

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. 3 février 2025

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

Table des matières

1 Avant-propos	4
2 Séance du 15 Janv.	4
2.1 Quelques faits marquants de l'année écoulée	5
2.2 Panorama de la génération par IA	7
2.3 Retour sur la malédiction de la dimension	8
2.4 Les modèles auto-régressifs LLM	12
2.5 Données multi-diemsionnelles	13
2.6 Modèles probabilistes avec énergie de Gibbs	14
2.7 Generative Adversarial Networks (GAN)	15
2.8 Modélisation par transport	18
2.8.1 Normalizing Flows (NF)	18
2.8.2 Transport par Score-diffusion	20
3 Séance du 22 Janv.	27
3.1 Modèles d'énergie de Gibbs	28
3.1.1 Typologie des paramétrisations	28
3.1.2 Optimisation des paramètres	30
3.2 Échantillonnage d'une $pdf \pi(x)$: chaines de Markov	34
3.3 La métrique de Fisher	39
3.4 Modèles à base de réseaux: les GAN	40

4 Séance du 29 Janv.	44
4.1 Retour sur les GANs: pourquoi ça coince?	44
4.2 Transport optimal	47
4.2.1 Le problème de G. Monge	47
4.2.2 Relaxation de L. Kantorovich	48
4.3 GANs conditionnels (cGAN)	52
4.4 Évaluation des GANs: critères de fidélité	53
4.5 Normalizing Flows (NF)	56
4.5.1 L'effet d'un transport	57
4.5.2 Optimisation de la vraisemblance	58

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 précédentes¹.

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. Mallat² donne en libre accès des chapitres de son livre "*A Wavelet Tour of Signal Processing*", 3ème édition. 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 GitHub³, 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 cœur 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 DeepMind⁴ à prédire la structure tridimensionnelle de presque toutes les protéines connue. 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 à 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 remarqué que G. Hinton a eu un impact dans le domaine de l'apprentissage notamment concernant l'algorithme de base de la back-propagation⁵ mais aussi sur toutes les architectures profondes. Concernant J. Hopfield, l'impact a été surtout dans le pont qu'il a tissé entre apprentissage neuronal et Physique Statistique⁶. Ce travail a été fondamental et d'ailleurs on l'a

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

déjà appréhendé dans les cours précédents⁷ 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 *dédiction* qui est celle utilisées 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 DeepMind¹⁰. 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 delà 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 cours https://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 DeepMind¹¹. **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 Google¹²: 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 JAX¹³ comme librairie de développement, et vous noterez qu'un certain nombre des notebooks que j'ai mis à votre disposition sont écrits dans ce langage¹⁴.

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 (entièrre ou continue). La stratégie est d'utiliser une base de données d'entraînement 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-forecasting/> 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 de 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’entraînement mais non labellisée, constituée d’échantillons $\{x_i\}_{i \leq n}$ que l’on suppose **indépendants identiquement distribués (*iid*)** (hypothèse fondamentale). A 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 degré 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

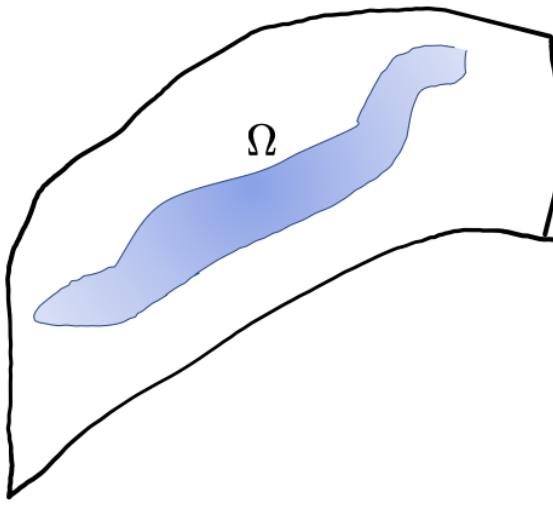


FIGURE 1 – Schématisation de la localisation des données sur une surface de dimension d' plongée dans l'espace de dimension d .

image d'un intérieur de maison, ne serait-ce que la texture des matériaux introduit une complexité qui fait exploser le nombre de degré 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 exemple¹⁶ $x_i \sim \mathcal{N}(0, \sigma^2/d)$. Ainsi

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

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.

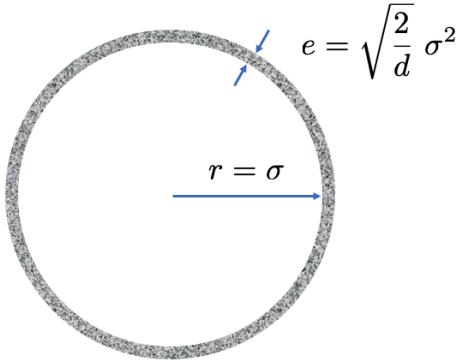


FIGURE 2 – Localisation de $\|x\|^2$ si toutes les composantes sont iid $x_i \sim \mathcal{N}(0, \sigma^2/d)$ en dimension d .

que $p(z)$ est la distribution du χ^2 à d degrés de liberté¹⁷ dont l'espérance est $\mathbb{E}[z] = d$. Par changement de variables, on obtient alors

$$\|x\|^2 = \sum_{i=1}^d x_i^2 \xrightarrow[d \rightarrow \infty]{} \sigma^2 \quad (2)$$

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

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

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 degré 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 adéquat 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 quelle est dérivable presque partout, soit $\forall x, y$ dans le domaine de définition il existe une constante K telle que

$$|p(x) - p(y)| \leq K\|x - y\| \quad (4)$$

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)

$$n \geq \varepsilon^{-d} \quad (5)$$

Remarquons que si l'on procède à un histogramme en découplant 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 dimensionalité.

