Notes and Comments on S. Mallat's Lectures at Collège de France (2025)

Learning and generation using random sampling

J.E Campagne *

Janv. 2025; rév. 25 mars 2025

*If you have any comments or suggestions, please send them to `jeaneric DOT campagne AT gmail DOT com`

Table des matières

1. Foreword

Disclaimer: *Below, you will find my freely styled notes taken along the way and reformatted with some comments ("NbJE" or dedicated sections). It is clear that errors may have slipped in, and I apologize in advance. You can use the email address provided on the cover page to report them to me. I wish you a pleasant reading.*

Please note that on the Collège de France website, you will find all the course videos, seminars, and lecture notes not only from this year but also from previous years [1].

I would like to thank the entire team at the Collège de France who produce and edit the videos, without which the publication of these notes would be less convenient.

Also note that S. Mallat [2] provides open access to chapters of his book *"A Wavelet Tour of Signal Processing"*, 3rd edition, as well as other materials on his ENS website.

This year, 2025, marks the eighth year of the Data Science Chair cycle of S. Mallat, with the theme: **Data Generation in AI through Transport and Denoising**.

Finally, in the GitHub repository [3], I have uploaded numerical application notebooks illustrating the course since 2022. As much as possible, the notebooks can be executed on Google Colab.

2. Lecture of Jan. 15th

During this session, Stéphane Mallat will outline the course program for the year, which focuses on generative models. These models are at the heart of many applications that the public has learned to use to generate text, images, videos, etc., for better or worse. The course's perspective remains the same as that adopted from the beginning of this lecture series: the *why*, rather than the *how*. That is, the course is not dedicated to the practical implementation of specific neural network architectures—there are other resources for that—but rather aims to explore the mathematical foundations of these

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

architectures and, in some respects, to examine the limits of our understanding of the increasingly astonishing performance of these models.

2.1 Some Key Events of the Past Year

To begin, S. Mallat provides an overview of what stood out to him this year in the field of interest to us. He is highly impressed by the speed and accumulation of research in the domain, driven by increasing funding and the attraction of high-quality researchers. The field is advancing rapidly.

First, it is remarkable that the 2024 Nobel Prizes in Chemistry and Physics have recognized achievements related to neural networks. The Chemistry Prize was awarded to Demis Hassabis and John Jumper for their success with DeepMind's `AlphaFold` model [4], which has predicted the three-dimensional structure of almost all known proteins. This was previously considered an impossible problem based solely on sequence databases—especially with such precision. Naturally, this has a significant impact on Chemistry, Pharmacology, and Medicine, helping to explain diseases caused by abnormal protein folding, among other things.

The Physics Prize, which surprised many in the physics community, was awarded to John J. Hopfield and Geoffrey E. Hinton for their pioneering work in the 1980s, which laid the groundwork for the major developments in artificial machine learning that followed. The two contributions highlighted by the Nobel committee are *Hopfield networks and Boltzmann Machines*. S. Mallat points out that G. Hinton has profoundly influenced the field of learning, particularly with the fundamental back-propagation algorithm [5], as well as with all deep architectures. As for J. Hopfield, his impact has been primarily in bridging the gap between neural learning and Statistical Physics [6]. This work has been fundamental and has already been discussed in previous courses [7], and we will also see it in this course, where Statistical Physics serves as a backdrop.

4. Jumper et al., 2021, https://www.nature.com/articles/s41586-021-03819-2
5. See 2019 Course, Sec. 8
6. NbJE: If you want to learn more, you can check a short note I posted on the GitHub repository of these courses https://github.com/jecampagne/cours_mallat_cdf/tree/main/Cours2025
7. See, for example, the 2023 course dedicated to Entropy

Next, regarding large language models (LLMs), which are now three years old (already!), a recent development that stands out concerns *proof systems* in Mathematics. S. Mallat explains that a key recent advancement is the convergence of two approaches: the first, by *induction*, is the classical method used in Machine Learning (ML), based on data, as in the case of LLMs; the second, by *deduction*, is the approach used by formal proof verification systems such as `Lean` [8] or `Coq` [9]. In these "hybrid" systems, proof generation is initially performed by LLMs, followed by reinforcement learning, which is validated by the proof system. This dual approach is implemented, for example, in DeepMind's `AlphaProof` and `AlphaGeometry` models [10]. These models have already won the silver medal at the International Mathematical Olympiad, and there is little doubt that they will soon claim the gold—if they haven't already at the time of writing. Beyond these challenges, S. Mallat notes that proof strategies have significantly improved, and the field remains very active.

Another rapidly evolving field is *robotics*, which, according to S. Mallat, represents the next revolution after LLMs. The reason for this is that as soon as we begin to acquire data from all communicating sensors, we will be exploring a vast domain that extends far beyond the current Internet. As a result, interpretation systems will need to aggregate highly heterogeneous data.

A major challenge in robotics for performing tasks that seem simple to us—such as grasping objects or walking on chaotic terrain—is that a vast amount of real-time information is required, and in a way, labeled data is needed. However, we cannot simply send a robot into an unknown environment because the amount of data required for learning is currently far too large. Instead, we rely on *simulations* in which these neural systems are immersed. Consequently, everything hinges on the ability to simulate increasingly realistic complex environments.

The final domain that has been significant this year is closely related to the previous one, as it concerns the use of AI simulations in Physics. Notably, meteorology and climatology have seen the emergence of models such as DeepMind's `GraphCast` and `GenCast` [11].

8. https://lean-lang.org/
9. https://coq.inria.fr/
10. See article https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/
11. See article https://deepmind.google/discover/blog/graphcast-ai-model-for-faster-and-more-accurate-global-weather-fo and models https://github.com/google-deepmind/graphcast

`GraphCast` provides 10-day forecasts and is highly competitive compared to traditional models that solve differential equations. `GenCast`, on the other hand, is a probabilistic model based on *diffusion* processes, which we will study this year, and generates predictions over a 15-day period. There are also other approaches that combine physical constraints with ML, such as Google's `NeuralGCM` [12]. This model produces forecasts ranging from 2 to 15 days and reconstructs temperature trends over the past 40 years more accurately than traditional atmospheric models. Of course, other models exist, but it is noticeable that all these systems are developed by private companies with significant capital, allowing them to access substantial computational resources.

NbJE: As a side note, we can observe the widespread adoption of JAX [13] as a development library. You may notice that several of the notebooks I have provided are written in this language [14].

2.2 Overview of AI-Generated Data

Generative AI is part of the field of statistical learning, which fundamentally involves constructing models from data. In previous years, we have seen that this field is generally divided into:

— **Supervised problems**, where we have data $x \in \mathbb{R}^d$ (with d being very large in practice, such as images, sounds, fields, etc.), and the goal is to answer a question y (either discrete or continuous). The strategy consists of using a training dataset composed of pairs $\{x_i, y_i\}_{i \leq n}$. The primary challenge lies in constructing this labeled dataset, which must be sufficiently populated and free of bias.

— **Unsupervised problems**, where the objective is to build a model of the data x in the form of a probability distribution (*pdf*) $p(x)$ that indicates where the data is concentrated within the large space $\mathbb{R}^d$. The strategy here is also to use a training dataset, but an unlabeled one, consisting of samples $\{x_i\}_{i \leq n}$ that are assumed to be **independent and identically distributed** (*iid*) (a fundamental assumption). Within this category, we identify the following subcategory:

12. See article https://research.google/blog/fast-accurate-climate-modeling-with-neuralgcm/
13. https://jax.readthedocs.io/en/latest/
14. For more information, I have made dedicated notebooks available https://github.com/jecampagne/JaxTutos, including this one to get started: `JAX_Cophy_tuto.ipynb`

— **Self-supervised problems**, which include LLMs, where labels y_i are defined from the dataset samples $\{x_i\}_{i\leq n}$. Typically, for a language model, given a sentence in which all words are known, one word is hidden to be predicted by the model; then, within a paragraph, a sentence is hidden, and so on. This masking approach can naturally be extended to images and generalized to other types of data.

In this course, we will primarily focus on unsupervised and self-supervised problems. Why is the problem of discovering the underlying pdf $p(x)$ from data so challenging? And why has AI revolutionized this problem?

2.3 Revisiting the Curse of Dimensionality

This topic has been addressed multiple times in previous courses, starting from the first one in 2017-18. However, it is always useful to understand why determining the probability density function (pdf) $p(x)$ is an inherently difficult problem *a priori*[15]. What we seek to determine is the localization of data that aggregates on a surface of dimension d' embedded in the space of dimension d (Fig. 1).

The problem is relatively easy if the dimension d' is very small, even if d is large. For example, consider a robotic system with joints where d can potentially be very large depending on the digitization of the explored space. However, the underlying number of degrees of freedom that defines d' is on the order of twenty, making the problem more manageable. The challenges we will face, however, are quite different: consider, for instance, an image of a house interior—just the texture of materials introduces complexity that significantly increases the number of degrees of freedom, making d' quite large, though still smaller than d. Typically, S. Mallat suggests that d' is on the order of $O(10^{3-4})$ for an image where $d = 10^6$. Hence, **the manifold on which data is concentrated is, in practice, high-dimensional**. However, even if a large d' makes the problem challenging, $p(x)$ may exhibit regularities. This aspect will be crucial in constructing models to achieve our goal.

Consider, for example, the case of Gaussian white noise (*gwn*), which represents the probability of d independent variables, each following a Gaussian distribution. For

15. NbJE. *A posteriori*, one might wonder whether there are underlying regularities that allow neural networks to approximate $p(x)$. This is, for example, discussed in the 2021 course.

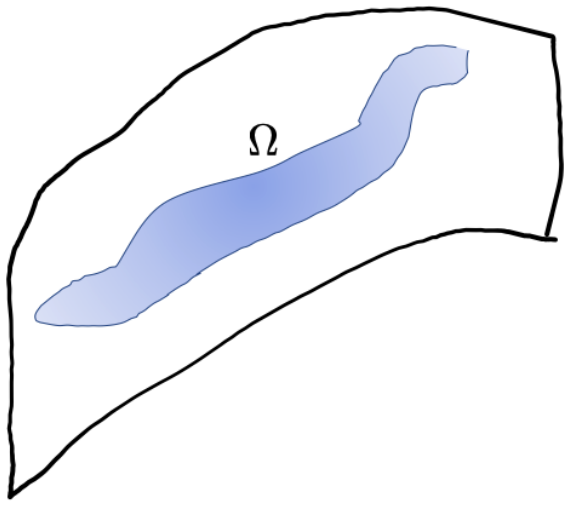


FIGURE 1 – Schematic representation of data localization on a surface of dimension d' embedded in a space of dimension d.

instance, [16] let $x_i \sim \mathcal{N}(0, \sigma^2/d)$. Then,

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

where $z = \sum_{i=1}^{d} x_i^2$ is a random variable. If the x_i are *iid* following $\mathcal{N}(0,1)$, then $p(z)$ follows a chi-squared distribution with d degrees of freedom [17], whose expectation is $\mathbb{E}[z] = d$. By applying a change of variables, we obtain:

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

Similarly, the variance of a chi-squared distribution with d degrees of freedom is $2d$, leading to:

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

Thus, the samples of Gaussian white noise localize within a spherical shell whose thickness

16. NbJE. This follows an argument developed in the 2024 course, where variance is uniformly distributed across the d variables.

17. NbJE. The $\chi^2(d)$ distribution is given by $(1/2)^{d/2}/\Gamma(d/2) z^{d/2-1} e^{-z/2}$.

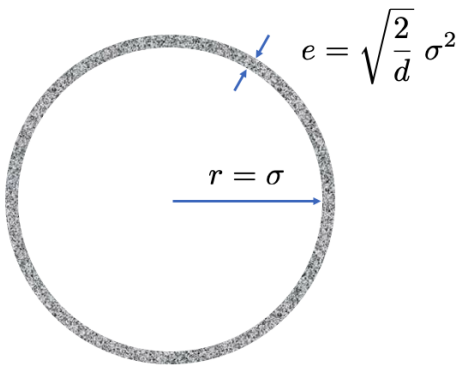


FIGURE 2 – Localization of $\|x\|^2$ when all components are iid $x_i \sim \mathcal{N}(0, \sigma^2/d)$ in dimension d.

tends to zero as d increases. In high dimensions, this localization can be viewed as a spherical manifold (Fig. 2) with a dimension equal to the space dimension minus one, which remains large, but with variance σ^2 as the only degree of freedom.

Thus, the strategy will be to seek parameterizations with a small number of degrees of freedom that can nonetheless describe high-dimensional surfaces, making them suitable for practical cases. Of course, a *gwn* is not very structured, so to speak, but one could consider parameterizing covariance matrices and using more sophisticated models, as we will see. The core challenge will be selecting an appropriate parameterization and learning it from the samples. What is the difficulty in this context?

Let us assume that to choose our parameterization, we adopt a rather simple hypothesis based on Lipschitz-type regularity, meaning that $p(x)$ is differentiable almost everywhere. That is, for all x, y in the domain of definition, there exists a constant K such that

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

Is this hypothesis sufficient to estimate $p(x)$? If so, the pdf is regular, and if $x \in [0,1]^d$, it would be enough to sample the space with a sufficiently fine interval ε to perform some form of interpolation and determine $p(x)$ throughout the space. However, to ensure that any x is at a distance smaller than ε from some x_i, it is easy to see that we need at least n samples such that (note: we use dimension d as that of the manifold/surface of x to

simplify notation)

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

Notice that if we construct a histogram by partitioning the space into small boxes of size ε^d, we also need multiple samples per box to obtain a meaningful average. For instance, with $\varepsilon = 1/10$ and d on the order of a few thousand, the required number of samples vastly exceeds the number of atoms in the Universe. This is the **curse of dimensionality**.

This suggests that we must adopt much stronger assumptions than Lipschitz regularity—significantly stronger regularities for $p(x)$. However, until "recently," no one believed it was possible to construct probability distributions of faces, bedrooms, proteins, etc. According to S. Mallat, before neural networks, probability modeling was primarily based on Gaussian models or slightly more sophisticated variants, but these remained confined to relatively simple classes of processes (see Course 2023). **This entire field has been completely transformed by generative models based on neural networks.**

S. Mallat then provides an introductory overview of generative models.

2.4 Autoregressive Models in LLMs

A key modeling strategy in LLMs is the use of autoregressive models. The data consists of *sequences*, meaning there is a total order, and each data point is *quantized* (*token*), meaning that it takes a value from a finite set. If the value of a *token* is represented by x_i, we can then consider the ordered sequence $\{x_1, x_2, \ldots, x_{t-1}\}$ and attempt to predict x_t, where t can represent a discrete time step. If we have access to the conditional pdf $p(x_t|x_1, x_2, \ldots, x_{t-1})$, we can then generate samples of x_t given the past *tokens*. Underlying this process, we use the following decomposition, which expresses increasingly complex conditional probabilities:

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

It is worth noting that the Markov assumption simplifies the problem (Course 2023) by assuming that $p(x_t|x_1, \ldots, x_{t-1}) = p(x_t|x_{t-1})$, meaning that if $t-1$ represents the present, knowing it is sufficient to predict the future. However, in general, we need to model all

conditional probabilities $p(x_t|x_1, \ldots, x_{t-1})$. This is where **Transformers**, introduced in 2017 [18], have demonstrated their ability to model long-range dependencies.

Consider, for instance, how the presence or absence of negation at the beginning of a sentence can drastically change its meaning: *"Failing to account for large-scale dependencies is fundamental"*. And let's not even mention commas, which have caused sleepless nights for diplomats drafting various treaties. The issue of long-range dependencies could have been intractable due to the curse of dimensionality, but it is precisely the clever use of the *attention* mechanism that makes it solvable.

This year, however, we will not be studying this type of problem, as the data we will be working with is *multi-dimensional*.

2.5 Multi-Dimensional Data

The LLM framework presented above is not well suited for multi-dimensional data, which is commonly encountered in physics problems, image analysis, and other domains. Let us consider an image to illustrate this point: there is no natural way to organize pixels sequentially. Take, for example, the two indexing methods shown in Figure 3. If we want to account for interactions between a pixel i and its 8 closest neighbors on a 2D grid, a clear issue arises: some neighboring pixels will be indexed sequentially as $i-1$ and $i+1$, suggesting a direct causal link, while others will have indices that are highly delocalized in the sequence. This artificially introduces large-scale causal links. Ultimately, when modeling probabilities, we impose dependencies based on the chosen one-dimensional traversal scheme, without leveraging the natural topology of the 2D grid. However, this topology is crucial. That being said, some existing methods do exploit one-dimensional representations, but they do not achieve the best results.

Our approach will be to make full use of the multi-dimensional information inherent in the problem.

18. Ashish Vaswani et al. 2017, *Attention Is All You Need*, Advances in Neural Information Processing Systems, pages 5998–6008. https://arxiv.org/abs/1706.03762

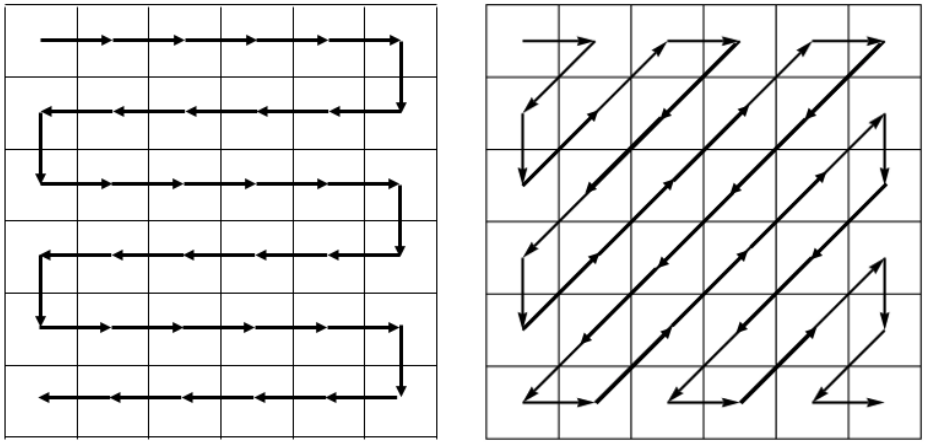


FIGURE 3 – Two ways to index the pixels of an image.

2.6 Probabilistic Models with Gibbs Energy

This type of modeling is based on the framework developed by R. Fisher in the early 1920s, which we discussed in the 2022 to 2024 courses. These are models at the core of Statistical Physics, where the idea of learning $p(x)$ is to define a sufficiently "large" parameterized family of pdfs $\{p_\theta(x)\}_\theta$ to be able to find a value θ^* for the parameter that best fits $p(x)$. The choice of the family is, of course, crucial. Assuming $p(x)$ is non-zero, the logarithm is then defined, and the exponential form is a classic choice:

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

The modeling task is then to find **the parametrization of the energy** $U_\theta(x)$ (by analogy to Physics) that is suited to the problem. The most probable events x are those that minimize the energy (i.e., U_θ). This is where *a priori* knowledge (e.g., symmetries, invariants, etc.) or general information about the problem comes into play to introduce structure into the parametrization $U_\theta(x)$. However, in ML, which deals with problems like face generation, we do not really have *a priori* information, unlike Physics, which is guided by Principles. Therefore, in ML, we start with a handicap.

That said, once a parametrization is chosen, we aim to **minimize the Kullback-**

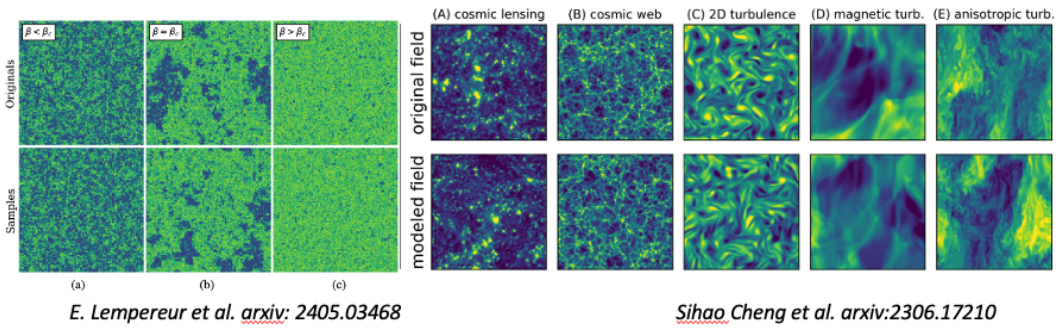


FIGURE 4 – Example of generated images of Physical fields.

Leibler divergence [19] between $p_\theta(x)$ and $p(x)$. This minimization is equivalent in Fisher's framework to using **Maximum Likelihood** to find θ^*. The x values are well localized where the probability is highest, and the typical set where x is located is a concept introduced by Cl. Shannon in the 1940s (see 2022 Course). This phase is an **optimization problem**.

In the Gaussian case, $U_\theta(x)$ is convex, and there is no issue. However, it **becomes very complicated when** $U_\theta(x)$ **is not convex**. A tricky point, for example, in a model with two minima of the same probability, is that the generation of samples x must account for the fact that x can be in either of the energy minima, as they correspond to different configurations. Certainly, if we only generated x corresponding to one minimum, we would obtain good samples, but we would fail to capture the diversity. Think, for instance, of face generation, where we would only produce faces of one gender. This problem of accounting for diversity, and thus **generalization, is the key issue encountered in this field**.

In the next session, we will summarize these techniques based on Markov chains and Monte Carlo methods. These techniques allow for modeling fields such as (Fig. 4): fields in cosmology, turbulence, and ϕ^4 (or Ising). In this case, x represents the equilibrium state of the system, and the probability is given by the exponential form mentioned earlier, which is the Gibbs distribution. These problems are "relatively classical" and well understood, as they belong to the category of **ergodic fields** (e.g., 2023 Course).

However, this ergodicity assumption is not suitable for cases like face image generation, interior views of houses, etc. (Reminder: ergodicity implies invariance under

19. See e.g., 2022 Course Sec. 6.3, 2024 Course Sec. 5.2

FIGURE 5 – Example of generating 2x6 images of faces: on the left, using a Normalizing Flows model (source: `Glow` Diederik P. Kingma et al. *arxiv:1807.03039*), on the right, using a Generative Adversarial Networks model (source: Xin Wang et al. *arxiv:2202.07145*).

translation). In order to implement generative models capable of producing images like those shown in Figure 5, we will need to opt for something different.

2.7 Generative Adversarial Networks (GAN)

GANs were introduced by Goodfellow et al. [20] in 2014. We mentioned them during the 2019 Course Sec. 2.8., and the original schematic is shown in Figure 6. Two networks act in opposition: the generative network (G), which, for example, takes Gaussian white noise ($z \sim \pi(z) = \mathcal{N}(\mathbf{0}, \mathbf{1})$) and generates new images x, and the discriminator (D), which provides the probability that the image presented to it comes from the training dataset (*real image*) rather than from a generation $G(z)$ (*fake image*). To optimize both networks using the training images (from the pdf p_{data}), the idea is to solve a `minmax` problem, as follows:

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

Despite the implementation challenges that we will mention below, the evolution of GANs has led to the development of generators in many domains: in Physics, medicine, meteorology, etc. For example, a technique used not white noise images but images as conditioning. S. Mallat shows us an example on screen of brain images from a "CT" scan and

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

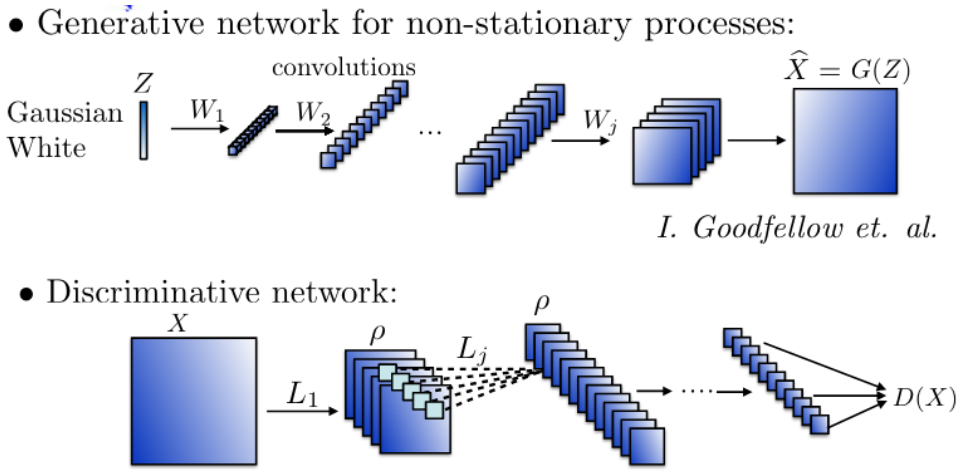


FIGURE 6 – Schematic of a GAN by Goodfellow et al. with two adversarial networks: a generator and a discriminator (classifier).

the prediction from a GAN. Similarly, he presents an example in meteorology using radar precipitation images [21] which, in 2021, were able to make predictions up to 90 minutes in advance for typical areas of 1.3 km radius.

The tricky part is that minimization is not easy because it involves a saddle point. Much research has been dedicated to developing improvements. What happens? In fact, in a GAN, there is nothing that forces the generator to explore the entire space of possible x. This is a sampling flaw known as "**mode collapse**", which is related to the failure to account for all the minima mentioned earlier. We get samples via $G(z)$ that are perfectly fine, but the set of generated x may represent only part of the possible modes (lack of diversity, **generalization problem**). To overcome these issues, various regularization techniques are employed [22] but ultimately, S. Mallat tells us that the optimization problem is still poorly understood. **The main danger is biasing the generations and inducing forms of memorization of the training dataset**. As long as one is satisfied with "pretty images," there is no issue, but as soon as one wants to use the samples for applications (e.g., medical, meteorological), this becomes very problematic because artifacts occur whose origin is the biases of the training dataset, which are reproduced by the generator.

21. NbJE. Likely from https://www.nature.com/articles/s41586-021-03854-z.
22. See, for example, the model `light-weight-GAN` by Liu B., Zhu Y., Song K., Elgammal A., 2021, in International Conference on Learning Representations. https://openreview.net/forum?id=1Fqg133qRaI. See Sec. 2 of the paper.

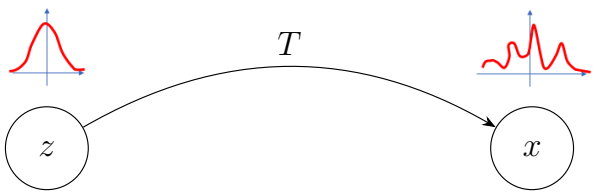


FIGURE 7 – Transport of the pdf of latent variables z to the pdf of the data x.

2.8 Transport Modeling

In fact, what we sought to do in the previous section was to transform a distribution $\pi(z)$ of latent variables z having, for example, a white noise pdf, into a distribution of the data x ($p(x)$). This is referred to as **transport** T such that $p = T_{\#}\pi$ (Fig. 7). Therefore, we can ask the more general question of estimating the transports: which class of transports should we seek? And how do we guide our choice?

When we talk about probability transport, we are referring to the search for *optimal transports* introduced by G. Monge in 1781 and later explored by Léonid Kantorovich in the 1940s through Measure Theory. The point is that **the transport T is not unique** in general; one can swap any pair (z_i, x_j) while preserving the initial and final PDFs. Therefore, we need a principle to guide our choice. If the domain of x and z is the same (e.g., an image of the same dimensions), we can use the cost function $d(x, T(x)) \propto \|x - T(x))\|$ or $\|x - T(x))\|^2$ which must then be minimized.

That said, S. Mallat points out that models based on this type of transport are not currently efficient for high-dimensional problems. Other algorithms have been employed.

2.8.1 Normalizing Flows (NF)

A transport such as a Normalizing Flow (or Flow) T is defined as a diffeomorphism (i.e., a bijection) between the data space and a latent space, such that:

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

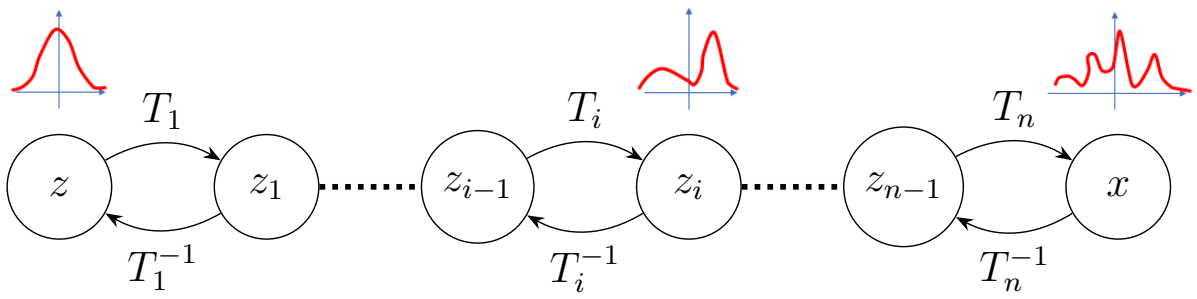


FIGURE 8 – Schematic Normalizing Flow process. On the top the *forward* or *generative* direction from a simple $\boldsymbol{z}$ distribution to a more complex one for $\boldsymbol{x}$. On the bottom the *backward* or *training* direction from complex to simple distributions.

If the distribution of $\boldsymbol{z}$ is $\pi(\boldsymbol{z})$ (e.g., $\mathcal{N}(\mathbf{0}, \mathbf{1})$), then the following relation applies:

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

where J_T and $J_{T^{-1}}$ are the Jacobians of the transformations T and T^{-1}, respectively. Unlike GANs, the dimensions of the latent and data spaces are the same by definition in models based on flows.

The flow T is typically constructed as a composition of several individual flows $\{T_i\}_{i<n}$ such that:

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

Figure 8 illustrates a schematic view of the *forward* or *generative* direction and the *backward/reverse* or *training* direction. If we denote $\boldsymbol{z}_i = T_i(\boldsymbol{z}_{i-1})$ for all $i < n$ (using $\boldsymbol{z}_0 = \boldsymbol{z}$ and $\boldsymbol{z}_n = \boldsymbol{x}$), then:

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

Each bijector T_i is parameterized, and we seek the structure that allows optimization, where the PDF $p(x)$ maximizes the likelihood of the data in the dataset. This is an interesting approach from a mathematical perspective that links optimal transport and the techniques in Section 2.6. A model based on this principle of flows is `Glow`, for which

examples of face generation are presented in Figure 5. However, S. Mallat tells us that it is difficult to structure these bijectors, and thus a lot of *a priori* information needs to be injected, and these models remain limited for very high-dimensional problems.

2.8.2 Transport by Score-Diffusion

This is a topic discussed during the last session of the 2024 Course. Referring to equation 11, one wonders how to decompose the operator T? In fact, instead of taking a discrete decomposition, we introduce a continuous "time" variable, and T can then be decomposed into a large number of "simpler" elementary transformations, where information is injected. We thus find ourselves in the domain of **ordinary differential equations** (ODE), where x_t is modified according to an equation of the form

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

The PDF $p_t(x_t)$ is then given by the Joseph Liouville equation, which he developed in the context of Hamiltonian Physics.

S. Mallat tells us that this type of transport is still somewhat too restrictive, especially because for there to be a solution to the equation above, the velocity v_t must be Lipschitz, which is a constraint that is difficult to obtain numerically. We then turn to **stochastic differential equations** (SDE), where the transport is locally more irregular. In this case, x_t is the solution to

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

where dW_t represents a Wiener process (Brownian motion) and T_{max} represents the maximum duration of the transport, assumed to be sufficiently large. At each step (t_i, t_{i+1}), the transport is no longer deterministic as in Normalizing Flows but probabilistic [23] (Fig. 9). The PDF $p_t(x_t)$ is governed as in ODEs, by a differential equation, which is a **Fokker-Planck** equation obtained in the years 1914-17 during the study of Brownian motion.

23. NbJE. You can modify the notebook `Ornstein_Uhlenbeck.ipynb` available on Github https://github.com/jecampagne/cours_mallat_cdf in the "cours2024" directory.

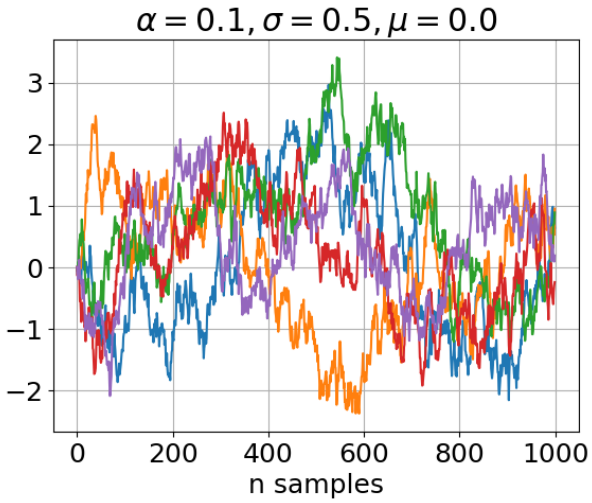


FIGURE 9 – Examples of transport realizations according to the stochastic differential equation $dX = -\alpha X\, dt + \sigma\, dW$ with $dW = \mathcal{N}(0, dt)$ ($\alpha = 0.1, \sigma = 0.2$).

In fact, this perspective of a process governed by a differential equation will give us a very classical probability sampling technique. Let us note that if we want to go from a PDF p_0 to $p(x)$, there must be a convergence occurring, and in a certain way $p(x)$ **must be a fixed point of the equation**. We will see that if we parameterize $p(x)$ according to a Gibbs distribution (Eq. 7), then the stationarity condition tells us that the velocity $v_t(x_t)$ (Eq. 13) must satisfy:

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

where $\nabla_x \log p(x)$ is called the **score**[24], which leads us to **Score Diffusion** algorithms. All of this provides us with a knowledge base from Statistical Physics.

