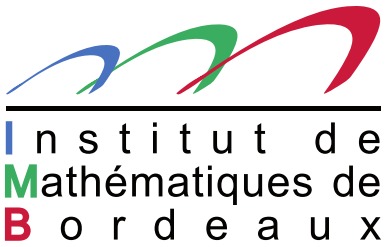


Université de Bordeaux
Institut de Mathématiques de Bordeaux UMR CNRS 5251
École doctorale Mathématiques et Informatique

Document de Synthèse

présenté par

Arthur LECLAIRE

dans le but d'obtenir

l'Habilitation à Diriger des Recherches

Spécialité : Mathématiques appliquées

Synthèse et Restauration d'Images par Transport Optimal et Apprentissage Profond

Présenté le 14 novembre 2023 devant le jury composé de

Hermine Biermé	Université de Tours	Examinatrice
Laure Blanc-Féraud	CNRS, Université Côte d'Azur	Examinatrice
Pierre Chainais	Centrale Lille Institut	Rapporteur
Bruno Galerne	Université d'Orléans	Invité
Lionel Moisan	Université de Paris	Invité
Jean-Michel Morel	ENS Paris-Saclay, CityU Hong-Kong	Examinateur
Gabriel Peyré	CNRS, Université Paris-Dauphine	Rapporteur
Nelly Pustelnik	CNRS, ENS Lyon	Rapportrice

Table des matières

Remerciements

Je souhaite remercier chaleureusement les membres du jury pour toute l'attention qu'ils ont portée à mes travaux. Je remercie profondément Pierre Chainais, Gabriel Peyré et Nelly Pustelnik de m'avoir fait l'honneur d'être rapporteurs de cette habilitation à diriger des recherches. Je remercie Hermine Biermé, Laure Blanc-Féraud et Jean-Michel Morel d'avoir accepté, avec beaucoup d'enthousiasme, de participer à ce jury. Je voudrais dire ici l'immense estime que je porte à tous ces collègues, qui font rayonner le spectre des mathématiques appliquées au traitement d'images, et contribuent à rendre cette discipline si riche et passionnante.

Bruno Galerne et Lionel Moisan ont accompagné le début de ma carrière universitaire et ont cherché à me transmettre une vision scientifique ambitieuse, sincère et exigeante. Je me réjouis qu'ils prennent part à cette soutenance. Au terme d'une dizaine d'années de recherche, mobilisant, à la croisée des chemins, des intelligences artificielles ou non, je suis heureux de présenter ici mes avancées qui, je l'espère, s'inscrivent dans cette vision scientifique. Bon nombre des problématiques étudiées avaient déjà pris racine lors de mon parcours doctoral et je mesure, avec humilité et gratitude, l'étendue de l'inspiration qui m'a été généreusement transmise. Je m'attache à la faire perdurer et à la transmettre à mon tour, que ce soit dans ma recherche ou mes enseignements.

Mes travaux de recherche ont bénéficié du soutien de nombreux collègues avec qui j'ai pu collaborer ou échanger et à qui je souhaite exprimer toute ma reconnaissance. Sans chercher à les lister exhaustivement (et que les omis m'en excusent), je souhaite remercier, par ordre chronologique, Julien Rabin, Agnès Desolneux, Mila Nikolova, Jean-Michel Morel, Julie Delon, Alain Durmus, Hermine Biermé, Philippe Carré, Edoardo Provenzi, Alasdair Newson, Jean-François Aujol, Yann Traonmilin, Camille Male, Jérémie Bigot, Bernard Bercu, Baudouin Denis de Senneville, Aurélie Bugeau, Laurent Facq, Elsa Cazelles, Andrés Almansa, Antonin Chambolle, Yann Gousseau et Florence Tupin.

Je souhaite remercier plus particulièrement Nicolas Papadakis, toujours volontaire pour écouter ou relire de façon précise et bienveillante, et avec qui j'ai eu la chance de collaborer étroitement pendant mes cinq années à l'Université de Bordeaux. J'ai pu maintes fois bénéficier de ses précieux conseils, de ses commentaires constructifs et de son recul scientifique, notamment dans la préparation de ce manuscrit. Je lui suis infiniment reconnaissant pour l'énergie qu'il a su insuffler à nos projets communs.

Je mesure aussi la chance d'avoir pu participer à l'encadrement de deux thèses préparées par des étudiants remarquables, Valentin de Bortoli et Samuel Hurault. Je les remercie pour ces belles aventures scientifiques partagées, nourries par une curiosité scientifique et une puissance de travail considérables. J'espère que les deux chapitres résumant leurs travaux sauront refléter tout le plaisir que j'ai pris à les accompagner dans la poursuite de ces axes de recherche. Je suis aussi heureux d'avoir pu tisser des liens pleins d'avenir avec d'autres jeunes chercheurs : Claire Launay,

Paul Freulon et plus récemment Marien Renaud.

Les travaux présentés dans ce manuscrit ont été menés alors que j'étais en poste à l'École Normale Supérieure de Cachan (Laboratoire CMLA), puis à l'Université de Bordeaux (Laboratoire IMB), environnements dans lesquels j'ai pu mener mes travaux confortablement, avec un grand soutien. Je remercie les directions respectives de ces structures d'enseignement et de recherche : Alain Trouvé, Nicolas Vayatis, Christine Bachoc, Marc Arnaudon, Karim Kellay et Vincent Koziarz. J'adresse de chaleureux remerciements à tous les collègues que j'ai eu le plaisir de côtoyer en équipe pédagogique, et notamment Frédéric Pascal, Claudine Picaronny, Sandrine Dallaporta, Laurent Herr, Robert Deville, François Clautiaux, Vincent Couallier, Adrien Richou et Michel Bonnefont. Je souhaite aussi remercier les personnes chargées de la gestion administrative et financière du CMLA et de l'IMB ainsi que la cellule info de l'IMB, qui ont à coeur de faciliter la vie des enseignants-chercheurs (et le succès de notre *workshop* sur les méthodes *Plug-and-Play* en Décembre 2022 leur doit beaucoup !).

Enfin, je souhaite exprimer ma profonde gratitude envers mes proches, mes amis, ma famille, ceux qui m'ont accompagné et ceux qui m'accompagnent aujourd'hui.

Chapitre 1

Introduction

Les travaux présentés dans ce document de synthèse s'inscrivent dans le domaine des mathématiques appliquées au traitement des images, et traitent plus précisément de synthèse de textures par l'exemple, de modèles génératifs d'images, et de restauration d'images. Nos avancées sont basées sur des outils mathématiques liés à la modélisation aléatoire, au transport optimal, aux méthodes de Monte-Carlo, à l'optimisation et à l'apprentissage profond.

Prenant leur source à la fin de ma thèse au laboratoire MAP5 de l'Université Paris Descartes, ces travaux ont été menés entre 2014 et 2018 au laboratoire CMLA de l'École Normale Supérieure de Cachan puis entre 2018 et 2023 à l'Institut de Mathématiques de Bordeaux. Ils ont bénéficié des apports de nombreux collègues et étudiants avec qui j'ai eu le plaisir de collaborer. Certains axes ont été poursuivis au cours des thèses de deux doctorants que j'ai pu co-encadrer : Valentin de Bortoli (entre 2017 et 2020 à l'École Normale Supérieure de Cachan Paris-Saclay), et Samuel Hurault (entre 2020 et 2023 à l'Institut de Mathématiques de Bordeaux).

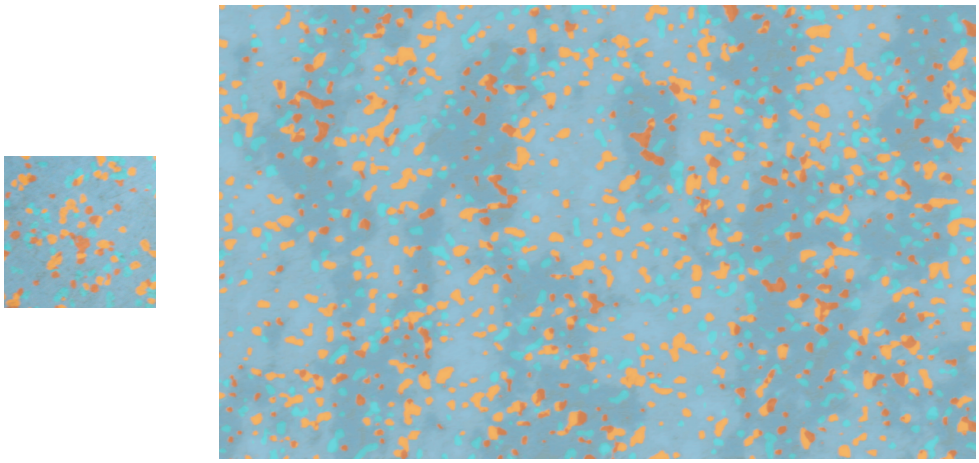
J'ai fait le choix de concentrer ce document de synthèse sur mes contributions principales en lien avec la modélisation des textures et la restauration d'images, que je vais maintenant résumer. J'espère que ce choix permettra de mettre en lumière un fil conducteur de mes travaux, consistant à tirer profit à la fois de méthodes d'optimisation et d'échantillonnage aléatoire, de façon à produire des algorithmes efficaces et stables pour la synthèse et la restauration d'images.

1.1 Synthèse de Textures par Transport Optimal de Patches

Le Chapitre 2 porte sur l'utilisation du transport optimal dans l'espace des patches pour la synthèse de textures [J7, J12].

L'objectif de la synthèse de textures par l'exemple est de concevoir des algorithmes prenant en entrée une texture exemple et produisant en sortie une nouvelle texture, possiblement beaucoup plus grande, et ayant les mêmes caractéristiques perceptuelles (voir Fig. 1.1). On se limitera ici au cas des textures homogènes définies sur une grille discrète. Étant donné une texture exemple $u_0 : \Omega \longrightarrow \mathbb{R}^d$ à d couleurs définie sur un rectangle $\Omega \subset \mathbb{Z}^2$, on cherche donc à proposer un modèle de champ aléatoire stationnaire $V : \mathbb{Z}^2 \longrightarrow \mathbb{R}^d$ qui est facilement simulable, et dont les réalisations ont le même aspect textural que celui de u_0.

De nombreuses solutions ont été proposées pour répondre à ce problème de synthèse de textures [158, 120]. Parmi celles-ci, on peut d'abord compter les méthodes paramétriques [118, 51] qui visent à produire une image respectant une



Original Synthèse avec le modèle Texto

FIGURE 1.1 – **Synthèse de textures par l'exemple.** On montre à gauche une texture exemple 256 × 256, et à droite une synthèse 1280 × 768 obtenue avec le modèle Texto.

collection donnée de statistiques importantes pour la perception. Le modèle paramétrique le plus simple est le modèle gaussien [50], qui permet de conserver exactement l'auto-corrélation spatiale de la texture, et donc aussi le contenu fréquentiel. On peut distinguer également les méthodes à patches [48, 47, 85] qui composent la synthèse en faisant des recopies locales astucieuses de la texture exemple. Nos travaux visent à concilier ces deux lignées, en exploitant des transformations locales qui induisent une cohérence statistique globale. Plus précisément, nous chercherons à faire en sorte que les distributions de patches (imagettes de taille $\mathsf{w} \times \mathsf{w}$) de u_0 et V soient proches, de façon à ce que les motifs de taille $\mathsf{w} \times \mathsf{w}$ aient la même fréquence d'apparition dans u_0 et dans V.

En collaboration avec Bruno Galerne et Julien Rabin, nous avons conçu des modèles de textures structurées en enrichissant le modèle gaussien à l'aide de transformations locales. Pour contraindre la distribution des patches des images synthétisées, nous avons utilisé des applications de transport optimal dans l'espace des patches. Plus précisément, étant donné une synthèse courante U (par exemple le modèle gaussien) de distribution de patches μ, et la texture exemple u_0 avec distribution empirique de patches ν, on considère une application T qui résout le problème de transport optimal entre μ et ν :

$$\inf_T \int_{\mathbb{R}^D} c(x, T(x)) d\mu(x), \tag{1.1}$$

où $c(x, y) = \|x - y\|^2$ est le coût quadratique sur l'espace des patches $\mathbb{R}^D$, et où l'infimum est pris sur l'ensemble des applications mesurables $T : \mathbb{R}^D \longrightarrow \mathbb{R}^D$ telles que la mesure image $T_\sharp \mu$ vaut ν. Ici, l'intérêt de ces applications de transport optimal par patch est qu'elles permettent de modifier très légèrement la texture localement de façon à renforcer la cohérence statistique globale.

Nous avons exploité en particulier le cadre de transport optimal semi-discret,

dans lequel μ est à densité sur $\mathcal{X} \subset \mathbb{R}^D$ et ν supportée par un ensemble fini $\mathcal{Y}$ de cardinal J. Pour ce cas, plusieurs auteurs [4, 81] ont montré que la solution a la forme suivante :

$$T_\psi(x) = \operatorname*{Argmin}_{y \in \mathcal{Y}} c(x,y) - \psi(y), \tag{1.2}$$

où $\psi \in \mathbb{R}^{\mathcal{Y}}$ est une solution d'un problème d'optimisation concave en dimension finie. Les préimages de cette application

$$\mathtt{L}_\psi(y) = \{\ x \in \mathcal{X} \mid \forall y' \neq y,\ c(x,y) - \psi(y) < c(x,y') - \psi(y')\ \} \tag{1.3}$$

définissent un diagramme de Laguerre illustré sur la Fig. 1.2, et l'application T_ψ est bien définie en dehors des cas d'ex aequo obtenus sur l'interface

$$A_\psi = \mathcal{X} \backslash \bigcup_{y \in \mathcal{Y}} \mathtt{L}_\psi(y). \tag{1.4}$$

On peut montrer que la résolution du transport optimal entre μ et ν revient exactement à trouver le diagramme de Laguerre qui divise la masse de μ en accord avec les poids de la mesure cible ν. Un avantage majeur de ces applications de transport semi-discret pour la synthèse de textures est qu'elles constituent une simple généralisation des assignements au plus proche voisin (NN pour *nearest neighbor*) déjà largement utilisés dans les méthodes à patches.

Dans le Chapitre 2, nous présenterons trois modèles de textures consistant à appliquer des transformations par patches à plusieurs résolutions. Ces trois

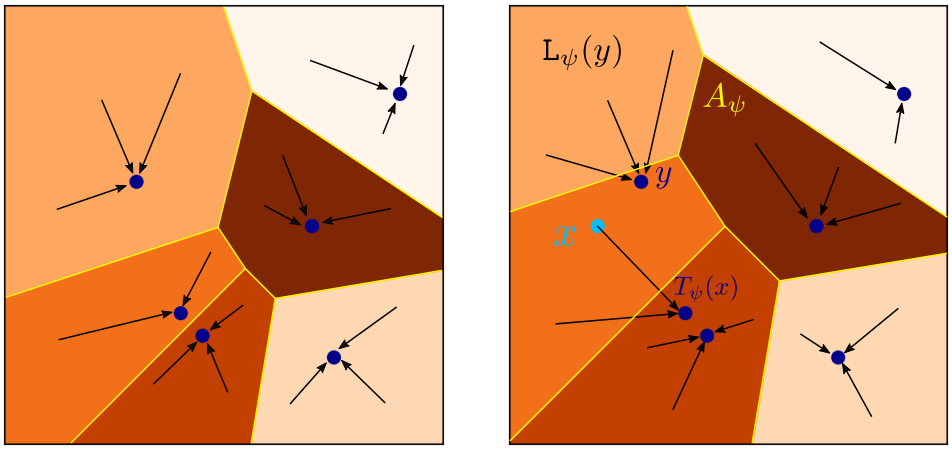


Diagramme de Voronoi Diagramme de Laguerre

FIGURE 1.2 – **Applications de transport semi-discret T_ψ.** On affiche ici deux partitions colorées correspondant à des diagrammes de Laguerre associés à un ensemble $\mathcal{Y}$ à 6 points (en bleu foncé), et pour $c(x,y) = \|x-y\|^2$ sur $[0,1]^2$. À gauche, pour $\psi = 0$ on retrouve le diagramme de Voronoi classique. À droite, on voit le diagramme de Laguerre associé à un $\psi \in \mathbb{R}^{\mathcal{Y}}$. L'application T_ψ est dessinée avec des flèches noires. L'interface $A_\psi = \mathcal{X} \backslash \bigcup_{y \in \mathcal{Y}} \mathtt{L}_\psi(y)$ est surlignée en jaune.

modèles sont basés sur différentes approximations du transport optimal semi-discret. Dans [J7], en travaillant avec les patches 3×3 de façon multi-résolution, nous avons introduit un premier modèle, baptisé Texto, qui permet de synthétiser des textures semi-structurées avec de bonnes garanties théoriques (stationnarité, indépendance à longue portée). Ce premier modèle étant limité à de petits patches (à cause du coût de l'algorithme stochastique de calcul du transport optimal), dans [J12] nous avons proposé un nouveau modèle TextoML, obtenu en approchant le transport optimal semi-discret à l'aide d'applications de transport multi-couche. Ce modèle rivalise avec les méthodes de l'état de l'art pour la synthèse de très grandes textures structurées, tout en étant moins gourmand en mémoire et en temps de calcul. Enfin, en collaboration avec Julie Delon et Agnès Desolneux [C16], nous avons exploité une nouvelle formulation du transport optimal spécifique aux mélanges de gaussiennes (GMM pour *Gaussian Mixture Model*), pour définir le modèle TextoGMM, qui peut être estimé plus efficacement. Un atout de tous ces modèles est que, une fois le transport optimal estimé, la synthèse nécessite seulement d'appliquer un petit nombre de transformations simples à tous les patches, ce qui peut se faire de façon rapide et parallélisée.

1.2 Réseaux Génératifs Wasserstein

Le Chapitre 3 concerne l'utilisation du transport optimal régularisé dans l'apprentissage de réseaux génératifs servant à la synthèse d'images [J14, J15, Js2].

Simultanément à nos travaux sur le modèle Texto, de nouveaux modèles de textures ont été proposés [13] profitant des avancées obtenues sur l'apprentissage de réseaux génératifs. Les réseaux génératifs [59, 2] forment une classe de réseaux de neurones artificiels qui répond à des problèmes de synthèse de signaux ou d'images. Plus précisément, si l'on dispose de données $y_1, \dots, y_J$, estimer un réseau génératif consiste à optimiser une fonction g_θ de telle sorte que les échantillons de la variable aléatoire (v.a.) $g_\theta(Z)$ (où Z est une v.a. suivant un modèle de bruit fixé) ne puissent pas être statistiquement discriminés en comparaison des données $y_1, \dots, y_J$.

L'apprentissage d'un réseau génératif Wasserstein (WGAN) se formule mathématiquement par le problème d'optimisation

$$\inf_{\theta \in \Theta} W_\lambda(\mu_\theta, \nu) \tag{1.5}$$

où W_λ est le coût de transport optimal avec régularisation entropique (avec paramètre $\lambda \geqslant 0$) entre la loi empirique ν associée aux données $y_1, \dots, y_J$, et la loi μ_θ du modèle génératif, c'est-à-dire la loi de $g_\theta(Z)$ et où g_θ est un réseau de neurones paramétré par $\theta \in \Theta$. Entre 2019 et 2021, j'ai porté le projet Rémoga du GdR ISIS portant sur l'étude théorique et pratique de l'estimation des WGAN. Ce projet a été nourri par une collaboration avec Nicolas Papadakis et Julien Rabin, avec qui j'ai co-encadré le post-doctorat d'Antoine Houdard, et également par une collaboration Paul Freulon, doctorant co-encadré par Jérémie Bigot et Boris Hejblum.

Un premier axe [J15] du projet Rémoga, présenté dans la Section 3.1 consistait à étudier le gradient de la fonction de coût (1.5) pour préciser le résultat de l'article

fondateur [2]. Nous avons donné des conditions suffisantes validant l'expression du gradient dans le cadre semi-discret, ainsi qu'un contre-exemple en cas discret illustrant les limites de la formule trouvée. Par ailleurs, ces résultats éclaircissent le comportement de l'algorithme d'optimisation alternée régulièrement utilisé dans la littérature pour l'apprentissage d'un WGAN [2, 129], et permettent de mieux le stabiliser.

Une deuxième ligne du projet Rémoga [J14], présentée dans la Section 3.2, consistait à étudier une variante du problème (1.5) adaptée à la synthèse de textures par l'exemple : en prenant μ_u la distribution des patches de la synthèse u et μ_{u_0} la distribution des patches de la texture exemple u_0, on peut obtenir une synthèse en cherchant parmi les minima locaux de la fonction $u \mapsto W_\lambda(\mu_u, \mu_{u_0})$. En comparaison de l'approche Texto, on parvient à contraindre directement la distribution des patches en sortie à plusieurs résolutions. De plus, grâce aux calculs de gradients expliqués ci-dessus, l'algorithme de synthèse correspondant peut s'interpréter comme une généralisation de l'algorithme de projections NN par patches de [85], mais aboutissant à un meilleur contrôle statistique. On a aussi montré que l'on pouvait utiliser cette approche pour contraindre les distributions des réponses à des couches de réseaux de neurones pré-appris de façon à augmenter la qualité visuelle de la texture synthétisée (au prix d'une optimisation plus longue).

Enfin, un dernier axe du projet Rémoga [Js2] consistait à obtenir des garanties statistiques sur des estimateurs construits par minimisation d'un coût Wasserstein basé sur des versions empiriques $\hat{\mu}_\theta$ et $\hat{\nu}$ de μ_θ, ν. et de comprendre comment le paramètre de régularisation $\lambda > 0$ impactait la qualité de l'estimation. En collaboration avec Paul Freulon et Jérémie Bigot, nous avons pu prouver plusieurs bornes sur le risque empirique atteint par ces estimateurs, présentées dans la Section 3.3.

1.3 Modèles de Maximum d'Entropie

Le Chapitre 4 porte sur une formulation du problème de synthèse de textures avec des modèles de maximum d'entropie, pour lesquels on a donné des procédés d'échantillonnage et d'estimation avec garanties de convergence [J11].

Plutôt que de chercher à réimposer des contraintes géométriques sur un champ gaussien, une autre façon de synthétiser des textures structurées est d'échantillonner directement un modèle aléatoire de textures qui soit conçu de façon à contraindre (en moyenne) les réponses à un banc de filtres. Une façon de répondre à ce problème est de considérer une distribution de maximum d'entropie [167, 102] sous contrainte, c'est-à-dire une loi de probabilité π sur l'espace $\mathbb{R}^D$ des images solution de

$$\min_{\pi} \mathsf{KL}(\pi|\mu) \text{ sous la contrainte } \int_{\mathbb{R}^D} F(u)d\pi(u) = F(u_0), \tag{1.6}$$

où $F : \mathbb{R}^D \longrightarrow \mathbb{R}^p$ définit la collection de statistiques à préserver, et où $\mathsf{KL}(\cdot|\mu)$ est la divergence de Kullback-Leibler (i.e. l'opposé de l'entropie relative) par rapport à une loi de référence $\mu(du) = e^{-r(u)}du$ (en pratique $\mathcal{N}(0, \varepsilon^2 I)$). La contrainte assure que sous le modèle solution π_*, les statistiques calculées avec F prennent en

moyenne la valeur observée sur u_0. Sous certaines conditions, ce problème admet une solution unique sous la forme d'une distribution exponentielle

$$\pi_\theta(u) = \frac{1}{Z(\theta)} e^{-\theta\cdot(F(u)-F(u_0))}\mu(du), \tag{1.7}$$

où $\theta \in \mathbb{R}^p$ est un paramètre à estimer et $Z(\theta)$ une constante de normalisation.

Un exemple très simple de modèle de maximum d'entropie est donné par le modèle gaussien, qui respecte exactement les moments d'ordre 1 et 2 de la texture exemple. Au cours de ma thèse dirigée par Lionel Moisan et Bruno Galerne, nous avons exploité les avantages du modèle gaussien pour répondre à des problèmes de synthèse de microtextures en temps réel [C3, J3] ou d'inpainting de microtextures [J5]. Plus tard, dans [J6], en collaboration avec Agnès Desolneux, nous avons commencé à utiliser des modèles de maximum d'entropie plus généraux pour faire une reconstruction d' images à partir de descripteurs locaux.

Le Chapitre 4 est consacré à des travaux [J11] effectués dans le cadre de la thèse de Valentin de Bortoli, qui portent sur une étude plus générale des modèles de maximum d'entropie et de leurs procédés d'estimation et d'échantillonnage. D'abord, en tirant parti de la formulation duale de (1.6), qui s'écrit

$$\max_{\theta\in\Theta_F} L(\theta) := \log\left(\int_{\mathbb{R}^d} e^{-\theta\cdot(F(u)-F(u_0))} d\mu(u)\right), \tag{1.8}$$
$$\text{où } \Theta_F = \left\{\theta\in\mathbb{R}^p \mid \int e^{-\theta\cdot F} d\mu < \infty\right\},$$

nous avons pu expliciter des conditions assurant l'existence et l'unicité de la distribution de maximum d'entropie.

L'échantillonnage du modèle repose sur l'algorithme de Langevin [125], qui s'écrit ici

$$U_{n+1} = U_n - \gamma_n \nabla\chi(U_n,\theta) + \sqrt{2\gamma_n} Z_n \ , \ \text{où } \chi(u,\theta) = -\log\frac{d\pi_\theta}{du}, \tag{1.9}$$

et où $Z_n \sim \mathcal{N}(0,I)$ et $\gamma_n > 0$ est une suite décroissante. En utilisant des résultats de [43], on peut donner des bornes sur la proximité de la loi de U_n et de la loi cible π_θ. Dans le Chapitre 4, nous verrons que la solution $\pi_{\theta*}$ du problème de maximum d'entropie (1.6) peut être approchée avec un algorithme stochastique double appelé *Stochastic Optimization with Unadjusted Langevin* (SOUL), qui combine un schéma de gradient stochastique sur θ_n et des échantillons de Langevin U_n^k tirés à chaque itération n. On peut utiliser les échantillons U_n^k pour répondre à la synthèse de textures par maximum d'entropie. De plus, nous avons pu obtenir une garantie de convergence sur les valeurs de log-vraisemblance $L(\theta_k)$ atteintes par SOUL.

Ces modèles de maximum d'entropie généralisent les modèles gaussiens, en permettant de contraindre des statistiques plus complexes que les moments d'ordre 1 et 2. Pour la synthèse de textures, des méthodes récentes [51] invitent à utiliser des réponses à des réseaux de neurones pré-appris : F s'écrit alors

$$F_k(u) = \frac{1}{|\Omega|}\sum_{i\in\Omega} \mathcal{F}_k(u)(i) \quad \text{où} \quad \mathcal{F}_j(u) = (\varphi\circ A_j\circ\varphi\circ A_{j-1}\circ\ldots\circ\varphi\circ A_1)(u) \tag{1.10}$$

où les A_j sont des opérations affines, et où $\varphi : \mathbb{R} \longrightarrow \mathbb{R}$ est une fonction, dite d'activation, appliquée composante par composante, et où l'on moyenne sur les positions spatiales $i \in \Omega$. Les résultats théoriques que nous avons obtenus s'appliquent au cas de statistiques F extraites sur des couches de réseaux de neurones. Nous avons ainsi obtenu une formulation de la synthèse de textures comme l'échantillonnage d'un modèle aléatoire explicite π_θ, en conservant des garanties théoriques sur la précision de l'échantillonnage et l'estimation des paramètres.

1.4 Problèmes Inverses et Restauration d'Images Plug-and-Play

Le Chapitre 5 est centré sur la résolution de problèmes inverses pour la restauration d'images. En particulier, ce chapitre résume les travaux effectués pendant la thèse de Samuel Hurault sur l'étude des méthodes Plug-and-Play liées à des régularisations par réseaux de neurones profonds.

La restauration d'images est une problématique consistant à annuler l'effet de différentes dégradations pouvant être liées au bruit d'acquisition, au flou de bouger, au flou de défocalisation, à une occultation, etc. L'attache aux données d'un problème inverse linéaire avec bruit blanc gaussien $\mathcal{N}(0, \sigma^2)$ s'écrit

$$f(x) = \frac{1}{2\sigma^2}\|Ax - y\|^2, \tag{1.11}$$

où A est un opérateur linéaire sur un espace vectoriel $\mathcal{X}$. Lorsque A n'est pas injectif, l'inverse A^{-1} est mal défini ou instable, et l'on cherchera alors à rajouter un a priori sur la solution x dans la minimisation de f.

Modèles de faible dimension Une première façon de lever ce problème est de supposer que x appartient à un sous-ensemble $\Sigma \subset \mathcal{X}$, appelé "modèle de faible dimension", ce qui ramène au problème d'optimisation contraint

$$\operatorname*{Argmin}_{x \in \Sigma} \|Ax - y\|^2. \tag{1.12}$$

Sous certaines hypothèses sur A et Σ (dites d'isométrie restreinte), on peut montrer que ce problème a une solution unique x^*, qui permet d'approcher le signal original x_0 à une certaine précision. Cependant, l'ensemble Σ est en général non-convexe (par exemple, une union de sous-espaces vectoriels), ce qui rend difficile la résolution de (1.12). En pratique, on peut décider de paramétrer l'ensemble Σ par une fonction $\phi : \mathbb{R}^d \longrightarrow \mathcal{X}$ dont l'image contient Σ. En notant $\Theta = \phi^{-1}(\Sigma)$, on aboutit au problème de minimisation non-convexe

$$\theta^* \in \operatorname*{Argmin}_{\theta \in \Theta} \|A\phi(\theta) - y\|^2. \tag{1.13}$$

La Section 5.2 résume des travaux effectués en collaboration avec Y. Traonmilin et J.F. Aujol [J13], visant à fournir un cadre général pour examiner la convergence de l'algorithme de gradient à pas fixe sur la fonctionnelle (1.13), avec des

hypothèses minimales sur ϕ. En particulier, nous nous sommes attachés à donner explicitement (en fonction de constantes liées à A, ϕ, Σ) des bassins d'attraction de l'algorithme, c'est-à-dire des ensembles dans lesquels on peut initialiser l'algorithme de descente de gradient de façon à garantir la convergence des itérées vers x^*. Ces résultats génériques s'appliquent à plusieurs problèmes inverses (e.g. déconvolution sans grille, apprentissage compressé).

Restauration d'images Plug-and-Play (PnP) Une autre façon de rajouter de l'a priori sur la solution est d'intégrer un terme de régularisation $g(x)$ au problème, ce qui conduit à minimiser

$$F(x) = f(x) + \lambda g(x), \tag{1.14}$$

où $\lambda \geqslant 0$ est un paramètre. Selon la régularité des fonctions f, g on dispose de différents algorithmes, dits de *splitting*, pour minimiser F, par exemple la descente de gradient proximale (PGD) consistant à itérer la fonction

$$T_{\text{PGD}} = \text{Prox}_{\tau\lambda g} \circ (\text{Id} - \tau \nabla f), \tag{1.15}$$

basée sur l'opérateur proximal défini (sous certaines conditions) par $\text{Prox}_f(x) = \text{Argmin}_z \frac{1}{2}||x - z||^2 + f(z)$. Plusieurs algorithmes de restauration d'images exploitent des régularisations g convexes [147, 127, 38], qui permettent l'obtention de garanties de convergence [34, 33].

Par ailleurs, depuis l'essor de l'apprentissage profond [84, 138], les méthodes basées sur des réseaux de neurones profonds ont permis d'obtenir des progrès significatifs sur plusieurs problématiques de restauration d'images [133, 152, 163]. Les méthodes PnP permettent de tirer profit des performances atteintes par les réseaux profonds de débruitage, pour traiter génériquement d'autres problèmes inverses [156, 109, 128], avec un a priori implicitement encodé par le réseau débruiteur. Elles consistent à remplacer, dans un algorithme de *splitting*, l'étape de régularisation $\text{Prox}_{\tau\lambda g}$ par l'application d'un algorithme de débruitage pré-conçu.

Au cours de la thèse de Samuel Hurault, doctorant à l'IMB co-encadré par Nicolas Papadakis et moi-même, nous nous sommes attachés à fournir des garanties de convergence pour des algorithmes PnP inspirés par différentes méthodes de *splitting*. En se restreignant à une classe de débruiteurs particuliers, ces algorithmes peuvent de nouveau être liés à la minimisation d'une fonctionnelle explicite, ce qui permet de retrouver un contrôle numérique précis tout en bénéficiant des performances des réseaux de neurones profonds pour le débruitage.

La première méthode, appelée *Gradient-step PnP* (GS-PnP) [C11] et présentée dans la Section 5.3.3, consiste à utiliser un débruiteur sous la forme d'un pas de gradient $D_\sigma = \text{Id} - \nabla g_\sigma$ avec une fonction de régularisation s'écrivant

$$g_\sigma(x) = \frac{1}{2}||x - N_\sigma(x)||^2, \tag{1.16}$$

où $N_\sigma : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ est un réseau de neurones différentiable. On peut alors minimiser $F = f + \lambda g_\sigma$ via l'algorithme de descente de gradient proximale

$$x_{k+1} = T(x_k) \quad \text{avec} \quad T = \text{Prox}_{\tau f} \circ (\text{Id} - \tau\lambda \nabla g_\sigma), \tag{1.17}$$

dans lequel l'opérateur proximal porte maintenant sur l'attache aux données. En exploitant des résultats d'optimisation non-convexe, nous avons formulé une garantie de convergence pour cet algorithme, qui, contrairement à d'autres travaux concurrents, n'est pas soumise à des hypothèses contraignantes. Expérimentalement, nous avons pu vérifier qu'en paramétrant N_σ par l'architecture du réseau DRUNET [162], le débruiteur $D_\sigma = \mathrm{Id} - \nabla g_\sigma$ associé à (1.16) pouvait atteindre les mêmes performances que DRUNET. De plus, nous avons montré que l'algorithme GS-PnP atteignait des performances visuelles et quantitatives à l'état de l'art pour plusieurs problèmes inverses, tout en ayant une garantie théorique de convergence.

Nous avons ensuite proposé une autre méthode, dite Prox-PnP [C12] et présentée dans la Section 5.3.4, qui permet d'obtenir des résultats de convergence pour d'autres algorithmes de *splitting*, au prix d'une contrainte sur le débruiteur. En supposant ∇g_σ L-Lipschitz avec $L < 1$, nous avons montré que le débruiteur D_σ construit ci-dessus peut s'écrire comme l'opérateur proximal d'une certaine fonction non-convexe φ_σ. En utilisant cette fonction φ_σ comme régularisation, nous pouvons prouver la convergence de l'algorithme PnP associés à d'autres schémas de *splitting*. Sur le plan expérimental, la contrainte portant sur ∇g_σ impose une légère baisse de performance en débruitage, mais la qualité visuelle des restaurations obtenues avec ces nouveaux algorithmes PnP est similaire à celle des résultats obtenus avec GS-PnP.

Remarque sur les Notations

Dans les chapitres qui suivent, nous conserverons des notations proches de celles employées dans les articles publiés sur chacun des thèmes, avec néanmoins quelques adaptations favorisant la cohérence de l'ensemble du texte.

Dans les Chapitres 2 et 3, les lettres u, v sont utilisées pour désigner des images de textures, et les lettres majuscules U, V pour désigner les variables aléatoires correspondantes. Dans les paragraphes de ces chapitres traitant de synthèse de textures par l'exemple, la texture exemple est notée u_0.

Dans les Chapitres 4 et 5, les images et signaux sont désignés par les lettres x, y, z. Dans le Chapitre 4, la texture exemple est notée x_0.

Pour le transport semi-discret vers une distribution discrète ν portée par un ensemble fini $\mathcal{Y} = \{y_1, \dots, y_J\}$, la variable duale associée $\psi \in \mathbb{R}^{\mathcal{Y}}$ sera parfois écrite en notation indicielle $(\psi_y)_{y\in\mathcal{Y}}$ (Section 2.3), ou même $\psi_j = \psi(y_j)$ (Section 3.2) pour alléger l'écriture.

Publications

Articles de revue publiés

[J15] A. Houdard, A. Leclaire, N. Papadakis et J. Rabin. On the Gradient Formula for learning Generative Models with Regularized Optimal Transport Costs. *Transactions on Machine Learning Research*, 2023.
https://hal.archives-ouvertes.fr/hal-03740368

[J14] A. Houdard, A. Leclaire, N. Papadakis et J. Rabin. A Generative Model for Texture Synthesis based on Optimal Transport between Feature Distributions. *Journal of Mathematical Imaging and Vision*, vol. 65, pp. 4–28, 2023.
https://arxiv.org/abs/2007.03408

[J13] Y. Traonmilin, J.F. Aujol et A. Leclaire The basins of attraction of the global minimizers of non-convex inverse problems with low-dimensional models in infinite dimension. *Information and Inference : A Journal of the IMA*, 2022.
https://doi.org/10.1093/imaiai/iaac011

[J12] A. Leclaire et J. Rabin. A Stochastic Multi-layer Algorithm for Semi-Discrete Optimal Transport with Applications to Texture Synthesis and Style Transfer. *Journal of Mathematical Imaging and Vision*, vol. 63, no. 2, pp. 282–308, 2021.
https://doi.org/10.1007/s10851-020-00975-4

[J11] V. De Bortoli, A. Desolneux, A. Durmus, B. Galerne, et A. Leclaire Maximum entropy methods for texture synthesis : theory and practice. *SIAM Journal on Mathematics of Data Science*, vol. 3, no. 1, 2021.
https://doi.org/10.1137/19M1307731

[J10] Y. Traonmilin, J.F. Aujol et A. Leclaire Projected gradient descent for non-convex sparse spike estimation. *IEEE Signal Processing Letters*, vol. 27, pp. 1110–1114, 2020.
https://doi.org/10.1109/LSP.2020.3003241

[J9] V. De Bortoli, A. Desolneux, B. Galerne, et A. Leclaire Redundancy in Gaussian Random Fields. *ESAIM : Probability and Statistics*, vol. 24, pp. 627–660, 2020.
https://doi.org/10.1051/ps/2020010

[J8] V. De Bortoli, A. Desolneux, B. Galerne, et A. Leclaire Patch Redundancy in Images : a Statistical Testing Framework and some Applications. *SIAM Journal on Imaging Sciences*, vol. 12, no. 2, pp. 893–926, 2019.
https://doi.org/10.1137/18M1228219

[J7] B. Galerne, A. Leclaire et J. Rabin. A Texture Synthesis Model Based on Semi-Dicrete Optimal Transport in Patch Space. *SIAM Journal on Imaging*

Sciences, vol. 11, no. 4, pp. 2456–2493, 2018.
https://doi.org/10.1137/18M1175781

[J6] A. Desolneux et A. Leclaire. Stochastic Image Models from SIFT-like descriptors. *SIAM Journal on Imaging Sciences*, vol. 11, no. 4, pp. 2305–2338, 2018.
https://doi.org/10.1137/18M116592X

[J5] B. Galerne et A. Leclaire. Texture Inpainting using Efficient Gaussian Conditional Simulation.
SIAM Journal on Imaging Sciences, vol. 10, no. 3, pp. 1446–1474, 2017.
http://dx.doi.org/10.1137/16M1109047

[J4] B. Galerne et A. Leclaire. An Algorithm for Gaussian Texture Inpainting. *Image Processing On Line*, vol. 7, pp. 262–277, 2017.
https://doi.org/10.5201/ipol.2017.198

[J3] B. Galerne, A. Leclaire, et L. Moisan. Texton Noise. *Computer Graphics Forum*, vol. 36, no. 8, pp. 205–218, 2017. http://dx.doi.org/10.1111/cgf.13073

[J2] A. Leclaire et L. Moisan. No-reference image quality assessment and blind deblurring with sharpness metrics exploiting Fourier phase information. *Journal of Mathematical Imaging and Vision, Spec. issue on "Scale Space and Variational Methods in Computer Vision"*, vol. 52, no. 1, pp. 145–172, 2015.
https://doi.org/10.1007/s10851-015-0560-5

[J1] M. Lebrun et A. Leclaire. An Implementation and Detailed Analysis of the K-SVD Image Denoising Algorithm. *Image Processing On Line*, 2012.
https://doi.org/10.5201/ipol.2012.llm-ksvd

Articles de revue soumis

[Js2] J. Bigot, P. Freulon, B.P. Hejblum, A. Leclaire. On the potential benefits of entropic regularization for smoothing Wasserstein estimators. Soumis aux *Annals of Applied Statistics*.
https://arxiv.org/abs/2210.06934

[Js1] Y. Traonmilin, J.F. Aujol, A. Leclaire, P.J. Bénard. Upper and lower bounds on the size of strong basins of attractions for non-convex sparse spike estimation. Soumis.

Thèse

[Th] A. Leclaire. Random phase fields and Gaussian fields for image sharpness assessment and fast texture synthesis. Thèse soutenue le 26 juin 2015 à l'Université Paris Descartes. https://hal.science/tel-01196693/

Articles de conférence soumis

[Cs1] S. Hurault, U. Kamilov, A. Leclaire, et N. Papadakis. Convergent Bregman Plug-and-Play Image Restoration for Poisson Inverse Problems. soumis à *NEURIPS*, 2023.

Articles de conférence publiés

[C16] J. Delon, A. Desolneux, L. Facq, et A. Leclaire. Optimal Transport between GMM for Multiscale Texture Synthesis. *Proceedings of SSVM*, vol. 14009, pp. 379–392, 2023.
https://link.springer.com/chapter/10.1007/978-3-031-31975-4_48

[C15] S. Hurault, A. Chambolle, A. Leclaire, et N. Papadakis. A Relaxed Proximal Gradient Descent Algorithm for Convergent Plug-and-Play with Proximal Denoiser
Proceedings of SSVM, vol. 14009, pp. 379–392, 2023. https://link.springer.com/chapter/10.1007/978-3-031-31975-4_29

[C14] J. Delon, A. Desolneux et A. Leclaire Transport optimal entre GMM pour la synthèse de texture. *Actes du GRETSI*, 2022.

[C13] S. Hurault, A. Leclaire, et N. Papadakis. Débruiteur "descente de gradient" pour la convergence d'une méthode Plug-and-Play. *Actes du GRETSI*, 2022.

[C12] S. Hurault, A. Leclaire, et N. Papadakis. Proximal denoiser for convergent plug-and-play optimization with nonconvex regularization. *Proceedings of the Thirty-ninth International Conference on Machine Learning*, 2022.

[C11] S. Hurault, A. Leclaire, et N. Papadakis. Gradient Step Denoiser for convergent Plug-and-Play. *Proceedings of the Tenth International Conference on Learning Representations*, 2022.

[C10] A. Houdard, A. Leclaire, N. Papadakis et J. Rabin. Wasserstein Generative Models for Patch-Based Texture Synthesis. *Proceedings of Scale Space and Variational Methods in Computer Vision*, vol. 12679, pp. 269–280, 2021.

[C9] C. Launay et A. Leclaire. Determinantal Patch Processes for Texture Synthesis. *Actes du GRETSI*, 2019.

[C8] A. Leclaire et J. Rabin. A Fast Multi-Layer Approximation to Semi-Discrete Optimal Transport. *Proceedings of Scale Space and Variational Methods in Computer Vision*, vol. 11603, pp. 341–353, 2019.

[C7] V. De Bortoli, A. Desolneux, B. Galerne, et A. Leclaire Macrocanonical Models for Texture Synthesis. *Proceedings of Scale Space and Variational Methods in Computer Vision*, vol. 11603, pp. 13–24, 2019.

[C6] B. Galerne, A. Leclaire et J. Rabin. Semi-Discrete Optimal Transport in Patch Space for Enriching Gaussian Textures. *Geometric Science of Information*, 2017.

[C5] A. Desolneux et A. Leclaire. Stochastic Image Reconstruction from Local Histograms of Gradient Orientation. *Proceedings of Scale Space and Variational Methods in Computer Vision*, vol. 10302, pp. 133–145, 2017.

[C4] B. Galerne, A. Leclaire, et L. Moisan. Microtexture Inpainting Through Gaussian Conditional Simulation. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 2016.

[C3] B. Galerne, A. Leclaire, et L. Moisan. A Texton for Fast and Flexible Gaussian Texture Synthesis. *Proceedings of European Signal Processing Conference*, 2014.

[C2] A. Leclaire et L. Moisan. Blind Deblurring Using a Simplified Sharpness Index. *Proceedings of Scale Space and Variational Methods in Computer Vision*, vol. 7893, pp. 86–97, 2013.

[C1] A. Leclaire et L. Moisan. Une Variante non Périodique du Sharpness Index. *Actes du GRETSI*, 2013.

Chapitre 2

Synthèse de Textures par Transport Optimal de Patches

Ce chapitre récapitule mes travaux sur l'utilisation du transport optimal dans l'espace des patches pour la synthèse de textures par l'exemple. Ces travaux ont été poursuivis avec Bruno Galerne et Julien Rabin depuis la fin de ma thèse, et ont mené plus récemment à une nouvelle collaboration avec Agnès Desolneux et Julie Delon. Dans la Section 2.1, on décrit le contexte des méthodes à patches pour la synthèse de textures, et l'on justifie l'utilisation de statistiques par patches pour la modélisation de textures. Dans la Section 2.2, on décrit le modèle de textures Texto que l'on a défini et étudié dans [C6, J7] ; ce modèle vise à respecter les statistiques de patches d'une texture à plusieurs résolutions en utilisant des applications de transport optimal semi-discret. La Section 2.3 est consacrée à l'étude d'une approximation multi-couche des applications de transport optimal semi-discret [C8, J12] permettant d'accélérer les temps de calcul pour des distributions en grande dimension ; on montre ensuite que cette variante multi-couche permet d'enrichir le modèle Texto en travaillant avec des patches de plus grande taille. Enfin, dans la Section 2.4, on montre comment intégrer dans le modèle Texto une variante du transport optimal spécifique aux modèles de mélanges de gaussiennes, de façon à accélérer l'estimation du modèle [C16].

2.1 Synthèse de Textures par Patch

La synthèse de textures par l'exemple consiste à produire, à partir d'un échantillon donné d'une texture (appelée *exemple*), une nouvelle image ayant les mêmes caractéristiques perceptuelles que la texture exemple, tout en évitant la recopie de grands morceaux de celle-ci. Ceci est utile en infographie ou en édition de films, pour habiller des objets 3D de façon réaliste. Pour ces applications, on privilégie les algorithmes permettant une synthèse très rapide et à la demande (i.e. qui puissent synthétiser un pixel localement sans avoir à calculer la texture entière).

Du point de vue mathématique, ce problème de synthèse de textures se reformule par l'estimation et la simulation d'un modèle de champ aléatoire sur $\mathbb{Z}^2$. De plus, si l'on considère uniquement des textures homogènes, on peut se restreindre à l'étude de champs aléatoires stationnaires, c'est-à-dire dont la loi est invariante par translation. Nous allons donc chercher à proposer des champs aléatoires stationnaires qui puissent être simulés efficacement, et dont les paramètres puissent être ajustés de sorte que les échantillons ressemblent à la texture exemple. Pour quantifier cette ressemblance, on prendra en compte des indicateurs statistiques inspirés

par les mécanismes de perception humaine des textures [76].

Nous ne ferons pas ici une description détaillée de la découverte des statistiques régissant la perception visuelle humaine, et nous renvoyons à l'introduction de [J7] pour une discussion sur l'utilisation de ces statistiques pour la conception de modèles de textures. Mentionnons seulement que la perception des textures est liée à l'analyse des interactions locales entre pixels voisins. Cela a mené à la modélisation de textures par champs de Markov [35, 26, 25, 54] basés sur une forme paramétrique simple de la distribution conditionnelle d'un pixel sachant son voisinage. Cette modélisation par champs de Markov a été poursuivie plus tard par Zhu *et al.* [167] et Lu *et al.* [102] qui ont fait le lien avec les lois de maximum d'entropie que l'on discutera dans le Chapitre 4. Le principal inconvénient de ces modèles markoviens est que leur simulation repose sur des algorithmes coûteux (comme l'algorithme d'échantillonnage de Gibbs) qui ne répondent pas aux besoins pratiques en infographie.