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 $\{x_1, x_2, \dots, x_{t-1}\}$ et tenter de prédire x_t où t peut matérialiser un temps discret. Si par la suite l'on dispose de la pdf conditionnelle $p(x_t|x_1, x_2, \dots, x_{t-1})$, 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:

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

Notons au passage que l'hypothèse de Markov simplifie le problème (Cours 2023) en requérant que $p(x_t|x_1, \dots, x_{t-1}) = p(x_t|x_{t-1})$, 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 $p(x_t|x_1, \dots, x_{t-1})$. C'est là où les **Transformer** introduits en 2017¹⁸ 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'étudieront 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>

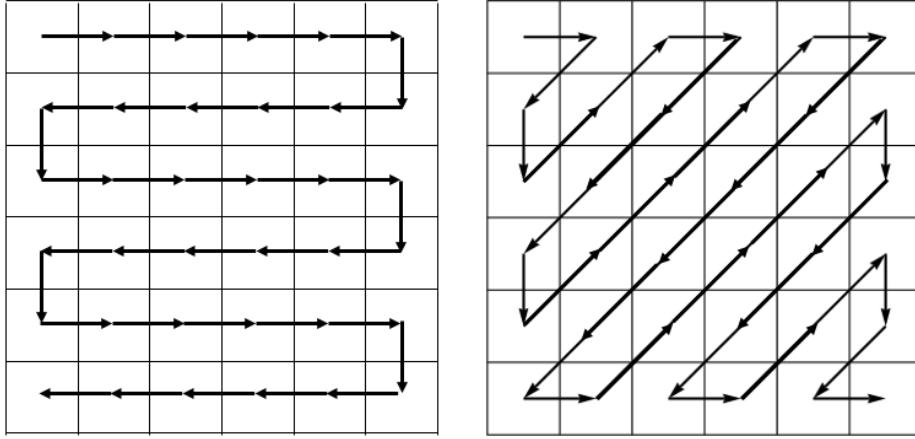


FIGURE 3 – Deux façons d’indexer les pixels d’une image.

2.5 Données multi-dimensionnelles

Le cadre des LLM présenté ci-dessus n'est pas bien adapté aux cas de données multi-dimensionnelles 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-dimensionnelle 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 $p(x)$. La choix de la famille est bien entendu crucial. Si l'on considère que $p(x)$ est non nulle, le log est alors défini et la forme exponentielle est un choix classique

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

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 introduite 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 de configurations différentes. Certes, si on se contentait de générer que des x correspondant à 1 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

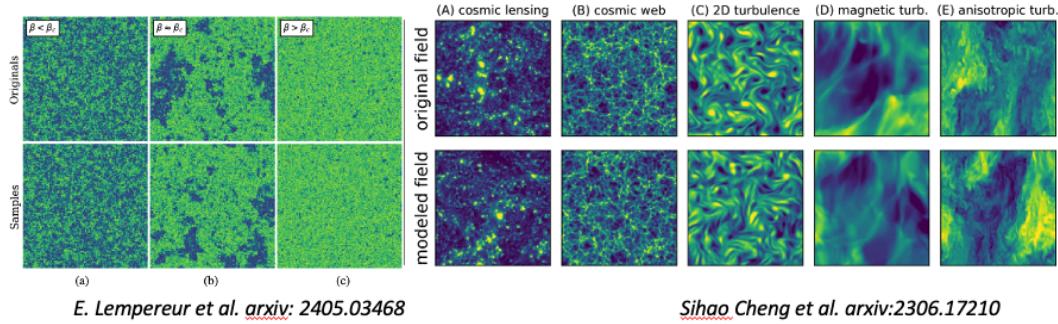


FIGURE 4 – Exemple de générations d’images de champs de Physiques.