To go further, we will notice that the goal is as follows. We need to discover a transport T that, starting from white noise, gives us the underlying data distribution. But looking at the diagram 8, it seems that it would be simpler to identify T as $(T^{-1})^{-1}$. In fact, it is easier to go from $p(x)$ to a distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$, thus defining T^{-1}. This is what motivated the work of Yang Song et al. (2021, *arXiv:2011.13456*).

Thus, starting from a sample from the database, which is a realization of $p(x)$ by

24. NbJE. See Course 2024 Sec. 5.2 for the remark I make on this naming, as it should be noted that here we are considering the gradient with respect to x and not with respect to any potential parameters θ_t.

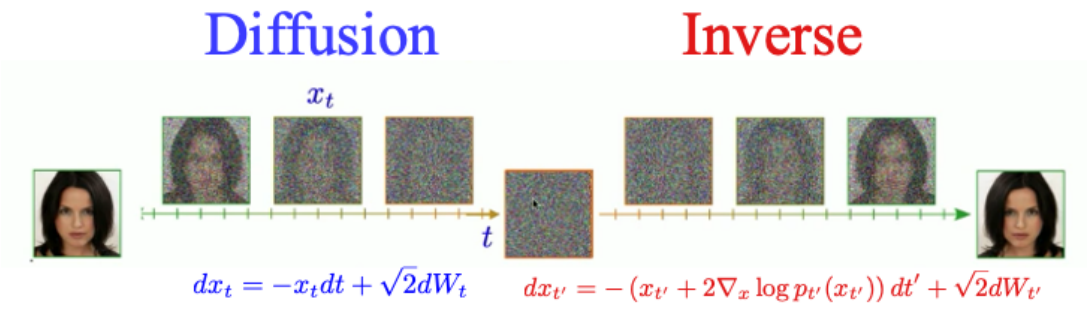


FIGURE 10 – Diffusion and Inverse Process to determine the transport between white noise and the data PDF.

assumption, to obtain a realization of pure Gaussian white noise, it is enough to gradually add noise (the "Diffusion" transport in Figure 10) (there is a renormalization at each step so that the original image disappears). This is the realization of a stochastic process governed by the **Ornstein-Uhlenbeck diffusion equation** [25] which is the simplest and can be written as

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

Once we have the path T^{-1}, we need to invert it. This is entirely possible using a **damped Langevin equation** [26] which is written as

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

where the score of the PDF at step t' appears.

If the diagram is mathematically clear, the remaining point to resolve is the ability to obtain the score at all steps of the "Inverse" transport (Fig. 10). This is *a priori* a difficult problem because we need to learn d functions (for the gradient calculation) in high-dimensional space. This is where there was a big surprise: deep neural networks allow us to learn these scores. What is the underlying idea?

Ultimately, we need to delve into a related problem that deals with the **denoising**

25. Leonard Salomon Ornstein (1880-1941) and George Eugene Uhlenbeck (1900-88), both physicists.
26. Paul Langevin (1872-1946)

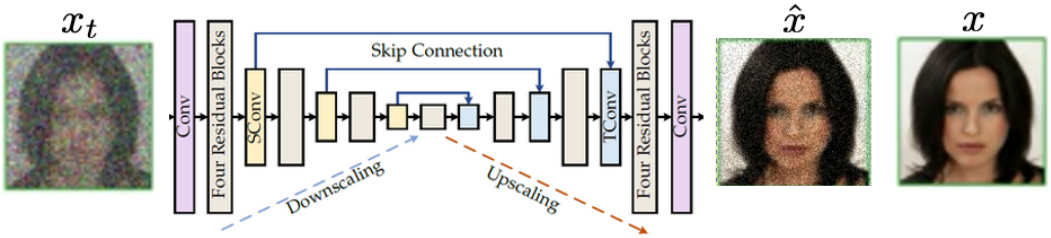


FIGURE 11 – Using a deep network for denoising a task. See for example Kai Zhang et al. (2021) *arXiv:2008.13751* (Sec. 3.1). Left: the image x_t to be denoised; right: $\hat{x}$, the result from the neural network, and x, the image such that the network must minimize $\|x - \hat{x}\|^2$ for all samples in the database and for the different values of the noise variance in the noisy images.

of a signal x_t such that

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

because, in fact, the Inverse transport (Fig. 10) performs denoising to recover a sample x. Now, denoising means finding an estimator $\hat{x}$ for x. There is a standard result from the 1950s-60s which states that searching for $\hat{x}$ such that it solves the minimization problem

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

is equivalent to performing the following gradient ascent (in one step)

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

Therefore, if the minimization problem is equivalent to finding $\hat{x}$, let us use a deep neural network (Fig. 11), for example, of the `U-Net` type [27], train it by minimizing the quadratic loss (supervised learning), and we will have $\hat{x}$ for x_t, and thus we will have access to $\nabla_x \log p_t(x_t)$ by inverting the above equation.

Now, the (fundamental) question that arises is whether the neural network is able to obtain the optimal denoiser. In fact, it seems to succeed, and even very well, and this technique is implemented in public applications such as `DALL-E`, `Midjourney`, `StableDiffusion`, etc (Fig. 12). As can be judged, the quality of the images is impressive.

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

FIGURE 12 – Example of image generation using the `ChatGPT` interface, which uses `DALL-E`.

However, S. Mallat draws our attention to the fact that there is always an underlying fundamental question: **are the results really samples from** $p(x)$, or through a clever assembly, are they mixtures of the original images? In the latter case, the produced images would depend entirely on the training database, leading to a generalization problem. The goal of this course is to understand the mathematical concepts and properties of these generative algorithms, and this **question of generalization is pervasive in machine learning**.

In the case of generative models, we obtain an estimator of $p(x)$ based on the training set, but it is considered a poor estimator if "unfortunately" it varies when the dataset is changed, even if it remains composed of bedroom images or faces, depending on the case. In this problem, if the estimator is good, its variance should tend to zero as the size of the dataset grows infinitely. This point was tested in the article [28] by Kadkhodaie et al. (2024), which S. Mallat mentioned in the 2024 Course (Sec. 9.4). What surprised them is that as the training dataset size increases, **towards** $N = O(10^5)$ **(for** 80×80 **images),**

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

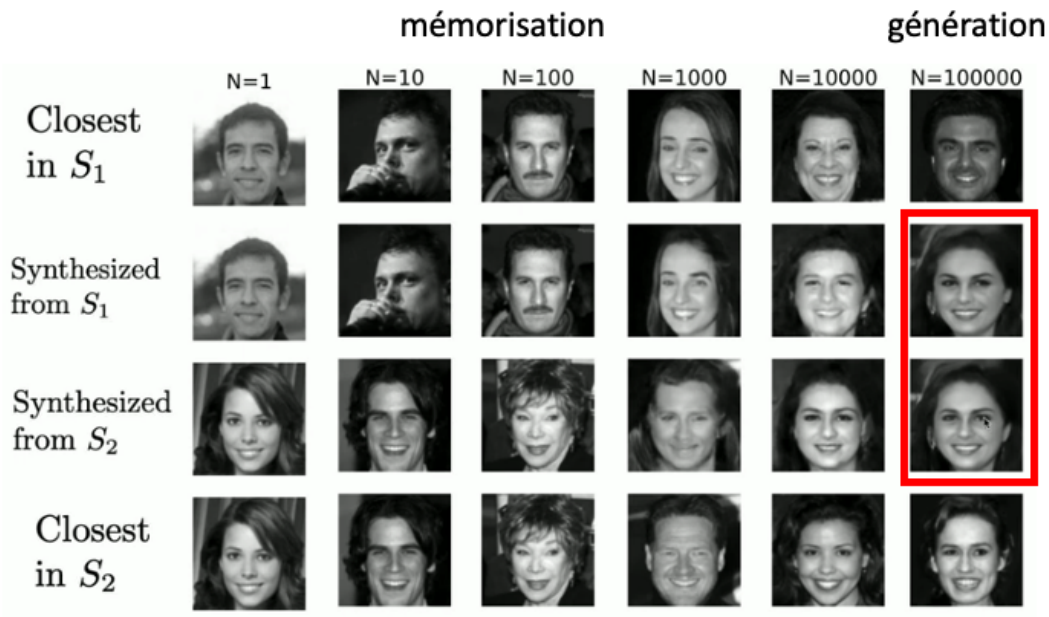


FIGURE 13 – Two generative models S_1 and S_2 are trained with two subsets of the same face image datasets (80×80 pixels) with no overlap, each of size N. Then the same noise image is given to both models to synthesize a new image. The image from the database that best matches the synthesized image is found. As long as $N < O(10,000)$, S_1 and S_2 produce different images that resemble more and more a database image as N becomes smaller. Once $N = 100,000$, the synthesized images from S_1 and S_2 are (almost) identical and different from the closest images in the two databases. A transition is observed towards a model that generalizes well: it has learned the probability distribution $p(x)$.

a transition occurs between a "memorization" type behavior and a "generalization" behavior, which suggests that the model has learned $p(x)$ (Fig. 13). However, S. Mallat notes that it must be realized that for small images, much data was required to achieve generalization!

Now, the previous result shows that we can achieve a **small variance** of the estimator, but we must also ensure that the estimator is **unbiased**. That is, we need to make sure that we have indeed converged to $p(x)$, which raises the question of obtaining the "correct score," or equivalently, the problem of **denoising optimality**. This leads us into a classical chapter of mathematics, namely **Harmonic Analysis**. NbJE. This topic was covered in the 2021 Course. In the case of denoising, which was studied from the 1950s to 2000s, we can

cite:

— Linear estimation: Wiener filter;

— Non-linear sparse estimation [29]: orthonormal bases;

— Wavelet bases, curvelets, bandelets: optimality?

By the way, note the diversity of mathematical subjects that aggregate around generative models: optimal transport, stochastic differential equations, optimization, and of course Harmonic Analysis. Why? In general, when we have a signal to denoise, the first idea that comes to mind is to perform a separation between low frequencies (mainly composed of the signal) and high frequencies (mainly composed of noise). This technique is the one used by Wiener (linear filtering), which is implemented in Fourier. In the 1980s, it was realized that better results could be achieved using non-linear operators, with the core problem being the representation of the data and the development of sparse bases (where the energy of the function to be processed is concentrated in a small number of coefficients). In fact, **neural networks do much better, in a way, they adapt better to the topology of the signal**, and one of the research challenges is to understand why.

Looking at the research in this field from a higher perspective, everything outside of neural networks, that is, as soon as they give us the score, the math is well understood. So, the thing to understand is what the network does when it learns the score, and that is what we are going to attempt to do in this course, which stands at the frontier of current research.

If time permits, we will go further to see how we can condition the learning, for example, by providing a prescription c to the generator. This is a problem that can be viewed from a Bayesian perspective:

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

where $p(c|x)$ is a classifier and $p(x)$ is the unconditional generator. This year's seminars will focus on topics of generation by AI.

29. NbJE. See Sec. 3.1 and 3.2 of the 2021 Course for the linear/non-linear distinction.

3. Lecture of Jan. 22th

During this session, we will explore generative models, which, as a reminder, fall under the category of unsupervised learning. These models aim to estimate, either directly or indirectly, the assumed underlying probability distribution $p(x)$ from a set of supposedly *iid* samples $\{x_i\}_{i\leq n}$, in order to sample from it again. In this context, we saw in the previous session that there are various approaches to designing such generative models. The ones we will cover in this session are the classical Gibbs energy models (Sec. 2.6), which are particularly important in Physics, and GANs (Sec. 2.7), where the power of deep neural networks has emerged in this context.

In the case of "energy-based" models, sampling is performed using Markov chains (Course 2024) via Monte Carlo techniques (MCMC), making it a stochastic method. In the case of GANs, sampling is a deterministic process using a neural network trained to minimize a cost function. Recall that GANs, by nature, must achieve a form of equilibrium between the generator and the discriminator (Nash equilibrium), which is difficult to control. Consequently, alternative methods have emerged, as mentioned in the previous session, namely Normalizing Flows (Sec. 2.8.1), which perform a discrete probability transport, and score-based diffusion models (Sec. 2.8.2), which, in contrast, operate a continuous transport with a denoising network at their core.

3.1 Gibbs Energy Models

3.1.1 Typology of Parameterizations

As previously mentioned, these models were introduced in the 2023 and 2024 courses; however, we revisit them this year because they form the foundation of Statistical Physics and can serve as a guide in Machine Learning. We define a parametric family of distributions $\{p_\theta\}_\theta$ such that

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

For reference, the normalization constant Z_θ is the Gibbs partition function, from which all classical thermodynamic functions can be derived. The main challenge in this context

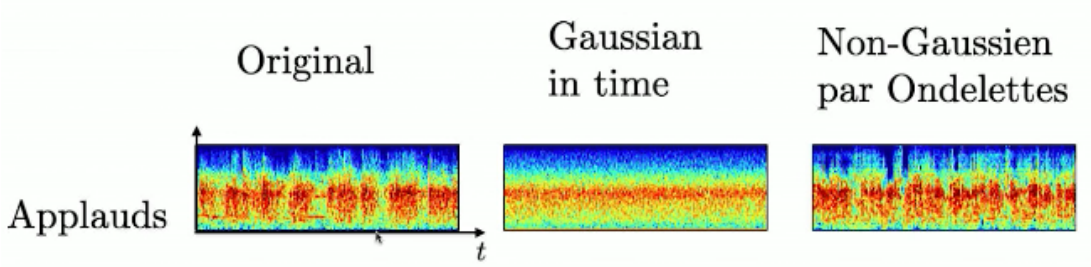


FIGURE 14 – Time-Frequency representations of an applause: the original sound on the left; in the middle, the generation from a Gaussian model, which only preserves the original energy across all scales/frequencies; on the right, the generation from a more realistic model that accounts for correlations across scales.

is choosing the model, specifically, what representation to use for the "energy" $U_\theta(x)$.

The classical models that have been extensively studied in Mathematics and Physics are **Gaussian models**, meaning those with a quadratic form in x:

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

where Σ_θ is the covariance matrix of the system. Generating samples x is relatively simple because Σ_θ can be diagonalized, revealing independent components. The typical sets (Course 2023) are revolution ellipsoids, whose principal axes, centered at μ, have directions given by the eigenvectors of Σ_θ. In this framework, we note that $Z_\theta = (2\pi)^{d/2}|\det(\Sigma_\theta)|^{1/2}$.

These Gaussian models are simple—perhaps too simple in the context of modeling intermittent phenomena. Figure 14 revisits an example from the 2023 course, illustrating the **lack of structure** when using a Gaussian model to describe a sound pattern exhibiting intermittency. Yet, these transient phenomena are ubiquitous, not only in speech processing but also in image processing, where object edges represent contrast transitions. These models can therefore serve as an initial approach since they are easy to implement, but they quickly become inadequate in practical cases where structural information is crucial.

To design more complex models, the natural idea was to introduce a **potential**, and in this case, the simplest models are those with a scalar potential:

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

In the hierarchy of complexity, **exponential models** then emerged, for which

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

where Θ is a parameter vector and $\Phi(x)$ is a family of functions $\{\phi_k(x)\}_k$ that must be chosen. Note that Gaussian models and scalar potential models are special cases of this type of modeling.

Extensive work in both Physics and Mathematics has been conducted to understand which functions $\phi_k(x)$ are most suitable. One idea is that if Gaussian models correspond to second-order polynomials, why not introduce higher-order polynomials? While this idea is appealing, it can be shown that as the polynomial degree increases, the estimation of parameters Θ becomes more unstable, requiring a vast number of samples to counteract the variance. Of course, other types of functions can be chosen, and there is a long-standing tradition of doing so in signal processing. However, what has "recently" emerged is the use of **neural networks**, where $U_\theta(x) = \mathrm{NN}_\theta(x)$, with a wide range of specific architectures to choose from. Estimating the parameters of the network in this framework is quite complex, which is why techniques such as GANs have emerged. Nevertheless, let us now study how optimization is performed.

3.1.2 Optimization of Parameters

What we seek is to obtain θ^* such that p_{θ^*} best estimates $p(x)$. To achieve this, we need a metric that constrains the choice of p_θ. The most classical approach, introduced by R. Fisher in 1922, is the **Maximum Likelihood** method [30] based on $\log p_\theta(x)$. If we have typical samples [31] x, it means their probabilities $p(x)$ are large, and the same holds for $\log p(x)$. Therefore, what we consider is $\mathbb{E}_{x\sim p}[\log p_\theta(x)]$, which is the expectation of the likelihood of $p_\theta(x)$ when x is distributed according to the *pdf* $p(x)$. We then seek to maximize this quantity to obtain θ^*:

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

30. NbJE. Course 2022 Sec. 3.5
31. nb. terminology of Cl. Shannon

The last equality introduces the cost function $\ell(\theta)$ to be minimized. This quantity is very natural in **Information Theory**. Expanding the expectation:

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

where we introduce the **Kullback-Leibler divergence** and the **entropy** of p. Thus, minimizing $\ell(\theta)$ (with respect to θ) is equivalent to minimizing $D_{KL}(p\|p_\theta)$. Moreover, we know [32] that

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

Therefore, minimizing $D_{KL}(p\|p_\theta)$ tends to adjust p_θ in the direction of p, and *ultimately*, this results in a good model of the data.

In the case of Gibbs energy models, $\ell(\theta)$ can be written as:

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

When we have samples $\{x_i\}_{i\leq n}$, we can compute an estimate of $\ell(\theta)$ using a Monte Carlo average to approximate the expectation:

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

Then, to estimate the optimal θ, we proceed by **gradient descent**: given an initial value θ_0, the transition from step t to step $t+1$ is performed as follows:

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

What are the challenges of this optimization?

32. e.g., Course 2023 Sec. 5.3 for the definition and positivity property of the Kullback-Leibler divergence. Hint: use of Jensen's inequality.

We need to compute:

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

We observe that the gradient of $\ell(\theta)$ is zero when $p = p_\theta$. In the case of exponential models:

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

which is manifestly independent of θ.

Now, the question of convergence arises, which amounts to determining whether $\ell(\theta)$ is convex or not. We therefore need to compute the second derivatives, i.e., the Hessian $H(\ell(\theta))$. In the case of exponential models, the first term of $\nabla_\theta \ell(\theta)$ (Eq. 32) does not depend on θ, so it does not contribute to the Hessian computation. It can be shown that if $A(\Theta) = \log Z_\theta$, then [33]

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

This tells us that **in the case of exponential models, the cost function is convex, and therefore we have a guarantee of convergence to a unique solution**, even though there may be optimization scenarios where the cost function is flat near the minimum. Nevertheless, these exponential models serve as a reference.

The situation is entirely different when $U_\theta(x)$ is modeled by a neural network, where we encounter many local minima. That being said, in cases where neural networks are used for classification or regression problems, even though there are local minima, the statistical properties of the models that converge differently are generally identical, even if we do not fully understand why most of the time. So, by and large, local minima do not present a fundamental obstacle. What, then, is the real difficulty?

Looking at equation 32, the first term is an expectation with respect to the data, meaning we can compute it easily by taking the empirical mean $\frac{1}{n}\sum_i$ over the sample set

33. NbJE. see Course 2023 Sec. 8.2, Th. 18

$\{x_i\}_i$. However, the problem arises from the second term, where the expectation is taken with respect to p_θ, which we are in the process of optimizing. A natural approach would be to estimate it at step t using the following method:

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

We realize that to perform *a single step* of gradient descent, **we need to sample p_{θ_t} a large number of times (N)** to estimate the empirical mean. However, sampling itself is a complex problem.

In summary, even though the mathematical framework is clear and we can, *in fine*, interpret the optimized energy functions, **the practical implementation is nearly prohibitive**. Thus, we must find another approach if we want to use a large number of parameters, particularly in the context of neural networks. And yet, we have no choice: we must use deep networks, as exponential models are not expressive enough to model probability distributions for the most complex cases.

However, as demonstrated by S. Mallat, **exponential models are capable of capturing the distributions of fields studied in physics**, such as those in Figure 4, whether in cosmology, turbulence modeling, or the statistical physics of spin models like Ising or ϕ^4 fields. This is significant. In fact, such problems can be tackled without neural networks. The structures that emerge at all scales can be captured through a judicious choice of $\{\phi_k(x)\}_k$, such as wavelet bases. More generally, there is **hierarchical information to be captured**. That being said, the fields in question are relatively "simple" because **they are stationary and ergodic**, meaning that if we examine these fields locally at different locations, we observe an underlying **translation invariance** in their statistical properties.

By contrast, the case of **faces** in Figure 5 or **bedrooms** in Figure 14 belongs to an entirely different category: **non-ergodic problems**.

Before moving on to the generation of such challenging cases, let us first examine how sampling is performed in the context of equation 35, why it can be difficult, and how we can develop methods to overcome these challenges.

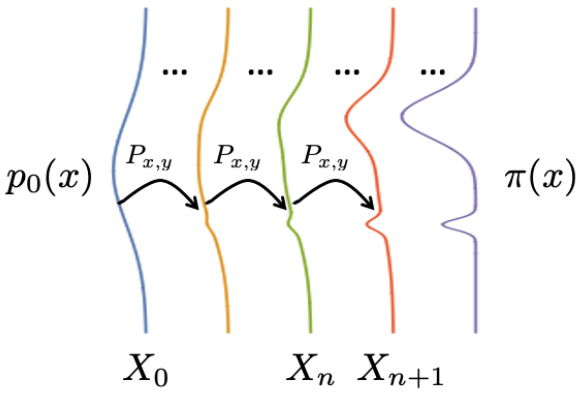


FIGURE 15 – Evolution of the initial distribution $p_0(x)$ towards the target distribution (here denoted as $\pi(x)$) through the application of the transition matrix $P_{x,y}$ of the Markov chain.

3.2 Sampling a *pdf* $\pi(x)$: Markov Chains

We aim to sample from a target distribution, for instance $\pi = p_\theta$, whose analytical expression is known (via that of U_θ), and we need to draw a typical and likely $x \sim \pi(x)$. What is the approach [34]? We will revisit the notion of transport. The idea is to start from a known *pdf* p_0, which is not the target but from which we can easily draw a sample x_i (e.g., uniform or Gaussian distribution), and then transport it in such a way that the set of $\{x_i\}_i$ follows the target *pdf*. Thus, **the problem is to find the transport** T. The classical approach to achieve this is by using a **Markov chain** [35]. This is a very powerful method, but as S. Mallat points out, it is, in a way, "too flexible."

Starting from $x_0 \sim p_0$, we progressively transform it by iteratively applying a well-chosen transition matrix $P_{x,y}$ to obtain a sample $x \sim \pi$ (Fig. 15). Thus, we consider the chain of random variables (*r.v.*) $(X_0, X_1, X_2, \ldots, X_n)$ and the corresponding sequence of *pdf*s $(p_0, p_1, p_2, \ldots, p_n)$. If we examine the conditional probability of the future given the entire past, the Markov property tells us that only the present matters in predicting the

34. NbJE. See also Course 2024 Sec. 7
35. Course 2024 Sec. 7.8

future. In other words,

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

where the right-hand side represents the transition probability from state $X_{n-1} = x_{n-1}$ to state $X_n = x_n$, denoted as $P_{x,y} = p(y|x)$, with x being the initial state and y the final state (note the order carefully). This Markov property is found in many physical systems [36]. This simplifies the general relation from equation 6:

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

Thus, we can easily access all joint statistics.

To this general framework, we add simplifications to obtain T. In principle, $p(X_n = x_n | X_{n-1} = x_{n-1}) = P_{x_{n-1},x_n}$ depends on n, but a simplification consists in imposing **independence with respect to n (stationary chain)**, leading to the generic notation $P_{x,y} = P$ (row-column matrix). Thus, at step $n+1$ in the evolution of the Markov chain, we have:

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

In matrix terms, if $\mu_k = [p(X_k = x)]_{x \in \chi}$ (column vector), then [37]:

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

We want μ_n to converge to $\mu_\pi = [\pi(x)]_{x \in \chi}$ as n tends to infinity. In the parameterized case, this is written as $\mu_\theta = [p_\theta(x)]_{x \in \chi}$, and the convergence must occur regardless of the initial distribution. This tells us that μ_π must be a **fixed point of the transformation**. Note that while the vector notation of μ refers to discrete states for simplicity, the idea generalizes to the continuous case. Now, we must ask: under what conditions on the matrix P is this convergence scheme possible?

The properties ensuring convergence are the following [38]: **irreducibility**, **positive recurrence** (or absence of cycles), and **uniqueness of the fixed point** (of the invariant mea-

36. NbJE. Some examples are presented in the Course 2023 Sec. 6.4.2 and in the Course 2024 Sec. 8.1.
37. NbJE. For consistency, I use the notations from 2023 and 2024
38. NbJE. See Course 2024 Sec. 8.4, including the ergodicity theorem (Th. 13).

sure). This provides a broad framework for constructing Markov chains (i.e., determining the transition matrix) with μ_π (and the target *pdf*) as a fixed point.

A specific algorithm achieves this: the **Metropolis-Hastings** algorithm [39]. We need to define the matrix $P_{x,y} = p(y|x)$, and the idea is to start from a known density $Q(x, y)$ [40], such as a Gaussian ($Q(x, y) \propto \exp(-\|x - y\|^2/(2\sigma^2))$), and modify it as follows:

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

If we retain only the Gaussian kernel, i.e., $\rho(x, y) = 1$, then for a fixed $x = x_0$, we sample $y \sim Q(x_0, y)$, and y in turn becomes the new value x_1, the next sample in the chain. However, in this case, there is no guarantee that π (p_θ) will be the convergence point. We must therefore modify the form of $\rho(x, y)$, which is nothing other than **the acceptance probability** of the sample y. This corresponds to a **rejection procedure** [41] applied to the proposed transition probability.

In fact, we need a **detailed balance property** related to the **property of reversible Markov chains**, which tells us that the invariant distribution π and the transition probabilities are related by the equation:

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

That is, the number of entities going from state x to state y is the same as those going from y to x. Thus, this provides a condition on $P_{x,y}$ for π to be a fixed point, and then gives the expression for ρ.

In the case considered by Nicholas C. Metropolis, $Q(x, y) = Q(y, x)$ (as in the Gaussian case), which gives [42]

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

In the case where $Q(x, y)$ is not symmetric, the algorithm 1 gives the general expression.

39. NbJE. Course 2024 Sec. 8.6, and see the 2023 GitHub repository, for example, the notebook `Monte_Carlo_Sampling.ipynb`.
40. NbJE. I follow the 2024 notation
41. NbJE. Course 2024 Sec. 7.5
42. NbJE. See Course 2023 Sec. 8.9

Algorithm 1 Metropolis-Hastings

Require: $Q(x, y)$ a distribution easy to sample to obtain x

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

In summary, we have a well-established theory based around Markov chains, with a guarantee of convergence using MCMC algorithms. What could go wrong in this well-oiled machine? The issue arises when $U_\theta(x)$ is **non-convex**, leading to a multi-modal distribution p_θ. Figure 16 illustrates the point: at each iteration of the Markov chain, if the Gaussian kernel $Q(x, y)$ has a width (σ) that is too small, we will most likely end up with a sample from p_θ corresponding to one mode (high probability), and we will almost have no chance of exploring other modes. Indeed, for this to happen, a new sample y from the kernel Q would need to "fall" into the attraction region of another mode (i.e., a high probability region) in order to have a chance to pass step 5, which requires a sufficiently large value of ρ for y to be accepted. However, if σ is too small, such a "jump" is impossible, and **the consequence is that we only generate one type of sample**. To address this, one might consider increasing σ significantly. In this extreme case, we may find that at each step, the proposed y has $p_\theta(y) \ll p_\theta(x_{n-1})$, and it is systematically rejected at step 5 of the algorithm, causing **the chain to stagnate**. Thus, we have two extreme cases where the Markov chain either generates a sample from a single mode or gets stuck in a trap where it makes no progress. **The chain does not correctly sample the probability density p_θ, and thus the collection of samples $(x_i)_i$ cannot be used either to compute statistical properties of p_θ, or to calculate integrals like the one encountered ($\mathbb{E}_{x \sim p_\theta}[\nabla_\theta U_\theta(x)]$) in the calculation of the gradient of the cost function to determine the optimal parameters θ^*. This is a major issue in high-dimensional Monte Carlo techniques, especially when using neural networks to model U_θ.** Of course, attempts have been made to modify the scheme outlined above, notably by trying the *replica-exchange* (RepEx) technique. First,

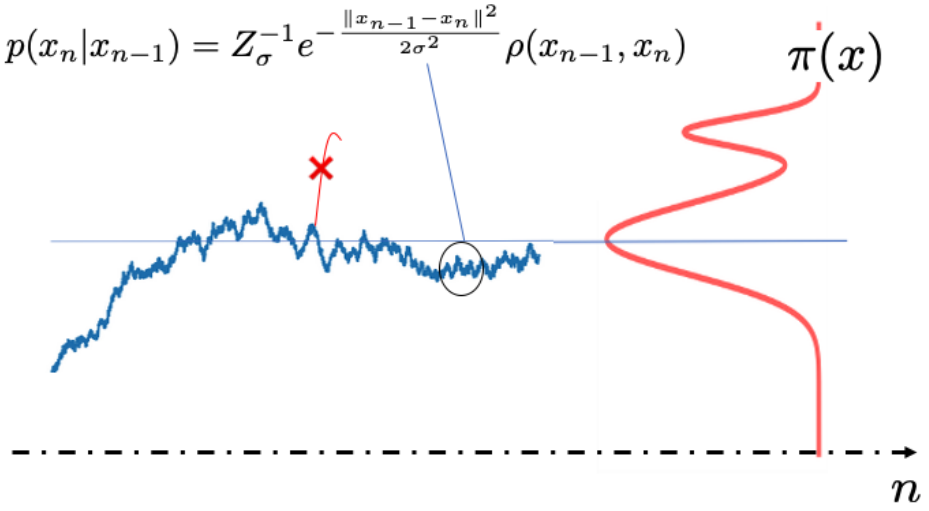


FIGURE 16 – Problematic case of a multi-modal target distribution $\pi = p_\theta$ ($U_\theta(x)$ non-convex and a Gaussian kernel $Q(x, y)$ with a too-small σ).

instead of trying to obtain samples from $p_\theta = \pi$ only, with its multi-modal issues, we aim to obtain samples from the family $\{\pi_\beta(x) = \pi(x)^\beta\}_{\beta\in[0,1]}$. Note that $\beta = 1$ gives us the target distribution, and for $\beta < 1$, we obtain flatter replicas where transitioning from one mode to another is easier. The parameter β is identified as the inverse of a temperature, reminiscent of thermodynamic models where $p \propto e^{-\beta U(x)}$. Second, imagine evolving several chains, each with different values of β. At step k, the chains i and j are in states (x_k^i, x_k^j), and the idea is that at step $k+1$ we swap the states, i.e., $x_{k+1}^i = x_k^j$ and $x_{k+1}^j = x_k^i$. Well, it's clear that if this exchange is done abruptly, neither chain will be a sample from π^{β_i} or π^{β_j}. We therefore need an acceptance probability for this exchange. In fact, the Metropolis criterion applies, namely

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

where we recognize the transition probabilities for chains i and j from equation 42. Now, we choose pairs of chains close in terms of β values, and the other chains use the "classic" MCMC scheme. There are many refinements in such algorithms. The fact is that we can potentially sample from multi-modal probabilities, but for this to work, note that in order

to obtain $x \sim p_\theta$ (at each step of minimizing $\ell(\theta)$), we need to sample the distributions $\beta < 1$, which ultimately do not interest us. Furthermore, as the dimensionality of the problem increases, the number of $(\beta_i)_i$ to consider also increases, which exacerbates the previous issue. Therefore, when using neural networks to parameterize U_θ, this approach does not work. The solution will be to **change the metric**.

3.3 Fisher Metric

The metric that seemed natural to us in order to make p_{θ^*} as close as we want to p was the Kullback-Leibler metric derived from Maximum Likelihood (Eq. 27). However, while the metric $D_{KL}(p||p_\theta)$ is indeed mathematically well-motivated, it is in some sense "too strong," as S. Mallat says: either we are dealing with a "simple" problem, and the metric is not an issue, or we are dealing with complex problems and must face the issues mentioned in the previous section, particularly that algorithms can be very slow. **Where does this optimization slowness come from?**

It arises from the estimation of the term $\mathbb{E}_{x\sim p_\theta}[\nabla_\theta U_\theta(x)]$ in the computation of the gradient of the cost function (Eq. 32). This term comes from the normalization constant Z_θ, which in turn comes from the calculation of $\log p_\theta(x)$. Can we get rid of it? What happens if we differentiate with respect to x rather than θ (careful!):

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

This is the **score** mentioned in the previous session. If we want to calculate expectations, we can do so directly from the training samples:

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

and the gradient with respect to θ is also simple to compute.

So, instead of maximizing $\mathbb{E}_{x\sim p}[\log p_\theta(x)]$ (maximum likelihood, Eq. 26) or minimizing with the negative sign, we will minimize the gradients, and the cost function then becomes

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

This is **the Fisher metric**. It forms the basis of the efficient sampling algorithms mentioned in the previous session. By bypassing the step of sampling according to p_θ at each gradient descent step, the algorithms are infinitely faster. However, there is a catch: in doing so, what happens to the normalization constant? In fact, we lose information, and this poses difficulties due to the weakness of the metric used. But by moving to differential evolution equations (the diffusion score algorithm), we will avoid these problems. But before that, let's look at the GAN step.