Au lieu de la modélisation markovienne, la caractérisation statistique locale peut se faire plus simplement à l'aide de la distribution des patches. Considérer la distribution des patches $\mathsf{w} \times \mathsf{w}$ est en effet une façon non-paramétrique de modéliser l'aspect local d'une texture, puisqu'elle englobe toutes les statistiques de réponses de filtres de taille $\mathsf{w} \times \mathsf{w}$, par exemple la loi marginale des couleurs, les corrélations de pixels voisins, les moments des dérivées d'ordre faible (e.g. le laplacien), ou encore la distribution d'une réponse à un filtre non-linéaire de support $\mathsf{w} \times \mathsf{w}$ (i.e. la densité d'un *texton* au sens de Julesz [76]). Varma et Zisserman [154] ont montré que l'utilisation de la distribution de patches ou de la distribution conditionnelle locale permettaient d'obtenir des performances en classification de textures au moins aussi bonnes qu'avec des statistiques calculées sur des réponses de filtres.

Dès lors, pour la synthèse de textures, il est opportun de concevoir des modèles permettant de respecter la distribution des patches de la texture exemple. Les premières méthodes de synthèse par patches [48, 157] vont jusqu'à s'affranchir d'une modélisation statistique précise, en suggérant de synthétiser l'image par recopies locales. On parle alors de synthèse non-paramétrique, car au lieu de synthétiser un champ markovien respectant des statistiques paramétriques locales (estimées avec l'exemple), on remplace l'échantillonnage conditionnel local par un simple tirage aléatoire d'un pixel (dans la texture exemple) avec une contrainte de voisinage. Ce procédé, qui a plus tard bénéficié de plusieurs améliorations algorithmiques [96, 47, 86], est une technique de synthèse de textures remarquablement efficace. En dépit de sa consistance asymptotique prouvée par Levina et Bickel [94], ce procédé souffre en pratique d'un manque de contrôle statistique à horizon fini (produisant des images qualifiées de *garbage copy*).

Une façon d'éviter la dégénérescence du processus de synthèse est de l'exprimer au travers d'un problème d'optimisation. C'est le point de vue adopté par Kwatra *et al.* [85], dont l'algorithme de synthèse vise à minimiser une fonction de coût agrégeant les distances de chaque patch de la texture synthétisée à son plus proche voisin dans la texture exemple. De plus, cet algorithme, qui consiste à itérer des projections aux plus proches voisins (NN), peut être appliqué à plusieurs résolutions et sur plusieurs tailles de patches, en procédant à la synthèse depuis les échelles

grossières jusqu'aux échelles fines. La minimisation de cette fonction de coût multi-résolution permet qu'à chaque résolution et en toute position, la texture synthétisée ressemble localement à la texture originale. Même si ce procédé de synthèse est plus stable que la synthèse progressive de [48], il ne débouche pas non plus sur un contrôle statistique précis. En effet, le fait que tout patch synthétisé ressemble à un patch de la texture exemple ne signifie pas que sa fréquence d'apparition soit la même que celle observée dans la texture exemple.

Pour conserver un contrôle statistique, on peut modifier la fonctionnelle utilisée de façon à ce qu'elle contraigne la distance entre la distribution des patches de la synthèse et celle de l'exemple. Le transport optimal fournit une manière de quantifier la distance entre ces distributions de patches, alors même que ces distributions ne sont pas nécessairement continues ou discrètes et n'ont pas nécessairement les mêmes supports. Par exemple, Tartavel *et al.* [143] ont proposé de synthétiser des images en combinant des coûts de transport optimal discret calculés sur les distributions de réponses à un banc de filtres non-linéaires. Aussi, Gutierrez *et al.* [66] ont proposé un algorithme de synthèse de textures par l'exemple qui, à plusieurs résolutions, applique un plan de transport optimal discret.

Les travaux présentés dans ce chapitre visent à proposer des modèles de textures permettant de restituer fidèlement la distribution des patches de la texture exemple à plusieurs résolutions, tout en autorisant un procédé de simulation rapide. Le principe commun des modèles proposés est de partir d'une synthèse grossière de la texture, calculée avec un champ gaussien, et de l'enrichir en appliquant des transformations locales visant à réimposer la distribution des patches de la texture exemple. Les transformations locales utilisées sont calculées via le plan de transport optimal entre la distribution de patches de la synthèse courante et la distribution des patches de la texture exemple. Ces applications de transport optimal par patch permettent de réimposer des structures locales tout en garantissant une cohérence statistique globale (le champ gaussien utilisé pour l'échelle grossière servant à préserver les corrélations à grande échelle).

Il convient alors de choisir un cadre de transport optimal adapté à l'espace des patches, et qui fournit des applications de transport optimal pouvant être facilement évaluées, et estimées en temps raisonnable. Le premier modèle, appelé Texto et présenté dans la Section 2.2 se base sur des applications de transport optimal semi-discret. Ces applications sont des projections NN biaisées et peuvent donc être évaluées efficacement en parallèle, et être estimées via un algorithme stochastique de résolution du transport optimal semi-discret dû à Genevay *et al.* [57]. De façon à pouvoir considérer de plus grands patches, nous avons proposé d'approcher la solution du transport optimal semi-discret avec des applications multi-couche, ce qui a débouché sur le modèle TextoML (ML pour *multi-layer*) présenté dans la Section 2.3. Il permet de synthétiser efficacement des textures très structurées, tout en gardant un contrôle statistique global. Enfin, plus récemment, nous avons proposé une nouvelle variante de ce modèle, baptisée TextoGMM et présentée dans la Section 2.4, qui est basée sur une formulation du transport optimal spécifique aux modèles de mélanges de gaussiennes (GMM). Ce dernier modèle peut être estimé plus vite tout en offrant des performances visuelles similaires à TextoML.

2.2 Transport Optimal Semi-discret par Patch

Le modèle Texto [J7] présenté dans cette section est un champ aléatoire obtenu en appliquant une succession de transformées locales à un champ stationnaire gaussien. Ces transformations locales sont conçues de façon à résoudre un problème de transport optimal semi-discret dans l'espace des patches, à plusieurs résolutions. Ces transports par patch permettent d'enrichir le modèle gaussien en réimposant des structures géométriques saillantes, tout en se rapprochant globalement de la distribution des patches de la texture exemple. L'algorithme de synthèse correspondant est adapté à la synthèse à la demande en parallèle, ce qui permet de synthétiser de grandes textures semi-structurées avec un temps de calcul favorable.

2.2.1 Transport Optimal Semi-Discret

Avant de décrire le modèle Texto, on commence par rappeler ici le cadre de transport optimal semi-discret dont nous aurons besoin. Dans tout ce chapitre, l'espace $\mathbb{R}^D$ est muni du coût quadratique $c(x,y) = \|x-y\|_2^2$. Soient μ, ν deux mesures de probabilité sur $\mathbb{R}^D$. On se place dans le cas semi-discret : on suppose que μ est absolument continue par rapport à la mesure de Lebesgue sur $\mathbb{R}^D$, et que ν est supportée par un ensemble fini $\mathcal{Y}$ de cardinal J, $\nu = \sum_{y\in\mathcal{Y}} \nu(y)\delta_y$.

La formulation de Monge [130] du problème de transport optimal entre les mesures de probabilité μ (source) et ν (cible) s'écrit

$$\inf_T \int_{\mathbb{R}^D} \|x - T(x)\|^2 d\mu(x) \tag{OT-M}$$

où l'infimum est pris sur les applications mesurables $T : \mathbb{R}^D \longrightarrow \mathbb{R}^D$ telles que la mesure image $T\sharp\mu$ de μ par T coïncide avec ν. Dans le cas semi-discret considéré, on va voir que ce problème est équivalent à un problème d'optimisation en dimension finie.

Pour cela, définissons l'application $T_{\mathcal{Y},\psi}$ en posant pour presque tout $x \in \mathbb{R}^D$,

$$T_{\mathcal{Y},\psi}(x) = \operatorname*{Argmin}_{y\in\mathcal{Y}} \|x-y\|^2 - \psi(y), \tag{2.1}$$

où $\psi \in \mathbb{R}^{\mathcal{Y}}$ est un paramètre. Cette application réalise une projection NN biaisée. Les préimages $\mathtt{L}_\psi(y) = T^{-1}_{\mathcal{Y},\psi}(y),\ y \in \mathcal{Y}$ forment une partition de $\mathbb{R}^D$ (à un négligeable près) appelée diagramme de Laguerre (voir le diagramme de la Fig. 1.2 donnée dans l'introduction). Définissons aussi la "c-transform" de ψ par

$$\forall x \in \mathbb{R}^D, \quad \psi^c(x) = \min_{y\in\mathcal{Y}} \|x-y\|^2 - \psi(y). \tag{2.2}$$

Le théorème suivant fait le lien entre (OT-M) et un problème d'optimisation concave en dimension finie.

Théorème 2.2.1 ([4, 81])**.** *Le problème de transport optimal semi-discret* (OT-M) *admet une solution de la forme $T_{\mathcal{Y},\psi}$ où ψ est une solution du problème d'optimisation concave suivant :*

$$\operatorname*{Arg\,max}_{\psi\in\mathbb{R}^{\mathcal{Y}}} H(\psi) \quad \textit{avec} \quad H(\psi) = \int_{\mathbb{R}^D} \psi^c(x) d\mu(x) + \sum_{y\in\mathcal{Y}} \psi(y)\nu(y). \tag{2.3}$$

Input : mesures μ (à densité), et ν (discrète)
Initialisation : $\tilde{\psi}^1 = 0$
for $k \geqslant 2$ **do**
  $\tilde{\psi}^k = \tilde{\psi}^{k-1} + \frac{C}{\sqrt{k}} \nabla_\psi h(x^k, \tilde{\psi}^{k-1})$ où $x^k \sim \mu$
  $\psi^k = \frac{1}{k}(\tilde{\psi}^1 + \cdots + \tilde{\psi}^k)$.
end
Output: ψ^K

Algorithm 1: Algorithme ASGD pour le transport optimal semi-discret

De plus, H est de classe $\mathscr{C}^1$ sur $\mathbb{R}^{\mathcal{Y}}$ et son gradient est donné par

$$\forall y \in \mathcal{Y}, \quad \frac{\partial H}{\partial \psi(y)} = -\mu(\mathtt{L}_\psi(y)) + \nu(y). \tag{2.4}$$

Par conséquent, $\nabla H(\psi) = 0$ si et seulement si $(T_{\mathcal{Y},\psi})_\sharp \mu = \nu$.

Une solution de (2.3) peut être approchée via un algorithme de gradient stochastique proposé par Genevay *et al.* [57] et basé sur l'écriture de H sous la forme d'une espérance : si X est une variable aléatoire de loi μ, on a

$$H(\psi) = \mathbb{E}[h(X, \psi)] \quad \text{où} \quad h(x, \psi) = \psi^c(x) + \sum_{y \in \mathcal{Y}} \psi(y)\nu(y). \tag{2.5}$$

Cet algorithme, noté ASGD pour *averaged stochastic gradient descent*, est résumé dans l'Algorithme 1.

Il admet la garantie de convergence $\max(H) - \mathbb{E}[H(\psi_k)] = \mathcal{O}(\frac{\log k}{\sqrt{k}})$ et ne nécessite que le tirage d'échantillons suivant la loi μ, ce qui nous permettra de l'utiliser ci-dessous en grande dimension, dans l'espace des patches. À noter aussi que dans [J7], nous avons examiné en pratique la vitesse de convergence sur des cas synthétiques en petites dimensions.

2.2.2 Définition du Modèle Texto

Le modèle Texto repose sur une succession de transformations locales appliquées à plusieurs résolutions. Commençons par décrire le modèle monorésolution. Notons $u_0 : \Omega \longrightarrow \mathbb{R}^d$ la texture exemple, définie sur un rectangle $\Omega \subset \mathbb{Z}^2$. On notera $\omega = \{0, \dots, w-1\}^2$ le domaine des patches de taille $\mathsf{w} \times \mathsf{w}$ avec $\mathsf{w} \in \mathbb{N}^*$. L'espace des patches $\mathbb{R}^\omega$ s'identifie à $\mathbb{R}^D$ pour $D = d\mathsf{w}^2$. On notera $u_{|a+\omega} \in \mathbb{R}^D$ la restriction de u au patch $a + \omega$ en position $a \in \mathbb{Z}^2$.

D'abord, on définit le champ aléatoire gaussien U associé à u_0, défini par

$$\forall\, a \in \mathbb{Z}^2, \quad U(a) = \bar{u}_0 + \sum_{b \in \mathbb{Z}^2} t_{u_0}(b) W(a-b), \tag{2.6}$$

où $\bar{u}_0 = \frac{1}{|\Omega|} \sum_{a \in \Omega} u_0(a)$, $t_{u_0} = \frac{1}{\sqrt{|\Omega|}}(u_0 - \bar{u}_0)\mathbf{1}_\Omega$, et où W est un bruit blanc gaussien normalisé sur $\mathbb{Z}^2$, c'est-à-dire que les variables aléatoires $W(a), a \in \mathbb{Z}^2$ sont

i.i.d. $\mathcal{N}(0,1)$. Ce champ U est stationnaire de moyenne $\bar{u}_0$, et sa covariance est égale à la covariance empirique de u_0. Par conséquent, la synthèse gaussienne U a le même contenu fréquentiel que u_0, mais n'a pas de détail géométrique saillant.

Pour raffiner localement U, on extrait tous ses patches $U_{|a+\omega}$ et on leur applique une même transformation $T : \mathbb{R}^D \longrightarrow \mathbb{R}^D$ réalisant un transport optimal par patch. Puis on les agrège avec une simple moyenne pour obtenir le champ transformé V défini par

$$\forall a \in \mathbb{Z}^2, \quad V(a) = \frac{1}{|\omega|} \sum_{b \in \omega} T(U_{|a-b+\omega})(b). \tag{2.7}$$

Le champ V est encore stationnaire et possède une propriété d'indépendance à longue portée (c'est-à-dire que deux restrictions V_A, V_B sont indépendantes dès que A et B sont deux parties finies de $\mathbb{Z}^2$ suffisamment espacées). Cette transformation locale est illustrée dans la Fig. 2.1.

De façon à réimposer des structures de façon statistiquement cohérente, nous utilisons l'application T qui résout le problème de transport optimal semi-discret entre la distribution μ d'un patch $U_{|\omega}$ de U (qui est une loi gaussienne explicite) vers la distribution empirique des patches de u_0. En pratique, nous utilisons l'application $T_{\mathcal{Y},\psi}$ où ψ est une solution approchée de (2.3) calculée par l'Algorithme 1. La texture exemple pouvant être très grande, on sous-échantillonne la distribution empirique cible, en définissant $\nu = \frac{1}{J}\sum_{j=1}^{J} \delta_{p_j}$, où $p_1, \ldots, p_J$ sont J patches tirés aléatoirement dans u. Dans [J7], nous avons constaté que pour des patches 3×3, il suffisait d'utiliser $J = 1000$ patches pour approcher la distribution empirique cible. Ce choix permet à la fois de s'assurer d'une bonne convergence pratique de l'ASGD, et d'avoir des applications $T_{\mathcal{Y},\psi}$ facilement évaluables. On verra dans la Section 2.3 comment modifier les transformations locales utilisées de façon à traiter de plus grands patches. Sur la Fig. 2.1, on peut constater que la transformation locale utilisée sur des patches 3×3 permet déjà de faire réapparaître des structures de la texture exemple.

Pour synthétiser des textures plus structurées, on définit une extension multirésolution de ce modèle. On va en effet travailler avec S versions sous-échantillonnées u_s, $(0 \leqslant s \leqslant S-1)$ de la texture exemple à plusieurs échelles. L'image u_s est définie sur une sous-grille $\Omega_s \subset 2^s\mathbb{Z}^2$ de pas 2^s. On notera ν_s la distribution discrète formée par J patches $\mathsf{w} \times \mathsf{w}$ pris aléatoirement dans u_s.

Le modèle multirésolution est défini par des transports successifs, et résumé dans l'Algorithme 2. L'estimation du modèle nécessite une passe de synthèse (pouvant être faite *offline*). À la résolution grossière $s = S-1$, on part du champ gaussien U_{S-1} associé à u_{S-1}. Ensuite, pour chaque échelle $s = S-1, \ldots, 0$, une transformation T_s est appliquée à tous les patches de la synthèse. L'application T_s est calculée dans la phase d'estimation du modèle, avec deux étapes :

- On estime un modèle GMM μ_s approchant la distribution des patches de U_s en utilisant l'algorithme *Expectation Maximization* (EM) [108].
- On calcule l'application T_s de transport optimal entre μ_s et ν_s.

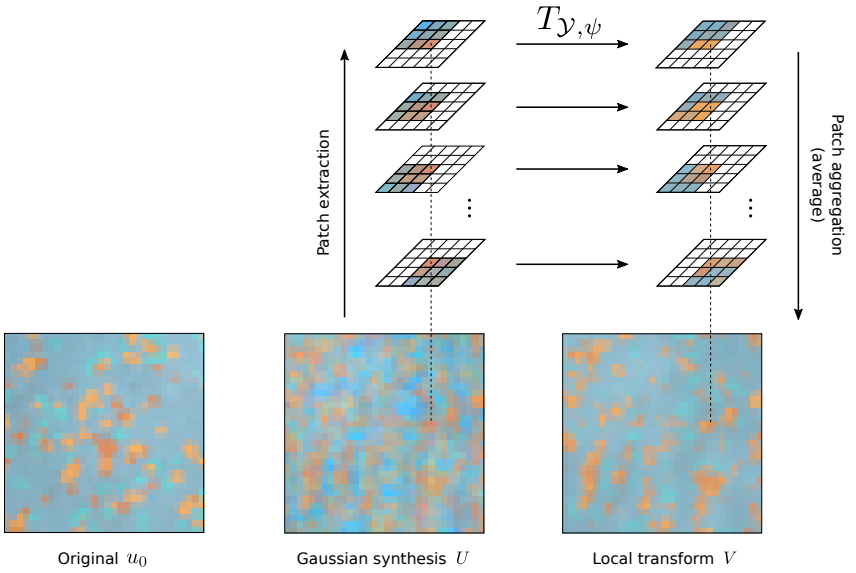


FIGURE 2.1 – **Transformée locale d'un champ gaussien.** On montre ici une texture originale u_0 (à gauche), le champ gaussien U (au milieu) et le champ V obtenu après une transformée locale $T_{\mathcal{Y},\psi}$ appliquée aux patches 3×3.

Une fois T_s estimée, les patches sont transformés et moyennés pour obtenir

$$V_s(a) = \sum_{b \in 2^s\omega} T_s(U_{s|a-b+2^s\omega})(b),\ a \in 2^s\mathbb{Z}^2. \tag{2.8}$$

Puisque T_s est un assignement NN biaisé, il existe une carte de coordonnées $C_s : 2^s\mathbb{Z}^2 \longrightarrow \Omega_s$ permettant d'écrire V_s comme

$$V_s(a) = \sum_{b \in 2^s\omega} u_s\big(C_s(a-b)+b\big),\ a \in 2^s\mathbb{Z}^2. \tag{2.9}$$

Pour $s \geqslant 1$, on peut alors initialiser la synthèse à la résolution $s-1$ avec une étape de sur-échantillonnage basée exemple : on prend les patches aux mêmes positions, mais deux fois plus grands, ce qui revient à définir

$$\forall a \in 2^s\mathbb{Z}^2,\ \forall t \in \{0, 2^{s-1}\}^2, \quad U_{s-1}(a+t) = \sum_{b \in 2^s\omega} u_{s-1}\big(C_s(a-b)+b+t\big). \tag{2.10}$$

Ce procédé *coarse-to-fine* est résumé dans l'Algorithme 2 et illustré sur la Fig. 2.2.

L'étape la plus coûteuse de la phase d'estimation de l'Algorithme 2 est l'estimation des applications de transport semi-discret T_s. Mais insistons sur le fait que ces applications T_s (paramétrées par les patches $p_{1,s}, \ldots, p_{J,s}$ extraits à l'échelle s et par les poids $\psi_s \in \mathbb{R}^J$) sont calculées une fois pour toutes au moment de l'estimation du modèle. Une fois le modèle appris, pour la phase de synthèse, il suffit de générer le champ gaussien à l'échelle grossière (ce qui se fait efficacement par transformée de Fourier discrète) et d'appliquer à la volée les transports par patches pré-calculés.

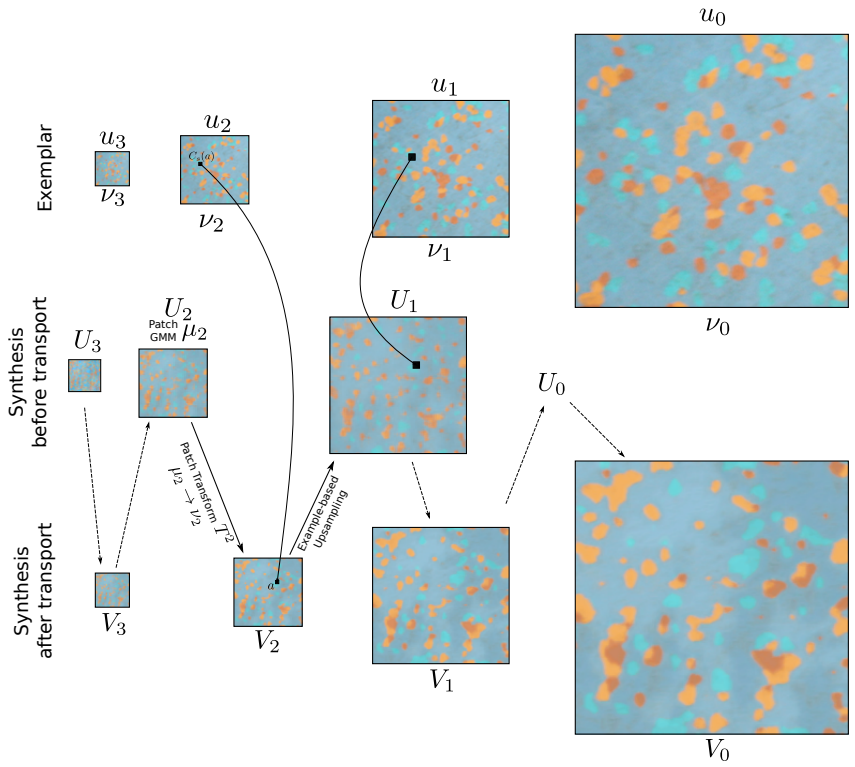


FIGURE 2.2 – **Synthèse de textures avec le modèle Texto.** La texture est synthétisée en partant du champ gaussien à la résolution grossière et en raffinant la texture en appliquant un transport optimal par patch à chaque résolution suivi d'une étape de sur-échantillonnage basée exemple. On peut voir, à chaque échelle, la texture originale (u^s, en haut), la texture synthétisée (V^s, en bas), et la texture obtenue avant le transport optimal par patch de l'échelle s (U^s, au milieu).

Param.: S : nombre d'échelles
n_{GMM} : nombre de composantes gaussiennes dans les GMM μ_s
J : nombre de patches dans les distributions ν_s
Input : texture exemple u_0, et versions réduites u_s définies sur $2^s\mathbb{Z}^2$
Initialisation : Échantillonner le champ gaussien U_{S-1} (2.6).
for $s = S-1, \dots, 0$ **do**
  if *Modèle à estimer* **then**
    · Estimer un GMM μ_s sur les patches de U_s (Algorithme EM)
    · Extraire J patches de u_s pour former ν_s
    · Estimer le transport optimal T_s de μ_s à ν_s (Algorithme ASGD)
  end
  Appliquer T_s aux patches de U_s
  Agréger les patches pour obtenir V_s (2.8)
  If $s > 1$, sur-échantillonner V_s pour obtenir U_{s-1} (2.10)
end
Output: V_0

Algorithm 2: Estimation et Synthèse avec le modèle Texto

2.2.3 Synthèses avec le Modèle Texto

Commentons quelques résultats de synthèse obtenus avec le modèle Texto, en reportant la comparaison avec les méthodes concurrentes à la section suivante.

La Fig. 2.3 contient des exemples de grandes textures synthétisées avec Texto. On voit que l'algorithme produit une synthèse satisfaisante lorsque l'image d'entrée présente des motifs suffisamment répétés dont la taille est comparable à la taille des patches multi-échelle utilisés. La définition même du modèle, avec une recomposition des patches par moyenne locale, fait que les motifs synthétisés ne se retrouvent que très rarement dans la texture exemple. Ces images montrent aussi que le tirage aléatoire de $J = 1000$ patches dans la texture exemple suffit à reproduire les motifs observés dans les patches 3×3. Notons aussi que l'initialisation avec le champ gaussien à l'échelle grossière permet de bien reproduire le contenu à basses fréquences de l'image.

Les limitations visuelles du modèle sont illustrées ici avec six cas d'échec (trois dernières lignes de la Fig 2.3). Certains des cas d'échec peuvent néanmoins être considérés comme succès si l'on se limite à un système de perception pré-attentif. En général, le modèle échoue pour des textures présentant un agencement de motifs avec des contraintes géométriques fortes. Une façon de mieux reproduire ces structures complexes est d'adapter le modèle à des espaces de patches plus grands, ce que nous ferons dans la section suivante. Par ailleurs, sur les textures exemples présentant des motifs périodiques, la synthèse gaussienne échoue à la résolution grossière, ce qui n'est pas corrigé par les transformations locales par patch.

Ces observations permettent de formuler deux conditions nécessaires de succès du modèle Texto : la texture exemple sous-échantillonnée à l'échelle S doit être une microtexture, et les motifs de la texture exemple doivent pouvoir être légèrement déformés sans perturber la perception de la texture.

Terminons ce paragraphe en comparant avec une autre méthode de synthèse due à Kwatra *et al.* [85] qui consiste à itérer des projections NN par patches à plusieurs résolutions. Ici, on simplifie cette méthode en ne considérant que des patches 3×3 et en n'utilisant que la distance ℓ^2 (au lieu de distances ℓ^2 repondérées), ce qui revient à minimiser la fonction de coût

$$E_{\mathrm{NN}} = \sum_{s=0}^{S-1} \sum_{x} \left\| V^s_{|x+2^s\omega} - \mathrm{NN}_{u^s}(V^s_{|x+2^s\omega}) \right\|^2 \tag{2.11}$$

où $\mathrm{NN}_{u^s}(p)$ désigne le patch de u^s qui est le plus proche de p au sens ℓ^2. Sur la Fig. 2.4, on voit que ces itérations NN permettent de mieux minimiser cette fonction de coût, en comparaison de la méthode Texto (qui n'est pas conçue pour minimiser E_{NN}). Et pourtant, on constate que les synthèses obtenues par ces projections NN itérées sont de qualité visuelle moindre que celles de Texto, ce qui justifie la pertinence du contrôle statistique réalisé par les applications de transport semi-discret (alors que les projections NN itérées n'offrent aucun contrôle statistique et pourraient dégénérer sur des patches aberrants). Dans la Section 3.2, nous discuterons la possibilité d'itérer le transport optimal à chaque résolution, et nous ferons le lien avec la minimisation d'une autre fonction de coût.

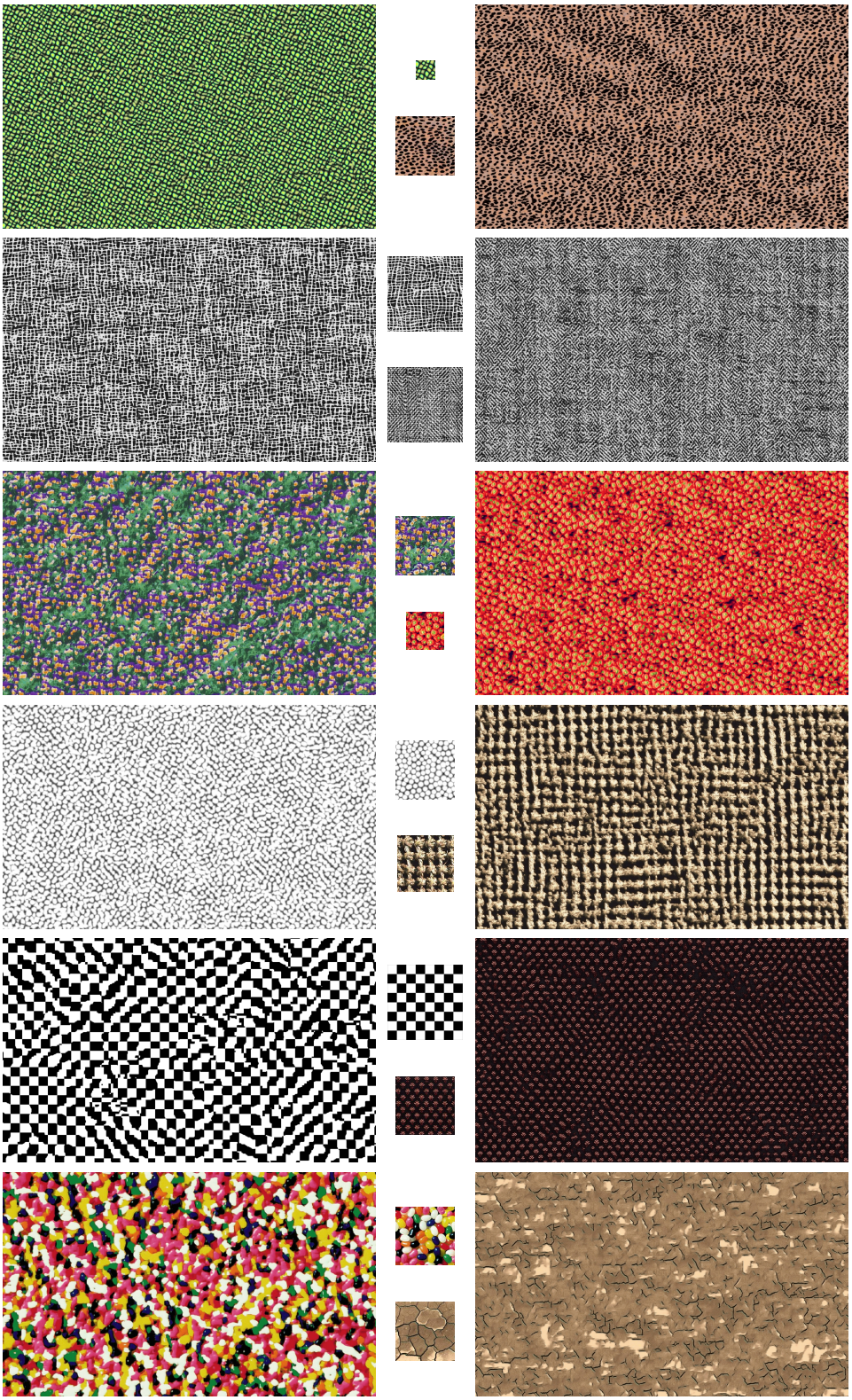


FIGURE 2.3 – **Synthèse avec le modèle Texto.** Pour chaque texture exemple montrée au milieu, on affiche la texture de taille 1280×768 synthétisée avec le modèle Texto où l'on a utilisé $S = 4$ échelles, et $n_{\mathrm{GMM}} = 4$.

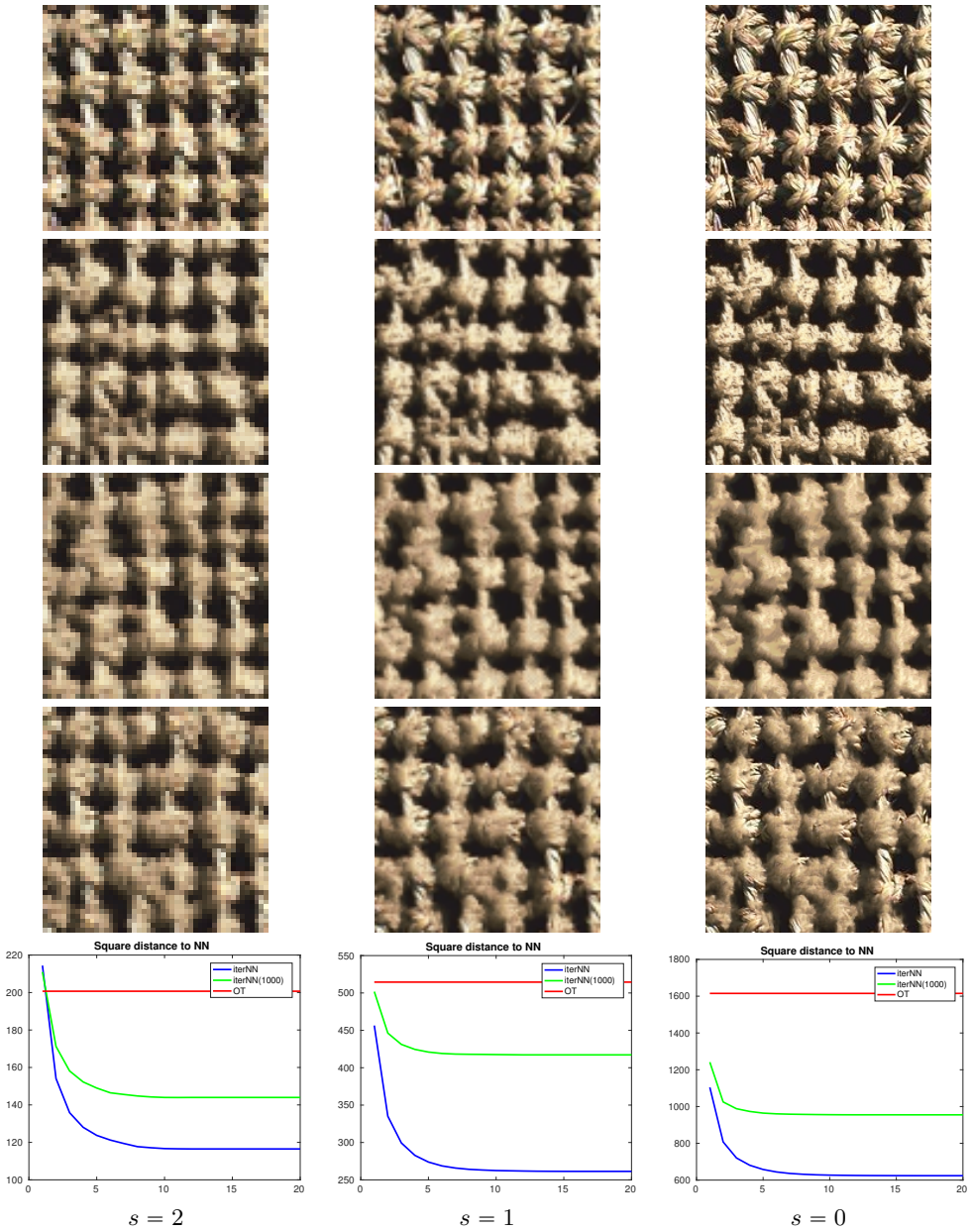


FIGURE 2.4 – **Comparaison avec les projections NN itérées.** On compare ici les synthèses obtenues avec le modèle Texto (2ème ligne) i.e. un transport optimal semi-discret à chaque échelle, ou bien, en appliquant des projections NN (itérées 20 fois) à chaque échelle (3ème et 4ème ligne). À chaque échelle, on projette sur les mêmes 1000 patches, sauf pour la 4ème ligne où l'on projette sur tous les patches de la texture exemple. Sur la dernière ligne, on affiche la fonction de coût intégrant la somme des distances au carré entre les patches de la synthèse et leur plus proche voisin dans la texture exemple pour l'échelle s. On constate ici que l'itération des projections NN est une bonne manière de minimiser cette fonction de coût. Mais la synthèse est visuellement plus satisfaisante avec le modèle Texto, qui n'est pas conçu pour minimiser cette fonction de coût.

2.3 Transport Optimal Semi-Discret Multi-Couche

La limitation principale du modèle Texto est la faible vitesse de convergence de l'algorithme stochastique utilisée pour estimer le transport optimal semi-discret, en particulier lorsque le nombre de points J dans la distribution cible augmente. Or, pour adapter le modèle Texto à de plus grands patches, il est nécessaire de prendre plus de patches dans les distributions cibles. Dans cette section, nous allons décrire une approximation multi-couche des applications de transport optimal semi-discret. Ces applications multi-couche peuvent être optimisées de façon plus rapide, et intégrées dans le modèle de synthèse de textures défini ci-dessus, aboutissant au modèle TextoML [J12] (pour *Texto MultiLayer*).

2.3.1 Transport Multi-couche

L'idée du transport multi-couche est de composer des applications de transport semi-discret ciblant des versions simplifiées, à plusieurs échelles [1], de la distribution ν. On va donc travailler sur une décomposition multi-échelle $\nu = \nu^0, \dots, \nu^{L-1}, \nu^L$ de la mesure cible ν, où pour chaque $\ell \in \{0, \dots, L\}$, la distribution ν^ℓ est supportée sur un ensemble $\mathcal{Y}^\ell$ de cardinal prescrit J^ℓ, avec $J = J^0 > \ldots > J^{L-1} > J^L = 1$. Comme dans [112], on construit cette décomposition récursivement : partant de ν^ℓ supportée par $\mathcal{Y}^\ell$, l'algorithme K-means pondéré permet d'obtenir un ensemble de $J^{\ell+1}$ centroïdes $\mathcal{Y}^{\ell+1} \subset \mathbb{R}^D$ correspondant à une partition

$$\mathcal{Y}^\ell = \bigsqcup_{y \in \mathcal{Y}^{\ell+1}} C_y^\ell. \tag{2.12}$$

Les centroïdes héritent alors des masses des cellules associées :

$$\forall\, y \in \mathcal{Y}^{\ell+1}, \quad \nu^{\ell+1}(y) = \nu^\ell(C_y^\ell). \tag{2.13}$$

La famille d'ensembles $(\mathcal{Y}^0, \mathcal{Y}^1, \dots, \mathcal{Y}^L)$ et de cellules $(C_y^\ell)_{0 \leqslant \ell < L, y \in \mathcal{Y}^{\ell+1}}$ sera appelée *clustering hiérarchique* de $\mathcal{Y}$ et notée $\boldsymbol{\mathcal{Y}}$.

Définition 2.3.1 (Voir Fig. 2.5)**.** Soit $\boldsymbol{\mathcal{Y}}$ le clustering hiérarchique introduit ci-dessus. Pour une collection de paramètres

$$\boldsymbol{\psi} = \left(\psi_y^\ell\right)_{0 \leqslant \ell < L, y \in \mathcal{Y}^\ell} \in \prod_{\ell=0}^{L-1} \mathbb{R}^{\mathcal{Y}^\ell}, \tag{2.14}$$

on définit une application multi-couche $T_{\boldsymbol{\mathcal{Y}},\boldsymbol{\psi}}$ comme suit : pour $x \in \mathbb{R}^d$ fixé, on pose $T^L(x) = y^L$, et pour $\ell = L-1, \dots, 0$, en notant $y = T^{\ell+1}(x)$, on pose

$$T^\ell(x) = T_{C_y^\ell, \psi_y^\ell}(x) = \operatorname*{Argmin}_{z \in C_y^\ell} \|x - z\|^2 - \psi_y^\ell(z). \tag{2.15}$$

L'application finale s'écrit $T_{\boldsymbol{\mathcal{Y}},\boldsymbol{\psi}}(x) = T^0(x)$.

1. On utilisera aussi le mot "échelle" pour désigner les différents niveaux de simplification de la distribution ν, qui permettront de générer les différentes couches de l'application de transport composée. Cette notion d'échelle liée à l'application de transport optimal ne doit pas être confondue avec les différentes résolutions d'images qui servent à estimer le modèle Texto.

En d'autres termes, chaque application $T_{C_y^\ell,\psi_y^\ell}$ définit un diagramme de Laguerre, dont les cellules sont re-divisées à l'échelle suivante en un diagramme de Laguerre plus fin. Calculer la valeur des applications intermédiaires $T^\ell(x)$ revient à remonter, à partir de x, au sein d'une hiérarchie de cellules de Laguerre. En reprenant la suite d'application T^ℓ de la définition, les pré-images définissent donc une collection de cellules emboîtées :

$$\mathcal{L}_y^\ell = (T^\ell)^{-1}(\{y\}), \quad (y \in \mathcal{Y}). \tag{2.16}$$

Définition 2.3.2. L'application multi-couche $T_{\boldsymbol{\mathcal{Y}},\boldsymbol{\psi}}$ est dite optimale si pour tout $0 \leqslant \ell < L$ et tout $y \in \mathcal{Y}^{\ell+1}$, $T_{C_y^\ell,\psi_y^\ell}$ est une solution du transport optimal semi-discret entre $\mu_{|\mathcal{L}_y^{\ell+1}}$ et $\nu^\ell_{|C_y^\ell}$.

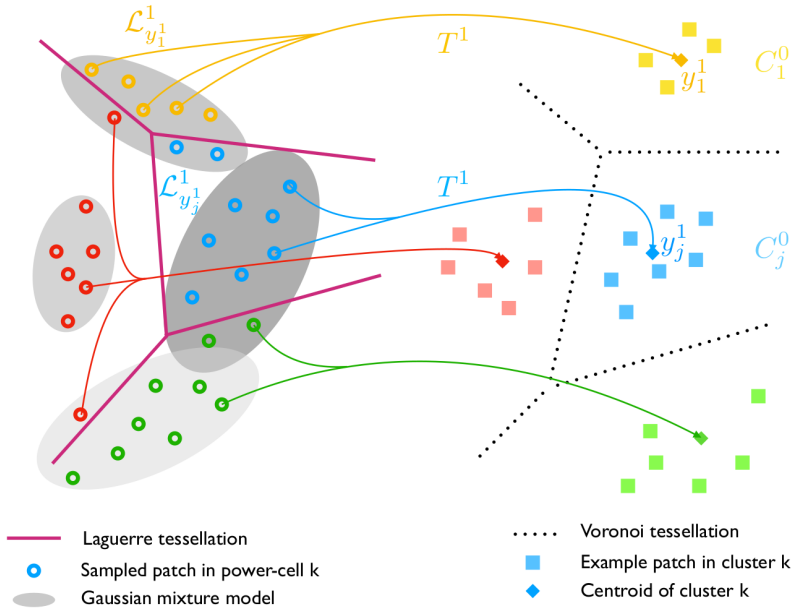
Dans [J12], nous avons étudié le lien entre l'optimalité d'une application multi-couche et le problème de transport optimal semi-discret d'origine. Comme les applications multi-couche ont une forme contrainte par le clustering hiérarchique $\boldsymbol{\mathcal{Y}}$, en dehors de configurations très particulières (par exemple en dimension 1 sous une hypothèse de croissance du clustering $\boldsymbol{\mathcal{Y}}$), la solution du transport optimal semi-discret entre μ et ν ne s'écrit pas comme une application multi-couche. On introduit donc un biais dans le problème, car le clustering $\boldsymbol{\mathcal{Y}}$ contraint la géométrie des cellules emboîtées.

Cependant, on peut encore décrire l'optimalité de l'application multi-couche en disant que les paramètres $\boldsymbol{\psi}$ maximisent une collection de fonctionnelles

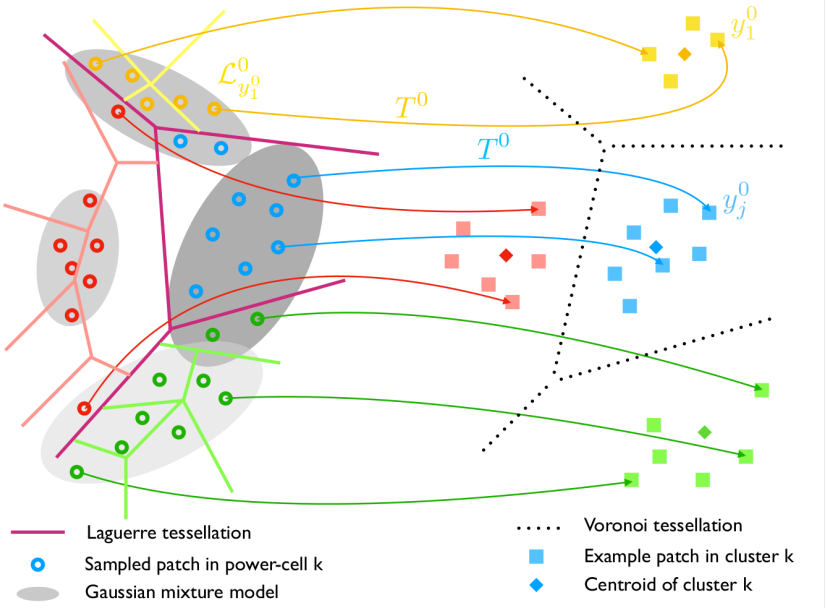
$$\tilde{H}_y^\ell(\psi_y^\ell) = \mathbb{E}_{X\sim\tilde{\mu}_y^{\ell+1}}\left[\tilde{h}_y^\ell(X,\psi_y^\ell)\right] \tag{2.17}$$

$$\text{où} \quad \tilde{h}_y^\ell(x,\psi_y^\ell) = \left(\min_{z\in C_y^\ell} \|x-z\|^2 - \psi_y^\ell(z)\right) + \sum_{z\in C_y^\ell} \psi_y^\ell(z)\tilde{\nu}_y^\ell(z). \tag{2.18}$$

et où l'on a défini les distributions normalisées $\tilde{\mu}_y^{\ell+1} = \frac{\mu_{|\mathcal{L}_y^{\ell+1}}}{\mu(\mathcal{L}_y^{\ell+1})}, \tilde{\nu}_y^\ell(z) = \frac{\nu_y^\ell(z)}{\nu_y^\ell(C_y^\ell)}$. Là encore, on peut proposer un algorithme d'optimisation stochastique permettant d'approcher une solution de ce problème de transport multi-couche. Chaque itération de cet algorithme, partant d'un échantillon $x \sim \mu$, remonte la hiérarchie multi-échelle et réajuste les coefficients ψ_y^ℓ par pas de gradient sur h_y^ℓ. L'intérêt de cette méthode, résumée dans l'Algorithme 3, est qu'elle permet de parvenir plus rapidement à des applications réalisant un transport mieux équilibré à grande échelle, même si, à convergence, l'erreur induite par le clustering hiérarchique $\boldsymbol{\mathcal{Y}}$ ne pourra pas être évitée. Ceci est confirmé par la simulation montrée dans la Fig. 2.6 portant sur un problème de transport optimal semi-discret en 1D, et par d'autres expériences en dimension supérieure données dans l'article [J12]. On trouvera aussi dans cet article quelques commentaires sur les paramètres : nous recommandons de former $J^{\frac{1}{L}}$ clusters par échelle pour avoir une optimisation plus équilibrée et efficace ; et nous recommandons aussi de ne pas augmenter le nombre de couches L au-delà de 2 ou 3 de façon à ne pas faire augmenter l'erreur due au clustering hiérarchique.



Première couche $\ell = 1$:
Le transport grossier T^1 envoie la cellule j vers le centroïde du cluster C^0_j de ν.



Deuxième couche $\ell = 0$:
Le transport fin $T = T^0$ est défini avec les applications $T_{C^0_j, \psi^\ell_j}$.

FIGURE 2.5 – **Illustration d'une application de transport à 2 couches.** On illustre ici avec une distribution μ qui est un mélange de gaussiennes à quatre composantes (en gris). Pour chaque couche ℓ, les flèches illustrent $T^\ell(x)$ qui envoie des points $x \sim \mu$ (points ronds à gauche) vers des points de ν^ℓ (losanges pour $\ell = 1$ et carrés pour $\ell = 0$).

