillonnage) est-elle tout aussi importante? Ces différentes questions ont déjà été abordées dans les cours précédents [106]. La question sous-jacente est **pourquoi un U-Net peut apprendre une fonction aussi complexe que le score en grande dimension?** C'est la grande question, nous dit S. Mallat, que l'on comprend encore assez mal d'un point de vue mathématiques. Or, dans le cas qui nous occupe ici, celui du **débruitage et l'estimateur optimal en erreur quadratique**, il a été considéré depuis les années 1945 et il y a un bon **corpus mathématiques sur le sujet**. Ainsi, on peut faire un aller-retour entre les maths d'un coté et les solutions trouvées par les réseaux de neurones. En quelque sorte, ce cadre est un **bon laboratoire pour essayer de comprendre** finement pourquoi un U-Net peut marcher.

105. NDJE. Architecture introduite dans Olaf Ronneberger et al. (2015), *U-Net: Convolutional Networks for Biomedical Image Segmentation*, *arxiv:1505.04597*
106. NDJE ex. Cours 2020 Sec 9.2

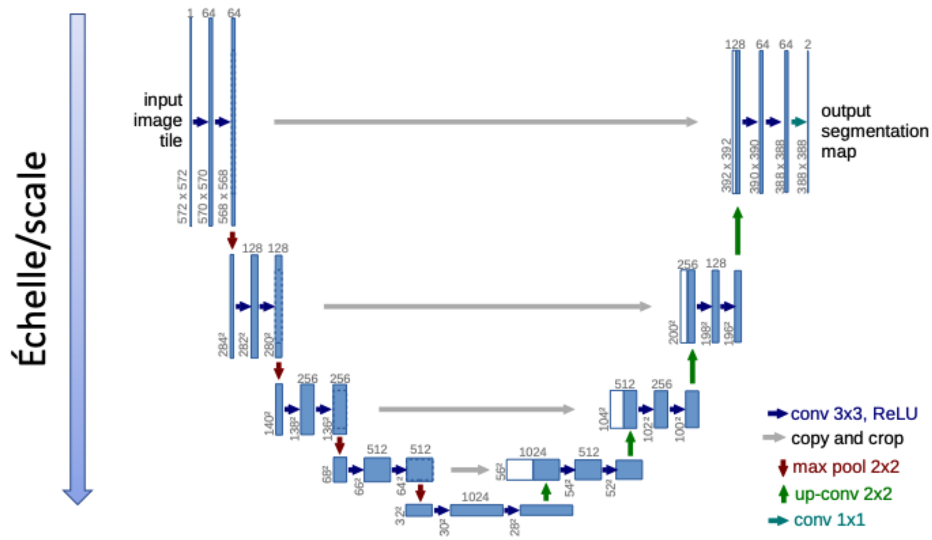


FIGURE 34 – Architecture multi-échelles d'un U-Net.

7.1.5 Retour d'expériences numériques

NDJE. Avant de plonger dans le domaine du débruitage, S. Mallat revient sur le problème de variance de l'estimateur du score, c'est-à-dire du problème généralisation vs mémorisation. C'est un point qu'il a déjà soulevé à la section 2.8.2, notamment à la faveur du commentaire de la figure 13 qu'il repasse à l'écran. Le message que l'on retient de l'expérimentation numérique entreprise par Kadkhodaie et al. est que si le nombre d'images des lots d'entrainement donnés aux deux réseaux U-Nets ne sont pas suffisamment grands, alors les trajectoires de diffusions inverses partant du même bruit blanc ne donnent pas la même image générée; et plus les lots sont petits plus l'image générée par chaque U-Net est manifestement une image de son lot d'entraînement. **Au-delà de ce seuil** [107] $O(10^5)$ pour des images 80×80, **les deux trajectoires de diffusions inverses (toujours partant du même bruit blanc) donne la même image exactement indépendamment des lots d'entrainement, et cette image ne fait pas partie de l'un ou l'autre lot d'entrainement des deux U-N**. Il y a bien un seuil qui caractérise **une transition d'un régime de mémorisation vers un régime de généralisation**.

107. NDJE. J'ai mis à disposition sur le site github le notebook `JAX_FLAX_NNX_UniversalDen_MNIST_jaxjit.ipynb` que vous pouvez modifier/adapter pour réaliser ce type d'expérience numérique.

C'est un résultat important à deux niveaux: la premier concerne le fait que l'**on peut obtenir un régime de généralisation** et non de la simple mémorisation qui fut un argument souvent avancé face aux premiers résultats des modèles génératifs de score-diffusion, mais le second niveau nous dit que pour atteindre ce régime, **il faut beaucoup de données**. Par exemple, si on interroge un LLM par une question trop ciblée où il y a peu de textes sur Internet, on a de forte chance qu'il se trouve dans le cas de mémorisation, à l'inverse si la question porte sur un large corpus, la réponse aura été formulée dans le cadre d'une généralisation.

S. Mallat nous montre d'autres exemples (Fig. 35) générés par les 2 U-Nets pour lesquelles les lots d'entrainement sont constitués de $N = 10^5$ images (régime de généralisation), et l'on change la graine de génération du bruit blanc tout gardant la même pour les deux réseaux. Si la quasi-totalité des images générées par les deux réseaux sont identiques, remarquons la différence pour la dernière ("les lunettes"). C'est un phénomène intéressant dit de **bifurcation**, où une petite erreur sur le score induit un effet non négligeable sur l'image restituée, d'ailleurs il ne s'agit pas d'une image avec des moitiés de lunettes mais une sorte de *tout-ou-rien*. On arrive à comprendre ce genre de phénomène avec des mélange de gaussiennes.

FIGURE 35 – Autres exemples générés avec $N = 10^5$. Les échantillons générés par chaque U-Net sont présentés dans des lignes séparées, chaque colonne indiquant la même initialisation pour les réseaux. Extrait de l'article en note bas de page 29.

Bien entendu, on peut changer le type de bases de données et rejouer l'expérience numérique, les conclusions sont les mêmes. **Le seuil dépend de la taille des images**. Le nombre de paramètres des réseaux utilisés dans les expérimentations était de 7.6 millions et avec une base d'images 80×80, la généralisation s'obtient quand on a de l'ordre de 10^2 données (pixels) par paramètre. Qu'en est-il pour des images plus grandes/petites? La loi d'échelle est semble-t-il inchangée [108]. Mais, ceci étant dit, on ne peut attacher un

108. NDJE. Par exemple avec MNIST qui est constitué de $60,000$ images de 784 pixels, on peut s'en

degré très élevé à ce genre de considérations, car il y a des phénomènes non-linéaires, de redondances, etc dans l'architecture du réseau. Finalement, on a du mal à comprendre la relation entre la taille de l'échantillon et la taille du réseau au seuil de la généralisation. Cependant, il semble somme toute que l'on a affaire à un phénomène assez classique, à savoir que si l'on veut qu'un estimateur paramétré converge bien, **il faut beaucoup plus de données que de paramètres**. Ainsi, avec de très grands modèles, on obtient certes de très belles images, mais attention à la mémorisation! C'est une chose importante à garder à l'esprit, nous dit S. Mallat, surtout quand on s'attache à simuler par exemple un écoulement fluide, pour étudier la réponse d'une aile d'avion, ou quand on essaye d'effectuer une prédiction médicale. Dans ces cas de figures, il faut plus s'attacher à savoir si l'on a suffisamment de données, afin de s'assurer de la diversité des échantillons générés.

Une autre façon de tester la généralisation, c'est de **calculer l'erreur quadratique moyenne** entre les images originales et celles débruitées par le réseaux. Classiquement, on utilise le *PSNR* (Peak Signal to Noise Ratio) définit selon la formule

$$PSNR = 10\,\log_{10} \frac{d^2}{\mathbb{E}\|x-\hat{x}\|^2} \tag{160}$$

avec $d = 255$ si le codage des valeurs des pixels se fait sur des entiers de 8-bits. Plus l'erreur est petite, plus le PSNR est grand. On peut regarder l'évolution de ce PSNR de sortie, selon l'origine du lots des images *training* vs *test*, selon la taille du lot d'entrainement, en fonction du bruit injecté, le PSNR d'entré proportionnel à $\log(1/\sigma)$. Les résultats avec des images de chambres à coucher sont présentés sur la figure 36.

doute atteindre un régime de généralisation avec un modèle comportant de l'ordre de $5\,10^5$ paramètres. Celui de `JAX_FLAX_NNX_UniversalDen_MNIST_jaxjit.ipynb` en a $8\,10^5$, il est peut être un peu trop gros, à vous d'en juger.