3.4 Network-Based Models: GANs

From the "classical" methods described in the previous sections, it is important to note that complex problems are difficult. That is, if we find an algorithm that works "miraculously," as S. Mallat says, we should always have a critical eye and ask ourselves: how promising is it really? Is there something that has been overlooked? This is the case for GANs mentioned (Sec. 2.7) in the previous session. They allow for the generation of beautiful images (Fig. 5 on the right), but the issue of getting stuck in a mode can persist in complex cases. That said, GANs have surprisingly shown: 1) that defining a "*face probability distribution*" can be credible, and 2) the expressiveness is provided by neural networks. Nevertheless, the question of generalization arises: can we distinguish the GAN from clever software arrangements of basic building blocks and modifications of the color, shape, or other parts of the face, etc.? In the following, we consider the image generation case in the background, but it generalizes, of course [43].

The basic idea is to start with white noise (we call these **latent variables**), and we need to find the appropriate transport to obtain x, a sample from the desired target distribution. See Figure 6 to illustrate this. If $x \in \mathbb{R}^d \sim p_{data}$, the latent random variable $z \sim p_z$ can, in general, be in a lower-dimensional space. We then define a neural network architecture, the **generator** parameterized by G_θ, such that $x = G_\theta(z)$. How can we test whether x is indeed a new sample from the underlying *pdf* of the data? As we saw earlier, if we go down the path of maximum likelihood, we know we will encounter optimization difficulties that may be prohibitive. Therefore, Goodfellow et al. wondered how to know if x is an image from the database (*true sample*) or if it comes from G (*fake sample*)? We

43. NbJE. In the 2025 notebook `JAX_blob_GAN_vanilla.ipynb`, I provide an example of "classical" GANs to generate 2D distributions made up of Gaussians centered on the vertices of regular polygons.

could simply call on some sort of *oracle* that would tell us "yes" or "no." In the absence of an oracle, we can create a classifier (or *discriminator*) again with a neural network architecture, $D_{\theta'}(x)$, giving the probability that x comes from the database.

To optimize both architectures, the idea is that G_θ and $D_{\theta'}$ act as adversaries: G_θ will attempt to make "real" samples from p_{data} as similar as possible, and $D_{\theta'}$ will try to detect them as "fake." When the discriminator can no longer differentiate between samples from G_θ and those from the data set, we consider that G_θ is producing good samples from p_{data}. Thus, we define the following cost function:

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

The first term maximizes the likelihood of the classifier's probability. The second term concerns both the generator and the discriminator: by minimizing it, we ensure that $D(G(z)) \approx 1$, i.e., that $G(z)$ produces a sample that resembles those from the data set. However, the discriminator will attempt to maximize this term and, if not fooled, will find that $G(z)$ is a fake image and will tend to give the response $D(G(z)) \approx 0$. The equilibrium situation (D^*, G^*) is a **Nash equilibrium** established in game theory. However, are we guaranteed convergence, and in particular, that $x = G^*(z)$ with $z \sim p_z$ are indeed samples from p_{data}? The answer is yes, provided we can find (D^*, G^*).

Let's consider $\max_D V(D,G)$, the cost function that the generator must minimize.

Lemma 1 *Thus, we are in the case where G is fixed in a state, and we seek the D^* that maximizes $V(D,G)$. Then,*

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

where $p_G(x)$ is the probability of x as the result of $G(z)$ with $z \sim p_z$. This is the push-forward of p_z, which is written as $p_G(x) = p_z(z) \times |J_G|^{-1}$, where J_G is the Jacobian of the transformation G.

Proof 1.

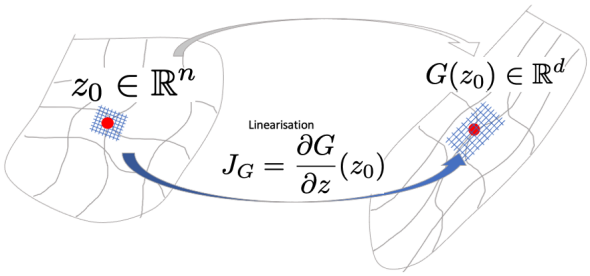


FIGURE 17 – Illustration of the Jacobian function of the generator G between the latent variable space (z) and the data space (x).

Let's write what $V(D, G)$ is:

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

Now, we can rewrite $p_z(z)dz$ by considering the transport G using the Jacobian of the transformation (Fig. 17). Thus,

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

where the last equality is simply the definition of $p_G(x)$. So,

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

Now, for $(a, b, y) \in [0, 1]$ and $(a, b) \neq (0, 0)$, the quantity $a \log y + b \log(1 - y)$ reaches its maximum at $y_0 = a/(a + b)$. Thus,

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

■

Having this result, how do we optimize the generator? In fact, we need to show that p_G converges to p_{data}, which incidentally gives that for all x, $D^*(x) = 1/2$. The metric to

minimize for G becomes, after optimizing D to D^*,

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

where $p_m(x) = (p_{data} + p_G)/2$ and D_{JS} is the **Jensen-Shannon metric** [44] which is symmetric, unlike the Kullback-Leibler divergence.

Now, the minimum of $D_{JS}(p_{data} \| p_G)$, according to the properties of the Kullback-Leibler divergence, is achieved *iff* $p_G = p_{data}$. Thus, we have our desired result, which is stated in the following theorem:

Theorem 1

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

Thus, **the GAN optimization in theory guarantees that we learn the *pdf* underlying the data**. The tricky part is that we need to find (D^*, G^*). In the next session, we will see that this is more complicated than it seems, and we will examine where the problem lies and how strategies have been implemented to avoid it.

4. Lecture of Jan. 29th

4.1 Review of GANs: Why does it fail?

As previously mentioned, the first paper on GANs dates back to 2014 [45], with the key point being that it was realized that one could **learn probability transports using deep neural networks**. In doing so, a whole evolution of ideas emerged, which we will explore. S. Mallat tells us that it is an archetype of "research that is currently underway," the driving force behind courses at the Collège de France. What makes this field unique is that research

44. NbJE. There may be different definitions involving a multiplicative factor.
45. NbJE. see footnote 20.

is progressing very quickly, even though the underlying mathematical concepts are not new. For example, the idea of "Transport," as previously discussed (Sec. 2.8), comes from the work of G. Monge in the late 18th century. Regarding GANs, it is the generator that plays the role of the transporter. As we saw in the previous session, to optimize the generator, we introduced an adversarial discriminator. The optimal discriminator, in turn, is given by the relation in lemma 1, and theorem 1 gives us the optimal generator-discriminator pair (D^*, G^*).

The proof of theorem 1 is simple. Indeed, to obtain G^*, we need to minimize the Jensen-Shannon metric (Eq. 53), which has the properties of the Kullback-Leibler divergence, with the added benefit of symmetry, making it indeed a metric. It is zero *iff* $p_{data} = p_G$, meaning the generator indeed learns the data distribution in theory. Thus, the theoretical scheme is clear, and we now need to move on to the practical side. Algorithm 2 is the one used by Goodfellow et al. (referred to as "vanilla GAN") [46].

Algorithm 2 GAN (Goodfellow et al., 2014)
It is necessary to specify the total number of generator update iterations, the number of times k that the discriminator is updated per generator optimization loop; the number m which gives the sample size, and the type of optimizer for the gradient descent.

---

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

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

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

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

Following the numerical experiments that followed, here is a list of the encountered

46. NbJE. I have provided a simple example in the 2025 notebook *JAX_blob_GAN_vanilla.ipynb* to help you get started.

issues:

— It is difficult to train due to the "adversarial" nature of the two networks, which leads to oscillation phenomena;

— As already mentioned, the main issue is that in the context of neural network optimization, there is a high likelihood of getting stuck in **a local minimum** of the discriminator. This leads to the **mode collapse** phenomenon, i.e., modes of p_{data} that the generation $G(z)$ does not produce *in fine*, because the discriminator would have rejected them as "fake" images during training. This is similar to the issue seen in the Markov chains in Figure 16. For example, in the case of face generation, the effect of *mode collapse* may result in generating only one type of face (or at least forgetting others). In the case of bedroom scenes, some might involve characters, and the *mode collapse* would result in only generating scenes without any people.

— Concerning oscillation phenomena, some may involve local minima characterized, for example, as follows: for a while, the generation fixes on one type of face, then suddenly switches to generating another type, and so on.

This all tells us that there are pitfalls because one can produce very beautiful images that clearly belong to one mode of the probability density (p_{data}), but there is no guarantee that the entire *pdf* will be reproduced. Ultimately, there is generally a **lack of diversity in the samples**. And unfortunately, the lack of diversity is very difficult to assess.

For roughly six years, the focus has been on producing beautiful images with increasingly larger networks to increase resolution. Figure 18 illustrates the evolution of GANs. However, do we have the correct probability distribution?

The critical point is the use (even in the symmetric form of Jensen-Shannon) of the Kullback-Leibler divergence, which causes problems because it is difficult to optimize [47] as we observed during the optimization of parametrized energy problems, which is found here

47. NbJE. Martin Arjovsky et al. provide a simple example in *arxiv:1701.07875*. Let's imagine a random variable Z with a uniform *pdf* over $[0, 1]$. Let p_0 be the distribution of points $(0, z)$ (this is the unit line segment on the vertical axis). Now, let p_θ be the *pdf* of points (θ, z) with θ a single real parameter. Considering $x \in \mathbb{R} \times [0, 1]$

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

we get:

— if $\theta = 0$, $D_{JS}(p_0 \| p_\theta) = 0$, and similarly $D_{KL}(p_0 \| p_\theta)$ and $D_{KL}(p_\theta \| p_0)$ are zero;

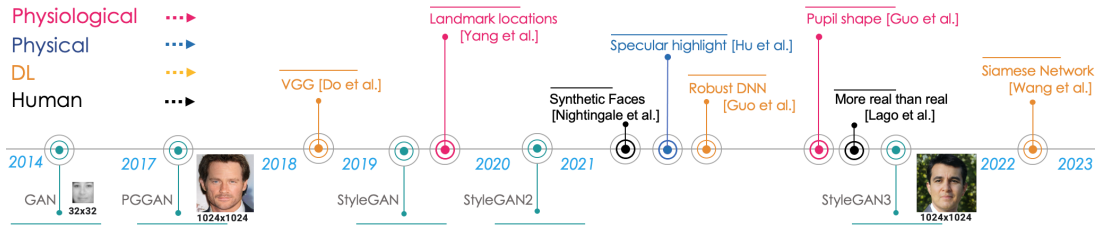


FIGURE 18 – A small timeline of GANs provided in the article by Xin Wang et al. *arxiv:2202.07145* (v6 dated Nov. 2023). After 2017, the series of models like StyleGAN enabled the generation of highly realistic faces, difficult to distinguish by the human eye. Parallel research is also focused on detecting images generated by such models to distinguish them from real images, for example, to help with *fact-checking*.

in a similar way. The question then arises: can we change the metric to provide information that p_G converges to p_{data}, without the disadvantages of the Jensen-Shannon and Kullback-Leibler metrics? It is during this search that we began to rethink the problem.

4.2 Optimal Transport

Concerning the GAN problem, we would like p_G, which is a *pdf* parameterized by a set of parameters θ_g, to be regular so that the gradient descent works well (NbJE. See footnote 47 for a simple example of the problem). In a way, we would like that if we consider a divergence/metric $D(p_{\theta_g} \| p_{data})$, we have the following property:

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

However, D_{KL} is very sensitive as we approach the optimum. We need to find something "weaker," while ensuring that $p_{\theta_g^*} = p_{data}$. Therefore, we are led to rethink the problem, and in particular, to use *optimal transport* metrics, which are associated with Gaspard Monge (1746-1818) and Léonid Kantorovich (1912-86).

— however, if $\theta \neq 0$, $x = (x_1, z)$ such that $p_0(x) = 0$ if $x_1 \neq 0$ while $p_\theta(x) = 0$ if $x_1 \neq \theta$; hence we conclude that $D_{JS}(p_0 \| p_\theta) = \log 2$, and $D_{KL}(p_0 \| p_\theta) = D_{KL}(p_\theta \| p_0) = +\infty$.

One can clearly see a convergence issue as $\theta \to 0$.

4.2.1 The Problem of G. Monge

G. Monge has a considerable body of work in geometry and analysis, co-founded the École des Arts et Métiers and the École Polytechnique (for which he wrote all the programs at its creation), was a peer of France, Minister of the Navy, member of academies, and more. Monge's problem, recorded in a report to the Academy of Sciences, was very practical: it concerned the problem of *Embankments and Excavations*[48]. The idea was to transport, with minimal effort, piles of sand to fill in holes. If z is the starting position of a grain of sand and $T(z)$ is the transposed position, the transportation cost is a function c that depends on the distance between these two positions, such as $c(x, y) = \|x - y\|^n$ (e.g., $n = 1, 2$). Then, the solution to the following problem must be found:

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

with p_d the distribution of the holes (p_{data} in our case). Monge's problem is to determine whether a solution exists, what the existence conditions are, and to find the solution.

Does a solution exist? In general, the answer is no. A counterexample can be constructed as follows:

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

T is a function, and $T(z_0)$ cannot be assigned simultaneously to both x_0 and x_1. "Reasonable" regularity conditions are required for solutions to exist.

4.2.2 Relaxation by L. Kantorovich

Regarding the construction of the solution to Monge's problem, it is actually a complex non-convex problem. It wasn't until Léonid Kantorovich's work in 1942[49] that a new perspective emerged. L. Kantorovich won the Bank of Sweden Prize (Nobel Prize in Economics) in 1973. What is his idea: instead of considering the deterministic transport

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

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

of G. Monge (moving grains of sand 1-to-1), he considers **stochastic transport**. This is the same idea as when considering Markov chains, where there is a transition probability between each step of the chain.

To perform the transport of p_z to p_{data}, let $\gamma(z, x)$ be the probability density over the space $\chi_z \times \chi_x$ that satisfies the following constraints:

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

These are schematically represented in figure 19. The first concerns the conservation of the probability density $p_z(z)$ when z is fixed, and the second symmetrically describes the conservation of the probability density $p_x(x)$ when x is fixed. These two properties guarantee that if γ exists, the equivalent property of Monge's transport $T_{\#} p_z = p_{data}$ holds, but this time in a probabilistic (or stochastic) manner. The set of distributions that satisfy the constraints 58 forms a set denoted $\Pi(p_z, p_d)$, which is the set of distributions on $\chi_z \times \chi_x$ that have marginals p_z and p_d.

We now need the equivalent of the transport at minimal cost. L. Kantorovich uses a metric that will later be called **Wasserstein** [50] which is written [51]

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

where the cost of **transport has been symmetrized**. The integral form of the cost is written as:

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

This metric has properties of a distance, and notably it is **convex** [52].

Why was this Wasserstein distance introduced? It is weaker than the Kullback-Leibler distance, meaning that if p and q are supported by a compact set, we have:

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

50. In honor of Leonid Vaserštein (1944-) even though it was L. Kantorovich who introduced it.
51. NbJE. Slightly different formulations may exist.
52. NbJE. Concerning the problem in footnote 47, $W_1(p_0, p_\theta) = |\theta|$ so it converges properly when θ tends towards 0.

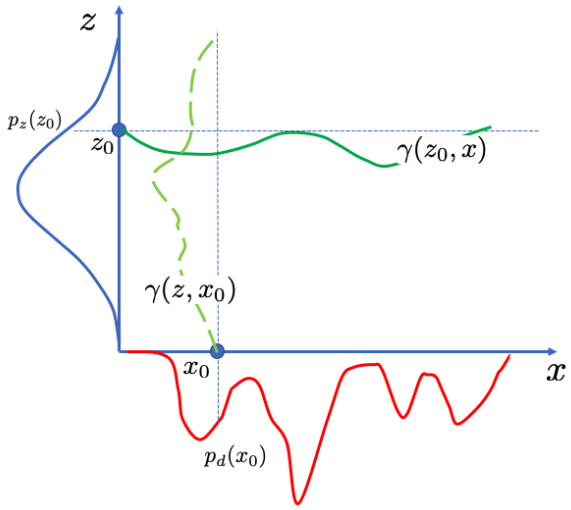


FIGURE 19 – Schematic representation of the constraints of L. Kantorovich's problem.

The Kullback-Leibler metric is more sensitive, but $W_1(p, q)$ is sufficient because if $W_1(p, q) = 0$, then $p = q$. Therefore, being weaker, W_1 should be easier to optimize. It follows that efficient algorithms can be developed as long as the dimensionality is not too high.

The idea of using W_1 was introduced by Martin Arjovsky et al. (see footnote 47) in 2017 to design the "*Wasserstein GAN*" [53]. The authors prove the following theorem, which we will accept [54]:

Theorem 2 *If T_θ is a transport from the pdf p to the pdf of $T_\theta(z)$ with $z \sim p$, denoted p_θ,*

1. *If T_θ is continuous in θ, then $W_1(p, p_\theta)$ is almost everywhere continuous in θ.*
2. *If, additionally, T_θ is locally Lipschitz, then $W_1(p, p_\theta)$ is almost everywhere differentiable in θ.*
3. *Both of the above properties are false for the Kullback-Leibler divergence and the Jensen-Shannon distance.*

53. NbJE. I have uploaded the notebook `JAX_blob_GAN_Wasserstein_regul.ipynb` to the repository, which is an adaptation of this GAN with gradient regularization.

54. nb. $f : A \to B$ (spaces equipped with a distance) is locally Lipschitz means that for every $(x, y) \in A$, there exists a constant K, denoted $\|f\|_L$, such that $d_B(f(x), f(y)) \leq K d_A(x, y)$.

The first property tells us that the W_1 distance adds **regularity to the generator's parameterized probability density function** (p_G), and the second property allows us to perform **gradient descent** for optimization.

So, *a priori*, we have a mathematically well-defined framework with the hope of implementing an efficient algorithm. Recall that our goal is to modify the cost function of the discriminator (Eq. 53) based on the Jensen-Shannon metric (itself composed of Kullback-Leibler divergences). There is a result by L. Kantorovich and G. Rubinstein (1958) stating that W_1 can be computed as follows:

Theorem 3 ***(Kantorovich-Rubinstein Duality)***

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

In practice, to compute $W_1(p, q)$, one can use an evaluation function (think of $D(x)$) to compute the difference of expectations, and when we reach the maximum, we obtain an evaluation of the metric. This can be translated for our pair of adversaries parameterized as $(D_{\theta_d}, G_{\theta_g})$, by searching in the set of $\{D_{\theta_d}\}_{\theta_d}$ locally Lipschitz ($\|D_{\theta_d}\|_L \leq 1$) and calculating [55]

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

Recall that we then need to minimize $V(D^*, G) = W_1(p_{data}, p_G)$ to obtain G^*. However, the convergence of p_G towards p_{data} will be facilitated by the differentiability of W_1. Note, by the way, that D_{θ_d} does not *a priori* give a probability, we no longer use likelihood, hence the disappearance of the log that was present in Algo. 2. In Martin Arjovsky et al.'s paper, the authors use the term "*critic*" to refer to D, but we will retain the term discriminator nonetheless.

By analyzing the above, we realize that by changing the cost function, we adapt the metric (or vice versa). **Optimizing the "discriminator" defines a metric between the data distribution p_{data} and p_G, the distribution transported by the generator.** And ultimately,

55. Recall that p_G is the pdf of $G(z)$ with $z \sim p_z$

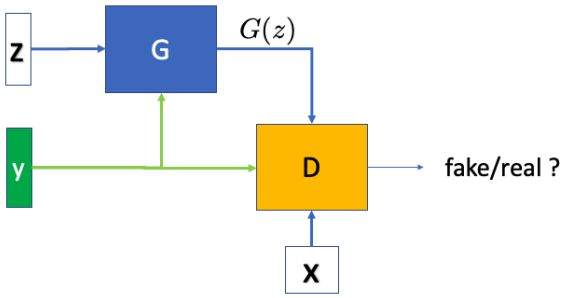


FIGURE 20 – Schematic of a conditional GAN: compared to the diagram in Figure 6, the information y is added to the input of both the generator and the discriminator.

what we want is a metric that is easy to optimize to get the correct generator, that is, the one for which $p_G = p_{data}$.

4.3 Conditional GANs (cGAN)

The idea[56] of Conditional GANs (Fig. 20) is that we will "guide" the generator and the discriminator by providing them with additional information y, which can be of various types (prompts, class labels, another image, etc.). In this context, we consider a cost function $V(D, G)$ conditioned on y, for example,

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

and we operate the min-max problem of the adversaries as in the case of classical GANs (Eq. 47). Note that we could just as well use the Wasserstein metric. Practically, y can be added to the vector z (resp. x) at the input of the generator (resp. discriminator).

S. Mallat presents examples of images generated by such "cGANs"[57]: for example, by training the GAN with images taken in both summer and winter and providing this

56. NbJE. Mehdi Mirza, Simon Osindero (2014) *arxiv:1411.1784*, released shortly after Goodfellow et al.'s work.

57. NbJE. These images are extracted from the review article by Yingxue Pang et al. (2021) *arxiv:2101.08629*. You can also read the earlier article by Ph. Isola et al. (2018) *Image-to-Image Translation with Conditional Adversarial Networks*, *arxiv:1611.07004* associated with the software `pix2pix`

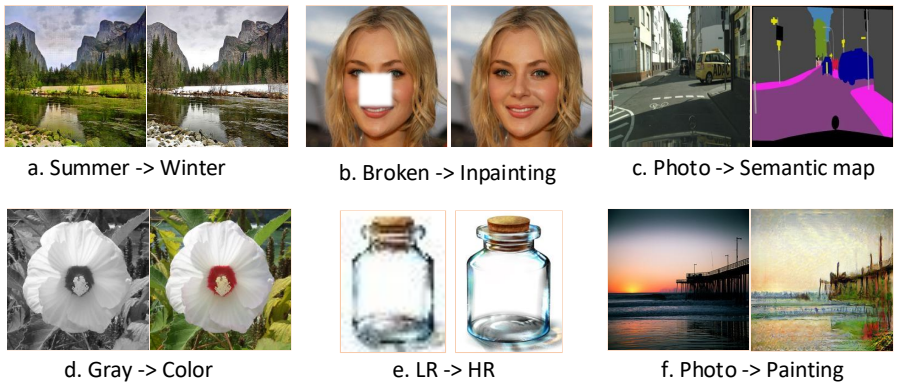


FIGURE 21 – Some illustrations of image-to-image transformation with a conditional GAN (source *arxiv:2101.08629*).

information, we can transform a photo of a landscape taken in summer to make it appear as if taken in winter. In this scheme, tasks such as *inpainting*, segmentation, and many others can be performed (Fig. 21). The problem becomes easier because it is more constrained. The article also shows comparisons between several cGANs (differing in architectures, cost functions, etc.) in the task of modifying a face based on five criteria (two hair colors, two genders, and age) (Fig. 22).

4.4 Evaluation of GANs: Fidelity Criteria

The numerical experiments conducted on GANs and cGANs over time seem convincing, but as always, the question arises: does it really work well? That is, do we have $p_G = p_{data}$? The underlying problem is that **there is no explicit optimization of a metric**. Regarding W_1, we rely on a duality argument (Th. 3), and to access the correct metric on G, we need to obtain the optimal D, but we are not ultimately certain of that.

"Experimenters" have turned to **qualitative evaluations** involving [58]:

— a **fidelity criterion** that answers the question: is the image of "good quality"? That is, can the generated image be considered faithful to what we perceive in an image

58. Note: The case of images is just for illustration, it generalizes.

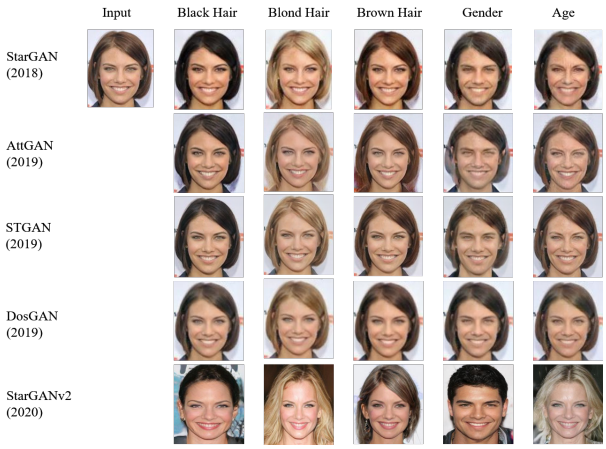


FIGURE 22 – Comparison of different cGANs in generating face images conditioned on 5 attributes (source *arxiv:2101.08629*).

from the dataset? Mathematically speaking, this roughly translates to the fact that the support of p_G (cf. typical sets) is such that

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

— to which we must add **diversity**, meaning we need the reverse inclusion

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

However, this is the big issue with GANs.

In practice, we have examples (*real*) from the dataset $\{x_i^{data}\}_i$ and examples (*fake*) generated by the generator $\{x_j^g\}_j$, and as a fidelity criterion, we can consider calculating a **metric on moments**. That is, starting from an x, we define a set of descriptors (*features*) ($\Phi(x)$), which could be simple things like moments of different orders, or the outputs of neural networks, etc. We can then calculate

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

and similarly for $\mathbb{E}_{x\sim p_G}[\Phi(x)]$. The question that arises then is whether

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

The difficulty lies in choosing [59] the $\Phi(x)$. The descriptors used in image processing are `FID` descriptors (*Fréchet Inception Distance*) introduced in 2017. Regarding "Fréchet" (Maurice Fréchet), it comes from the distance between two random variables (mean μ, standard deviation σ, or covariance matrix)

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

We use the random variables $X = \Phi(X^{data})$ and $Y = \Phi(X^g)$. Regarding Φ, we train a particular neural network architecture, Google's `Inception` (v3, 2016), on the `ImageNet` dataset to ensure it is a good classifier, and choose the middle layer of the architecture as $\Phi(x)$.

Now, **nothing guarantees that the descriptors are sufficiently "fine"** to evaluate the response to the question posed earlier (Eq. 68), nor even if they account for the quality/diversity of the generation. The choice of Inception, the ImageNet dataset, and its training inevitably introduce bias [60]. NbJE: In practical terms, in the context of astro analysis, we might question if galaxy images can be treated this way (FID test), given that ImageNet does not contain any? But more dramatically, S. Mallat points out that the FID method is not very effective at detecting **mode collapse** issues.

Ultimately, the community reached a point where the images were certainly becoming more and more beautiful, but there was no demonstration that $p_G = p_{data}$. **The problem of diversity is really not a trivial matter**; if we simulate environments to optimize a system but the simulation "neglects" some scenarios, the system in question will not be robust to the occurrence of perhaps rare events, but not necessarily. **Therefore, *in the end*, these generators are not sufficiently reliable in cases where this notion is important.** Other approaches were eventually proposed to address these issues.

59. NbJE. In the case of galaxy image generation, for example, variables related to "morphology" are used, such as those produced by the Python software `statmorph` https://github.com/vrodgom/statmorph.

60. NbJE. One could even wonder if a GAN can be conditioned to pass the "FID" test.

4.5 Normalizing Flows (NF)

S. Mallat will present an algorithm implemented by Kingma and Dhariwal [61] (`Glow`), which is a culmination in this field [62] can be cited. It is worth noting that Goodfellow et al.'s article was only four years earlier. The mathematical framework is identical, where a transport T is defined from a simple distribution $p_z(z)$ to a more complex data distribution $p_{data}(x)$ that is more difficult to sample from initially.

The first additional idea introduced in Section 2.8.1 is to impose that the **transport be reversible** (Fig. 8), and that both T and T^{-1} be **differentiable**, making them **diffeomorphisms**. This imposes that the **dimension of the latent space** (the space of z) be **the same as the dimension of the data space** (the space of x). It is worth noting that in the case of GANs, there is no such constraint.

The second additional idea, also already mentioned, is that the global transport from p_z to p_{data} is decomposed into a discrete collection of **elementary transports** that are easier to optimize:

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

There are three advantages that *in principle* are provided by this approach:

— z is a **latent variable** that can be computed from x (unlike in GANs). For example, once the generator is optimized, one can generate samples $\{x_i^g\}$, transport them in reverse to a collection of $\{T^{-1}(x_i^g) = z_i^g\}_i$, and test if they are indeed drawn from p_z, which is much easier to handle. Additionally, starting from a z^g, one can perform an Euclidean translation (a metric in the space of a *bbg*), $z' = z^g + \delta_z$, and again generate a new sample $x^{g'} = T(z')$, allowing for continuous image deformations and also adding (or removing) glasses from a face, etc.

— We will perform an **explicit calculation of distances**, in this case, the **likelihood**, which means the **Kullback-Leibler divergence**. We can directly compare algorithms by calculating the value of this divergence.

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

62. NbJE. Concerning Normalizing Flows, an earlier article by 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

— Finally, **we avoid the Nash equilibrium** underlying the min-max of GANs, which is the origin of the oscillation phenomena in GANs.

What remains from the experiments performed with GANs is that **the transport is computed using a deep neural network**.

After the release of the article and the implemented GAN (*Wasserstein GAN*), further experimentation was carried out [63]. While it works a bit better, it is not a groundbreaking improvement over the initial solution by Goodfellow et al. Overall, **we find that it does not solve the problems** mentioned at the end of section 4.1. We are still in the context of a Nash equilibrium, and there are still oscillations between the two adversaries. The problem remains very complex, with local minima that can block the maximization of D, preventing the accurate computation of the Wasserstein distance (Eq. 63), so ultimately the optimization of G suffers. The issue is that we are in **very high dimensions**, and the algorithms become quite heavy [64].

4.5.1 The Effect of a Transport

To understand how a NF is optimized, we will examine the effect of a transport on a probability distribution. We note that $p_g = T_{\#}\, p_z$, but how is p_g obtained? We can refer to Figure 23: the key idea is that the transport being deterministic, there is **conservation of probability** for a set $A \subset \Omega_x$ (the space of generated x) during the transport of $z \in \Omega_z$ that generated it. We can use any function h defined on A, and the conservation constraint then becomes:

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

If $h = \mathbf{1}_A$, the indicator function of A, then the equality becomes:

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

63. NbJE. See, for example, Ishaan Gulrajani et al. (*arxiv:1704.00028*) published after Martin Arjovsky et al.'s work.

64. NbJE. For example, constraints must be added to the gradients at each optimization step of the discriminator, which significantly slows down the algorithms.

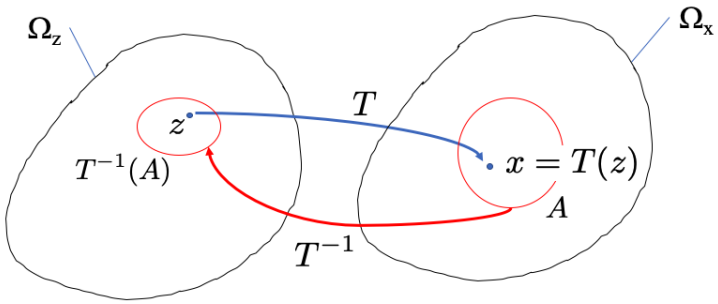


FIGURE 23 – Schematic of the law of conservation of probabilities (Eq. 71)

Now, what is the relationship between dx and dz, with $x = T(z)$? This is a change of variables where the determinant appears [65] of the square Jacobian matrix [66] of the transformation (Fig. 17):

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

Therefore, equation 71 becomes:

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

and this holds for any h, leading to the relation mentioned in Section 2.8.1, namely that p_g is computed from p_z and the transport T^{-1} according to:

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

The Jacobian accounts for the difference in volume between A and $T^{-1}(A)$.

4.5.2 Optimization of the Likelihood

Consider a parametrized transport T_θ where the θ are, in particular, the weights of a deep neural network. What we want is for the distribution p_z to be transported to p_θ (e.g., p_G) in such a way that p_θ best approximates p_{data}. As already seen previously, we use the likelihood as in the case of parametric models (Sec. 3.1.2), where the cost function

65. NbJE. The determinant is either denoted det A or $|A|$, but in any case, we take the absolute value.
66. NbJE. See Course 2019 Sec. 8.1.3

to minimize is written as:

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

The advantage is that we can estimate the value of the likelihood once the optimization is completed. Now:

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

So, since p_z is known as white noise, $\log p_z$ corresponds to a squared norm, and we now need an invertible transport whose Jacobian we must calculate. And when considering the chain of transports, we need to compute the sum of all the log of the determinants of the Jacobians. The technical challenge is therefore the calculation of these Jacobians.

We will see that we can simplify this calculation for specific choices of transport types [67]. There will be a new evolution, and at a certain point, the community realized that discretizing the transport becomes too complicated, and it would be better to attempt the path of *continuous transports*.

5. Lecture of Feb. 5th

NbJE. A brief note on notations. In what follows, the notations of the pdf p_{data} and p may be confused, p_g is the pdf of a generator, and p_z that of the latent variables. Moreover, p_0 is the initial pdf in a process where p_t represents the pdf of the samples at an instant t of evolution. Depending on the case, p_0 is identical to either p_z or p_{data}. It is therefore necessary to pay attention to the context.

We continue our exploration of Normalizing Flows from the previous session. The new aspect compared to GANs is that we have a metric, likelihood, to assess the quality of the model (Eqs. 76, 77). If we have data $\{x_i\}_{i\leq n}$ *iid* according to p_{data}, then the likelihood can be approximated using the Monte Carlo mean:

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

67. NbJE. On the GitHub repository, I have provided 1) a note (in French) that I wrote in 2022 showing some architectural details, 2) two notebooks `TensorFlow_bijector_1D_simple.ipynb` and `JAX_FLows_MAF_NVP_simple.ipynb` that implement simple NFs.