Input : Mesure source μ, mesure cible ν, pas de gradient $C > 0$, nombre de couches L, nombre d'itérations T

Calculer un clustering hiérarchique $\{\nu^0, \dots, \nu^{L-1}\}$ of ν

Définir les mesures normalisées $\tilde{\nu}_y^\ell \; \forall\, \ell, y$

$\tilde{\psi}_y^\ell \leftarrow 0$, $\forall\, \ell, y$ (initialization)

$n_y^\ell \leftarrow 0$, $\forall\, \ell, y$ (nombre de visites du cluster C_y^ℓ)

for $t = 1, \dots, T$ **do**

  Tirer $x \sim \mu$

  for $\ell = L-1, \dots, 0$ **do**

    En utilisant $T^{\ell+1}(x)$, calculer l'indice $y = T^\ell(x)$ (avec (2.15))

    $n_y^\ell \leftarrow n_y^\ell + 1$

    $g \leftarrow \nabla_{\tilde{\psi}_y^\ell} \tilde{h}_y^\ell(x, \tilde{\psi}_y^\ell)$ (avec (2.18))

    $\tilde{\psi}_y^\ell \leftarrow \tilde{\psi}_y^\ell + \frac{C}{\sqrt{n_y^\ell}} g$

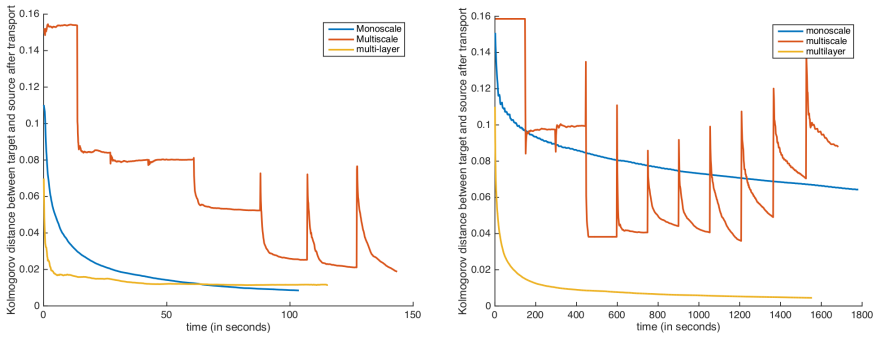
    $\psi_y^\ell \leftarrow \psi_y^\ell + \frac{1}{n_y^\ell}\left(\tilde{\psi}_y^\ell - \psi_y^\ell\right)$

  end

end

Output: $\{\nu^\ell\}_{0 \leqslant \ell < L}$ et $\boldsymbol{\psi} = \left(\psi_y^\ell\right)_{0 \leqslant \ell < L, y \in \mathcal{Y}^\ell}$

Algorithm 3: Algorithme stochastique pour le transport multi-couche



$J = 1000$ points $\qquad$ $J = 10\,000$ points

FIGURE 2.6 – **Transport 2-couches v.s. 1-couche.** Dans cette figure, on compare trois algorithmes pour le transport optimal semi-discret : l'algorithme mono-couche (Algorithme 1), l'algorithme à deux couches (Algorithme 3 pour $L = 2$), et l'algorithme multi-échelle de [112]. On compare les applications de transport obtenues en mesurant la distance de Kolmogorov entre $T_\sharp \mu$ et ν (en ordonnée), le long du temps de calcul (en abscisse). On constate que pour J très grand, l'approche multi-couche permet d'atteindre une bonne approximation en temps court, même si elle ne converge pas vers la vraie solution du transport semi-discret.

2.3.2 Application à la Synthèse de Textures

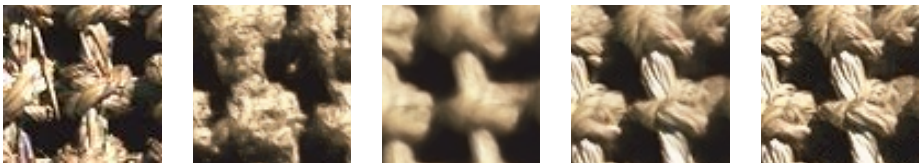
Dans [J12], nous avons utilisé ces applications de transport multi-couche pour améliorer le modèle Texto. Ce nouveau modèle, appelé TextoML, permet de considérer des distributions cibles avec un plus grand nombre de points. Ceci ouvre la possibilité de travailler avec des distributions plus riches, notamment lorsque l'on considère des distributions de patches de plus grande dimension w.

Le modèle TextoML est construit de la même manière que le modèle Texto, en remplaçant les applications de transport utilisées à chaque résolution par des applications multi-couche, dont l'estimation repose sur un algorithme stochastique. Sur la Fig. 2.7, on peut comparer des résultats de synthèse obtenus avec Texto et TextoML. On peut voir que l'utilisation directe du modèle Texto avec de grands patches ($\mathsf{w} = 7$) donne des résultats de synthèse aberrants, puisque les applications de transport optimal ne sont pas correctement estimées à chaque échelle. À l'inverse, le transport multi-couche donne des applications de transport cohérentes au bout de 10^6 itérations de l'Algorithme 3, même en grande dimension (ici $D = 147$ pour des patches 7×7 en couleur). Le modèle TextoML permet ainsi de synthétiser efficacement des textures structurées. Son défaut principal est que les synthèses TextoML sont affectées par une légère quantité de flou due à l'agrégation des patches. Pour pallier partiellement ce défaut, nous avons proposé de rajouter une dernière étape (PP pour *post-processing*) de transport multi-couche sur les patches 3×3, qui permet mieux réimposer les statistiques aux échelles fines, sans pour autant détruire les structures plus larges déjà mises en place par les transports précédents.

On trouvera d'autres exemples de synthèse dans la Fig. 2.10 du paragraphe suivant et dans [J12]. Sur la Fig. 2.8 on peut comparer TextoML avec d'autres méthodes neuronales récentes [51, 150], sur deux grandes textures comportant des structures régulières à grande échelle et des détails saillants à échelle fine. On observe que TextoML parvient à reproduire des corrélations à longue portée grâce à l'initialisation avec un champ gaussien à basse résolution, et de façon parfois plus convaincante que les méthodes neuronales. TextoML parvient également à produire de nouveaux motifs locaux, ressemblant à ceux de la texture exemple, mais de façon plus floue que [51]. Un avantage de TextoML par rapport aux méthodes neuronales est que ce modèle se comporte de façon plus stable (en évitant des dérives de couleurs comme dans le premier exemple de la Fig. 2.8), et permet de mieux comprendre les cas d'erreurs (et donc les limites du modèle). Curieusement, lorsque l'on cherche à examiner la distribution des patches de toutes ces images synthétisées, on s'aperçoit que toutes ces méthodes respectent relativement bien la distribution de patches de la texture exemple, même si elles ne cherchent pas directement à s'en rapprocher.

2.3.3 Autres Applications

Terminons cette section en mentionnant brièvement deux autres applications du modèle TextoML. La première concerne l'inpainting de textures, où l'on cherche à compléter une région masquée dans une texture. Si l'on dispose d'un modèle pour reproduire cette texture, il s'agit essentiellement de tirer un échantillon du



Original | 1-couche w = 3 | 1-couche w = 7 | 2-couches w = 7 | 2-couches w = 7 et PP

FIGURE 2.7 – **Comparaison entre Texto et TextoML.** On montre ici des résultats de synthèse de textures obtenus avec les modèles Texto et TextoML (avec des transports à $L = 2$ couches) sur des patches de tailles $\mathsf{w} = 3$ ou $\mathsf{w} = 7$. Sur la droite, on montre aussi la synthèse en intégrant l'étape de *post-processing* (PP) sur les patches 3×3.

modèle en conditionnant par les valeurs observées en dehors du masque. Cependant, l'inpainting de textures est en pratique plus difficile que la synthèse simple, parce que, si la texture non masquée n'est pas exactement fidèle au modèle, un humain va pouvoir distinguer aisément l'interface entre les deux textures au bord du masque. Ainsi, l'inpainting de textures nécessite d'avoir un modèle de textures suffisamment riche pour faire un raccord indiscernable au bord du masque.

Le modèle TextoML est assez riche pour aborder cette tâche, et sa construction multi-échelle s'adapte bien au cas de l'inpainting, comme illustré sur la Fig. 2.9, en ciblant la distribution des patches non masqués. En effet, à la résolution grossière, la région masquée peut être complétée en utilisant une simulation conditionnelle du champ gaussien estimé hors du masque, sachant les valeurs au bord du masque, comme décrit dans [J5]. Ensuite, à chaque échelle, on peut transformer les patches touchant le masque par des applications de transport multi-couche envoyant vers la distribution de patches hors du masque. On peut alors passer d'une échelle à l'autre en faisant le même sur-échantillonnage basé exemple (en utilisant les patches non masqués). Sur les exemples de la Fig. 2.9 et de [J12], on peut constater que cette méthode permet de prolonger les structures géométriques au bord du masque. Elle est cependant limitée par la petite quantité de flou que l'on observait déjà dans les résultats de synthèse, mais qui sont encore plus visibles lorsque l'on observe côte-à-côte la texture d'origine et la texture synthétisée.

Pour terminer, mentionnons que le modèle TextoML permet aussi d'aborder le problème du transfert de style, qui consiste à appliquer à une image naturelle l'aspect textural donné par une texture exemple. Pour cela, la proposition faite dans [J12] consiste à estimer le modèle TextoML associé à la texture exemple, et à utiliser les applications de transport par patch directement sur l'image à styliser, de la résolution grossière à la résolution fine. À chaque résolution, les structures géométriques de l'image source sont réimposées grâce à un masque obtenu par extraction de contours. Bien que les applications de transport utilisées ne soient pas spécifiques à l'image source, ce procédé permet tout de même de transférer une partie du style de la texture exemple, comme le montrent les exemples donnés dans [J12].

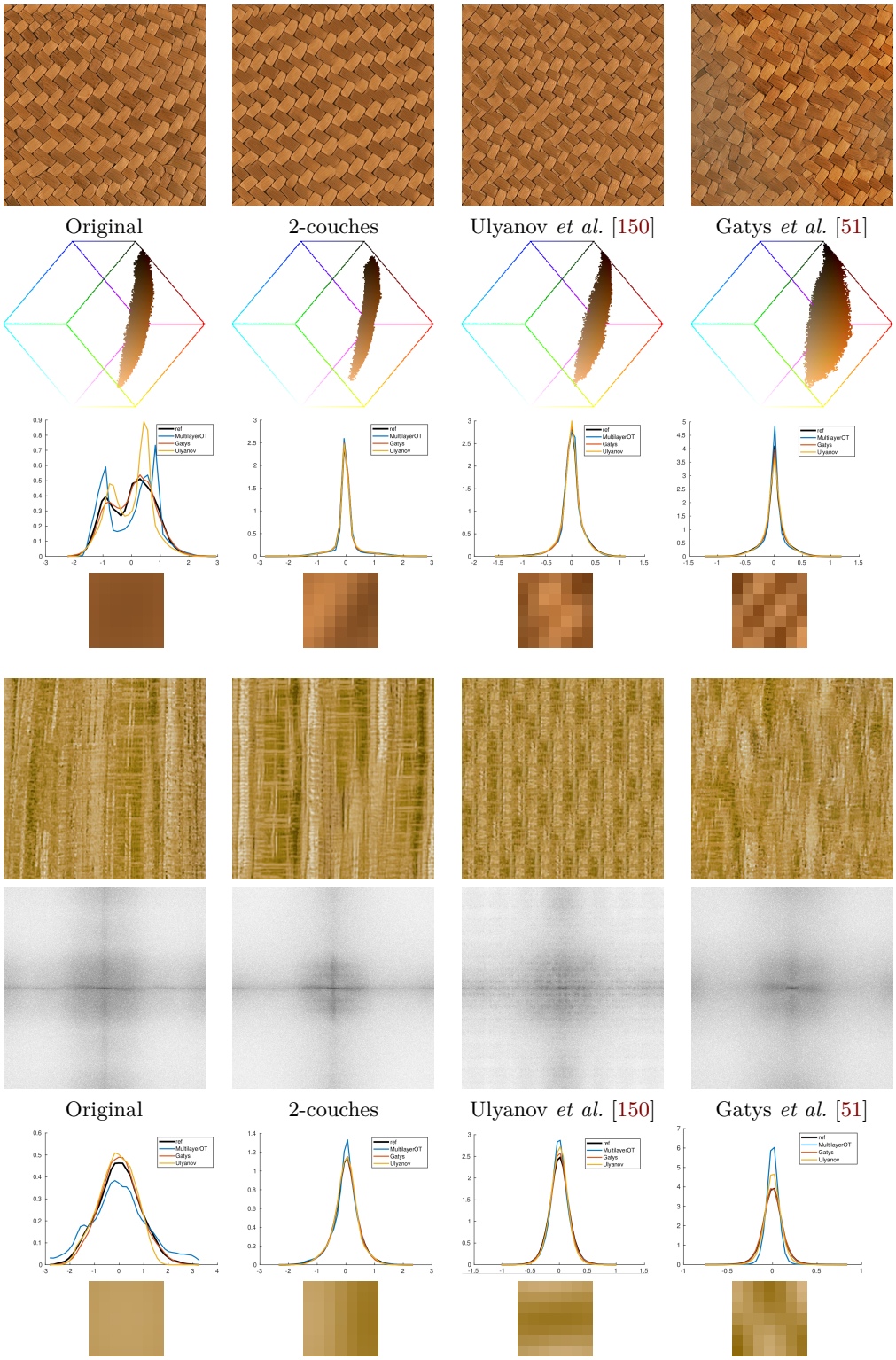


FIGURE 2.8 – Dans cette figure, on compare les synthèses obtenues avec TextoML et les méthodes neuronales de [51, 150]. On montre aussi les distributions de patch obtenues (lignes 3, 7) en projetant sur quelques composantes principales (lignes 4, 8), ainsi que les distributions couleurs (ligne 2) et le spectre de Fourier (ligne 6).

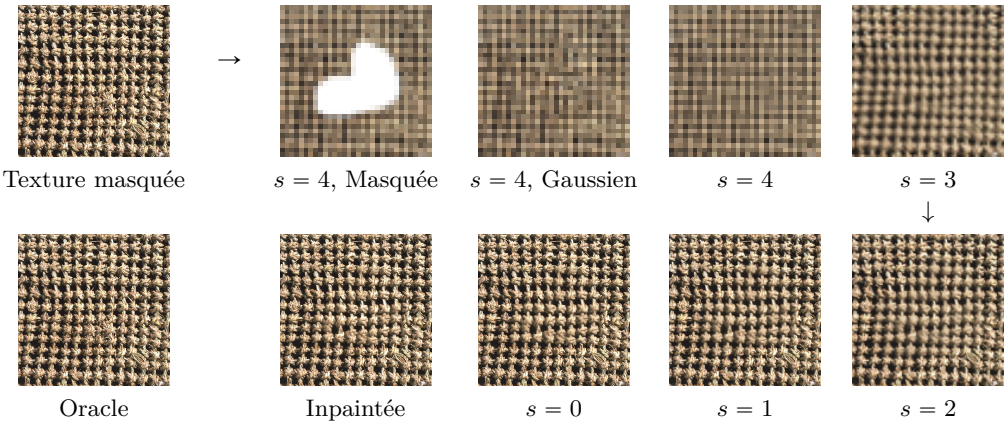


FIGURE 2.9 – **Inpainting de textures avec TextoML.** Le modèle TextoML s'adapte à l'inpainting en utilisant une synthèse conditionnelle du champ gaussien à la résolution grossière, et en utilisant des transports multi-couche par patch à chaque résolution (en ciblant la distribution de patches hors du masque).

2.4 Transport Optimal GMM par Patch

Nous terminons ce chapitre avec des travaux plus récents portant sur l'utilisation d'une formulation du transport optimal spécifique aux GMM, notée GMMOT. Cette nouvelle variante du coût de transport peut être calculée efficacement (même en grande dimension) car elle repose sur un problème de transport discret de petite taille opérant sur les composantes gaussiennes au départ et à l'arrivée. On va voir ci-dessous comment l'appliquer au transport optimal par patch, et comment le GMMOT peut être intégré au modèle de textures Texto précédemment décrit. Ceci débouche sur un nouveau modèle TextoGMM [C16] permettant une estimation du modèle beaucoup plus rapide qu'avec Texto ou TextoML, tout en s'appliquant encore sur de grands patches.

2.4.1 Transport Optimal GMM

L'idée proposée par Delon et Desolneux [41] consiste à formuler un problème de transport sur $\mathbb{R}^D$ en restreignant les couplages admissibles à être des GMM sur $\mathbb{R}^D \times \mathbb{R}^D$. Pour tout $d \in \mathbb{N}^*$, notons GMM_d l'ensemble des mesures de probabilités qui s'écrivent comment un mélange de gaussiennes sur $\mathbb{R}^d$. En suivant [41], on définit une distance MW_2 sur GMM_D en posant

$$\forall \mu_0, \mu_1 \in \mathsf{GMM}_D, \quad MW_2^2(\mu_0, \mu_1) := \inf_{\gamma \in \Pi^{\mathrm{GMM}}(\mu_0, \mu_1)} \int_{\mathbb{R}^{2D}} \|y_0 - y_1\|^2 d\gamma(y_0, y_1), \tag{2.19}$$

où $\Pi^{\mathrm{GMM}}(\mu_0, \mu_1)$ est l'ensemble des $\gamma \in \mathsf{GMM}_{2D}$ de marginales μ_0 et μ_1. De plus, si $\mu_0 = \sum_{k=1}^{K_0} \pi_0^k \mu_0^k$ et $\mu_1 = \sum_{l=1}^{K_1} \pi_1^l \mu_1^l$, où les π_0^k, π_1^l sont des scalaires positifs de somme 1, et les μ_0^k, μ_1^l des mesures gaussiennes, on peut montrer que

$$MW_2^2(\mu_0, \mu_1) = \min_{w \in \Pi(\pi_0, \pi_1)} \sum_{k,l} w_{kl} W_2^2(\mu_0^k, \mu_1^l), \tag{2.20}$$

où $\Pi(\pi_0, \pi_1)$ est l'ensemble des matrices $w \in \mathbb{R}_+^{K_0 \times K_1}$ de marginales π_0 et π_1. Cette définition aboutit à un problème de transport optimal discret dont le coût entre composantes gaussiennes s'exprime explicitement par

$$W_2^2(\mathcal{N}(m,\Sigma),\mathcal{N}(\tilde{m},\tilde{\Sigma})) = \|m-\tilde{m}\|^2 + \operatorname{tr}\left(\Sigma + \tilde{\Sigma} - 2\left(\Sigma^{\frac{1}{2}}\tilde{\Sigma}\Sigma^{\frac{1}{2}}\right)^{\frac{1}{2}}\right). \tag{2.21}$$

Le problème de transport discret (2.20) peut être résolu par programmation linéaire (peu coûteux car K_0, K_1 sont en pratique souvent < 100).

Cependant, en restreignant ainsi les couplages à GMM_{2D}, on s'écarte du problème initial, si bien que le couplage optimal γ^* de (2.19) n'est en général pas associé à une vraie application de transport (il n'est pas supporté sur le graphe d'une application). Pour l'adaptation à la synthèse de textures, on préférera donc utiliser l'application

$$T(x) = \mathbb{E}_{(X,Y)\sim\gamma^*}(Y|X=x), \tag{2.22}$$

qui peut se réécrire

$$T(x) = \frac{\sum_{k,l} w^*_{k,l} g_{m_0^k,\Sigma_0^k}(x) T_{k,l}(x)}{\sum_k \pi_0^k g_{m_0^k,\Sigma_0^k}(x)}, \tag{2.23}$$

où w^* est la solution optimale du problème discret (2.20), et les $T_{k,l}$ sont les cartes affines optimales entre les gaussiennes μ_0^k et μ_1^l, et $g_{m,\Sigma}$ est la densité de $\mathcal{N}(m,\Sigma)$. On notera néanmoins que cette application T ne satisfait pas la contrainte marginale du transport puisque $T\sharp\mu_0 \neq \mu_1$ en général.

2.4.2 Modèle TextoGMM

Cette formulation GMMOT peut être intégrée dans le modèle Texto de la Section 2.2. Le modèle proposé, noté TextoGMM, consiste à appliquer à plusieurs résolutions $s = S-1, \dots, 0$, des applications de transport GMM par patch, en initialisant l'échelle grossière $s = S-1$ par le champ gaussien. Le passage d'une échelle à la suivante se fait par une étape de sur-échantillonnage basée exemple.

Supposons que la texture U_s a été synthétisée à l'échelle s. Comme pour Texto, on utilise une application de transport optimal entre la distribution μ_s des patches de U_s et la distribution des patches ν_s de la texture exemple u_s. Avec l'algorithme EM, on estime des GMM à K composantes $\tilde{\mu}_s, \tilde{\nu}_s$ qui approchent μ_s, ν_s respectivement. On peut alors calculer le plan de transport γ_s solution de $MW_2(\mu_s,\nu_s)$, et celui-ci induit une application de transport T_s via (2.23). Cette application est utilisée pour transporter les patches de U_s, que l'on recompose ensuite en

$$\forall a \in 2^s\mathbb{Z}^2, \quad \tilde{U}_s(a) = \frac{1}{|\omega|}\sum_{b\in 2^s\omega} T_s(U_{s|a-b+2^s\omega})(b). \tag{2.24}$$

Pour pouvoir sur-échantillonner, on compose avec une projection au plus proche voisin dans les patches de u_s : on pose

$$\forall a \in 2^s\mathbb{Z}^2, \quad V_s(a) = \frac{1}{|\omega|}\sum_{b\in 2^s\omega} u_s(C_s(a-b)+b), \tag{2.25}$$

où la carte de coordonnée C_s est définie par

$$\forall a \in 2^s\mathbb{Z}^2, \quad C_s(a) = \underset{a' \text{ t.q. } a'+2^s\omega\subset\Omega_s}{\text{Argmin}} \|T_s(U_{s|a+2^s\omega}) - u_{s|a'+2^s\omega}\|^2. \tag{2.26}$$

L'échelle suivante est initialisée en utilisant les patches deux fois plus grands comme dans (2.10). Ceci permet d'obtenir en sortie l'image synthétisée V_0.

Comme pour les modèles Texto et TextoML, l'estimation du modèle TextoGMM associé à une texture originale nécessite une passe de synthèse (en estimant les distributions de patches et les transport GMMOT à chaque résolution). Mais une fois estimé, le modèle peut être utilisé pour la synthèse, à la demande, de nouvelles textures potentiellement beaucoup plus grandes.

2.4.3 Synthèse avec le Modèle TextoGMM

Le modèle TextoGMM peut être vu comme une version allégée des modèles Texto et TextoML, dans laquelle les applications de transport par patch s'écrivent comme une combinaison convexe bien choisie d'applications affines, composée avec une projection NN. L'intérêt de cette approche est qu'elle s'applique immédiatement en très grande dimension, et permet donc de traiter de grands patches, de même que TextoML, au prix d'une légère dégradation de la qualité visuelle, comme on peut le voir sur la Fig. 2.10. De plus, le modèle TextoGMM est plus rapide à estimer que les modèles Texto et TextoML, car il repose sur la solution du GMMOT qui est aisément calculable, même en très grande dimension. En fait, l'étape la plus coûteuse de l'estimation du modèle TextoML est l'estimation des modèles GMM avec l'algorithme EM. Pour accélérer cette étape ainsi que les projections NN, on limite le cardinal des distributions de patches utilisées à 10^4 points (donc dix fois plus qu'avec Texto). Pour donner un ordre de grandeur, résoudre le GMMOT sur des distributions à 10^4 points en dimension 75 (patches couleur 5×5), prend quelques secondes (en incluant l'étape EM), ce qui est bien inférieur au temps de calcul requis pour l'estimation du transport semi-discret vers une distribution cible à 10^4 points. En tout, l'estimation du modèle TextoGMM associé à une texture exemple, avec des patches 5×5, prend environ 15" sur un ordinateur récent.

Sur la Fig 2.10, on compare les synthèses obtenues avec les modèles Texto, TextoML et TextoGMM. Ces résultats confirment que le modèle TextoGMM atteint une qualité de synthèse proche de celle obtenue avec le modèle TextoML de la section précédente, tout en reposant sur un modèle plus léger, et plus rapide à estimer. Du point de vue visuel, on constate encore la limitation commune de ces trois modèles, provenant du procédé d'agrégation des patches qui introduit une petite quantité de flou. Nous renvoyons à [C16] pour une analyse plus détaillée des résultats. Mentionnons aussi, pour terminer, que le modèle TextoGMM peut (comme TextoML) s'adapter au transfert de style, mais permet également de faire des mélanges de textures, en exploitant la formule close des barycentres GMMOT donnée dans [41]. On montre en Fig. 2.11 quelques résultats de mélanges de texture avec le modèle TextoGMM.

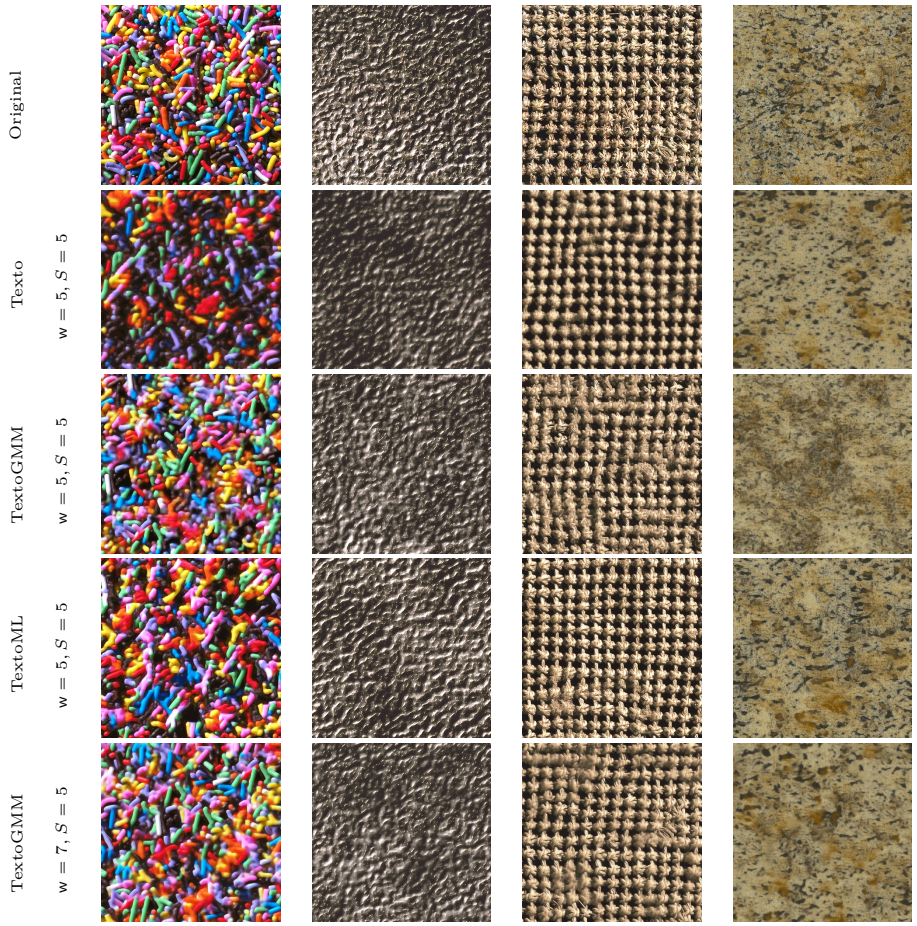


FIGURE 2.10 – **Comparaison de Texto, TextoML et TextoGMM**. Pour plusieurs textures exemples 512×512 (colonne 1), on montre les synthèses obtenues avec ces trois modèles et différents paramètres w (taille de patch) et S (nombre d'échelles). On conseille de zoomer sur les images pour mieux percevoir les détails.

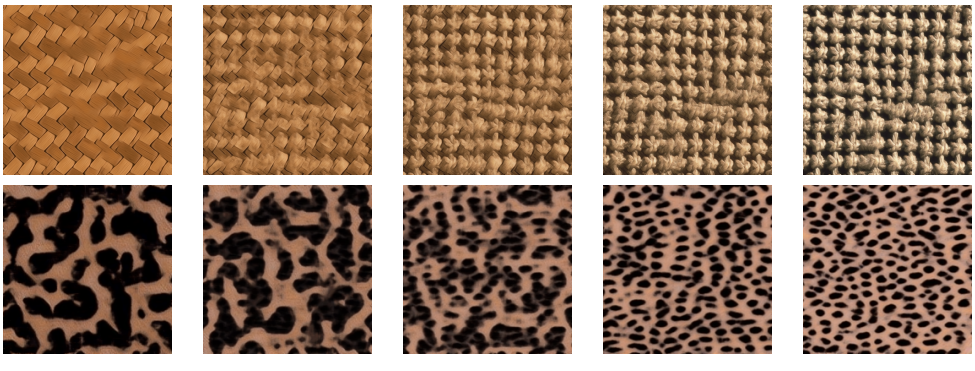


FIGURE 2.11 – **Mélange de textures avec le modèle TextoGMM.** On montre des échantillons du modèle de textures mélangé avec pondération $\alpha \in [0, 1]$. On voit que l'utilisation de barycentres GMMOT permet de mélanger de façon non-linéaire les structures géométriques des textures exemples.

2.5 Conclusion Partielle

Dans ce chapitre, nous avons introduit un modèle de textures construit de façon multi-résolution grâce à des applications de transport optimal dans l'espace des patches, en exploitant notamment le fait que le transport optimal semi-discret admet des solutions sous la forme d'assignements au plus proche voisin biaisés. Nous avons expliqué trois variantes de ce modèle, Texto, TextoML et TextoGMM, basées sur des approximations différentes du problème de transport optimal. Ces trois variantes permettent de synthétiser des textures structurées, tout en conservant de bonnes garanties théoriques (stationnarité, indépendance longue portée) ainsi qu'un temps de calcul avantageux (grâce à la possibilité de paralléliser les transports par patch).

La limitation principale de ce modèle en termes de qualité visuelle vient de la façon de recomposer une image à partir des patches transportés : l'agrégation par simple moyenne induit toujours une petite quantité de flou. Autrement dit, le défaut de ce modèle est qu'on ne peut imposer la distribution des patches qu'avant l'étape d'agrégation, et pas après. On proposera dans le chapitre suivant un autre modèle de textures visant à contraindre directement la distribution des patches de la synthèse. Ceci nécessite une formulation mathématique différente et plus lourde à mettre en œuvre. Notons aussi que le principe de recopie par patches contraint la capacité d'innovation de l'algorithme, ce qui peut poser problème si les paramètres du modèle sont mal adaptés à la texture exemple (par exemple s'il n'y a que trop peu de patches dans la texture à basse résolution). Sur cette question, mentionnons que nous avons développé dans [J8, J9] des outils statistiques pour détecter la redondance par patch.

L'architecture du modèle Texto peut être considérée comme un réseau génératif très léger, avec des couches de transports par patches paramétrées par les sous-ensembles de patches utilisés et les scalaires v associés. Cette approche pourrait maintenant être rattachée aux réseaux basés sur des couches d'attention [155]. De plus, en comparaison, les réseaux génératifs proposés par Ulyanov *et al.* [150], Shaham *et al.* [135] sont plus difficiles à entraîner (en raison d'un grand nombre de paramètres) et, bien qu'ils produisent des textures plus nettes, respectent moins bien les statistiques de la texture d'entrée.

Les différentes variantes de Texto et les travaux de Shaham *et al.* [135] montrent que, si l'on veut produire une texture de bonne qualité visuelle, il n'est pas forcément nécessaire de résoudre à très grande précision le problème de transport optimal par patch. Nous pourrions donc accélérer encore l'étape de synthèse avec une nouvelle approximation du transport semi-discret, en combinant une recherche approchée de plus proches voisins comme dans [29] et une représentation condensée des patches.

Chapitre 3

Réseaux Génératifs Wasserstein

Dans ce chapitre sont présentés des travaux traitant de l'apprentissage de modèles génératifs à l'aide de distances de transport optimal, menés en collaboration avec d'une part Antoine Houdard, Nicolas Papadakis et Julien Rabin, et d'autre part Jérémie Bigot et Paul Freulon. Dans la Section 3.1.3 on pose le cadre de l'estimation de réseaux génératifs Wasserstein (WGAN pour *Wasserstein Generative Adversarial Networks*) et l'on étudie la fonctionnelle associée en adoptant une formulation semi-discrète du coût de transport optimal [J15]. La Section 3.2 résume les travaux publiés dans [C10, J14], où l'estimation de réseaux génératifs est appliquée au problème de synthèse de textures, en considérant des distances de transport optimal par patch à plusieurs résolutions. Enfin, la Section 3.3 [Js2] offre une perspective statistique sur ce problème, où l'on cherche à contrôler l'erreur d'estimation faite lorsque l'on a seulement accès à des échantillons des mesures.

3.1 Estimation Wasserstein de Réseaux Génératifs

L'apprentissage de modèles génératifs est une problématique désormais répandue de la science des données, qui consiste à approcher une distribution empirique de données par une distribution paramétrée facilement simulable. Ceci permet de générer des échantillons ressemblant aux données, d'interpoler entre plusieurs points des données, de révéler une structure géométrique sous-jacente aux données [122, 137], ou encore de fournir un modèle a priori permettant de régulariser des problèmes inverses [16, 67, 73, 68, 111, 136, 37, 93].

Étant donné la distribution empirique des données ν à support dans un compact $\mathcal{Y} \subset \mathbb{R}^d$, l'apprentissage d'un réseau génératif consiste à résoudre

$$\operatorname*{Argmin}_{\theta \in \Theta} \mathcal{L}(\mu_\theta, \nu) \tag{3.1}$$

où μ_θ est la loi du modèle génératif à support dans un compact $\mathcal{X} \subset \mathbb{R}^d$ et paramétrée par un θ dans un ouvert $\Theta \subset \mathbb{R}^q$, et où $\mathcal{L}$ est une fonction permettant d'évaluer la proximité entre lois de probabilités. Souvent, la distribution μ_θ est la loi d'une v.a. $g_\theta(Z)$ où g_θ est un réseau de neurones et Z une v.a. de loi fixée.

Dans l'article [59] introduisant les réseaux génératifs adverses (GAN), la fonction de coût utilisée est la divergence de Jensen-Shannon exprimée de façon duale, menant à un problème minimax. Cette divergence n'étant pas bien adaptée au cas où μ_θ, ν n'ont pas le même support, Arjovsky *et al.* [2] ont proposé de minimiser le coût de transport optimal $W(\mu_\theta, \nu)$, définissant ainsi les Wasserstein GANs (WGANs). En statistique, ce problème est aussi parfois appelé "estimation Wasserstein" [5, 8], dénomination que nous adopterons ici.

L'une des difficultés de l'estimation Wasserstein vient du fait que la loi μ_θ est une distribution continue en grande dimension, et sa résolution numérique passe souvent par l'utilisation de solveurs duaux pour $W(\mu_\theta, \nu)$. En se restreignant au coût Wasserstein-1, on peut simplifier le problème en n'optimisant qu'une seule variable duale paramétrée par un réseau 1-lipschitzien (par méthode *weight clipping* dans [2] ou *gradient penalty* dans [65]). À l'inverse, les auteurs de [134, 129, 98] conservent un coût générique mais adoptent une régularisation entropique du transport optimal, avec différentes paramétrisations des variables duales. En exploitant le fait que ν est discrète, plusieurs auteurs [28, 104] ont exploité la formulation semi-discrète du transport optimal en paramétrant l'une des variables comme une c-transform.

On donnera ci-dessous la formule (Grad-OT) exprimant le gradient de $\theta \mapsto W(\mu_\theta, \nu)$ grâce à la solution duale du coût $W(\mu_\theta, \nu)$. Dans [105, 140, 83, 82] a été soulevée la question de l'impact de l'erreur commise par le solveur dual de $W(\mu_\theta, \nu)$ dans la qualité du réseau génératif estimé. Mallasto *et al.* [105] ont montré qu'en utilisant des méthodes plus précises pour estimer le coût de transport optimal, on pouvait obtenir *in fine* un modèle génératif produisant des échantillons de moins bonne qualité visuelle. Stanczuk *et al.* [140] ont confirmé ces résultats et proposent plusieurs raisons à ce phénomène.

Notre étude vise à mieux comprendre le comportement de ces algorithmes d'apprentissage de réseaux génératifs. Après avoir introduit en détail le transport optimal régularisé dans la Section 3.1.1 et le problème d'estimation Wasserstein dans la Section 3.1.2, nous donnerons dans la Section 3.1.3 des conditions suffisantes permettant de justifier la formule (Grad-OT) qui sous-tend l'optimisation du réseau génératif. En explicitant cette formule dans un cadre semi-discret (et sans se limiter au coût Wasserstein-1), nous examinerons l'algorithme d'optimisation alternée utilisé pour l'apprentissage du réseau génératif, et nous verrons comment interpréter l'erreur faite par le solveur dual grâce à la visualisation du diagramme de Laguerre correspondant. Les simulations présentées dans la Section 3.1.4 montreront que l'optimisation du générateur peut se stabiliser ou osciller selon les stratégies adoptées sur les pas de gradient. On examinera aussi l'impact du paramètre de régularisation entropique $\lambda \geqslant 0$ dans l'apprentissage, sur des cas synthétiques en dimension 2 ou réels en dimension 784 avec la base de données MNIST.

3.1.1 Transport Optimal Régularisé

Dans ce paragraphe, on rappelle la définition et la formulation duale du coût de transport optimal régularisé, que l'on utilise pour l'estimation de réseaux génératifs.

Soient $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^d$ compacts, et $c : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathbb{R}$ continue.

Définition 3.1.1 (Formulation Primale)**.** Soient μ, ν deux mesures de probabilité sur $\mathcal{X}, \mathcal{Y}$ respectivement. Pour $\lambda \geqslant 0$, le coût de transport optimal régularisé s'écrit

$$W_\lambda(\mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} c \, d\pi + \lambda \, \mathrm{KL}(\pi | \mu \otimes \nu) \tag{3.2}$$

où $\Pi(\mu, \nu)$ est l'ensemble des lois de probabilité sur $\mathcal{X} \times \mathcal{Y}$ de marginales μ et ν, et où $\mathrm{KL}(\pi|\mu\otimes\nu) = \int \log(\frac{d\pi}{d\mu\otimes\nu}) d\pi$ si la densité $\frac{d\pi}{d\mu\otimes\nu}$ existe et $+\infty$ sinon.

Théorème 3.1.1 (Formulation duale [130, 55]). *Le coût de transport régularisé s'exprime aussi par*

$$W_\lambda(\mu,\nu) = \max_{\varphi\in\mathscr{C}(\mathcal{X}),\psi\in\mathscr{C}(\mathcal{Y})} \int \varphi(x)d\mu(x) + \int \psi(y)d\nu(y) - \int m_\lambda(\varphi(x)+\psi(y)-c(x,y))d\mu(x)d\nu(y) \tag{3.3}$$

où pour $\lambda = 0$, $m_0(t) = 0$ *si* $t \geqslant 0$, *et* $+\infty$ *sinon, et pour* $\lambda > 0$, $m_\lambda(t) = \lambda(e^{\frac{t}{\lambda}} - 1)$. *Pour* $\lambda > 0$, *les solutions sont uniques à constantes près (*i.e. *si* (φ,ψ) *est solution, alors les autres s'écrivent* $(\varphi - k, \psi + k)$ *avec* $k \in \mathbb{R}$*).*

Pour $\psi \in \mathscr{C}(\mathcal{Y})$, on définit la c-transform régularisée par

$$\forall x \in \mathcal{X}, \quad \psi^{c,\lambda}(x) = \operatorname*{softmin}_{y\in\mathcal{Y}} c(x,y) - \psi(y) \tag{3.4}$$

où

$$\operatorname*{softmin}_{y\in\mathcal{Y}} f(y) = \begin{cases} \min_{y\in\mathcal{Y}} f(y) & \text{if } \lambda = 0, \\ -\lambda \log \int e^{-\frac{f(y)}{\lambda}} d\nu(y) & \text{if } \lambda > 0. \end{cases} \tag{3.5}$$

On définit la c-transform analogue sur la variable x. On peut montrer que le problème dual (3.3) peut être restreint aux fonctions φ, ψ s'écrivant comme des c-transforms, qui héritent de la régularité de c et qui peuvent être extrapolées.

Théorème 3.1.2 (Formulation semi-duale [55]). *Le coût de transport optimal régularisé s'exprime aussi par*

$$W_\lambda(\mu,\nu) = \sup_{\psi\in\mathscr{C}(\mathcal{Y})} \int \psi^{c,\lambda}(x)d\mu(x) + \int \psi(y)d\nu(y). \tag{3.6}$$

3.1.2 Le Problème d'Estimation Wasserstein

Soit $\Theta \subset \mathbb{R}^q$ ouvert. Estimer un réseau génératif Wasserstein (WGAN) pour la distribution des données ν revient à minimiser

$$h_\lambda(\theta) = W_\lambda(\mu_\theta, \nu), \tag{3.7}$$

où le modèle génératif μ_θ est la loi de $g_\theta(Z)$ avec Z une v.a. de loi fixée ζ sur un espace mesurable $\mathcal{Z}$. On note $g_\theta = g(\theta,\cdot)$ où $g : \Theta \times \mathcal{Z} \longrightarrow \mathbb{R}^d$ est définie sur le produit $\Theta \times \mathcal{Z}$. Nous allons donner des conditions suffisantes permettant de calculer le gradient de h_λ. La proposition suivante énonce la formule désirée, qui est vraie sous réserve d'existence des gradients. Pour cela, définissons, pour $\varphi \in \mathscr{C}(\mathcal{X}), \psi \in \mathscr{C}(\mathcal{Y}), \theta \in \Theta$,

$$I(\varphi,\theta) = \int_{\mathcal{X}} \varphi d\mu_\theta = \mathbb{E}[\varphi(g_\theta(Z))], \tag{3.8}$$

$$F_\lambda(\psi,\theta) = I(\psi^{c,\lambda},\theta) = \int_{\mathcal{X}} \psi^{c,\lambda} d\mu_\theta = \mathbb{E}[\psi^{c,\lambda}(g_\theta(Z))], \tag{3.9}$$

$$H_\lambda(\psi,\theta) = F_\lambda(\psi,\theta) + \int_{\mathcal{Y}} \psi d\nu, \tag{3.10}$$

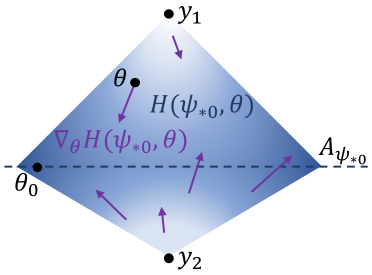


FIGURE 3.1 – On illustre ici le contre-exemple de la Proposition 3.1.4 pour $p = 2$. On voit que $\theta \mapsto H(\psi_{*0}, \theta)$ n'est pas différentiable en θ_0.

de sorte que

$$W_\lambda(\mu_\theta, \nu) = h_\lambda(\theta) = \max_{\psi \in \mathscr{C}(\mathcal{Y})} F_\lambda(\psi, \theta) + \int_{\mathcal{Y}} \psi d\nu = \max_{\psi \in \mathscr{C}(\mathcal{Y})} H_\lambda(\psi, \theta). \tag{3.11}$$

Pour $\lambda = 0$, on utilisera les notations sans indice h, W, H, and F.

Proposition 3.1.3 ([2]). *Soient θ_0, ψ_0 tels que $h_\lambda(\theta_0) = H_\lambda(\psi_0, \theta_0)$.*
Si h_λ et $\theta \mapsto F_\lambda(\psi_0, \theta)$ sont différentiables en θ_0, alors

$$\nabla h_\lambda(\theta_0) = \nabla_\theta F_\lambda(\psi_0, \theta_0). \tag{Grad-OT}$$

Montrer la formule (Grad-OT) revient à différentier sous le maximum dans (3.11). N'intervient dans $\nabla h_\lambda(\theta_0)$ que le potentiel ψ_0 optimal pour $W_\lambda(\mu_{\theta_0}, \nu)$. Dans la section suivante, nous donnerons des conditions suffisantes permettant de différentier sous le maximum rigoureusement. Cependant, notons qu'en toute généralité, à θ_0 fixé et ψ_0 le potentiel optimal correspondant, la fonction $F_\lambda(\psi_0, \theta)$ peut présenter une singularité en θ_0. Ceci arrive par exemple dans le cas discret de la proposition suivante, illustré en Fig. 3.1.

Proposition 3.1.4. *Soit $\mu_\theta = \delta_\theta$ et $\nu = \frac{1}{2}\delta_{y_1} + \frac{1}{2}\delta_{y_2}$ pour $\theta, y_1, y_2 \in \mathbb{R}^d$. Pour $p \geqslant 1$, soit $c(x, y) = \|x - y\|^p$, où $\|\cdot\|$ est la norme euclidienne sur $\mathbb{R}^d$. Alors*

- *$h(\theta) = W(\mu_\theta, \nu)$ est différentiable sur $\theta \notin \{y_1, y_2\}$ pour $p = 1$, et en tout $\theta \in \mathbb{R}^d$ pour $p > 1$,*
- *$\forall \psi_{*0} \in \operatorname{Arg\,max}_\psi H(\psi, \theta_0)$, $\theta \mapsto F(\psi_{*0}, \theta)$ n'est pas différentiable en θ_0.*

Ainsi, la formule (Grad-OT) n'a pas de sens dans ce cas.

3.1.3 Gradient du Coût d'Estimation Wasserstein

On va maintenant formuler des conditions permettant de différentier h_λ. Les hypothèses concernent en premier lieu la régularité du coût et du générateur. On décline l'étude en deux parties : on étudie d'abord le cas semi-discret non-régularisé (Théorème 3.1.6 et Théorème 3.1.5) qui nécessite une hypothèse supplémentaire sur le positionnement de g_{θ_0} par rapport à un diagramme de Laguerre ; on évoquera ensuite le cas régularisé (Théorème 3.1.7) ne nécessitant pas cette hypothèse technique.

Définition 3.1.2. On dit que c est x-régulier s'il existe $L > 0$ et un ouvert U tel que $\mathcal{X} \subset U \subset \mathbb{R}^d$ tels que pour tout $y \in \mathcal{Y}$, $c(\cdot, y)$ s'étend en une fonction $\mathscr{C}^1$ et L-Lipschitz sur U.

Définition 3.1.3. Pour tout $\theta_0 \in \Theta$, on dit que $g : \Theta \times \mathcal{Z} \longrightarrow \mathcal{X}$ satisfait (G_{θ_0}) s'il existe un voisinage V de θ_0 et $K \in L^1(\zeta)$ tels que $\theta \mapsto g(\theta, Z)$ est p.s. $\mathscr{C}^1$ sur V de différentielle $\theta \mapsto D_\theta g(\theta, Z)$ et que

$$\forall \theta \in V, \quad \zeta\text{-p.s..} \quad \|g(\theta, Z) - g(\theta_0, Z)\| \leqslant K(Z)\|\theta - \theta_0\|. \tag{3.12}$$

On dit que g satisfait (G_Θ) si g satisfait (G_{θ_0}) en tout $\theta_0 \in \Theta$.

Dans [J15] on a montré que l'hypothèse est vérifiée pour une large classe de réseaux de neurones avec activations $\mathscr{C}^1$ et avec un bruit d'entrée Z intégrable.

Pour exprimer la dernière hypothèse technique, on se place dans le cas du transport semi-discret où ν est supportée par un ensemble $\mathcal{Y}$ de cardinal J. On utilise les notations introduites dans l'introduction (Section 1.1 et Fig. 1.2) pour les applications de transport semi-discret. La différentiation se fait en deux temps, en différentiant d'abord F, puis h.

Lemma 3.1.1. *Supposons $\mathcal{Y}$ fini à J points et que c est x-régulier. Supposons que g satisfait (G_{θ_0}) en $\theta_0 \in \Theta$. Supposons que pour tout $\psi \in \mathbb{R}^J$, $\mu_{\theta_0}(A_\psi) = 0$,* i.e. *$g(\theta_0, Z) \in \mathcal{X}\backslash A_\psi$ p.s. Alors, pour tout $\psi \in \mathbb{R}^J$, $\theta \mapsto F(\psi, \theta)$ est différentiable en θ_0 et l'on a*

$$\nabla_\theta F(\psi, \theta_0) = \mathbb{E}[D_\theta g(\theta_0, Z)^T \nabla \psi^c(g(\theta_0, Z))]. \tag{3.13}$$

Théorème 3.1.5. *Supposons $\mathcal{Y}$ fini à J points et que c est x-régulier.*

Supposons aussi que

1. *pour tout $\theta \in \Theta$, le potentiel de Kantorovich ψ_* pour $W(\mu_\theta, \nu)$ est unique à constante près,*
2. *pour tout $\theta \in \Theta$ et tout $\psi \in \mathbb{R}^J$, $\mu_\theta(A_\psi) = 0$,* i.e. *$g(\theta, Z) \in \mathcal{X}\backslash A_\psi$ p.s.,*
3. *g satisfait (G_Θ) .*

Alors $h(\theta) = W(\mu_\theta, \nu)$ est différentiable en tout $\theta \in \Theta$ et

$$\nabla h(\theta) = \nabla_\theta F(\psi_*, \theta) = \mathbb{E}\left[D_\theta g(\theta, Z)^T \nabla \psi_*^c(g_\theta(Z))\right]. \tag{3.14}$$

L'hypothèse de coût x-régulier n'est pas satisfaite dans certains cas, par exemple dans le cas $c(x, y) = \|x - y\|$ in $\mathbb{R}^d$, engendrant la distance Wasserstein-1 qui est souvent utilisée dans l'apprentissage de WGAN [2]. Le résultat suivant permet d'inclure ce cas moins régulier.