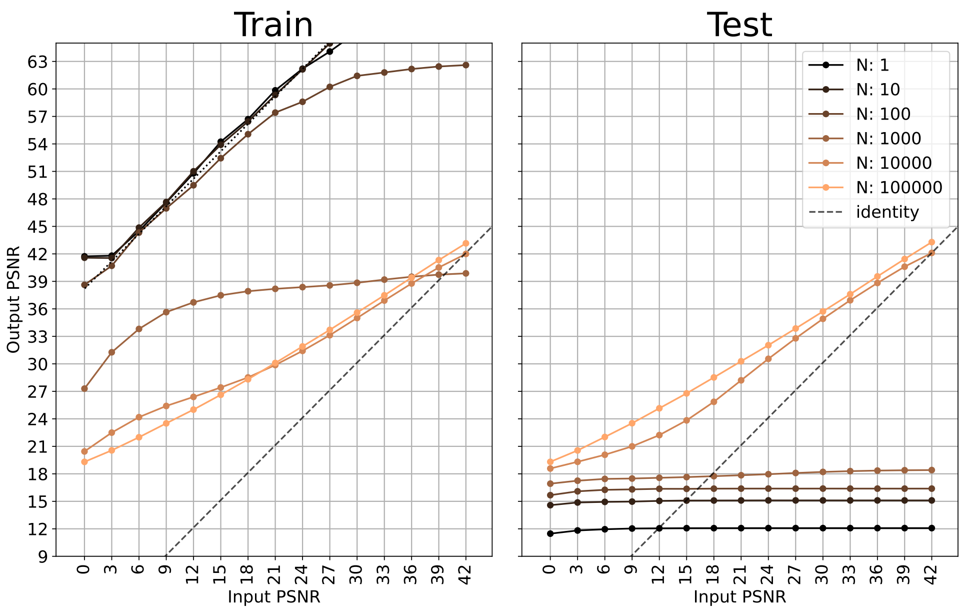


FIGURE 36 – Chaque courbe montre l'erreur de débruitage (PSNR de sortie) en fonction du niveau de bruit (PSNR d'entrée), pour un lot d'entrainement de chambres à coucher de taille N. Extrait de l'article en note de bas age 29.

Quand N est petit (les courbes foncées) les courbes en *Training* et *Test* sont totalement différentes et l'on constate un **overfitting** manifeste. Dans le cas où $N = 10^5$, les courbes en orange clair de *Training* et *Test* sont quasi-identiques, il y a bien une **généralisation**.

Une fois que l'on assure que le réseau est dans le régime de la *généralisation*, qu'en est-il du *biais*? En fait quand on regarde attentivement les images de chambres à coucher produites avec des U-Nets entrainés avec des lots de tailles $N = 10^5$, on se rend compte qu'il y a des défauts, même si la résolution 80×80 n'est pas très grande (Fig. 37).

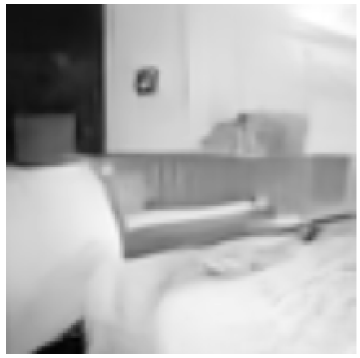


FIGURE 37 – Focus sur une image de chambre à coucher généré par un U-Net entrainés avec des lots de tailles $N = 10^5$. Extrait de l'article en note bas de page 29.

On peut se convaincre qu'une image de chambre à coucher est assez complexe, donc que le réseau ait du mal à en générer une belle est tout à fait possible. Il faut s'en doute plus de données et des modèles plus gros.

Enfin, la question qui reste en suspend est: même si on a des tests de la petitesse de la variance, qu'en est-il de la couverture de tous les modes de p_{data}? Pour mémoire, nous avions évoqué des phénomènes de *mode collapse* pour les GANs (Sec. 2.7). Ce type de **phénomène de manque de diversité n'est pas contrôlé par la variance de l'estimateur**. On aimerait que

$$\mathbb{E}_{x \in \mathcal{D}}[s_\theta(x)] = \nabla_x \log p_\sigma(x) \tag{161}$$

c'est-à-dire que le modèle restitue bien le vrai score, c'est-à-dire **il nous faut une absence de biais**. Et l'on constate que c'est beaucoup plus **difficile à vérifier**, pour la simple et bonne raison qu'**il nous faudrait connaitre le vrai score**. Or, s'il y a des cas simples (ex. processus gaussien) où l'on connait l'expression du score, on serait bien en peine d'en dire autant concernant les images de chambres à coucher.

Il nous faut donc ouvrir le chapitre mathématique du calcul de l'estimateur (débruiteur), pour comprendre dans quels cas on obtient bien l'estimateur optimal, ce qui garantirait que le score est parfaitement estimé, et qui garantirait *in fine* la bonne estimation de p_{data}.

7.2 Débruitage optimal: panorama mathématique

C'est un nouveau chapitre mathématique qui s'ouvre, cependant nous verrons qu'il s'adosse à celui des **représentations parcimonieuses** du signal étudiées dans le cours de 2021. Les questions auxquelles on aimerait répondre sont par exemple: est-ce qu'un réseau de neurones est capable d'effectuer un débruitage optimal? A quelles conditions et comment? Dés que l'on sera en mesure de répondre à ces questions, cela signifiera que la théorie des réseaux de neurones est sous contrôle. Car, on saura quelles sont les classes de fonctions peuvent être approximées par un réseau. A l'heure actuelle, on n'a pas la réponse finale. L'enjeu du cours est d'aller dans cette direction, et de donner les éléments mathématiques que l'on maitrise actuellement pour progresser sur ce chemin.

Dans ce chapitre mathématique, nous allons aborder:

— en premier lieu **l'approximation linéaire optimale** [109]: si l'on impose que l'estimateur optimal est un opérateur linéaire, comment le calculer? Il s'agit du **filtrage de Wiener**. Nous verrons alors le lien avec les **bases de PCA**, et dans le cas de **phénomènes stationnaires** va apparaître la notion de **convolution**, présente dans les réseaux et associée aux **bases de Fourier** (notion de puissance spectrale). Finalement, il s'agit du chapitre de l'**Analyse Harmonique**.

— en second lieu, nous irons dans le domaine de **l'approximation non-linéaire** pour explorer la question: peut-on améliorer l'estimateur, et à quelles conditions? Dans ce contexte apparaît la notion de **parcimonie** dans une base. C'est-à-dire que lors d'une décomposition d'un signal dans cette base, la plupart des coefficients y sont nuls. De quelle(s) base(s) s'agit-il? Il nous faudra remplacer Fourier. C'est dans ce domaine que l'on trouve les **bases d'Ondelettes**, et la notion d'**échelle dyadique** (sous et sur échantillonnage présents dans l'architecture U-Net). Nous verrons aussi que la non-linéarité **ReLU** est un opérateur très naturel pour effectué un débruitage dans ce cadre. La question en suspend, est celle de **l'optimalité?** La question sous-jacente est de savoir si le problème auquel on est confronté rentre ou pas dans la classe de fonctions accessibles. On connait par exemple les *espaces de Besov* où l'optimalité est atteignable. Il s'agit d'appréhender la **régularité du processus sous-jacent**. En particulier, dès qu'il y a de la géométrie complexe, ex. dans les images de visages, de chambre à coucher, etc, il y a un hic.

109. NDJE. Un point de terminologie *linéaire vs non-linéaire* voir Cours 2021 Sec. 3.2.

— il nous faut alors ouvrir le chapitre des la **Géométrie**, en particulier celle des images avec des contours ayant des régularités. Dans cas, on ne peut plus trouver une base qui peut donner une représentation parcimonieuse optimale. Il nous faut aller vers les **familles de bases**. C'est en gros le point où la recherche en était environ dans les années 2005, nous dit S. Mallat. Et il nous décrit que les **modèles devenaient très compliqués**, et la communauté s'est un peu découragée, car malgré la complexité des modèles, les **résultats numériques n'étaient pas au rendez-vous**. Sur des images simples on pouvait s'en tirer, mais pas sur des images réelles.

— **Que se passe-t-il alors dans les réseaux de neurones**? ils ont la capacité à calculer des **bases harmoniques** (qui oscillent) avec de la **géométrie**. Dans les cas où l'on contrôle la régularité des contours, les bases trouvées par les réseaux sont bien les bonnes bases attendues avec les propriétés d'optimalité. Donc, **on a des cas complexes (plus que le cas gaussien) où l'on démontre l'optimalité des bases trouvées par les réseaux**.

Entamons alors ce programme par la base: le filtrage de Wiener.

7.3 Filtrage de Wiener, PCA

Il s'agit d'un résultat de base de traitement du signal établi par Norbert Wiener (1894-1964) qui permet de bien cerner le problème du débruitage avec des opérateurs linéaires et qui justifie l'usage des convolutions. ***Notation****: tout ce qui est aléatoire sera en lettres capitales, et tout ce qui est déterministe sera en lettre minuscules.* Soit le problème classique d'un bruit additif suivant

$$Y = X + Z \tag{162}$$

avec Z un bruit blanc gaussien (bbg) de moyenne nulle et de covariance $\sigma^2\, Id$ et **indépendant du signal** X. On se place en dimension finie.

On veut comme précédemment calculer un estimateur $\hat{X}$ obtenu par l'application d'un opérateur H (nb. déterministe) sur le signal bruité Y. Ainsi,

$$\hat{X} = H[Y] \tag{163}$$

Pour obtenir l'expression de H, on utilise la minimisation de l'erreur quadratique moyenne

$$\min_H \mathbb{E}_{X,Z}[\|X - H[Y]\|^2] \tag{164}$$

avec comme contrainte que H appartienne à une famille paramétrée d'opérateurs

$$H \in \{H_\theta\}_\theta \tag{165}$$

Nous imposons alors que H **soit linéaire**. Si la représentation de H est une matrice en dimension d, θ représente les d^2 éléments de cette matrice.