To find the optimal θ^*, we perform a gradient descent to minimize $\ell(\theta)$, knowing that the problem is likely not convex. However, we hope that convergence is possible towards a sufficiently deep local minimum (comparable to the global minimum). Thus, $\ell(\theta^*)$ provides information about the quality of the optimized model. We have the framework—how is this implemented in practice?

5.1 In practice: the `Glow` model

The first already mentioned point is that the global transport T (from p_z to p_g) is factorized into multiple elementary transports $(T_i)_{i\leq k}$ (Fig. 8, Eq. 11). Thus, if $z_{i-1} = T_i^{-1}(z_i)$ ($z_0 = z$, $z_k = x$), we have:

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

Therefore, the determinant can be easily computed provided that the determinant of each transport T_i can be computed. This is the potential bottleneck if the type of transport is not chosen wisely. It must be **sufficiently flexible not to hinder the model's expressivity while having a determinant that is simple to compute**. It is important to realize that the determinants involved correspond to very large matrices, typically of size d^2 with $d = O(10^6)$. For this reason, the literature has explored operators with triangular matrices, where the determinant is simply the product of the d diagonal elements.

S. Mallat describes the algorithm leading to the architecture of the `Glow` model by Kingma and Dhariwal[68]. The model is built using the following types of transport:

— **affine coupling**: the variable x of dimension d is split into two vectors (x_0, x_1) of respective dimensions (d_0, d_1) ($d = d_0 + d_1$), and the (inverse) transformation is given by

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

68. Kingma D. P., Dhariwal P., 2018, "*Glow: Generative Flow with Invertible 1x1 Convolutions*", in Bengio S., Wallach H., Larochelle H., Grauman K., Cesa-Bianchi N., Garnett R., eds, NIPS '18 Vol. 31, Advances in Neural Information Processing Systems. Curran Associates, *arxiv:1807.03039*. NbJE. For those interested in structures used in models based on Normalizing Flows, see the review by Papamakarios G., Nalisnick E., Rezende D. J., Mohamed S., Lakshminarayanan B., 2021, *Normalizing Flows for Probabilistic Modeling and Inference*, Journal of Machine Learning Research, 22, 1, *arxiv:1912.02762*

where one part remains the identity, and the other applies the Hadamard product $\odot$ so that each component of x_1 is multiplied by a nonzero coefficient (the exponential) that depends only on x_0, to which a bias is added. The functions s_θ and t_θ depend on parameters and can be quite complex, particularly in the case of **neural networks**. This operator is easy to invert ($z = (z_0, z_1)$):

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

Moreover, the determinant of $J_{T^{-1}}$ is given by the d_1 components of the vector $e^{s_\theta(x_0)}$, and by taking the logarithm, we obtain

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

which is easy to compute.

— **(linear)** 1×1 **convolution**. Notice that during an affine transformation, only part of x, namely x_1, is affected. Thus, the idea is to apply **cascades of affine operations** (referred to as layers) by interleaving each layer with an **operation that mixes the components**. To achieve this, a linear operation known as "1×1 convolution" is used (Fig. 24). This operator acts along the channel dimension (the same for all pixels) such that if (i, j) identifies a pixel in an "image patch" of size $W \times H \times C$ (*width, height, channels*), then:

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

where w_θ is a linear mixing operator along the channels, imposed to be invertible, so that:

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

The determinant of the Jacobian of T_θ^{-1} is given by [69]

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

(Note: if C is the number of channels, the complexity of computing $\log|w_\theta|$ is $O(C^3)$, which can be reduced to $O(C)$ using an LU decomposition if necessary.)

69. Recall: for a linear operator A, $|A|$ denotes $|det(A)|$.

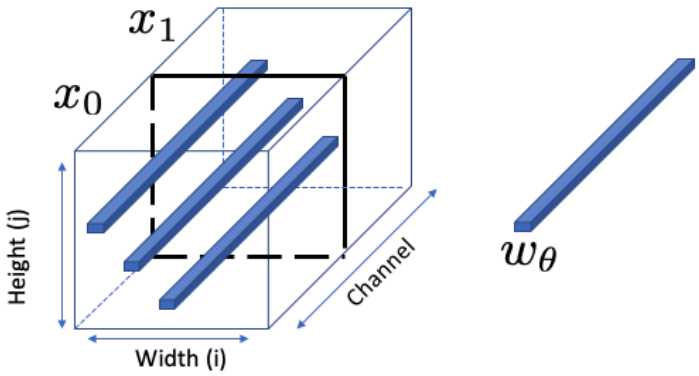


FIGURE 24 – Diagram of the 1×1 convolution operation (vector w_θ of dimension C) that acts identically on the channels across the $H \times W$ pixels.

— **activation normalization**. Using the notation of the image patch above, a normalization along the channels is applied to each pixel (similar to BatchNorm).

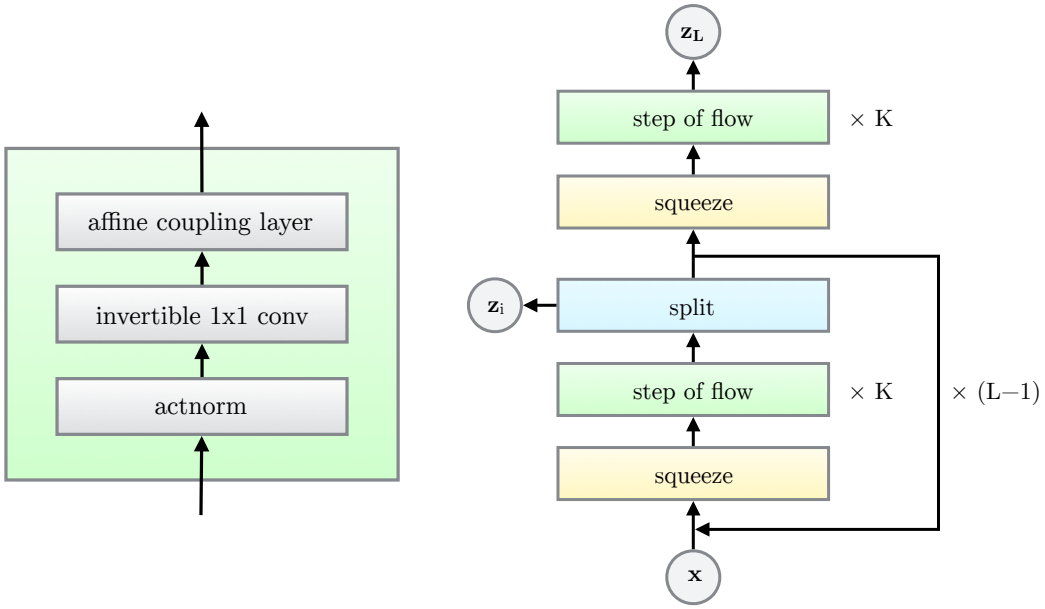
$$z_{i,j} = s_\theta \odot x_{i,j} + b_\theta \tag{86}$$

The determinant of the Jacobian of interest is $H \times W \times \log |s_\theta|$.

Thus, the `Glow` algorithm operates through cascades (layers), as illustrated in the figure extracted from the original paper: it shows the different actions mentioned above, forming a set that is essentially repeated multiple times, yielding a multi-scale architecture [70]. The number of parameters is on the order of 44 million.

S. Mallat presents examples of images similar to those in Figure 5. NbJE. I provide some examples in Figure 26. These are quite remarkable, but they do not reach the state-of-the-art. It is evident that there are flaws in these images, particularly in the background and hair details. If training images of indoor scenes or churches are used, the general observation is that **the generated images are of lower quality** than those produced by GANs, although we have noted that while quality is important, diversity is another key factor. Why is the quality not optimal? This stems from the fact that the constraints

70. NbJE. Some operations are derived from the paper by Dinh L., Sohl-Dickstein J., Bengio S., 2017, *Density estimation using Real NVP*, in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, https://openreview.net/forum?id=HkpbnH9lx (updated 2023)



(a) One step of flow.

(b) Multi-scale architecture.

FIGURE 25 – Architecture of the `Glow` model (source: see footnote 68).

FIGURE 26 – Some images generated by `Glow`.

on **the operator** T result in it being **not expressive enough**.

Despite this, there is an interesting aspect of these models, which consists of leveraging the fact that z **is a latent variable that allows us to manipulate images** [71]. Given two images (x_0, x_1), we can find their corresponding latent variables (z_0, z_1) in the latent space (via the transformation $z_i = T^{-1}(x_i)$). This space, for which $p(z) \propto \exp(-\|z\|^2/2\sigma^2)$, is equipped with a natural metric: the Euclidean metric. We can use it to measure distances between images.

If we want to obtain a series of images "interpolating" between x_0 and x_1 (Fig. 27) while ensuring they remain samples from the probability density function $p_g(x)$, it is easy to compute them using $z_\varepsilon = z_0\varepsilon + z_1(1-\varepsilon)$ $(\varepsilon \in [0,1])$, then apply $T(z_\varepsilon) = x_\varepsilon$ (Fig. 28). Any linear interpolation of the form $x_0\varepsilon + x_1(1-\varepsilon)$ will produce images, but they will likely exhibit artifacts and thus will not be valid samples from p_g. To interpolate correctly, one would need to understand the geometry of this space and follow geodesics, which is highly complex. Note that geodesics in the latent space correspond to circular arcs (in high dimensions), which are often approximated by straight lines.

It is also possible to add "structures": for example, making a face appear more or less smiling, modifying skin or hair color, or adjusting the apparent age of a person (Fig. 29). To achieve this, we need examples from the dataset where a criterion (label) has been defined, providing, for instance, an indication of the "smile" intensity (which is not necessarily easy to implement). Suppose we have a binary label $\ell \in \{0,1\}$ that identifies two sets of images: $X_0 = \{x \ / \ \ell = 0\}$ and $X_1 = \{x \ / \ \ell = 1\}$. For each image in

71. NbJE. You can visit the site https://openai.com/index/glow/, where you can perform some manipulations.

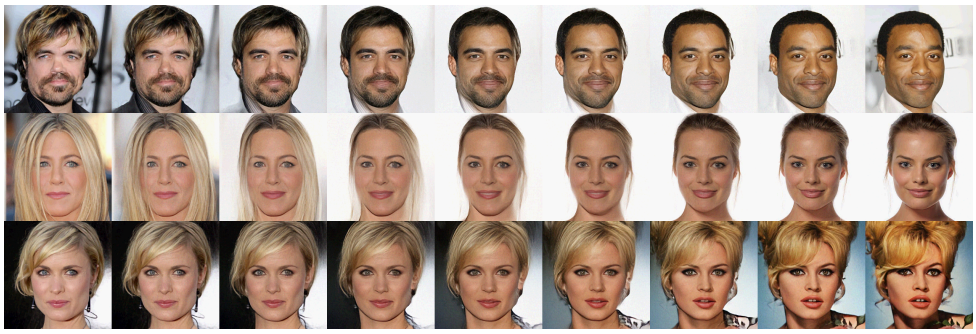


FIGURE 27 – Some images generated by `Glow` through interpolation in the latent space.

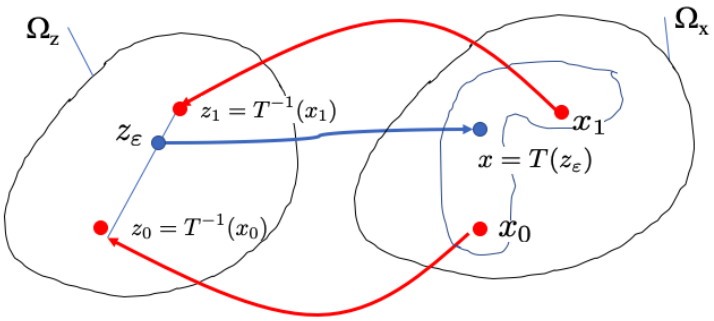


FIGURE 28 – Diagram illustrating image interpolation (see text).

FIGURE 29 – Example of transformations (Glow) applied to a person's smile.

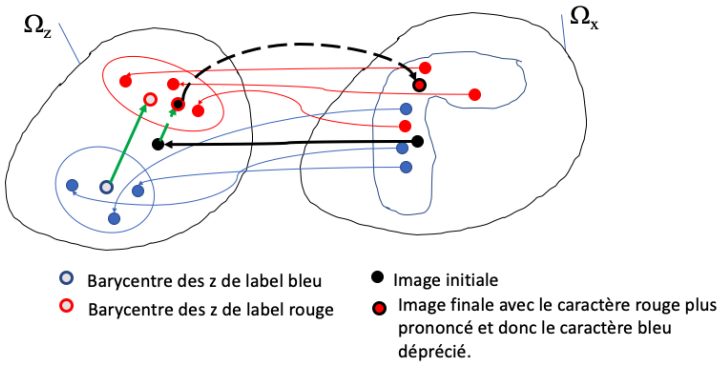


FIGURE 30 – Diagram illustrating the addition of a characteristic (a structure) while diminishing its opposite. Here, the characteristic is represented in red, and its opposite in blue. By identifying images in the dataset labeled as red or blue, we can construct two sets in the latent space via the action of T^{-1}, compute their barycenters (gray points encircled in blue or red), and determine the direction "blue to red" (solid green arrow). Then, given an image (black point), we use T^{-1} to find its corresponding latent variable, which we can propagate in the direction of the red-labeled space (dashed green arrow, black point encircled in red). Applying T yields an image with the red label (red point encircled in black).

these two sets, we can apply T^{-1} to obtain the corresponding latent variables Z_0 and Z_1, with $Z_i = \{T^{-1}(x) \ / \ x \in X_i\}$. We hope that the label is such that the sets Z_0 and Z_1 are well separated.

Now, consider the barycenter z_i of each set Z_i, so that z_0 represents the latent variables of type $\ell = 0$ (and similarly for z_1). Given an image x from the dataset, we can determine its corresponding latent variable $z = T^{-1}(x)$. If we want to obtain a version of x that exhibits more of the characteristic $\ell = 0$, we construct the direction ($e_{1,0}$, unit vector) supported by the segment (z_1, z_0) and translate z in this direction to obtain $z_\varepsilon = z + \varepsilon e_{1,0}$. Then, by applying the transport $T(z_\varepsilon) = x_\varepsilon$, we obtain the desired result (Fig. 30).

Here, we have explored a specific model [72], but the ideas are quite generic for this type of Normalizing Flow approach. At this stage, **we have a mathematical framework that is better understood** compared to GANs. However, as previously stated, **the transports are not expressive enough** to model complex problems. This limitation arises due to constraints on the determinants required for practical implementation. Therefore, we need

72. NbJE. See footnote 67 for simple NF code notebooks.

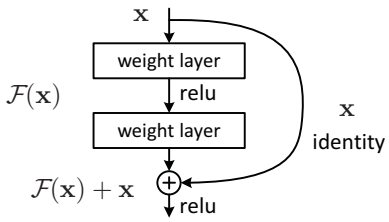


FIGURE 31 – Illustration of a *skip connection* (source: *arXiv:1512.03385*).

to revisit the method, and in particular, we will explore **continuous transports**.

5.2 Ordinary Differential Equation (ODE)

The Normalizing Flow approach introduced a discretization of the transport T. The idea is to extend this discretization to a continuum of transformations, leading us to a differential equation (ODE for *ordinary differential equation*). Thus, the transformation from z in latent space to x in image space occurs through an infinitesimal increment $dx(t)$, where t is interpreted as a "time" variable (note: in NF, we consider macroscopic Δt for analogy). The ODE is then written as:

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

where $V_{\theta,t}$ plays the role of a parameterized "velocity" function. A discretized Euler version is given by:

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

In terms of neural networks, where $V_{\theta,t}$ can be considered as a layer, the structure resembles that of a ResNet [73], i.e., a network with *skip connections* (Fig. 31).

Conversely, if each layer of a ResNet allows for a transformation like that of Equation 88, then one can conceive that **a complete ResNet can, in the limit, represent a transport scheme associated with an ODE**. This is the idea behind *Neural ODEs*, stu-

73. NbJE. Kaiming He et al. (2016), *Deep Residual Learning for Image Recognition*, Conference on Computer Vision and Pattern Recognition. *arXiv:1512.03385*.

died around 2018-19 [74]. This led to a research question: *Can all trained neural networks be approximations of differential equations*? The answer is *no*, but the idea is intriguing, and we will explore it further.

Once we have $V_{\theta,t}$, the probability distribution of x_t, denoted p_t, also follows an evolution equation. This is a fundamental topic in physics: what is the equation governing the phase space (position-momentum) of a collection of particles? This problem gives rise to the **Liouville equation** [75], which is essential for understanding dynamical systems. This is particularly relevant to chaotic systems, where predicting the trajectory of each individual particle is unrealistic since even the smallest instabilities cause initially close particles to diverge over time. However, the statistical approach allows us to compute the evolution of the probability density of the entire set of particles. This is a typical case in meteorology, where initial conditions in a Navier-Stokes equation are perturbed to generate an ensemble of weather forecasts, enabling statistical analysis of future weather conditions.

How is p_t related to $V_{\theta,t}$? In this context, how are the "determinants" computed, which posed a challenge in the case of Normalizing Flows? First, does a solution to the ODE (Eq. 87) exist? The answer is given by the following classical theorem:

Theorem 4 ***(Cauchy-Lipschitz)***
Consider the following problem where $V(t,x)$ is defined on an open set Ω of $\mathbb{R}\times X$ (X being a normed vector space of x), and is continuous from Ω to X:

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

where $(t_0,x_0)\in\Omega$.

Under the assumption that V is locally Lipschitz in the variable x, i.e., if we

74. NbJE. See, for example, RTQ Chen et al. (2018), *Neural ordinary differential equations*, Advances in Neural Information Processing Systems. *arXiv:1806.07366*.

75. Joseph Liouville (1809-32), mathematician (complex analysis, differential topology, mathematical physics, etc.), professor at the Collège de France

denote $A \subset \Omega$, then there exists a constant k such that

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

(meaning the velocity fields are regular, i.e., differentiable almost everywhere with a bounded derivative).

Then, there exists a solution $x(t)$ within a time interval that includes t_0.

Proof 4. A priori, to compute $x(t)$, one would proceed with the following integration:

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

The question is whether there exists an x that satisfies this equation. Define the operator T:

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

We need to find the solution where $T[x(t)] = x(t)$ (**fixed point**), which can be approached using the following simple iterative algorithm:

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

For this, we need to ensure the contraction property of the operator T, expressed as:

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

Now, for all $u \in [t_0, t]$:

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

thus [76]

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

where we have used the Lipschitz condition of V with respect to x for fixed s. Since this inequality holds for all u, we conclude that:

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

To obtain the contractiveness of the operator T, we must choose t such that $k|t - t_0| = \gamma < 1$. Over the interval $[t_0, t]$, a solution $x(s)$ can then be obtained using the algorithm described above. Banach's theorem guarantees that this fixed point is unique. ■

From this, we deduce that the solution to the ODE can be constructed, but at the cost of a fairly strong constraint on the velocity field, which must be sufficiently regular to enable transport. In a way, this brings us back to the diffeomorphism property of Normalizing Flows. The question now is whether we have gained in expressiveness to improve upon them.

5.3 Liouville's Equation

The remaining question concerns the relationship between probability density and the velocity field. Returning to the analogy of particles moving in the phase space of classical mechanics, we can formulate a **continuity equation** for probabilities.

76. NbJE. Suppose that X is equipped with the norm $\|.\|$.

Theorem 5 ***(Liouville)***
Under the assumptions of the Cauchy-Lipschitz theorem, the evolution of $p_t(x)$ is given by[a]

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

a. NbJE. p and V are functions of (t, x), where x itself is a function of t. The partial derivative ∂t acts on the first variable of (t, x) (and not on the t of $x(t)$). The divergence operator $\nabla_x \bullet$ applied to a vector $A(x)$ is computed using the partial derivatives of the components of the vector $x = (x_1, \ldots, x_d)$ as follows: $\nabla_x \bullet A(x) = \sum_{i=1}^{d} \frac{\partial A_i}{\partial x_i}(x)$. This can be recognized as the dot product of $\nabla_x = (\partial/\partial x_1, \ldots, \partial/\partial x_d)$ and $A = (A_1, \ldots, A_d)$.

Before proving the theorem, let us analyze its implications:

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

The second term involves the gradient of the probability density $p_t(x)$, which is a vector with components $(\partial p_t(x)/\partial x_1, \ldots, \partial p_t(x)/\partial x_d)$ evaluated at $x(t)$. When substituted into the evolution equation, if we consider only this term, we recognize a **transport equation** for the probability density function, analogous to the equation that describes the transport of heat ($\propto T$) in a moving fluid:

$$\frac{\partial T}{\partial t} + \vec{v} \bullet \nabla_x T = 0.$$

Thus, this term tells us that **the probability density follows the velocity field**.

The first term, involving the divergence of V_t, corresponds to the **Jacobian term** of Normalizing Flows. When there is a gradient in the velocity field, it produces **compression or expansion effects** on the volume occupied by the particles. Consequently, once $V_t(x(t))$ is determined, the computation of the "Jacobian" naturally follows through the divergence.

Proof 5. The idea of the proof follows the evolution of a collection of particles $(x^{(i)})$ within a volume $\Omega \subset \Omega_x$, considering exchanges at its boundary. The conservation of probability will yield an equation similar to that in the context of an incompressible fluid.

Let $P_t(\Omega)$ be the probability associated with $x \in \Omega$:

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

The total time variation of this probability consists of both the local temporal variation at each point, expressed as

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

and a flux contribution. Define the quantity

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

which represents the **probability current density**. The flux of J through the boundary Σ of Ω is given by:

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

where dS is the surface element of Σ, and $n(x)$ is the outward normal vector at point x. Applying the divergence theorem, we obtain

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

The total probability variation is then given by the sum of these two contributions:

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

Since probability is conserved, we have $\frac{dP(\Omega)}{dt} = 0$ for any volume Ω, which implies the local conservation equation:

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

■

5.4 Stochastic Differential Equation (SDE)

The problem with the ODE approach (Eq. 87) is that it inherently requires perfect knowledge of the velocity field $V_t(x)$. However, in practice, there are errors, as even parameterized models $V_{\theta,t}$ are not perfect. Furthermore, the problem at hand may have an intrinsic random phenomenon. We will thus study stochastic differential equations of the type [77]

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

where W_t is a **Brownian motion**, meaning we add a **white noise term** to the ODE velocity term (all components are independent *Gaussian random variables*). The physical model that comes to mind is **diffusion** [78]. Thus, if we only retain the Brownian term, particles over time explore an ever-larger domain. If we add the velocity term, the trajectories will indeed follow the flow, but they **are no longer invertible**. The individual knowledge of the trajectories no longer makes sense.

However, we can have a probabilistic description of the problem (weak solution) by finding the evolution equation for the probability density that will replace the Liouville equation, and we will see that it is invertible. In fact, **the Wiener measure term will bring flexibility to the modeling of invertible transport on probability distributions**. Here is the evolution equation (1913):

Theorem 6 *(**Fokker-Planck**)*

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

77. NbJE. a_t is a shorthand for $a(t)$, and we use the notations from the Liouville theorem.
78. NbJE. The diffusion equation is of the form

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

with D being a coefficient expressed in $m^2.s^{-1}$. Typically, this gives Gaussian solutions $\phi(x,t) \propto e^{-x^2/(4Dt)}$, where there is a spreading of particle density over time. For example, for water vapor in the air (25°C, 1 atm), the diffusion coefficient is $D \approx 2 \times 10^{-5}\ m^2.s^{-1}$, so it takes about 3 minutes for a water vapor molecule to travel 1 meter in the air. In reality, one must also account for convection terms. But that is another story...

Compared to the Liouville equation, we have an additional term given by the Laplacian $\Delta_x = \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2}$. The probability current $J_t(x_t)$ (Eq. 100), which initially only included an **advection term**, will be modified by the **diffusion of particles**. The laws of diffusion in matter were established by Adolf Fick (1855) and are analogous to Joseph Fourier's heat laws, established in 1822. The material density current is proportional to the gradient of this density (with a negative sign). Particles move from regions of high density to regions of low density, a process of homogenization, which eventually leads to a steady state.

Theorem 7 ***(Fick)***

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

Thus, as is generally the case, noting $J^{tot} = J^{advection} + J^{diffusion}$, we obtain the following balance:

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

When we inject the expression for $J^{diffusion}$, and noting that $\nabla_x \bullet \nabla_x = \Delta_x$, we obtain the Fokker-Planck equation.

Fokker-Planck in 1D

Let us provide a justification for the 1D Fokker-Planck equation, which helps to make the various contributions more accessible. We will focus on ($V_t = 0$) the Brownian motion, which is characterized by stationary Gaussian increments, and the variance is proportional to time. The first step is to discretize the problem, meaning we approximate Brownian motion by a random walk, which is reminiscent of studies on Markov chains [79]. Over a time step Δt, a particle initially at x can make a jump ε, either in the positive or negative direction (e.g., $\varepsilon = \pm\Delta x$) with equal probability 1/2 (no global velocity). After N time steps, or after a duration $T = N\Delta t$, if the initial position is $x_0 = 0$, the particle will be at position x_T such that

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

79. NbJE. See Course 2024 Sec. 8.1, Course 2023 Sec. 6.4.2.2.

The $(\varepsilon_k)_{k\leq N}$ are independent *random variables* (Rademacher random variables). Thus, as N tends to infinity, we have the central limit theorem [80] which tells us that the distribution of x_T will become Gaussian. The variance is simply

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

Does this converge to Brownian motion? We want

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

Thus, we need to adjust the random walk accordingly.

Regarding the probability that the particle is at point $x(t+\Delta t)$, denoted $p(x = x(t+\Delta t))$ or $p(x, t+\Delta t)$, it decomposes as follows by considering the jumps of particles coming from the left and right of x:

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

A first-order Taylor expansion in t for the left-hand side, and a second-order expansion in x for the right-hand side (dominant terms as $\Delta x, \Delta t$ tend to 0), gives

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

This justifies the term of Fritz in the Fokker-Planck equation.

6. Lecture of Feb. 12th

The Fokker-Planck equation, partially established [81] in the previous session, is an ordinary differential equation (ODE) for the evolution of a probability density, associated with a stochastic differential equation (SDE) for the motion of a particle. This ODE

80. NbJE. Course 2022 Sec. 5.5 Th 10.

81. NbJE. S. Mallat revisits the proof at the beginning of this session by changing the notation, which I compiled at the end of the previous session.

on $p_t(x)$ has an important consequence, namely its sampling by a Langevin diffusion. We will address this method before moving on to score-diffusion algorithms, which introduce a modification of the noise level (sometimes associated with a modification of the temperature) and yield the best-performing results at the moment.

6.1 Sampling with the Langevin Equation

6.1.1 Finding the Velocity and Variance

We are considering a Gibbs probability framework (Sec. 3.1) written in the form

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

where β, reminiscent of the Boltzmann factor, is the inverse of a sort of temperature ($\beta = 1/T$). Decreasing β corresponds to increasing the temperature, i.e., increasing thermal agitation. We would like to obtain typical samples [82] from this *pdf*. These samples will tend to correspond to the energy minimum (U).

Let $z = x_0 \sim p_0$ be an initial point where p_0 can be any *a priori* distribution (e.g., a normal distribution). We need to define a transport mechanism to obtain a sample from $p(x)$. This transport is continuous, indexed by time t. We seek an SDE that enables this transport (σ is a constant here):

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

The question is how to adjust V_t and σ so that as t evolves, we end up evolving within the support of $p(x)$ (like a Markov chain). In fact, we would like $p(x)$ **to be an invariant of the transport**. To do this, we will impose the **stationarity** of the Fokker-Planck equation for p:

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

Let us recall that for the samples $x \sim p$, $U(x)$ should be small, so if we proceed with a gradient descent evolution, the direction of evolution of the x_t is fixed (opposite of the

82. NbJE. Example: Course 2022 Sec. 6.4

gradient), hence a good candidate for $V_t(x)$ is

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

Thus, since $\nabla_x \log p(x) = (\nabla_x p(x))/p(x)$ and $\nabla_x \bullet \nabla_x = \Delta_x$, we get

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

Thus, the problem of sampling $p(x)$ can be reformulated according to the SDE/ODE system of evolution for $x_t = x(t)$ and $p_t(x_t) = p(t, x(t))$ over time ($t = 0$, $x(0) = x_0$ and $p(0, x) = p_0(x)$):

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

with [83] the score of p written as $s(x) = \nabla_x \log p(x) = -\beta \nabla_x U(x)$.

6.1.2 Convergence

The questions are whether, starting from an initial condition (p_0), we converge to $p(x)$ (the invariant of the transport) and whether the convergence is unique. This can be proven with relatively weak conditions, notably when $U(x)$ **tends to infinity as** $\|x\|$ **tends to infinity**, meaning the probability becomes very small at the boundaries, leading to a kind of confinement in low-energy regions. It is generally shown that [84]

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

and if we add the constraint on $U(x)$, then **there is convergence of** p_0 **towards** p**, but it may occur at a very slow rate**.

83. NbJE. Note that this refers to the score of the *pdf* p, so no t index.

84. NbJE. For example, A. R. Plastino et al. (1997), *Minimum Kullback entropy approach to the Fokker-Planck equation*, Phys. Rev. E 56, 3927, https://www.researchgate.net/publication/235474425_Minimum_Kullback_entropy_approach_to_the_Fokker-Planck_equation

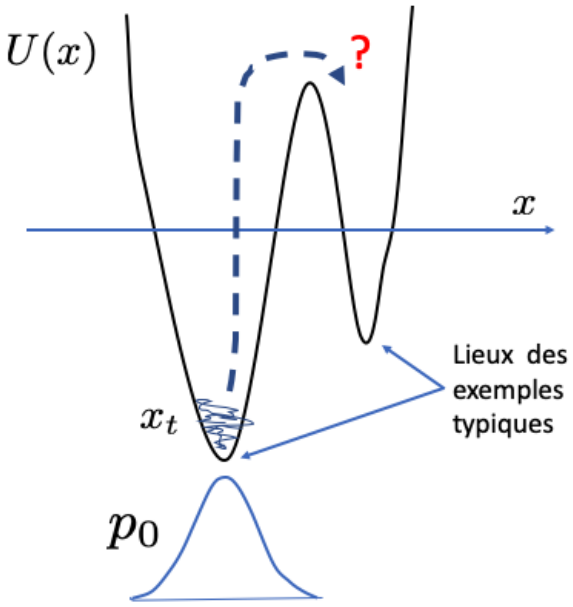


FIGURE 32 – Case of an energy $U(x)$ with two modes and an initial distribution p_0 centered on the deepest mode. How to jump the potential barrier?

Why? We can visualize the following scenario (Fig. 32) where $U(x)$ has 2 modes, one deeper than the other: if p_0 is concentrated around the deeper mode, the x_t are confined to this mode because the velocity term of the ODE is almost zero, and only the Brownian term matters. In order to explore the entire profile of $U(x)$ and especially the second minimum, it is necessary to make a jump to pass over the potential barrier. However, this can only happen if the Brownian term has a large variance (β^{-1} large), i.e., at high temperature. This leads to strong noise in the gradient descent, resulting in x_t trajectories with large variances, which prevents them from concentrating in the typical regions of $p(x)$. Ultimately, **there is no good choice for the factor β (or the temperature) of the Brownian motion**.

What has been done experimentally is to perform what is called simulated annealing. That is to say, to change the temperature (or β), and thus the term σ of the Brownian motion, as a function of time:

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

At the beginning, we take a high temperature to allow the exploration of trajectories, and then gradually lower the temperature. We also use replica exchange (see discussion in Section 3.2) of Langevin chains obtained with different values of β_0. These are fairly complex algorithms that require **adjusting the temperature to both properly sample the typical regions while allowing the exploration of the different modes**.

6.1.3 Euler-Maruyama Discretization

Finally, in practice, we need to discretize the differential equations. In particular, by using the **Euler-Maruyama discretization** [85] (σ is time-independent here):

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

The discretization introduces an error, so the transport of p_0 does not perfectly correspond to p, or in other words, the samples x_t explore a bit around the typical regions of $p(x)$. That being said, there are correction algorithms like the Metropolis-adjusted Langevin algorithm (known as `MALA`), but they are quite slow when the energy is not convex. Nevertheless, these algorithms have been used to sample Gibbs probabilities.

6.2 Generation by Score Diffusion

One thing that we retain from the **Langevin diffusion** method is that in the transport leading to p, **the initial condition p_0 is forgotten** due to the Brownian motion. However, this comes with **slow convergence**, and we must be able to calculate (Eq. 117) $\nabla_x \log p(x)$, that is, we need to **have the score** of $p(x)$. This score poses two problems: 1) we need to know $U(x)$, and 2) more critically, the typical locations of $p(x)$ form a manifold of $\mathbb{R}^d$, so the gradient has singularities, making its **computation highly unstable**. This issue is particularly encountered in the case of images.

Score-diffusion algorithms will address these problems by a **regularization** of p. In general, a simple way to perform such a regularization of a function is to smooth it by

85. Gisiro Maruyama (1916-86). NbJE. An example of solving the Ornstein-Uhlenbeck equation using this discretization technique is provided in a notebook created for the 2024 course and updated [86].

convolution with a Gaussian. The convolution acts in the probability space; what about the x_t side? In fact, this amounts to **adding white noise**. Thus, these algorithms are focused on adding noise to regularize the probabilities. This allows us to compute the transport of data smoothed by noise, towards samples from p.