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 est la probabilité est donnée par la forme exponentielle citée et 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’ergodicité 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)

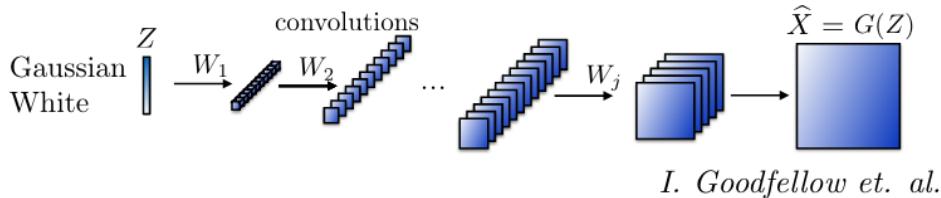
Les GAN ont été introduits par Goodfellow et al.²⁰ en 2014. Nous les avons évoqués durant le Cours de 2019 Sec. 2.8., et le schéma à l’origine est donné sur la figure 6. Deux réseaux agissent en adversaires: le réseau génératif (G) qui par exemple à partir d’un bruit

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 Diederik P. Kingma et al. *arxiv:1807.03039*), à droite par un modèle de type Generative Adversarial Networks (source: Xin Wang et al. *arxiv:2202.07145*).

- Generative network for non-stationary processes:



- Discriminative network:

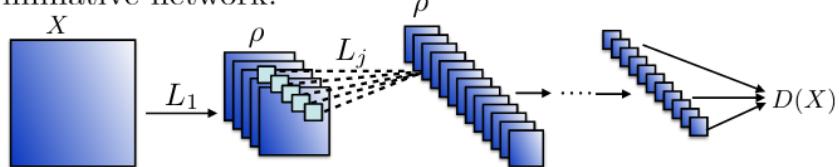


FIGURE 6 – Schéma d'un GAN de Goodfellow et al. avec deux réseaux adversaires: un générateur et un discriminant (classificateur).

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'entraînement (*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:

$$\min_G \max_D \left\{ \mathbb{E}_{x \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{z \sim \pi(z)} [\log(1 - D(G(z)))] \right\}, \quad (8)$$

Malgré les problèmes de mise en oeuvre que nous allons mentionné 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 précipitations²¹ qui en 2021 était capable de faire des prédictions à 90 minutes sur des zones typique de 1.3km 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 a 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'œuvre²² mais finalement S. Mallat nous dit que le problème d'optimisation est assez mal maîtrisé. **Le danger principal est de biaiser les générations et d'entraîner des formes de mémorisation de la base d'entraînement.** 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'entraînement reproduits par le générateur.

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.

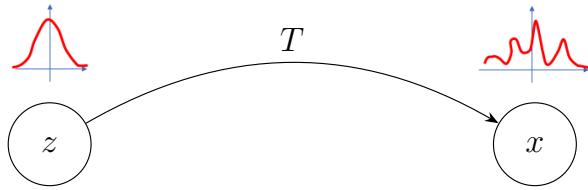


FIGURE 7 – Transport de la pdf des variables latentes z vers la pdf des données x .

2.8 Modélisation par transport

En fait, ce que l'on a chercher 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.

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

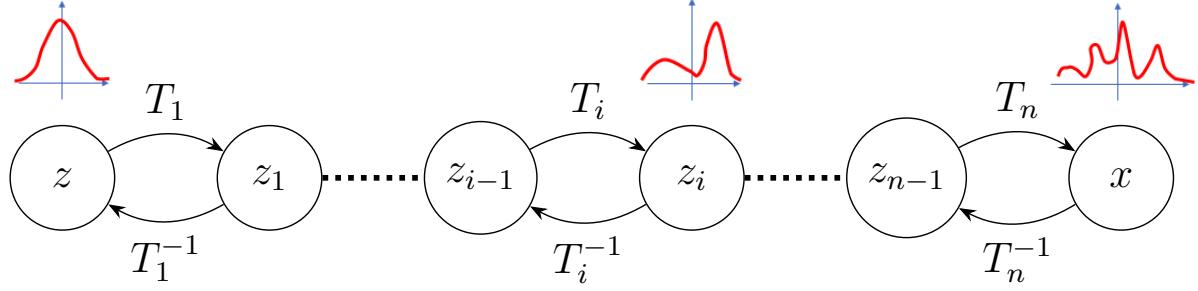


FIGURE 8 – Schematic Normalizing Flow process. On the top the *forward* or *generative* direction from a simple \mathbf{z} distribution to a more complex one for \mathbf{x} . On the bottom the *backward* or *training* direction from complex to simple distributions.

que:

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

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

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

où J_T et $J_{T^{-1}}$ sont les jacobiens des transformations T et T^{-1} , respectivement. Contrairement aux GAN, 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:

$$T = T_1 \circ T_2 \circ \dots \circ T_n \Leftrightarrow T^{-1} = T_n^{-1} \circ T_{n-1}^{-1} \circ \dots \circ T_1^{-1}. \quad (11)$$

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

utilisant $\mathbf{z}_0 = \mathbf{z}$ et $\mathbf{z}_n = \mathbf{x}$), alors:

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

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

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

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

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

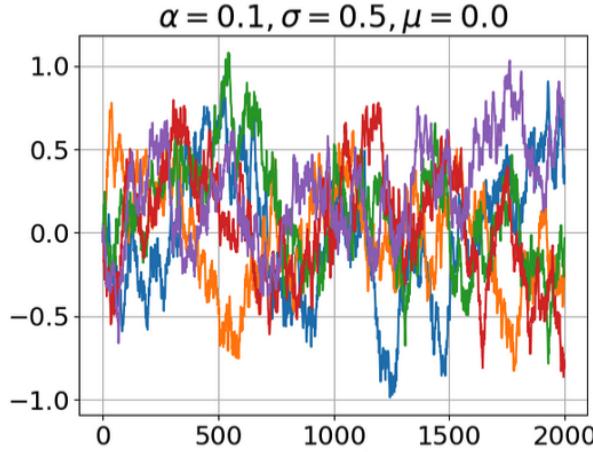


FIGURE 9 – Exemples de réalisations de transports selon l'équation différentielle stochastiques $dX = -\alpha X \ dt + \sigma \ dW$ avec $dW = \mathcal{N}(0, dt)$ ($\alpha = 0.1, \sigma = 0.2$).

où dW_t représente un processus de Wiener (mouvement brownien) et T_{max} représente la durée maximale du transport supposée suffisamment grande. A chaque étape (t_i, t_{i+1}) le transport n'est plus déterministe comme pour les Normalizing Flows mais probabiliste²³ (Fig. 9). La pdf $p_t(x_t)$ est régie comme pour les ODE, par une équation différentielle, qui est une équation²⁴ 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:

$$v_t(x) = \nabla_x \log p_t(x) = -\nabla U_t(x) \quad (15)$$

où $\nabla_x \log p(x)$ s'appelle le **score**²⁵, ce qui nous amène aux algorithmes de **Score Diffusion**.

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)

25. NDJE. Voir Cours 2024 Sec. 5.2 la remarque que je fais sur cette appellation car il faut bien

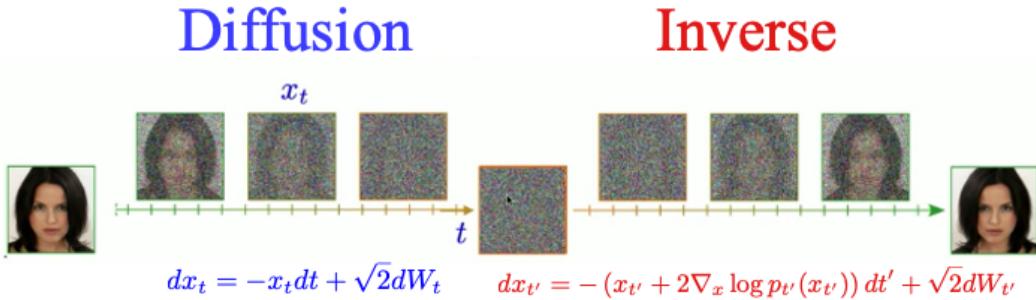


FIGURE 10 – Processus de Diffusion et Inverse permettant de déterminer le transport entre un bruit blanc et la pdf des données.

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 paraît 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 disparaîsse). 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

$$dx_t = -x_t dt + \sqrt{2} dW_t \quad (16)$$

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

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.

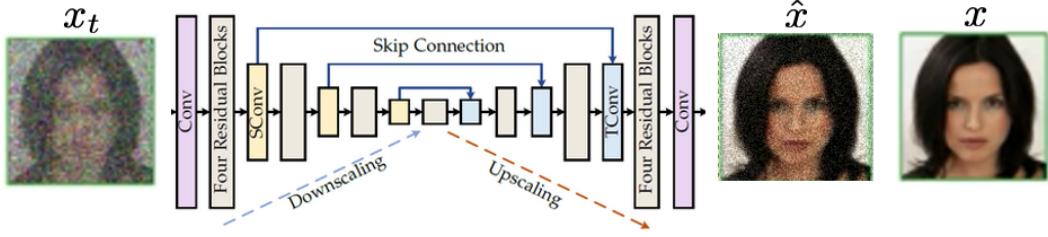


FIGURE 11 – Usage d'un réseau profond pour effectuer une tâche de débruitage. Voir par exemple Kai Zhang et al. (2021) *arXiv:2008.13751* (Sec. 3.1). A gauche: l'image x_t à débruiter; à droite: \hat{x} résultat du réseau de neurones et x l'image telles que le réseau doit minimiser $\|x - \hat{x}\|^2$ 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 Langevin²⁷ amortie** qui s'écrit

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

où apparaît 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

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

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

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

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)

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

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-Net²⁸, entraînons-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

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

question fondamentale sous-jacente: **les résultats sont-ils vraiment des échantillons de $p(x)$** , 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'entraînement, 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'entraînement, 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, fusse-t'elle 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'article²⁹ de Kadkhodaie et al. (2024) que S. Mallat avait évoqué dans le Cours de 2024 (Sec. 9.4). Le point qui les a étonné est qu'au fur et à mesure que la taille de la base d'entraînement 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 du 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-linéaire³⁰ 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.