Théorème 3.1.6. *Supposons $\mathcal{Y}$ fini à J points. Supposons que $c : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathbb{R}$ est continu, et qu'il existe $L > 0$, un ouvert U avec $\mathcal{X} \subset U \subset \mathbb{R}^d$ et $B \subset \mathcal{X}$ fermé dans $\mathbb{R}^d$ tel que pour tout $y \in \mathcal{Y}$, $c(\cdot, y)$ s'étend en une fonction $\mathscr{C}^1$ et L-Lipschitz sur $U\backslash B$. Supposons que g satisfait les trois hypothèses du Théorème 3.1.5, et supposons que $\mu_\theta(B) = 0$ pour tout $\theta \in \Theta$.*

Alors $h(\theta) = W(\mu_\theta, \nu)$ est différentiable en tout $\theta \in \Theta$ et son gradient est donné par la formule (3.14).

Terminons le paragraphe en donnant l'adaptation des résultats précédents au cas du transport régularisé. Dans ce cas, les c-transforms sont effectivement régulières partout même à l'interface des cellules de Laguerre. On n'a donc plus besoin de l'hypothèse technique correspondante. En fait, on n'a même pas besoin de se restreindre au cas semi-discret : les deux résultats ci-dessous s'appliquent pour une mesure de probabilité ν sur $\mathcal{Y} \subset \mathbb{R}^d$ compact.

Lemma 3.1.2. *Soit $\lambda > 0$. Supposons que c est x-régulier et que g satisfait (G_Θ). Soit $\psi \in \mathscr{C}(\mathcal{Y})$. Alors $\theta \mapsto F_\lambda(\psi, \theta)$ est différentiable sur Θ et*

$$\forall \theta_0 \in \Theta, \quad \nabla_\theta F_\lambda(\psi, \theta_0) = \mathbb{E}\left[D_\theta g(\theta_0, Z)^T \nabla \psi^{c,\lambda}(g(\theta_0, Z))\right]. \tag{3.15}$$

Théorème 3.1.7. *Supposons que c est x-régulier que g satisfait (G_Θ).*
Alors h_λ est $\mathscr{C}^1$ sur Θ et

$$\forall \theta \in \Theta, \quad \nabla_\theta h_\lambda(\theta) = \nabla_\theta F_\lambda(\psi_*, \theta) = \mathbb{E}\left[D_\theta g(\theta, Z)^T \nabla \psi_*^{c,\lambda}(g(\theta, Z))\right], \tag{3.16}$$

où $\psi_* \in \operatorname{Arg\,max}_\psi H_\lambda(\psi, \theta)$.

Nous avons également prouvé une extension du dernier résultat en remplaçant W_λ par la divergence de Sinkhorn, qui est parfois utilisée en pratique pour débiaiser des résultats d'estimation Wasserstein. On renvoie à [J15] pour ce théorème.

3.1.4 Résultats Numériques sur les WGAN

Dans cette partie, on résume quelques résultats expérimentaux sur le comportement d'un algorithme d'optimisation alternée utilisé pour l'apprentissage d'un réseau génératif Wasserstein, avec un cas de données synthétiques en dimension 2, puis le cas des données MNIST en dimension 784. On renvoie à [J15] pour une discussion détaillée de ces expériences.

Dans ces simulations, $\mathcal{Y}$ est l'ensemble des J points de données, et les éléments $\psi \in \mathscr{C}(\mathcal{Y})$ s'identifient à des vecteurs de $\mathbb{R}^J$. Le problème d'optimisation se réécrit

$$\min_{\theta \in \Theta} h_\lambda(\theta) = \min_{\theta \in \Theta} \max_{\psi \in \mathscr{C}(\mathcal{Y})} H_\lambda(\psi, \theta), \tag{3.17}$$

où

$$H_\lambda(\psi, \theta) = \int_{\mathcal{X}} \psi^{c,\lambda} d\mu_\theta + \int_{\mathcal{Y}} \psi d\nu = F_\lambda(\psi, \theta) + \sum_{y \in \mathcal{Y}} \psi(y) \nu(\{y\}) \ . \tag{3.18}$$

On adopte un algorithme (résumé dans l'Algorithme 4) qui alterne entre une mise à jour de θ (par exemple avec un pas de gradient), et une mise à jour de ψ (par exemple avec plusieurs itérations de l'algorithme ASGD décrit dans la Section 2.2). La mise à jour de θ par un algorithme de gradient utilise la valeur $\nabla_\theta H_\lambda(\psi, \theta) = \nabla F_\lambda(\psi, \theta)$ explicitée dans la partie précédente, ou plus précisément une approximation de Monte-Carlo obtenue sur un batch de b valeurs de $g_\theta(Z)$. En position générique, le générateur ne chargera pas l'interface de la cellule de Laguerre, et on s'attend donc à ne pas tomber sur un point (ψ, θ) singulier au cours de l'algorithme.

Initialisation : θ (aléatoire)
For $n = 1, \dots, N$
· $\psi_n \approx \operatorname{Arg\,max} H_\lambda(\cdot, \theta)$ (K iterations d'ASGD initialisé en ψ_{n-1})
· Mettre à jour θ (un pas d'algorithme ADAM sur $H_\lambda(\psi_n, \cdot)$)
Output : paramètre θ du modèle génératif appris

Algorithm 4: Apprentissage du modèle génératif μ_θ

Sur la Fig. 3.2, on montre le comportement de l'algorithme (le long des itérations n) sur un cas synthétique avec un ensemble $\mathcal{Y}$ à 6 points en dimension 2, et un modèle génératif μ_θ paramétré par un réseau *multilayer perceptron*. Dans ce type d'expériences, on constate que le réseau génératif se stabilise sur une configuration qui couvre bien l'ensemble des données lorsque $\lambda = 0$ et qui dégénère sur un barycentre lorsque λ est trop grand. On constate que la fonction de coût a une tendance décroissante, mais ne converge pas parfaitement. En particulier, quand $\lambda = 0$, selon la stratégie utilisée pour les pas de gradients en ψ et θ, on peut exhiber des cas particuliers où l'algorithme oscille entre différentes configurations. Sur la deuxième ligne de la Fig. 3.2, on constate en particulier que si le nombre d'itérations K de l'algorithme ASGD pour estimer ψ est trop faible, alors le diagramme de Laguerre sous-jacent n'est pas suffisamment ajusté pour permettre une mise à jour précise de θ. Ce cas illustre bien l'impact de l'erreur sur la variable duale dans l'apprentissage du réseau génératif : on n'obtient encore un modèle génératif qui interpole une partie des données mais sans les couvrir entièrement.

Examinons maintenant les résultats montrés en Fig. 3.3 correspondant à l'apprentissage d'un réseau génératif (*multilayer perceptron* MLP, ou réseau convolutionnel DCGAN) pour la génération de chiffres, entraîné sur la base MNIST [92] composée de 60000 données dans $\mathbb{R}^{784}$. Ici, les paramètres θ sont mis à jour avec l'algorithme ADAM [79], et la variable ψ est optimisée avec l'implémentation *Pytorch* de l'algorithme ASGD appliqué directement sur la variable ψ sans paramétrisation (SDOT) ou bien à une paramétrisation de ψ avec un réseau de neurones (SDOTNN). Dans le cas non-régularisé ($\lambda = 0$), on constate que le modèle génératif produit des échantillons convaincants (pour la plupart) légèrement plus flous que ceux de la base de données. Certains échantillons ne ressemblent pas à un chiffre, mais à un mélange de plusieurs chiffres, ce qui illustre encore que le réseau génératif interpole entre plusieurs points des données. Si l'on augmente le paramètre de régularisation λ, le réseau génératif se concentre sur un barycentre flou de toute la base de données.

Sur les résultats numériques supplémentaires reportés dans [J15], en termes de vitesse de convergence, on a pu constater un léger bénéfice à l'utilisation de la paramétrisation neuronale de ψ en particulier pour $\lambda > 0$. On a aussi pu constater que l'utilisation de la régularisation entropique ne change pas radicalement la vitesse de décroissante de la fonction de coût estimée le long des itérés θ_n de l'Algorithme 4 (même si le profil de décroissance observé est plus lisse). Notre étude permet de com-

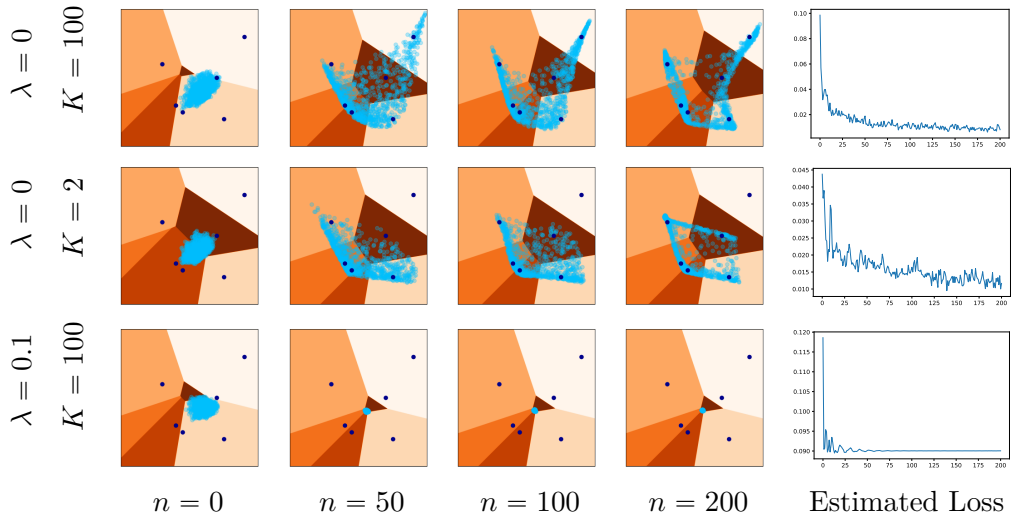


FIGURE 3.2 – **Apprentissage d'un réseau génératif sur un ensemble $\mathcal{Y} \subset \mathbb{R}^2$ à 6 points.** On montre, le long des itérations n de l'Algorithme 4, le positionnement de $\mathcal{Y}$ (bleu foncé) et du modèle génératif courant μ_θ (échantillons en bleu clair). À gauche, on indique le paramètre de régularisation λ et le nombre d'itérations K dans la boucle interne ASGD. On montre aussi l'évolution de la fonction de coût.

menter avec prudence ce type de résultats obtenus sur un cas d'images en grande dimension : même avec une estimation assez grossière de la variable duale, l'optimisation en θ pousse le réseau génératif à couvrir une partie de la base de données. Selon la stratégie d'optimisation utilisée, le réseau génératif peut se stabiliser sur une configuration menant à des échantillons visuellement convaincants, mais sans nécessairement couvrir toute la base de données (ce qui ne peut être observé à la seule vue des échantillons de la Fig. 3.3). On comprend ainsi que la "qualité visuelle" du modèle génératif appris dépend de la fonction de coût utilisée (1-Wasserstein, 2-Wasserstein, Jensen-Shannon, ...) mais aussi crucialement des choix d'architecture du réseau génératif et de stratégie d'optimisation.

3.2 Réseaux Génératifs pour la Synthèse de Textures

Une façon de synthétiser une texture à partir d'un original est de faire en sorte que certaines statistiques de la texture synthétisée soient proches de celles observées dans la texture originale. Les statistiques extraites sont souvent calculées sur des *features*, que l'on entendra ici au sens de "caractéristiques locales" (par exemple, des coefficients d'ondelettes, ou réponses à un réseau de neurones). Pour produire une synthèse respectant ces statistiques, on peut avoir recours à un algorithme itératif qui, partant d'un bruit, cherche à minimiser la distance entre les statistiques calculées sur la synthèse courante et celles calculées sur l'original. Ce type de méthode est parfois qualifiée de *synthèse de textures variationnelle*, et est en lien avec la simulation du modèle microcanonique que l'on introduira dans la Section 4.2

À l'inverse, lorsque l'on cherche à préserver la distribution des patches, ou la

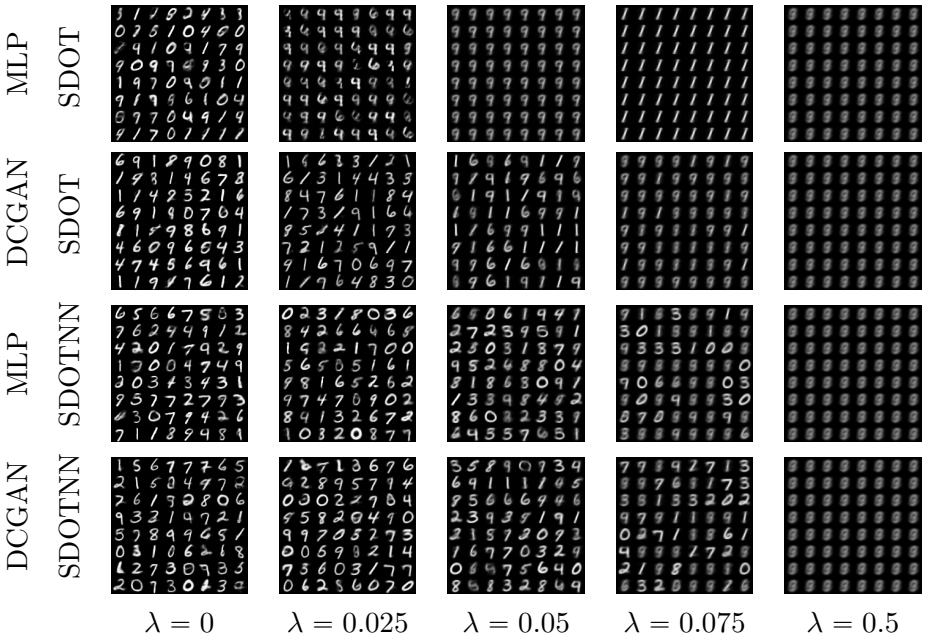


FIGURE 3.3 – **Apprentissage d'un réseau génératif pour la base MNIST.** On compare les modèles génératifs entraînés avec plusieurs architectures pour g_θ (MLP ou DCGAN) et pour la variable duale ψ (sans paramétrisation SDOT, ou avec paramétrisation neuronale SDOTNN). De gauche à droite, on fait varier le paramètre de régularisation λ.

distribution de réponses de filtres, l'information extraite ne se résume pas à un nombre fini de statistiques calculées sur l'image. L'objectif de cette partie est d'introduire un cadre de synthèse de textures variationnelle visant à respecter les distributions d'une collection finie de *features*. On va reprendre le problème de réseaux génératifs présenté dans la section précédente, en calculant le coût de transport optimal désormais sur les distributions des *features* considérées. On en déduira un algorithme d'optimisation alternée, dénommé GOTEX (pour *generative model based on optimal transport for synthesizing textures*) permettant, soit d'optimiser directement la texture synthétisée, soit d'optimiser un réseau convolutionnel pour la synthèse. À la différence du modèle Texto présenté dans le Chapitre 2, ce nouvel algorithme vise à optimiser directement la distribution de *features* en sortie (sans être contraint par une architecture incluant nécessairement l'agrégation des patches).

3.2.1 Coûts de Transport sur les Distributions de *Features*

Partant d'une texture originale $u_0 \in \mathbb{R}^{dm}$ à m pixels, on va synthétiser une texture $v \in \mathbb{R}^{dn}$ à n pixels. Pour chaque pixel i, on considère une application mesurable $f_i : \mathbb{R}^{dn} \longrightarrow \mathbb{R}^D$ servant à extraire une *feature* en position i. Par exemple, $f_i(v)$ peut être un patch couleur de dimension $D = d \times \mathsf{w} \times \mathsf{w}$ en position i, ou encore une réponse neuronale de dimension D en position i. Les *features* de v sont regroupées dans un vecteur $F(v) = (f_i(v))_{1 \leqslant i \leqslant n} \in \mathbb{R}^{Dn}$.

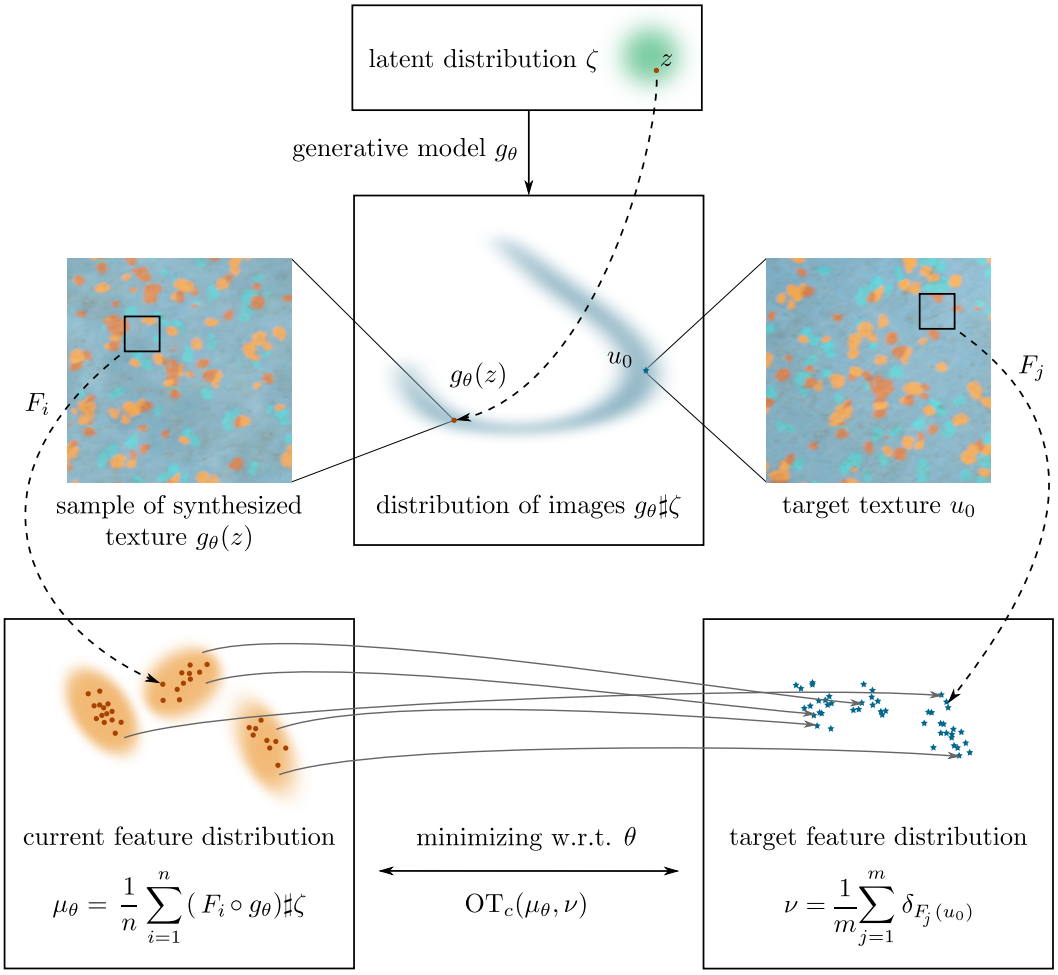


FIGURE 3.4 – Diagramme explicatif de l'approche GOTEX. En sortie du réseau génératif g_θ, on a une distribution de *features* μ_θ. L'algorithme GOTEX vise à minimiser $W(\mu_\theta, \nu)$ entre la distribution de *features* courante μ_θ et la distribution de *features* ν de la texture exemple u_0.

Comme dans la section précédente, nous allons formuler le problème au travers de l'apprentissage d'un réseau génératif. On se fixe donc une fonction mesurable $g : \Theta \times \mathcal{Z} \longrightarrow \mathbb{R}^{dn}$ où $\Theta \subset \mathbb{R}^q$ est ouvert. On obtiendra des textures synthétisées en prenant des échantillons de $g(\theta, Z)$ où Z est une v.a. de loi fixée ζ sur un ensemble mesurable $\mathcal{Z}$. Notons qu'en prenant $g(\theta, z) = \theta$ où $\theta = v \in \mathbb{R}^{dn}$ est une image, cette formulation permet d'optimiser directement l'image à synthétiser.

On rassemble les distributions de *features* en chaque position i en considérant la distribution

$$\mu_\theta = \frac{1}{n} \sum_{i=1}^{n} (f_i \circ g_\theta) \sharp \zeta. \tag{3.19}$$

Dans le cas $g(\theta, z) = \theta$, on obtient simplement la distribution empirique $\mu_\theta = \frac{1}{n} \sum_{i=1}^{n} \delta_{f_i(\theta)}$. Par ailleurs, en extrayant les *features* de la texture exemple u_0, on obtient la distribution cible $\nu = \frac{1}{m} \sum_{j=1}^{m} \delta_{f_j(u_0)}$.

La fonction de coût utilisée pour l'apprentissage du modèle génératif de textures s'écrit

$$\mathcal{L}_{\text{GOTEX}}(\theta, \mathbf{v}) = W(\mu_\theta, \nu) \tag{3.20}$$

où $c : \mathbb{R}^D \times \mathbb{R}^D \longrightarrow \mathbb{R}$ est un coût dans l'espace des *features*.

Comme ν est une distribution discrète, la formulation semi-duale du transport optimal donne

$$W(\mu_\theta, \nu) = \max_{\psi \in \mathbb{R}^m} H(\psi, \theta) := \int_{\mathcal{X}} \psi^c(x) d\mu_\theta(x) + \frac{1}{m} \sum_{j=1}^{m} \psi_j, \tag{3.21}$$

où le calcul de la c-transform $\psi^c(x) = \min_j [c(x, f_j(\mathbf{v})) - \psi_j]$ revient à une recherche de plus proche voisin (biaisée) dans l'espace des *features*. À θ fixé, $H(\theta, \cdot)$ est une fonction concave, que l'on peut maximiser à l'aide d'une montée de sur-gradients, toujours avec l'algorithme ASGD [57]. D'autre part, les résultats de la section précédente permettent d'écrire le gradient par rapport à θ.

Théorème 3.2.1. *Supposons que c est de classe $\mathscr{C}^1$ et Lipschitz, et supposons que pour chaque i, $f_i : \mathbb{R}^{dn} \longrightarrow \mathbb{R}^D$ est $\mathscr{C}^1$ et Lipschitz. Supposons que g satisfait (G_Θ). Soit (ψ, θ_0) tel que ζ-p.s., pour tout i, $f_i \circ g_{\theta_0}(Z) \notin A_\psi$.*

Alors $H(\psi, \cdot)$ est différentiable en θ_0 avec

$$\nabla_\theta H(\psi, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{Z \sim \zeta} \left[D_\theta (f_i \circ g)(\theta_0, Z)^T \nabla \psi^c (f_i \circ g(\theta_0, Z)) \right]. \tag{3.22}$$

3.2.2 Combiner Plusieurs Types de *Features*

La construction du paragraphe précédent permet de contraindre la distribution d'un seul type de *features* moyennées spatialement. Mais pour synthétiser des textures fidèlement, on doit contraindre les distributions de plusieurs types de *features*, comme par exemple les distributions de patches extraits à différentes résolutions (comme dans le Chapitre 2.2).

Input : texture originale u_0, paramètre initial θ_0, nombres d'itérations N_θ, N_ψ, pas $\eta_\psi > 0$, optimiseur Optim (Adam ou L-BFGS) de pas η_θ,
Output: paramètre appris θ^*
Initialisation : $\theta \leftarrow \theta_0$ et $\psi^{l,0} = 0$ pour $l = 1 \dots N_F$
for $k = 1$ ***to*** N_θ **do**
  for $l = 1$ ***to*** N_F **do**
    $\psi^{l,k} \approx \text{Arg max}_{\psi^l} H^l(\theta^k, \psi^l)$ (N_ψ itérations d'ASGD initialisé en $\psi^{l,k-1}$)
  end
  $\theta^k \leftarrow \text{Optim}(\sum_{l=1}^{N_F} H^l(\theta^{k-1}, \psi^{l,k}), \eta_\theta)$ (un pas d'optimisation en θ)
end
Algorithm 5: Apprentissage GOTEX d'un modèle génératif de texture

Pour cela, on considère une collection $(f^l)_{1 \leqslant l \leqslant N_F}$ de *features*, chacune prise en n_l positions spatiales grâce aux fonctions

$$f_i^l : \mathbb{R}^{dn} \longrightarrow \mathbb{R}^{D_l}, \quad 1 \leqslant i \leqslant n_l. \tag{3.23}$$

On introduit une fonction de coût agrégeant les coûts de transport calculés sur ces différentes *features* :

$$\mathcal{L}_{\text{GOTEX}}(\theta) = \sum_{l=1}^{N_F} W(\mu_\theta^l, \nu^l) \tag{3.24}$$

où $\mu_\theta^l = \frac{1}{n_l} \sum_{i=1}^{n_l} (f_i^l \circ g_\theta) \sharp \zeta$ (resp. $\nu_\theta^l = \frac{1}{m_l} \sum_{i=1}^{m_l} \delta_{f_i^l(u_0)}$) est la distribution de la *feature* l du modèle génératif (resp. de l'image originale). En utilisant de nouveau la formulation semi-duale du transport optimal, on peut écrire

$$\mathcal{L}_{\text{GOTEX}}(\theta) = \sum_{l=1}^{N_F} \max_{\psi^l \in \mathbb{R}^{m_l}} H^l(\theta, \psi^l) \tag{3.25}$$

$$\text{où} \quad H^l(\theta, \psi^l) = \mathbb{E}_{Z \sim \zeta} \left[\frac{1}{n_l} \sum_{i=1}^{n_l} (\psi^l)^c (f_i^l \circ g_\theta(Z)) + \frac{1}{m_l} \sum_{j=1}^{m_l} \psi_j^l \right]. \tag{3.26}$$

Pour minimiser cette fonction de coût, on utilise un schéma alterné résumé dans l'Algorithme 5. Pour la mise à jour de θ, on peut utiliser différents algorithmes. On utilisera L-BFGS [99] pour l'optimisation directe de l'image synthétisée, et ADAM [79] pour l'optimisation d'un réseau génératif.

Dans les expériences ci-dessous, on va appliquer cette méthodologie avec quatre collections de *features*, notées *patch*, *VGG*, *mix* et *all*. Pour la première (*patch*), comme dans le Chapitre 2, on extrait les distributions de patches à plusieurs résolutions (obtenues par simple sous-échantillonnage) : ainsi, la *feature* f_i^l extrait le patch en position i à la résolution l. Pour la deuxième (*VGG*), on extrait les distributions de réponses neuronales à plusieurs couches du réseau convolutionnel `VGG19` [138] : sur $N_F = 5$ couches du réseau (prises dans les premières couches), f_i^l extrait la réponse en position i à la couche l. Pour la troisième (*mix*), on considère les *features* (*VGG*) prises sur des couches profondes auxquelles on adjoint la distribution de patches aux résolutions fines. Enfin, pour la dernière (*all*), on rassemble les *features* (*patch*) et (*VGG*).

Lien avec les projections NN itérées. Avant de présenter les synthèses, on va voir que si l'on utilise les *features patch* avec le coût quadratique, alors l'Algorithme 5, alors la descente de gradient pour optimiser directement la synthèse v peut s'interpréter à l'aide de projections NN par patch itérées. Par exemple, si les (P_i) extraient les patches à une résolution, avec le coût quadratique, le Théorème 3.2.1 assure que

$$\nabla_\theta H(v^{k-1}, \psi^k) = \frac{1}{n}\left(\sum_{i=1}^n P_i^T P_i v^{k-1} - \sum_{i=1}^n P_i^T P_{\sigma^k(i)} u_0\right), \tag{3.27}$$

qui existe en tout point (v^{k-1}, ψ^k) où l'on peut définir de façon univoque l'assignement

$$\sigma^k(i) = \operatorname*{Argmin}_j \frac{1}{2}\|P_i v^k - P_j u_0\|^2 - \psi_j^k \quad \forall i = 1, \dots, n. \tag{3.28}$$

En utilisant un pas de gradient $\eta\frac{n}{s^2}$, la mise à jour de v par descente de gradient s'écrit

$$v^k = (1-\eta)v^{k-1} + \eta\tilde{v}^k, \tag{3.29}$$

où $\tilde{v}^k$ est une image obtenue par agrégation des patches de u_0 assignés par (3.28). L'algorithme de synthèse GOTEX peut donc être vu comme une amélioration de [85] respectant mieux les distributions de patches de la texture originale.

3.2.3 Expériences

Commentons maintenant les synthèses obtenues avec l'algorithme GOTEX utilisé soit pour optimiser directement l'image synthétisée (Fig. 3.5) soit pour optimiser un réseau génératif (Fig. 3.6). Dans la Fig. 3.5, on peut comparer les synthèses avec différents types de *features*. On constate ainsi que l'utilisation des *features patch* permet de reproduire les motifs locaux de l'original, avec néanmoins une légère perte de netteté. À l'inverse, avec les *features VGG*, l'algorithme produit des images plus nettes, mais présentant parfois des artefacts de couleur (qui avaient déjà été constatés avec l'algorithme de [51]). Ces artefacts peuvent être contournés en considérant de façon conjointe les *features patch* et *VGG*.

Sur la Fig. 3.6, on voit que l'algorithme GOTEX est capable d'estimer un réseau génératif produisant des synthèses de qualité comparable à celles obtenues avec le modèle TextoML du Chapitre 2. L'avantage de cette technique est qu'une fois le réseau génératif appris sous forme d'un réseau convolutionnel, il peut être utilisé pour synthétiser plusieurs grandes textures de façon particulièrement rapide. On voit que les synthèses GOTEX incluant les *features VGG* sont plus nettes que celles de TextoML. Cependant, on a pu constater que l'apprentissage d'un tel réseau génératif produit plus de cas d'échec qu'avec TextoML, et est aussi très sensible aux paramètres utilisés pour l'optimisation. Les résultats visuels restent néanmoins comparables à ceux d'autres méthodes apprenant un réseau génératif pour synthétiser une texture [151]. Les synthèses obtenues avec la méthode de [13] basés sur des réseaux génératifs adverses (sans coût de transport optimal) sont généralement de qualité visuelle bien inférieure. Enfin, la méthode SinGAN de [135] (qui ne se limite

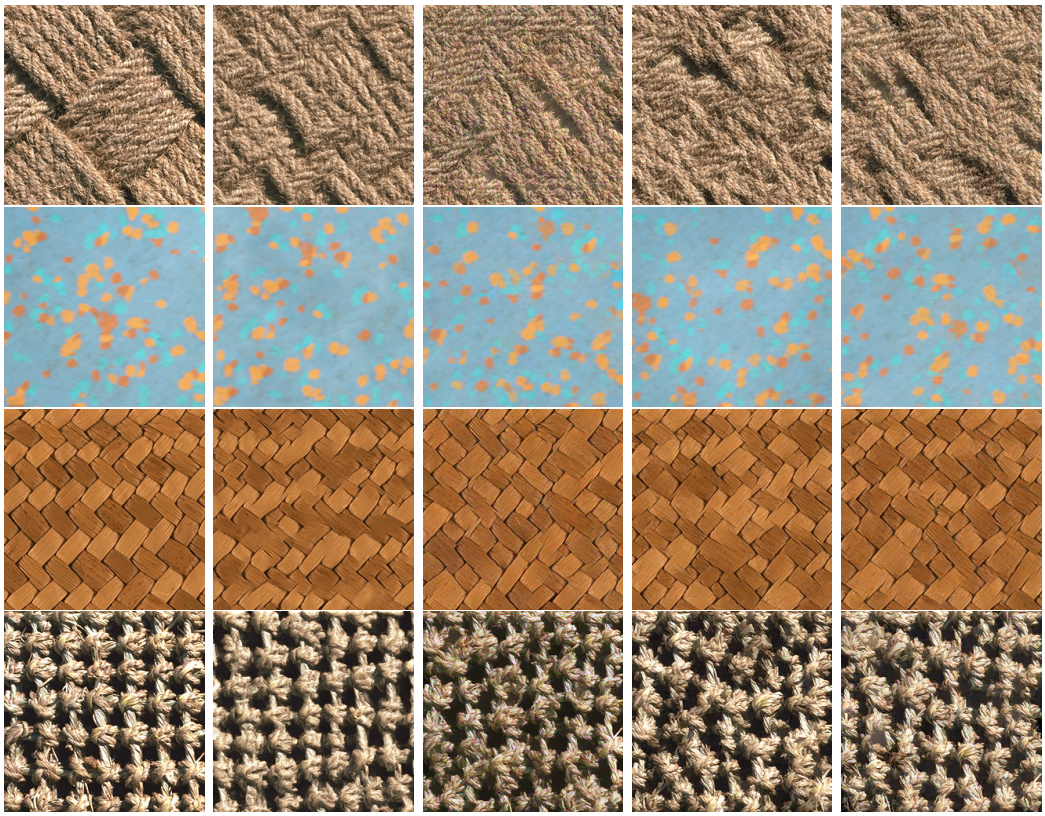


FIGURE 3.5 – Synthèse GOTEX par l'Algorithme 5 pour l'optimisation directe de l'image synthétisée, avec plusieurs collections de *features* : *patch*, *VGG*, *mix*, *all*.

pas à la synthèse de textures) produit des images d'une qualité similaire à GOTEX, mais présentant parfois des dérives statistiques et des artefacts de recopie. En résumé, on constate ici nettement l'intérêt à concilier les distributions de patches à basse résolution et les distributions de *features VGG*, pour préserver respectivement la bonne organisation spatiale et le grain de la texture.

3.3 Garanties Statistiques pour les Estimateurs Wasserstein

Ce paragraphe est consacré à des travaux menés en collaboration avec Paul Freulon, doctorant de l'IMB dirigé par Jérémie Bigot et Boris Hejblum. Ces travaux, publiés dans [Js2], concernent l'estimation d'un modèle paramétrique μ_θ approchant une mesure cible ν, en supposant que ces mesures ne sont accessibles qu'au travers d'échantillons.

Cet estimateur sera construit en minimisant un coût Wasserstein comme dans (3.7), et l'on cherchera à contrôler le risque empirique associé. Un objectif de ce travail était d'étudier l'impact du paramètre de régularisation en termes d'erreur d'estimation statistique.

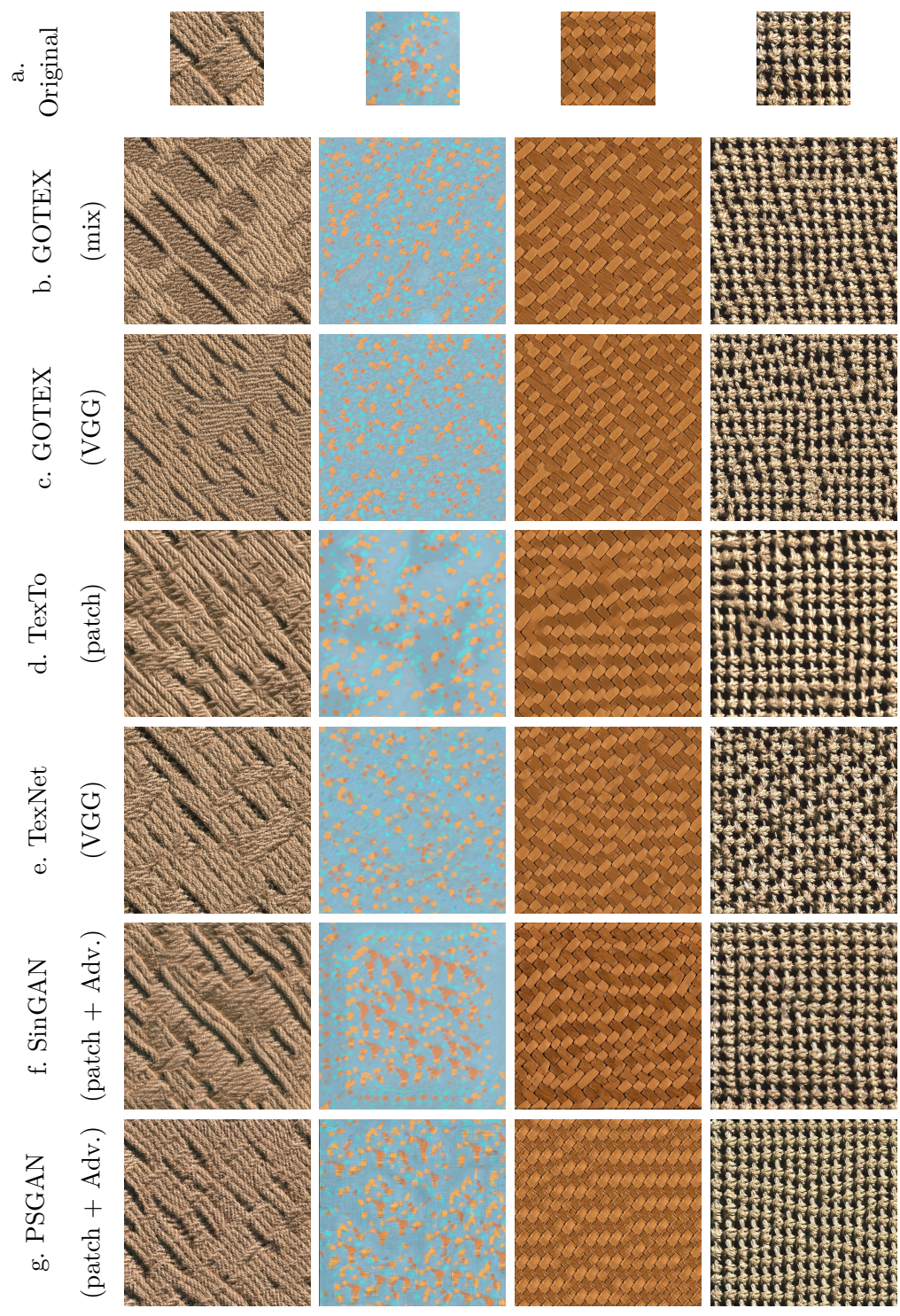


FIGURE 3.6 – Synthèse de textures avec différents modèles génératifs appris sur l'image originale (a). On compare les méthodes GOTEX-mix (b), GOTEX-VGG (c), TextoML (d), TexNet [150] (e), SinGAN [135] (f) et PSGAN [13] (g).

Soient $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^d$ des compacts contenus dans la boule ouverte $B(0,R)$ pour la norme ℓ^2. Soit $\Theta \subset \mathbb{R}^q$ et soit $(\mu_\theta)_{\theta\in\Theta}$ une collection de mesures de probabilité à support inclus dans $\mathcal{X}$. Soit ν une mesure de probabilité à support inclus dans $\mathcal{Y}$. Dans ce paragraphe, on se restreindra au coût quadratique $c(x,y) = \|x-y\|_2^2$ sur $\mathbb{R}^d$.

On considère une loi empirique

$$\hat{\mu}_\theta = \frac{1}{n}\sum_{i=1}^{n} \delta_{X_i} \tag{3.30}$$

basée sur n échantillons $X_1, \dots, X_n$ i.i.d. de loi μ_θ, et de même $\hat{\nu}$ basée sur n échantillons $Y_1, \dots, Y_n$ i.i.d. de loi ν.

Pour $\lambda \geqslant 0$, on suppose disposer d'un estimateur optimal pour le coût Wasserstein empirique régularisé

$$\hat{\theta}_\lambda \in \widehat{\Theta}_\lambda := \operatorname*{Argmin}_{\theta\in\Theta} W_\lambda(\hat{\mu}_\theta, \hat{\nu}). \tag{3.31}$$

On suppose aussi disposer d'une solution du problème non-régularisé :

$$\theta^* \in \operatorname*{Argmin}_{\theta\in\Theta} W_0(\mu_\theta, \nu). \tag{3.32}$$

L'existence de $\hat{\theta}_\lambda$ et θ^* est assurée dès que l'on adopte une hypothèse de compacité de Θ et de continuité de $\theta \mapsto \mu_\theta$.

On va chercher à contrôler le risque associé à l'estimateur $\hat{\theta}_\lambda$, défini par

$$R(\hat{\theta}_\lambda) = \mathbb{E}[W_0(\mu_{\hat{\theta}_\lambda}, \nu) - W_0(\mu_{\theta^*}, \nu)] \geqslant 0. \tag{3.33}$$

Notons que ce risque s'exprime avec le coût Wasserstein non-régularisé (alors que la définition de $\hat{\theta}_\lambda$ repose sur W_λ).

Pour séparer l'impact de la régularisation entropique et de l'échantillonnage, on utilise une décomposition de l'erreur donnée par le lemme suivant.

Lemma 3.3.1. *Pour $\lambda \geqslant 0$, le risque de l'estimateur $\hat{\theta}_\lambda$ est majoré par*

$$W_0(\mu_{\hat{\theta}_\lambda}, \nu) - W_0(\mu_{\theta^*}, \nu) \leqslant \underbrace{2\sup_{\theta\in\Theta} |W_\lambda(\hat{\mu}_\theta, \hat{\nu}) - W_\lambda(\mu_\theta, \nu)|}_{\textit{Erreur d'estimation}} + \underbrace{2B(\lambda)}_{\textit{Erreur d'approximation}}, \tag{3.34}$$

où l'on a posé

$$B(\lambda) = \sup_{\theta\in\Theta} |W_0(\mu_\theta, \nu) - W_\lambda(\mu_\theta, \nu)|. \tag{3.35}$$

D'une part, l'erreur d'approximation peut être majorée pour des distributions à support borné, grâce à la proposition suivante.

Proposition 3.3.1 (Genevay *et al.* [56])**.** *Supposons que $\mathcal{X}, \mathcal{Y} \subset B(0,R)$. Alors*

$$\forall \lambda > 0, \quad B(\lambda) \leqslant 2d\lambda \log\left(\frac{8\exp(2)R^2}{\sqrt{d}\lambda}\right) = \mathcal{O}_{\lambda\longrightarrow 0}(\lambda|\log\lambda|). \tag{3.36}$$

D'autre part, l'erreur d'estimation avec W_λ est contrôlée par la proposition suivante.

Proposition 3.3.2. *Supposons que $\mathcal{X}, \mathcal{Y} \subset B(0,R)$. Alors*

$$\mathbb{E}\left[\sup_{\theta\in\Theta} |W_\lambda(\hat{\mu}_\theta, \hat{\nu}) - W_\lambda(\mu_\theta, \nu)|\right] \lesssim \frac{M_\lambda}{\sqrt{n}} \quad \text{où} \quad M_\lambda = C_d \max\left(R^2, \frac{R^{\lfloor \frac{d}{2} \rfloor + 1}}{\lambda^{\lfloor \frac{d}{2} \rfloor}}\right), \tag{3.37}$$

où $\lesssim$ signifie $\leqslant$ à une constante absolue près, et où C_d ne dépend que de d.

La preuve de la proposition précédente repose sur des majorations de processus empiriques du type

$$\sup_{\psi\in\mathcal{F}} \left| \int \psi d(\nu - \hat{\nu}) \right|, \tag{3.38}$$

où $\mathcal{F}$ est une classe de fonctions dans laquelle on peut restreindre le problème semi-dual de transport optimal. En particulier, pour obtenir (3.37), on utilise que pour $\lambda > 0$ les c-transforms régularisées sont (pour le coût quadratique) de classe $\mathscr{C}^\infty$ et que l'on peut majorer uniformément leurs dérivées sur $B(0,R)$ par la constante M_λ.

En utilisant la décomposition du risque, on doit ainsi faire un compromis entre deux termes, l'un en $\mathcal{O}(\lambda \log \lambda)$ et l'autre en $\mathcal{O}(\frac{1}{\sqrt{n}\lambda^{\lfloor \frac{d}{2} \rfloor}})$. Ceci conduit à un choix de paramètre λ_n tendant vers zéro, et à une vitesse de convergence du risque exprimée dans le théorème suivant. Pour alléger l'écriture, on ne donnera les énoncés suivants que pour $d > 4$.

Théorème 3.3.3. *Supposons que $\mathcal{X}, \mathcal{Y} \subset B(0,R)$ et $d > 4$, et notons $d' = 2\lfloor \frac{d}{2} \rfloor$. On rappelle que pour $\lambda \geqslant 0$, $\hat{\theta}_\lambda$ est l'estimateur défini par (3.31) et basé sur n échantillons de μ_θ et de ν. En prenant λ_n proportionnel à $n^{-\frac{1}{d'+2}}$, on a*

$$\mathbb{E}[W_0(\mu_{\hat{\theta}_{\lambda_n}}, \nu) - W_0(\mu_{\theta*}, \nu)] \lesssim n^{-\frac{1}{d'+2}} \log(n). \tag{3.39}$$

Avec ce procédé de régularisation adaptative, on obtient une vitesse de convergence moins bonne que la vitesse minimax qui est en $\mathcal{O}(n^{-\frac{2}{d}})$. Dans [Js2], on a aussi étudié un procédé d'estimation similaire basé sur la divergence de Sinkhorn au lieu de W_λ, que l'on ne présentera pas ici car il nécessite des hypothèses techniques sur les mesures en jeu. Mentionnons néanmoins que cette dernière technique permet d'améliorer la vitesse en $\mathcal{O}(n^{-\frac{2}{d'+4}})$, ce qui est toujours sous-optimal, comme cela a été remarqué dans [31]. En utilisant d'autres processus empiriques, on peut montrer qu'on atteint de meilleures vitesses de convergence :

Théorème 3.3.4. *Supposons que $\mathcal{X}, \mathcal{Y} \subset B(0,R)$ et supposons que $d > 4$. On rappelle que pour $\lambda \geqslant 0$, $\hat{\theta}_\lambda$ est l'estimateur défini par (3.31) et basé sur n échantillons de μ_θ et ν.*

1. *Pour l'estimateur non-régularisé ($\lambda = 0$), on a*

$$\mathbb{E}[W_0(\mu_{\hat{\theta}_0}, \nu) - W_0(\mu_{\theta*}, \nu)] \lesssim n^{-\frac{2}{d}}. \tag{3.40}$$

2. *En prenant λ_n proportionnel à $n^{-\frac{2}{d}}$, on a*

$$\mathbb{E}[W_0(\mu_{\hat{\theta}_{\lambda_n}}, \nu) - W_0(\mu_{\theta*}, \nu)] \lesssim n^{-\frac{2}{d}} \log(n). \tag{3.41}$$

La preuve du résultat précédent se base sur une formulation alternative de coût de transport régularisé W_λ : en développant $\|x-y\|^2$, on fait un lien avec le transport optimal pour le coût $s(x,y) = -2\langle x, y\rangle$. L'intérêt est qu'avec ce nouveau coût, on peut montrer que les s-transforms (régularisées ou non) sont des fonctions convexes, ce qui permet d'améliorer le contrôle du processus empirique. En effet, Chizat *et al.* [31] ont montré que si $\mathcal{F}_R$ désigne l'ensemble des fonctions convexes et R-Lipschitz sur la boule $B(0,R)$, alors pour $d > 4$,

$$\mathbb{E}\left[\sup_{\varphi\in\mathcal{F}_R}\left|\int \varphi d(\nu - \hat{\nu})\right|\right] \lesssim R^2 n^{-2/d} \tag{3.42}$$

Dans [31], ce contrôle est utilisé pour majorer la vitesse obtenue avec l'estimation $\hat{\theta}_0$, mais nous avons montré que la majoration s'étend au cas de l'estimateur régularisé $\hat{\theta}_\lambda$, au prix d'un terme logarithmique dans (3.41).