6.2.1 Ornstein-Uhlenbeck Equation

NbJE. The study was also covered in the 2024 course, Sec. 9.1, see also the footnote 23.

We will approach the diffusion methodology with Normalizing Flows (Sec. 4.5) in mind, where we have seen that the transports T are invertible (see Figure 8). More specifically, we have seen (Sec. 4.5) that it is natural to construct T from T^{-1}, that is, to start from the data of p to transform them into samples from p_0, a much simpler distribution (e.g., white noise). This is the part of the process known as **forward diffusion**, and one can refer to Figure 10 to get an idea. Starting from a distribution p_0 (here, it is p_{data} or p), whose x_0 is a sample, adding noise is done using the **Ornstein-Uhlenbeck** equation

Theorem 8

Let the following Ornstein-Uhlenbeck equation be given:

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

The solution is given by:

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

Proof 8.

The proof proceeds by a change of variables $y_t = e^{\kappa t} x_t$ which satisfies the white noise equation

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

The solution y_t therefore involves an integral (sum) over increments $dW(u)$ of the Brownian motion, which are independent Gaussian *r.v.s* ($dW(u) \sim \mathcal{N}(0,1)$). The result (cf. the integral) is thus a *Gaussian r.v.*. What is its variance [87]?

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

Thus,

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

which gives us the solution when we make the inverse change of variables $x_t = e^{-\kappa t} y_t$. **We notice the "forgetting of the initial distribution" and the convergence happens exponentially towards the white Gaussian noise of variance $\sigma/\sqrt{2\kappa}$.**

Next, we take the following set of parameters:

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

to obtain the formulation of Eq 14. ■

What about the expression of $p_t(x_t)$? To answer that, we need to recall two results: let the *r.v.s* (X, Z, X_1, X_2) with their corresponding *pdf*s and α a real, we have

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

Thus, starting from equation 8, the expression for p_t is given by [88]

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

with $g_{\sigma_t} = \mathcal{N}(0, \sigma_t^2)$ and $\sigma_t^2 = 1 - e^{-2t}$. **This corresponds to a compression of p_0 followed by a Gaussian smoothing**. We could also have started from p_0 and convolved it with a dilated Gaussian, which well represents the dilution of the initial images into white noise.

87. NbJE. Think of $z = \sum_i a_i z_i$ with $z_i \sim \mathcal{N}(0,1)$: the mean is zero, and the variance is $\sum_i a_i^2$.
88. NbJE. $p_t(x) = \int_{-\infty}^{\infty} e^t \, p_0(e^t \, u) g_{\sigma_t}(x-u) du = \int_{-\infty}^{\infty} p_0(v) g_{\sigma_t}(x - e^{-t} v) dv$.

This process erases all the details of the initial distribution. Conversely, we can recover the initial distribution p_0 starting from a distribution at time t, that is, p_t.

6.2.2 Inverse Diffusion Equation

What is the expression for the transport T that acts during the generation of samples x from $p(x)$ starting from samples of the distribution $p_z(z) = p_0(z)$ ($p_z = \mathcal{N}(0, Id)$)? To do so, we need to invert T^{-1}. One can refer to Figure 10 (backward/inverse). Starting [89] from $x_0 \sim p_0$ by diffusion (T^{-1}), after a sufficiently long time T, we obtain $x_T \approx x_\infty$, which we consider to be entirely dominated by the noise term, meaning that it is a sample from $\mathcal{N}(0, Id)$. The operation T, on the other hand, starts from a white noise sample to obtain a sample from $p(x)$.

Let $y_t = x_{T-t}$ ($t \in [0, T]$), we would like y_0 to be a sample from $p(x)$. Recall that we cannot ask for $y_0 = x_0$, because the Brownian process is irreversible. Since we are reasoning about probability distributions, we need the Fokker-Planck equation (Th. 6). In our case with normalized coefficients, the evolution equation for x_t reads (note that here p_0 is the *pdf* of the data distribution p_{data} or p, while p_T is the *pdf* of white noise):

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

which is associated with

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

If we denote q_t as the probability distribution associated with y_t, we would like $q_t = p_{T-t}$, so the evolution equation becomes (note that here q_0 is the *pdf* of white noise, and q_T is the distribution of the data p_{data} or p):

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

If we want to translate this Fokker-Planck equation in terms of the evolution of y_t, the first term (*drift*) has the opposite sign, which is not an issue, **the only problem comes from the sign change in the second term ($-\Delta$)**. This term would tend to reverse the

89. NbJE. Note that $p_0 = p_{data}$.

Brownian process, but how can we reconstruct a sharp image (e.g., a face) from noise? Let's use an example in signal processing: if we convolve a function f with a Gaussian g, we can see the result in the Fourier domain:

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

Now, $\hat{g}(\omega)$ is a Gaussian, so convolution significantly attenuates the high frequencies of f, and even though $\hat{g}(\omega)$ is not zero, inversion becomes very unstable at high frequency. Therefore, we might think that recovering f is nearly impossible. This is our *a priori* case when trying to obtain the second term in the evolution equation of y_t, are we at an impasse? What will save us is that we want an inversion on average, particle by particle. The study of inverse diffusion was done by Brian D.O. Anderson in 1982 [90]

Theorem 9 *(**reverse diffusion, Anderson 1982**)*
The equation governing $y_t = x_{T-t}$ is given by

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

Proof 9.
Let's try the following: replace the term $-\Delta_y q_t(y_t)$ according to

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

Thus, the Fokker-Planck equation can be written as

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

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

with a modified *drift* term, and a new variance

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

Now that we have a $+\Delta_y$ term, the evolution equation of y_t can be written by analogy with the case of dx_t as

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

Note that Figure 10 shows the case where $\alpha = 1$ with also a different definition of the direction of the arrow of time, which can vary from one publication to another [91].

Note that **the modified drift term (V_t) allows to guide the trajectories of the particles** so that they adopt the correct distribution p_{data} when $t = T$. This is Langevin's idea of **giving the velocity the component of the score** (Sec. 6.1.1). ■

Since 1982, the community knew that one could invert a diffusion equation, but the main challenge lay in the calculation of the **score**. Note that it is the score of $q_t = p_{T-t}$ and not that of $p = p_{data}$, and thus, it is somewhat easier, since it has been **regularized by a Gaussian**. However, the problem remains that we do not know the distribution p. It is in this context that Yang Song [92] and colleagues in 2021 provided the missing piece by allowing to **compute the score using a deep neural network**. However, it requires **a lot of data**. S. Mallat tells us that this result was a huge surprise, because the score is a function in very high dimensions. The technique used is that of **denoising**.

6.2.3 The Use of Denoising

The first thing we perform is the renormalization of the variables x_t in the Ornstein-Uhlenbeck equation (Th. 8) to eliminate the factor $e^{-\kappa t}$. *NbJE. We take the definition of*

91. NbJE. In the slide presented by S. Mallat, the definition is the one taken in this paragraph, with B replacing W and y_t appearing in its form x_{T-t}.

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

y_t from the proof of the theorem, and we denote it as x_{σ_t}. So, by taking the equation 127, we can write (x_0 is the original image, which we denote as x without dependence on t):

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

Next, we use the normalized parameters (Eq. 128), which give the expression for the standard deviation of the noise z, $\sigma_t = \sqrt{e^{2t} - 1}$. The random variable x_{σ_t} has a probability density denoted $p_{\sigma_t}(x_{\sigma_t})$.

We then find ourselves with a classical problem [93] of a variable x corrupted by white noise. The problem we are tackling is to find an **estimator $\hat{x}(x_\sigma)$ of** x that removes the noise as effectively as possible. One way to find this estimator is by using the **mean squared error**

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

In this context, there is a result, rediscovered and reformulated several times, by M. C. K. Tweedie in 1947, H. Robbins in 1956 who acknowledged Tweedie's work, K. Miyasawa in 1961, and more recently by M. Raphan and E. P. Simoncelli in 2011 [94]. This result is as follows:

Theorem 10
The optimal estimator is given by

$$\hat{x}(x_{\sigma_t}) = x_{\sigma_t} + \sigma_t^2 \, \nabla_{x_{\sigma_t}} \log p_{\sigma_t}(x_{\sigma_t}) \tag{142}$$

This means that the optimal denoising is obtained by guiding the gradient descent using the score of the probability distribution of the noisy signal.

Proof 10.
To simplify, we remove the time index t. Since $x_\sigma = x + z$ is the sum of two independent

93. NbJE. Also see Course 2024 Sec. 9.2, Course 2021 Sec. 2.2.1 (Figure 3).
94. NbJE. See for example https://www.cns.nyu.edu/pub/lcv/raphan07b.pdf.

r.v.s, its probability density is the convolution of p_x and p_z

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

Since p_z is a Gaussian with zero mean and variance σ^2, we get

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

Now, notice that $p_z(x_\sigma - x)$ is nothing other than the conditional probability $p(x_\sigma|x)$. Thus, by denoting $p(x, x_\sigma)$ as the joint probability of x and x_σ, we get

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

Since $p(x, x_\sigma) = p(x|x_\sigma) p(x_\sigma)$, we obtain

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

Furthermore, if we seek a constant μ that estimates x as well as possible in quadratic error, then

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

Thus, for a fixed x_σ, we can identify $\mathbb{E}_x[x|x_\sigma]$ with $\hat{x}(x_\sigma)$. ■

Thus, **to compute the score, we need to obtain an optimal denoiser**. For this, we use a deep neural network such as the (U-Net) shown in Figure 11. Considering this network, the input data consists of noisy images x_σ for which we know the original images x, and by using the cost function $\|x - x_\sigma\|^2$ to minimize, we obtain the optimal estimator $\hat{x}$. Finally, $(\hat{x} - x)/\sigma$ provides the score of p_σ. **However, we need a lot of images x and noise realizations**[95] **and moreover, we have no guarantee that the network optimization will**

95. NbJE. In the version presented by S. Mallat, the network does not know the noise level in the image x_σ. In the study by Y. Song, the network is conditioned by σ_t, since there is a noise evolution scheme over time. You can see this dependency in the notebook `JAX_blob_diffusion.ipynb` from this year's course.

lead to the optimal denoiser.

And indeed, it works [96]! The procedure is as follows: 1) obtain an image denoising network from face or bedroom data, etc., 2) then, starting from a noise realization, perform the reverse-diffusion evolution (Th. 9). This gives a sample from a "probability" of face images, bedroom images, etc., even though we don't know exactly what that means.

However, there is a mystery, because we are trying to estimate the **score, which is a function in very high dimensions**, and let us remind you that it had singularities which we regularized by convolution with a Gaussian. For this to work, we must **overcome the curse of dimensionality**, which is possible only if this function has **regularities**. The observation is that this seems to be the case. Now, what we can test is whether the generated samples are clever mixtures of images from the database, or whether they are independent samples from the database? In other words, is there **memorization or generalization?** In the upcoming sessions, we will explore this and examine the architecture of U-Net to understand why it may constitute a good denoiser.

7. Lecture of Feb. 26th

From this session onwards, we will focus on the notion of **optimal denoising** and the mathematics developed in this field. Let us first revisit the score-diffusion algorithm to understand where the notion of denoising comes into play.

7.1 Score-Diffusion Transport

7.1.1 Diffusion and Inversion

As a reminder, we first defined [97] the transport $\boldsymbol{T}^{-1}$, which maps a sample $x \in \Omega_x$ from the data space to a noise sample $z \in \Omega_z$ (Fig. 23). This transport is performed using

96. NbJE. See footnote 92

97. NbJE. I take the liberty of writing the transport operators in bold here, knowing that a time T will appear later. Moreover, this notation ambiguity is already present in the previous sections.

the Ornstein-Uhlenbeck equation (Th. 8), which, for reference, is written (with normalized parameters Eq. 128):

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

which means that noise is progressively added to the signal. The solution at any time t is expressed as

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

where the initial component x_0 is gradually attenuated in favor of a pure noise component (the "Diffusion" part in Fig. 10).

The question we then addressed was: can we invert $\boldsymbol{T}^{-1}$ to obtain the transport $\boldsymbol{T}$, which enables the generation of new samples from the data *pdf*? The answer is yes, thanks to Anderson's theorem (Th. 9). If we denote by T the time at which the diffusion process is considered representative of the infinite-time solution, and if we consider the evolution of x_{T-t} for the inverse process [98]

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

As a reminder, the score $\nabla_{x_{T-t}} \log p_{T-t}(x_{T-t})$ appears when attempting to invert diffusion in the Fokker-Planck equation, in order to change the sign of the Laplacian (Eq. **??**). Once the score is known, the generation of new samples x becomes possible (the "Inversion" part in Fig. 10).

7.1.2 Obtaining the Score: Denoising

The central point is therefore to obtain the **score**, and the fact that this can be achieved using deep networks has been a major breakthrough. This is done through a denoising procedure that we introduced in Section 6.2.3. It relies on Theorem 10, which we rewrite in a renormalized form [99]. If a signal x is corrupted by noise z (independent of x) according to $x_\sigma = x + z$ with $z \overset{iid}{\sim} \mathcal{N}(0, \sigma^2\, Id)$, its optimal estimator in the mean

98. NbJE. In the previous session, we used the notations y_t and q_t for x_{T-t} and p_{T-t}. Moreover, $\alpha = 1$ in the case considered here.

99. $x_{\sigma_t} \leftarrow x_t e^t$, $x \leftarrow x_0$, $\sigma \leftarrow \sqrt{e^{2t} - 1}$.

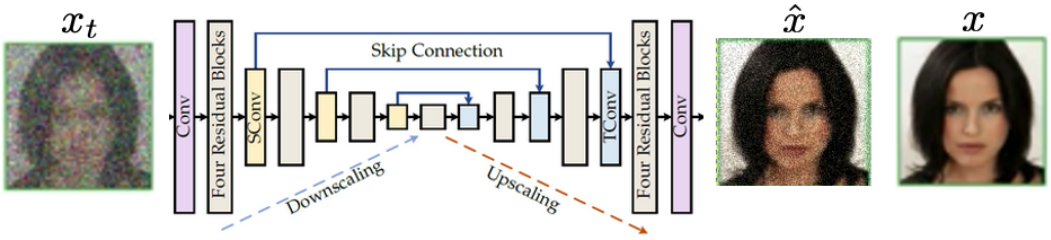


FIGURE 33 – U-Net network used as a denoiser.

squared error sense, denoted $\hat{x}(x_\sigma)$, is related to the score as follows:

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

Thus, we need, for example, a neural network like the one [100] shown in Fig. 33. In a first approach, the network is fed with data of the form (x_σ, σ), meaning noisy data along with their individual noise level, and the output is $f_\theta(x_\sigma)$. The loss function [101] is simply the mean squared error (MSE):

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

Minimization is performed using a standard gradient descent method (SGD, Adam, etc.). If optimization achieves the best possible θ, denoted θ^*, then we can identify $f_{\theta^*}(x_\sigma)$ with $\hat{x}(x_\sigma)$, leading to:

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

7.1.3 The Case of a Noisy Gaussian Process

Let us consider the case where x follows a Gaussian distribution with covariance $\boldsymbol{C}$, meaning that

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

100. NbJE. This corresponds to Fig. 11, reproduced here for convenience.
101. NbJE. Alternatively, the network can be trained to learn the "noise" instead of the "signal," meaning that $f_\theta(x_\sigma)$ should recover z. This corresponds to a residual learning technique.

Then x_σ is also Gaussian, with covariance $\boldsymbol{C}_\sigma = \boldsymbol{C} + \sigma^2 Id$ (recall that x and z are independent). Thus,

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

In this case, the score associated with $p(x_\sigma)$ is given by

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

And *in fine*, **the optimal estimator for a Gaussian process contaminated by Gaussian noise** is given by the following expression:

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

We will now derive this formula using an alternative approach.

7.1.4 Score Estimation Error

Now, if the optimization is not perfect and the score estimation is therefore imperfect as well, the question arises: will this **error in the score** induce a significant error in the probability distribution obtained by the inverse diffusion algorithm or not? If we denote by $s_\theta(x_\sigma)$ the score obtained from the denoising network, we have:

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

What we would like to control is the divergence $D_{KL}(p\|p_\theta)$ between the true data distribution p and the one obtained through inverse diffusion, denoted here as p_θ. We know that if $s_\theta(x_\sigma)$ exactly matches the true score, then the operator $\boldsymbol{T}^{-1}$ is correctly inverted, and the divergence is zero. In this framework, there is a theorem by Y. Song et al. [102] which states that, under certain assumptions,

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

102. NbJE. Yang Song et al. (2021), *Maximum Likelihood Training of Score-Based Diffusion Models, arxiv:2101.09258. It should be adapted to the case where $dx_t = g(t)dw_t$, in which case the solution is expressed as $x_t = x_0 + \sigma_t\, z$ with $z \sim \mathcal{N}(0,1)$ and $\sigma_t^2 = \int_0^t g(u)^2 du$, so that the factor $1/2g(t)^2 dt$ in Y. Song's theorem transforms into $\sigma_t d\sigma_t$. We have also omitted the index t, which is not relevant here.*

This tells us that if the score error is small and integrates well, then **we can estimate the error in the distribution obtained through inverse diffusion**. Other bounds are available using the Wasserstein distance (W_2). This is a key difference compared to GANs, where there is no clear way to control errors in the *pdf*s.

That said, the challenge remains to estimate the score with an error that is not too large. In fact, one can identify two types of errors that correspond to the **classical errors of an estimator** [103].

1. **Variance/Generalization**: The first type of error relates to a **variance problem**, or a trade-off between **generalization** and **memorization**. The key question here is: how dependent is θ^* (the network parameters) on the training dataset? If we change the training batch (while preserving the underlying *pdf*), do we obtain the same θ^* and thus the same score s_θ, or not? If we expect numerical differences, the question of the variance of the estimator s_{θ^*} arises. Intuitively, if the variance is too high, it suggests that the estimator is strongly dependent on the training dataset. This would indicate that image generation is influenced by *memorization* of the training set. Conversely, if the estimator is robust to changes in the training set, we gain confidence that it exhibits *generalization*, meaning that the generated samples are genuinely new representatives of p_{data}.
2. **Bias/Model Error**: The second type of error arises when variance is negligible (i.e., the estimator remains invariant across different training batches). The question then becomes whether the estimator s_{θ^*} truly matches the correct score. In other words, even if the model is well-optimized and yields the same θ^* across different training batches, it may still provide an incorrect score estimate. This issue is driven by two factors: *first*, the **network architecture**, which must be sufficiently flexible to model complex transport processes; and *second*, the **optimization process**, which must correctly converge to the optimal θ^* (i.e., the loss function must truly reach its minimum).

We will address these questions using a specific neural network: the **U-Net** [104], which is illustrated in more detail in Figure 34. Originally, it was designed for segmentation

103. NbJE. See Course 2018 Sec. 2.3
104. NbJE. Architecture introduced in Olaf Ronneberger et al. (2015), *U-Net: Convolutional Networks for Biomedical Image Segmentation*, *arxiv:1505.04597*

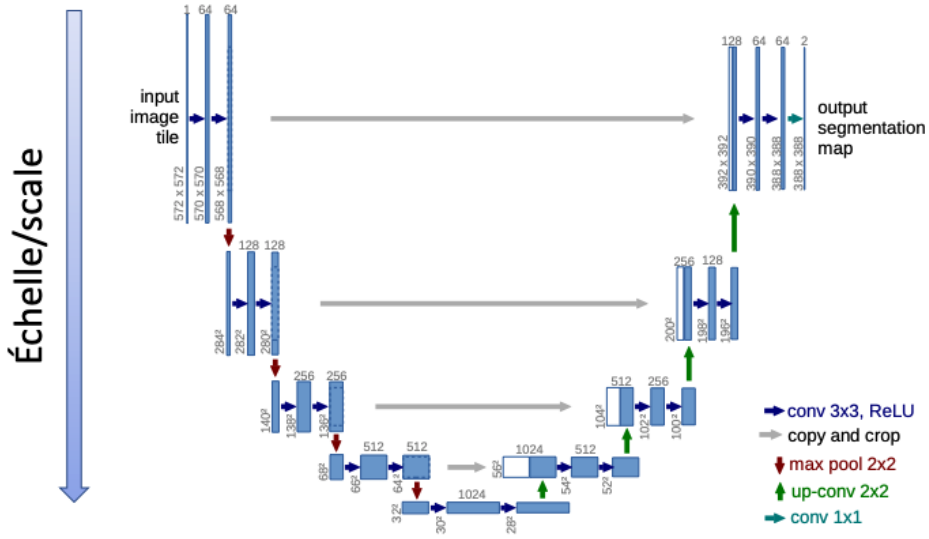


FIGURE 34 – Multi-scale architecture of a U-Net.

tasks in the biomedical field (e.g., recognizing neuronal structures in images obtained through electron microscopy). Later, it was applied to many other tasks due to its unique characteristics.

First, the input and output images have the same dimensions ($H \times W \times C$), making it well-suited for a denoiser. Second, its architecture consists of two phases. The first phase acts as an *encoder*, compressing the information by applying a cascade of operations (3×3 convolutions, ReLU, BatchNorm) interleaved with downsampling steps. The second phase functions as a *decoder*, decompressing the information through a cascade of "inverse" operations (upsampling, deconvolution, ReLU, etc.), while incorporating the corresponding outputs from the *encoder* at each step (represented by the horizontal arrows in the figure). This structure enables **multi-scale interaction modeling**.

That being said, why are the different operations (convolution, non-linearity, normalization) important, and why is the concept of scale (downsampling/upsampling) just as crucial? These questions have already been discussed in previous courses [105]. The underlying question is: **why can a U-Net learn a function as complex as the score in high dimensions?**

105. NbJE e.g., Course 2020 Sec 9.2

This is the major question, as noted by S. Mallat, that we still do not fully understand from a mathematical perspective. However, in the specific case of **denoising and the optimal estimator in terms of quadratic error**, this problem has been studied since the 1940s, and there exists a solid **mathematical foundation on the subject**. Therefore, we can bridge the gap between mathematics on one side and the solutions discovered by neural networks on the other. In a way, this framework serves as a **valuable laboratory to gain deeper insights** into why a U-Net can be effective.

7.1.5 Feedback from Numerical Experiments

NbJE. Before diving into the topic of denoising, S. Mallat revisits the issue of variance in the score estimator, i.e., the problem of generalization vs. memorization. This is a point he previously raised in Section 2.8.2, particularly in his discussion of Figure 13, which he brings up again on screen.

The key takeaway from the numerical experiments conducted by Kadkhodaie et al. is that if the number of training images provided to the two U-Net networks is not sufficiently large, then the inverse diffusion trajectories, starting from the same white noise, do not produce the same generated image. Moreover, the smaller the training sets, the more the image generated by each U-Net clearly resembles one from its own training set.

Beyond a certain threshold [106], around $O(10^5)$ for 80×80 images, **the two inverse diffusion trajectories (still starting from the same white noise) produce exactly the same image, independently of the training sets, and this image does not belong to either of the two training sets.** This clearly indicates a threshold marking a **transition from a memorization regime to a generalization regime**.

This result is significant in two ways. First, it confirms that **a generalization regime can indeed be achieved**, rather than mere memorization, which was a common criticism against the early results of score-based generative diffusion models. Secondly, it shows that in order to reach this regime, **a large amount of data is required**. For instance, when

106. NbJE. I have made available on the GitHub site the notebook `JAX_FLAX_NNX_UniversalDen_MNIST_jaxjit.ipynb`, which you can modify/adapt to perform this type of numerical experiment.

querying an LLM with an overly specific question on a topic with little textual information available online, it is likely to operate in a memorization regime. Conversely, if the question pertains to a broad corpus, the response is generated within a generalization framework.

S. Mallat presents additional examples (Fig. 35) generated by the two U-Nets trained on datasets of $N = 10^5$ images (generalization regime), where only the random seed used to generate the initial white noise differs while remaining the same for both networks. While nearly all generated images are identical, an interesting exception occurs in the last example ("the glasses"). This is an intriguing phenomenon known as a **bifurcation**, where a small error in the score leads to a significant effect on the reconstructed image. Interestingly, the generated image does not contain fragmented glasses but rather exhibits a *binary all-or-nothing* effect. Such phenomena can be understood through Gaussian mixture models.

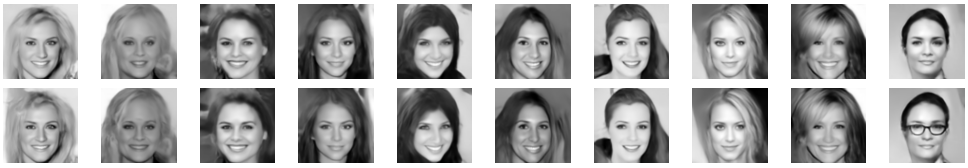


FIGURE 35 – Additional examples generated with $N = 10^5$. Samples generated by each U-Net are presented in separate rows, with each column corresponding to the same initialization for both networks. Extracted from the reference article in footnote 28.

Of course, one can change the type of dataset and rerun the numerical experiment—the conclusions remain the same. **The threshold depends on the image size**. The number of parameters in the networks used in these experiments was 7.6 million, and with an image dataset of 80×80, generalization is achieved when there are approximately 10^2 data points (pixels) per parameter. What happens for larger or smaller images? The scaling law appears to remain unchanged [107]. However, that being said, we cannot attach too much certainty to such considerations, as non-linear phenomena, redundancies, etc., exist within the network architecture. Ultimately, the relationship between sample size and network size at the threshold of generalization remains difficult to grasp. Nevertheless,

107. NbJE. For example, with MNIST, which consists of $60,000$ images of 784 pixels each, one could expect to reach a generalization regime with a model having around $5\,10^5$ parameters. The model in `JAX_FLAX_NNX_UniversalDen_MNIST_jaxjit.ipynb` has $8\,10^5$ parameters, which may be slightly too large—you be the judge.

it seems to follow a fairly classical pattern: if we want a well-converging parameterized estimator, **we need far more data than parameters**.

Thus, while very large models can generate impressive images, one must be cautious of memorization! This is an important consideration, S. Mallat reminds us, especially when simulating fluid dynamics for studying an aircraft wing's response or making medical predictions. In such cases, the priority should be ensuring that there is a sufficient amount of data to guarantee diversity in the generated samples.

Another way to test generalization is by **computing the mean squared error** between the original and denoised images. A common metric used is the *PSNR* (Peak Signal-to-Noise Ratio), defined by the formula:

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

where $d = 255$ if pixel values are encoded as 8-bit integers. The smaller the error, the higher the PSNR. One can analyze how the output PSNR evolves based on various factors: the origin of the image batch (*training* vs. *test*), the training set size, and the amount of injected noise, with the input PSNR being proportional to $\log(1/\sigma)$. The results obtained for bedroom images are shown in Figure 36.

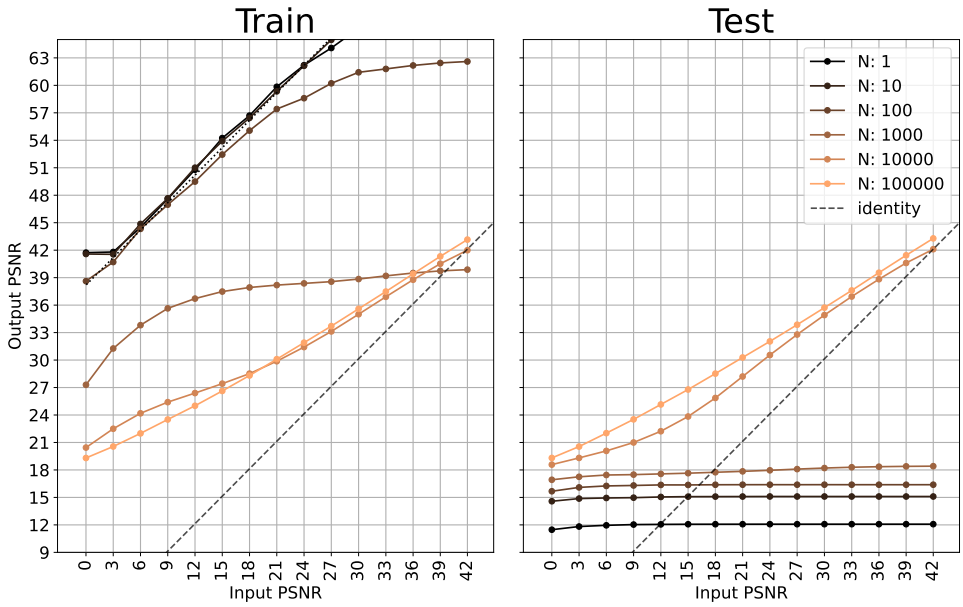


FIGURE 36 – Each curve shows the denoising error (output PSNR) as a function of the noise level (input PSNR), for a training set of bedroom images of size N. Extracted from the reference article in footnote 28.

When N is small (dark curves), the *Training* and *Test* curves are entirely different, revealing clear signs of **overfitting**. In contrast, when $N = 10^5$, the light orange *Training* and *Test* curves are nearly identical, indicating clear **generalization**.

Once we establish that the network is in the *generalization* regime, what about *bias*? Looking closely at the bedroom images generated by U-Nets trained on datasets of size $N = 10^5$, we notice some defects, even though the resolution of 80×80 is relatively low (Fig. 37).

FIGURE 37 – Close-up of a bedroom image generated by a U-Net trained with datasets of size $N = 10^5$. Extracted from the reference article in footnote 28.

It becomes evident that a bedroom image is quite complex, so it is understandable that the network struggles to generate a high-quality one. More data and larger models are likely required.

Finally, the remaining open question is: even if we have tests confirming the small variance of the estimator, what about the coverage of all modes of p_{data}? As a reminder, we previously discussed *mode collapse* phenomena in GANs (Sec. 2.7). This type of **lack-of-diversity phenomenon is not controlled by the variance of the estimator**. Ideally, we would like

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

meaning that the model correctly reproduces the true score—in other words, **we need an absence of bias**. However, verifying this is significantly more **challenging**, for the simple reason that **we would need to know the true score**. While there are simple cases (e.g., Gaussian processes) where the score expression is known, the same cannot be said for bedroom images.

Thus, we need to delve into the mathematical framework for computing the estimator (denoiser) to understand under what conditions the optimal estimator is obtained. This would ensure that the score is accurately estimated, ultimately guaranteeing a proper estimation of p_{data}.

7.2 Optimal Denoising: A Mathematical Overview

This opens a new mathematical chapter; however, we will see that it builds upon the framework of **sparse representations** of signals, as studied in the 2021 course. The questions we aim to answer include: Can a neural network perform optimal denoising? Under what conditions, and how? If we can answer these questions, it would imply that neural network theory is well understood. We would then know which classes of functions can be approximated by a network. At present, we do not have a definitive answer. The goal of this course is to move in that direction, providing the mathematical tools currently available to advance along this path.

In this mathematical chapter, we will cover:

— First, the topic of **optimal linear approximation**[108]: if we impose that the optimal estimator is a linear operator, how can it be computed? This leads to **Wiener filtering**. We will then explore its connection to **PCA bases**, and in the case of **stationary phenomena**, the concept of **convolution** will emerge, a key component in neural networks and closely linked to **Fourier bases** (power spectrum analysis). Ultimately, this falls within the domain of **Harmonic Analysis**.

— Second, we will explore **non-linear approximation** to investigate whether the estimator can be improved and under what conditions. This introduces the concept of **sparsity** in a basis—meaning that in a signal decomposition within this basis, most coefficients are zero. Which basis (or bases) does this refer to? Fourier must be replaced. This domain leads to **Wavelet bases** and the notion of a **dyadic scale** (downsampling and upsampling, as seen in U-Net architectures). We will also see that the non-linearity **ReLU** is a highly natural operator for denoising within this framework. The remaining question concerns **optimality**. The underlying issue is whether the problem at hand falls within the class of accessible functions. For instance, in *Besov spaces*, optimality can be achieved. This requires understanding the **regularity of the underlying process**. In particular, when dealing with complex geometries, such as facial images or bedroom scenes, complications arise.

— This leads us to the topic of **Geometry**, specifically the geometry of images with structured contours. In such cases, it is no longer possible to find a single basis that

108. NbJE. A note on terminology *linear vs. non-linear*; see 2021 Course Sec. 3.2.

provides an optimally sparse representation. Instead, we must turn to **families of bases**. According to S. Mallat, this was the state of research around 2005. He describes how **models became increasingly complex**, leading to some discouragement in the community. Despite the increasing model complexity, **numerical results did not meet expectations**. While simple images could be handled, real-world images posed significant challenges.

— **What happens in neural networks**? They have the ability to compute **harmonic bases** (which oscillate) while incorporating **geometry**. In cases where contour regularity is controlled, the bases found by neural networks correspond to the expected optimal bases. Thus, **we have complex cases (beyond Gaussian processes) where the optimality of network-discovered bases can be demonstrated**.

Let us now begin this program with the fundamental concept: Wiener filtering.

7.3 Wiener Filtering, PCA

This is a fundamental result in signal processing, established by Norbert Wiener (1894–1964), which provides a clear framework for understanding denoising using linear operators and justifies the use of convolutions. ***Notation****: all random variables will be written in uppercase letters, while deterministic quantities will be in lowercase.* Consider the classic problem of additive noise, modeled as:

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

where Z is Gaussian white noise (gwn) with zero mean and covariance $\sigma^2\, Id$, and it is **independent of the signal** X. We assume a finite-dimensional setting.