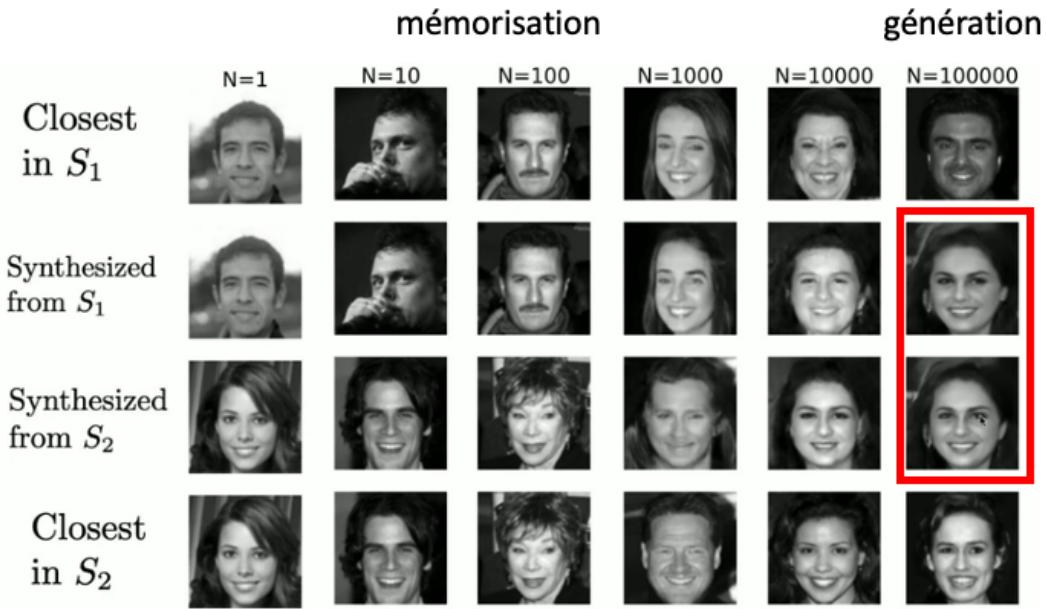


FIGURE 13 – Deux modèles génératifs S_1 et S_2 sont entraînés avec deux bases de données d’images de visages (80×80 pixels) sans recouvrement et chacune de taille N . On donne ensuite la même image de bruit aux deux modèles afin de synthétiser une nouvelle image. On peut trouver l’image de la base de 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:

$$p(x|c) \propto p(c|x)p(x) \quad (21)$$

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 séance dernière 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 chaînes 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 entraîné 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 maîtriser, 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

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

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 à savoir 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

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

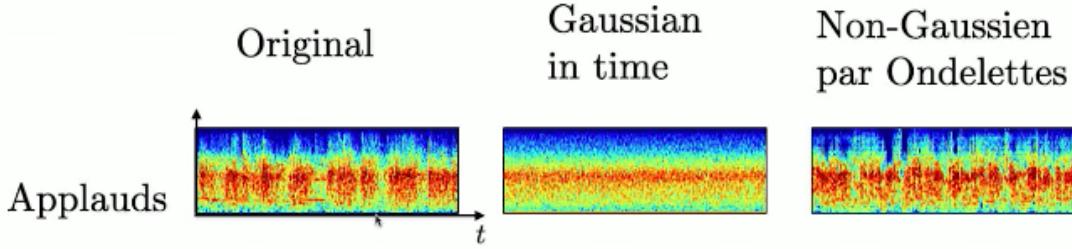


FIGURE 14 – Représentations Temps-Fréquence d’un applaudissement: le son original à gauche; au milieu la génération d’un modèle gaussien qui ne conserve que l’énergie originale à toutes les échelles/fréquences; à droite la génération par un modèle plus réaliste qui tient compte des corrélations entre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

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

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

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

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 $U_\theta(x) = \text{NN}_\theta(x)$, avec là aussi un florilège de choix d'architectures spécifiques. Comment estimer les paramètres du réseaux 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 constraint 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 typiques³² 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^* :

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

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 savons³³ que

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

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:

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

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

$$\nabla_\theta U_\theta(x) = \Phi(x) \tag{33}$$

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 $A(\Theta) = \log Z_\theta$ alors³⁴

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

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 modéliser 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 maîtrise pas bien pourquoi la plus part 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 calcul 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 tenter d'effectuer une estimation à l'étape t selon la méthode suivante

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

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 peuvent être attaqués 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 toute 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.

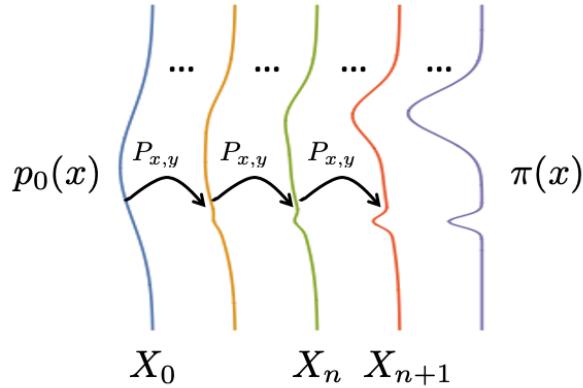


FIGURE 15 – Évolution de la distribution initiale $p_0(x)$ vers la distribution cible (ici notée $\pi(x)$) par application de la matrice de transition $P_{x,y}$ de la chaîne de Markov.