Donc quel est le meilleur H et comment se calcule-t-il? *S. Mallat évoque les différents champs auxquels N. Wiener a jeté les bases mathématiques*[110] *notamment en Analyse Harmonique, processus stochastiques, en traitement du signal, en contrôle, en cybernétique, etc.*

Voici le théorème de base:

Théorème 11
Soit d mesures $Y(k)_k$, l'estimateur linéaire du signal sous-jacent X est donné part

$$\hat{X} = H[Y] = \sum_{k \leq d} h[k]Y(k) \tag{166}$$

qui minimise l'erreur quadratique moyenne, ssi il réalise la décorrelation par rapport aux données, c'est-à-dire

$$\mathbb{E}_{X,Z}((X - \hat{X})Y(k)) = 0 \quad \forall k \tag{167}$$

Démonstration 11. Soit l'erreur quadratique

$$\varepsilon = \|X - \sum_{k \leq d} h[k]Y(k)\|^2 \tag{168}$$

Il s'agit d'une forme quadratique en h, et son minimum peut se calculer en annulant la

110. NDJE. Voir Cours 2019 Sec. 4.1

dérivée:

$$\frac{\partial \varepsilon}{\partial h[k]} = 0 \Leftrightarrow \mathbb{E}_{X,Z}((X - \hat{X})Y(k)) = 0 \tag{169}$$

Donc, la condition Eq. 167 est nécessaire. Elle est suffisante car la forme quadratique est convexe. Incidemment, le théorème nous dit que $\mathbb{E}(X\,Y(k)) = \mathbb{E}(\hat{X}\,Y(k))$.

■

On peut maintenant généraliser, nous écrivons à présent un signal connu à un "instant" n, $X(n)$, et son estimation se fait à l'aide de d données

$$\hat{X}(n) = \sum_k h[k,n]Y(k) \tag{170}$$

avec $H = (h[k,n])_{k,n}$ une matrice. On va la relier à l'auto-corrélation du processus

$$C = (\mathbb{E}(X[k]X[n]))_{k,n} \tag{171}$$

On considère X et Y comme des vecteurs aléatoires de d composantes.

Théorème 12 ***(estimateur linéaire optimal)***
L'opérateur optimal H s'écrit

$$H = (C + \sigma^2 Id)^{-1}C \tag{172}$$

Démonstration 12. On veut minimiser

$$\mathbb{E}(|X - \hat{X}|^2) = \sum_n (X(n) - \hat{X}(n))^2 \tag{173}$$

c'est-à-dire trouver le minimum de chaque élément de la somme. D'après le théorème précédent, nous avons $\mathbb{E}(X(n)Y(k)) = \mathbb{E}(\hat{X}(n)Y(k))$ $(\forall k)$ que l'on peut développer

$$\mathbb{E}(X(n)Y(k)) = \sum_\ell h[\ell,n]\,\mathbb{E}(Y(k)Y(\ell)) \tag{174}$$

Or, $Y(k) = X(k) + Z(k)$ et selon les hypothèses de bruit considérée, nous avons alors $\forall k, n$

$$\mathbb{E}(X(n)Y(k)) = \mathbb{E}(X(n)X(k)) = \sum_{\ell} [\mathbb{E}(X(k)X(\ell)) + \sigma^2 \delta_{k,\ell}]\, h[\ell, n] \qquad (175)$$

Ainsi, sous forme matricielle nous pouvons écrire

$$C = (C + \sigma^2 Id)H \qquad (176)$$

Ce qui donne le résultat escompté car la matrice entre parenthèse est toujours inversible.

Un résultat remarquable: on peut réécrire l'expression de H selon

$$H = Id - \sigma^2 (C + \sigma^2\, Id)^{-1} \qquad (177)$$

qui **correspond exactement à la formule du débruiteur optimal obtenu avec le score dans le cas gaussien** (Eq. 157). Donc, **l'estimateur linéaire est aussi l'estimateur optimal (linéaire et non-linéaire) pour des processus gaussiens**. ■

Comment obtenir l'expression la plus adéquat de H? D'une certaine façon, il faut trouver une base dans laquelle C est la plus simple possible, à savoir trouver **la base qui diagonalise la matrice d'auto-corrélation**, il s'agit de la **base de PCA** [111]. Soit $(e_k)_k$ cette base, les valeurs propres notées $(\lambda_k)_k$ sont telles que [112]

$$\lambda_k = \langle e_k, C\, e_k \rangle = \mathbb{E}[|\langle X, e_k \rangle|^2] \qquad (178)$$

Maintenant, $C + \sigma^2 Id$ est également diagonale dans la base PAC, donc il en est de même de H, et nous avons que ses valeurs propres sont telles que

$$(v.p)(H)_k = \frac{\lambda_k}{\lambda_k + \sigma^2} \approx \begin{cases} 1 & \text{si } \lambda_k \gg \sigma^2 \\ 0 & \text{si } \lambda_k \ll \sigma^2 \end{cases} \qquad (179)$$

111. nb. Analyse en Composantes Principales
112. NDJE. Voir Cours 2021 Sec. 4.3.

Que vaut $H[X]$? Il vient

$$H[X] = \sum_k \frac{\lambda_k}{\lambda_k + \sigma^2} \langle X, e_k \rangle \, e_k \approx \sum_{k\, tq.\, \lambda_k \gg \sigma^2} \langle X, e_k \rangle \, e_k \tag{180}$$

On voit alors que H filtre X, pour ne grader que les composantes d'énergie non négligeable sur les vecteurs de la base PCA. On peut alors ordonner les valeurs λ_k ou $|\langle X, e_k \rangle|^2$ en fonction de k comme sur la figure 38, notons que k indique aussi via e_k une direction. Le bruit blanc a la même énergie dans toutes les directions (trait pointillé horizontal), il constitue un seuil: $\forall k < k_M$ le signal a plus d'énergie que le bruit, donc H garde ces composantes.

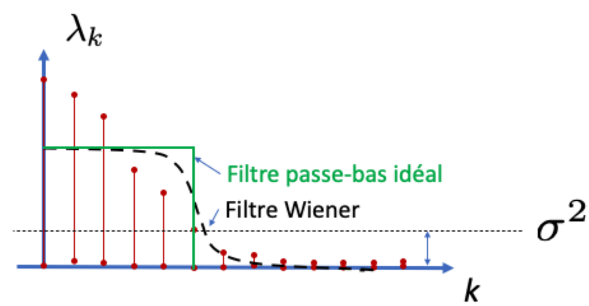


FIGURE 38 – Décroissance des différents valeurs propres λ_k dans la base PCA, effet de seuillage par le bruit. En pointillé noir le filtre de Wiener (Eq.180), et en trait plein vert le filtre passe-bas idéal.

C'est l'idée de base du filtrage linéaire optimal. Il reste à (re)connaître la base qui diagonalise la covariance du signal. Pour cela nous allons utiliser une hypothèse naturelle de **stationnarité**.

7.4 Stationnarité, Fourier

La notion de **stationnarité** nous dit que si $(\forall n)$ $X(n)$ est translaté par un facteur τ, $X_\tau(n) = X(n - \tau)$, alors X **et** X_τ **ont la même densité de probabilité**. Cette hypothèse est assez naturelle. Considérons l'exemple de photos prises par une caméra pour fixer les idées: si on ne recentre pas chaque photo sur un objet particulier (un vase), ce même objet se retrouve à plusieurs endroits des photos successives d'une même scène filmée aléatoirement. Or, il s'agit du même objet, sa probabilité d'apparaitre est identique dans chaque photo. Ainsi, toute image de l'objet a la même probabilité d'apparaitre que sa translatée par n'importe quelle valeur (**invariance par translation**).

Quelles sont les conséquences de cette invariance par translation? En particulier, nous avons que la moyenne de $X(n)$ ne dépend pas de n, et on peut regarder les termes de second ordre, etc:

$$\begin{aligned} &\forall n \qquad && \mathbb{E}[X(n)] = \mu \\ &\forall n,k \qquad && C_{n,k} = \mathbb{E}[X(n)X(k)] = \mathbb{E}[X(0)X(k-n)] = \mathbb{E}[X(n-k)X(0)] = C(|n-k|) \end{aligned} \tag{181}$$

La matrice d'auto-corrélation a une structure en bandes comme sur la figure 39.

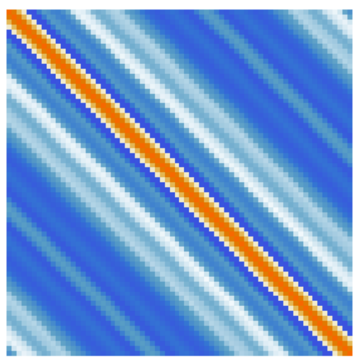


FIGURE 39 – Exemple de matrice d'auto-corrélation sous l'hypothèse d'invariance par translation: on y voit la structure en bandes d'une matrice Toeplitz symétrique.