We want, as before, to compute an estimator $\hat{X}$ obtained by applying an operator H (deterministic nb.) on the noisy signal Y. Thus,

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

To obtain the expression of H, we use the minimization of the mean squared error

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

with the constraint that H belongs to a parameterized family of operators

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

We then impose that H **be linear**. If the representation of H is a matrix of dimension d, θ represents the d^2 elements of this matrix.

So, what is the best H and how is it calculated? *S. Mallat discusses the various fields to which N. Wiener laid the mathematical foundations*[109]*, particularly in Harmonic Analysis, stochastic processes, signal processing, control, cybernetics, etc.*

Here is the fundamental theorem:

Theorem 11

Let d measurements $Y(k)_k$, the linear estimator of the underlying signal X is given by

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

which minimizes the mean squared error, iff it performs decorrelation with respect to the data, that is to say

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

Proof 11. Let the squared error be

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

This is a quadratic form in h, and its minimum can be calculated by setting the derivative to zero:

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

Thus, the condition Eq. 167 is necessary. It is sufficient because the quadratic form is convex. Incidentally, the theorem tells us that $\mathbb{E}(X\,Y(k)) = \mathbb{E}(\hat{X}\,Y(k))$. ■

109. NbJE. See Course 2019 Sec. 4.1

We can now generalize, we now write a known signal at a "time" n, $X(n)$, and its estimation is done using d data

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

with $H = (h[k,n])_{k,n}$ a matrix. We will relate it to the autocorrelation of the process

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

We consider X and Y as random vectors of d components.

Theorem 12 *(**Optimal linear estimator**)*
The optimal operator H is given by

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

Proof 12. We want to minimize

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

that is to say, to find the minimum of each element of the sum. According to the previous theorem, we have $\mathbb{E}(X(n)Y(k)) = \mathbb{E}(\hat{X}(n)Y(k))\ (\forall k)$, which we can expand as

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

Now, $Y(k) = X(k) + Z(k)$ and according to the noise assumptions, we have for all k, n

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

Thus, in matrix form, we can write

$$C = (C + \sigma^2 Id)H \tag{176}$$

which gives the expected result, since the matrix in parentheses is always invertible.

A remarkable result: we can rewrite the expression for H as

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

which **exactly corresponds to the formula of the optimal denoiser obtained with the score in the Gaussian case** (Eq. 157). Therefore, **the linear estimator is also the optimal (both linear and non-linear) estimator for Gaussian processes**. ■

How can we obtain the most suitable expression for H? In a sense, we need to find a basis in which C is as simple as possible, that is, to find **the basis that diagonalizes the autocorrelation matrix**, which is the **PCA basis** [110]. Let $(e_k)_k$ be this basis, the eigenvalues denoted $(\lambda_k)_k$ are such that [111]

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

Now, $C+\sigma^2 Id$ is also diagonal in the PCA basis, so is H, and we obtain that its eigenvalues are such that

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

What is $H[X]$? It is given by

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

We then see that H filters X, keeping only the components with significant energy along the vectors of the PCA basis. We can then order the values λ_k or $|\langle X, e_k \rangle|^2$ as a function

110. nb. Principal Component Analysis
111. NbJE. See Course 2021 Sec. 4.3.

of k, as shown in Figure 38. Note that k also indicates a direction via e_k. White noise has the same energy in all directions (dashed horizontal line), and it constitutes a threshold: $\forall k < k_M$, the signal has more energy than the noise, so H keeps these components.

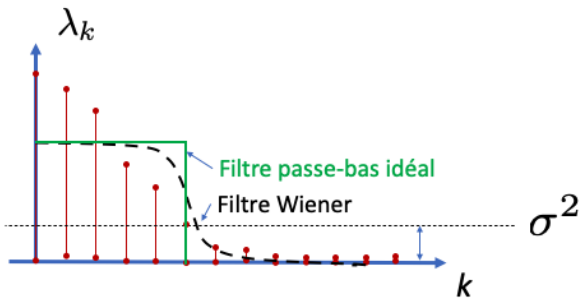


FIGURE 38 – Decay of the different eigenvalues λ_k in the PCA basis, noise thresholding effect. The Wiener filter (Eq.180) is shown with a black dashed line, and the ideal low-pass filter with a solid green line.

This is the basic idea behind optimal linear filtering. It remains to (re)identify the basis that diagonalizes the signal's covariance. For this, we will use a natural hypothesis of **stationarity**.

7.4 Stationarity, Fourier

The concept of **stationarity** tells us that if $(\forall n)$ $X(n)$ is shifted by a factor τ, $X_\tau(n) = X(n-\tau)$, then X **and** X_τ **have the same probability density**. This assumption is quite natural. Consider the example of photos taken by a camera to illustrate the point: if we do not center each photo on a specific object (a vase), that same object will appear at different locations in successive photos of a randomly filmed scene. However, it is the same object, and its probability of appearing is identical in each photo. Thus, any image of the object has the same probability of appearing as its translation by any value (**translation invariance**).

What are the consequences of this translation invariance? In particular, we know that the mean of $X(n)$ does not depend on n, and we can look at the second-order terms,

etc.:

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

The autocorrelation matrix has a band structure, as shown in Figure 39.

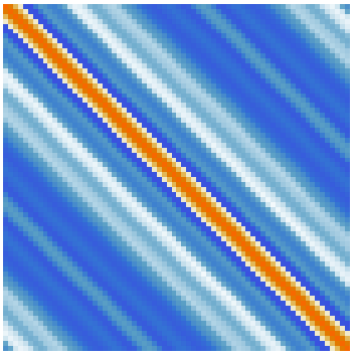


FIGURE 39 – Example of an autocorrelation matrix under the assumption of translation invariance: it shows the band structure of a symmetric Toeplitz matrix.

Now, if we consider C as an operator, then

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

The autocorrelation matrix acts as a convolution operator. Therefore, "translation invariance" also means "convolution operator."

We can then answer the question of the basis that diagonalizes C, that is, the basis that **diagonalizes a convolution operator**: it is the **Fourier basis**, whose vectors are written as $e_\omega(n) = e^{i\omega n}$ (here, $n \in \mathbb{Z}$). Indeed,

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

Thus, the family $\{e_\omega = e^{i\omega \cdot}\}_\omega$ forms the eigenvectors of C, with the eigenvalues being the values taken by **the Fourier transform of the autocorrelation** of the signal at ω

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

which is referred to as the **spectral power**.

In the case of finite-size signals, a periodic extension procedure [112] must be applied to consider circular convolution. The vectors of the basis are also periodic and written as

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

with quantized frequencies. This is the **Discrete Fourier Basis**.

Thus, whenever we deal with a stationary process, the Karhunen-Loève (PCA) basis is the Fourier basis. We can then visualize the spectral power (Fig. 40) of the decay of the λ_k in Figure 39. The **Wiener filter** is then analyzed as a **low-pass filter** with a cutoff depending on the ratio between the signal's energy and the noise's energy.

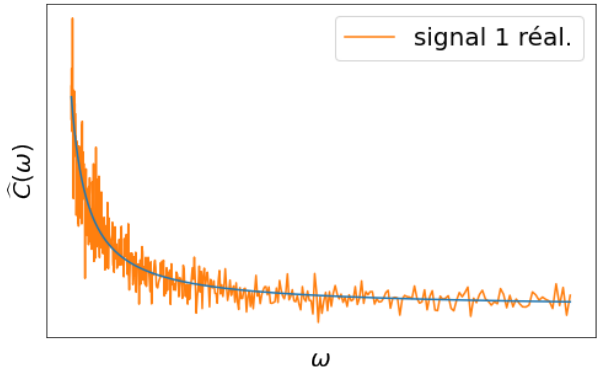


FIGURE 40 – Analysis of Figure 39 from a spectral perspective, with the correspondence $\omega \leftrightarrow k$.

An example of Wiener filtering is presented by S. Mallat on the projector with a Gaussian process. Another example [113] is presented in Figure 41.

112. NbJE. See e.g., Course 2018 Sec. 5.2.2
113. NbJE. You can replay it on the GitHub repository with the notebook `WienerFilter_GP.ipynb`

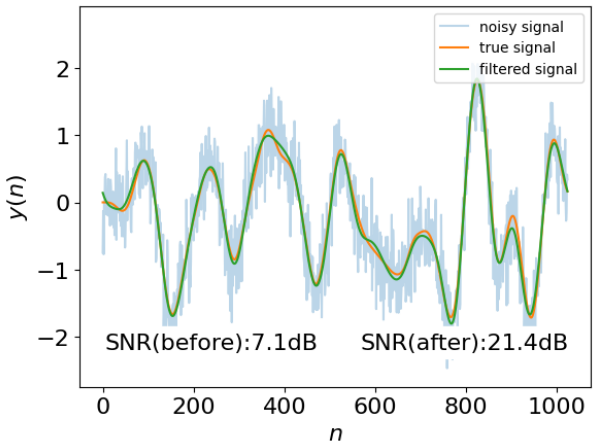


FIGURE 41 – Wiener filtering applied to a Gaussian process corrupted by white noise.

The result is quite satisfactory. In terms of SNR (*signal-to-noise ratio*[114]), we observe a gain of +14.3dB. The Wiener filter, as a low-pass filter, performs a local averaging. We cannot do better than this, whether in linear or nonlinear terms, due to the optimality of the denoiser for a Gaussian process (Th. 10).

That being said, when considering images of bedrooms, vocal or instrumental sound tracks, there are **transient phenomena** that are no longer Gaussian processes (Fig. 42)[115]. A significantly lower grain is observed in the denoising (+4.5dB). There is still some noise, and we understand that there is a competition between reducing the noise, which tends to suppress high frequencies, while wanting to preserve the sharp transitions that tend not to suppress high frequencies too much. This optimization, in the case of Wiener filtering, is performed by minimizing the quadratic error.

114. $SNR = 10\log_{10}(\|x\|^2/\|\hat{x} - x\|^2)$
115. NbJE. On the GitHub repository, you will find the notebook `WienerFilter_transitoires.ipynb` to replay the example.

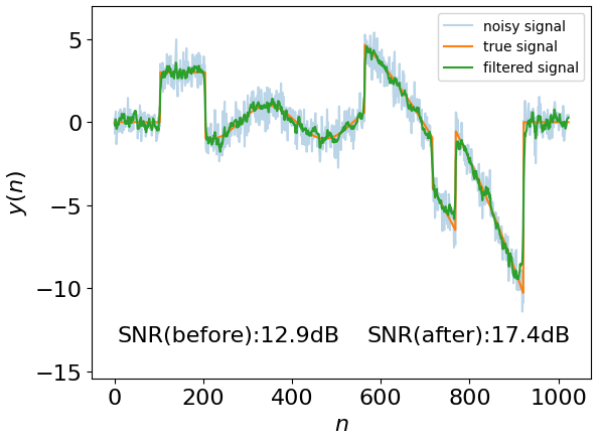


FIGURE 42 – Wiener filtering applied to a process with transients, corrupted by white noise.

The question then arises: can we improve denoising, and how? Imagine that we have the ability to detect the discontinuities in the signal, then during the "calm" phases, we could apply a severe filtering of high frequencies to completely reduce the noise. But in doing so, the algorithm becomes signal-dependent. We then move *de facto* into the **non-linear** world. We need to think of an estimator that adapts according to the discontinuities of the signal being processed. It cannot be a convolution operator because, by nature, it is associated with translation invariance. Here, **we need to adapt to the signal** in order to perform local operations that do not occur everywhere. This will be the subject of the next sections.

8. Lecture of March 5th

We began in the last session exploring the mathematical chapter on denoising, with the aim of explaining the properties of the **U-Net** network (Fig. 33) used to compute the *score* in high dimensions. We covered the "**linear**" version of denoising and highlighted the **optimal Wiener filter** for linear operators, and in the *stationary* case, the emergence of the triptych "**convolution, translation invariance, Fourier basis**". While Wiener filtering is optimal for Gaussian processes (you cannot do better) as shown in figure 41, when it comes to *transient phenomena* (Fig. 42), we must consider *adaptive denoising*, and

therefore shift towards the **non-linear** domain, where we will see the emergence of the notions of **sparsity** and **scales**. *NbJE. As with the previous session, it is useful to keep an eye on the earlier courses, such as the one from 2021.*

8.1 Non-linear diagonal estimation in a basis

The goal is to find orthonormal bases for which the signals have representations where the energy is concentrated on a small number of coefficients (the notion of sparsity). The questions we have in mind through the lens of U-Net are: why convolutions, why non-linearities like ReLU, and why is the multi-scale aspect important?

We consider the case of an additive noise corrupting a **deterministic fixed signal** x [116] according to

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

where Z is a white Gaussian noise (wgn) with zero mean and covariance $\sigma^2 Id$, and is **independent of the signal** x. We want to compute an estimator $\hat{x}$ obtained by applying an operator H to the noisy signal Y. Thus,

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

To obtain the expression for H, we naturally use the minimization of the mean squared error:

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

However [117], now H is an operator that is no longer constrained to be *linear* [118]. We allow the operator to **adapt to the irregularities of the signal**, so H is now signal-dependent. By the way, we need to be able to detect singularities caused by transient phenomena. Now, if we imagine taking the derivative of the signal (Fig. 42), due to the noise, this technique is doomed to fail. However, we can look at the increments of the signal at different *scales*. This was the basis of increasingly complex algorithms in the 1970s and 1980s, as S. Mallat tells us. All this was considerably simplified when we adopted the

116. NbJE. Note the difference from the previous session where X was a *random variable*.
117. NbJE. S. Mallat provides a summary of the previous session, which I do not transcribe here.
118. NbJE. See footnote 108.

point of view of the **sparse representation** of the signal and applied a **hard thresholding** on the small coefficients. In the 2021 course, the application of *sparse representations* was focused on signal compression, whereas here we see the other major application, which is denoising, hence the triptych "**representation, compression, denoising**".

We will continue with the case of orthonormal bases $\{e_k\}_{k\leq d}$ [119], bases to be discovered later. If we are looking, for a fixed x, for the expression of the **non-linear** H, we will nonetheless **constrain it to be diagonal**, meaning that for any signal x

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

This is a diagonal representation, because the coefficients $\langle x, e_k\rangle$ are only affected by a multiplicative factor. Thus, the coefficients $h[k;x]$ depend on the signal x itself. Later, to simplify notation, we write $h[k]$, but we must keep in mind this dependency in the background. The minimization of the squared error gives the expressions for the coefficients $h[k]$. Being in an orthonormal basis makes this easier, as we get [120]

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

The best $h[k]$ will cancel the derivative of the error (or risk r), as in the linear case. Thus, we obtain

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

We end up with the expression for the *eigenvalues* of the operator H in the linear case (Eq. 179). However, although we obtain the expression for the best $h[k]$, this calculation is futile, because **we do not know** x! Nonetheless, this *ideal* estimator is what is called an **oracle**. The questions then are whether it is possible to do as well, and whether the estimator is optimal? These questions are related to the properties of the chosen orthonormal basis. S. Mallat mentions that two researchers have marked the field by answering these

119. $\langle e_\ell, e_k\rangle = \delta_{\ell,k}$.
120. nb. $\mathbb{E}_Z[\langle Z, e_k\rangle] = 0$, $\mathbb{E}_Z[|\langle Z, e_k\rangle|^2] = \sigma^2$.

questions: David L. Donoho and Iain M. Johnstone, whose first publication dates back to 1994 [121].

The first idea that came up was to simplify the expression for $h[k]$. Referring to the linear case, the best $\lambda_k = \mathbb{E}_X[\langle X, e_k \rangle]$ can very well be filtered by an **ideal low-pass filter**. Therefore, let's try an expression for $h[k]$ that is either 1 or 0 (**binary filter**). The optimal choice according to the criterion of minimizing r would be the threshold that operates the $0 \leftrightarrow 1$ transition. We then get:

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

Certainly, the expression is simplified, but it is still an oracle due to the dependence on the unknown signal x. What would the error be in this case? We denote it by r_b for *binary risk*:

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

to be compared with the minimum risk expression (Eq. 191). However, for any constant pair (a, b) we have:

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

so

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

Thus, if we opt for the binary choice, at worst, we lose a factor of 2, which is not very significant when we are looking for asymptotic calculations up to a constant.

However, this is still an oracle result. But let's agree that this raises the problem in more general terms: **when should we keep or eliminate the signal**? Figure 43 can serve as a support for reflection. Let's assume for a moment that the basis has been well chosen; we still only have access to the coefficients $\langle y, e_k \rangle$ of the noisy signal $y = x + z$. Now, we hope that the coefficients of the noise z are small compared to the few most significant coefficients of the signal, otherwise, we would observe only noise. In this context, we can then apply a threshold on $|\langle y, e_k \rangle|$ as shown in (Fig. 43):

121. David L. Donoho and Iain M. Johnstone, *Ideal Spatial Adaptation by Wavelet Shrinkage*, Biometrika, 81(3):425–455, 1994. NbJE. Their Stanford report dates to 1992. https://imjohnstone.su.domains/WEBLIST/1994/isaws.pdf.

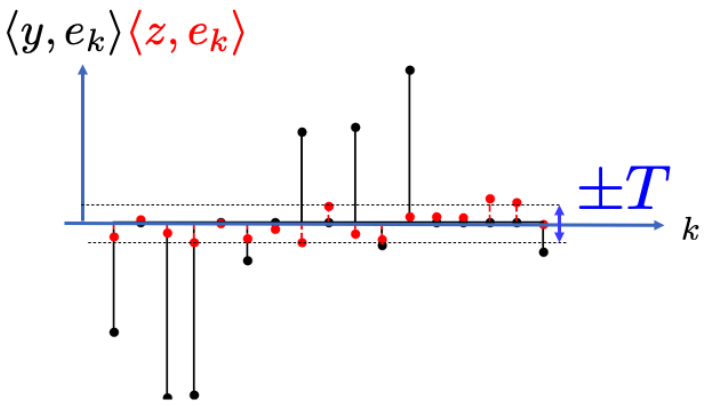


FIGURE 43 – Distribution of the scalar products of the noisy signal y and the noise z. A threshold at $\pm T$ can remove the noise without significantly distorting the characteristics of the signal x.

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

The next step is to choose the value of the threshold T, and for what efficiency? The idea is to choose T such that if we keep a coefficient, then it has a very high probability of being larger than the noise. We then look at the maximum values that the noise can reach. Note that the noise z is Gaussian, so the more we sample the distribution, although most of them lie in the interval $[-\sigma, \sigma]$ (68% C.L.), the more we explore the tails of the distribution. We thus have the following result:

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

with $o(1)$ of the order of $\log(\log d)/\log d$. Now, the operator H can be defined for any signal x according to the expression:

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

for example, with the following thresholding function (**hard threshold**):

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

However, note that the function ρ_T as defined is discontinuous: up to some extent, a coefficient near the threshold can either be kept or rejected, which is not optimal when we return to the prediction of the "non-binary" oracle (Eq. 191). This motivates the use of a *soft* thresholding (**soft threshold**) defined as shown in (Fig. 44):

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

The idea behind attenuating the value of a is that the noise has also contaminated the most significant coefficients of the underlying signal. Note that the soft thresholding is Lipschitz, and it can be rewritten using 2 ReLUs [122]:

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

When we formulate the action of H on the noisy signal:

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

we realize that we have a kind of primitive U-Net with: **a decomposition on a basis to obtain $\langle Y, e_k \rangle$, followed by a non-linearity ρ_T, and a reconstruction $\sum_k (.) e_k$**. Thus, we have an effective denoising algorithm, and the question is whether it is effective.

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

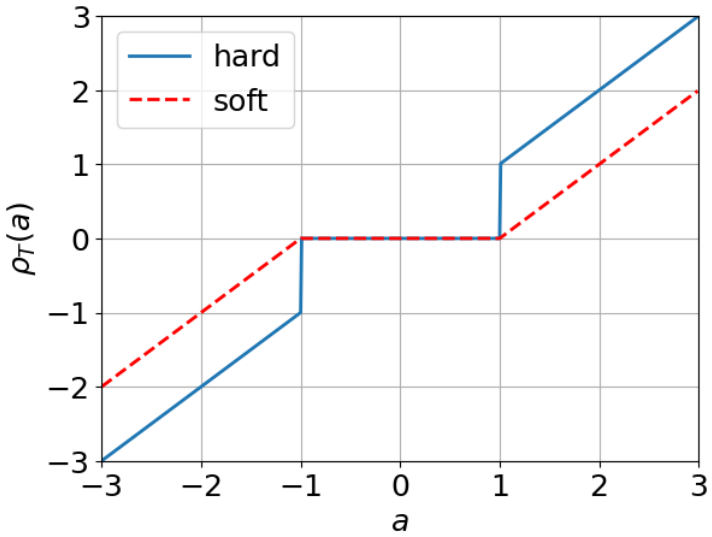


FIGURE 44 – Example of "hard" thresholding (Eq. 199) and "soft" thresholding (Eq. 200) with $T = 1$.

Donoho and Johnstone showed the very nice result below:

Theorem 13 *(**Donoho and Johnstone**, 1994)*
If $T = \sigma\sqrt{2\log d}$, then the thresholding error (soft or hard) $r_s(x)$ is bounded as:

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

Moreover, the factor $2\log d$ is optimal ($d \to \infty$) in the case of a diagonal operator.

This theorem tells us that if we choose the threshold well, the efficiency is good compared to the oracle result. Up to a constant, we have a risk that is $2\log d$ times greater than the risk of an oracle (binary or non). Note that the constant $2\sigma^2 \log d = T^2$ is the risk incurred from keeping only one coefficient above the noise. The second result of the theorem tells us that we cannot find a multiplicative constant less than $2\log d$ for any x. **We thus have an efficient, almost optimal non-linear algorithm, and it is very simple to implement** [123].

123. NbJE. Note that it remains to determine the value of σ. You can see how this

From this very interesting result, S. Mallat tells us that the community has focused on finding the **right basis**. What makes a basis "good"?

8.2 Sparsity Bases

Recall that in the linear case, the appropriate basis is the one that diagonalizes the covariance (PCA), which, in the stationary case, is the Fourier basis. In the non-linear case, it is a bit more complex, because the goal is to minimize the *binary* risk (Eq. 193). We can expand the sum over k using the following formulation:

$$r_b = \sum_k \min(|\langle x, e_k\rangle|^2, \sigma^2) = \underbrace{\sum_{|\langle x,e_k\rangle|\leq\sigma} |\langle x, e_k\rangle|^2}_{bias} + \underbrace{M\sigma^2}_{variance} \tag{204}$$

where M is the number of coefficients such that $|\langle x, e_k\rangle| > \sigma$. We would like to minimize both terms of this sum, keeping in mind that

$$\sum_k |\langle x, e_k\rangle|^2 = \|x\|^2 = \text{constant}. \tag{205}$$

Now, we can interpret the second term as follows: if we keep M coefficients whose energy is above the noise, we also retain some noise [124] whose energy is $M\sigma^2$. This is a *variance term*, because, in a sense, it provides information on the fluctuations of the denoising estimator. The first term concerns all the coefficients of the decomposition that we have eliminated. As a result, we have removed some of the signal's energy, so this term is identified as a *bias term* in estimation.

Note that an ideal case would be achieved if there existed only a single vector in the basis that accounts for all the energy of the signal. In this case, the binary risk would be $r_b = \sigma^2$. In a less extreme case, we aim to obtain a **small number M of coefficients that capture almost all the energy**, meaning the other coefficients are nearly zero, and

is obtained in the notebooks of the GitHub repository: `Wavelet_transitoires.ipynb` and `Wavelet_stationnary_transitoires.ipynb`. Libraries such as `PyWavelets` in Python and `Mathematica` provide access to other thresholding functions that perform intermediates between *hard* and *soft* thresholding.

124. Reminder: the signal is contaminated by the random variable Z.

in this case, $r_b \approx M\sigma^2$. Thus, **we seek bases for which the representation of the signal is as sparse as possible.**

S. Mallat tells us that in this context, there are many papers exploring how/why denoising neural networks perform sparse representation calculations. That said, there is an entire chapter of mathematics that deals with this type of representation, which S. Mallat briefly covers through some slides. *NbJE. One can simultaneously refer to his 2021 course.*

8.2.1 A Brief Return to Fourier

Notation: We work in infinite dimensions, and the signal $x(u)$ becomes a function that is denoted as $f(t)$ from now on. Why is it important to move from the *discrete* world to a *continuous* one? For example, we want to understand the properties of the coefficients in the basis and relate them to the **regularity properties of the function** with notions of continuity, derivatives, etc. The *discrete-to-continuous* transition occurs by fixing the support of u, e.g., $[0, 2\pi]$, $[0, 1]$, etc., while increasing the sample density (or the value of d). Note that in the context of Fourier, for functions $f \in L^2([0, 2\pi])$, we have

$$\hat{f}(\omega) = \langle f(t), e^{i\omega t} \rangle = \int_0^{2\pi} f(t)\, e^{i\omega t}\, dt \tag{206}$$

which interprets $\hat{f}(\omega)$ as the inner product of $f(t)$ with the sinusoid $e^{i\omega t}$: $\hat{f}(\omega)$ reflects the variations of f at the "frequency" ω. Now, if f is a smooth function, there are more low-frequency components than high-frequency ones. Fourier theory links the decay rate of $\hat{f}(\omega)$ to the overall smoothness of f (Fig. 45-top).

The problem in Fourier analysis becomes evident when we add a small peak to f with regularity C^α: since the Fourier transform is linear, by superposition, all the coefficients of f are affected, particularly those at high frequencies (Fig. 45-bottom). Note that if we applied a "hard" cutoff on the frequencies, e.g., $\omega < 10$, to keep only the most important low-frequency coefficients, we would lose any description of the small peak.

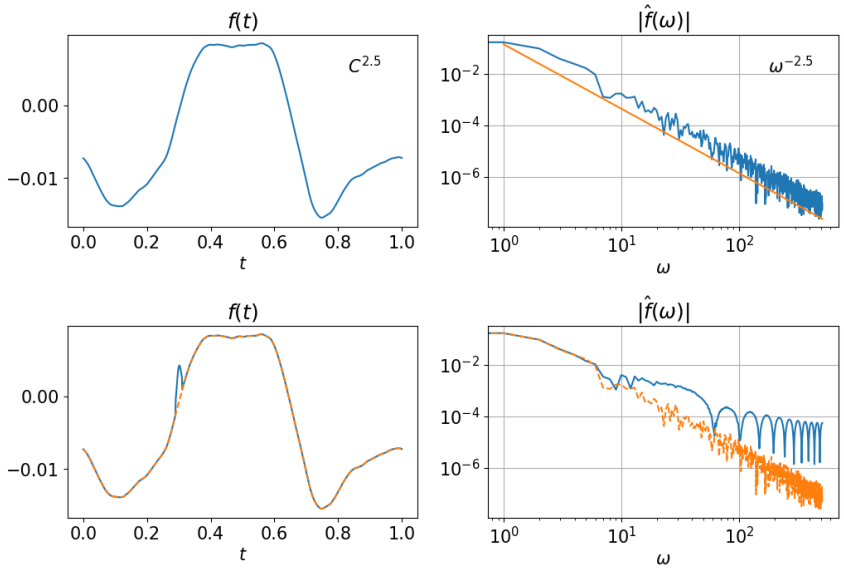


FIGURE 45 – Example of Fourier processing of a function with regularity $C^{2.5}$ (top) to which a small localized peak is added in time (bottom). We notice that in the first case, the decay of the Fourier coefficients provides good information about the function's regularity (see the orange curve $\propto \omega^{-2.5}$ on the spectrum), and in the second case, the high-frequency spectrum is dominated by that of the perturbation (the initial spectrum in orange).

The moral of the story is that if the function is smooth, the Fourier representation is sparse, but **as soon as there is a singularity or a transient, sparsity is lost**.

8.2.2 Wavelet Analysis

NbJE. You can also refer to Section Courss 2021 Sec.5.3, as S. Mallat, due to time constraints, focuses on the essential to show the mathematical path.

The question is how to construct orthonormal bases that use the global regularity of the signal while taking into account the singularities to produce a sparse representation? We would actually like the number of significant coefficients not to be affected by the presence of singularities. Note that in Figure 45 (bottom), to preserve the peak, which may be an important feature, we must keep all coefficients whose value is above 10^{-4}. Thus, we need to provide a property of **spatial localization of the basis vectors**, as illustrated

in Figure 46. This is the idea developed in the 1980s independently by **Jean Morlet** and **Alex Grossmann**, the former for seismic analysis in oil exploration at Elf-Acquitaine and the latter for the study of coherent states in quantum physics: this is the **wavelet analysis**.

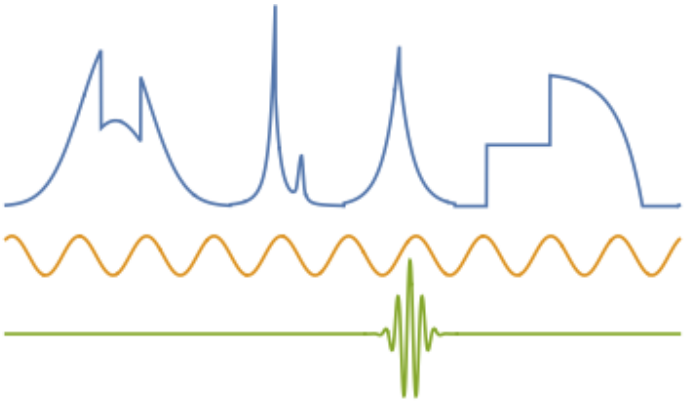


FIGURE 46 – Different perspectives of analyzing a function: either the *linear* framework that assumes uniform regularity, giving Fourier analysis with delocalized sinusoids in time/space, or the *non-linear* framework that studies non-uniformly regular functions with wavelet analysis that performs a local analysis of transients/discontinuities.

The idea is to start with a locally oscillating function $\psi(t)$ with zero mean, such that

$$\int_{-\infty}^{\infty} \psi(t)\, dt = 0 \tag{207}$$

To accommodate the time scales and localization of the signal's discontinuities, translations and dilations are applied to ψ (Fig. 47):

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{t-u}{s}\right) \tag{208}$$

The support is given by the scale term s, and its frequency domain [125] is multiplied by $1/s$.

125. Note that $\frac{1}{s}h(\frac{t}{s}) \xrightarrow{TF} \hat{h}(s\,\omega)$

FIGURE 47 – Illustration of translation and scaling operations applied to a wavelet to obtain different versions of $\psi_{u,s}$, where s is the support of the new wavelet.

As with Fourier, we compute inner products:

$$\langle f, \psi_{u,s} \rangle = \int f(t)\psi_{u,s}(t)\, dt \tag{209}$$

and produce time-scale or time-frequency maps (scalogram) like the one in Figure 48. For each point on the map, the value of the inner product is represented. We note: 1) that most of the coefficients are zero, because if the wavelet is well localized spatially (small scale, high frequency), the inner product is locally proportional to the average of the wavelet, which is zero; 2) that the most significant coefficients at a fixed scale are localized at the "instants" of the discontinuities, particularly at high frequencies; and 3) the lower the frequency, the less effective the localization becomes.

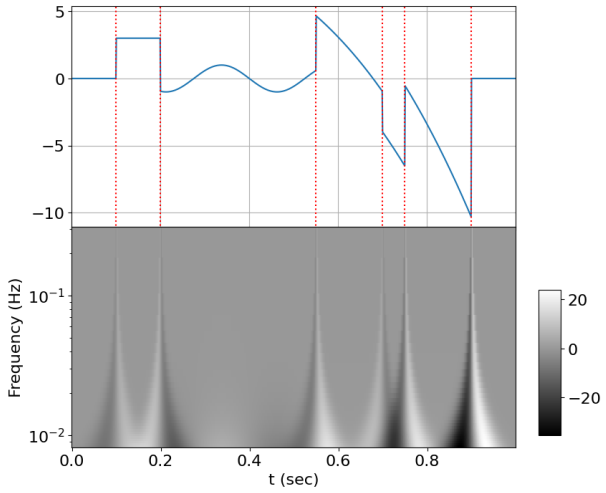


FIGURE 48 – Top: piecewise regular signal. Bottom: map with the translation parameter on the x-axis and frequency on the y-axis (high frequency = small scale). Here we used the real wavelet in the shape of a Mexican hat ("mexh" for the `PyWavelets` library.)

From all the coefficients/inner products, we can reconstruct the function f, which is the **continuous inverse wavelet transform**:

$$f(t) = C \iint \langle f, \psi_{u,s} \rangle \psi_{u,s}(t)\, du \frac{ds}{s^2} \tag{210}$$

Similarly, one may ask the question of the **characterization of the regularity** of f via the decay of the inner products as in Fourier? And most importantly, can we **construct orthonormal bases**?

It can be noted that the description in terms of time-frequency maps, like the one in Figure 48, is highly redundant and not optimal in terms of sparsity. We can significantly reduce the information without loss by performing a **dyadic sampling**: $s = 2^j$, and similarly for the translations. From this, we can construct sparse orthonormal bases.

If we fix $s = 2^j$, then let $\psi_j(t) = \frac{1}{\sqrt{2^j}} \psi\left(\frac{-t}{2^j}\right)$. We realize that the coefficient/inner product of f with $\psi_{u,2^j}$ is the result of a convolution:

$$d_j(u) = \langle f, \psi_{u,2^j} \rangle = \int f(t) \frac{1}{\sqrt{2^j}} \psi\left(\frac{t-u}{2^j}\right) dt = f * \psi_j(u) \tag{211}$$

Thus, **the wavelets ψ_j are bandpass filters** [126] with a typical width of 2^{-j} (Fig. 49).