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

On veut échantillonner une distribution cible, par exemple $\pi = p_\theta$, dont on connaît 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 démarche³⁵? 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 chaîne de variables aléatoires (*v.a*) $(X_0, X_1, X_2, \dots, X_n)$ et la chaîne correspondante des *pdf* $(p_0, p_1, p_2, \dots, p_n)$. 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

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

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 physiques³⁷. Cela a pour conséquence de simplifier la relation générale de l'équation 6:

$$p(x_0, x_1, \dots, x_n) = p(x_0) \prod_{i=1}^n p(x_i | x_{i-1}) \quad (37)$$

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

A 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 chaîne de Markov

$$p_{k+1}(x) = \sum_{x_k \in \chi} p(x|x_k) p_k(x_k) \quad (38)$$

En termes matriciels, si $\mu_k = [p(X_k = x)]_{x \in \chi}$ (vecteur colonne), alors³⁸

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

On veut que lorsque quand n tend vers l'infini μ_n converge vers $\mu_\pi = [\pi(x)]_{x \in \chi}$ qui dans le cas paramétré s'écrit $\mu_\theta = [p_\theta(x)]_{x \in \chi}$, et la convergence doit s'opérer quelque 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 suivantes³⁹: 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 chaînes 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-Hastings**⁴⁰. Il nous faut définir la matrice $P_{x,y} = p(y|x)$ et l'idée est de partir d'une densité $Q(x,y)$ connue⁴¹, par exemple une gaussienne ($Q(x,y) \propto \exp(-\|x - y\|^2/(2\sigma^2))$) et de la modifier selon

$$p(y|x) = Q(x,y) \times \rho(x,y) \quad (40)$$

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 chaîne. 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 chaînes de Markov réversibles** qui nous dit que la distribution invariante π et les probabilités de transitions sont reliées par la relation:

$$\pi(x)p(y|x) = \pi(y)p(x|y) \quad (41)$$

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 donne⁴³

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

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** $i : 1, \dots, n$ **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 chaîne 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 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 requiert 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 chaîne stagne**. Donc, on a deux cas

43. NDJE. Voir Cours 2023 Sec. 8.9

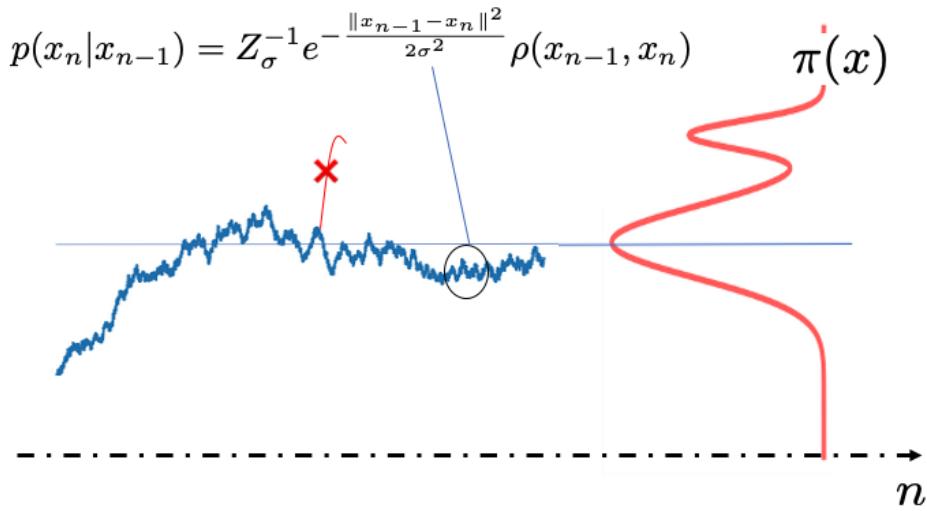


FIGURE 16 – Cas problématique d'une distribution cible $\pi = p_\theta$ multi-modale ($U_\theta(x)$ non convexe et un noyau $Q(x, y)$ gaussien avec un σ trop petit.

extrêmes où la chaîne de Markov donne soit un échantillon d'un seul mode, soit elle tombe dans une trappe où elle n'avance pas. **La chaîne 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 *replica-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 chaînes, chacune avec des valeurs de β différentes. A l'étape k , les chaînes 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 chaîne 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

$$\rho^{RepEx}(x_{k+1}^i = x_k^j, x_{k+1}^j = x_k^i) = \min \left(1, \frac{\pi_{\beta_i}(x_k^j)}{\pi_{\beta_i}(x_k^i)} \times \frac{\pi_{\beta_j}(x_k^i)}{\pi_{\beta_j}(x_k^j)} \right) \quad (43)$$

où l'on reconnaît les probabilités de transition pour la chaîne i et la chaîne j de l'équation 42. Maintenant, on choisit des paires de chaînes proches en valeur de β et les autres chaînes 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 agrave le constat précédent. Donc, quand on utilise des réseaux de neurones pour paramétriser 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!):

$$-\nabla_x \log p_\theta(x) = \nabla_x U_\theta(x) \quad (44)$$

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'entraînement:

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

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

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

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 entendu⁴⁴.