Si maintenant on considère C comme un opérateur, alors

$$C\,x(n) = \sum_k C_{n,k}\,x(k) = \sum_k C(n-k)\,x(k) = (C * x)(n) \tag{182}$$

La matrice d'auto-corrélation agit comme un opérateur de convolution. Donc, qui dit "**invariance par translation**" dit aussi "**opérateur de convolution**".

On peut alors répondre à la question de la base qui diagonalise C, c'est-à-dire qui **diagonalise un opérateur de convolution**: il s'agit de la **base de Fourier** dont les vecteurs s'écrivent $e_\omega(n) = e^{i\omega n}$ (ici $n \in \mathbb{Z}$). En effet,

$$\begin{aligned} C\, e_\omega(n) = C e^{i\omega n} &= \sum_k C_{n-k} e_\omega(k) = \sum_k C(k) e_\omega(n-k) = \sum_k C(k) e^{i\omega(n-k)} \\ &= e^{i\omega n} \sum_k C(k) e^{-i\omega k} \\ &= \widehat{C}(\omega)\, e_\omega(n) \end{aligned} \tag{183}$$

Donc, la famille $\{e_\omega = e^{i\omega\,\cdot}\}_\omega$ constitue les vecteurs propres de C, dont les valeurs propres sont les valeurs prises en ω de **la transformée de Fourier de l'auto-corrélation** du signal

$$\widehat{C}(\omega) = \sum_k C(k) e^{-i\omega k} \tag{184}$$

que l'on nomme la **puissance spectrale**.

Dans le cas de signaux de taille finie, il faut opérer une procédure de périodisation [113] afin de pouvoir considérer une convolution circulaire. Les vecteurs de la base sont également périodiques et s'écrivent

$$e_k(n) = e^{i\omega_k n}; \quad 0 \leq k < d;\, 0 \leq n < d;\, \omega_k = 2\pi k/d \tag{185}$$

avec des fréquences quantifiées. Il s'agit alors de la base de **Fourier discrète**.

Donc, dès que l'on a affaire à un processus stationnaire, la base de Karhunen-Loève (PCA) est la base de Fourier. On peut donc avoir la vision en termes de puissance spectrale (Fig. 40) de la décroissance des λ_k de la figure 39. Le **filtre de Wiener** est alors analysé comme un **filtre basses fréquences** avec une coupure qui dépend du rapport entre l'énergie du signal et l'énergie du bruit.

113. NDJE. Voir par ex. Cours 2018 Sec. 5.2.2

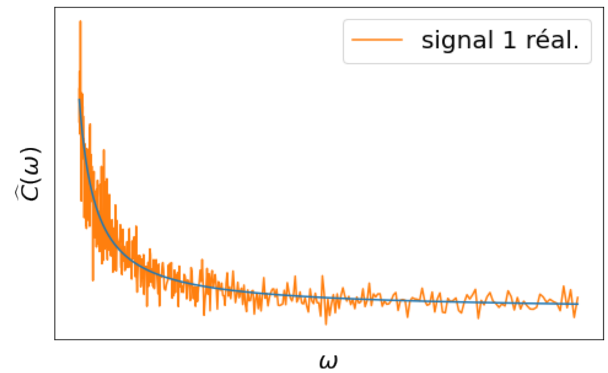


FIGURE 40 – Analyse de la figure 39 d'un point de vue spectral, avec la correspondance $\omega \leftrightarrow k$.

Un exemple de l'usage du filtrage de Wiener nous est présenté par S. Mallat au projecteur avec un processus gaussien. Un autre exemple [114] est présenté sur la figure 41.

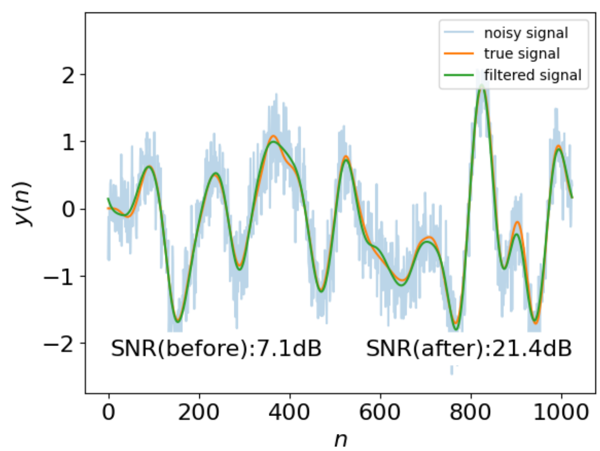


FIGURE 41 – Filtrage de Wiener appliqué à un processus gaussien corrompu par un bruit blanc.

Le résultat est tout à fait satisfaisant, en termes de SNR (*signal-to-noise ratio* [115]) on observe un gain de +14.3dB. Le filtre de Wiener, en tant que filtre passe-bas, effectue

114. NDJE. Sur le repository github vous pouvez le rejouer avec le notebook `WienerFilter_GP.ipynb`
115. $SNR = 10 \log_{10}(\|x\|^2 / \|\hat{x} - x\|^2)$

une moyenne locale. On ne peut faire mieux que cela soit en linéaire ou en non-linéaire en raison de l'optimalité du débruiteur pour un processus gaussien (Th. 10).

Ceci dit quand on considère des images de chambres à coucher, des trames sonores vocales ou instrumentales, il y a des **phénomènes transitoires** qui ne sont plus des processus gaussiens (Fig. 42) [116]. On constate un grain nettement moindre dans le débruitage (+4.5dB). Il reste du bruit, et l'on comprend qu'il y a une compétition qui s'opère entre réduire le bruit qui tend à supprimer les hautes fréquences, tout en voulant garder les transitions franches qui tend à ne pas supprimer trop les hautes fréquences. Cette optimisation dans le cas du filtrage de Wiener est effectuée par la minimisation de l'erreur quadratique.

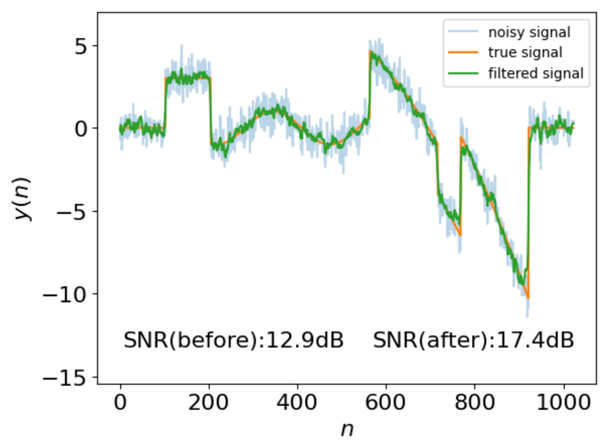


FIGURE 42 – Filtrage de Wiener appliqué à un processus à transitoires corrompu par un bruit blanc.

La question qui vient alors: peut-on améliorer le débruitage et comment? Imaginons que l'on ait la possibilité de détecter les discontinuités du signal, alors durant les phases "calmes" on pourrait appliquer un filtrage sévère des hautes fréquences pour réduire totalement le bruit. Mais ce faisant l'algorithme est dépendant sur signal. On passe alors *de-facto* dans le monde **non-linéaire**. Il nous faut penser un estimateur qui réalise une adaptation par rapport aux discontinuités du signal à traiter. Il ne peut s'agir d'un opérateur de convolution, car par nature il est associé à une invariance par translation. Ici,

116. NDJE. Sur le repository github vous trouverez le notebook `WienerFilter_transitoires.ipynb` pour rejouer l'exemple.

il nous faut s'adapter au signal afin de réaliser des opérations locales qui n'ont pas lieu partout. Cela sera le sujet des sections suivantes.

8. Séance du 5 Mars

Nous avons débuté à la dernière séance l'exploration du chapitre mathématique du débruitage afin d'expliquer les propriétés du réseau **U-Net** (Fig. 33) utilisé pour calculer le *score* en grande dimension. Nous avons parcouru la version "**linéaire**" du débruitage, et mis en évidence le **filtre de Wiener** optimal pour les opérateurs linéaire, et dans le cas *stationnaire* l'apparition du triptyque "**convolution, invariance par translation, base de Fourier**". Si le filtrage de Wiener est optimal pour les processus gaussien (on ne peut faire mieux) comme sur la figure 41, concernant les *phénomènes transitoires* (Fig. 42), il nous faut envisager un *débruitage adaptatif*, donc se tourner vers le **non-linéaire** où nous allons voir apparaître les notions de **parcimonie** et d'**échelle**. *NDJE. Comme pour la séance précédente, il est bon d'avoir un œil sur les cours précédents comme celui de l'année 2021.*

8.1 Estimation non-linéaire diagonale dans une base

Il va s'agir de trouver des bases orthonormales pour lesquelles les signaux ont des représentations où l'énergie est concentrée sur un petit nombre de coefficients (notion de parcimonie). Les questions que l'on a en tête avec le prisme de l'U-Net sont: pourquoi des convolutions, pourquoi les non-linéarités comme le ReLU, et en quoi l'aspect multi-échelles est important?