Là encore, dans le Théorème 3.3.4, on constate que la vitesse obtenue avec la régularisation entropique est légèrement moins bonne que celle obtenue sans régularisation. Cependant, le calcul du coût de transport régularisé W_λ entre deux distributions discrètes de taille n peut se faire efficacement grâce à l'algorithme de Sinkhorn. En pratique, on a donc intérêt à régulariser légèrement pour permettre l'utilisation d'un algorithme efficace de calcul de W_λ. En contrôlant la vitesse de convergence de l'algorithme de Sinkhorn, on peut expliciter le compromis à faire en fonction du nombre d'échantillons et du nombre d'itérations de l'algorithme, ce qui a été exprimé dans [31] et que nous avons également détaillé dans [Js2] pour le cas d'estimation de modèles de mélange.

3.4 Conclusion Partielle

Dans ce chapitre, nous avons suivi plusieurs axes de recherche portant sur l'estimation de modèles génératifs. Les résultats de la Section 3.1.3 permettent d'éclaircir le comportement des algorithmes d'optimisation alternée souvent utilisés pour l'apprentissage de réseaux génératifs, et notamment de mieux comprendre, dans un cadre semi-discret, l'impact de l'erreur d'estimation de la variable duale du transport optimal. Dans la Section 3.2, on a adapté la méthodologie d'estimation de réseaux génératifs Wasserstein pour le problème de la synthèse de textures, en contraignant les distributions de patches. Enfin, la Section 3.3 fournit un contrôle sur le risque empirique des estimateurs Wasserstein construits à partir de versions échantillonnées des mesures en jeu.

L'un des enjeux de cette ligne de travaux était d'analyser, du point de vue théorique et pratique, le rôle de la régularisation entropique du transport optimal. Sur le plan théorique, les bornes d'estimation de la Section 3.3 montrent que l'utilisation de la régularisation entropique mène à une vitesse de convergence du risque empirique légèrement sous-optimale. Sur le plan pratique (avec des expériences sur la génération de chiffres MNIST et la synthèse de textures), dans le cadre semi-discret où l'algorithme de Sinkhorn n'est pas directement accessible, on observe que la régularisation entropique n'accélère pas nettement l'algorithme alterné, alors

qu'elle peut dégrader sensiblement la qualité visuelle des échantillons du modèle génératif. Néanmoins, en travaillant dans un cadre échantillonné à la source et à la cible, l'utilisation de l'algorithme de Sinkhorn peut parfois aider à améliorer la vitesse d'estimation observée en pratique, pour peu que l'on ajuste finement les paramètres (paramètre de régularisation λ en fonction du nombre d'échantillons n, nombre d'itérations de l'algorithme de Sinkhorn).

Pour ce qui est de la synthèse de textures, l'intérêt de l'approche GOTEX est qu'elle permet de contraindre directement les distributions de patches multi-résolution en sortie du modèle génératif. L'agrégation des patches n'apparaît plus explicitement comme une étape de synthèse comme c'était le cas pour le modèle Texto, mais est intégrée implicitement dans l'algorithme itératif de synthèse (voir la fin de la Section 3.2.2).

Les approches Texto et GOTEX illustrent deux façons distinctes d'utiliser le transport optimal pour l'apprentissage de modèles génératifs. Dans Texto, le plan de transport optimal semi-discret est utilisé directement pour générer les échantillons (en transportant les patches), alors que dans GOTEX, il sert à exprimer le coût pour optimiser un réseau génératif à architecture fixée. À la lumière des résultats obtenus avec ces deux approches pour la synthèse de textures, on peut légitimement questionner l'applicabilité de la deuxième, qui nécessite une expertise fine pour la conception de l'architecture du réseau permettant de générer une texture. En fait, on peut réinterpréter le procédé de synthèse de Texto comme un réseau génératif (avec des couches de projections au plus proche voisin biaisées) astucieusement paramétré de façon à permettre une synthèse fidèle à plusieurs résolutions. En définitive, le choix de l'approche adoptée peut être guidé par les contraintes applicatives (par exemple, besoin de synthèse parallèle à la demande, d'empreinte mémoire contrôlée).

Dans le chapitre suivant, nous introduirons un nouveau modèle pour la synthèse de textures fournissant des garanties sur les statistiques d'ordre 1 calculées sur un ensemble de *features* prédéfinies.

Chapitre 4

Modèles de Maximum d'Entropie pour la Synthèse d'Images

Ce chapitre porte sur l'étude de modèles de maximum d'entropie pour la synthèse d'images de textures [J11]. Ces travaux ont été menés dans le cadre de la thèse de Valentin de Bortoli, en collaboration avec Bruno Galerne, Agnès Desolneux et Alain Durmus. Ils portent sur la formulation de la synthèse de textures comme l'échantillonnage d'un modèle aléatoire de maximum d'entropie, contraint par des réponses moyennes à un banc de filtres (possiblement non-linéaires). D'abord, on a cherché à étudier l'existence et l'unicité de ces distributions de maximum d'entropie, et le cas échéant, à en expliciter la forme (Section 4.2). On a ensuite proposé un algorithme d'échantillonnage et d'estimation de ces modèles (Section 4.3) basé sur l'algorithme de Langevin, et nous avons obtenu bornes explicites sur la convergence de cet algorithme d'échantillonnage. Enfin, nous en avons déduit un algorithme de synthèse de textures en contraignant le modèle par des réponses moyennes à des couches d'un réseau de neurones pré-appris (Section 4.4).

4.1 Synthèse Paramétrique et Maximum d'entropie

Les méthodes dites paramétriques de synthèse de textures consistent, à partir d'une collection finie de statistiques bien choisies, à formuler un algorithme de synthèse produisant une image *la plus aléatoire possible* (dans un sens à préciser) sur laquelle les statistiques choisies sont les mêmes que sur la texture exemple. Plus précisément, fixons $\tilde{F} : \mathbb{R}^d \longrightarrow \mathbb{R}^p$ une fonction définie sur l'ensemble $\mathbb{R}^d$ des images, représentant la collection des p statistiques à respecter. En notant $x_0 \in \mathbb{R}^d$ la texture exemple, et $F(x) = \tilde{F}(x) - \tilde{F}(x_0)$, on va chercher à proposer un modèle de textures sous la forme d'une loi de probabilité π sur $\mathbb{R}^d$ telle que si $X \sim \pi$, alors $F(X)$ est proche de 0, ce qui peut signifier $F(X) = 0$ p.s., ou bien, de façon moins contraignante, $\mathbb{E}[F(X)] = 0$.

L'exemple le plus simple de modèle paramétrique est le modèle gaussien [Th], qui permet de préserver (en espérance) la valeur moyenne et l'autocorrélation de la texture. Le modèle gaussien reproduit bien le contenu fréquentiel de l'image, qui est encodé dans la densité spectrale, c'est-à-dire dans la transformée de Fourier de la covariance spatiale. Par ailleurs, on peut voir que le modèle gaussien est d'entropie maximale parmi les lois ayant les mêmes moments d'ordre 1 et 2. En fait, on peut montrer que ce modèle gaussien a des phases de Fourier i.i.d. uniformes, ce qui le

limite à des textures très peu structurées que l'on appelle parfois microtextures. Les propriétés du modèle gaussien le rendent cependant très utile pour la synthèse rapide [J3] et l'inpainting [J5] de microtextures, ou encore pour l'analyse de netteté par cohérence de phase [J2]. Le modèle gaussien apparaît aussi comme initialisation du modèle Texto que nous avons étudié au Chapitre 2.

Le modèle paramétrique de Portilla et Simoncelli [118] permet de synthétiser des textures plus structurées avec un ensemble de statistiques calculées sur les réponses de la texture à une transformée en ondelettes complexes. En particulier, les statistiques incluent des corrélations de coefficients d'ondelettes calculés à des échelles adjacentes (ce qui contraint la cohérence des phases locales). L'algorithme de synthèse proposé consiste, à partir d'un bruit blanc, à opérer des projections alternées sur les contraintes statistiques considérées. Cet algorithme n'a pas de raison d'être convergent, mais on observe qu'il se stabilise sur des images réalisant un bon compromis entre les contraintes statistiques désirées. La percée de [118] a consisté à trouver un ensemble de 710 statistiques génériques permettant de reproduire une large classe de textures structurées.

Plus récemment, Gatys *et al.* [51] ont proposé un nouveau modèle de textures paramétrique prenant en compte des statistiques d'ordre 2 calculées sur des réponses au réseau de neurones `VGG19` [138]. L'algorithme de synthèse proposé consiste à minimiser la fonctionnelle $x \mapsto F(x)^2$ (en initialisant avec un bruit blanc) de façon à contraindre, au mieux, toutes les statistiques en même temps (et non plus d'alterner entre la projection sur les différentes contraintes statistiques). En comparaison de [118], ce modèle neuronal permet d'étendre largement la classe de textures bien reproduites par l'algorithme, au prix d'une optimisation lourde basée sur environ 300 000 statistiques, qui sont plus difficilement interprétables.

Dans les trois modèles évoqués, on constate une distinction selon que l'on veut imposer $F(X) = 0$ vaut zéro presque sûrement, ou bien seulement en espérance. Ceci est lié à une distinction entre des modèles dits microcanonique et macrocanonique [20], respectivement. En plus de la contrainte sur F, ces modèles répondent à un principe de maximum d'entropie qui est une manière d'introduire de l'aléa tout en respectant les contraintes données par F. Par exemple, si $F(x)$ extrait le module au carré de la transformée de Fourier de x, le modèle macrocanonique associé est le modèle gaussien déjà évoqué, et le modèle microcanonique associé est le modèle *random phase noise* [50] où l'on fixe le module de Fourier et où l'on tire la phase uniformément. En dehors de cet exemple simple, il n'y a pas d'algorithme générique pour l'échantillonnage du modèle microcanonique, et la méthode de synthèse correspondante repose sur la minimisation d'une fonctionnelle avec initialisation aléatoire, ce qui rejoint la synthèse de textures variationnelle évoquée dans le chapitre précédent.

En revanche, on va voir que l'étude du modèle macrocanonique se ramène à celle des modèles exponentiels, bien connus des statisticiens. Pour la modélisation de textures, ce lien avait déjà été établi, dans un cadre d'échantillonnage discret, par Zhu *et al.* [167], qui ont introduit le modèle FRAME, pour *Filters, random fields, and maximum entropy*. Ce modèle FRAME est un modèle de maximum d'entropie permettant de respecter des statistiques en sortie d'un banc de filtres non-linéaires.

Une instance du modèle associée à une texture exemple fixée est construite en sélectionnant quelques filtres non-linéaires parmi une collection de filtres pré-conçus. L'échantillonnage du modèle repose sur l'échantillonneur de Gibbs appliqué dans un ensemble fini d'images (obtenu en discrétisant les niveaux de gris). Plus tard, Lu *et al.* [102] ont conçu l'extension deepFRAME, permettant d'inclure des réponses à un réseau de neurones profond. L'algorithme de synthèse correspondant repose sur l'échantillonneur de Langevin, qui ne nécessite pas de discrétiser l'ensemble des images. En contraignant seulement des statistiques d'ordre 1 calculées sur un banc de filtres non-linéaires, les modèles FRAME et deepFRAME rejoignent les dernières conjectures de Julesz [77] sur la perception texturale pré-attentive.

Les travaux présentés dans ce chapitre visent à préciser l'étude mathématique des modèles de maximum d'entropie, et notamment ceux obtenus en contraignant des réponses moyennes à des couches de réseaux de neurones.

4.2 Distributions de Maximum d'Entropie

Étant donné une fonction $F : \mathbb{R}^d \longrightarrow \mathbb{R}^p$ représentant les statistiques à respecter, nous allons maintenant étudier l'existence et l'unicité de la distribution π de maximum d'entropie telle que $\int F d\pi = 0$. Pour mesurer l'entropie, nous introduisons une loi de probabilité de référence μ sur $\mathbb{R}^d$. L'entropie d'une loi de probabilité π sur $\mathbb{R}^d$ relativement à μ est mesurée grâce à la divergence de Kullback-Leibler définie par

$$\mathrm{KL}\left(\pi|\mu\right) = \begin{cases} \int_{\mathbb{R}^d} \frac{\mathrm{d}\pi}{\mathrm{d}\mu}(x) \log\left(\frac{\mathrm{d}\pi}{\mathrm{d}\mu}(x)\right) \mathrm{d}\mu(x) & \text{si } \pi \ll \mu\ , \\ +\infty & \text{sinon}\ . \end{cases} \tag{4.1}$$

On peut voir avec l'inégalité de Jensen que $\mathrm{KL}\left(\pi|\mu\right) \in [0, +\infty]$ dès que π, μ sont des lois de probabilité. Si π, μ sont supportées dans un ensemble fini S, avec μ uniforme, alors $-\mathrm{KL}\left(\pi|\mu\right)$ est à une constante près l'entropie de Shannon de la loi discrète π. On va chercher à maximiser l'entropie relative, avec des contraintes presque sûre ou en espérance, ce qui mène aux deux définitions suivantes.

Définition 4.2.1. Une loi de probabilité π sur $\mathbb{R}^d$ est appelée *modèle microcanonique* pour F par rapport à μ, si $\pi(\{F = 0\}) = 1$ et si $\mathrm{KL}\left(\pi|\mu\right) \leqslant \mathrm{KL}\left(\nu|\mu\right)$ pour toute autre loi de probabilité ν sur $\mathbb{R}^d$ telle que $\nu(\{F = 0\}) = 1$.

Les algorithmes de synthèse proposés dans [118, 51, 20] visent à échantillonner approximativement un modèle microcanonique. Pour obtenir des garanties théoriques, il est utile de considérer un autre modèle pour lequel on relâche la contrainte statistique.

Définition 4.2.2. Une loi de probabilité π sur $\mathbb{R}^d$ est appelée *modèle macrocanonique* pour F par rapport à μ, si $\int F d\pi = 0$ et si $\mathrm{KL}\left(\pi|\mu\right) \leqslant \mathrm{KL}\left(\nu|\mu\right)$ pour toute autre loi de probabilité ν sur $\mathbb{R}^d$ telle que $\int F d\nu = 0$.

Ce modèle macrocanonique apparaît dans [75] et a été utilisé en synthèse de textures dans [167, 102]. Les auteurs de [20, 159] ont donné des conditions pour que

les modèles microcanonique et macrocanonique tendent vers un même objet limite lorsque la taille du domaine tend vers l'infini. Dans la suite du paragraphe, nous allons donner des conditions d'existence et d'unicité du modèle macrocanonique. Pour cela, nous allons introduire deux problèmes d'optimisation en dualité. Introduisons les contraintes de moments : pour $\alpha \geqslant 0$ fixé, on note $\mathcal{P}_\alpha$ l'ensemble des mesures de probabilité π sur $\mathbb{R}^d$ telles que $\int_{\mathbb{R}^d} ||x||^\alpha \mathrm{d}\pi(x) < +\infty$. On utilisera aussi la notation $\pi(F) = \int F d\pi$.

Le problème de maximum d'entropie correspond au problème primal :

$$v(\mathrm{P}) := \inf_{\pi \in \mathcal{P}_\alpha^F} \mathrm{KL}\left(\pi|\mu\right) \quad \text{où} \quad \mathcal{P}_\alpha^F = \{\pi \in \mathcal{P}_\alpha \,:\, \pi(F) = 0\} \ . \tag{P}$$

Sans hypothèse sur F, on pourrait avoir $\mathcal{P}_\alpha^F = \varnothing$ auquel cas $v(\mathrm{P}) = +\infty$. Et même si $\mathcal{P}_\alpha^F \neq \varnothing$, on pourrait encore avoir $v(\mathrm{P}) = +\infty$; dans ce cas, on dit que les solutions de (P) sont dégénérées. En dehors de ces cas dégénérés, on a unicité de la solution :

Proposition 4.2.1. *Si $v(\mathrm{P}) < +\infty$, alors* (P) *admet une unique solution $\pi \in \mathcal{P}_\alpha^F$.*

On introduit maintenant le problème dual

$$v(\mathrm{Q}) := \max_{\theta \in \Theta_F} \inf_{\pi \in \mathcal{P}_\alpha} \mathrm{KL}\left(\pi|\mu\right) + \langle \theta, \pi(F)\rangle \ , \tag{Q}$$

où

$$\Theta_F = \left\{ \theta \in \mathbb{R}^p \mid \int_{\mathbb{R}^d} \exp\left[-\langle \theta, F(x)\rangle\right] \mathrm{d}\mu(x) < +\infty \right\} \ . \tag{4.2}$$

L'inégalité de Hölder implique que Θ_F est convexe.

En définissant $\mathcal{L}(\pi, \theta) = \mathrm{KL}\left(\pi|\mu\right) + \langle \theta, \pi(F)\rangle$ sur $\mathcal{P}_\alpha \times \Theta_F$, on a

$$v(\mathrm{Q}) = \sup_{\Theta_F} \inf_{\mathcal{P}_\alpha} \mathcal{L} \leqslant \inf_{\mathcal{P}_\alpha} \sup_{\Theta_F} \mathcal{L} \leqslant v(\mathrm{P}) \ . \tag{4.3}$$

Dans la suite, on va voir que les éventuelles solutions de (P) sont des lois dont la densité s'écrit, à un négligeable près, sous la forme

$$\frac{\mathrm{d}\pi_\theta}{\mathrm{d}\mu}(x) = \exp[-\langle \theta, F(x)\rangle - L(\theta)] \ , \tag{4.4}$$

où la fonction de log-partition est définie

$$\forall \theta \in \Theta_F, \quad L(\theta) = \log\left[\int_{\mathbb{R}^d} \exp[-\langle \theta, F(x)\rangle] \mathrm{d}\mu(x)\right] \ . \tag{4.5}$$

En fait, le problème (Q) est équivalent à

$$v(\mathrm{Q}) = \min_{\theta \in \Theta_F} L(\theta) \tag{Q'}$$

c'est-à-dire que $\theta^\star$ est solution de (Q) si et seulement s'il est solution de (Q′).

Pour préciser les conditions d'existence du modèle macrocanonique, nous aurons besoin de deux hypothèses :

A1 (α) F est continue et il existe $C_\alpha \in (0, \infty)$ tel que $\sup_{x \in \mathbb{R}^d} \dfrac{||F(x)||}{(1 + ||x||^\alpha)} \leqslant C_\alpha$.

A2 (β) Il existe $\eta > 0$ tel que $\int_{\mathbb{R}^d} \exp[\eta||x||^\beta] \mathrm{d}\mu(x) < +\infty$.

Ces hypothèses permettent de montrer le résultat d'existence suivant.

Proposition 4.2.2. *Supposons* **A1**(α) *avec* $\alpha \geqslant 0$.

(a) Si $v(\mathrm{P}) < +\infty$, *alors il existe* $\theta^\star \in \mathbb{R}^p$ *tel que la solution de* (P) *soit définie par*

$$\forall x \in \mathbb{R}^d, \quad \begin{cases} \frac{\mathrm{d}\pi_{\theta^\star}}{\mathrm{d}\mu}(x) = \dfrac{\exp\left[-\langle \theta^\star, F(x)\rangle\right]}{\displaystyle\int_{\mathbb{R}^d \setminus \mathsf{N}} \exp\left[-\langle \theta^\star, F(y)\rangle\right] \mathrm{d}\mu(y)} & \text{si } x \notin \mathsf{N}\,, \\ \frac{\mathrm{d}\pi_{\theta^\star}}{\mathrm{d}\mu}(x) = 0 & \text{sinon}\,, \end{cases} \tag{4.6}$$

où $\mathsf{N} \in \mathcal{B}(\mathbb{R}^d)$ *est* π*-négligeable pour tout* $\pi \in \mathcal{P}_\alpha^F$ *tel que* $\mathrm{KL}(\pi|\mu) < +\infty$. *Si de plus il existe* $\pi \in \mathcal{P}_\alpha^F$ *tel que* $\mathrm{KL}(\pi|\mu) < +\infty$ *et* $\mu \ll \pi$ *alors* $\theta^\star$ *est une solution de* (Q) *et* $v(\mathrm{Q}) = v(\mathrm{P})$.

(b) Supposons **A2**(β) *avec* $\beta > \alpha$. *Alors* $v(\mathrm{P}) < +\infty$ *si et seulement s'il existe* $\pi \in \mathcal{P}_\alpha^F$ *telle que* $\mathrm{KL}(\pi|\mu) < +\infty$.

En pratique, il est difficile de vérifier les hypothèses de Proposition 4.2.2 car il n'est en général pas aisé de construire $\pi \in \mathcal{P}_\alpha^F$ telle que $\mathrm{KL}(\pi|\mu) < +\infty$. En fait, la formulation duale, qui est un problème en dimension finie, s'avère plus maniable :

Proposition 4.2.3. *Supposons* **A1**(α) *et* **A2**(α) *avec* $\alpha \geqslant 0$. *Alors la fonction* L *définie par* (4.5) *est* $\mathscr{C}^\infty$ *sur* $\mathrm{int}(\Theta_F)$, *et pour tout* $\theta \in \mathrm{int}(\Theta_F)$, $\nabla L(\theta) = \int F d\pi_\theta$ *où* π_θ *est définie par* (4.4). *De plus, s'il existe* $\theta^\star \in \mathrm{int}(\Theta_F)$ *tel que* $\nabla L(\theta^\star) = 0$, *alors* $\pi_{\theta^\star}$ *est une solution de* (P).

La proposition suivante donne une condition plus explicite d'existence du modèle macrocanonique.

Proposition 4.2.4. *Supposons* **A1**(α), **A2**(β) *avec* $\alpha \geqslant 0$, $\beta > \alpha$.

(a) Si $\forall \theta \in \mathbb{R}^p \setminus \{0\}$, *on a* $\mu(\{x \in \mathbb{R}^d : \langle F(x), \theta\rangle < 0\}) > 0$, *alors* (Q) *admet une solution* $\theta^\star$ *et* $\pi_{\theta^\star}$ *(donnée par* (4.4)*) est solution de* (P).

(b) L'hypothèse de (a) est satisfaite si $\mu(\mathsf{A}) > 0$ *pour tout ouvert non-vide* $\mathsf{A} \subset \mathbb{R}^d$, *si* F *est continue sur* Θ_F *et s'il existe* $x \in F^{-1}(\{0\})$ *tel que* F *est différentiable en* x *avec* $\det(DF(x)DF(x)^T) > 0$.

Notons que l'hypothèse de la proposition précédente induit une sorte d'indépendance entre les composantes de F. Nous verrons plus loin comment vérifier cette hypothèse dans le cas de réponses à un réseau de neurones.

4.3 Échantillonnage et Optimisation Stochastique

Dans ce paragraphe, nous allons donner des outils pour échantillonner la distribution exponentielle π_θ rencontrée au paragraphe précédent, et estimer le paramètre θ qui résout le problème de maximum d'entropie sous contrainte, sous la forme (Q′).

Sauf cas très particulier (par exemple, gaussien), l'échantillonnage de la loi π_θ définie par (4.4) passe par l'utilisation d'outils de chaîne de Markov. On peut considérer par exemple l'algorithme de Metropolis-Hastings, qui fournit une chaîne de Markov dont π_θ est une loi invariante. Mais en grande dimension, le taux d'acceptation est souvent trop faible pour que cet algorithme puisse être utilisé en pratique. On va donc se tourner vers d'autres algorithmes sans rejet, et notamment l'algorithme de Langevin.

4.3.1 Algorithme de Langevin

Pour formuler cet algorithme, nous allons préciser la forme de la loi cible π_θ, en posant d'abord une hypothèse sur la mesure de référence μ :

B1 La loi μ admet une densité par rapport à la mesure de Lebesgue, qui s'écrit

$$\frac{d\mu}{dx}(x) = \frac{e^{-r(x)}}{\int_{\mathbb{R}^d} e^{-r(y)}\mathrm{d}y}, \tag{4.7}$$

où $r : \mathbb{R}^d \longrightarrow \mathbb{R}$ est mesurable.

Sous cette hypothèse, la loi cible π_θ de maximum d'entropie admet une densité par rapport à la mesure de Lebesgue donnée par

$$\frac{\mathrm{d}\pi_\theta}{dx}(x) = \frac{e^{-U(\theta,x)}}{\int_{\mathbb{R}^d} e^{-U(\theta,y)}\mathrm{d}y}, \quad \text{où} \quad U(\theta,x) = \langle \theta, F(x)\rangle + r(x). \tag{4.8}$$

En supposant que F et r sont différentiables, l'algorithme de Langevin non-ajusté (ULA) [125] est la chaîne de Markov $(\mathrm{Y}_n)_{n\in\mathbb{N}}$ définie par

$$\forall n \in \mathbb{N}, \quad \mathrm{Y}_{n+1} = \mathrm{Y}_n - \gamma \nabla_x U(\theta, \mathrm{Y}_n) + \sqrt{2\gamma}\mathrm{Z}_{n+1}\ , \tag{4.9}$$

c'est-à-dire

$$\forall n \in \mathbb{N}, \quad \mathrm{Y}_{n+1} = \mathrm{Y}_n - \gamma \left(\sum_{i=1}^{p} \theta(i) \nabla F_i(\mathrm{Y}_n) + \nabla r(\mathrm{Y}_n) \right) + \sqrt{2\gamma}\mathrm{Z}_{n+1}\ , \tag{4.10}$$

où $\gamma > 0$ et où $(\mathrm{Z}_n)_{n\in\mathbb{N}}$ est une suite de v.a. i.i.d. $\mathcal{N}(0, I_d)$. Cet algorithme est la discrétisation d'Euler-Maruyama de l'équation différentielle stochastique de Langevin pour laquelle π_θ est mesure invariante [44]. Le paramètre γ est le pas de discrétisation de cette équation différentielle. L'étude de la convergence géométrique de cette chaîne de Markov pour différentes métriques a été menée dans [43, 44, 36].

Sous des hypothèses techniques, on peut voir que la chaîne discrétisée $(\mathrm{Y}_n)_{n\in\mathbb{N}}$ admet une loi invariante $\pi_{\gamma,\theta}$. Cette loi n'est pas la même que π_θ, mais sous certaines conditions, on peut contrôler le biais entre $\pi_{\gamma,\theta}$ et π_θ. On renvoie à [J11] pour le détail de ces conditions.

4.3.2 Optimisation Stochastique avec Algorithme de Langevin

Dans la suite, nous allons intégrer l'algorithme de Langevin dans un schéma d'optimisation stochastique permettant d'approcher la loi de maximum d'entropie.

Soit $\mathsf{K} \subset \mathrm{int}(\Theta_F)$ un compact convexe non-vide tel que $\mathsf{K} \cap \mathrm{Argmin}_{\Theta_F} L \neq \varnothing$ où L est la log-partition définie par (4.5). Comme L est convexe, on peut chercher à résoudre $\mathrm{Argmin}_{\mathsf{K}} L$ avec la descente de gradient projetée

$$\tilde{\theta}_{n+1} = \Pi_{\mathsf{K}}[\tilde{\theta}_n - \delta \nabla L(\tilde{\theta}_n)] \tag{4.11}$$

où $\delta > 0$ et où Π_{K} est la projection orthogonale sur K. Mais on ne peut évaluer directement $\nabla L(\theta) = \pi_\theta(F)$, ce qui invite à utiliser un gradient stochastique.

L'algorithme "Stochastic Optimization with Unadjusted Langevin" (SOUL) [39] consiste à approcher le gradient en utilisant, à l'itération n, m_n échantillons de Monte-Carlo X_k^n obtenus par l'algorithme de Langevin. En partant de $\theta_0 \in \mathsf{K}$ et $\mathrm{X}_0^0 \in \mathbb{R}^d$, pour chaque $n \in \mathbb{N}$ et $k \in \{0, \dots, m_n - 1\}$, on définit

$$\begin{aligned} \mathrm{X}_{k+1}^n &= \mathrm{X}_k^n - \gamma_n \left(\sum_{i=1}^p \theta_n(i) \nabla F_i(\mathrm{X}_k^n) + \nabla r(\mathrm{X}_k^n) \right) + \sqrt{2\gamma_n} w_{k+1}^n \quad \text{et } \mathrm{X}_0^{n+1} = \mathrm{X}_n^{m_n} \ ; \\ \theta_{n+1} &= \Pi_{\mathsf{K}} \left[\theta_n + \delta_{n+1} m_n^{-1} \sum_{k=1}^{m_n} F(\mathrm{X}_k^n) \right] \ , \end{aligned} \tag{4.12}$$

où $(\delta_n)_{n\in\mathbb{N}*}$, $(\gamma_n)_{n\in\mathbb{N}}$ sont des suites de pas > 0 et où $(\mathrm{Z}_k^n)_{n\in\mathbb{N}, k\in\{1,\dots,m_n\}}$ est une suite de v.a. i.i.d. $\mathcal{N}(0, I_d)$. Cet algorithme permet à la fois de générer une suite (θ_n) approchant la solution de $\mathrm{Argmin}_{\mathsf{K}} L$, et de générer des échantillons X_k^n suivant approximativement la loi de maximum d'entropie.

4.3.3 Garantie de Convergence

Les auteurs de [39] ont formulé des garanties de convergence pour une version généralisée de SOUL. Dans [J11] nous avons adapté ces résultats au cas des lois de maximum d'entropie π_θ, en montrant que les bornes de convergence obtenues ont une dépendance polynomiale en la dimension. Fixons $\alpha \geqslant 1$. Nous aurons besoin des hypothèses suivantes.

B2 (α) Il existe $\mathsf{K} \subset \mathbb{R}^p$ tel que

(a) K est convexe compact non-vide inclus dans $\mathrm{int}(\Theta_F)$ et dans la boule fermée $\bar{\mathrm{B}}(0, R_{\mathsf{K}})$ pour $R_{\mathsf{K}} > 0$;

(b) F est différentiable et il existe $\mathtt{M} \geqslant 0$ tel que

$$\forall x, y \in \mathbb{R}^d, \quad ||F(x) - F(y)|| \leqslant \mathtt{M}(1 + ||x||^{\alpha-1} + ||y||^{\alpha-1})||x - y||; \tag{4.13}$$

(c) il existe $\theta^\star \in \mathrm{int}(\mathsf{K})$ solution de (Q).

B3 Il existe $U_i : \mathsf{K} \times \mathbb{R}^d \longrightarrow \mathbb{R}$, $i \in \{1, 2\}$ telles que

$$\forall \theta \in \mathsf{K}, \ \forall x \in \mathbb{R}^d, \quad U(\theta, x) = U_1(\theta, x) + U_2(\theta, x), \tag{4.14}$$

et

(a) il existe $\mathtt{N} \geqslant 0$ tel que pour tout $i \in \{1, 2\}$, pour tout $\theta \in K$, $U_i(\theta, \cdot)$ est $\mathscr{C}^1$ et l'on a pour tous $x, y \in \mathbb{R}^d$, $||\nabla_x U_i(\theta, x) - \nabla_x U_i(\theta, y)|| \leqslant \mathtt{N}||x - y||$;

(b) il existe $\mathtt{m}_1 > 0$ et $x^\star \in \mathbb{R}^d$ tel que pour tout $\theta \in \mathsf{K}$, $U_1(\theta, \cdot)$ est $\mathtt{m}_1$-fortement convexe et $x^\star \in \mathrm{Argmin}_{x\in\mathbb{R}^d} U_1(\theta, x)$;

(c) il existe $\mathcal{M} \geqslant 0$ tel que pour tous $\theta \in \mathsf{K}, x \in \mathbb{R}^d$, $\|\nabla_x U_2(\theta, x)\| \leqslant \mathcal{M}$.

B4 $F \in \mathrm{C}^1(\mathbb{R}^d, \mathbb{R}^p)$ et a une différentielle $\mathtt{B}$-Lipschitz (avec $\mathtt{B} \geqslant 0$).

À noter qu'au lieu d'avoir un minimum fixe en x^* dans **B**3-(b), on pourrait supposer qu'il existe $R \geqslant 0$ tel que pour tout $\theta \in \mathsf{K}$ il y ait $x^\star_\theta \in \mathrm{Argmin}_{x\in\mathbb{R}^d} U_1(\theta, x)$ avec $x^\star_\theta \in \bar{\mathrm{B}}(0, R)$. Notons aussi que **B**3-(c) peut être remplacée par la condition suivante : il existe $\mathtt{N}_2 \geqslant 0$ avec $\mathtt{N}_2 \leqslant \mathtt{m}_1/2$ tel que pour tous $\theta \in \Theta$ et $x, y \in \mathbb{R}^d$, on ait

$$||\nabla_x U_2(x, \theta) - \nabla_x U_2(y, \theta)|| \leqslant \mathtt{N}_2 ||x - y|| \, . \tag{4.15}$$

On peut maintenant formuler les bornes d'erreur de l'algorithme SOUL, en commençant par le cas fortement convexe, c'est-à-dire quand $U_2 = 0$.

Théorème 4.3.1. *Soit $\alpha \geqslant 1$. Supposons* **A**2(α), **B**1, **B**2(α), **B**3 *avec $U_2 = 0$. Soient $(\gamma_n)_{n\in\mathbb{N}}$, $(\delta_n)_{n\in\mathbb{N}}$ dans $(0, \infty)$ décroissantes, $(m_n)_{n\in\mathbb{N}}$ dans $\mathbb{N}^*$ telles que $\delta_n < 1/(\sup_{\theta\in\mathsf{K}} \|\nabla^2 L(\theta)\|)$ et $\gamma_n < \min(\mathtt{m}_1/(2\mathtt{N}^2), 1/2)$ pour tout $n \in \mathbb{N}$.*

Alors pour tout $n \in \mathbb{N}^*$

$$\mathbb{E}\left[\left\{\sum_{k=1}^n \delta_k L(\theta_k) \Big/ \sum_{k=1}^n \delta_k\right\} - \min_{\mathsf{K}} L\right] \leqslant E_n \Big/ \left(\sum_{k=1}^n \delta_k\right) \, , \tag{4.16}$$

où l'on a défini $(E_n)_{n\in\mathbb{N}}$ selon deux cas :

(a) si $m_n = m_0$ pour tout $n \in \mathbb{N}$ et $\sup_{n\in\mathbb{N}} |\delta_{n+1} - \delta_n| \, \delta_n^{-2} < +\infty$

$$E_n = C_1(1 + d^{\varpi}) \left(1 + \sum_{k=0}^{n-1} \delta_{k+1}\gamma_k^{1/2} + \sum_{k=0}^{n-1} \delta_{k+1}\gamma_{k+1}^{-3}(\gamma_k - \gamma_{k+1}) + \sum_{k=0}^{n-1} \delta_{k+1}^2/\gamma_k^{3/2} + \delta_n/\gamma_n\right) ; \tag{4.17}$$

(b) sinon

$$E_n = C_2(1 + d^{\varpi}) \left(1 + \sum_{k=0}^{n-1} \delta_{k+1}\gamma_k^{1/2} + \sum_{k=0}^{n-1} \delta_{k+1}/(m_k\gamma_k) + \sum_{k=0}^{n-1} \delta_{k+1}^2\gamma_k + \sum_{k=0}^{n-1} \delta_{k+1}^2/(m_k\gamma_k)^2\right) , \tag{4.18}$$

avec $C_1, C_2, \varpi \geqslant 0$ qui ne dépendent pas de la dimension d.

Le théorème suivant donne l'extension au cas non-convexe.

Théorème 4.3.2. *Supposons* **A**2(1), **B**1, **B**2(1), **B**3 *and* **B**4. *Soient $(\gamma_n)_{n\in\mathbb{N}}$, $(\delta_n)_{n\in\mathbb{N}}$ dans $(0, \infty)$ décroissantes, et $(m_n)_{n\in\mathbb{N}}$ dans $\mathbb{N}^*$ avec $\delta_n < 1/(\sup_{\theta\in\mathsf{K}} \|\nabla^2 L(\theta)\|)$ et $\gamma_n < \min(\mathtt{m}_1/(8\mathtt{N}^2), 1/2)$ pour tout $n \in \mathbb{N}$.*

Alors, pour tout $n \in \mathbb{N}^*$,

$$\mathbb{E}\left[\left\{\sum_{k=1}^n \delta_k L(\theta_k) \Big/ \sum_{k=1}^n \delta_k\right\} - \min_{\mathsf{K}} L\right] \leqslant E_n \Big/ \left(\sum_{k=1}^n \delta_k\right) \, , \tag{4.19}$$

où l'on a défini $(E_n)_{n\in\mathbb{N}}$ selon deux cas :

Input : $\theta_0 \in \mathsf{K}$, $\mathrm{X}_0^0 \in \mathbb{R}^d$, $(\gamma_n)_{n\in\mathbb{N}}$, $(\delta_n)_{n\in\mathbb{N}}$, $(m_n)_{n\in\mathbb{N}}$, N
for $n \in \{1, \dots, N-1\}$ **do**
  for $k \in \{0, \dots, m_n - 1\}$ **do**
    Sample $\mathrm{Z}_{k+1}^n \sim \mathrm{N}(0, \mathrm{Id})$
    $\mathrm{X}_{k+1}^n = \mathrm{X}_k^n - \gamma_n \left(\sum_{i=1}^p \theta_n(i) \nabla F_i(\mathrm{X}_k^n) + \nabla r(\mathrm{X}_k^n)\right) + \sqrt{2\gamma_n}\mathrm{Z}_{k+1}^n$
  end
  $X_0^{n+1} = X_{m_n}^n$
  $\theta_{n+1} = \Pi_{\mathsf{K}}\left[\theta_n + \delta_{n+1} m_n^{-1} \sum_{k=1}^{m_n} F(\mathrm{X}_k^n)\right]$
end
Output: Paramètre $\hat{\theta}_N = \frac{\sum_{n=1}^N \delta_n \theta_n}{\sum_{n=1}^N \delta_n}$; Échantillon X_0^N

Algorithm 6: Optimisation Stochastique par Langevin Non-ajusté (SOUL)

(a) si $m_n = m_0$, $\gamma_n = \gamma_0$ *pour tout* $n \in \mathbb{N}$ *avec* $\sup_{n\in\mathbb{N}} |\delta_{n+1} - \delta_n| \, \delta_n^{-2} < +\infty$

$$E_n = C_1(1 + d^{\varpi}) \left(1 + \sum_{k=0}^{n-1} \delta_{k+1} \gamma_0^{1/2} + \sum_{k=0}^{n-1} \delta_{k+1}^2/\gamma_0 + \delta_n/\gamma_0\right) ; \tag{4.20}$$

(b) sinon

$$E_n = C_2(1 + d^{\varpi}) \left(1 + \sum_{k=0}^{n-1} \delta_{k+1} \gamma_k^{1/2} + \sum_{k=0}^{n-1} \delta_{k+1}/(m_k \gamma_k) + \sum_{k=0}^{n-1} \delta_{k+1}^2\right) , \tag{4.21}$$

où $C_1, C_2, \varpi \geqslant 0$ *ne dépendent pas de la dimension* d.

En prenant pour tout $n \in \mathbb{N}$, $m_n = m_0$, $\gamma_n = \gamma_0$ et $\lim_{n\longrightarrow+\infty} \delta_n = 0$ avec $\sum_{k=0}^{+\infty} \delta_n = +\infty$, on peut voir que $\lim_{n\longrightarrow+\infty} \sum_{k=0}^n \delta_k^2 / \sum_{k=0}^n \delta_k = 0$ et le point (a) du Théorème 4.3.1 donne à la limite

$$\limsup_{n\longrightarrow+\infty} \mathbb{E}\left[L(\hat{\theta}_n)\right] - \min_{\mathsf{K}} L \leqslant C_1(1 + d^{\varpi})\gamma_0^{1/2} , \tag{4.22}$$

où le membre de droite ne dépend plus que du pas de discrétisation γ_0.

4.4 Application à la Synthèse de Textures

Pour terminer ce chapitre, nous allons montrer comment les résultats précédents s'appliquent au cas où F encode les réponses moyennes à certaines couches de réseaux de neurones. Cet algorithme SOUL (Algorithme 6) a déjà été utilisé pour la synthèse de textures dans [102], et notre étude vise à préciser les bornes théoriques de convergence associées.

4.4.1 Statistiques d'Ordre 1 sur des Couches de Réseaux de Neurones

Nous allons maintenant appliquer l'algorithme SOUL pour approcher la distribution de maximum d'entropie contrainte par des réponses à un réseau de neurones pré-appris. Nous considérons des réseaux de neurones de la forme

$$\mathscr{G}_j(x) = (\varphi \circ A_j \circ \varphi \circ A_{j-1} \circ \cdots \circ \varphi \circ A_1)(x) , \qquad \mathscr{G}_0(x) = x . \tag{4.23}$$

où $A_j : \mathbb{R}^{c_{j-1} \times n_{j-1}} \longrightarrow \mathbb{R}^{c_j \times n_j}$ est affine, et où $\varphi : \mathbb{R} \longrightarrow \mathbb{R}$ est une fonction, dite d'activation, appliquée composante par composante (par léger abus de notation). On supposera que φ est croissante, dérivable, Lipschitz, avec φ' Lipschitz, que $\lim_{+\infty} \varphi = +\infty$, et qu'elle est à croissance au plus linéaire :

$$\exists C_{d,\varphi} \geqslant 0,\ \forall x \in \mathbb{R}^d, \quad ||\varphi(x)|| \leqslant C_{d,\varphi}(1 + ||x||) . \tag{4.24}$$

Le paramètre J est le nombre de couches du réseau, et par convention, $n_0 = d$ et $c_0 = 1$. Sur chaque couche $j \in \{1, \dots, J\}$, les valeurs de $\mathscr{G}_j(x)$ se décomposent en c_j canaux $\mathscr{G}_j^k(x) \in \mathbb{R}^{n_j}$, $1 \leqslant k \leqslant c_j$. Dans le cas d'un réseau convolutionnel, la partie linéaire de A_j est une convolution (par un noyau matriciel), et $\mathscr{G}_j^k(x)$ se visualise comme une image à n_j pixels. Pour construire le modèle de textures, nous allons extraire les réponses aux couches d'indices $\mathbf{j} \subset \{1, \dots, M\}$, moyennées spatialement. Pour cela, on pose

$$\forall x \in \mathbb{R}^d, \quad \begin{cases} \overline{\mathscr{G}}_j(x) & = \dfrac{1}{n_j} \displaystyle\sum_{i=1}^{n_j} \mathscr{G}_j(x)(i) \quad \in \mathbb{R}^{c_j} \\ F(x) & = \Big(\overline{\mathscr{G}}_j(x) - \overline{\mathscr{G}}_j(x_0)\Big)_{j \in \mathbf{j}} , \end{cases} . \tag{4.25}$$

Ainsi, on a $F : \mathbb{R}^d \longrightarrow \mathbb{R}^p$ avec $p = \sum_{j \in \mathbf{j}} c_j$.

Nous utiliserons le réseau `VGG19` [138], avec trois choix de couches $\mathbf{j}$:

- $\mathbf{j} = \{1, 3, 6, 8, 11, 13\}$ (*shallow CNN*)
- $\mathbf{j} = \{15, 24, 26, 31\}$ (*deep CNN*)
- $\mathbf{j} = \{1, 3, 6, 8, 11, 13, 15, 24, 26, 31\}$ (*full CNN*)

ce qui donne un nombre de paramètres respectivement de $p = 896, 1792, 2688$. En comparaison, la méthode de [51] se base sur des matrices de Gram calculées sur les couches de `VGG19`, ce qui mène à $p = 352256$ statistiques [120]. À noter que le réseau `VGG19` est normalement basé sur la fonction d'activation $\varphi(t) = \max(0, t)$ (RELU) qui n'est pas dérivable. Pour pouvoir appliquer les résultats précédents, dans les expériences on a remplacé cette fonction d'activation par une version régularisée, notée CELU [7].

Nous allons maintenant discuter la validité des hypothèses des résultats précédents pour cette fonction F. Les couches de réseau de neurones étant sous-linéaires, **A**1(1) est vraie. En choisissant pour mesure de référence $\mu = \mathcal{N}(0, \varepsilon\,\mathrm{Id})$, il est clair que **A**2(β) est vraie pour $\beta \in [0, 2)$, et que **B**1 est vraie. Vu que φ est différentiable, la Proposition 4.2.4 est vraie dès qu'il existe $x \in \mathbb{R}^d$ tel que

$F(x) = F(x_0)$ avec $\mathrm{d}F(x)$ surjective, ce qui permet d'assurer l'existence d'une solution de (Q) notée $\theta^\star$.

Dans ce qui suit, on supposera que $\theta^\star \in \mathsf{K} = [-10^4, 10^4]^p$. Cette hypothèse semble réaliste car on a pu constater que les itérés de l'algorithme vérifient $\|\theta_n\|_\infty \leqslant 10^3$ même pour un grand nombre d'itérations n. Avec ce compact K, on obtient que $\mathbf{B}2(\alpha)$ est vraie pour $\alpha \geqslant 1$.

La loi cible s'écrit $\pi_\theta = e^{-U(\theta,x)}dx$ avec

$$U(\theta, x) = U_1(\theta, x) + U_2(\theta, x) \quad \text{avec} \quad \begin{cases} U_1(\theta, x) = \frac{1}{2\varepsilon}\|x\|^2 \\ U_2(\theta, x) = \langle \theta, F(x) \rangle \end{cases} . \tag{4.26}$$

Ainsi, avec des fonctions d'activation régulières, $\mathbf{B}3$, $\mathbf{B}4$ sont valides, et toutes les hypothèses sont réunies pour pouvoir appliquer le Théorème 4.3.2.

Pour le choix des pas de gradients, en prenant

$$\delta_n \sim n^{-a}, \quad \gamma_n \sim n^{-b}, \quad m_n \sim n^c, \tag{4.27}$$

avec $a \in [0, 1]$, $b > 2 - 2a$ et $c > b + 1 - a$ (et donc $c > 3 - 3a$), le Théorème 4.3.2 fournit un taux de convergence

$$\limsup_{n \longrightarrow +\infty} \frac{\sum_{k=1}^n \delta_k L(\theta_k)}{\sum_{k=1}^n \delta_k} - \min_{\mathsf{K}} L = \mathcal{O}(n^{a-1}). \tag{4.28}$$

En particulier, avec $a = 0$, on obtient un taux de convergence en $\mathcal{O}(n^{-1})$, au prix d'un nombre m_n d'échantillons de Langevin augmentant au moins en n^3. Un tel choix de m_n serait très coûteux en temps de calcul. Pour cette raison, les taux de convergence précédents sont difficiles à observer de façon empirique. En pratique, nous utiliserons essentiellement des valeurs fixes de δ_n, γ_n, m_n.

Dans le cadre non-convexe du Théorème 4.3.2, le choix d'un nombre m_n fixe d'échantillons de Langevin ne permet pas d'obtenir la convergence de $\mathbb{E}[L(\hat{\theta}_n)]$ mais donne seulement une borne fixe sur l'erreur, qui ne tend pas vers zéro.

4.4.2 Synthèse de textures avec Algorithme SOUL

L'Algorithme 6 (SOUL) est lancé pendant 10^5 itérations, avec les paramètres suivants

$$\delta_n = 10^{-3}, \quad \gamma_n = 10^{-5}, \quad m_n = 1. \tag{4.29}$$

Les résultats de synthèse obtenus sont donnés en Fig. 4.1, Fig. 4.2 et Fig. 4.3.