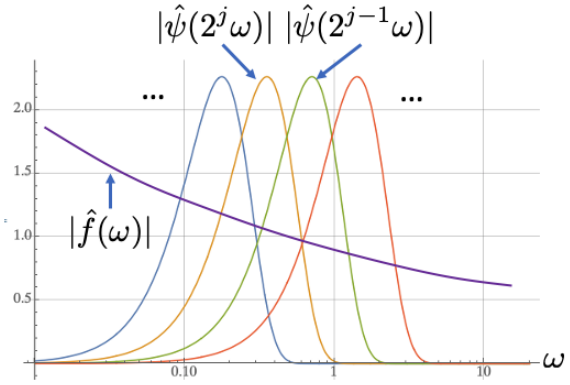


FIGURE 49 – Collection of filters $\hat{\psi}(2^j\omega)$ that analyze the signal f.

We then have a collection (infinite) of coefficients $\hat{d}_j(\omega)$ that account for the signal's power in all the bandpass filters:

$$\hat{d}_j(\omega) = \hat{f}(\omega)\,\sqrt{2^j}\,\hat{\psi}(2^j\omega) \tag{212}$$

The reconstruction of the signal from the $\{d_j\}_{j\in\mathbb{Z}}$ is possible if the filters $\hat{\psi}(2^j\omega)$ overlap to **avoid leaving gaps in the Fourier spectrum**. This is the Littlewood-Paley condition (1930s):

$$\sum_j |\hat{\psi}(2^j\omega)|^2 = 1 \Rightarrow f(t) = \sum_{j\in\mathbb{Z}} \int d_j(u)\,\psi_{u,2^j}(t)\,du \tag{213}$$

Thus, we can limit ourselves to scales $s = 2^j$. Now, can we compress the translation information, for example, by sampling the time axis according to $u = 2^j n$ as shown in Figure 50?

126. Note: in fact, $\hat{\psi}(0) = \int \psi(t)\,dt = 0$, so we indeed have a filter whose low-frequency part tends to 0.

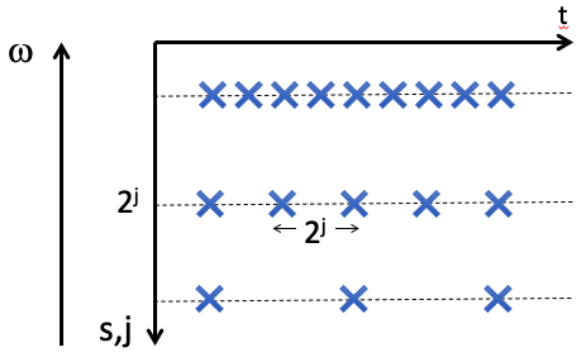


FIGURE 50 – Dyadic sampling of the "time-scale" (or "time-frequency") map.

The answer comes from the work of **Alfréd Haar** (1909), who using a particular wavelet (a term unknown at the time) ψ (Fig. 51) constructed **the first orthonormal basis of** $L^2(\mathbb{R})$:

$$\left\{\psi_{j,n}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t-2^j n}{2^j}\right)\right\}_{(j,n\in\mathbb{Z}^2)} \tag{214}$$

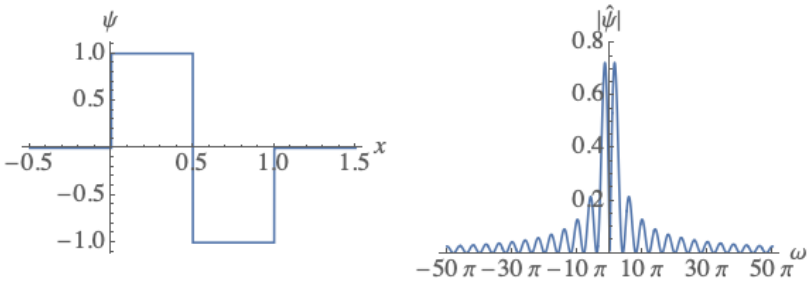


FIGURE 51 – "Haar wavelet" (also "db1" in I. Daubechies' wavelet family) constructed in real space. Decay as $1/\omega$ in the Fourier space due to discontinuities.

Then, **Claude Shannon** and **Harry Nyquist** in the years 1948-49 demonstrated the *sampling theorem* using perfect bandpass filters, and it can be interpreted in terms of wavelet decomposition (Fig. 52).

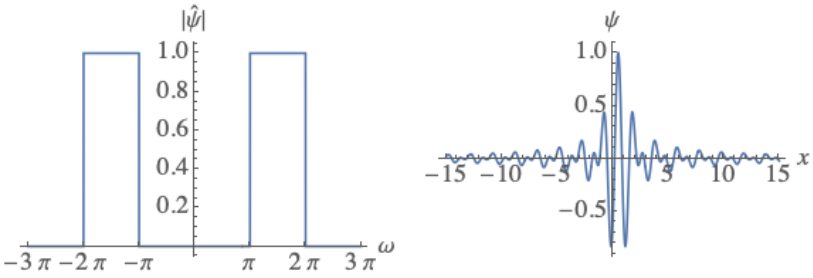


FIGURE 52 – Shannon wavelet constructed in the Fourier space. Decay as $1/t$ in real space.

The problem with the Haar and Shannon wavelets is that in real space, the first is discontinuous, and the second decays very slowly. However, we are looking for "local sinusoids" (Fig. 46) to not only allow localization of the signal's discontinuities but also to form orthonormal bases. But is this possible? It was believed not to be, and the subject was left aside. It took **Yves Meyer** in 1986 (Abel Prize 2017) attempting to prove that this was impossible, only to find a solution! An example of a Meyer wavelet is shown in Figure 53: it is a smooth (C^∞) function with rapid decay, and in Fourier space, it is a bandpass filter. **We now have an orthonormal basis of wavelets that satisfy our localization criteria in both real and Fourier space** [127].

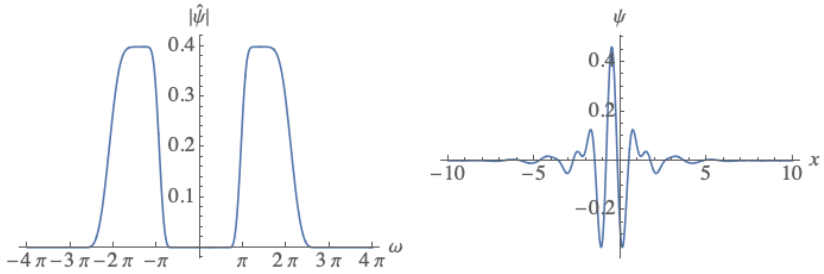


FIGURE 53 – Example of a Meyer wavelet constructed in Fourier space. Note its rapid decay in real space while being localized in Fourier space.

From this fundamental result, an entire field of research followed, particularly **Multi-**

127. Note: The uncertainty principle constrains us from achieving perfect localization simultaneously in both spaces.

resolution Analysis (or Multi-scale Analysis), the search for **compact support wavelets**, the development of **fast Wavelet Transform algorithms** with cascades, etc. We will see in the next session how this allows us to perform **denoising by thresholding** on the few coefficients that capture the entire signal's energy (1D, 2D). We will then ask the question: when is **denoising optimal**? We will see that **in 1D, there are many situations where the optimum is achieved, but in 2D (images), as soon as there is geometric regularity, we lose optimality.** We will need to change our perspective. In this context, we will see that **neural networks** compute much more subtle bases than wavelets: while they share properties of harmonicity (oscillation), they adapt to **the geometry of the signal**.

9. Lecture of March 12th

9.1 Brief Summary of Previous Sessions

We briefly recall that during the previous sessions, we studied the properties of a neural network such as the **U-Net**, which helps us estimate the **score** through denoising. This network, and thus the score estimation, allows us to use a **transport method** with the help of a stochastic differential equation to **generate new samples** from p_{data}. This U-Net incorporates classical components such as **convolutions**, **non-linearities** (ReLU), and also **downsampling/upsampling cascades** with connections that enable the computation of correlations across different **scales**. Ultimately, the central question we ask is: why is the U-Net an optimal denoiser?

We began addressing this question in the last session through the lens of **Harmonic Analysis**. Thus, we initially tackled the problem by restricting ourselves to **"linear" operators**, studying Wiener filtering and diagonal operators in Fourier bases. However, this approach reaches its limits when dealing with signals exhibiting **transient phenomena**, which are nonetheless fundamental (e.g., object recognition in an image, identification of musical instruments in an audio track, speaker identification in a voice recording, shock waves in fluid mechanics, etc.).

We saw that we could adapt to signal discontinuities by using **"non-linear" operators**. During this session, we will extend this **non-linear approach**, which studies **sparse**

representations that use **thresholding** to reduce noise while preserving the main characteristics of the signal, particularly its discontinuities. This led us to discuss **wavelet bases**, which indeed improve denoising performance in certain cases. However, we will see that these cases remain fairly limited, especially since these tools do not adapt well to the geometry of complex images, such as faces, bedrooms, etc. Nevertheless, the "classical" harmonic approach allows us to tackle problems with well-mastered tools. This enables us to design a simple two-layer network model and highlight its limitations, particularly regarding the **capture of image geometry**. We will explore how adapting the representation basis can improve the results. Ultimately, in these complex cases, we realize that **the U-Net can be reinterpreted as an operator that computes bases adapted to specific encountered cases and applies thresholding to achieve optimal denoising**. However, let us keep in mind, as S. Mallat states, that *"we are far from understanding how these bases are computed"*.

NbJE. S. Mallat reviews, using a projector, the results presented since Section 7.2.

For reference, if $Y = x + Z$, where x is a fixed deterministic signal and Z is a Gaussian white noise random vector with zero mean and variance $\sigma^2\, Id$, an estimator of the signal is given by:

$$\hat{x} = H_T[Y] = \sum_k \rho_T(\langle Y, e_k \rangle)\, e_k = \sum_k h[k] \langle Y, e_k \rangle e_k \tag{215}$$

where ρ_T is a non-linearity of the "soft-thresholding" type, for example (Fig. 44). The first formulation can be viewed as a succession of operators (inner product, non-linearity, linear projection), similar to what is done in a neural network:

$$H[Y] = W^T\, ReLU\, W \tag{216}$$

The question is whether we can obtain optimal estimators.

We saw (Sec. 8) the comparison between a **thresholding estimator** and an **optimal estimator within the class of diagonal operators** in bases. The best we can achieve in this class is to compute the **binary risk of an oracle**, assuming that the selection retains the coefficient $\langle x, e_k \rangle$ only if it is greater than the threshold σ. Its expression is given by

(Eq. 204), which we recall here for convenience:

$$r_b(x) = \sum_k \min(|\langle x, e_k\rangle|^2, \sigma^2) = \sum_{|\langle x,e_k\rangle|\leq\sigma} |\langle x, e_k\rangle|^2 + M\sigma^2 \tag{217}$$

Donoho and Johnstone's theorem 13 provides us with a **thresholding algorithm** that can be practically implemented, with a threshold value $T = \sigma\sqrt{2\log d}$ and a risk expression r_s close to the binary risk, given by:

$$r_s(x) = \mathbb{E}_Z[\|H_T[Y] - x\|^2] \leq (2\log d + 1)\,(\sigma^2 + r_b(x)) \tag{218}$$

with the key fact that the factor $2\log d + 1$ cannot be reduced. This teaches us that with a thresholding algorithm, we can nearly achieve the performance of an oracle's binary estimator.

Let us note that the (oracle-type) approximation x_M of the signal x is given by [128]

$$x_M = \sum_{|\langle x,e_k\rangle|>\sigma} |\langle x, e_k\rangle|\, e_k \tag{219}$$

We can then rewrite r_b as:

$$r_b = \underbrace{\|x - x_M\|^2}_{bias} + \underbrace{M\sigma^2}_{variance} \tag{220}$$

where we identify the two types of errors (bias-variance).

The next step was therefore to minimize r_b, meaning we need to find sparse bases. This led us to introduce orthonormal wavelet bases. Specifically, we examined Yves Meyer's result (Fig. 53), which generalized Alfréd Haar's work by making it possible to design **orthonormal bases of** $L^2(\mathbb{R})$ using dyadic scales and translations:

$$\left\{\psi_{j,n}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t - 2^j n}{2^j}\right)\right\}_{(j,n\in\mathbb{Z}^2)} \tag{221}$$

with C^∞ **wavelets with rapid decay in real space**, whose Fourier support is essentially confined to the interval $[\pi, 2\pi]$ (and its symmetric counterpart), making them **band-pass filters** that, through scaling, cover the entire Fourier domain.

128. NbJE. See also Lecture 2021 Sec. 5.1

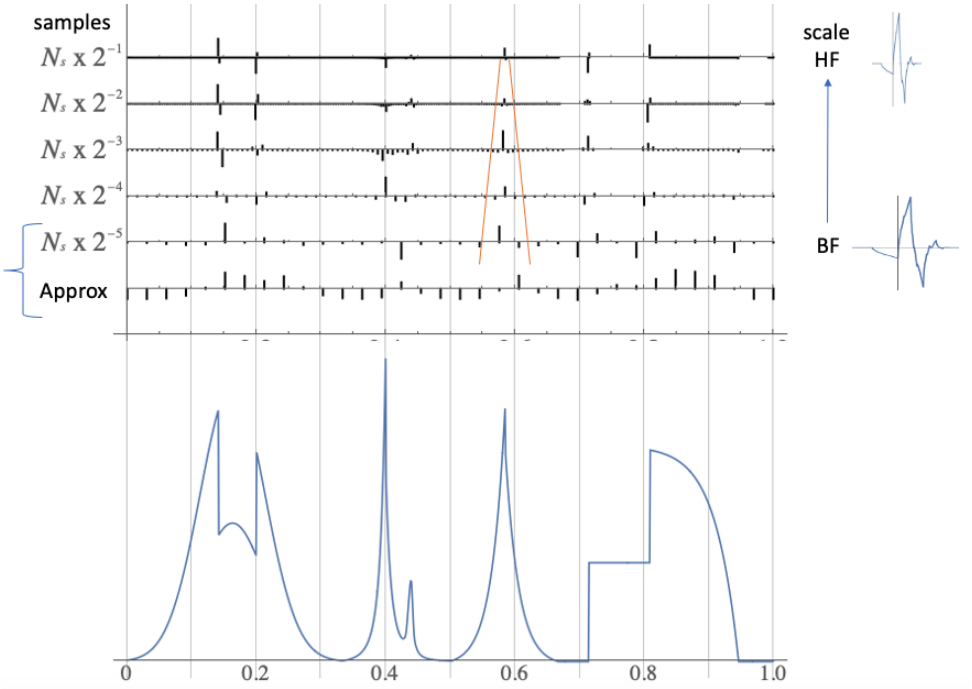


FIGURE 54 – From bottom to top: the sampled function yielding $2^S = 1024$ samples ($S = 10$), then through successive decomposition with the "Db2" wavelet, we obtain the low-frequency approximation coefficients, totaling $2^{S-5} = 32$, and the detail coefficients at the same scale, followed by detail coefficients of increasingly higher frequencies, ultimately reaching $2^{S-1} = 512$. Note that most detail coefficients are nearly zero, and the most significant ones concentrate at the function's discontinuities within the cone of Theorem 14 (in orange; see Th. 12, Course 2021). The vertical scales differ for each set of coefficients. *Nb. S. Mallat's projector presentation follows the reverse order for the arrangement of detail coefficients, from high to low, moving from low to high frequencies. Moreover, the low-resolution approximation is not represented.*

9.2 Regularity of a Function and Amplitude of Wavelet Coefficients

In the following, the vectors e_k are replaced by $\psi_{j,n}$, and the reconstruction of f (here, the signal x is denoted as f, Sec. 8.2.1) is written as:

$$f = \sum_{j,n \in \mathbb{Z}^2} \langle f, \psi_{j,n} \rangle \psi_{j,n} \tag{222}$$

The open question is whether wavelet bases allow for sparse representations of the signal. In other words, with M wavelets, is the approximation x_M accurate? If so, what is the thresholding risk?

We can truncate the above sum by applying a low-pass filter (large scales $j > J$), which we denote [129] as ϕ. Convolution of the signal with dilated/translated versions of ϕ provides low-frequency approximations of the signal. Note that for a sampled signal, there is also a high-frequency cutoff (small scales $j < j_{spl}$).

Figure 54 shows [130] an example of the decomposition of a signal with discontinuities analyzed using wavelets by I. Daubechies [131]. At small scales (high frequency), the coefficient $\langle f, \psi_{j,n} \rangle$, which measures the correlation between the signal and the wavelet, is zero when the wavelet is in a region where the function is constant over the wavelet's support. The correlation (or local contrast) becomes significant only at discontinuities. What we observe in this example is that there are indeed **very few large coefficients**, confirming a sparse representation.

Can we relate the regularity of the signal to the amplitude of the coefficients? The answer is yes, which makes wavelet analysis a very powerful tool, as S. Mallat explains. Consider the following theorem [132]:

129. NbJE. This can be a function associated with the wavelet ψ in the case of a multiresolution analysis, Course 2021 Sec. 7.1, as well as a Gaussian of appropriately chosen width so that, *ultimately*, the Fourier domain is fully covered.
130. NbJE. Taken from Course 2021 Fig. 46.
131. NbJE. See Course 2021 Sec. 8.1
132. NbJE. You may compare this with Th. 12 by S. Jaffard in Course 2021 Sec. 6.2.

Theorem 14 *If f is uniformly Lipschitz α, then there exists a constant $C > 0$ such that:*

$$(1) \qquad \forall j, n \in \mathbb{Z}^2 \quad |\langle f, \psi_{j,n} \rangle| \leq C\, 2^{j(\alpha+1/2)} \tag{223}$$

and if (1) holds, then f is locally Lipschitz $\alpha' < \alpha$.

Proof 14.

We prove here only the implication for a function that is uniformly Lipschitz α over a domain containing the points we will use. Let us explicitly write the inner product, keeping in mind that the mean of a wavelet is zero:

$$\begin{aligned} \langle f, \psi_{j,n} \rangle &= \int f(t) \frac{1}{\sqrt{2^j}} \psi\left(\frac{t - 2^j n}{2^j}\right) dt \\ &= \int \left[f(t) - f(2^j n)\right] \frac{1}{\sqrt{2^j}} \psi\left(\frac{t - 2^j n}{2^j}\right) dt \end{aligned} \tag{224}$$

Taking the absolute value, we obtain:

$$\langle f, \psi_{j,n} \rangle \leq \frac{1}{\sqrt{2^j}} \int |f(t) - f(2^j n)| \left|\psi\left(\frac{t - 2^j n}{2^j}\right)\right| dt \tag{225}$$

Since the function is Lipschitz α, we know that there exists $C > 0$ such that:

$$|f(t) - f(2^j n)| \leq C |t - 2^j n|^\alpha \tag{226}$$

thus:

$$\langle f, \psi_{j,n} \rangle \leq \frac{C}{\sqrt{2^j}} \int |t - 2^j n|^\alpha \left|\psi\left(\frac{t - 2^j n}{2^j}\right)\right| dt \tag{227}$$

Applying the change of variable $t' = (t - 2^j n)/2^j$, we obtain:

$$\langle f, \psi_{j,n} \rangle \leq C\, 2^{j(1/2+\alpha)} \times \int |t|^\alpha |\psi(t)| dt \tag{228}$$

The integral is a well-defined constant, as the wavelet has rapid decay (or is compactly supported in the case of Daubechies wavelets). Thus, we have the desired result. ■

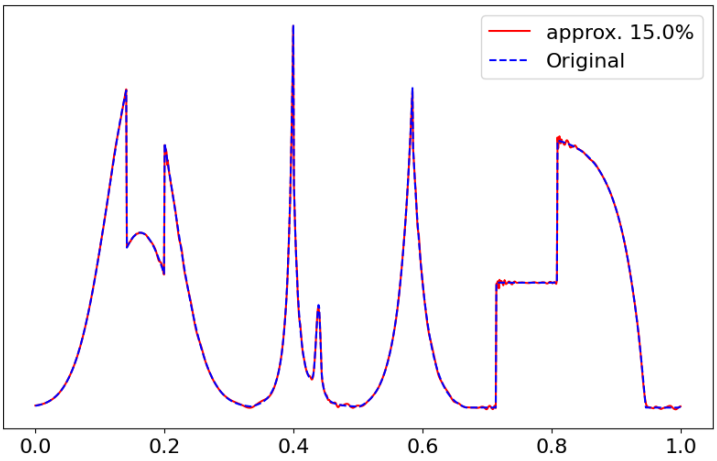


FIGURE 55 – Original signal (blue) from Figure 54 containing $2^{10} = 1,024$ samples. The Daubechies "db6" wavelet allows decomposition up to level $J = 5$. Then, only 15% of the computed coefficients (the most significant ones) are retained for reconstruction (orange).

In essence, this important theorem tells us that the correlations between a function and a dilated wavelet inform us about the dilation properties of the function. This property is illustrated by the "cones" (orange) in Figure 54, where the nonzero coefficients of the wavelet decomposition are concentrated. Thus, there is indeed a **link between the regularity of the function and the decay of wavelet coefficients**. This is a generic example of mathematical analysis. Note that the link between the decay of Fourier coefficients and differentiability in the Sobolev sense is another example.

9.3 Using the Sparse Representation

Once the wavelet coefficients $\langle f, \psi_{j,n} \rangle$ have been obtained, we can choose to keep only a fraction of them, retaining the most significant ones [133]. Figure 55 retains, for instance, only 15% of the coefficients to reconstruct the signal from Figure 54.

133. NbJE. See also Course 2021 Sec. 8.3. I have also made available the notebook `Wavelet_approx.ipynb` on the GitHub repository, which allows you to reproduce the exercise.

The signal is reconstructed very accurately [134]:

$$\frac{\|f - f_{\text{app.}}\|^2}{\|f\|^2} = 1.4\ 10^{-4} \tag{229}$$

We successfully reconstructed the signal using a very simple thresholding method, without explicitly considering the function's regularity. This is due to the "automatic" adaptation of the approximation to the function's (signal's) regularity.

This sparsity property can then be used for denoising. The coefficients of Gaussian white noise are themselves random variables that add to the signal's coefficients. By applying thresholding (soft or hard) with Donoho and Johnstone's universal threshold T, a nonlinear approximation can be obtained. An example is given in Figure 56, using the same noisy signal as in the Wiener filtering illustration (Fig. 42). The "nonlinear" approximation provides a significant improvement in SNR, increasing from 19.5dB to 24.9dB.

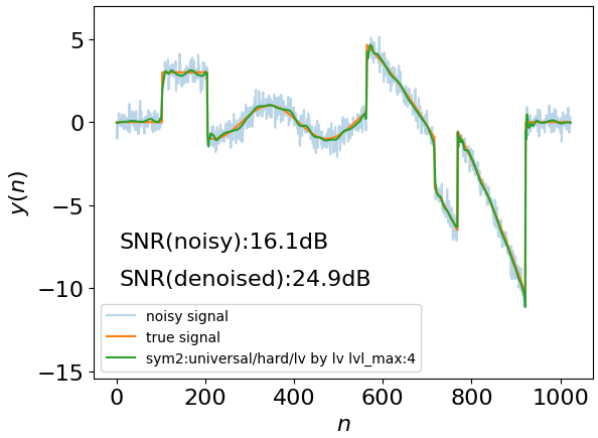


FIGURE 56 – Noise filtering using a hard "universal" threshold applied to detail coefficients obtained via a stationary wavelet transform using the "*sym4*" wavelet from the `PyWavelets` library. This should be compared to Figure 42, which used Wiener filtering.

The question then arises: is the signal estimator optimal? The answer is yes, but only

134. NbJE. Note the small ripples at some discontinuities; this is an intrinsic phenomenon caused by removing oscillatory wavelet components through thresholding (note that in the case of Fourier, we would have had more significant oscillations, known as the Gibbs phenomenon).

if the approximation with M coefficients is optimal within the given basis. This depends on the properties of the signal.

9.4 Regularity Assumptions (1D)

S. Mallat states that recent theorems over the past thirty years have aimed to weaken the regularity assumptions on functions. Here are two such theorems. The first states that:

> **Theorem 15** *If $x(t)$ is uniformly Lipschitz α (with no discontinuities) and only M coefficients are retained to construct an approximation x_M, then:*
>
> $$(1) \qquad \exists C > 0 \quad \|x - x_M\|^2 \leq C M^{-2\alpha} \tag{230}$$
>
> *and the thresholding risk $r_s(x)$ is bounded as*
>
> $$(2) \qquad \exists C > 0 \quad r_s(x) \leq C |\log \sigma| \sigma^{2\alpha/(\alpha+1/2)} \tag{231}$$
>
> *and the exponents of M and σ are optimal.*

This means that no nonlinear approximation technique can perform better with M coefficients. In fact, the same result would hold in either a Fourier or a wavelet basis. The more interesting result is the following:

> **Theorem 16** *If $x(t)$ has a finite number of discontinuities and is uniformly Lipschitz α elsewhere, then results (1) and (2) from the previous theorem remain valid.*

In other words, **using a nonlinear approximation, it is as if the discontinuities did not exist from the point of view of the exponents** of M and σ (although the constants change). To understand where this comes from, S. Mallat outlines elements of the proof [135]. In fact, to distinguish between the case of the signal in Theorem 15 and that in Theorem 16, one must simply look at the wavelets that produce a signal within the orange cones of Figure

135. NbJE. See Th. 19 and 20 in Course 2021 Sec. 9.2 for more details, except for the part on thresholding risk calculation.

54. Outside the singularities, we know from Theorem 14 that the coefficients decay as

$$|\langle x, \psi_{j,n} \rangle| \leq C\, 2^{j(\alpha+1/2)} \tag{232}$$

Thus, if we remove the high-frequency coefficients (in the Fourier-like manner) beyond 2^ℓ, then the error produced by this removal is given by the energy of the missing coefficients, namely (if the signal's support is of order 1) up to a constant C

$$\|x - x_M\|^2 = \sum_{n,j<\ell} |\langle x, \psi_{j,n} \rangle|^2 \leq \sum_{n,j<\ell} 2^{-j} 2^{j(2\alpha+1)} \approx 2^{2\alpha\ell} \tag{233}$$

and the number of retained coefficients is, up to a factor of 2, equal to $2^{-\ell}$ (Fig. 50), thus (up to constants)

$$\|x - x_M\|^2 \leq M^{-2\alpha} \tag{234}$$

This is result (1). **When we add discontinuities, it is important to notice that the number of coefficients in the cones (orange) is negligible compared to** M, which implies that the dynamics of nonlinear and linear approximation errors are the same.

Regarding the thresholding error $r_x(x)$, what threshold σ should be set to eliminate high frequencies beyond 2^ℓ? Typically,

$$\sigma \approx 2^{\ell(\alpha+1/2)} \approx M^{-(\alpha+1/2)} \tag{235}$$

and thus, by replacing M with this value of σ in the expression of $\|x - x_M\|^2$, we obtain the thresholding risk expression (at least the exponent's dynamics).

The key takeaway is that **the approximation is optimal if discontinuities are represented by a small number of large coefficients in the wavelet basis. This is the case for 1D signals.**

9.5 Filter cascade: 2D and 1D

NbJE. Implicitly, the presentation given by S. Mallat on the projector refers to his 2021 course, for example, sections 8.1 to 8.4. I am trying to provide a restitution of this year's presentation while keeping the notations from the 2021 course.

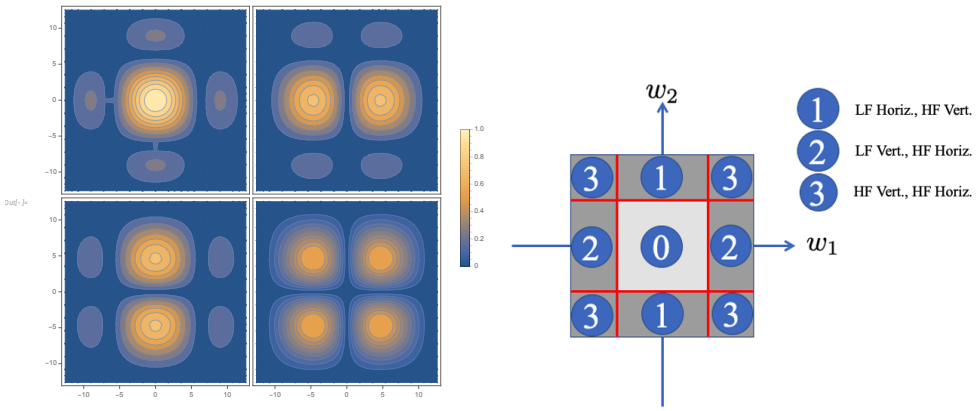


FIGURE 57 – Left: Coverage of the Fourier plane by different types of 2D wavelets constructed from 1D wavelets (low-pass filter at the top left, the other panels corresponding to directional band-pass filters; "LF": low frequency, "HF": high frequency). Right: Corresponding scheme with the low-pass filter ("0") and the various band-pass filters detecting details in three different directions. The base wavelet is the Haar wavelet ("db1").

First, we will explore the link between wavelets and neural networks, i.e., cascades of convolutions and subsampling. Regarding images, to cover the entire Fourier plane, we need to design 2D wavelets. This can be done as in Course 2021 Sec 8.4: we define a set of three wavelets $\{\psi^1, \psi^2, \psi^3\}_{j,n\in\mathbb{Z}^2}$ that detect details (horizontal, vertical, and diagonal), along with a filter ϕ that covers all low frequencies and provides an approximation at a given scale (Fig. 57).

Concerning the various approximations (low frequency) obtained by iteration [136], at step j, if h is the filter associated with ϕ, then $\tilde{h}[n] = h[-n]$

$$a_j[n] = \langle f, \phi_{j,n}\rangle = \sum_m h[m-2n]a_{j-1}[m] = (\tilde{h} * a_j)[2n] \tag{236}$$

This is indeed a cascade of convolution and downsampling by a factor of 2. Figure 58 illustrates this cascade applied twice, and it can continue as long as the image size is

136. NbJE. Recall from the 2021 Course: in 1D, the relationships between the filters and the functions ϕ, ψ in the case of a Multiresolution Analysis are as follows:

$$h(n) = \langle \frac{1}{\sqrt{2}}\phi\left(\frac{u}{2}\right), \phi(u-n)\rangle \qquad g(n) = \langle \frac{1}{\sqrt{2}}\psi\left(\frac{u}{2}\right), \phi(u-n)\rangle$$

and $\hat{h}(\omega) = \sum_{n\in\mathbb{Z}} h(n)e^{-in\omega}$, likewise for $\hat{g}$.

compatible with the filter's support.

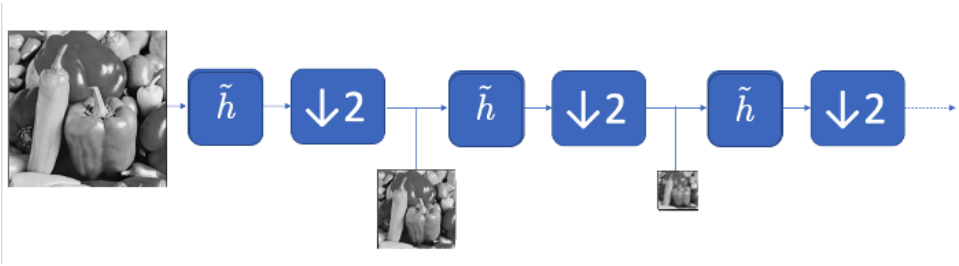


FIGURE 58 – Illustration of the successive approximations at larger scales obtained by a cascade of convolutions and downsamplings by a factor of 2.

At each stage of this low-pass filtering cascade (large scales), we lose detail information (small scales, high frequencies). If we call g the filter associated with one of the base wavelets, then at step j

$$d_j[n] = \langle f, \psi_{j,n} \rangle = \sum_m g[m-2n] a_{j-1}[m] = (\tilde{g} * a_j)[2n] \tag{237}$$

This is illustrated in figure 59. An example in the case of an image is shown in figure 60. Thus, **at each step, there is a lossless distribution between low frequencies (approximation) and high frequencies (details)**.

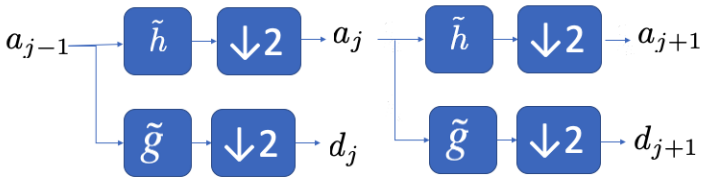


FIGURE 59 – Starting from the approximation coefficients a_{j-1}, the approximation coefficients a_j and the detail coefficients d_j are obtained through filtering and downsampling.

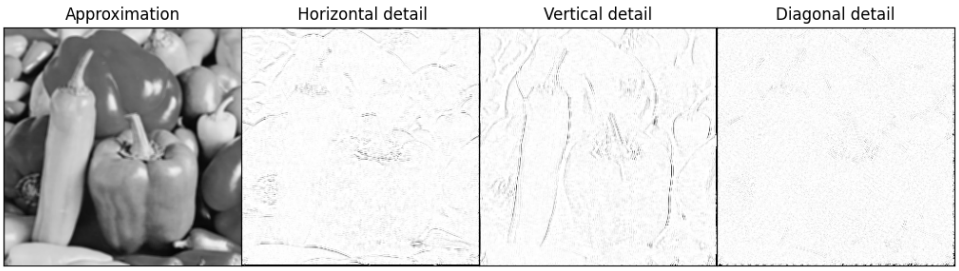


FIGURE 60 – Example of decomposition (one step) of the image from figure 58, into one approximation image and 3 detail images. The approximation image is in grayscale, while the details are in inverted grayscale. To highlight the non-zero detail coefficients (the black pixels), an appropriate scaling change was applied.