On se place dans le cas d'un bruit additif qui corrompt un signal x **ici déterministe fixé**[117] selon

$$Y = x + Z \tag{186}$$

avec Z un bruit blanc gaussien (*bbg*) de moyenne nulle et de covariance $\sigma^2\, Id$ et **indépendant du signal** x. On veut calculer un estimateur $\hat{x}$ obtenu par l'application d'un opérateur H sur le signal bruité Y. Ainsi,

$$\hat{x} = H[Y] \tag{187}$$

117. NDJE. notez la différence par rapport à la séance précédente ou X est une *v.a.*

Pour obtenir l'expression de H, on utilise naturellement la minimisation de l'erreur quadratique moyenne

$$\min_H \mathbb{E}_Z[\|x - H[Y]\|^2] \tag{188}$$

Cependant [118], à présent H est un opérateur qui n'est plus contraint d'être *linéaire* [119]. On va permettre l'opérateur de **s'adapter aux irrégularités du signal**, H est donc dépendant du signal lui-même. Au passage, il nous faut pouvoir détecter des singularités engendrées par les phénomènes transitoires. Or, si l'on imagine effectuer une dérivé du signal (Fig. 42), à cause du bruit, cette technique est vouée à l'échec. Néanmoins, on peut regarder les incréments du signal à différentes *échelles*. C'était la base d'algorithmes de plus en plus complexes des années 1970-80, nous dit S. Mallat. Tout cela s'est considérablement simplifié quand on a pris le point de vue de la **représentation parcimonieuse** du signal, et appliqué un **seuillage dur** (*hard threshold*) sur les petits coefficients. Dans le cours de 2021, l'application des représentations parcimonieuses était tournée vers la compression du signal, ici nous voyons l'autre grande application qu'est le débruitage, d'où le triptyque "**représentation, compression, débruitage**".

Nous allons nous placer toujours dans la cas de bases orthonormales $\{e_k\}_{k\leq d}$ [120] à découvrir par la suite. Si on cherche, à x fixé, quelle est l'expression de H **non-linéaire**, nous allons néanmoins le **contraindre à être diagonal**, c'est-à-dire que pour tout signal x

$$H[x] = \sum_k h[k;x]\langle x, e_k\rangle e_k \tag{189}$$

Il s'agit d'une représentation diagonale car les coefficients $\langle x, e_k\rangle$ ne sont affectés que par un facteur multiplicatif. Ainsi, les coefficients $h[k;x]$ dépende du signal x lui-même. Par la suite pour simplifier la notation on écrit $h[k]$ mais il faut avoir cette idée de dépendance en arrière fond. La minimisation de l'erreur quadratique fournit les expressions des coefficients $h[k]$. Le fait d'être dans une base orthonormale nous facilite la tâche, il

118. NDJE. S. Mallat fait un résumé du cours de la séance précédente que je ne retranscris pas ici.
119. NDJE. voir note de bas de page 109.
120. $\langle e_\ell, e_k\rangle = \delta_{\ell,k}$.

vient [121]

$$r = \mathbb{E}_Z[\|H[Y] - x\|^2] = \sum_k \mathbb{E}_Z\left[|(h[k]-1)\langle x, e_k\rangle + h[k]\langle Z, e_k\rangle|^2\right]$$
$$= \sum_k (h[k]-1)^2\langle x, e_k\rangle^2 + h[k]^2\sigma^2 \quad (190)$$

Le meilleur $h[k]$ va annuler la dérivée de l'erreur (ou risque r) comme dans le cas linéaire. Ainsi, nous obtenons

$$\frac{\partial r}{\partial h[k]} = 0 \Leftrightarrow h[k] = \frac{\langle x, e_k\rangle^2}{\langle x, e_k\rangle^2 + \sigma^2}, \qquad r_{min} = \sum_k \frac{\sigma^2\langle x, e_k\rangle^2}{\langle x, e_k\rangle^2 + \sigma^2} \quad (191)$$

On retombe sur l'expression des *v.p* de l'opérateur H dans le cas linéaire (Eq. 179). Cependant, si on obtient bien une expression du meilleur $h[k]$, ce calcul est vain car **on ne connait pas** x! Cependant, cet estimateur *idéal* est ce que l'on nomme un **oracle**. Les questions sont de savoir s'il est possible de faire aussi bien, et l'estimateur est-il optimal? Ces questions sont en relation avec les propriétés de la base orthonormale choisie. S. Mallat mentionne qu'il y a deux chercheurs qui ont marqué le domaine en répondant à ces questions, il s'agit de David L. Donoho et Iain M. Johnstone, dont la première publication date de 1994 [122].

La première idée qui est venue est de simplifier l'expression de $h[k]$. Si on se réfère au cas linéaire, les meilleurs $\lambda_k = \mathbb{E}_X[\langle X, e_k\rangle]$ peuvent très bien être filtrés par un **filtre passe-bas idéal**. Ainsi, essayons une expression de $h[k]$ qui vaut soit 1 soit 0 (**filtre binaire**). Le choix optimal alors selon le critère de la minimisation de r se porte sur le seuil qui opère la transition $0 \leftrightarrow 1$. Il vient alors:

$$h[k] = \begin{cases} 1 & \text{si } |\langle x, e_k\rangle| \gg \sigma \\ 0 & \text{si } |\langle x, e_k\rangle| \ll \sigma \end{cases} \quad (192)$$

Certes l'expression est simplifiée, mais il s'agit toujours d'un orale à cause de la dépendance vis-à-vis du signal x inconnu. Quelle serait néanmoins l'erreur dans ce cas? On la

121. nb. $\mathbb{E}_Z[\langle Z, e_k\rangle] = 0$, $\mathbb{E}_Z[|\langle Z, e_k\rangle|^2] = \sigma^2$.
122. David L. Donoho and Iain M. Johnstone, *Ideal Spatial Adaptation by Wavelet Shrinkage*, Biometrika, 81(3):425–455, 1994. NDJE. Leur rapport à Stanford date de 1992. https://imjohnstone.su.domains/WEBLIST/1994/isaws.pdf.

note r_b pour *risque binaire*:

$$r_b = \sum_k \min(|\langle x, e_k \rangle|^2, \sigma^2) \tag{193}$$

à comparer à l'expression du risque minimum (Eq. 191). Or, pour toute constante (a, b) nous avons

$$\frac{1}{2}\min(a, b) \leq \frac{ab}{a+b} \leq \min(a, b) \tag{194}$$

donc

$$\frac{r_b}{2} \leq r_{min} \leq r_b \tag{195}$$

Ainsi, si on opte pour le choix binaire, on perd au pire un facteur 2, ce qui n'est pas très grand dans le cas où l'on cherche des calculs asymptotiques à une constante près.

Cependant, il s'agit toujours d'un résultat d'oracle. Mais convenons que cela pose le problème en termes génériques: **quand doit-on garder ou éliminer le signal**? La figure 43 peut servir de support à la réflexion. Admettons un instant que la base a bien été choisie, il reste que l'on a accès uniquement aux coefficients $\langle y, e_k \rangle$ du signal bruité $y = x + z$. Or, on espère que les coefficients du bruit z soient petits par rapport aux quelques coefficients les plus significatifs du signal, sinon on observerait que du bruit. Dans ce contexte, on peut alors appliquer un seuillage de $|\langle y, e_k \rangle|$ selon (Fig. 43):

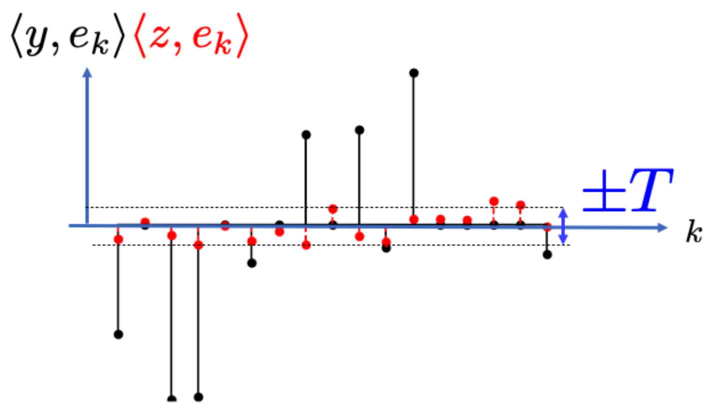


FIGURE 43 – Distribution des produits scalaires du signal bruité y et du bruit z. Un seuillage en $\pm T$ peut enlever le bruit sans trop dénaturer les caractéristiques du signal x.

$$h[k] = \begin{cases} 1 & \text{si } |\langle y, e_k \rangle| \geq T \\ 0 & \text{si } |\langle y, e_k \rangle| < T \end{cases} \tag{196}$$