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** paramétrisé 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:

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

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 fausse image 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.

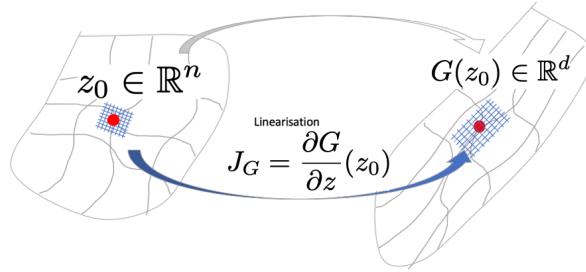


FIGURE 17 – Illustration de la fonction du Jacobien du générateur G entre l'espace des variables latentes (z) et l'espace des données (x).

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*

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

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 $V(D, G)$:

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

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

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

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

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

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

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

■

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^* ,

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

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

$$\min_G \max_D V(D, G) = -2 \log 2 \Leftrightarrow p_G = p_{data} \quad (54)$$

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^*) . A 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 2014⁴⁶ dont le point marquant fut que l'on c'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 XVIII^e 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'Algorithm 2 est celui utilisé par Goodfellow et al (dit "GAN vanilla")⁴⁷.

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

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

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

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

- C'est difficile à entraîner en raison du caractère "adversaires" des deux réseaux qui entraîne 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 entraîne 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'entraînement. C'est un cas similaire au problème de la figure 16 vu dans le cadre des chaînes 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 d'images vraies par exemple pour aider au *fact-checking*.

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 grossso 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 optimiser⁴⁸ 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 $x \in \mathbb{R} \times [0, 1]$

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

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 $\theta \rightarrow 0$.

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

$$D(p_{\theta_g} \| p_{data}) \xrightarrow{\theta_g \rightarrow \theta_g^*} 0 \quad (55)$$

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 Gaspar 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 co-créé 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:

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

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:

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

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 1942⁵⁰ 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 chaînes de Markov, entre chaque étape de la chaîne, 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

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

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>.

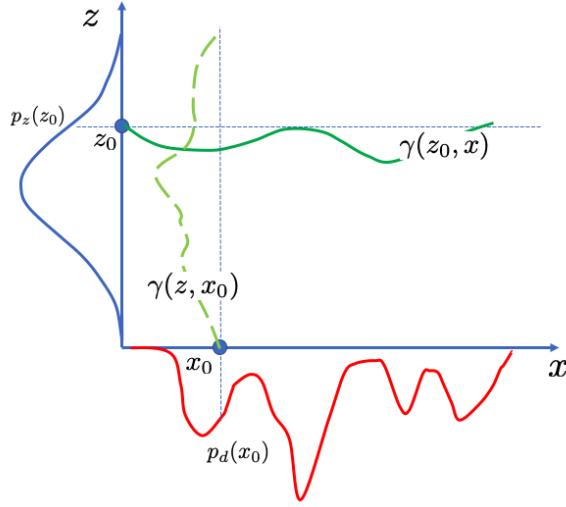


FIGURE 19 – Schématisation des contraintes du problème de L. Kantorovich.

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'écrit⁵²

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

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

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

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

51. En l'honneur de Leonid Vaserštejn (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.

Pourquoi a-t'on introduit cette distance de Wasserstein? Elle est plus faible que celle de Kullback-Leibler, c'est-à-dire que si p et q soient supportées par un compact

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

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 admettrons⁵⁵:

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 continue 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. $f : A \rightarrow B$ (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 $d_B(f(x), f(y)) \leq Kd_A(x, y)$.

Théorème 3 (*Dualité de Kantorovich-Rubinstein*)

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

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 calculer⁵⁶

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

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 oeuvre (*Wasserstein GAN*), d'autres expérimentations ont été effectuées⁵⁷. 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.

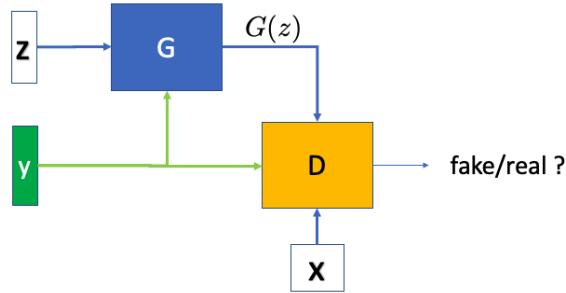


FIGURE 20 – Schématisation d'un GAN conditionnel: par rapport au schéma de la figure 6, on ajoute l'information y à l'entrée du générateur et du discriminateur.

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 lourd⁵⁸.

4.3 GANs conditionnels (cGAN)

L'idée⁵⁹ 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

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

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.