On constate que cet algorithme SOUL permet de générer des textures synthétisées d'une qualité visuelle comparable à celles de Gatys *et al.* [51], alors même que notre algorithme SOUL se base sur un nombre de paramètres bien inférieur (voir Fig. 4.2). On observe aussi un gain de qualité par rapport aux modèles décrits dans les chapitres précédents (au prix d'un algorithme de synthèse bien plus coûteux). Alors que les résultats de Texto étaient affectés par une petite quantité de flou (essentiellement due à l'agrégation des patches), les synthèses de SOUL sont toujours affectées d'un léger bruit (inhérent à l'échantillonneur de Langevin). Sur la

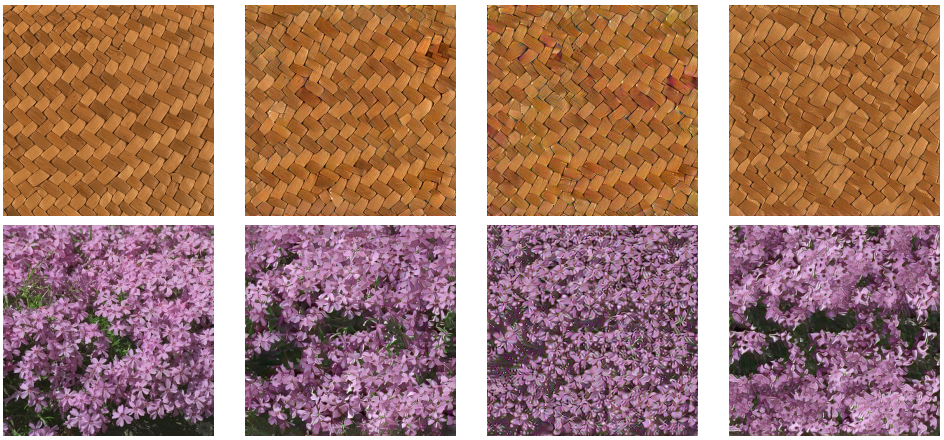
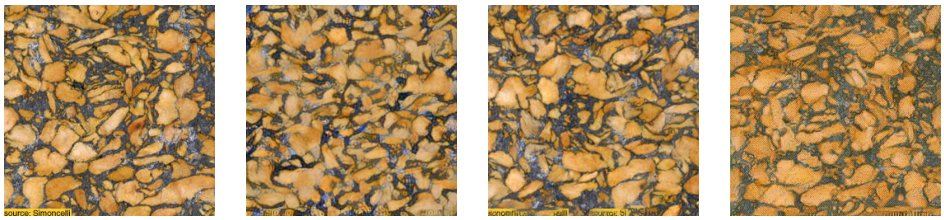


FIGURE 4.1 – **Synthèse de textures avec l'Algorithme SOUL**. On montre ici quelques résultats de synthèse de textures avec l'algorithme SOUL. On constate ici le rôle joué par les couches plus ou moins profondes dans la reproduction des structures géométriques ou fréquentielles de la texture.

Fig. 4.1, on peut constater le rôle joué par les statistiques plus ou moins profondes dans la restitution des structures géométriques.

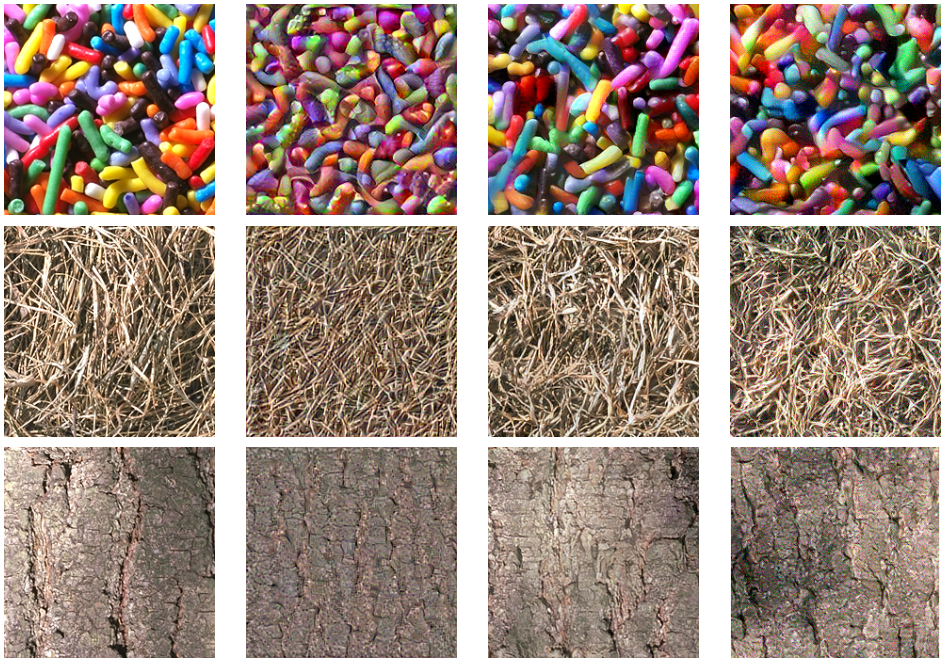
Enfin, sur la Fig. 4.3, on compare par rapport à la méthode deepFRAME introduite par Lu *et al.* [102]. L'algorithme deepFRAME exploite aussi un échantillonnage de Langevin, mais l'apprentissage des paramètres se fait d'une façon progressive : les paramètres associés aux réponses des premières couches sont fixés avant de pouvoir mettre à jour les paramètres associés aux couches suivantes. On constate ici l'intérêt à appliquer l'algorithme SOUL globalement sur toutes les réponses neuronales simultanément. Dans [J11], on a aussi montré que l'algorithme SOUL peut synthétiser des textures de taille quelconque, alors que l'algorithme de Lu *et al.* [102] ne le permet pas.

On renvoie à [J11] (et à ses annexes) pour une collection d'expériences examinant le comportement de l'algorithme plus en détail. Néanmoins, reportons ici qu'avec les paramètres utilisés pour l'algorithme SOUL appliqué aux réponses F extraites de `VGG19`, la convergence des paramètres $(\hat{\theta}_n)$ est difficile à observer en pratique ; on observe néanmoins une stabilisation après un grand nombre d'itérations. On peut en revanche l'observer dans des cas plus simples, et notamment le cas du modèle gaussien ; pour cela, il est utile de faire augmenter m_n après une période de chauffe de la chaîne de Markov. Mentionnons aussi que la gestion du paramètre γ_n est sensible : en le prenant plus grand, l'algorithme de Langevin peut mieux explorer l'espace des images. Mais, en si grande dimension, il est difficile de trouver un compromis permettant à l'algorithme de bien explorer sans diverger.



Original Gatys Ordre 1 Gatys SOUL

FIGURE 4.2 – **Comparaison entre les modèles macrocanonique et microcanonique**. On compare ici les synthèses obtenues "Gatys Ordre 1" (échantillonnage microcanonique avec nos statistiques F), "Gatys" [51] (échantillonnage microcanonique avec des statistiques d'ordre 2), et "SOUL" (échantillonnage macrocanonique avec nos statistiques F). Pour les trois synthèses on utilise les mêmes couches $\mathbf{j} = \mathbf{j}_G = \{1, 3, 6, 11, 13, 15, 24, 26, 31\}$ du réseau `VGG19` .



Original deepFRAME Gatys SOUL

FIGURE 4.3 – **Comparaison avec [102] et [51]**. L'algorithme deepFRAME proposé par Lu *et al.* [102] est aussi inspiré d'un échantillonneur de Langevin, mais l'estimation des paramètres se fait d'une façon progressive depuis les premières couches jusqu'aux couches profondes. On constate aussi que la qualité visuelle avec notre algorithme SOUL est proche de celle de [51] alors que nous n'utilisons que des statistiques d'ordre 1.

4.4.3 Conclusion Partielle

Dans ce chapitre, nous avons fourni des outils pour l'étude des modèles de maximum d'entropie contraints par des réponses moyennes à une collection de statistiques fixée. Nous avons donné des conditions assurant l'existence et l'unicité de la distribution de maximum d'entropie, en se basant sur des formulations primale et duale de ce problème. La solution, qui a la forme d'une distribution exponentielle, peut être échantillonnée avec l'algorithme de Langevin. L'estimation des paramètres solutions peut se faire avec un algorithme d'optimisation stochastique appelé SOUL.

En synthèse de textures, les modèles de maximum d'entropie avaient déjà été utilisés dans [167] et plus récemment dans [102] dont les auteurs proposaient déjà d'échantillonner le modèle avec l'algorithme de Langevin. Notre contribution principale a été de fournir un cadre théorique débouchant sur des résultats de convergence s'appliquant à des statistiques de réponses à des réseaux de neurones avec activations différentiables. De plus, les échantillons d'images obtenus au cours de l'optimisation stochastique fournissent des synthèses dont la qualité visuelle est comparable à des méthodes neuronales récentes (comme celle de Gatys *et al.* [51]), au prix d'un léger bruit résiduel, inhérent à l'échantillonneur de Langevin.

Dans ces travaux, une idée remarquable est que le modèle de maximum d'entropie se base sur des statistiques d'ordre 1 sur les sorties calculées avec F (i.e. on contraint $\mathbb{E}[F(X)]$). En cela, la construction de ce modèle rejoint l'une des conjectures de Julesz [76] : notre perception pré-attentive ne serait sensible qu'à des statistiques de premier ordre calculées sur des *textons*, unités perceptuelles non-linéaires dont la liste n'est pas décrite exhaustivement. Un autre apport majeur de ces travaux tient en l'utilisation de l'utilisation de l'algorithme de Langevin pour répondre à un problème d'optimisation stochastique en très grande dimension. Les résultats théoriques obtenus ouvrent des perspectives intéressantes dans l'utilisation de l'algorithme de Langevin en restauration d'images bayésiennes [89].

Chapitre 5

Problèmes Inverses et Restauration d'Images Plug-and-Play

Ce chapitre regroupe plusieurs avancées obtenues sur la résolution de problèmes inverses pour la restauration de signaux et d'images. Le chapitre commence par quelques rappels sur la formulation variationnelle des problèmes inverses (Section 5.1). Dans la Section 5.2 sont présentés des travaux sur les bassins d'attraction d'algorithmes d'optimisation pour la résolution de problèmes inverses [J13], effectués en collaboration avec Jean-François Aujol et Yann Traonmilin. Enfin, la Section 5.3 est consacrée à l'étude d'une nouvelle classe d'algorithmes de restauration d'images, dits Plug-and-Play [C11, C12]. Les résultats présentés dans cette section font l'objet de la thèse de Samuel Hurault, doctorant à l'Institut de Mathématiques de Bordeaux que je co-encadre avec Nicolas Papadakis.

5.1 Problèmes Inverses, Prior, Optimisation

La restauration d'images est une problématique consistant à annuler l'effet de différentes dégradations pouvant être liées au bruit d'acquisition, au flou de bouger, au flou de défocalisation, à une occultation, etc. Bien que certaines dégradations soient une combinaison complexe d'effets non linéaires et non stationnaires, on se limitera ici à l'étude de problèmes inverses linéaires. Plus précisément, on considérera un modèle de dégradation

$$y = Ax_0 + w \tag{5.1}$$

où le signal propre x_0 vit dans un espace vectoriel normé $\mathcal{X}$ de dimension finie ou infinie, où $A : \mathcal{X} \longrightarrow \mathbb{R}^m$ est un opérateur linéaire, et où $w \in \mathbb{R}^m$ est un bruit blanc gaussien d'écart-type $\sigma > 0$. Ce cadre de problème inverse linéaire permet aussi d'englober d'autres problèmes de traitement de signal comme par exemple la factorisation de matrices de rang faible, la déconvolution parcimonieuse (*spike recovery*), ou encore l'échantillonnage compressé.

Avec le modèle de bruit blanc gaussien, on peut chercher à retrouver x_0 en maximisant la vraisemblance $p(y|x)$ du modèle, ce qui revient à minimiser

$$f(x) = -\log p(y|x) = \frac{1}{2\sigma^2}\|Ax - y\|^2. \tag{5.2}$$

Dans bon nombre de problèmes inverses, l'opérateur A n'est pas injectif, et la minimisation de (5.2) présente alors une infinité de solutions. Même lorsqu'il est injectif,

l'opérateur inverse A^{-1} peut être instable, ce qui conduit à amplifier le bruit lors de la résolution de (5.2). Pour pallier cela, on ajoute de l'information a priori sur la solution pour stabiliser le processus d'inversion.

Modèles de faible dimension. Une première façon de lever ce problème est de supposer que x appartient à un sous-ensemble $\Sigma \subset \mathcal{X}$, appelé "modèle de faible dimension". Cela ramène à un problème d'optimisation contraint

$$\operatorname*{Argmin}_{x \in \Sigma} \|Ax - y\|^2. \tag{5.3}$$

Sous certaines hypothèses sur A et Σ (dites d'isométrie restreinte) [18], on peut montrer que ce problème a une solution unique x^*, qui permet d'approcher le signal original $x_0 \in \Sigma$ à une certaine précision :

$$\|x^* - x_0\|^2 \leqslant C\|w\|^2 \tag{5.4}$$

où $C > 0$ est une constante. Cependant, l'ensemble Σ est en général non-convexe (par exemple, une union de sous-espaces vectoriels), ce qui complique la résolution de (5.3). En pratique, on peut décider de paramétrer l'ensemble Σ par une fonction $\varphi : \mathbb{R}^d \longrightarrow \mathcal{X}$ dont l'image contient Σ. En notant $\Theta = \varphi^{-1}(\Sigma)$, on aboutit au problème

$$\theta^* \in \operatorname*{Argmin}_{\theta \in \Theta} \|A\varphi(\theta) - y\|^2. \tag{5.5}$$

Dans la Section 5.2, on donnera un cadre général pour l'étude de la convergence de l'algorithme de gradient à pas fixe visant à minimiser (5.5). En particulier, on s'attachera à expliciter des bassins d'attraction de l'algorithme, c'est-à-dire des ensembles dans lesquels on peut initialiser l'algorithme de descente de gradient de façon à garantir la convergence des itérées vers x^*.

Régularisation explicite ou non. Une autre façon de stabiliser l'inversion est d'intégrer une information a priori sur x au travers d'une modélisation bayésienne. Ceci consiste à adopter une loi a priori notée π, en lien avec un certain modèle de régularité. Ce prior s'écrit souvent $\pi(x) = e^{-\lambda g(x)}$ où $g(x)$ quantifie la régularité de x et où $\lambda \geqslant 0$ est un paramètre de régularisation. La loi a posteriori s'écrit alors

$$p(x|y) \propto p(y|x)\pi(x) = e^{-f(x)-\lambda g(x)}, \tag{5.6}$$

si bien que la recherche du maximum a posteriori débouche sur la minimisation de la fonction

$$F(x) = f(x) + \lambda g(x). \tag{5.7}$$

De nombreux a priori ont été proposés dans la littérature, permettant de contrôler différents types de régularité. En premier lieu, on peut évoquer les a priori basés sur des semi-normes : la semi-norme H^1 permettant de contrôler l'énergie du gradient [147], la variation totale [127] (norme L^1 du gradient) ou encore la norme L^1 des coefficients d'ondelettes [38]. Ces a priori élémentaires ont l'avantage d'être des fonctions convexes. Le minimum de F peut alors admettre une forme explicite ou

être approché par des algorithmes itératifs avec garantie de convergence [38, 23]. Mais les performances en restauration avec ces a priori sont limitées puisqu'ils ne permettent pas de rendre compte de la complexité des structures présentes dans les images naturelles.

Pour construire des modèles plus performants, on peut tirer profit des structures récurrentes présentes dans une base de données d'images propres, ou même des redondances internes observées dans l'image à restaurer. C'est le point de vue adopté par les auteurs de la méthode NL-means de débruitage par moyennes non locales [21], qui consiste à débruiter un patch en faisant simplement une moyenne pondérée des patches similaires. Ce principe de moyennes non locales ne s'exprime pas à proprement parler comme un a priori global sur l'image, mais cela débouche sur l'idée d'effectuer des traitements locaux au niveau des patches, qui seront agrégés de façon à obtenir l'image restaurée. Ceci suggère donc de localiser la modélisation bayésienne en adoptant un log-prior de la forme

$$g(x) = -\sum_{i \in I} \log \psi_i(P_i x) \tag{5.8}$$

où $(P_i x)_{i \in I}$ est la collection des patches de x, et où ψ_i est un prior local adapté au patch en position i. L'optimisation est découplée en introduisant des variables de patches auxiliaires $z = (z_i)_{i \in I}$

$$\tilde{F}(x, z) = \|Ax - y\|^2 + \beta \sum_{i \in I} \|P_i x - z_i\|^2 - \lambda \log \psi_i(z_i), \tag{5.9}$$

et l'optimisation en z_i correspond alors à un débruitage bayésien du patch $P_i x$.

Cette modélisation bayésienne par patch permet d'englober plusieurs lignes de travaux ultérieurs. Certains vont utiliser un prior générique ψ pour tous les patches. C'est le cas des méthodes d'apprentissage de dictionnaires [49, 103, J1] où l'a priori impose une décomposition parcimonieuse des patches dans un dictionnaire (pouvant être une famille orthogonale ou non). Ces méthodes sont donc basées sur l'utilisation au niveau local de modèles de faible dimension, déjà évoqués ci-dessus sous la forme contrainte (5.3). À l'inverse, une autre ligne de travaux consiste à utiliser plusieurs modèles gaussiens φ_i pouvant être estimés de façon générique [161] ou bien directement sur l'image [168, 90]. Ces modèles gaussiens locaux peuvent être obtenus, à la manière de [168, 161], comme les composantes d'un modèle de mélange de gaussiennes estimé avec l'algorithme *Expectation-Maximization* (EM). Ils peuvent aussi être appris plus directement en groupant les patches selon leur similarité, et en estimant un modèle gaussien par groupe (deux patches similaires $P_i x \approx P_j x$ se verront alors attribuer le même modèle gaussien $\psi_i = \psi_j$).

Ces diverses modélisations par patches ont encore nettement relevé le niveau des performances en restauration d'images jusque [91, 71]. Cependant, le contrôle numérique de ces méthodes est délicat. D'une part, le maximum a posteriori global n'est généralement obtenu que de façon approchée en restaurant d'abord tous les patches puis en les combinant avec une méthode d'agrégation, ce qui revient à une optimisation alternée de (5.9). D'autre part, il y a une interaction non triviale entre l'estimation des modèles gaussiens et le calcul du maximum a posteriori ; en pratique on alterne entre ces deux étapes, avec un critère d'arrêt arbitraire.

Depuis l'essor spectaculaire des réseaux de neurones en traitement d'images, de nouveaux algorithmes de restauration d'images ont été proposés, exploitant un réseau débruiteur appris sur une base de données d'images propres [164, 109, 128, 162]. Ces méthodes reposent sur un principe d'éclatement (*splitting*) consistant à utiliser un algorithme itératif alternant entre une étape de descente sur l'attache aux données et une étape de régularisation. Un débruiteur pré-appris est alors utilisé en lieu et place de l'étape de régularisation, qui n'est donc pas directement exprimée avec un critère numérique $g(x)$. L'intérêt est ici qu'il s'avère plus facile de concevoir un algorithme de débruitage atteignant de très bonnes performances sur une base d'images plutôt que de concevoir un priori représentant bien ces données.

L'objectif principal des travaux présentés dans la Section 5.3 est de proposer une méthode de restauration d'images dans laquelle la fonction de régularisation $g(x)$ est explicite, tout en bénéficiant des performances en débruitage atteintes par les réseaux de neurones profonds. Une fois la régularisation $g(x)$ apprise, on pourra utiliser des algorithmes itératifs de minimisation de la fonctionnelle (5.7) fournissant des garanties de convergence. On veillera à ce que ces garanties soient basées sur des hypothèses réalistes satisfaites en pratique par les réseaux de neurones profonds utilisés, menant ainsi à des résultats de restauration à l'état de l'art, et permettant de garder un contrôle numérique précis.

5.2 Bassins d'Attraction pour un Problème Inverse

Cette section est consacrée à des travaux effectués en collaboration avec Yann Traonmilin et Jean-François Aujol [J13], portant sur l'étude des bassins d'attraction d'algorithmes résolvant le problème (5.5) avec un modèle de faible dimension Σ paramétré par une fonction φ. Dans la Section 5.2.1 on donne un cadre général permettant d'étudier la convergence de la descente de gradient à pas fixe associée à (5.5). Sous une hypothèse d'isométrie restreinte, on fournit dans la Section 5.2.2 des conditions suffisantes pour exhiber un bassin d'attraction. On discute enfin dans la Section 5.2.3 les bassins obtenus pour trois exemples de problèmes inverses.

5.2.1 Problème Paramétré

Pour permettre l'étude de problèmes inverses où l'objet d'intérêt x est, par exemple, une distribution, on introduit un espace vectoriel topologique localement convexe $\mathcal{D}$ et son dual $\mathcal{D}^*$. On supposera que le signal inconnu x_0 appartient à un ensemble $\Sigma \subset \mathcal{D}^*$ appelé "modèle de faible dimension" et tel que $\forall \lambda > 0, \forall x \in \Sigma$, $\lambda x \in \Sigma$. On suppose disposer d'une paramétrisation φ de Σ, c'est-à-dire d'une fonction $\varphi : \mathbb{R}^d \longrightarrow \mathcal{D}^*$ telle que $\Sigma \subset \varphi(\mathbb{R}^d) = \{\varphi(\theta) : \theta \in \mathbb{R}^d\}$. On peut alors résoudre le problème inverse contraint (5.3) en minimisant sur $\Theta := \varphi^{-1}(\Sigma)$ la fonction

$$h(\theta) := \|A\varphi(\theta) - y\|_2^2. \tag{5.10}$$

À cause de la paramétrisation, cette fonction h peut être non-convexe. Dans la suite, on va étudier la convergence de la descente de gradient à pas fixe $\tau > 0$:

$$\theta_{n+1} = \theta_n - \tau \nabla h(\theta_n). \tag{5.11}$$

Définition 5.2.1. On dit que $\Lambda \subset \mathbb{R}^d$ est un h-bassin d'attraction de $\theta^* \in \Lambda$ si

$$\exists \tau_0 > 0,\ \forall \tau \in (0, \tau_0],\ \forall \theta_0 \in \Lambda, \quad h(\theta_n) \longrightarrow h(\theta^*). \tag{5.12}$$

Pour contourner le défaut de convexité de h, on va contrôler la hessienne de h dans certaines directions qui suffisent à assurer la stabilité de (θ_n) dans Λ et la convergence de $h(\theta_n)$. Nous aurons besoin de dériver φ en un sens faible :

Définition 5.2.2. On dit que $\varphi : \mathbb{R}^d \longrightarrow \mathcal{D}^*$ est Gateaux-différentiable en θ s'il existe une application linéaire $\varphi'(\theta) : \mathbb{R}^d \longrightarrow \mathcal{D}^*$ telle que pour tout $u \in \mathbb{R}^d$,

$$\frac{\varphi(\theta + hu) - \varphi(\theta)}{h} \xrightarrow[h \longrightarrow 0]{} \varphi'(\theta)u \quad \text{dans } \mathcal{D}^*. \tag{5.13}$$

On écrira aussi $\partial_u \varphi(\theta) = \varphi'(\theta)u$, et $\nabla \varphi(\theta) = \left(\frac{\partial \varphi(\theta)}{\partial \theta_i}\right)_{1 \leqslant i \leqslant d}$ le Gateaux-gradient.

Proposition 5.2.1. *Supposons $A : \mathcal{D}^* \longrightarrow \mathbb{C}^m$ linéaire continu et $\varphi : \mathbb{R}^d \longrightarrow \mathcal{D}^*$ deux fois Gateaux-différentiable. Alors h est deux fois Gateaux-différentiable en tout $\theta \in \mathbb{R}^d$ et*

$$\frac{\partial h(\theta)}{\partial \theta_i} = 2\mathcal{R}e\langle A\frac{\partial \varphi(\theta)}{\partial \theta_i}, A\varphi(\theta) - y\rangle. \tag{5.14}$$

$$\frac{\partial^2 h(\theta)}{\partial \theta_i \partial \theta_j} = 2\mathcal{R}e\langle A\frac{\partial \varphi(\theta)}{\partial \theta_i}, A\frac{\partial \varphi(\theta)}{\partial \theta_j}\rangle + 2\mathcal{R}e\langle A\frac{\partial^2 \varphi(\theta)}{\partial \theta_i \partial \theta_j}, A\varphi(\theta) - y\rangle. \tag{5.15}$$

Fixons une paramétrisation $\varphi : \mathbb{R}^d \longrightarrow \Sigma$ Gateaux-différentiable. Dans les formules ci-dessus, on voit apparaître les valeurs de A sur les Gateaux-dérivées du paramétrage φ. Pour contrôler ces termes, on a besoin d'une propriété de stabilité de A sur un ensemble $\overline{\mathcal{S}(\Sigma)}$ appelé "sécant généralisé" défini comme l'union du sécant $\Sigma - \Sigma = \{\ x - y\ ,\ x, y \in \Sigma\ \}$ et de $\{\partial_u \varphi(\theta), u, \theta \in \mathbb{R}^d\}$. Cette propriété s'exprime via la norme d'un espace de Hilbert $(\mathcal{H}, \|\cdot\|_{\mathcal{H}})$ contenant Σ ainsi que les dérivées $\partial_u \varphi(\theta)$.

Définition 5.2.3. On dit que l'opérateur A vérifie la propriété d'isométrie restreinte RIP(γ) sur $\mathcal{S}(\Sigma)$ pour la norme $\|\cdot\|_{\mathcal{H}}$ si

$$\forall x \in \mathcal{S}(\Sigma), \quad (1-\gamma)\|x\|_{\mathcal{H}}^2 \leqslant \|Ax\|_2^2 \leqslant (1+\gamma)\|x\|_{\mathcal{H}}^2. \tag{5.16}$$

On peut montrer [119] que certains opérateurs aléatoires A à valeurs dans $\mathbb{R}^m$ vérifient la RIP avec forte probabilité dès que $m \geqslant O(d\text{polylog}(d))$, c'est-à-dire lorsque le nombre de mesures m est de l'ordre de la dimension d aux facteurs logarithmiques près. Sous une hypothèse de compatibilité du sécant généralisé avec $\|\cdot\|_{\mathcal{H}}$, on peut alors étendre la RIP à $\overline{\mathcal{S}(\Sigma)}$.

Lemma 5.2.1 (RIP sur le sécant généralisé)**.** *Supposons $A : \mathcal{D}^* \longrightarrow \mathbb{C}^m$ linéaire continu et $\varphi : \mathbb{R}^d \longrightarrow \mathcal{D}^*$ Gateaux-différentiable. Supposons que A vérifie la* RIP(γ) *sur $\Sigma - \Sigma$ et que $\overline{\mathcal{S}(\Sigma)}$ est compatible avec $\|\cdot\|_{\mathcal{H}}$, c'est-à-dire que pour tout $x = \partial_u \varphi(\theta)$ et toute suite $h_n \longrightarrow 0$, $\left\| \frac{\varphi(\theta+|h_n|u)-\varphi(\theta)}{|h_n|} \right\|_{\mathcal{H}}$ converge vers une limite $\|\partial_u \varphi(\theta)\|_{\mathcal{H}}$ indépendante de (h_n). Alors*

$$\forall \nu \in \overline{\mathcal{S}(\Sigma)}, \quad (1-\gamma)\|\nu\|_{\mathcal{H}}^2 \leqslant \|A\nu\|_2^2 \leqslant (1+\gamma)\|\nu\|_{\mathcal{H}}^2 . \tag{5.17}$$

5.2.2 Existence des Bassins d'Attraction

On va maintenant pouvoir contrôler la hessienne de h avec l'hypothèse suivante.

Hypothèse 5.2.1 Supposons que

(a) $A : \mathcal{D}^* \longrightarrow \mathbb{C}^m$ est linéaire continu,

(b) $\varphi : \mathbb{R}^d \longrightarrow \mathcal{D}^*$ est deux fois Gateaux-différentiable,

(c) A vérifie la RIP(γ) sur $\Sigma - \Sigma$,

(d) $\overline{\mathcal{S}(\Sigma)}$ est compatible avec $\|\cdot\|_{\mathcal{H}}$ (voir Lemme 5.2.1).

Lemma 5.2.2. *Supposons l'Hypothèse 5.2.1. Soit $\theta^* \in \Theta$ tel que $h(\theta^*) = \min_\Theta h$. Soit $\Lambda \subset \mathbb{R}^d$ tel que pour tout $\theta \in \Lambda$, $\varphi(\theta) - \varphi(\theta^*) \in \mathcal{S}(\Sigma)$. Pour $\theta \in \Lambda$ et $u \in \mathbb{R}^d$, on a*

$$u^T \nabla^2 h(\theta) u \geqslant 2(1-\gamma)\|\partial_u \varphi(\theta)\|_{\mathcal{H}}^2 - 2\|A\partial_u^2 \varphi(\theta)\|_2 (\sqrt{1+\gamma}\|\varphi(\theta) - \varphi(\theta^*)\|_{\mathcal{H}} + \|e\|_2), \tag{5.18}$$

$$u^T \nabla^2 h(\theta) u \leqslant 2\|A\partial_u \varphi(\theta)\|_2^2 + 2\|A\partial_u^2 \varphi(\theta)\|_2 (\sqrt{1+\gamma}\|\varphi(\theta) - \varphi(\theta^*)\|_{\mathcal{H}} + \|e\|_2). \tag{5.19}$$

Ce contrôle de la hessienne s'avère suffisant pour donner des bassins d'attraction qui seront de la forme

$$\Lambda_\beta := \{\theta \in \Theta \mid d(\theta, \theta^*) < \beta\}, \quad (\beta > 0) \tag{5.20}$$

où l'on a défini une distance et une projection ensembliste sur $\varphi^{-1}(\varphi(\theta^*))$:

$$d(\theta, \theta^*) := \min_{\substack{\tilde{\theta} \in \Theta \\ \varphi(\tilde{\theta}) = \varphi(\theta^*)}} \|\tilde{\theta} - \theta\|_2 , \quad p(\theta, \theta^*) := \operatorname*{argmin}_{\substack{\tilde{\theta} \in \Theta \\ \varphi(\tilde{\theta}) = \varphi(\theta^*)}} \|\tilde{\theta} - \theta\|_2 \subset \Theta. \tag{5.21}$$

L'hypothèse suivante permet de contrôler les variations de φ dans des directions particulières qui servent pour l'analyse de la descente de gradient.

Hypothèse 5.2.2 Étant donné $\theta^* \in \Theta$, $\beta > 0$. On suppose que

(a) $\forall \theta \in \Lambda_{2\beta},\ \varphi(\theta) \in \Sigma$,

(b) $\exists C_{\varphi,\theta^*} > 0,\ \forall \theta \in \Lambda_{2\beta}, \quad \|\varphi(\theta) - \varphi(\theta^*)\|_{\mathcal{H}} \leqslant C_{\varphi,\theta^*} d(\theta, \theta^*)$,

(c) $M_1 := \sup_{\theta \in \varphi^{-1}(\varphi(\theta^*))} \sup_{\|u\|_2 = 1} \|A\partial_u \varphi(\theta)\|_2 < +\infty$,

(d) $M_2 := \sup_{\theta \in \Lambda_{2\beta}} \sup_{\|u\|_2 = 1, \|v\|_2 = 1} \|A\partial_v \partial_u \varphi(\theta)\|_2 < +\infty$.

Théorème 5.2.2. *Soit $\theta^* \in \Theta$ tel que $h(\theta^*) = \min_{\Theta} h$.*
Soit Λ_β défini par (5.20) avec $\beta > 0$. Supposons les Hypothèses 5.2.1, 5.2.2. Supposons enfin que pour tout $\theta \in \Lambda_\beta$, il existe $\tilde{\theta} \in p(\theta, \theta^*)$ tel que

$$\forall z \in [\theta, \tilde{\theta}], \quad \frac{(1-\gamma)\|\partial_{\tilde{\theta}-\theta}\varphi(z)\|_{\mathcal{H}}^2}{\sqrt{1+\gamma}\|A\partial^2_{\tilde{\theta}-\theta}\varphi(z)\|_2} \geqslant C_{\varphi,\theta^*}\beta + \frac{1}{\sqrt{1+\gamma}}\|e\|_2. \tag{5.22}$$

Alors Λ_β est un h-bassin d'attraction de θ^*.

Ce théorème montre que l'on peut exhiber un bassin d'attraction avec une hypothèse RIP sur A, des bornes sur les variations du paramétrage φ (vues au travers de A) et une borne (5.22) sur des taux de variation d'ordre 2. On peut simplifier cette dernière hypothèse lorsque $p(\theta, \theta^*)$ est un singleton :

Corollary 5.2.3. *Soit $\theta^* \in \Theta$ tel que $h(\theta^*) = \min_{\Theta} h$. Supposons l'Hypothèse 5.2.1, et supposons l'Hypothèse 5.2.2 pour un certain $\beta_1 > 0$ tel que pour tout $\theta \in \Lambda_{2\beta_1}$, il existe un unique $\tilde{\theta} \in p(\theta, \theta^*)$. Supposons enfin que*

$$\beta_2 := \frac{(1-\gamma)}{C_{\varphi,\theta^*}\sqrt{1+\gamma}} \inf_{\theta\in\Lambda_{\beta_1}} \inf_{z\in[\theta,\tilde{\theta}]} \left(\frac{\|\partial_{\tilde{\theta}-\theta}\varphi(z)\|_{\mathcal{H}}^2}{\|A\partial^2_{\tilde{\theta}-\theta}\varphi(z)\|_2} \right) - \frac{1}{C_{\varphi,\theta^*}\sqrt{1+\gamma}}\|e\|_2 > 0. \tag{5.23}$$

Alors $\Lambda_{\min(\beta_1,\beta_2)}$ est un h-bassin d'attraction de θ^*.

Avec ce corollaire, pour peu que $\tilde{\theta} \in p(\theta, \theta^*)$ soit unique, on obtient un bassin d'attraction dès que

$$\inf_{\theta\in\Lambda_{\beta_1}} \inf_{z\in[\theta,\tilde{\theta}]} \left(\frac{\|\partial_{\tilde{\theta}-\theta}\varphi(z)\|_{\mathcal{H}}^2}{\|A\partial^2_{\tilde{\theta}-\theta}\varphi(z)\|_2} \right) > 0, \tag{5.24}$$

ce qui revient à borner les variations d'ordre 2 de $A\varphi$ relativement aux variations du paramétrage dans $\mathcal{H}$. À noter aussi que la preuve du théorème donne une vitesse de convergence en $h(\theta_n) - h(\theta^*) = \mathcal{O}(\frac{1}{n})$, et que l'on peut également obtenir une vitesse de convergence de $\varphi(\theta_n)$ vers x_0 à condition de réduire légèrement la taille du bassin d'attraction.

Les résultats précédents assurent la convergence de la descente de gradient, en transmettant la RIP au sécant généralisé, et en minorant un certain taux de variation de $A\varphi$ relativement au paramétrage. La preuve repose sur l'obtention de la convexité de h dans les directions $z \in [\theta, \tilde{\theta}]$ liant le θ courant et une solution $\tilde{\theta} \in p(\theta, \theta^*)$ proche de z. On va évoquer dans le paragraphe suivant plusieurs contextes où ces résultats s'appliquent.

Mais avant, notons que Puy *et al.* [119] ont montré que l'on peut construire des opérateurs aléatoires A qui, avec forte probabilité, satisfont RIP(γ) pour $m = \mathcal{O}(\frac{d}{\gamma^2})$. Dans ce contexte, on peut considérer que γ est de l'ordre de $\frac{1}{\sqrt{m}}$, et donc, lorsque le nombre de mesures $m \longrightarrow \infty$, la taille β des bassins d'attraction obtenue est de l'ordre de $\frac{1-\gamma}{\sqrt{1+\gamma}} = 1 - \frac{3}{2}\gamma + o(\gamma) \approx 1 - \frac{3}{2\sqrt{m}}$. Dans une étude ultérieure [Js1], nous avons questionné la taille maximale des bassins d'attraction, et en particulier discuté la dépendance en m.

5.2.3 Applications

Dans [J13] nous avons appliqué les résultats précédents à plusieurs problèmes inverses, notamment la factorisation de matrices de rang faible. Ici nous allons seulement évoquer brièvement deux autres applications.

Déconvolution parcimonieuse Pour le problème de déconvolution sans grille de signaux impulsionnels, on se place dans le dual $\mathcal{D}^*$ de l'espace $\mathcal{D} = \mathcal{C}^2_b(\mathbb{R}^p)$ des fonctions $\mathcal{C}^2$ bornées, qui contient les distributions d'ordre $\leqslant 2$ à support compact. On considère l'ensemble paramétré des combinaisons de k impulsions ϵ-séparées

$$\Sigma_{k,\epsilon} := \left\{ \sum_{i=1}^{k} a_i \delta_{t_i} \ , \ \|t_i - t_j\|_2 > \epsilon, \|t_i\|_2 < R \right\} \tag{5.25}$$

$$\text{avec } \varphi(\theta) = \sum_{i=1}^{k} a_i \delta_{t_i} \ , \ \theta = (a_1, \ldots, a_k, t_1, \ldots, t_k) \in \mathbb{R}^k \times (\mathbb{R}^p)^k. \tag{5.26}$$

L'opérateur A consiste en des mesures fréquentielles $(Ax)_l = \langle x, \alpha_l \rangle$ où les α_l correspondent à une sélection de fréquences, régulière ou aléatoire. Pour un tel A, on peut effectivement montrer une RIP.

Sur $\Sigma_{k,\epsilon}$, on utilise la norme à noyau associée à $K(t,s) \propto e^{-\frac{\|t-s\|_2^2}{2\sigma^2}}$:

$$\Big\| \sum_i a_i \delta_{t_i} \Big\|^2_{\mathcal{H}} = \sum_i a_i a_j K(t_i - t_j), \tag{5.27}$$

où σ^2 est le paramètre de précision de la norme à noyau. La RIP pour cet opérateur, et son extension au sécant généralisé a été prouvée dans Traonmilin et Aujol [149]. Le théorème générique ci-dessus permet alors de retrouver les bassins d'attraction donnés dans [149, Corollary 3.1].

Estimation de GMM par échantillonnage compressé Pour l'estimation compressée de modèles de mélanges de gaussiennes (GMM), on se place dans le dual $\mathcal{D}^*$ de l'espace $\mathcal{D} = \mathcal{C}_b(\mathbb{R}^p)$, qui contient les mesures finies sur $\mathbb{R}^p$. On se restreint ici au cas où les composantes gaussiennes ont toutes la même matrice de covariance Γ fixée. En notant $\mu_{t_i} = e^{-\frac{1}{2}(t-t_i)^T\Gamma^{-1}(t-t_i)}dt$, on considère le modèle paramétré

$$\Sigma_{k,\epsilon,\Gamma} := \{\sum_{i=1}^{k} a_i \mu_{t_i} : \|t_i - t_j\|_\Gamma > \epsilon, \|t_i\|_2 < R\} \tag{5.28}$$

$$\text{avec } \varphi(\theta) = \sum_{i=1}^{k} a_i \mu_{t_i} \ , \ \theta = (a_1, \ldots, a_k, t_1, \ldots, t_k) \in \mathbb{R}^k \times (\mathbb{R}^p)^k. \tag{5.29}$$

On va de nouveau utiliser la norme à noyau (5.27). Comme pour la déconvolution parcimonieuse, l'opérateur A effectue des mesures fréquentielles. Pour une distribution gaussienne de fréquences aléatoires, la RIP(γ) sur $\mathcal{S}(\Sigma_{k,\epsilon,\Gamma})$ a été prouvée par Gribonval *et al.* [63].

Théorème 5.2.3. *Supposons que l'opérateur de mesures fréquentielles A vérifie* $\mathrm{RIP}(\gamma)$ *sur* $\mathcal{S}(\Sigma_{k,\frac{\epsilon}{2},\Gamma})$. *Soit* $\theta^* = (a_1, \dots, a_k, t_1, \dots, t_k)$ *une solution de* $\min_\Theta h$. *Sous une hypothèse technique sur la norme à noyau* $\|\cdot\|_{\mathcal{H}}$, *il existe une constante explicite* β_{GMM} *telle que, en supposant* $\forall i\ \|t_i\|_2 \leqslant R - 2\beta_{GMM}$ *pour tout* i, *on a un* h*-bassin d'attraction de* θ^* *de la forme*

$$\Lambda_{\beta_{GMM}} := \{\ \theta \in \mathbb{R}^{k(p+1)} \mid \|\theta - \theta^*\|_2 < \beta_{GMM}\ \}. \tag{5.30}$$

En pratique, l'estimation d'un modèle GMM se fait souvent avec des covariances variables. Mais pour un tel modèle, on ne dispose pas encore d'un résultat analogue sur les bassins d'attraction.

5.3 Restauration d'Images Plug-and-Play

Cette section est consacrée aux travaux effectués par Samuel Hurault dans le cadre de sa thèse co-encadrée par Nicolas Papadakis et moi-même, traitant de modèles de restauration d'images en lien avec l'apprentissage profond.

5.3.1 Apprentissage Profond pour les Problèmes Inverses

En parallèle des méthodes basées sur une régularisation explicite, de nouvelles techniques de restauration d'images ont été développées, profitant de l'essor de l'apprentissage profond [84, 138]. Des réseaux de neurones pour le débruitage ont d'abord été proposé dans [74, 22].Quelques années plus tard, des débruiteurs plus performants [163, 131, 165] ont été construits en bénéficiant d'une part des avancées sur les architectures de réseaux de neurones et leur optimisation, et d'autre part de l'augmentation de la puissance de calcul sur GPU.

Ces méthodes suivent un principe simple de régression : on optimise le réseau de neurones pour qu'il apprenne la relation entre l'image dégradée et l'image propre. Plus précisément, à partir d'une base de données $(x_n)_{1\leqslant n\leqslant N}$ d'images propres modélisée par une distribution empirique p, on considère $y = x + w$ où w est un bruit blanc de loi gaussienne $\mathcal{N}$ indépendant de $x \sim p$. Le réseau débruiteur D est entraîné pour minimiser la fonction de coût

$$\mathcal{L}(D) = \mathbb{E}_{x\sim p, w\sim\mathcal{N}}\Big[\|D(x+w) - x\|^2\Big]. \tag{5.31}$$

Il y a alors un travail expérimental considérable pour trouver les architectures de réseaux menant à des bonnes performances. Par exemple, la conception du réseau DnCNN [163] (*Denoising Convolutional Neural Network*), outre l'utilisation d'un réseau suffisamment profond, tire profit de l'apprentissage résiduel (le débruiteur étant décomposé en $D(y) = y - R(y)$ où R est un réseau) et d'un procédé de normalisation des couches (*batch normalization*). Le réseau FFDNet [165] (*Fast and Flexible Denoising Convolutional Neural Network*) n'utilise pas d'apprentissage résiduel, mais prend en entrée une carte du niveau de bruit, ce qui permet de traiter des bruits non stationnaires. À noter que certains réseaux débruiteurs [74, 22, 131] sont spécifiques à un niveau de bruit alors que d'autres non [163, 165]. Il nous

sera utile de distinguer les réseaux spécifiques à un niveau de bruit en précisant l'écart-type σ du bruit dans la notation D_σ. Une fois appris, ces débruiteurs sont très peu coûteux en temps de calcul car l'architecture des réseaux convolutionnels est particulièrement bien adaptée au calcul sur GPU. Dans la suite, on parlera de *débruiteur profond* pour désigner une fonction de débruitage D modélisée par un réseau de neurones profond.

On peut étendre cette démarche d'apprentissage profond à la résolution d'un problème inverse (5.1) : on entraîne un réseau de restauration R en minimisant

$$\mathcal{L}(R) = \mathbb{E}_{x\sim p, w\sim\mathcal{N}}\Big[\|R(Ax+w)-x\|^2\Big], \tag{5.32}$$

où $Ax_n + w_n$ est la version dégradée de l'image x_n. Il y a alors différentes manières d'adapter l'architecture du réseau de restauration selon le problème inverse à résoudre [107]. Pour la déconvolution non aveugle, l'architecture de restauration peut exploiter l'opérateur de pseudo-inverse de A, soit en adjoignant à la pseudo-inverse un réseau permettant de corriger les artefacts [133], soit en proposant un réseau dont la forme des couches est inspirée par la décomposition en valeurs singulières de la pseudo-inverse [160]. Pour la super-résolution, Dong *et al.* [42] proposent un réseau à trois couches prenant en entrée l'image sur-échantillonnée à la résolution désirée. On peut mentionner également les approches dites *unrolling* [60] où l'architecture du réseau de restauration est inspirée par des algorithmes de *splitting* avec un nombre d'itérations fixé. L'inconvénient commun de ces approches est qu'elles nécessitent l'apprentissage d'un réseau spécifique au problème inverse considéré. De plus, la dépendance en la base de données peut causer des problèmes de stabilité, notamment dans les contextes où la base d'apprentissage est petite, comme en imagerie médicale.

Une question essentielle est de savoir comment exploiter ces architectures de réseaux de restauration pour construire de nouveaux a priori permettant de traiter, de façon générique, une grande classe de problèmes inverses. Une réponse surprenante a été formulée avec les *Deep Image Priors* de [152], qui consistent à n'utiliser que l'architecture d'un réseau de neurones (non appris) pour paramétrer la restauration. L'image restaurée s'écrit $f_\theta(z)$ où z est un bruit aléatoire fixé, et où f_θ est un réseau de neurones dont on optimise les paramètres θ. Calculer l'image restaurée avec ce *deep image prior* revient alors à la résolution de (5.5) en prenant $\varphi(\theta) = f_\theta(z)$. Ces auteurs montrent que la paramétrisation du problème par la seule structure du réseau permet à l'algorithme de minimisation d'aller vers des images correspondant à des restaurations visuellement satisfaisantes. Cependant, cette méthode directe n'atteint pas les performances de l'état de l'art, et est difficile à contrôler numériquement du fait du peu de régularité de la fonction $\theta \mapsto f_\theta(z)$.

Une autre réponse à la question ci-dessus, plus proche de la modélisation bayésienne, consiste à adopter un prior encodé implicitement par un réseau (ou un algorithme) de débruitage. C'est le point de vue adopté par les méthodes Plug-and-Play décrites dans le paragraphe suivant.

5.3.2 Débruitage et Régularisation, Méthodes Plug-and-Play

Dans ce paragraphe, nous allons voir comment la conception d'un débruiteur performant peut aider à résoudre une large classe de problèmes inverses, en assimilant le débruitage à une étape de régularisation pour un prior implicite $\pi = e^{-\lambda g}$. La clé pour cela est de considérer la résolution du problème (5.7) par les méthodes dites de *splitting*. Par exemple, le *half-quadratic splitting* (HQS) [53] consiste à découpler l'attache aux données et la régularisation en considérant la fonctionnelle augmentée

$$L_\tau(x,z) = f(z) + \frac{1}{2\tau}\|x-z\|^2 + \lambda g(x), \tag{5.33}$$

où $\tau > 0$ est un paramètre voué à tendre vers zéro. L'algorithme HQS est une minimisation alternée de L_τ sur les deux variables x, z. On va l'écrire en utilisant l'opérateur proximal

$$\mathrm{Prox}_f(x) = \underset{z}{\mathrm{Argmin}}\, \frac{1}{2}||x-z||^2 + f(z), \tag{5.34}$$

qui est défini de façon univaluée dès que f est convexe semi-continue inférieurement (s.c.i.) et propre (c'est-à-dire que f n'est pas égale à $+\infty$ partout). Ainsi, l'algorithme HQS s'écrit

$$\begin{cases} z_{k+1} &= \mathrm{Prox}_{\tau f}(x_k) \\ x_{k+1} &= \mathrm{Prox}_{\tau\lambda g}(z_{k+1}) \end{cases}, \tag{5.35}$$

et peut se résumer en l'itération sur la variable x de l'opérateur

$$T_{\mathrm{HQS}} = \mathrm{Prox}_{\tau\lambda g} \circ \mathrm{Prox}_{\tau f}\,. \tag{5.36}$$

L'opérateur proximal (5.34) peut être vu comme une descente de gradient implicite, et a l'avantage de pouvoir être défini dans des cas où la fonction f n'est pas différentiable. Lorsque l'on travaille avec des fonctions différentiables, on peut remplacer l'opérateur proximal par une étape de descente de gradient explicite. Lorsque f est différentiable, on peut donc aussi considérer l'algorithme de descente de gradient proximale (PGD), aussi appelé *Forward-Backward*, qui s'écrit

$$\begin{cases} z_{k+1} &= (\mathrm{Id} - \tau\nabla f)(x_k) \\ x_{k+1} &= \mathrm{Prox}_{\tau\lambda g}(z_{k+1}) \end{cases}, \tag{5.37}$$

et peut se résumer en l'itération de l'opérateur

$$T_{\mathrm{PGD}} = \mathrm{Prox}_{\tau\lambda g} \circ (\mathrm{Id} - \tau\nabla f). \tag{5.38}$$

Le paramètre τ utilisé dans ces algorithmes sera parfois appelé le pas (*step size*).