Reste alors dans ce contexte à choisir la valeur du seuil T, et pour quelle efficacité? L'idée est de choisir T pour que si l'on garde un coefficient, alors il a une très grande probabilité d'être plus grand que le bruit. On regarde donc les valeurs maximales que le bruit peut atteindre. Notez que le bruit z est gaussien, donc plus on échantillonne la distribution, bien que beaucoup d'entre eux se trouvent dans l'intervalle $[-\sigma, \sigma]$ (68% C.L), plus on explore les queues de distribution. Or, nous avons le résultat suivant

$$\mathbb{E}_{Z\sim\mathcal{N}(0,1)}[\max_{k\leq d}(|\langle Z, e_k|)] = \sqrt{2\log d}(1 + o(1)) \tag{197}$$

avec $o(1)$ de l'ordre de $\log(\log d)/\log d$. A présent, l'opérateur H peut être défini pour tout signal x selon l'expression

$$H[x] = \sum_{k=1}^{d} \rho_T(\langle x, e_k \rangle)\, e_k \tag{198}$$

avec par exemple la fonction de seuillage suivante (**hard threshold**)

$$\rho_T^{hard}(a) = \begin{cases} a & \text{si } |a| \geq T \\ 0 & \text{si } |a| < T \end{cases} \tag{199}$$

Remarquons cependant que la fonction ρ_T ainsi définie est discontinue: à quelque chose près un coefficient proche du seuil peut être soit gardé soit rejeté, ce qui n'est pas optimal quand on revient à la prédiction de l'oracle "non binaire" (Eq. 191). Cela motive l'usage d'un seuillage *doux* (**soft threshold**) défini selon (Fig 44):

$$\rho_T^{soft}(a) = \begin{cases} a - T & \text{si } a \geq T \\ a + T & \text{si } a \leq -T \\ 0 & \text{si } |a| < T \end{cases} \tag{200}$$

L'idée d'atténuation de la valeur de a est que le bruit a contaminé également les coefficients les plus significatifs du signal sous-jacent. Notons que le seuillage doux est Lipschitz, et l'on peut le réécrire à l'aide de 2 ReLU [123]

$$\rho_T^{soft}(a) = ReLU(a - T) - ReLU(-a - T) \tag{201}$$

Si l'on formule l'action de H sur le signal bruité

$$H_T[Y] = \sum_k \rho_T(\langle Y, e_k \rangle)\, e_k \tag{202}$$

on se rend compte que l'on a une sorte de U-Net primitif avec: **une décomposition sur une base pour obtenir $\langle Y, e_k \rangle$, suivie par une non-linéarité ρ_T et une reconstruction $\sum_k (.) e_k$.** On a donc un algorithme de débruitage effectif, et la question est de savoir si cela est effectif.

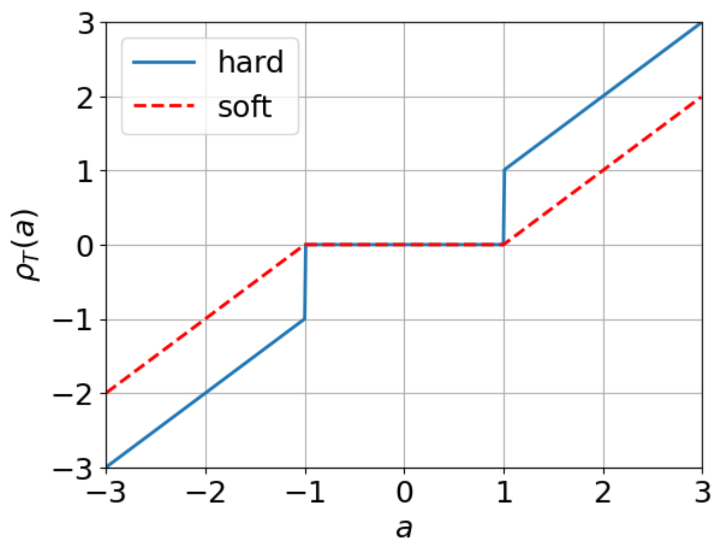


FIGURE 44 – Exemple de seuillage "hard" (Eq. 199) et "soft" (Eq. 200) avec $T = 1$.

Donoho et Johnstone ont montré le très joli résultat suivant:

Théorème 13 *(**Donoho et Johnstone**, 1994)*

123. $ReLU(x) = \max(0, x)$.

Si $T = \sigma\sqrt{2\log d}$, alors l'erreur de seuillage (soft ou hard) $r_s(x)$ est bornée selon

$$r_s(x) = \mathbb{E}_Z[\|H_T[Y] - x\|^2] \leq (2\log d + 1)\left(\sigma^2 + \begin{cases} r_b(x) \\ r_{min}(x) \end{cases}\right) \tag{203}$$

De plus, le facteur $2\log d$ est optimal ($d \to \infty$) dans le cas d'opérateur diagonal.

Ce théorème nous dit que si l'on choisit bien le seuil, l'efficacité est bonne par rapport au résultat de l'oracle. A une constante près, on a un risque qui est $2\log d$ plus grand que le risque d'un oracle (binaire ou non). Notez que la constante $2\sigma^2\log d = T^2$, il s'agit du risque pris de ne retenir qu'un coefficient au dessus du bruit. Le second résultat du théorème nous dit que l'on ne peut trouver pour tout x une constante multiplicative inférieure à $2\log d$. **On a donc un algorithme non-linéaire efficace quasi-optimal, et il est très simple à mettre en œuvre** [124].

A partir de ce résultat très intéressant, S. Mallat nous dit que la communauté s'est attelée à trouver la **bonne base**. En quoi une base peut-elle être bonne?

8.2 Bases parcimonieuses

Rappelons que dans le cas linéaire, la base adéquate est celle qui diagonalise la covariance (PCA), qui dans le cas stationnaire est la base de Fourier. Dans le cas non-linéaire, c'est un peu plus complexe, car il s'agit de minimiser le risque *binaire* (Eq. 193). Or, nous pouvons expliciter la somme sur k selon la formulation suivante

$$r_b = \sum_k \min(|\langle x, e_k\rangle|^2, \sigma^2) = \underbrace{\sum_{|\langle x,e_k\rangle|\leq\sigma} |\langle x, e_k\rangle|^2}_{biais} + \underbrace{M\sigma^2}_{variance} \tag{204}$$

124. NDJE. Remarquons qu'il reste à déterminer la valeur de σ. Vous pourrez voir comment cela s'obtient dans les notebooks du repository Github: `Wavelet_transitoires.ipynb` et `Wavelet_stationnary_transitoires.ipynb`. Des librairies telles que `PyWavelets` en Python et `Mathematica` donnent accès à d'autres fonctions de seuillage qui réalisent des intermédiaires entre le seuillage *hard* et *soft*.

où M est le nombre de coefficients tels que $|\langle x, e_k \rangle| > \sigma$. On aimerait minimiser les deux termes de cette somme en gardant à l'esprit que

$$\sum_k |\langle x, e_k \rangle|^2 = \|x\|^2 = cte \tag{205}$$

Or, on peut interpréter le second terme de la façon suivante: si on garde M coefficients dont l'énergie est au-dessus du bruit, on garde également un peu de bruit [125] dont l'énergie est $M\sigma^2$. Il s'agit d'un *terme de variance*, car en quelque sorte il renseigne sur les fluctuations de l'estimateur de débruitage. Le premier terme concerne tous les coefficients de la décomposition que l'on a éliminés. De ce fait, on a supprimé une partie de l'énergie du signal, ce terme est donc identifié comme un *terme de biais* d'estimation. Notons qu'un cas idéal serait réalisé s'il existait qu'un unique vecteur de la base qui rende compte de toute l'énergie du signal. Dans ce cas, nous aurions comme risque binaire la valeur $r_b = \sigma^2$. D'une manière moins radicale, on cherche à obtenir un **faible nombre M de coefficients qui emportent la quasi-totalité de l'énergie** (c'est-à-dire que les autres coefficients sont quasi-nuls), et dans ce cas $r_b \approx M\sigma^2$. Ainsi, **on cherche des bases pour lesquelles la représentation du signal est la plus parcimonieuse possible.**

S. Mallat nous dit que dans ce contexte, il y a beaucoup d'articles qui explorent en quoi/comment les réseaux de neurones débruiteurs font un calcul de représentations parcimonieuses. Ceci dit, il y a tout un chapitre de mathématiques qui traite de ce type de représentations que nous brosse rapidement S. Mallat à travers quelques slides projetés. *NDJE. on peut en même temps se référer à son cours de 2021.*

8.2.1 Petit retour sur Fourier

Notation: on se place en dimension infinie, le signal $x(u)$ devient une fonction qui est notée $f(t)$ par la suite. Pourquoi est-ce important de sortir du monde *discret* pour opter pour une vision *continu*: on veut par exemple comprendre les propriétés des coefficients dans la base, et les reliés aux **propriétés de régularités de la fonction** avec les notions de continuité, dérivée, etc. Le passage *discret-continu* s'effectue en fixant le support de u ex. $[0, 2\pi]$, $[0, 1]$, etc tout en augmentant la densité d'échantillons (ou la valeur de d). Notez

125. rappel: le signal total est bruité par la v.a Z.

que dans le cadre de Fourier, pour des fonctions $f \in L^2([0, 2\pi])$, nous avons

$$\hat{f}(\omega) = \langle f(t), e^{i\omega t} \rangle = \int_0^{2\pi} f(t)\, e^{i\omega t}\, dt \tag{206}$$

qui interprète $\hat{f}(\omega)$ comme le produit scalaire de $f(t)$ avec la sinusoïde $e^{i\omega t}$: $\hat{f}(\omega)$ rend compte des variations de f à la "fréquence" ω. Or, si f est régulière, il y a plus de composantes à basses fréquences qu'à hautes fréquences. La théorie de Fourier relie la vitesse de décroissance de $\hat{f}(\omega)$ avec la régularité globale de f (Fig. 45-haut).