Thus, all these cascades of **convolutions and downsampling resemble the first part of a U-Net** (apart from the nonlinearities, for example). Now, how do we reconstruct the signal, which would correspond to the **second part of a U-Net**? This is done as follows:

$$a_{j-1}[n] = \sum_{m\in\mathbb{Z}} (a_j[m]h[n-2m] + d_j[m]g[n-2m]) \tag{238}$$

and illustrated in figure 61.

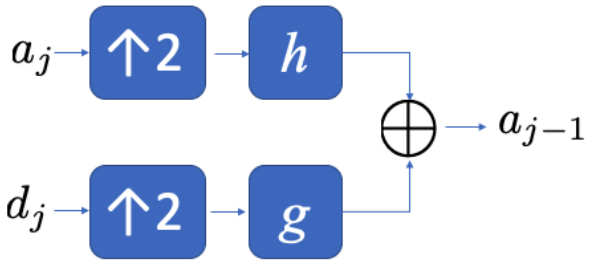


FIGURE 61 – From the coefficients a_j and d_j, the coefficients a_{j-1} can be reconstructed, with the last step showing a factor of 2 enlargement in the size of the coefficients a_j and d_j by inserting 0s before filtering.

The condition for perfect signal reconstruction is given by the theorems from the 1977-90 period. For example, it is required that in the Fourier domain, the following

relationships hold:

$$\sum_{k\in\mathbb{Z}} |\hat{\psi}(\omega - 2k\pi)|^2 = 1 \qquad \sum_{k\in\mathbb{Z}} \hat{\phi}^*(\omega - 2k\pi)\hat{\psi}(\omega - 2k\pi) = 0 \tag{239}$$

which translates into relations for the filters h and g:

$$|\hat{g}(\omega)|^2 + |\hat{g}(\omega + \pi)|^2 = 2 = |\hat{h}(\omega)|^2 + |\hat{h}(\omega + \pi)|^2 \qquad \hat{g}(\omega) = e^{-i\omega}\hat{h}^*(\omega + \pi) \tag{240}$$

The main idea is **the possibility to decompose signals into two parts, one of low frequency and the other of high frequency** (Fig. 62), then **after processing** (e.g., denoising) to proceed with **reconstruction**. This type of procedure was notably used for multiplexing, as stated by S. Mallat, who contributed to these algorithms [137].

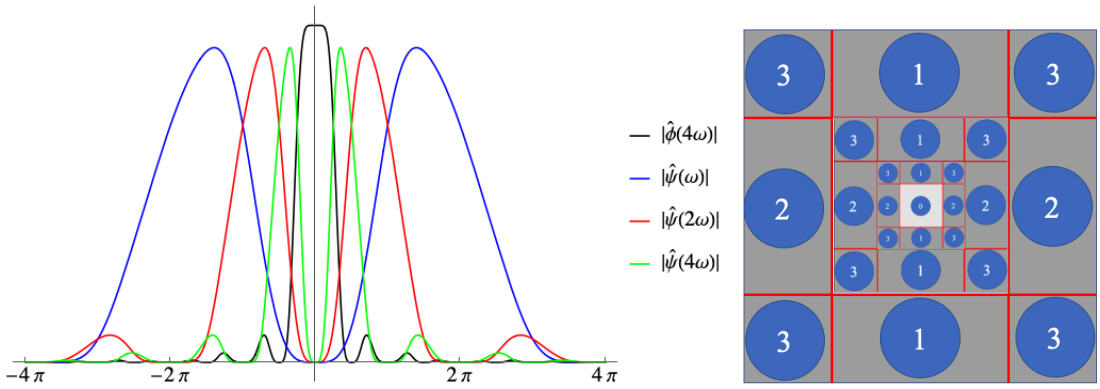


FIGURE 62 – Left: 1D coverage of the Fourier domain using a low-pass filter $\hat{\phi}_n$ that covers all the low frequencies (black, fixed scale) and a collection of band-pass filters $\hat{\psi}_{j,n}$ to cover high frequencies. This diagram is the low-frequency modification of that in figure 49, which only uses $\hat{\psi}_{j,n}$ filters to cover the entire spectrum. Right: in 2D, we use the diagram from figure 57, which contains a low-pass filter and a series of band-pass filters. In this context, the "0" part is sliced through successive iterations using this diagram, like Russian dolls.

Note that in 1D, we obtained figure 54 by using the filter cascade (low-pass, band-pass) to ultimately obtain a low-frequency approximation and a collection of high-frequency

137. NbJE. S. Mallat, *A theory for multiresolution signal decomposition: the wavelet representation*, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 11, p. 674-693, July 1989. Article available on his ENS website mentioned in the "Preface" section.

details. In the case of an image, we have the same **pyramidal decomposition** scheme, and at a fixed decomposition stage (level), we can compactly arrange the approximation image and the detail images in the 3 directions as illustrated in figure 63. It can be visually observed, as in 1D, that we have a very sparse decomposition since almost all of the detail coefficients are zero. **The non-zero pixels** (apart from the approximation ones) **concentrate along the contours**. The wavelets calculate a local contrast that depends on the orientation of the contours.

FIGURE 63 – Compact presentation of the decomposition of the image in figure 58 512×512 at step 2: the decomposition of figure 60 was repeated on the approximation image. We then obtain in the top left corner an approximation cA at step 2 (128×128) surrounded by the 3 detail images of step 2 $cD2$ (128×128), which in turn are surrounded by the 3 detail images of step 1 $cD1$ (256×256). Another type of processing was applied to highlight the detail images (black pixels), knowing that the light pixels are zero.

Now, the separation in terms of "small *vs* large" scales, corresponding to the "high *vs* low" frequencies, and "details *vs* approximation" allows us to form a new perspective on the design of wavelet bases. **Instead of thinking in terms of functions in real space, we think rather in terms of filters** in the Fourier space, and we try to find **the conditions on these filters for their cascades to produce vectors of an orthonormal basis**. This scheme is at work in a Multiresolution Analysis, as illustrated in the following theorem:

Theorem 17 *Let the filters $\hat{h}$ and $\hat{g}$ satisfy the relations 240, and suppose further that $\hat{h}(\omega) \neq 0$ on $[0, 2\pi]$, and*

$$\hat{\psi}(\omega) = \hat{g}(\omega) \prod_{p=1}^{\infty} \hat{h}(2^{-p}\omega) \tag{241}$$

then $\{\psi_{j,n}\}_{(j,n)\in\mathbb{Z}^2}$ is an orthonormal basis of $L^2(\mathbb{R})$.

We can then synthesize wavelets with properties that go beyond those of Y. Meyer, such as I. Daubechies' *compact support* wavelets, which are well-suited for fast algorithms (1D and 2D).

9.6 Wavelet Decomposition and Neural Networks

In light of the previous sections, we begin to understand how we can construct bases that resemble what is done in architectures such as U-Net. Figure 64 schematizes (top) an architecture based on wavelet filters fixed by the orthonormal basis, and (bottom) a basic neural network architecture (without non-linearity) where the convolution kernels (followed by subsamplers) are learned during optimization. **The key difference is that in a wavelet-based network, the focus is on the orthonormal basis, whereas in a network, it is the architecture that is chosen *a priori***. In this case, the choice of network architecture is fundamental [138], and is part of the **a priori information** about the problem. By using **convolution kernels** motivated by translation invariance arguments, of **small size**, since we only want to be sensitive to local contrasts, we considerably reduce the number of parameters. The network becomes easier to learn, compared to the implementation of a *Multi-Layer Perceptron* (MLP) network, which has no internal structures. **The filter cascade** in the same way as wavelet decomposition allows a neural network to address the problem at **different scales**.

138. NbJE. Course 2020.

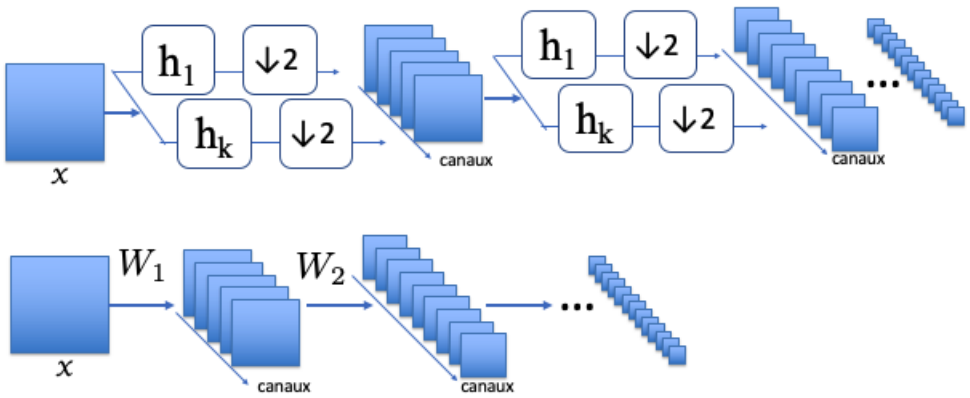


FIGURE 64 – Schematic views of two architectures. Top: application of wavelet filters h fixed by the choice of orthonormal bases followed by subsamplers. Bottom: application of learned convolution kernels W_i (followed by unrepresented subsamplers) during optimization of a network with a fixed architecture. Non-linearities between layers are not depicted.

9.7 Image Denoising: Rethinking Wavelets

Having a sparse representation in a 2D orthonormal basis, we can apply the same thresholding technique developed in 1D in section 9.3 to the signal in figure 56 in order to denoise an image. An example [139] is shown in figure 65.

139. NbJE. Two notebooks available in the GitHub repository allow for the implementation of this type of image denoising: `Wavelet_image.ipynb` and `Wavelet_stationnary_image.ipynb`.

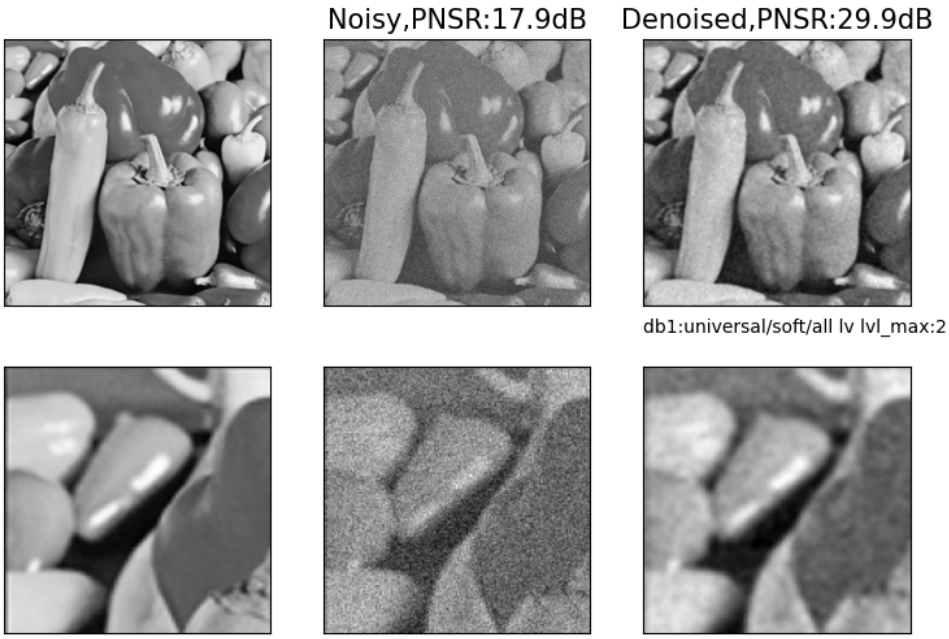


FIGURE 65 – Denoising an image by applying a hard thresholding of Donoho & Johnstone to the coefficients of a (stationary) decomposition using a Haar wavelet ("db1") from the `PyWavelets` library.

The next question concerns the **optimality of denoising** in the ongoing quest for obtaining the *score* using the method from section 7.1.2. **The answer is no**, the 2D wavelet representation developed in section 9.5 does not provide an optimal denoiser. Something is missing that neural networks capture: **geometry**. To understand this, we need to uncover why the algorithms used until now fail for images.

Let's revisit the theorems 15, 16 from section 9.4 established in the linear and nonlinear frameworks in 1D. If we now place ourselves in 2D within the linear framework:

Theorem 18 *If $x(t_1, t_2)$ is uniformly Lipschitz α (no discontinuities here) and if we keep only M coefficients that determine an approximation x_M, then*

$$(1) \qquad \exists C > 0 \quad \|x - x_M\|^2 \leq CM^{-\alpha} \qquad (242)$$

and the thresholding risk $r_s(x)$ is bounded as

$$(2) \qquad \exists C > 0 \quad r_s(x) \leq C|\log \sigma|\sigma^{2\alpha/(\alpha+1)} \tag{243}$$

and the exponents of M and σ are optimal.

Propositions (1) and (2) adapt the decay rate to the problem dimensionalty Moreover, in the nonlinear framework, it is shown that theorem 16 remains valid in 2D, meaning there is **optimality of the thresholding algorithm when there is a finite number of discontinuities**.

What changes is the case where the number of discontinuities is infinite, which is natural in images, since discontinuities form the contours of objects.

Theorem 19 *If $x(t_1, t_2)$ has **discontinuities along curves** C^α and is uniformly Lipschitz α elsewhere, then the results* (1) *and* (2) *from the previous theorem are no longer valid, and we have*

$$\exists C > 0 \quad \|x - x_M\|^2 \leq CM^{-1} \tag{244}$$

What this tells us is that if we add a shape with regular edges on a regular background, it is no longer true that the number of wavelet coefficients added is negligible. In figure 66, we can clearly see that **wavelet coefficients concentrate along the edges of the shape and are numerous**. Indeed, at scale $s = 2^j$, the number is roughly $M_j = L \times 2^{-j}$, where L is the length of the contour.

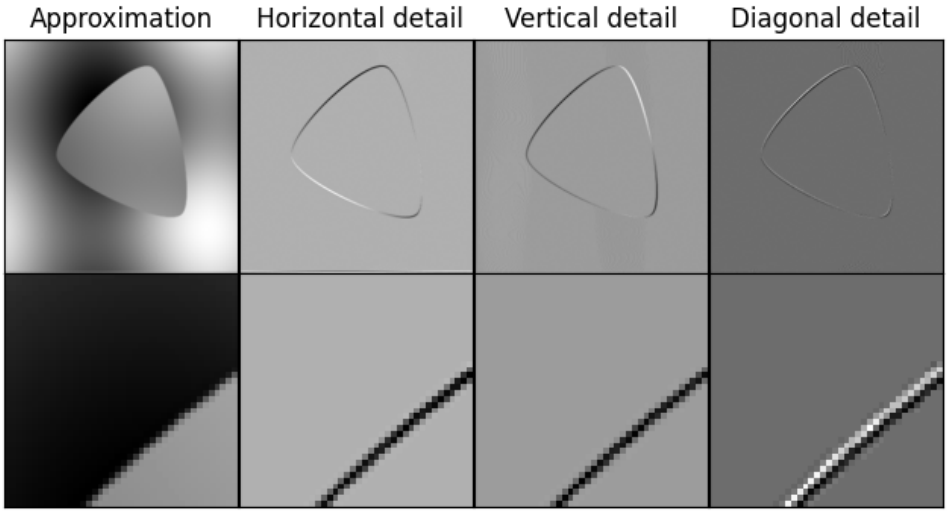


FIGURE 66 – Wavelet decomposition ("db1", one level) of an image consisting of a regular background and a shape itself regular, enclosing another equally regular background. Bottom: a simple zoom on the edge of the shape in the upper left part.

In conclusion, a thresholding algorithm applied to such images is not optimal; however, it is noteworthy that the coefficients concentrate along the singularities. Therefore, to improve things and obtain result (2), we must opt for a different decomposition or other types of wavelets. In any case, we need to use **the geometry of the contours** of the shape. This is the aspect where neural networks provide a significant improvement.

9.8 Taking Geometry into Account

S. Mallat tells us that in the 2000s-2010s, a series of results [140] improved upon the *classical* 2D wavelet decomposition presented in section 9.5. The key idea behind these approaches is that instead of finding an orthogonal basis that aims to be applicable in all cases, **the basis must be adapted according to the signal**, in order to achieve the most sparse representation possible.

140. NbJE. See for example (and the references included, including those of Le Pennec who developed the first bandlet bases): S. Mallat and G. Peyré, *A Review of Bandlet Methods for Geometrical Image Representation, Numerical Algorithms*, Vol. 44(3), p. 205-234, March 2007, pdf available on S. Mallat's ENS website. Also see https://hal.science/hal-00359740.

Focusing on the wavelet detail coefficients from figure 66, we notice that they exhibit a rather slow variation along the contour and a much faster variation perpendicular to the contour.

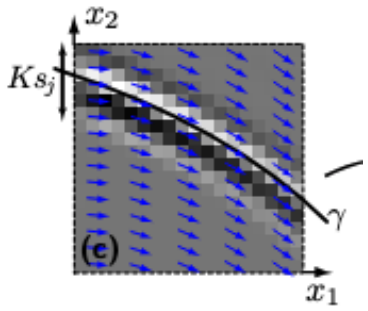


FIGURE 67 – Wavelet detail coefficients exhibit a slow variation along the curve defining a contrast discontinuity in the example from figure 66. Extracted from the article in the footnote 140.

How can we account for this directional regularity? One could consider defining **a wavelet transform along the contour direction**. But this becomes much more complicated, as S. Mallat explains (Fig. 68):

1. One must first perform a "classic" wavelet decomposition;
2. For each detail image, an adaptive segmentation of the contours is needed, for example using dyadic squares of variable sizes;
3. For each square, the contour's orientation field must be recognized;
4. Finally, new coefficients are calculated in an orthonormal base of "**bandlets**". The basis vectors oscillate like wavelets, but are elongated along the contours.

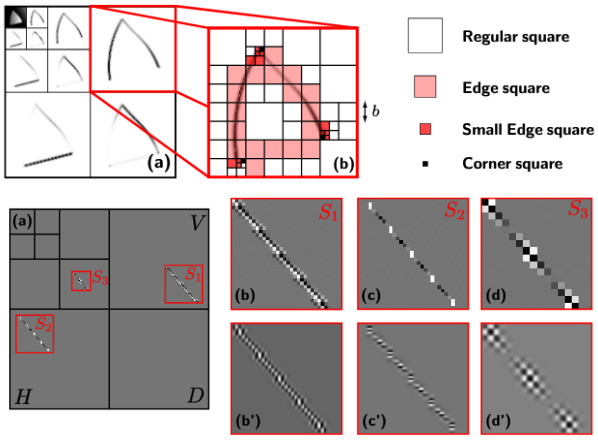


FIGURE 68 – Bandlet decomposition procedure. Top images: classic wavelet decomposition (left), and segmentation of a detail image into dyadic squares of variable sizes adapting to the approaching contour discontinuities (right). Bottom images: bandlet decomposition adapting to the contours. Extracted from the article in the footnote 140.

The inefficiency of "classic" wavelets is addressed by using **supports of bandlets that stretch along the contours**. But for that, the geometry of the contour must be known, and thus **the basis itself depends on the signal**. We then have the following results using C^α image analysis (both the backgrounds and the contours are C^α):

Theorem 20 ***Korostelev, Tsybakov*** *(1993)*
For these C^α images, the optimal denoising estimator has a decay according to

$$\|x - \hat{x}\|^2 \sim \sigma^{2\alpha/(1+\alpha)} \tag{245}$$

And in the case of a decomposition into bandlets:

Theorem 21 ***Le Pennec, Peyré, Mallat*** *(2004)*
For these C^α images, the estimator obtained from thresholding the coefficients of the

bandlet decomposition has a decay according to

$$\|x - \hat{x}\|^2 \sim (|\log \sigma|\sigma)^{2\alpha/(1+\alpha)} \tag{246}$$

Thus, we obtain the optimum possible, except for a factor of $\log \sigma$. But what is the limit of this kind of "bandlet and thresholding" algorithm? First, as S. Mallat tells us, the algorithm as described above is complicated, and second, it only works on rather simple images (such as C^α images). Neural networks not only adapt perfectly to the geometry but also perform much better than the previously described algorithms.

9.9 Neural Networks and Image Geometry

9.9.1 What does the network perform?

Let us return to the framework of obtaining the *score* from a denoising network (Sec. 7.1.2), i.e.

$$\hat{x}(x_t) = x_t + \sigma_t^2 \nabla_x \log p_t(x_t) \tag{247}$$

Later, we simplify the notation by setting $\sigma_t = 1$, knowing that x_t is actually x_{σ_t}. If the neural network (e.g., U-Net) is built using only **layers** (convolution, batchnorm, ...) **without any biases** and **ReLUs**, then the output $\hat{x}(x_t) = f(x_t)$ of the network is locally linear, so $\hat{x}(x_t) = \nabla_x f(x_t)\, x_t$. Thus,

$$\hat{x}(x_t) = (Id + \nabla^2 \log p_t)\, x_t \tag{248}$$

where ∇^2 is the Hessian operator. This manipulation allows for a more straightforward interpretation of what happens. Indeed, since the operators Id and the Hessian are symmetric, we can diagonalize the operator that applies to x_t in an orthonormal basis, denoted $\{\psi_k\}_k$, with eigenvalues $(\lambda_k)_k$ such that

$$\hat{x}(x_t) = \sum_k \lambda_k \langle x_t, \psi_k \rangle \psi_k \tag{249}$$

Note the similarity with the formulation given by expression 215, where the $h[k]$ (e_k) are replaced by the λ_k (ψ_k). Since **the orthonormal basis** $\{\psi_k\}_k$ diagonalizes $\nabla^2 \log p_t$, it **adapts to each signal** (note that the network is optimized for a specific type of images).

If we denote $x_t = x + z_t$, meaning that the noisy signal is the sum of the deterministic signal x and the random noise z_t, then

$$\lambda_k \langle x_t, \psi_k \rangle = \lambda_k (\langle x, \psi_k \rangle + \langle z_t, \psi_k \rangle) \tag{250}$$

If the denoiser is optimal, it will select the coefficients $\langle x, \psi_k \rangle$ that are above the noise level (Sec. 8.1), as shown in Figure 69.

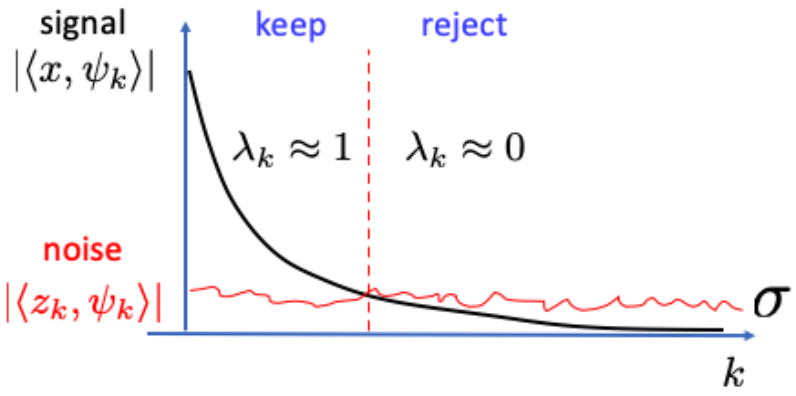


FIGURE 69 – Selection of coefficients in the basis obtained by the network according to the eigenvalues of the operator.

Now, the network minimizes the quadratic error $\|x - \hat{x}\|^2$, and it automatically adapts the basis such that the total number of coefficients retained is as small as possible. In other words, **it discovers a basis in which the signal representation is as sparse as possible**. This is manifested by a rapid decay of the coefficients according to $\langle x_t, \psi_k \rangle \propto k^{-\beta}$ with a large value of β. **Ultimately, the calculation of the score can be seen as the calculation of an optimal orthonormal basis that diagonalizes the Hessian.**

9.9.2 Numerical Experiment Results

S. Mallat presents results from the article [141] by Z. Kadkhodaie, F. Guth, E. P. Simoncelli from 2024, of which he is a co-author. For reference, the U-Nets used in the

141. NbJE. See the footnote 28.

subsequent experiments were optimized on a sufficient number of training images to be in a generalization regime (Sec. 7.1.5). Each time, after network optimization, we provide the network with a sufficiently noisy image (validation image) ($x_\sigma = x + z_\sigma$), and in practice [142] $f(x_\sigma)$ is output such that $\hat{x} = x_\sigma - f(x_\sigma)$ (the network learns the noise); the Hessian H of f is then computed, and an SVD decomposition of $Id - H$ is performed to extract the vectors $\{\psi_k\}_k$ and eigenvalues $(\lambda_k)_k$. Let's see what this gives on a few examples.

9.9.2.1 Disk Images

Consider a case where the optimal basis can be calculated independently: it involves images consisting of a disk with a variable radius filled with a uniform background, randomly placed on another uniform background. In the limit of low noise, it is known that there are 5 basis vectors corresponding to the number of degrees of freedom in the problem (2 for the position of the disk, 1 for its radius, and 2 for the backgrounds, inside and outside). These are shown in Figure 70 (top row) for the chosen image example. These are the 5 basis vectors corresponding to the Lie algebra (translation and dilation of the circle) in the tangent plane to the hypersurface in which the image evolves. At the same time, we can observe the basis vectors found by the network corresponding to significant eigenvalues (Fig. 70 bottom row).

142. NbJE. The authors provide a companion repository on Github where you can find the Python codes.

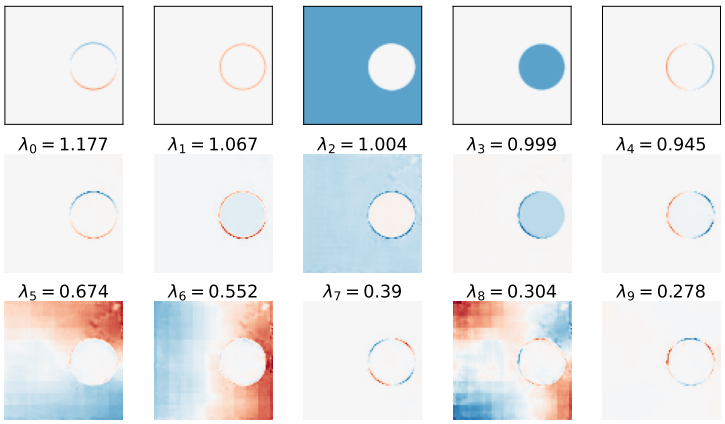


FIGURE 70 – Top row: the 5 vectors of the optimal basis calculated independently of the network. Middle and bottom rows: the first 10 vectors of the basis found by the U-Net. Note the correspondence between the first 5 and those from the top row.

It can be observed that **the network has found the 5 most important eigenvectors in order**, and then, due to the noise in the provided image, it finds vectors that describe the outer background and the circle, while respecting the geometry of the latter, and with somewhat higher oscillation frequencies.

Now, returning to the questions raised in Section 7.1.5 about *generalization* and *score estimation bias*, generalization was addressed with the *2-model test*: once the models are trained with enough data, they generate the same images of faces and bedrooms when provided with the same noise image. However, the question of bias remained: do we estimate the correct score? In the case of simple disk images, we know that the network correctly computes the right basis.

9.9.2.2 C^α Images

We can proceed [143] in the same way with C^α images (Lipschitz backgrounds α separated by a curve that is also Lipschitz α). The dimensionality of the space is now infinite. The network then finds a basis whose main vectors are shown in Figure 71. Again, the vectors are oscillating functions in the backgrounds that respect the boundary, and oscillating along the boundary contour in a manner similar to bandlets. However, the network has found a **basis perfectly adapted to the geometry**.

143. NbJE. Note that the U-Net network is re-optimized for each type of image to be processed.

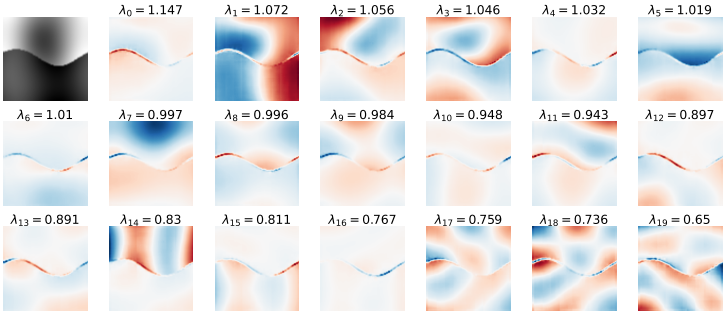


FIGURE 71 – Main vectors of the basis found by the U-Net in the case of C^α images ($\alpha = 4$).

We can test the optimality by estimating the scaling law from Theorem 20. In the case where we estimate the PSNR of the denoised image versus the PSNR of the image with noise variance σ^2, the optimal slope is $\alpha/(1+\alpha)$. The result from the network is shown in Figure 72. It can be observed that **the network achieves the optimal estimation**. Thus, applied in the case of generation by score estimation through the network, not only is it in a generalization regime, but also **it estimates the score without bias**. **The network has been able to adapt the basis to the geometry and perform thresholding for an optimal result**.

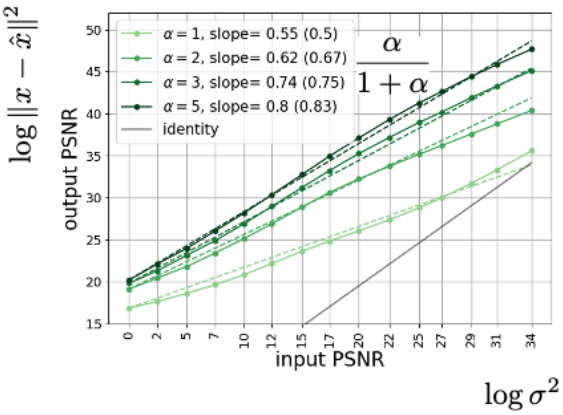


FIGURE 72 – Denoising result of C^α images by the optimized network. The estimated slopes are compared to the optimal values (in parentheses) given by the law $\alpha/(1+\alpha)$.

9.9.2.3 Face Images

Building on the success of the previous two tests, we can proceed in the same way with much more complex images, such as those of faces. Figure 73 shows the main vectors found by the newly optimized network. Again, we observe that the vectors are oscillating functions that perfectly respect the geometry of the face, hair, etc. In fact, the network uses **convolutions**, so we expect oscillating functions with Fourier in mind, however **the non-linearities** (ReLU) constrain these functions to respect the geometry.

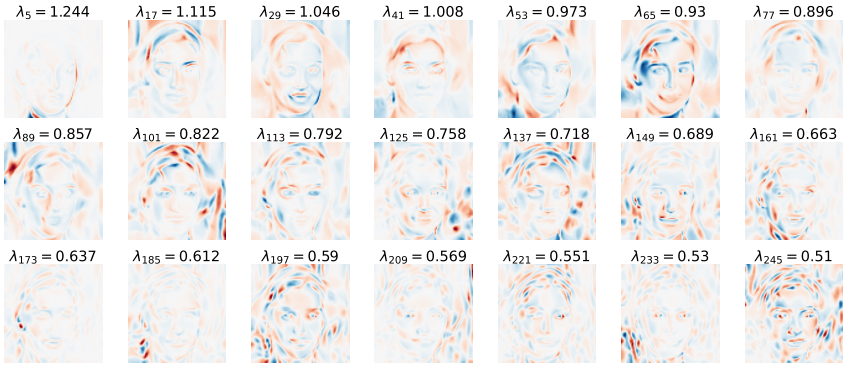


FIGURE 73 – Main vectors of the basis found by the U-Net.

However, *stricto sensu*, we do not have results (like in the Korostelev-Tsybakov theorem) to know whether the basis found is optimal or not.

9.10 Summary

S. Mallat provides a summary and opens up new perspectives. First, we have a perfect framework for mathematics with these diffusion methods (*score-based*). When using GANs and Normalizing Flows, it is ultimately very difficult to describe the nature of the transport. In contrast, diffusion methods are built from a **single mathematical object: the score**. The question we have tried to answer is: can a neural network optimally estimate this score? What we have seen in this session is that **estimating the score is equivalent to estimating the basis that produces the most sparse representation possible** which, by applying a threshold (through the network), gives the optimal denoiser. The numerical tests show that this works very often, and in any case, the use of the network yields

performance far superior to all algorithms developed, such as those mentioned in the case of bandlets. The advantage of the latter lies in the fact that we have theorems in certain cases, but the cases in question remain simple compared to images of faces, bedrooms, etc.

The research questions in this field are: **what is the nature of the bases found by the network, what classes of scores are accessible, what type(s) of regularity must the signal have**? According to S. Mallat, we must realize that this method will not work in all cases, simply because the number of parameters of the network does not explode with the dimensionality, so its adaptation capacity is somewhat limited.

It is worth noting that almost all current research treats the neural network as a "black box" and focuses, for example, on changing the differential equation of the transport. **In this course, we have opened the box to understand the properties that give the correct score.** In this research field, there is also the issue of learning with small batch sizes. We saw (Sec. 7.1.5) that the generalization regime is indeed reached with batches of a sufficiently large size, typically $O(10^5)$ for 80×80 images and a U-Net with 7.5 million parameters. However, for problems such as those in medicine, where the number of examples is small compared to the available high-resolution images, it is almost certain that these diffusion algorithms are not in the generalization regime. There is, therefore, a strong suspicion of memorization. But, there is a way forward: **we must build neural networks with a significantly reduced number of parameters**. It is then crucial to understand the architecture of these networks to know which parameters can be fixed *a priori* and which ones must be optimized. **Thus, opening the black box has stakes not only from a mathematical perspective but also from the perspective of very concrete applications.**