Deux autres méthodes de *splitting* (ADMM et DRS) seront rappelées dans la Section 5.3.4, correspondant aux opérateurs T_{ADMM} et T_{DRS}. Sous certaines hypothèses sur f, g, on peut voir que les points fixes de l'opérateur T_{PGD} coïncident avec les minimiseurs de la fonction $F = f + \lambda g$, et que pour certaines valeurs de τ, λ,

l'algorithme PGD converge vers l'un de ces minimiseurs [34, 33]. Quant à lui, l'algorithme HQS cible en fait les minimiseurs de la fonction $\tilde{F}(x) = \inf_z L_\tau(x, z)$ [11, 33], qui se rapprochent de ceux de F lorsque $\tau \longrightarrow 0$.

L'idée au cœur de la méthodologie PnP est de remarquer que l'étape de régularisation $\text{Prox}_{\tau\lambda g}$ est un simple débruitage pour la régularisation $\tau\lambda g$, à quoi l'on peut substituer l'application d'un algorithme de débruitage des plus performants. Ceci permet de formuler un algorithme de résolution de problème inverse dans lequel le prior n'apparaît que de manière implicite au travers du débruitage utilisé. Cette méthodologie a été explicitée et validée expérimentalement sur un problème de tomographie par Venkatakrishnan *et al.* [156], en se basant sur l'algorithme ADMM et en testant plusieurs débruiteurs (débruitage TV, BM3D, KSVD). Sur le plan expérimental, d'autres succès ont ensuite été obtenus en suivant cette méthodologie, parfois avec des débruiteurs classiques comme BM3D [24, 78], NL-means [115], ou un débruiteur GMM [144], parfois avec des débruiteurs profonds [109, 128]. Les meilleurs résultats visuels ont finalement été obtenus en utilisant des débruiteurs profonds qui sont spécifiquement conçus pour s'adapter aux méthodes PnP comme dans [164] (réseau IRCNN) ou [162] (réseau DRUNET).

Le succès expérimental des méthodes PnP soulève une question cruciale : comment peut-on interpréter le résultat de la restauration sans garantie de convergence de l'algorithme itératif utilisé ? En pratique, il est en effet utile de disposer de critères numériques indiquant que l'algorithme s'est bien stabilisé, ou au contraire qu'il est nécessaire de l'itérer plus longuement. Plusieurs auteurs ont formulé des résultats de convergence d'algorithmes PnP, avec divers jeux d'hypothèses sur le débruiteur, l'attache aux données étant en général supposée différentiable à gradient Lipschitz. Sreehari *et al.* [139] reviennent à un résultat de convergence pour ADMM en écrivant le débruiteur comme l'opérateur proximal d'une fonction convexe, dont on sait qu'il est non-expansif, c'est-à-dire 1-Lipschitz. Chan *et al.* [24] et Gavaskar et Chaudhury [52] construisent une preuve ad-hoc de convergence de l'algorithme PnP-ADMM en supposant $\text{Id} - D_\sigma$ borné proportionnellement à σ^2. En supposant la forte convexité de l'attache aux données f, Ryu *et al.* [128] montrent que T_{PGD} est contractant, avec l'hypothèse que $\text{Id} - D$ est Lipschitz. Enfin, les auteurs de [141, 142] obtiennent la convergence de variantes de ces schémas en les exprimant comme des itérations de Krasnosel'skii-Mann [10], sous l'hypothèse que le débruiteur appartient à la classe des opérateurs *averaged*. On remarquera que toutes ces méthodes reposent sur une analyse de la constante de Lipschitz de l'opérateur itéré, en exploitant des résultats sur les compositions d'opérateurs Lipschitz ou *averaged*.

Cependant, même si ces méthodes PnP ont permis d'atteindre des performances à l'état de l'art sur de nombreux problèmes inverses, leur contrôle numérique reste délicat puisqu'elles ne peuvent a priori pas être reliées à la minimisation d'une fonctionnelle explicite. De plus, les garanties de convergence reposent sur des hypothèses de bornes ou de constantes de Lipschitz sur le débruiteur qui ne sont généralement pas vérifiées par les débruiteurs les plus utilisés. On notera néanmoins qu'il est possible [128, 145] de pousser le débruiteur appris à satisfaire ces hypothèses (au moins approximativement), mais contraindre fortement la constante de Lipschitz réduit la qualité du débruiteur [15, 69].

5.3.3 Plug-and-Play avec Débruiteur par Pas de Gradient

Dans ce paragraphe, nous allons montrer qu'en se restreignant à une classe de débruiteurs particuliers, l'algorithme PnP peut de nouveau être lié à la minimisation d'une fonctionnelle explicite, ce qui permettra de retrouver un contrôle numérique précis tout en bénéficiant des performances en débruitage des réseaux de neurones profonds. Les travaux de ce paragraphe font l'objet de la première partie de la thèse de Samuel Hurault [C11].

5.3.3.1 Régularisation et débruiteur par pas de gradient

Comme on l'a dit au-dessus, le principe des méthodes PnP est de substituer un débruiteur pré-conçu à l'étape de régularisation d'un algorithme de *splitting*. Mais si l'on veut exprimer une fonctionnelle minimisée par l'algorithme, il faut revenir en arrière et trouver une fonction de régularisation associée au débruiteur. Pour cela, Romano *et al.* [126] ont proposé la méthodologie *Regularization by denoising* (RED), qui consiste, en partant d'un débruiteur D, à considérer la fonction de régularisation

$$g_{\text{RED}}(x) = \frac{1}{2}\langle x, x - D(x)\rangle. \tag{5.39}$$

Sous l'hypothèse que D est localement homogène [126] et différentiable à jacobien symétrique, on peut montrer [123] que $\nabla g_{\text{RED}} = \text{Id} - D$. Cette expression a été utilisée dans [126] pour rapprocher les algorithmes PnP de la minimisation de la fonction $f + \lambda g_{\text{RED}}$, avec des mises-à-jour pouvant se calculer directement avec D. Là encore, les algorithmes obtenus avec cette interprétation produisent de très bons résultats sur plusieurs problèmes inverses. Cependant, on peut constater expérimentalement que la plupart des débruiteurs répandus (en particulier BM3D et NL-means) ne satisfont pas les hypothèses requises.

Nous avons proposé de considérer la régularisation $g_\sigma : \mathbb{R}^n \longrightarrow \mathbb{R}$ définie par

$$g_\sigma(x) = \frac{1}{2}||x - N_\sigma(x)||^2, \tag{5.40}$$

où $N_\sigma : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ est un réseau de neurones différentiable. Cette régularisation induit un débruiteur particulier sous la forme d'un "pas de gradient" $D_\sigma = \text{Id} - \nabla g_\sigma$, c'est-à-dire

$$D_\sigma(x) = x - \nabla g_\sigma(x) = N_\sigma(x) + J_{N_\sigma}(x)^T(x - N_\sigma(x)). \tag{5.41}$$

On le notera GS pour *Gradient-Step*. Contrairement à l'approche RED, la symétrie du jacobien de D_σ (lorsqu'il existe) est maintenant évidente puisque D_σ est un champ de gradients. Ce débruiteur est entraîné de façon à minimiser la fonction de coût

$$\mathcal{L}(D_\sigma) = \mathbb{E}_{x\sim p, w_\sigma \sim \mathcal{N}(0,\sigma^2 I)}[||D_\sigma(x + w_\sigma) - x||^2], \tag{5.42}$$

où p est la distribution empirique associée à la base de données d'images propres.

Nous allons étudier le schéma itératif

$$x_{k+1} = T^{\tau,\lambda}_{\text{GS-PnP}}(x_k) \tag{5.43}$$

basé sur l'opérateur

$$\begin{aligned} T^{\tau,\lambda}_{\text{GS-PnP}} &= \text{Prox}_{\tau f} \circ (\tau\lambda D_\sigma + (1-\tau\lambda)\,\text{Id}) \\ &= \text{Prox}_{\tau f} \circ (\text{Id} - \tau\lambda \nabla g_\sigma). \end{aligned} \tag{5.44}$$

Le schéma (5.43) correspond à l'algorithme PGD appliqué à la fonctionnelle $F = f + \lambda g_\sigma$ avec pas τ. Ainsi on retombe sur un vrai problème de minimisation avec une régularisation explicite, ce qui nous permettra d'obtenir les garanties de convergence énoncées ci-dessous.

Notons au passage que les opérateurs de descente de gradient (implicite et explicite) sur l'attache aux données $f(x) = \frac{1}{2}\|Ax - y\|^2$ s'écrivent

$$\text{Id} - \tau\nabla f(x) = (\text{Id} - \tau A^T A)x + \tau A^T y, \tag{5.45}$$

$$\text{Prox}_{\tau f}(x) = (\text{Id} + \tau A^T A)^{-1}(\tau A^T y + x). \tag{5.46}$$

Selon la forme de A, il pourra être plus aisé ou pertinent d'utiliser l'un ou l'autre de ces opérateurs de descente, et ce sera souvent cela qui guidera le choix de l'algorithme de *splitting* utilisé.

Avant de donner les garanties de convergence, formulons quelques remarques sur la forme du débruiteur. Avec les formules (5.40) et (5.41), on voit que la régularisation et le débruiteur sont tous deux paramétrés par un réseau différentiable quelconque $N_\sigma : \mathbb{R}^n \longrightarrow \mathbb{R}^n$. Par conséquent, on peut construire N_σ en tirant profit des architectures de réseaux de neurones ayant mené aux meilleurs résultats possibles en débruitage, à la différence près que l'on imposera ici la différentiabilité du réseau. Par ailleurs, remarquons que le débruiteur réalisant l'erreur quadratique minimale (MMSE) est défini par

$$D^*_\sigma = \underset{D}{\text{Argmin}}\, \mathbb{E}_{x\sim p, w_\sigma \sim \mathcal{N}(0,\sigma^2 I)}[||D(x + w_\sigma) - x||^2], \tag{5.47}$$

où le minimum est pris sur toutes les fonctions mesurables ($D^*_\sigma(x)$ est alors l'espérance conditionnelle de x sachant $x + w_\sigma$). On peut montrer que ce débruiteur MMSE vérifie la formule de Tweedie [46]

$$D^*_\sigma = \text{Id} - \nabla g^*_\sigma \quad \text{où} \quad g^*_\sigma = -\sigma^2 \log p_\sigma \tag{5.48}$$

où p_σ est la convolution de p avec la densité gaussienne $\mathcal{N}(0, \sigma^2)$. Approcher le débruiteur optimal D^*_σ par notre paramétrisation $D_\sigma = \text{Id} - \nabla g_\sigma$ revient donc à ajuster la régularisation g_σ de telle sorte que $\nabla g_\sigma \approx -\sigma^2 \nabla \log p_\sigma$. Ceci rejoint les approches dites de *score matching* [132, 14].

5.3.3.2 Convergence de l'algorithme GS-PnP

Énonçons maintenant les résultats de convergence vérifiés par l'algorithme

$$x_{k+1} = T(x_k) \quad \text{avec} \quad T = \text{Prox}_{\tau f} \circ (\text{Id} - \tau\lambda\nabla g), \tag{5.49}$$

pour la minimisation de la fonction $F = f + \lambda g$ définie sur $\mathbb{R}^n$. Nous formulons le résultat avec peu d'hypothèses sur f, et en particulier, sans supposer la continuité,

ce qui permet de contraindre l'optimisation à un domaine convexe. Quant à la fonction g, elle est supposée différentiable à gradient Lipschitz, ce qui permet d'englober les régularisations g_σ rencontrées dans le paragraphe précédent. Le premier résultat fournit la convergence en termes de valeurs de la fonctionnelle F.

Théorème 5.3.1. *Soient $f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$ et $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ deux fonctions propres semi-continues inférieurement. Soient $\lambda > 0$ et $F = f + \lambda g$. On suppose que f est convexe, que g est différentiable à gradient L-Lipschitz, et que $\inf F > -\infty$. Alors, pour $\tau < \frac{1}{\lambda L}$, la suite (x_k) vérifiant (5.49) vérifie*

(i) $(F(x_k))$ est décroissante et converge.

(ii) Le résidu $||x_{k+1} - x_k||$ tend vers 0.

(iii) Toute valeur d'adhérence de (x_k) est un point critique de F.

La preuve, inspirée par [12] consiste à utiliser des lemmes de descente sur la fonction d'enveloppe

$$Q_\tau(x, y) = g(y) + \langle x - y, \nabla g(y)\rangle + \frac{1}{2\tau}||x - y||^2 + f(x). \tag{5.50}$$

Cette preuve donne aussi une vitesse de convergence sur les résidus, à savoir

$$\min_{0\leqslant i\leqslant k} \|x_{i+1} - x_i\|^2 \leqslant \frac{1}{k} \frac{F(x_0) - \lim F(x_i)}{\frac{1}{2\tau} - \frac{L}{2}}. \tag{5.51}$$

Le résultat suivant donne la convergence des itérés (x_k), pour peu que la suite (x_k) soit bornée et que F vérifie la condition de Kurdika-Łojasiewicz, dont on pourra trouver la définition précise dans [3].

Théorème 5.3.2. *Soient $f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$ et $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ deux fonctions propres semi-continues inférieurement. Soient $\lambda > 0$ et $F = f + \lambda g$. On suppose que f est convexe, que g est différentiable à gradient L-Lipschitz, et que $\inf F > -\infty$. On suppose de plus que F vérifie la condition de Kurdika-Łojasiewicz et que la suite (x_k) donnée par (5.49) est bornée. Alors, pour $\tau < \frac{1}{\lambda L}$, la suite (x_k) converge, vers un point critique x^* de F.*

Ce théorème est une conséquence de [3, Theorem 5.1]. Les hypothèses sur f et g sont bien vérifiées pour nos problèmes inverses avec la régularisation g_σ donnée par (5.40). De plus, grâce à la décroissance de $(F(x_k))$, la suite (x_k) est bornée dès que F est coercive. Dans notre cas pratique où les images sont à valeurs dans $[0, 1]$, on peut aisément modifier g_σ de façon à ce que F soit coercive, en prenant par exemple

$$\tilde{g}_\sigma(x) = g_\sigma(x) + \frac{1}{2}\|x - \Pi_C(x)\|^2 \tag{5.52}$$

où Π_C est la projection orthogonale sur un convexe compact $C \subset \mathbb{R}^n$ (e.g. $[-1, 2]^n$).

Le défaut principal de ces deux théorèmes est qu'en général, la constante de Lipschitz L n'est pas connue explicitement. Pour éviter ce problème, on met en place une procédure de réglage automatique des pas, dite de *backtracking* permettant de garder le pas τ maximal tout en ayant une garantie suffisante de décroissance sur F.

Param.: initialisation z_0, $\lambda > 0$, $\sigma \geqslant 0$, $\epsilon > 0$, $\tau_0 > 0$, $K \in \mathbb{N}^*$, $\eta \in (0,1)$, $\gamma \in (0, 1/2)$.
Input : image dégradée y.
Output: image restorée $\hat{x}$.
$k = 0$; $x_0 = \text{Prox}_{\tau f}(z_0)$; $\tau = \tau_0/\eta$; $\Delta > \epsilon$;
while $k < K$ *and* $\Delta > \epsilon$ **do**
  $z_k = \lambda\tau D_\sigma(x_k) + (1 - \lambda\tau)x_k$;
  $x_{k+1} = \text{Prox}_{\tau f}(z_k)$;
  if $F(x_k) - F(x_{k+1}) < \frac{\gamma}{\tau}||x_k - x_{k+1}||^2$;
  then $\tau = \eta\tau$;
  else $\Delta = \frac{F(x_k) - F(x_{k+1})}{F(x_0)}$; $k = k + 1$;
end
$\hat{x} = \lambda\tau D_\sigma(x_K) + (1 - \lambda\tau)x_K$;

Algorithm 7: Gradient-Step Plug-and-Play (GS-PnP)

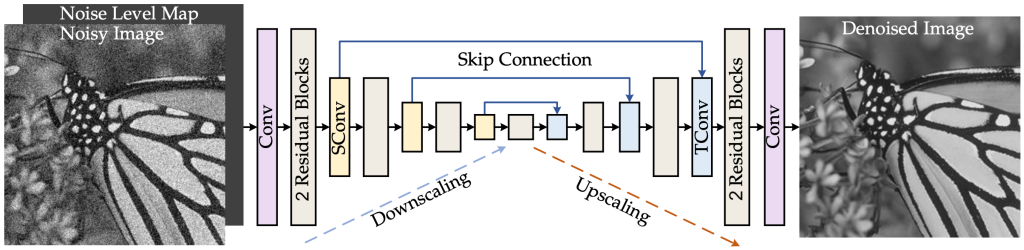


FIGURE 5.1 – Architecture DRUNet simplifiée [162] paramétrant N_σ.

Nous avons montré que cette étape de *backtracking* ne modifie pas la garantie de convergence. L'algorithme PnP ainsi obtenu avec le débruiteur GS est résumé dans l'Algorithme 7. Afin d'améliorer légèrement les performances visuelles, on a intégré à l'algorithme une dernière étape de débruitage appliquant Id $-\tau\lambda\nabla g_\sigma$. Ceci permet d'éliminer le bruit résiduel qui apparaît après l'étape proximale $\text{Prox}_{\tau f}$.

5.3.3.3 Performance du débruiteur GS-DRUNet

Le débruiteur D_σ est paramétré ici par un réseau de neurones différentiable N_σ via l'équation (5.41), et on l'entraîne en minimisant la fonction de coût (5.42). Pour le réseau de neurones N_σ, on s'inspire de l'architecture DRUNet proposée dans [162], qui est un réseau U-Net intégrant quatre blocs résiduels à chaque échelle. Pour réduire le temps de calcul, on adopte ici une version légèrement simplifiée intégrant deux blocs résiduels à chaque échelle au lieu de quatre, comme représentée dans la Fig 5.1. Pour maintenir la différentiabilité, on change la fonction de non-linéarité non-lisse RELU par une version différentiable ELU. Dans l'Appendice B de [C11], nous avons montré que la régularisation induite g_σ est bien différentiable à gradient Lipschitz, rendant légitime l'application des résultats de la Partie 5.3.3.2.

σ(./255)	5	15	25	50	Time (ms)
FFDNet	39.95	33.53	30.84	27.54	1.9
DnCNN	39.80	33.55	30.87	27.52	2.3
DRUNet	**40.31**	**33.97**	**31.32**	**28.08**	69.8
GS-DRUNet	<u>40.26</u>	<u>33.90</u>	<u>31.26</u>	<u>28.01</u>	10.4

TABLE 5.1 – Performances (mesurées en PSNR moyen) sur la base de données CBSD68 [106] du débruiteur GS-DRUNet, en comparaison de plusieurs débruiteurs récents : FFDNet [165], DnCNN [163], et DRUNet [162]. Notre débruiteur GS-DRUNet atteint presque les performances de DRUNet.

Le réseau débruiteur D_σ ainsi obtenu est noté GS-DRUNet. Ce réseau D_σ prend le niveau de bruit σ en entrée, et on l'entraîne sur la même base de données que celle utilisée dans [162] : la distribution empirique p est formée de patches 128×128 d'images propres auxquels on rajoute des bruits blancs gaussiens w_σ d'écart-type σ choisi aléatoirement dans $[0, \frac{50}{255}]$. En pratique, pour calculer $D_\sigma(x)$, on utilise la relation $D_\sigma = \mathrm{Id} - \nabla g_\sigma$: on calcule $g_\sigma(x)$ et on en déduit $\nabla g_\sigma(x)$ grâce aux outils de différentiation automatique de Pytorch. Pour l'entraînement du réseau, la fonction de coût (5.42) est minimisée via l'algorithme ADAM [79], qui nécessite de calculer le gradient de $\mathcal{L}$ par rapport aux paramètres du réseau. On profite donc ici de la possibilité d'une double différentiation automatique en Pytorch, d'abord par rapport à x, puis par rapport aux paramètres du réseau.

Sur la Table 5.1, on reporte les performances atteintes par le débruiteur GS-DRUNet sur les images 256×256 de la base de données CBSD68. On constate que GS-DRUNet atteint à 0.05dB près le même niveau de performance que DRUNet [162], tout en étant sept fois plus rapide à appliquer, et en satisfaisant intrinsèquement la contrainte d'être un vrai champ de gradients. Il reste aussi nettement plus performant que DRUNet [163] et FFDNet [165].

5.3.3.4 Performance de l'algorithme GS-PnP

Décrivons maintenant les performances atteintes par l'Algorithme 7 GS-PnP appliqué à différents problèmes inverses. On considère ici le modèle d'observation linéaire $y = Ax + w$ où w est un bruit blanc gaussien d'écart-type $\nu > 0$. Le terme d'attache aux données s'écrit donc $\frac{1}{2\nu^2}\|Ax - y\|^2$, et l'on adopte la régularisation g_σ associée au débruiteur GS-DRUNet via l'équation (5.40). En intégrant le niveau de bruit ν dans le paramètre de régularisation $\lambda_\nu = \lambda\nu^2$ (avec $\lambda > 0$ fixé), on aboutit à la fonctionnelle

$$F(x) = f(x) + \lambda_\nu g_\sigma(x) \quad \text{où} \quad \begin{cases} f(x) & = \frac{1}{2}\|Ax - y\|^2 \\ g_\sigma(x) & = \frac{1}{2}\|N_\sigma(x) - x\|^2 \end{cases}. \tag{5.53}$$

On notera bien que le niveau de régularisation dépend de $\lambda_\nu = \lambda\nu^2$ mais aussi du paramètre $\sigma > 0$ qui n'a a priori pas de raison d'être égal à ν. En testant sur une grande base d'images, on évalue empiriquement que la constante de Lipschitz L de g_σ est proche de 1 (mais pas $\leqslant 1$). Par conséquent, on fixe le pas de temps

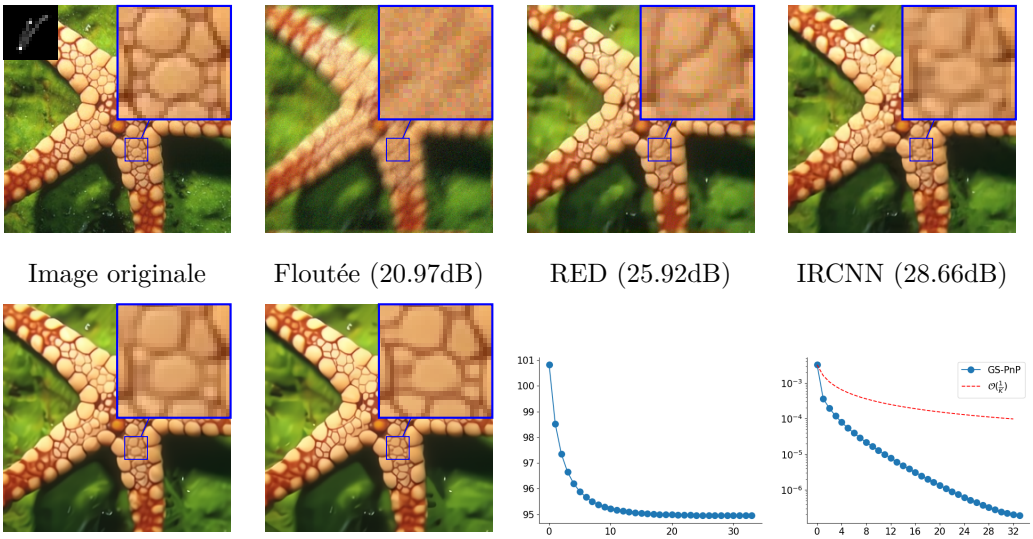


FIGURE 5.2 – Comparaison de résultats de déflouage obtenus avec plusieurs algorithmes de restauration appliqués à une image floutée (avec le noyau montré en haut à gauche) et bruitée ($\nu = 0.03$). Les diagrammes montrent l'évolution de $F(x_k)$ et de $\gamma_k = \min_{0 \leqslant i \leqslant k} ||x_{i+1} - x_i||^2 / ||x_0||^2$ au cours des itérations de l'algorithme GS-PnP. On constate que GS-PnP atteint ici la meilleure performance visuelle, et les diagrammes confirment la convergence de l'algorithme prédite par les résultats théoriques.

		(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	
ν	Méthode											*Moy.*
0.01	EPLL	28.32	28.24	28.36	25.80	29.61	27.15	26.90	26.69	25.84	26.49	*27.34*
	RED	30.47	30.01	30.29	28.09	31.22	28.92	28.90	28.67	26.66	28.45	*29.17*
	IRCNN	32.96	32.62	32.53	32.44	33.51	33.62	32.54	32.20	28.11	29.19	*31.97*
	MMO	32.35	32.06	32.24	31.67	31.77	33.17	32.30	31.80	27.81	**29.26**	*31.44*
	DPIR	**33.76**	**33.30**	**33.04**	**33.09**	**34.10**	**34.34**	**33.06**	**32.77**	**28.34**	29.16	***32.50***
	GS-PnP	33.52	33.07	32.91	32.83	34.07	34.25	32.96	32.54	28.11	29.03	*32.33*
0.03	EPLL	25.31	25.12	25.82	23.75	26.99	25.23	25.00	24.59	24.34	25.43	*25.16*
	RED	25.71	25.32	25.71	24.38	26.65	25.50	25.27	24.99	23.51	25.54	*25.26*
	IRCNN	28.96	28.65	28.90	28.38	30.03	29.87	28.92	28.52	25.92	27.64	*28.58*
	IRCNN	28.96	28.65	28.90	28.38	30.03	29.87	28.92	28.52	25.92	27.64	*28.58*
	DPIR	**29.38**	**29.06**	**29.21**	**28.77**	30.22	**30.23**	**29.34**	28.90	26.19	27.81	***28.91***
	GS-PnP	29.22	28.89	29.20	28.60	**30.32**	30.21	29.32	**28.92**	**26.38**	**27.89**	*28.90*
0.05	EPLL	24.08	23.91	24.78	22.57	25.68	23.98	23.70	23.19	23.75	24.78	*24.04*
	RED	22.78	22.54	23.13	21.92	23.78	22.97	22.89	22.67	22.01	23.78	*22.84*
	IRCNN	27.00	26.74	27.25	26.37	28.29	28.06	27.22	26.81	24.85	26.83	*26.94*
	DPIR	**27.52**	**27.35**	**27.73**	**27.02**	28.63	**28.46**	27.79	27.30	25.25	27.11	*27.42*
	GS-PnP	27.45	27.28	27.70	26.98	**28.68**	28.44	**27.81**	**27.38**	**25.49**	**27.15**	***27.44***

TABLE 5.2 – Comparaison en PSNR(dB) de méthodes de déflouage sur la base CBSD10, avec plusieurs noyaux de flou (première ligne) et avec plusieurs niveaux de bruits ν (première colonne). Pour chaque colonne, le meilleur résultat est en gras, et le deuxième souligné. La dernière colonne donne les PSNR moyens sur la base.

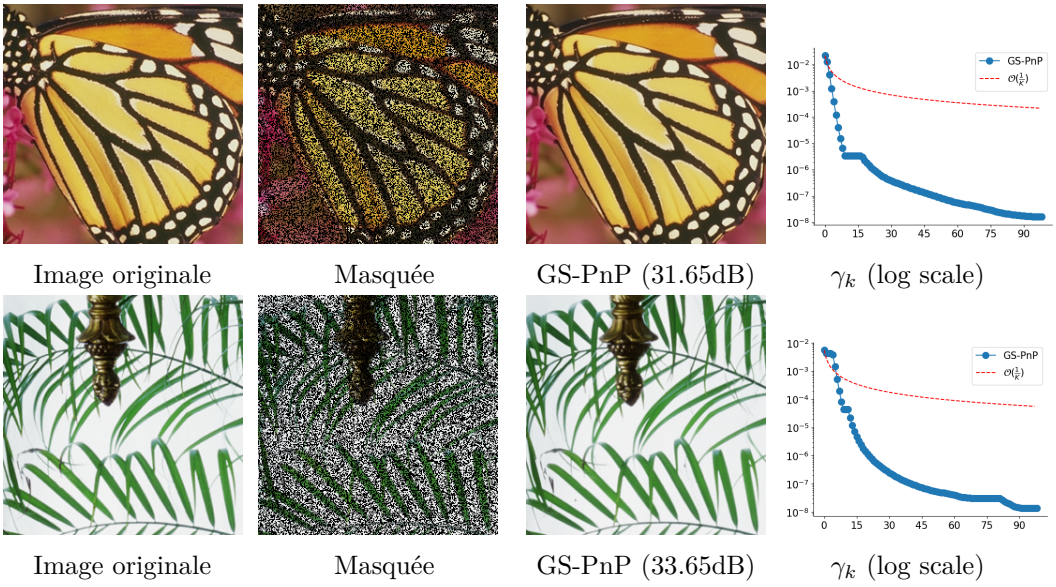


FIGURE 5.3 – Comparaison de résultats d'inpainting obtenus avec plusieurs algorithmes de restauration appliqués à trois images de Set3C dont on a masqué aléatoirement les pixels avec probabilité $p = 0.5$. La dernière colonne montre l'évolution de $\gamma_k = \min_{0 \leqslant i \leqslant k} ||x_{i+1} - x_i||^2 / ||x_0||^2$ au cours des itérations de l'algorithme GS-PnP.

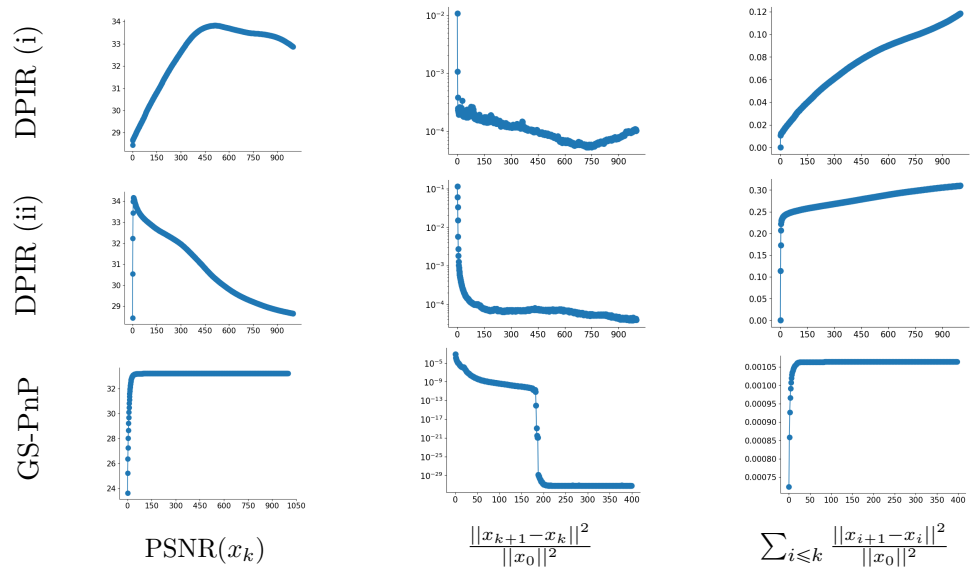


FIGURE 5.4 – Ces diagrammes illustrent le comportement (en termes de PSNR, et de résidu $\|x_{k+1} - x_k\|^2$) des algorithmes DPIR [162] et GS-PnP, sur le cas de déflouage montré dans la Fig. 5.2. On illustre le comportement de DPIR avec deux stratégies de décroissance de σ : soit σ décroît constamment en 1000 itérations (ligne 1), soit il décroît sur les 8 premières itérations puis est laissé fixe (ligne 2). Ces diagrammes confirment les résultats théoriques de convergence obtenus pour GS-PnP, alors que l'algorithme DPIR ne semble pas se stabiliser.

initial $\tau_0 = \frac{1}{\nu^2\lambda}$, ce qui permet d'avoir (avec éventuellement un petit nombre d'étapes de *backtracking*) la condition $\tau < \frac{1}{L\lambda\nu^2}$, assurant la convergence de l'algorithme grâce aux résultats de la Section 5.3.3.2.

On va illustrer les performances de GS-PnP en déflouage et en inpainting, et l'on renvoie à [C11] pour d'autres expériences. Pour adapter l'algorithme GS-PnP à l'un de ces problèmes inverses, il convient de calculer l'opérateur $\mathrm{Prox}_{\tau f}$ associé à l'attache aux données. Pour le déflouage (non aveugle), l'opérateur de dégradation $A = H$ est une convolution avec un noyau connu effectuée avec conditions de bords périodiques, et l'opérateur proximal admet une expression explicite dans le domaine de Fourier. Pour l'inpainting sans bruit, A est une matrice diagonale à valeurs dans $\{0, 1\}$, et on utilise pour attache aux données l'indicatrice f de l'ensemble convexe $\{Ax = y\}$: $f(x) = 0$ si $Ax = y$ et $+\infty$ sinon. Ce cas illustre la possibilité d'avoir une attache aux données discontinue, puisque l'algorithme ne fait appel à f qu'au travers de son opérateur proximal (qui est la projection orthogonale sur $\{Ax = y\}$).

Les résultats quantitatifs sont calculés en testant l'algorithme sur la base CBSD10 formée de 10 images extraites de la base CBSD68, ainsi que sur trois images de la base Set3C (*butterfly, leaves, starfish*). On compare nos résultats avec ceux obtenus avec plusieurs méthodes récentes de restauration d'images : la méthode par patches EPLL [168, 72], les méthodes PnP avec débruiteurs profonds IRCNN [164], DPIR [162] et MMO [116], ainsi que la méthode RED [126] (précisément "RED-fixed point" appliquée avec le débruiteur TRND de [27]). Parmi ces méthodes, seule MMO garantit la convergence de l'algorithme PnP en se basant sur un débruiteur entraîné de façon à contraindre sa constante de Lipschitz. À l'inverse, les méthodes IRCNN et DPIR, basées sur HQS, n'ont pas de garantie de convergence, et sont implémentées avec peu d'itérations et une décroissance rapide des paramètres τ et σ. Les paramètres de GS-PnP ont été ajustés sur chaque problème inverse.

Les résultats pour le déflouage sont donnés en Fig 5.2 et dans la Table 5.2, et pour l'inpainting en Fig. 5.3. Sur les tableaux de PSNR, on constate que GS-PnP atteint des performances analogues à celles de DPIR [162], et surpasse ainsi les autres méthodes comparées. Sur les résultats qualitatifs, l'algorithme GS-PnP parvient à restaurer des structures géométriques propres de façon au moins aussi nette que DPIR. De plus, les diagrammes de convergence confirment les résultats théoriques de la Partie 5.3.3.2. À l'inverse, on peut constater sur la Fig. 5.4 que DPIR ne semble pas converger. Au passage, on a remarqué qu'au cours de l'algorithme GS-PnP, le débruiteur D_σ passe par des images x_k pour lesquelles $\frac{\|D_\sigma(x_{k+1}) - D_\sigma(x_k)\|}{\|x_{k+1} - x_k\|}$ est légèrement supérieur à 1. Par conséquent, le débruiteur D_σ n'est pas 1-Lipschitz, ce qui illustre l'intérêt de ne pas avoir posé l'hypothèse de non-expansivité dans nos résultats de convergence. Mentionnons pour terminer que l'influence des paramètres de régularisation σ et λ a été étudiée dans [C11, Appendice J.5], et on a tiré un choix de paramètres générique fournissant de bons résultats pour les problèmes inverses considérés. En particulier, il faut veiller à ne pas prendre σ plus petit que le niveau de bruit ν de façon à ce que le débruiteur D_σ puisse atténuer suffisamment le bruit rajouté à chaque itération par l'étape proximale $\mathrm{Prox}_{\tau f}$.

5.3.4 Plug-and-Play avec Débruiteur Proximal

L'approche présentée dans la section précédente donne une garantie de convergence sur l'algorithme de descente de gradient proximale (5.49), alors la méthodologie PnP s'applique aussi à d'autres algorithmes de *splitting*. Dans ce paragraphe, nous allons voir que, sous certaines conditions, le débruiteur GS (5.41) peut s'écrire comme l'opérateur proximal d'une certaine fonction ϕ_σ constituant une nouvelle régularisation possible pour le problème inverse. En utilisant ce débruiteur proximal, on peut donner des garanties de convergence de l'algorithme PnP avec différents schémas de *splitting*. Sur le plan expérimental, la contrainte portant sur le débruiteur proximal impose une légère baisse de performance par rapport au débruiteur GS, mais la qualité visuelle des restaurations obtenues avec ces nouveaux algorithmes PnP est similaire à celle des résultats de GS-PnP. Les travaux de ce paragraphe constituent la deuxième partie de la thèse de Samuel Hurault [C12].

5.3.4.1 Algorithmes ADMM et DRS

Commençons par rappeler la formulation des autres méthodes de *splitting* considérées dans cette section. L'algorithme *Alternating Direction Method of Multipliers* (ADMM) [19] pour la minimisation de (5.7) est basé sur la considération du lagragien augmenté défini par

$$\forall x, y, z \in \mathbb{R}^n, \quad \mathcal{L}_\tau(x, y, z) = f(y) + \lambda g(z) + \frac{1}{\tau}\langle x, z - y\rangle + \frac{1}{2\tau}||z - y||^2, \tag{5.54}$$

où $\tau > 0$ est un paramètre. L'algorithme ADMM consiste à alterner des étapes de minimisation en y, z et un pas de gradient en x :

$$\begin{aligned} z_{k+1} &= \operatorname*{Argmin}_z \mathcal{L}_\tau(x_k, y_k, z) = \operatorname{Prox}_{\tau\lambda g}(y_k - x_k) \\ y_{k+1} &= \operatorname*{Argmin}_y \mathcal{L}_\tau(x_k, y, z_{k+1}) = \operatorname{Prox}_{\tau f}(z_{k+1} + x_k) \\ x_{k+1} &= x_k + (z_{k+1} - y_{k+1}), \end{aligned} \tag{5.55}$$

en supposant que les opérateurs proximaux sont bien définis.

L'algorithme de Douglas-Rachford (DRS) [45, 97] quant à lui, s'écrit

$$\begin{aligned} y_{k+1} &= \operatorname{Prox}_{\tau f}(x_k) \\ z_{k+1} &= \operatorname{Prox}_{\tau\lambda g}(2y_{k+1} - x_k) \\ x_{k+1} &= x_k + \mu(z_{k+1} - y_{k+1}). \end{aligned} \tag{5.56}$$

où $\mu \in (0, 2)$ et $\tau > 0$ sont des paramètres. En introduisant

$$T_{\mathrm{DRS}} = \operatorname{Rprox}_{\tau\lambda g} \circ \operatorname{Rprox}_{\tau f} \quad \text{où} \quad \operatorname{Rprox}_f = 2\operatorname{Prox}_f - \mathrm{Id}, \tag{5.57}$$

on peut le réécrire sous la forme

$$x_{k+1} = x_k + \frac{\mu}{2}(T_{\mathrm{DRS}}(x_k) - x_k), \quad y_{k+1} = \operatorname{Prox}_{\tau f}(x_k). \tag{5.58}$$

À changement de variables ($\tilde{x}_k = x_k + z_k$) et d'indices près, ADMM est un cas particulier de DRS pour $\mu = 1$, et il nous suffira donc d'étudier l'algorithme DRS.

5.3.4.2 Algorithmes PnP avec Débruiteur Proximal

D'abord, pour conserver le paramètre de régularisation dans l'algorithme PnP associé à ces nouveaux schémas de *splitting*, on doit diviser la fonctionnelle à optimiser par λ, ce qui mène au *splitting* $\frac{1}{\lambda}F(x) = \frac{1}{\lambda}f(x) + g(x)$. Le paramètre λ interviendra donc maintenant dans l'opérateur proximal de f, et non de g.

En remplaçant à nouveau l'opérateur proximal $\mathrm{Prox}_{\tau g}$ par un débruiteur D_σ dans les algorithmes de *splitting* ci-dessus, on obtient l'algorithme PnP-ADMM :

$$\begin{cases} z_{k+1} = D_\sigma(y_k - x_k) \\ y_{k+1} = \mathrm{Prox}_{\tau\frac{1}{\lambda}f}(z_{k+1} + x_k) \\ x_{k+1} = x_k + (z_{k+1} - y_{k+1}) \end{cases} \tag{5.59}$$

et l'algorithme PnP-DRS avec paramètre $\beta \in [0, 2]$:

$$\begin{cases} y_{k+1} &= \mathrm{Prox}_{\tau\frac{1}{\lambda}f}(x_k) \\ z_{k+1} &= D_\sigma(2y_{k+1} - x_k) \\ x_{k+1} &= x_k + \beta(z_{k+1} - y_{k+1}) \end{cases} \tag{5.60}$$

Lorsque $\beta = 1$, l'algorithme PnP-DRS revient à itérer l'opérateur

$$T_{\text{PnP-DRS}} = (2D_\sigma - \mathrm{Id}) \circ \mathrm{Rprox}_{\tau\frac{1}{\lambda}f}\,. \tag{5.61}$$

Les auteurs de [95] et [146] ont donné des preuves de convergence de l'algorithme DRS (et donc aussi d'ADMM) pour minimiser la somme de deux fonctions dont l'une est différentiable à gradient Lipschitz. Nous allons appliquer ces résultats aux algorithmes PnP-ADMM et PnP-DRS obtenus à partir du débruiteur GS (5.41). Pour cela, nous allons voir qu'en supposant $\mathrm{Id} - D_\sigma$ contractant, ce débruiteur peut s'écrire comme l'opérateur proximal associé à une certaine fonction non-convexe.

Proposition 5.3.3. *Soit $\mathcal{X} \subset \mathbb{R}^n$ ouvert convexe. Soit une fonction $g : \mathcal{X} \longrightarrow \mathbb{R}$ de classe $\mathscr{C}^{k+1}$, $k \geqslant 1$. Supposons que ∇g est L-Lipschitz avec $L < 1$. De plus, notons*

$$h : x \mapsto \frac{1}{2}||x||^2 - g(x) \quad \text{et} \quad D = \nabla h = \mathrm{Id} - \nabla g. \tag{5.62}$$

On a alors les propriétés suivantes.

(i) h est $(1 - L)$-fortement convexe

(ii) D est injectif, $\mathrm{Im}(D)$ est ouverte et $\forall x \in \mathcal{X}$, $D(x) = \mathrm{Prox}_\phi(x)$, avec $\phi : \mathcal{X} \longrightarrow \mathbb{R} \cup \{+\infty\}$ définie par

$$\phi(x) = \begin{cases} g(D^{-1}(x))) - \frac{1}{2}||D^{-1}(x) - x||^2 & \text{si } x \in \mathrm{Im}(D), \\ +\infty & \text{sinon.} \end{cases} \tag{5.63}$$

(iii) $\forall x \in \mathcal{X}$, $\phi(x) \geqslant g(x)$ et pour $x \in \mathrm{Fix}(D)$, $\phi(x) = g(x)$.

(iv) ϕ est $\mathscr{C}^k$ sur $\mathrm{Im}(D)$ et $\forall x \in \mathrm{Im}(D)$, $\nabla\phi(x) = D^{-1}(x) - x = \nabla g(D^{-1}(x))$

(v) $\nabla\phi$ est $\frac{L}{1-L}$-Lipschitz sur $\mathrm{Im}(D)$

Nous avons prouvé cette proposition dans [C12] en s'inspirant des caractérisations des opérateurs proximaux de [62, 64]. Dans cette proposition, $D = \text{Prox}_\phi$ est univalué même si ϕ est potentiellement non-convexe. Notons aussi que l'hypothèse n'implique pas que D soit 1-Lipschitz.

On verra dans la Section 5.3.4.4 comment on peut en pratique apprendre une fonction de régularisation g en gardant un contrôle sur la constante de Lipschitz de ∇g. On ne pourra néanmoins pas garantir rigoureusement que cette constante de Lipschitz soit valable sur tout $\mathbb{R}^n$. Cependant, si ∇g est L-Lipschitz avec $L > 1$, on peut introduire un paramètre $\alpha \in (0, \frac{1}{L})$ de telle sorte que le débruiteur relaxé $D^\alpha = \alpha D + (1-\alpha)\,\text{Id} = \text{Id} - \alpha\nabla g$ puisse s'écrire comme un opérateur proximal.

5.3.4.3 Convergence des algorithmes PnP avec débruiteur proximal

La Proposition 5.3.3 permet d'écrire D_σ comme l'opérateur proximal d'une fonction non-convexe ϕ_σ. Par conséquent, les algorithmes de *splitting* basés sur l'utilisation de D_σ (vu comme opérateur proximal) visent à minimiser non plus $f + \lambda g$ mais une nouvelle fonctionnelle intégrant la régularisation ϕ_σ (qui est liée à g_σ au travers de l'équation (5.63)) :

$$F_{\lambda,\sigma}(x) = \frac{1}{\lambda} f(x) + \phi_\sigma(x). \tag{5.64}$$

Dans [C12] nous avons donné des garanties de convergence pour plusieurs algorithmes de *splitting* avec cette nouvelle régularisation. Mais nous n'inclurons ici que les résultats sur PnP-DRS en distinguant deux variantes selon l'ordre des opérateurs proximaux.

La première variante étudiée, notée PnP-DRSdiff s'écrit

$$\left\{ \begin{array}{l} y_{k+1} = \text{Prox}_{\frac{1}{\lambda} f}(x_k) \\ z_{k+1} = D_\sigma(2y_{k+1} - x_k) = \text{Prox}_{\phi_\sigma}(2y_{k+1} - x_k) \\ x_{k+1} = x_k + (z_{k+1} - y_{k+1}) \end{array} \right. \tag{PnP-DRSdiff}$$

Dans le cas où f est convexe différentiable, le théorème suivant prouve la convergence de cet algorithme, qui découle directement d'un résultat de [146], dont la preuve repose sur l'enveloppe de Douglas-Rachford

$$F^{DR,1}_{\lambda,\sigma}(x) = \phi_\sigma(z) + \frac{1}{\lambda} f(y) + \langle y - x, y - z\rangle + \frac{1}{2}||y - z||^2, \tag{5.65}$$

où $y = \text{Prox}_{\lambda f}(x)$ et $z = \text{Prox}_{\phi_\sigma}(2y - x)$.

Théorème 5.3.4 (Convergence de PnP-DRSdiff). *Soit $g_\sigma : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$ de classe $\mathscr{C}^2$ à gradient L-Lipschitz avec $L < 1$ et soit $D_\sigma := \text{Id} - \nabla g_\sigma$. Soit $f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$ convexe différentiable à gradient L_f-Lipschitz. Supposons que f et g_σ sont bornées inférieurement. Alors, pour $L_f < \lambda$, la suite (y_k, z_k, x_k) définie par (PnP-DRSdiff) vérifie*

(i) $(F^{DR,1}_{\lambda,\sigma}(x_k))$ est décroissante et convergente.

(ii) Le résidu $||y_k - z_k||$ tend vers 0.

(iii) *Les suites (y_k) et (z_k) ont mêmes valeurs d'adhérences, qui sont toutes des points critiques atteignant la même valeur de $F_{\lambda,\sigma}$.*

(iv) *Si de plus f vérifie la condition de Kurdika-Łojasiewicz, si g_σ est semi-algébrique, et si (y_k, z_k, x_k) est bornée, alors (y_k, z_k, x_k) converge, et les suites (y_k) et (z_k) convergent vers le même point critique de $F_{\lambda,\sigma}$.*

En plus de se limiter au cas où f est différentiable, l'inconvénient majeur du Théorème 5.3.4 est qu'il impose une condition $\lambda > L_f$ sur le paramètre de régularisation. Autrement dit, on peut garantir la convergence à condition de régulariser suffisamment le problème. En pratique, il est toujours possible de régulariser moins le problème en jouant également sur le paramètre σ contrôlant le niveau de débruitage. Cependant, il reste intéressant de pouvoir formuler des garanties valables sans condition sur λ.