Le problème dans l'analyse de Fourier se manifeste quand on rajoute un petit pic à f de régularité C^α: comme la transformation de Fourier est linéaire, par superposition tous les coefficients de f sont affectés, et notamment ceux à haute fréquence (Fig. 45-bas). Notons au passage que si on pratiquait une coupure "hard" sur les fréquences, ex. $\omega < 10$, pour ne garder que les coefficients à basse fréquence les plus importants, on perdrait toute description du petit pic.

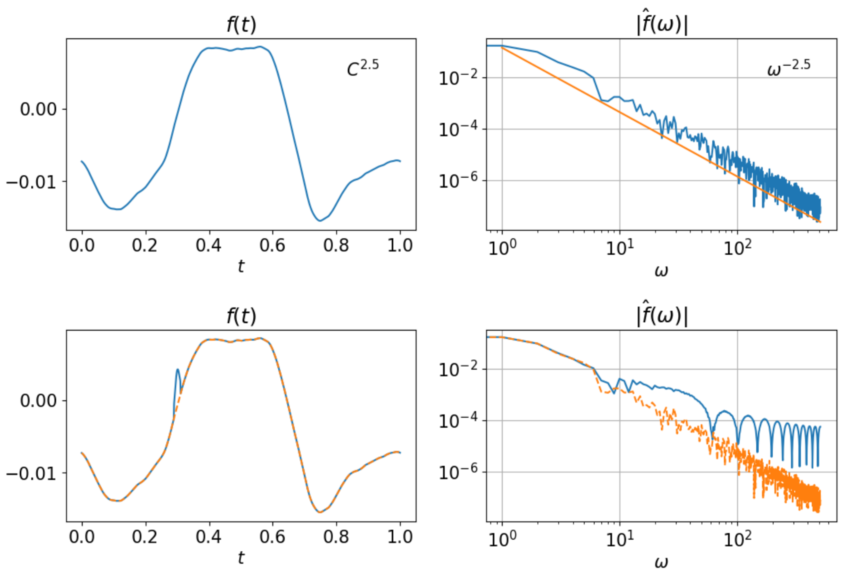


FIGURE 45 – Exemple de traitement par Fourier d'une fonction de régularité $C^{2.5}$ (en haut) à laquelle on ajoute un petit pic localisé en temps (en bas). On remarque que dans le premier cas la décroissance des coefficients de Fourier renseigne bien de la régularité de la fonction (voir la courbe orange $\propto \omega^{-2.5}$ sur le spectre), et dans le second cas le spectre à haute fréquence est dominé par celui de la perturbation (en orange le spectre initial).

La moralité est que si la fonction est régulière, la représentation de Fourier est parcimonieuse, mais **dès qu'il y a une singularité ou un transitoire, la parcimonie est perdue**.

8.2.2 Analyse par ondelettes

NDJE. vous pouvez lire aussi la section Cours 2021 Sec.5.3, car S. Mallat dans le temps imparti va à l'essentiel pour montrer le cheminement mathématique.

La question est de savoir comment construire des bases orthonormales qui utilisent la régularité globale du signal tout en tenant compte des singularités afin de produire une représentation parcimonieuse? On aimerait en fait que le nombre de coefficients significatifs ne soit pas affecté par la présence des singularités. Notons que sur la figure 45 du bas, pour garder le pic qui peut être une caractéristique importante, il nous faut garder tous les coefficients dont la valeur est supérieure à 10^{-4}. Il faut alors pouvoir donner une propriété de **localisation spatiale des vecteurs de la base**, comme illustré sur la figure 46. C'est l'idée développée dans les années 1980 indépendamment par **Jean Morlet** et **Alex Grossmann**, le premier pour l'analyse de phénomènes sismiques dans la recherche de pétrole chez Elf-Acquitaine et, le second pour l'étude des états cohérents en physique quantique: il s'agit de l'**analyse par ondelettes**.

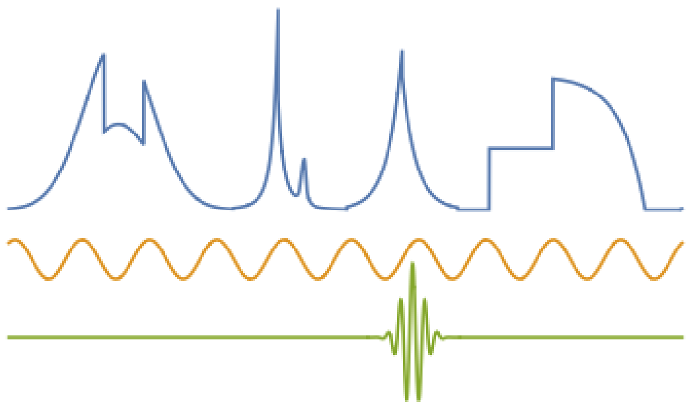


FIGURE 46 – Différents points de vue de l'analyse d'une fonction: soit le cadre *linéaire* qui prend pour partie une régularité uniforme ce qui donne l'analyse de Fourier avec des sinusoïdes délocalisées en temps/espace, soit le cadre *non-linéaire* qui étudie des fonctions non uniformément régulières avec l'analyse par Ondelettes qui opère une analyse locale des transitoires/discontinuités.

L'idée est de partir d'une fonction localement oscillante $\psi(t)$ de moyenne nulle, soit telle que

$$\int_{-\infty}^{\infty} \psi(t)\, dt = 0 \tag{207}$$

Pour s'accommoder des échelles de temps et localisation des discontinuités du signal, on opère sur ψ des translations et dilatations (Fig. 47):

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right) \tag{208}$$

le support est donné par le terme d'échelle s, et son domaine en fréquence [126] est multiplié par $1/s$.

126. nb. $\frac{1}{s}h(\frac{t}{s}) \xrightarrow{TF} \hat{h}(s\,\omega)$

Figure 47 – Illustration des opérations de translation et changement d'échelle appliquée à une ondelette pour obtenir différentes versions de $\psi_{u,s}$ où s est le support de la nouvelle ondelettes.

Comme pour Fourier, on calcule des produits scalaires

$$\langle f, \psi_{u,s} \rangle = \int f(t) \psi_{u,s}(t)\, dt \tag{209}$$

et l'on produit des cartes *temps-échelle* ou *temps-fréquence* (*scalogramme*) comme celle de la figure 48. Pour chaque point de la carte est représentée la valeur du produit scalaire. On note 1) que la quasi-totalité des coefficients sont nuls, car si l'ondelette est bien localisée spatialement (petite échelle, grande fréquence) le produit scalaire est localement proportionnel à la moyenne de l'ondelette qui est nulle; 2) que les coefficients les plus significatifs à échelle fixée sont localisés aux "instants" des discontinuités particulièrement à hautes fréquences et, 3) plus la fréquence baisse plus la localisation est de moins en moins effective.

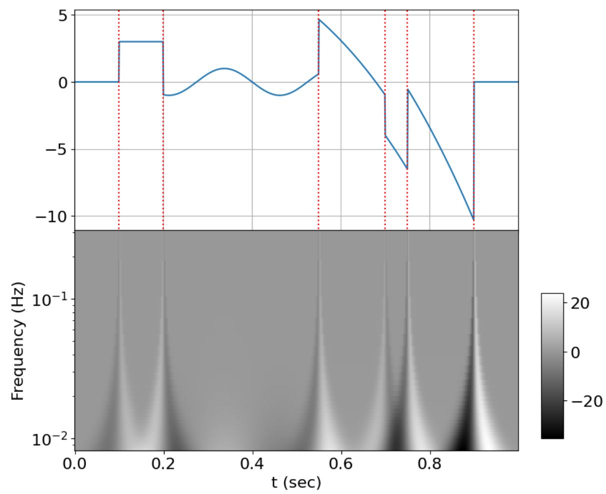


FIGURE 48 – En haut: signal régulier par morceaux. En bas: carte avec en abscisse le paramètre de de translation et en ordonnée la fréquence (grande fréquence = petite échelle). Ici on a utilisé l'ondelette réelle en forme de chapeau mexicain ("mexh" pour la librairie `PyWavelets`.)

A partir de tous les coefficients/produits scalaires, on peut reconstruire la fonction f, il s'agit de la **transformée en ondelette inverse continue**:

$$f(t) = C \iint \langle f, \psi_{u,s} \rangle \psi_{u,s}(t)\, du \frac{ds}{s^2} \tag{210}$$

De même, on peut se poser la question de la **caractérisation de la régularité** de f via la décroissance des produits scalaires comme en Fourier? et surtout peut-on **construire des bases orthonormales**?