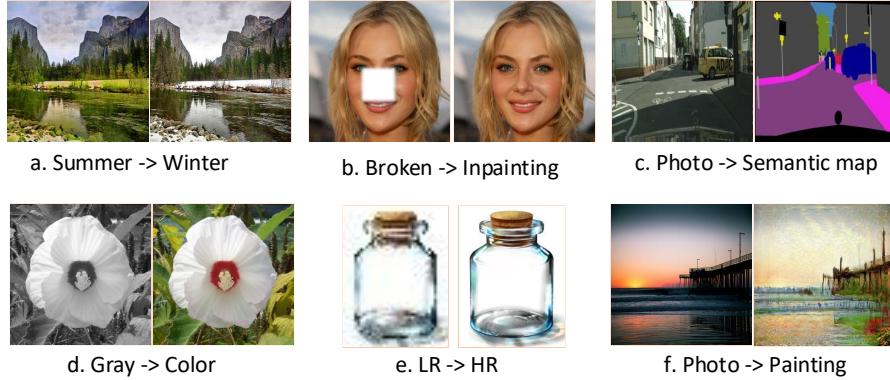


FIGURE 21 – Quelques illustrations de la transformation d’image-à-image avec un GAN conditionnel (source [arxiv:2101.08629](https://arxiv.org/abs/2101.08629)).

S. Mallat nous présente des illustrations d’images issues de tels "cGAN"⁶⁰: par exemple en entraînant 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 apparaître 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’âge) (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](https://arxiv.org/abs/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](https://arxiv.org/abs/1611.07004) associé au software **pix2pix**

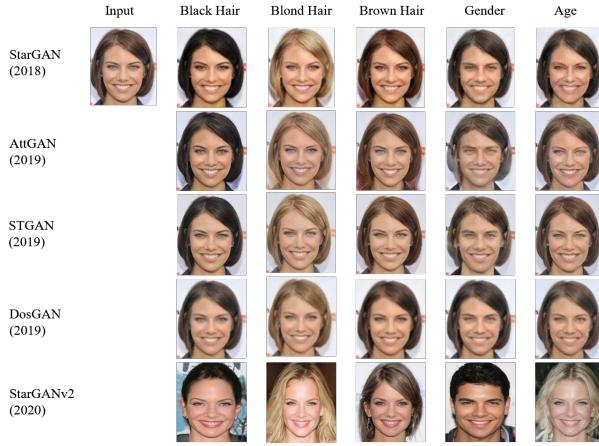


FIGURE 22 – Comparaison de différents cGANs dans la génération d’images de visages conditionnées par 5 attributs (source *arxiv:2101.08629*).

Les "expérimentateurs" se sont tournés vers des **évaluations qualitatives** mettant en jeu⁶¹:

- 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

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

- auquel il nous faut ajouter de la **diversité**, c'est-à-dire que l'on a besoin de l'inclusion inverse

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

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

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

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

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

La difficulté est de choisir⁶² 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)

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

On va utiliser les *v.a* $X = \Phi(X^{data})$ et $Y = \Phi(X^g)$. Concernant Φ , on entraîne 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 entraînement introduisent inévitablement un biais⁶³. NDJE: Concrètement, on peut dans le cadre d'analyse astro, se poser la question si des images de galaxies peuvent traitées 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. **Donc, *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 Dhariwal⁶⁴ (**Glow**) qui est un aboutissement dans ce domaine⁶⁵. 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 impose 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:

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

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](https://arxiv.org/abs/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 maîtresse 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 générée. On peut utiliser n'importe quelle fonction h définie sur A , et la contrainte de conservation s'écrit alors:

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

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

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

Maintenant quel est le lien entre dx et dz , avec $x = T(z)$? Il s'agit d'un changement de variables où apparaît le déterminant⁶⁶ de la matrice carrée Jacobienne⁶⁷ de la transformation (Fig. 17):

$$dx = dz \times |J_T(z)| \quad (73)$$

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

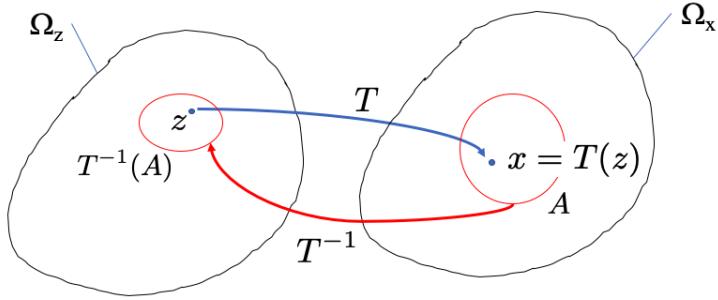


FIGURE 23 – Schématisation de la loi de conservation des probabilités (Eq. 71)

Donc, l'équation 71 devient

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

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

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

Le Jacobien est là pour tenir compte de la différence de volume entre A et $T^{-1}(A)$.

4.5.2 Optimisation de la vraisemblance

Considérons un transport paramétrisé 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

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

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

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

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 chaîne 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 transports⁶⁸. Il va y avoir de nouveau une évolution, et à un certain point, la communauté s'est rendue compte que discréteriser le transport devient trop compliqué, et qu'il vaudrait mieux tenter la voie des *transports continu*s.

^{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_bijection_1D_simple.ipynb` et `JAX_FLOWS_MAF_NVP_simple.ipynb` qui mettent en œuvre simplement des NFs.