On peut remarquer que la description en termes de cartes *temps-fréquence* comme celle de la figure 48 est très redondante, ce n'est pas optimal en termes de parcimonie. On va pouvoir réduire suffisamment l'information sans perte, en pratiquant un **échantillonnage dyadique**: $s = 2^j$, idem pour les translations. A partir de là, on peut construire des bases orthonormales parcimonieuses.

Si l'on fixe $s = 2^j$, soit alors $\psi_j(t) = \frac{1}{\sqrt{2^j}} \psi\left(\frac{-t}{2^j}\right)$, alors on se rend compte que le coefficient/produit scalaire de f avec $\psi_{u,2^j}$ est le résultat d'une convolution car:

$$d_j(u) = \langle f, \psi_{u,2^j} \rangle = \int f(t) \frac{1}{\sqrt{2^j}} \psi\left(\frac{t-u}{2^j}\right) dt = f * \psi_j(u) \tag{211}$$

Donc, **les ondelettes ψ_j sont des filtres passe-bande**[127] de largeur typique 2^{-j} (Fig. 49).

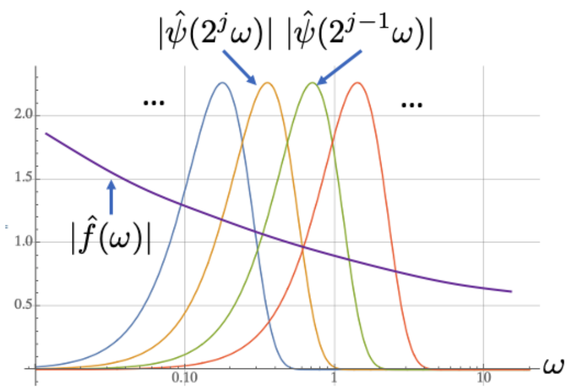


FIGURE 49 – Collection de filtres $\hat{\psi}(2^j\omega)$ qui analysent le signal f.

On peut alors disposer d'une collection (infinie) de coefficients $\hat{d}_j(\omega)$ qui rend compte de la puissance du signal dans tous les filtres passe-bandes:

$$\hat{d}_j(\omega) = \hat{f}(\omega)\, \sqrt{2^j}\, \hat{\psi}(2^j\omega) \tag{212}$$

La reconstruction du signal à partir des $\{d_j\}_{j\in\mathbb{Z}}$ est possible si les filtres $\hat{\psi}(2^j\omega)$ se recouvrent pour **ne pas laisser de trous dans le spectre de Fourier**. Il s'agit de la condition de Littlewood-Paley (les années 1930):

$$\sum_j |\hat{\psi}(2^j\omega)|^2 = 1 \Rightarrow f(t) = \sum_{j\in\mathbb{Z}} \int d_j(u)\, \psi_{u,2^j}(t)\, du \tag{213}$$

Donc, on peut se limiter à des échelles $s = 2^j$. Maintenant peut-on comprimer l'information des translations par exemple en échantillonnant l'axe des temps selon $u = 2^j n$ comme sur la figure 50?

127. nb. en fait $\hat{\psi}(0) = \int \psi(t)\, dt = 0$ donc on a bien un filtre dont la partie basse fréquence tend vers 0.

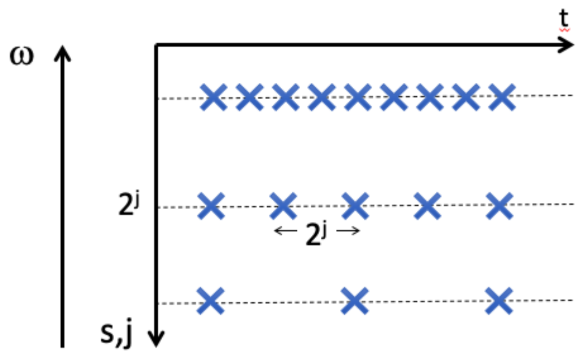


FIGURE 50 – Échantillonnage dyadique de la carte "temps-échelle" (ou "temps-fréquence").

La réponse vient de travaux de **Alfréd Haar** (1909) qui en utilisant une ondelette particulière (terme inconnu à l'époque) ψ (Fig. 51) a construit **la première base orthonormale de** $L^2(\mathbb{R})$:

$$\left\{\psi_{j,n}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t-2^j n}{2^j}\right)\right\}_{(j,n\in\mathbb{Z}^2)} \tag{214}$$

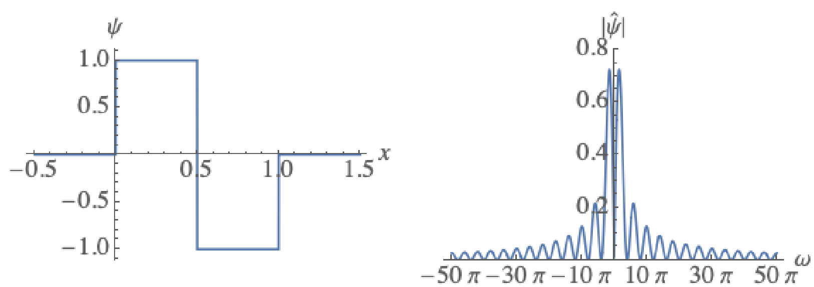


FIGURE 51 – "Ondelette" de Haar (également "db1" dans la famille des ondelettes d'I. Daubechies) construite dans l'espace réel. Décroissance selon $1/\omega$ dans l'espace de Fourier à cause des discontinuités.

Puis, **Claude Shannon** et **Harry Nyquist** dans les années 1948-49 démontrent *le théorème d'échantillonnage* à l'aide de filtre passe-bandes parfaits, et l'on peut l'interpréter en termes de décomposition en ondelettes (Fig. 52).

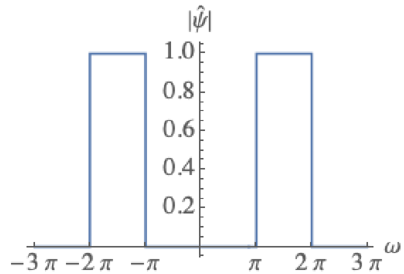


FIGURE 52 – Ondelette de Shannon construite dans l'espace de Fourier. Décroissance en $1/t$ dans l'espace réel.

Le problème des ondelettes de Haar et Shannon, c'est que dans le domaine réel la première est discontinue et la seconde décroit très lentement. Or, on cherche des "sinusoïdes locales" (Fig. 46) afin non seulement de permettre la localisation des discontinuités du signal, mais aussi de constituer des bases orthonormales. Mais est-ce possible? On pensait que non et le sujet était laissé de coté. Il a fallu qu'**Yves Meyer** en 1986 (prix Abel 2017) essaye de prouver que cela n'était pas possible, pour qu'il trouve une solution! Un exemple d'ondelette de Meyer est donné sur la figure 53: c'est une fonction (esp. réel) C^∞ à décroissance rapide, et dans l'espace de Fourier c'est un filtre passe-bande. **On a donc une base orthonormale d'ondelettes qui satisfont nos critères de localisation dans l'espace réel et l'espace de Fourier** [128].

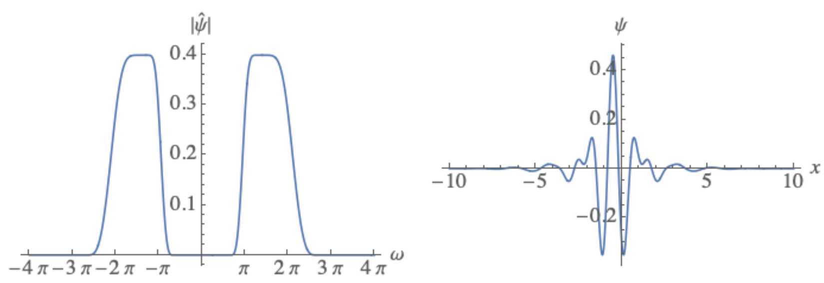


FIGURE 53 – Exemple d'une ondelette de Meyer dont la construction se fait dans l'espace de Fourier. Notez sa décroissance rapide dans l'espace réel tout en étant localisée dans l'espace de Fourier.

128. nb. Le principe d'incertitude contraint à ce que l'on ne peut pas réaliser une localisation parfaite simultanément dans les deux espaces.

A partir de ce résultat fondamental, il s'en est suivi tout un domaine de recherche en particulier l'**Analyse Multi-résolution** (ou Multi-échelle), la recherche d'**ondelettes à support compact**, le développement d'**algorithmes de transformée en Ondelette rapides** avec des cascades, etc. Nous verrons à la séance prochaine comment cela nous permet de réaliser un **débruitage par seuillage** sur les quelques coefficients qui emportent toute l'énergie du signal (1D, 2D). On se posera la question alors: quand **est-ce que le débruitage est optimal**? Nous verrons qu'**en 1D, il y a beaucoup de situations où l'optimum est atteint, mais en 2D (image) dès qu'il y a de la régularité géométrique on perd l'optimalité.** Il nous faudra changer notre point de vue. Dans ce contexte, nous verrons que **les réseaux de neurones** calculent des bases beaucoup plus subtiles que les ondelettes: tout en partagent des propriétés d'harmonicité (oscillation), ils s'adaptent à **la géométrie du signal**.