Pour cela, on étudie maintenant une autre version de PnP-DRS, en échangeant l'ordre des opérateurs proximaux :

$$\begin{cases} y_{k+1} = D_\sigma(x_k) \\ z_{k+1} = \mathrm{Prox}_{\frac{1}{\lambda}f}(2y_{k+1} - x_k) \\ x_{k+1} = x_k + (z_{k+1} - y_{k+1}) \end{cases} \tag{PnP-DRS}$$

Cet algorithme peut même être étendu au cas où f n'est pas convexe, en prenant dans la deuxième étape un z_{k+1} quelconque dans le $\mathrm{Prox}_{\frac{1}{\lambda}f}$ multivalué. L'enveloppe de Douglas-Rachford associée à (PnP-DRS) s'écrit

$$F^{DR,2}_{\lambda,\sigma}(x) = \phi_\sigma(y) + \frac{1}{\lambda}f(z) + \langle y - x, y - z\rangle + \frac{1}{2}||y - z||^2. \tag{5.66}$$

Sans supposer de différentiabilité sur f, nous parvenons à formuler la garantie de convergence suivante.

Théorème 5.3.5 (Convergence de PnP-DRS)**.** *Soit $g_\sigma : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$ de classe $\mathscr{C}^2$ à gradient L-Lipschitz avec $L < \frac{1}{2}$, et soit $D_\sigma := \mathrm{Id} - \nabla g_\sigma$. Supposons que $\mathrm{Im}(D_\sigma)$ est convexe. Soit $f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$ propre et semi-continue inférieurement. Supposons que f et g_σ sont bornées inférieurement. Alors, pour tout $\lambda > 0$, la suite (y_k, z_k, x_k) définie par (PnP-DRS) vérifie*

(i) *$(F^{DR,2}_{\lambda,\sigma}(x_k))$ est décroissante et convergente.*

(ii) *Le résidu $||y_k - z_k||$ tend vers 0.*

(iii) *Si (y^*, z^*, x^*) est une valeur d'adhérence de (y_k, z_k, x_k), alors y^* et z^* coïncident en un point critique de $F_{\lambda,\sigma}$.*

(iv) *Si de plus f vérifie la condition de Kurdika-Łojasiewicz, si g_σ est semi-algébrique, et si (y_k, z_k, x_k) est bornée, alors (y_k, z_k, x_k) converge, et les suites (y_k) et (z_k) convergent vers le même point critique de $F_{\lambda,\sigma}$.*

On parvient donc à formuler une garantie de convergence pour PnP-DRS sans contraindre le paramètre de régularisation λ, au prix d'une contrainte plus forte sur ∇g_σ ($L < \frac{1}{2}$), et d'une hypothèse de convexité sur $\mathrm{Im}(D_\sigma)$ qui est difficile à vérifier en pratique.

$\sigma(./255)$	5	10	15	20	25
FFDNet	39.95	35.81	33.53	31.99	30.84
DnCNN	39.80	35.82	33.55	32.02	30.87
DRUNet	40.19	36.10	33.85	32.34	31.21
GS-DRUNet	40.27	36.16	33.92	32.41	31.28
Prox-DRUNet ($\mu = 10^{-3}$)	40.12	35.93	33.60	32.01	30.82
Prox-DRUNet ($\mu = 10^{-2}$)	40.04	35.86	33.51	31.88	30.64

TABLE 5.3 – Performances (mesurées en PSNR moyen) sur la base de données CBSD68 [106] du débruiteur Prox-DRUNet, en comparaison de GS-DRUNet et d'autres débruiteurs récents : FFDNet [165], DnCNN [163], et DRUNet [162]. Le débruiteur proximal atteint des performances similaires aux débruiteurs concurrents, et on observe une légère baisse de performance lorsque l'on cherche à réduire la constante de Lipschitz de ∇g_σ (en augmentant μ).

$\sigma(./255)$	0	5	10	15	20	25
GS-DRUNet ($\mu = 0$)	0.94	1.26	2.47	1.96	2.50	3.27
Prox-DRUNet ($\mu = 10^{-2}$)	0.87	0.92	0.95	0.99	0.96	0.96
Prox-DRUNet ($\mu = 10^{-3}$)	0.86	0.94	0.97	0.98	0.99	1.19

TABLE 5.4 – Valeurs de la norme spectrale $||\nabla^2 g_\sigma(x)||_S$ obtenues pour le débruiteur proximal sur des images de la base CBSD68 dataset bruitées à différents niveaux σ.

5.3.4.4 Performances des algorithmes PnP avec débruiteur proximal

On va maintenant donner brièvement quelques résultats obtenus avec PnP-DRS. Pour cela, on réutilise les fonctions D_σ, g_σ définies par (5.40), (5.41), en paramétrant là encore le réseau N_σ avec l'architecture DRUNet représentée en Fig 5.1 et des activations $\mathscr{C}^2$. Même si l'on ne peut imposer que ∇g_σ soit une contraction, on modifie l'apprentissage du débruiteur $D_\sigma = \mathrm{Id} - \nabla g_\sigma$ de telle sorte que la fonction de coût utilisée pénalise la norme spectrale $\|\nabla^2 g_\sigma\|_S$ de la hessienne de g_σ :

$$\mathcal{L}_S(\sigma) = \mathbb{E}_{x\sim p, w_\sigma \sim \mathcal{N}(0,\sigma^2)} \Big[||D_\sigma(x + w_\sigma) - x||^2 + \mu \max(||\nabla^2 g_\sigma(x + w_\sigma)||_S, 1 - \epsilon) \Big] \tag{5.67}$$

Le débruiteur ainsi obtenu est appelé **débruiteur proximal** et noté Prox-DRUNet ci-dessous. Sur les Tables 5.3 et 5.4, on peut lire les performances de ce nouveau débruiteur proximal, et les valeurs de la norme spectrale de $\nabla^2 g_\sigma(x)$. Avec la pénalisation de (5.67), on parvient à réduire les valeurs de cette norme spectrale, au prix d'une légère baisse de performance. Bien que l'on ne puisse garantir que ∇g_σ soit globalement contractante, les valeurs indiquent que la fonction g_σ n'est pas loin de vérifier les hypothèses des théorèmes de convergence du paragraphe précédent.

Une fois le débruiteur proximal entraîné, on peut l'utiliser dans les algorithmes PnP-PGD, PnP-DRS et PnP-DRSdiff, pour traiter différents problèmes inverses. On montre sur la Fig. 5.5 un résultat de déflouage, et l'on renvoie à [C12] pour d'autres expériences. Les résultats qualitatifs et quantitatifs de restauration obtenus avec ce nouveau débruiteur proximal sont à un niveau similaire à ceux obtenus avec l'algorithme GS-PnP. On constate aussi que les trajectoires réalisées avec les

algorithmes PnP-PGD et PnP-DRSdiff sont très similaires, ce que nous ne savons pas encore expliquer.

Pour finir, remarquons que, dans tous les diagrammes de convergence montrés ici, la convergence vers zéro du résidu semble plus rapide que celle prédite par les garanties de convergence démontrées théoriquement. Alors que ces garanties de convergence reposent sur des constantes de Lipschitz globales, on peut imaginer qu'une analyse plus locale soit mieux à même d'expliquer le taux de convergence observé. Il serait donc intéressant de présenter de nouveaux résultats de convergence localisés, dont les hypothèses pourraient être vérifiées avec des certificats numériques calculables explicitement.

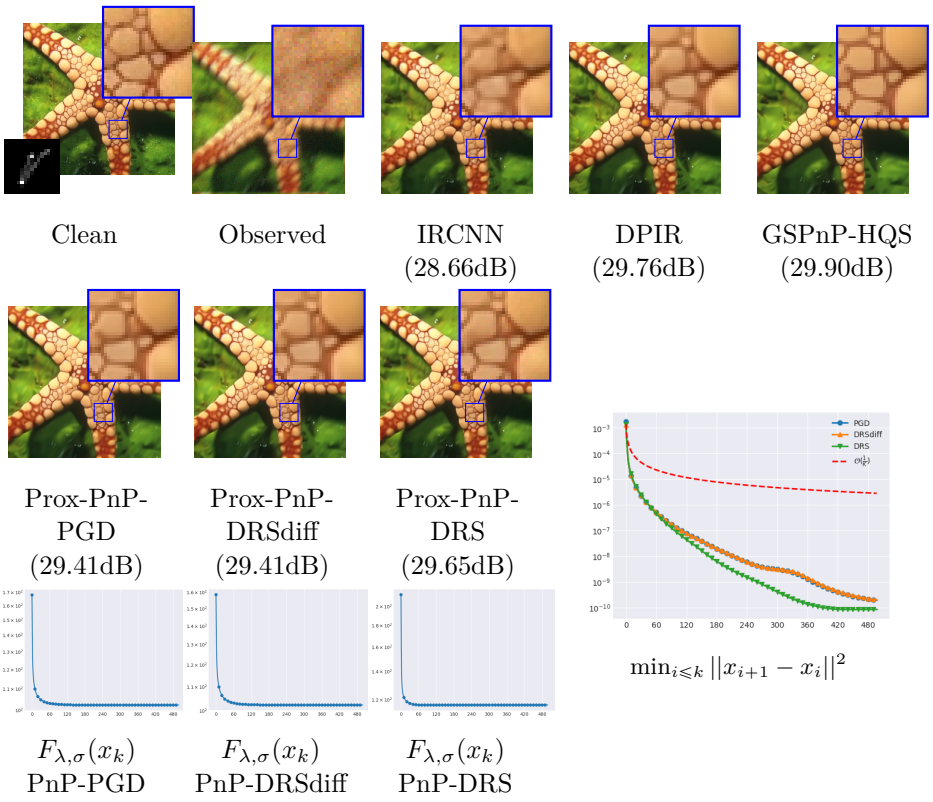


FIGURE 5.5 – Comparaison de résultats de déflouage obtenus avec plusieurs algorithmes de restauration appliqués à une image floutée (avec le noyau montré en haut à gauche) et bruitée ($\nu = 0.03$). Les diagrammes montrent l'évolution des fonctions de coût utilisées et du résidu au cours de l'algorithme, qui confirment les garanties de convergence obtenues dans la Section 5.3.4.3. Le réglage du paramètre λ est le même pour PnP-PGD et PnP-DRSdiff, mais pas pour PnP-DRS (qui optimise donc une fonctionnelle $F_{\lambda,\sigma}$ différente). Les algorithmes basés sur le débruiteur proximal produisent des résultats visuels au même niveau que ceux basés sur le débruiteur GS.

5.4 Conclusion Partielle

Dans ce chapitre nous avons étudié plusieurs algorithmes de restauration d'images Plug-and-Play basés sur différents algorithmes de *splitting*. Notre contribution principale est d'avoir introduit une architecture de réseau débruiteur par pas de gradient, qui permet d'interpréter l'algorithme Plug-and-Play associé comme un algorithme de minimisation d'une fonctionnelle explicite. En utilisant un a priori implicitement encodé dans un réseau débruiteur appris, on peut bénéficier des performances récentes de réseaux débruiteurs profonds, tout en conservant un contrôle numérique précis sur l'algorithme de restauration utilisé. On exploite l'observation suivante : il est plus facile de concevoir un débruiteur atteignant de bonnes performances sur une base de données, plutôt que de chercher directement un a priori explicite qui permette de refléter la régularité de ces données.

Nous avons formulé des garanties de convergence soumises à des hypothèses relativement légères sur le réseau débruiteur et les paramètres utilisés. Il est possible d'assouplir certaines des hypothèses introduites en utilisant des variantes d'algorithmes de *splitting*, comme nous l'avons fait récemment dans [C15]. Une autre piste consiste à modifier les hypothèses de régularité en jeu pour s'adapter à d'autres types de données ou d'autres modèles de bruit. Dans [Cs1] nous avons commencé à adapter la méthodologie Plug-and-Play aux problèmes inverses avec bruit de Poisson, en exploitant des étapes de descente implicite et explicite encodées via une divergence de Bregman [9]. L'adaptation fine de l'architecture du débruiteur à ce nouveau modèle de bruit fait l'objet d'un travail en cours.

Un enjeu majeur de la méthodologie Plug-and-Play est de mieux comprendre le type de régularité qui est encodée dans le débruiteur ou la régularisation explicite g_σ associée. Pour en avoir une idée empirique, on peut chercher à explorer les points critiques de g_σ, c'est-à-dire les points fixes du débruiteur D_σ. Mais il serait élégant du point de vue théorique, de pouvoir faire le lien avec un espace fonctionnel, de la même manière que la variation totale est liée aux fonctions à variation bornée. Dans cette optique, on pourrait certainement profiter de la construction de régularisations convexes [61] associées à des débruiteurs atteignant les performances de l'état de l'art.

Chapitre 6

Conclusion et Perspectives

Pour conclure ce document de synthèse, nous allons dresser quelques perspectives de recherche, découpées selon deux grands axes : le premier concernant les modèles génératifs d'images, et le deuxième portant sur la restauration d'images.

6.1 Modèles Génératifs

Les travaux présentés dans les trois premiers chapitres de ce manuscrit portent sur l'étude de modèles génératifs d'images, avec une application récurrente en synthèse de textures. Nous avons proposé différentes formulations mathématiques du problème de synthèse de textures, qui débouchent sur plusieurs algorithmes que l'on peut brièvement comparer.

Le modèle Texto est conçu de façon à permettre la synthèse avec un algorithme léger consistant en des projections au plus proche voisin par patch à plusieurs résolutions. Ce modèle permet d'englober une large classe de textures structurées, mais les synthèses portent toujours une légère quantité de flou due à l'agrégation des patches. La méthode GOTEX permet un meilleur contrôle de la qualité visuelle, tout en ayant un coût de synthèse limité (une simple passe dans un réseau génératif convolutionnel). Mais son optimisation est plus délicate, et la qualité de la synthèse est impactée (d'une façon peu lisible) par le choix d'architecture du réseau génératif. Enfin, les modèles exponentiels constituent une réponse exacte au problème de maximisation de l'entropie sous contraintes statistiques, pour laquelle on peut proposer des algorithmes convergents d'échantillonnage et d'estimation. Mais ces algorithmes sont en pratique très coûteux, et les synthèses avec ce modèle sont légèrement bruitées (ce qui est quasiment inévitable si l'on cherche à maximiser l'entropie).

L'une des caractéristiques de tous ces travaux est qu'ils tirent profit à la fois d'outils déterministes et stochastiques. Lorsque l'on découvre les articles écrits depuis les années 1980 sur la synthèse de textures, on peut se demander maintes fois si, pour bien synthétiser une texture, il faut optimiser ou échantillonner. La réponse est bien sûr : "un peu des deux"... Les trois premiers chapitres de ce manuscrit portent sur plusieurs cadres mathématiques, qui, je l'espère, justifient rigoureusement cette façon de faire "un peu des deux".

Dressons maintenant quelques perspectives en lien avec les modèles génératifs.

6.1.1 Accélération du Modèle TextoGMM

Le modèle TextoGMM de la Section 2.4 est limité, sur le plan théorique et pratique, par l'étape de sur-échantillonnage par projection NN par patch. Du point de

vue théorique, il serait intéressant d'analyser l'impact sur le coût de transport de cette projection NN, et de voir si elle peut être modifiée à bas coût pour mieux respecter les contraintes statistiques du transport optimal. Du point de vue pratique, l'étape de projection NN pourra être largement accélérée en exploitant des algorithmes de projection NN approchés [6, 29]. Une autre piste possible est de calculer un transport GMMOT prenant en compte simultanément deux échelles adjacentes, et ensuite d'exploiter un conditionnement par l'échelle grossière comme dans [121].

6.1.2 Utilisation de GMMOT pour les Wasserstein GAN

Une autre limitation du modèle Texto est qu'il ne permet pas de contrôler directement la distribution des patches de la texture synthétisée, à cause de l'étape d'agrégation des patches par simple moyennage. Nous avons pallié ce défaut avec le modèle GOTEX de la Section 3.2, qui minimise, pour chaque résolution ℓ, la distance de transport W_2^2 entre la distribution μ_θ^ℓ des patches de la texture synthétisée et la distribution ν^ℓ de patches de la texture exemple. Mais l'apprentissage du réseau génératif de GOTEX est coûteuse, à cause de la difficulté à estimer W_2^2 en grande dimension. On propose ici de remplacer le coût W_2^2 par un coût GMMOT, dont le calcul est beaucoup plus rapide, même en grande dimension. Ceci nécessitera d'approcher μ_θ^ℓ et ν^ℓ par un GMM à l'aide de l'algorithme EM, ce qui, malgré sa stabilité, sera l'étape la plus coûteuse. On devra donc voir comment l'estimation du modèle GMM peut être mise à jour efficacement en parallèle de l'algorithme d'optimisation du réseau.

Plus généralement, on pourra chercher à exploiter cette formulation GMMOT dans l'apprentissage WGAN pour des problèmes plus généraux de synthèse d'images [2]. En remplaçant là encore W_2^2 par un coût GMMOT, on s'affranchit de l'optimisation du réseau discriminateur, mais l'on doit garder une modélisation GMM courante des distributions μ_θ et ν. On cherchera à étudier le problème d'optimisation associé, en donnant l'expression des gradients en θ, et en analysant comment la modélisation GMM interagit avec cette expression.

6.1.3 GMMOT et Autoencodeurs Variationnels

Les autoencodeurs variationnels forment une classe de modèles génératifs menant à des résultats spectaculaires de génération d'images photoréalistes [30, 153]. Dans ces modèles, la distribution p_G des images générées se décompose avec une variable latente z de petite dimension, sous la forme $p_G(x) = p_\theta(x|z)p_Z(z)$ où $p_\theta(x|z)$ est un décodeur stochastique. Elle est apprise conjointement à un encodeur stochastique $q_\varphi(z|x)$ en maximisant un minorant de la log-vraisemblance du modèle [80] par rapport à la loi empirique des observations p_X. Les lois $p_\theta(x|z), q_\varphi(z|x)$ sont toutes deux paramétrées par des réseaux de neurones.

En se concentrant sur le cas des décodeurs déterministes $x = f_\theta(z)$, Tolstikhin *et al.* [148] ont proposé d'estimer le modèle génératif p_G en minimisant un coût de transport optimal relaxé

$$\mathcal{L}_{\mathrm{WAE}}(p_X, p_G) = \mathbb{E}_{p_X(x)}\mathbb{E}_{q_\varphi(z|x)}[\|x - f_\theta(z)\|^2] + \lambda D(q_\varphi(z), p_Z(z)) \tag{6.1}$$

où D est une mesure de l'écart entre les lois de probabilité $q_\varphi(z)$ et $p_Z(z)$, pondérée par un $\lambda \geqslant 0$. Ce deuxième terme permet d'imposer que la distribution des variables latentes soit proche d'une distribution prédéfinie, e.g. $P_Z = \mathcal{N}(0, I)$. Ces auteurs ont utilisé pour D la divergence de Jensen-Shannon, ou bien des métriques à noyau, mais il est possible ici de considérer des distances de transport optimal. Alors que l'algorithme de [148] repose sur une méthode duale stochastique pour le calcul approché de $D(q_\varphi(z), p_Z(z))$, l'approche de [166] est basée sur la formule close de la distance de Wasserstein W_2 entre deux gaussiennes.

On pourrait étendre cette construction en exploitant le transport optimal entre mélanges de gaussiennes GMMOT. Par exemple, en prenant P_Z un mélange de K gaussiennes avec des modes bien séparés, on pourrait modéliser des distributions d'images multi-modales. Avec des modèles génératifs simples (avec peu de couches non-linéaires dans f_θ), il serait intéressant de comparer cette procédure d'inférence à un algorithme d'estimation de modèles de mélanges de gaussiennes. On obtiendrait une représentation de la distribution des données pouvant se voir comme une extension courbée de la modélisation GMM. Au-delà d'un gain algorithmique, ceci ouvrirait de nouvelles possibilités de structuration de l'espace latent permettant de faire émerger des attributs des données de façon pertinente [117].

6.2 Restauration d'Images Plug-and-Play

Avec Samuel Hurault et Nicolas Papadakis, nous avons proposé de nouvelles régularisations par réseaux profonds permettant d'obtenir des garanties de convergence pour des algorithmes PnP. Ces algorithmes produisent des résultats de restauration à l'état de l'art, tout en conservant un contrôle numérique précis. Ces travaux constituent une percée inédite en fournissant des garanties de convergence réalistes dans un cadre non-convexe, avec une régularisation explicite liée à un débruiteur profond qui, contrairement à d'autres travaux concurrents, n'est pas soumise à des hypothèses contraignantes. Ils ouvrent aussi de larges perspectives de recherche, dont plusieurs axes vont être esquissés dans les paragraphes suivants.

6.2.1 Méthodes PnP pour d'autres types de bruit

Nous pourrons chercher à adapter nos algorithmes PnP à d'autres types de données, notamment en imagerie radar. L'une des spécificités de l'imagerie radar par synthèse d'ouverture (SAR) [40] est que les données sont affectées par un bruit de *speckle* dû à des effets d'interférence dans les échos radar. Ceci nécessite d'utiliser une attache aux données f non-convexe, différentiable qui n'est pas à gradient Lipschitz. L'utilisation du Bregman *splitting* [9] débouche sur de nouveaux algorithmes PnP [1] dont on pourra poursuivre l'analyse mathématique. Ceci permettra d'adapter la méthodologie PnP à des bruits non gaussiens (en imaginant de nouvelles paramétrisations du débruiteur), ou à des données structurées (e.g. pour l'imagerie SAR multivariée).

6.2.2 Plug-and-Play avec Algorithme ADMM Linéarisé

Certains algorithmes PnP nécessitent un calcul explicite de $\mathrm{Prox}_{\tau\lambda f}$, qui s'avère délicat pour des attaches aux données liées à un modèle d'acquisition réaliste (convolution avec restriction des bords, flou non uniforme, etc). On peut contourner cette difficulté en utilisant un algorithme ADMM linéarisé. Cette approche a déjà mené à de bons résultats expérimentaux pour le déflouage non uniforme [88], et son étude théorique a été entamée dans [100, 87]. On pourra poursuivre cette étude en prouvant la convergence des itérés de l'algorithme, par exemple en exploitant la connexion avec [C15]. De plus, on pourra chercher à étendre cet algorithme à d'autres fonctions f, g_σ, et notamment aux attaches aux données provenant de problèmes inverses non-linéaires.

6.2.3 Analyse des Régularisations Profondes, Lien avec la Diffusion

On constate en pratique que les performances en restauration sont meilleures lorsque le niveau σ du débruiteur est choisi plus grand que le niveau de bruit ν de l'attache aux données. Alors que ν peut être simplement absorbé par le paramètre de régularisation λ, le réglage du paramètre σ nécessite une étude approfondie. En notant p un prior et p_σ sa convolution par $\mathcal{N}(0, \sigma^2)$, la formule de Tweedie [46] assure que le débruiteur D^*_σ MMSE vérifie $D^*_\sigma(x) = x + \sigma^2 \nabla_x \log p_\sigma(x)$. Cette formule invite à étudier le comportement de $\frac{D_\sigma - \mathrm{Id}}{\sigma^2}$ lorsque $\sigma \longrightarrow 0$. On pourra chercher à expliciter ce comportement pour les architectures DRUNET [162], et *bias-free* [113]. Cette question est liée à la convergence faible du prior lissé e^{-g_σ} quand $\sigma \longrightarrow 0$. Il est plausible que la loi dégénère sur un ensemble d'images "propres" au sens de notre régularisation, ce qui invite à chercher un lien avec les modèles de diffusion [17].

6.2.4 Échantillonnage de la loi a posteriori

À plus long terme, un objectif ambitieux consiste en l'échantillonnage de la loi a posteriori [89, 32], proportionnelle à $e^{-F} = e^{-f - \lambda g_\sigma}$. Cette méthode stochastique permet d'approcher d'autres estimateurs statistiques pour la restauration, comme l'espérance a posteriori, qui peut avoir des propriétés plus favorables que le maximum a posteriori [101].

Avec les régularisations explicites g_σ, on pourra étudier la loi a posteriori $p_\sigma(x|y)$ en suivant [89, 70, 110, 114]. Selon la régularité de $x \mapsto \nabla \log p_\sigma(x|y)$, on pourra formuler des garanties de convergence de l'algorithme de Langevin permettant d'échantillonner $p_\sigma(x|y)$, en réutilisant les outils développés dans [J11]. On pourra également essayer d'échantillonner le prior lissé e^{-g_σ} évoqué dans le paragraphe précédent, et étudier le comportement des échantillons quand $\sigma \longrightarrow 0$. Enfin, nous pourrions aussi considérer d'autres priors basés sur des modèles à variables latentes, comme ceux de [58, 70]. Toutes ces techniques d'échantillonnage du posterior permettront de faire de la quantification d'incertitude [124, 70], ce qui peut s'avérer crucial dans des applications sensibles telles que l'imagerie médicale ou l'imagerie satellitaire.

Bibliographie

[1] Al-Shabili, A. H., Xu, X., Selesnick, I. et Kamilov, U. S. (2022). Bregman plug-and-play priors. *In 2022 IEEE International Conference on Image Processing (ICIP)*, pages 241–245. IEEE.

[2] Arjovsky, M., Chintala, S. et Bottou, L. (2017). Wasserstein generative adversarial networks. *In International Conference on Machine Learning*, pages 214–223.

[3] Attouch, H., Bolte, J. et Svaiter, B. F. (2013). Convergence of descent methods for semi-algebraic and tame problems : proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods. *Mathematical Programming*, 137(1-2):91–129.

[4] Aurenhammer, F., Hoffmann, F. et Aronov, B. (1998). Minkowski-type theorems and least-squares clustering. *Algorithmica*, 20(1):61–76.

[5] Ballu, M., Berthet, Q. et Bach, F. (2020). Stochastic optimization for regularized wasserstein estimators. *In International Conference on Machine Learning*, pages 602–612. PMLR.

[6] Barnes, C., Shechtman, E., Finkelstein, A. et Goldman, D. B. (2009). Patchmatch : a randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24 :1–24 :11.

[7] Barron, J. T. (2017). Continuously differentiable exponential linear units.

[8] Bassetti, F., Bodini, A. et Regazzini, E. (2006). On minimum Kantorovich distance estimators. *Statistics & probability letters*, 76(12):1298–1302.

[9] Bauschke, H. H., Bolte, J. et Teboulle, M. (2017). A descent lemma beyond Lipschitz gradient continuity : first-order methods revisited and applications. *Mathematics of Operations Research*, 42(2):330–348.

[10] Bauschke, H. H. et Combettes, P. L. (2011). *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer.

[11] Bauschke, H. H., Combettes, P. L. et Reich, S. (2005). The asymptotic behavior of the composition of two resolvents. *Nonlinear Analysis : Theory, Methods & Applications*, 60(2):283–301.

[12] Beck, A. et Teboulle, M. (2009). Gradient-based algorithms with applications to signal recovery. *Convex optimization in signal processing and communications*, pages 42–88.

[13] Bergmann, U., Jetchev, N. et Vollgraf, R. (2017). Learning texture manifolds with the periodic spatial GAN. *In Proceedings of the 34th International Conference on Machine Learning*, pages 469–477.

[14] Bigdeli, S. A., Lin, G., Portenier, T., Dunbar, L. A. et Zwicker, M. (2020). Learning generative models using denoising density estimators. *arXiv preprint arXiv :2001.02728*.

[15] Bohra, P., Goujon, A., Perdios, D., Emery, S. et Unser, M. (2021). Learning Lipschitz-Controlled Activation Functions in Neural Networks for Plug-and-Play Image Reconstruction Methods. *In NeurIPS Workshop on Deep Learning and Inverse Problems*.

[16] Bora, A., Jalal, A., Price, E. et Dimakis, A. G. (2017). Compressed sensing using generative models. *In International Conference on Machine Learning*, pages 537–546. PMLR.

[17] Bortoli, V. D. (2022). Convergence of denoising diffusion models under the manifold hypothesis. *Transactions on Machine Learning Research*.

[18] Bourrier, A., Davies, M., Peleg, T., Perez, P. et Gribonval, R. (2014). Fundamental performance limits for ideal decoders in high-dimensional linear inverse problems. *IEEE Transactions on Information Theory*, 60(12):7928–7946.

[19] Boyd, S., Parikh, N. et Chu, E. (2011). *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc.

[20] Bruna, J. et Mallat, S. (2019). Multiscale sparse microcanonical models. *Mathematical Statistics and Learning*, 1(3):257–315.

[21] Buades, A., Coll, B. et Morel, J.-M. (2005). A review of image denoising algorithms, with a new one. *Multiscale modeling & simulation*, 4(2):490–530.

[22] Burger, H. C., Schuler, C. J. et Harmeling, S. (2012). Image denoising : Can plain neural networks compete with BM3D ? *In Proceedings of the IEEE Conference on computer vision and pattern recognition*, pages 2392–2399. IEEE.

[23] Chambolle, A. et Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145.

[24] Chan, S. H., Wang, X. et Elgendy, O. A. (2016). Plug-and-play ADMM for image restoration : Fixed-point convergence and applications. *IEEE Transactions on Computational Imaging*, 3(1):84–98.

[25] Chellappa, R. et Chatterjee, S. (1985). Classification of textures using Gaussian Markov random fields. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33(4):959–963.

[26] Chellappa, R. et Kashyap, R. (1985). Texture synthesis using 2D noncausal autoregressive models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33(1):194–203.

[27] Chen, Y. et Pock, T. (2016). Trainable nonlinear reaction diffusion : A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272.

[28] Chen, Y., Telgarsky, M., Zhang, C., Bailey, B., Hsu, D. et Peng, J. (2019). A gradual, semi-discrete approach to generative network training via explicit Wasserstein minimization. *In International Conference on Machine Learning*, pages 1071–1080. PMLR.

[29] Chérel, N., Almansa, A., Gousseau, Y. et Newson, A. (2022). A patch-based algorithm for diverse and high fidelity single image generation. *In IEEE International Conference on Image Processing (ICIP)*, pages 3221–3225.

[30] Child, R. (2021). Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images. *In International Conference on Learning Representations*.

[31] Chizat, L., Roussillon, P., Léger, F., Vialard, F.-X. et Peyré, G. (2020). Faster Wasserstein Distance Estimation with the Sinkhorn Divergence. *In Proc. NeurIPS'20*.

[32] Coeurdoux, F., Dobigeon, N. et Chainais, P. (2023). Plug-and-Play split Gibbs sampler : embedding deep generative priors in Bayesian inference. *arXiv preprint arXiv :2304.11134*.

[33] Combettes, P. L. et Pesquet, J.-C. (2011). Proximal splitting methods in signal processing. *In Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer.

[34] Combettes, P. L. et Wajs, V. R. (2005). Signal recovery by proximal forward-backward splitting. *Multiscale modeling & simulation*, 4(4):1168–1200.

[35] Cross, G. et Jain, A. (1983). Markov random field texture models. *IEEE Trans. PAMI*, 5(1):25–39.

[36] Dalalyan, A. S. (2017). Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, 79(3):651–676.

[37] Damara, M. F., Kornhardt, G. et Jung, P. (2021). Solving inverse problems with conditional-gan prior via fast network-projected gradient descent. *arXiv preprint arXiv :2109.01105*.

[38] Daubechies, I., Defrise, M. et De Mol, C. (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics : A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457.

[39] De Bortoli, V., Durmus, A., Pereyra, M. et Vidal, A. F. (2021). Efficient stochastic optimisation by unadjusted langevin monte carlo : Application to maximum marginal likelihood and empirical bayesian estimation.

[40] Deledalle, C.-A., Denis, L., Tabti, S. et Tupin, F. (2017). MuLoG, or how to apply Gaussian denoisers to multi-channel SAR speckle reduction ? *IEEE Transactions on Image Processing*, 26(9):4389–4403.

[41] Delon, J. et Desolneux, A. (2020). A Wasserstein-type distance in the space of Gaussian mixture models. *SIAM Journal on Imaging Sciences*, 13(2):936–970.

[42] Dong, C., Loy, C. C., He, K. et Tang, X. (2015). Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307.

[43] Durmus, A. et Moulines, E. (2017). Nonasymptotic convergence analysis for the unadjusted Langevin algorithm. *Ann. Appl. Probab.*, 27(3):1551–1587.

[44] Durmus, A., Roberts, G. O., Vilmart, G., Zygalakis, K. C. *et al.* (2017). Fast langevin based algorithm for mcmc in high dimensions. *The Annals of Applied Probability*, 27(4):2195–2237.

[45] Eckstein, J. et Bertsekas, D. P. (1992). On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55:293–318.

[46] Efron, B. (2011). Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614.

[47] Efros, A. A. et Freeman, W. T. (2001). Image quilting for texture synthesis and transfer. *In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, pages 341–346.

[48] Efros, A. A. et Leung, T. K. (1999). Texture synthesis by non-parametric sampling. *In Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 1033–1038.

[49] Elad, M. et Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745.

[50] Galerne, B., Gousseau, Y. et Morel, J. (2011). Random phase textures : Theory and synthesis. *IEEE Trans. Image Processing*, 20(1):257–267.

[51] Gatys, L. A., Ecker, A. S. et Bethge, M. (2015). Texture synthesis using convolutional neural networks. *In Advances in Neural Information Processing Systems*, pages 262–270.

[52] Gavaskar, R. G. et Chaudhury, K. N. (2019). On the proof of fixed-point convergence for plug-and-play ADMM. *IEEE Signal Processing Letters*, 26(12): 1817–1821.

[53] Geman, D. et Yang, C. (1995). Nonlinear image recovery with half-quadratic regularization. *IEEE transactions on Image Processing*, 4(7):932–946.

[54] Geman, S. et Graffigne, C. (1986). Markov random field image models and their applications to computer vision. *In Proceedings of the international congress of mathematicians*, volume 1, page 2.

[55] Genevay, A. (2019). *Entropy-regularized optimal transport for machine learning*. Thèse de doctorat, Université Paris Dauphine.

[56] Genevay, A., Chizat, L., Bach, F., Cuturi, M. et Peyré, G. (2019). Sample complexity of Sinkhorn divergences. *In The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1574–1583. PMLR.

[57] Genevay, A., Cuturi, M., Peyré, G. et Bach, F. (2016). Stochastic optimization for large-scale optimal transport. *In Proc. of NIPS*, pages 3432–3440.

[58] González, M., Almansa, A. et Tan, P. (2022). Solving inverse problems by joint posterior maximization with autoencoding prior. *SIAM Journal on Imaging Sciences*, 15(2):822–859.

[59] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. et Bengio, Y. (2014). Generative adversarial nets. *In Advances in neural information processing systems*, pages 2672–2680.

[60] Gossard, A. et Weiss, P. (2022). Training adaptive reconstruction networks for inverse problems. *arXiv preprint arXiv :2202.11342*.

[61] Goujon, A., Neumayer, S., Bohra, P., Ducotterd, S. et Unser, M. (2022). A neural-network-based convex regularizer for image reconstruction. *arXiv preprint arXiv :2211.12461*.

[62] Gribonval, R. (2011). Should penalized least squares regression be interpreted as maximum a posteriori estimation ? *IEEE Transactions on Signal Processing*, 59(5):2405–2410.

[63] Gribonval, R., Blanchard, G., Keriven, N. et Traonmilin, Y. (2021). Compressive Statistical Learning with Random Feature Moments. *Math. Stat. Learn., In press*, 3(2):113–164.

[64] Gribonval, R. et Nikolova, M. (2020). A characterization of proximity operators. *Journal of Mathematical Imaging and Vision*, 62(6):773–789.

[65] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. et Courville, A. C. (2017). Improved training of Wasserstein GANs. *In Advances in neural information processing systems*, pages 5767–5777.

[66] Gutierrez, J., Galerne, B., Rabin, J. et Hurtut, T. (2017). Optimal patch assignment for statistically constrained texture synthesis. *In Proceedings of SSVM*.

[67] Hand, P. et Joshi, B. (2019). Global guarantees for blind demodulation with generative priors. *Advances in Neural Information Processing Systems*, 32.

[68] Heckel, R. et Soltanolkotabi, M. (2020). Denoising and regularization via exploiting the structural bias of convolutional generators. *In International Conference on Learning Representations*.

[69] Hertrich, J., Neumayer, S. et Steidl, G. (2021). Convolutional proximal neural networks and plug-and-play algorithms. *Linear Algebra and its Applications*, 631:203–234.

[70] Holden, M., Pereyra, M. et Zygalakis, K. C. (2022). Bayesian imaging with data-driven priors encoded by neural networks. *SIAM Journal on Imaging Sciences*, 15(2):892–924.

[71] Houdard, A., Bouveyron, C. et Delon, J. (2018). High-dimensional mixture models for unsupervised image denoising (HDMI). *SIAM Journal on Imaging Sciences*, 11(4):2815–2846.

[72] Hurault, S., Ehret, T. et Arias, P. (2018). EPLL : an image denoising method using a Gaussian mixture model learned on a large set of patches. *Image Processing On Line*, 8:465–489.

[73] Hyder, R. et Asif, M. S. (2020). Generative models for low-dimensional video representation and reconstruction. *IEEE Transactions on Signal Processing*, 68: 1688–1701.

[74] Jain, V. et Seung, S. (2008). Natural image denoising with convolutional networks. *Advances in neural information processing systems*, 21.

[75] Jaynes, E. T. (1957). Information theory and statistical mechanics. *Phys. Rev.*

[76] Julesz, B. (1981a). Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97.

[77] Julesz, B. (1981b). A theory of preattentive texture discrimination based on first-order statistics of textons. *Biological Cybernetics*, 41(2):131–138.

[78] Kamilov, U. S., Mansour, H. et Wohlberg, B. (2017). A plug-and-play priors approach for solving nonlinear imaging inverse problems. *IEEE Signal Processing Letters*, 24(12):1872–1876.

[79] Kingma, D. et Ba, J. (2015). Adam : A method for stochastic optimization. *In Proceedings of the International Conference on Learning Representations*.

[80] Kingma, D. P. et Welling, M. (2019). An introduction to variational autoencoders. *arXiv preprint arXiv :1906.02691*.

[81] Kitagawa, J., Mérigot, Q. et Thibert, B. (2017). A Newton algorithm for semi-discrete optimal transport. *Journal of the European Math Society*.

[82] Korotin, A., Kolesov, A. et Burnaev, E. (2022). Kantorovich Strikes Back ! Wasserstein GANs are not Optimal Transport ? *In Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[83] Korotin, A., Li, L., Genevay, A., Solomon, J. M., Filippov, A. et Burnaev, E. (2021). Do neural optimal transport solvers work ? a continuous wasserstein-2 benchmark. *Advances in Neural Information Processing Systems*, 34:14593–14605.

[84] Krizhevsky, A., Sutskever, I. et Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. *In Advances in neural information processing systems*, pages 1097–1105.

[85] Kwatra, V., Essa, I., Bobick, A. et Kwatra, N. (2005). Texture optimization for example-based synthesis. *In ACM Transactions on Graphics*, volume 24, pages 795–802.

[86] Kwatra, V., Schödl, A., Essa, I., Turk, G. et Bobick, A. (2003). Graphcut textures : Image and video synthesis using graph cuts. *ACM TOG*, 22(3):277–286.

[87] Laroche, C., Almansa, A., Coupeté, E. et Tassano, M. (2023a). Provably Convergent Plug & Play Linearized ADMM, applied to Deblurring Spatially Varying Kernels. *In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

[88] Laroche, C., Almansa, A. et Tassano, M. (2023b). Deep model-based super-resolution with non-uniform blur. *In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1797–1808.

[89] Laumont, R., Bortoli, V. D., Almansa, A., Delon, J., Durmus, A. et Pereyra, M. (2022). Bayesian imaging using Plug & Play priors : when Langevin meets Tweedie. *SIAM Journal on Imaging Sciences*, 15(2):701–737.

[90] Lebrun, M., Buades, A. et Morel, J. (2013). A nonlocal Bayesian image denoising algorithm. *SIAM Journal on Imaging Sciences*, 6(3):1665–1688.

[91] Lebrun, M., Colom, M., Buades, A. et Morel, J.-M. (2012). Secrets of image denoising cuisine. *Acta Numerica*, 21:475–576.

[92] LeCun, Y., Bottou, L., Bengio, Y. et Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[93] Leong, O. (2021). *Learned Generative Priors for Imaging Inverse Problems*. Thèse de doctorat, Rice University.

[94] Levina, E. et Bickel, P. (2006). Texture synthesis and nonparametric resampling of random fields. *The Annals of Statistics*, 34(4):1751–1773.

[95] Li, G. et Pong, T. K. (2016). Douglas–Rachford splitting for nonconvex optimization with application to nonconvex feasibility problems. *Mathematical programming*, 159(1):371–401.

[96] Liang, L., Liu, C., Xu, Y.-Q., Guo, B. et Shum, H.-Y. (2001). Real-time texture synthesis by patch-based sampling. *ACM TOG*, 20(3):127–150.

[97] Lions, P. L. et Mercier, B. (1979). Splitting algorithms for the sum of two nonlinear operators. *Siam J. Numer. Anal.*, 16:964–979.

[98] Liu, D., Vu, M. T., Chatterjee, S. et Rasmussen, L. K. (2019a). Entropy-regularized optimal transport generative models. *In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3532–3536. IEEE.

[99] Liu, D. C. et Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528.

[100] Liu, Q., Shen, X. et Gu, Y. (2019b). Linearized ADMM for nonconvex nonsmooth optimization with convergence analysis. *IEEE Access*, 7:76131–76144.

[101] Louchet, C. et Moisan, L. (2013). Posterior expectation of the total variation model : properties and experiments. *SIAM Journal on Imaging Sciences*, 6(4):2640–2684.

[102] Lu, Y., Zhu, S. et Wu, Y. (2016). Learning FRAME Models Using CNN Filters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1).

[103] Mairal, J., Elad, M. et Sapiro, G. (2008). Sparse representation for color image restoration. *IEEE Transactions on image processing*, 17(1):53–69.

[104] Mallasto, A., Frellsen, J., Boomsma, W. et Feragen, A. (2019a). (q, p)-Wasserstein GANs : Comparing Ground Metrics for Wasserstein GANs. *arXiv preprint arXiv :1902.03642*.

[105] Mallasto, A., Montúfar, G. et Gerolin, A. (2019b). How well do WGANs estimate the wasserstein metric ? *arXiv preprint arXiv :1910.03875*.

[106] Martin, D., Fowlkes, C., Tal, D. et Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *In Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423.

[107] McCann, M. T., Jin, K. H. et Unser, M. (2017). Convolutional neural networks for inverse problems in imaging : A review. *IEEE Signal Processing Magazine*, 34(6):85–95.

[108] McLachlan, G. et Krishnan, T. (2007). *The EM algorithm and extensions*, volume 382. John Wiley & Sons.

[109] Meinhardt, T., Moller, M., Hazirbas, C. et Cremers, D. (2017). Learning proximal operators : Using denoising networks for regularizing inverse imaging problems. *In Proceedings of the IEEE International Conference on Computer Vision*, pages 1781–1790.

[110] Melidonis, S., Dobson, P., Altmann, Y., Pereyra, M. et Zygalakis, K. C. (2022). Efficient Bayesian computation for low-photon imaging problems. *arXiv preprint arXiv :2206.05350*.

[111] Menon, S., Damian, A., Hu, S., Ravi, N. et Rudin, C. (2020). Pulse : Self-supervised photo upsampling via latent space exploration of generative models. *In Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 2437–2445.

[112] Mérigot, Q. (2011). A multiscale approach to optimal transport. *Computer Graphics Forum*, 30(5):1583–1592.

[113] Mohan, S., Kadkhodaie, Z., Simoncelli, E. P. et Fernandez-Granda, C. (2020). Robust and interpretable blind image denoising via bias-free convolutional neural networks. *In International Conference on Learning Representations*.

[114] Mukherjee, S., Hauptmann, A., Öktem, O., Pereyra, M. et Schönlieb, C.-B. (2022). Learned reconstruction with convergence guarantees. *arXiv preprint arXiv :2206.05431*.

[115] Nair, P., Gavaskar, R. G. et Chaudhury, K. N. (2021). Fixed-point and objective convergence of plug-and-play algorithms. *IEEE Transactions on Computational Imaging*.

[116] Pesquet, J.-C., Repetti, A., Terris, M. et Wiaux, Y. (2021). Learning maximally monotone operators for image recovery. *SIAM Journal on Imaging Sciences*, 14(3):1206–1237.

[117] Pham, C.-H., Ladjal, S. et Newson, A. (2022). PCA-AE : Principal Component Analysis Autoencoder for Organising the Latent Space of Generative Networks. *Journal of Mathematical Imaging and Vision*, 64(5):569–585.

[118] Portilla, J. et Simoncelli, E. P. (2000). A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70.

[119] Puy, G., Davies, M. E. et Gribonval, R. (2017). Recipes for Stable Linear Embeddings From Hilbert Spaces to $\mathbb{R}^m$. *IEEE Trans. Inform. Theory*, 63(4): 2171–2187.

[120] Raad, L., Davy, A., Desolneux, A. et Morel, J.-M. (2018). A survey of exemplar-based texture synthesis. *Annals of Mathematical Sciences and Applications*, 3(1):89–148.

[121] Raad, L., Desolneux, A. et Morel, J. (2016). A Conditional Multiscale Locally Gaussian Texture Synthesis Algorithm. *J. Math. Imag. Vision*, 56(2):260–279.

[122] Radford, A., Metz, L. et Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv :1511.06434*.

[123] Reehorst, E. T. et Schniter, P. (2018). Regularization by denoising : Clarifications and new interpretations. *IEEE Transactions on Computational Imaging*, 5(1):52–67.

[124] Repetti, A., Pereyra, M. et Wiaux, Y. (2019). Scalable Bayesian uncertainty quantification in imaging inverse problems via convex optimization. *SIAM Journal on Imaging Sciences*, 12(1):87–118.

[125] Roberts, G. O. et Tweedie, R. L. (1996). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363.

[126] Romano, Y., Elad, M. et Milanfar, P. (2017). The little engine that could : Regularization by denoising (RED). *SIAM Journal on Imaging Sciences*, 10(4):1804–1844.

[127] Rudin, L. I., Osher, S. et Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D : nonlinear phenomena*, 60(1-4):259–268.

[128] Ryu, E., Liu, J., Wang, S., Chen, X., Wang, Z. et Yin, W. (2019). Plug-and-play methods provably converge with properly trained denoisers. *In International Conference on Machine Learning*, pages 5546–5557. PMLR.

[129] Sanjabi, M., Ba, J., Razaviyayn, M. et Lee, J. D. (2018). On the convergence and robustness of training gans with regularized optimal transport. *In Advances in Neural Information Processing Systems*, pages 7091–7101.

[130] Santambrogio, F. (2015). Optimal transport for applied mathematicians. *Birkäuser, NY*.

[131] Santhanam, V., Morariu, V. I. et Davis, L. S. (2017). Generalized deep image to image regression. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5609–5619.

[132] Saremi, S. et Hyvarinen, A. (2019). Neural empirical bayes. *arXiv preprint arXiv :1903.02334*.

[133] Schuler, C. J., Christopher Burger, H., Harmeling, S. et Scholkopf, B. (2013). A machine learning approach for non-blind image deconvolution. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1067–1074.

[134] Seguy, V., Damodaran, B. B., Flamary, R., Courty, N., Rolet, A. et Blondel, M. (2018). Large-scale optimal transport and mapping estimation. *In ICLR 2018-International Conference on Learning Representations*, pages 1–15.

[135] Shaham, T. R., Dekel, T. et Michaeli, T. (2019). SinGAN : Learning a Generative Model from a Single Natural Image. *In Proceedings of the IEEE International Conference on Computer Vision*, pages 4570–4580.

[136] Shamshad, F. et Ahmed, A. (2020). Compressed sensing-based robust phase retrieval via deep generative priors. *IEEE Sensors Journal*, 21(2):2286–2298.

[137] Shen, Y., Gu, J., Tang, X. et Zhou, B. (2020). Interpreting the latent space of gans for semantic face editing. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[138] Simonyan, K. et Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *In* Bengio, Y. et LeCun, Y., éditeurs : *Proceedings of the International Conference on Learning Representations*.

[139] Sreehari, S., Venkatakrishnan, S. V., Wohlberg, B., Buzzard, G. T., Drummy, L. F., Simmons, J. P. et Bouman, C. A. (2016). Plug-and-play priors for bright field electron tomography and sparse interpolation. *IEEE Transactions on Computational Imaging*, 2(4):408–423.

[140] Stanczuk, J., Etmann, C., Kreusser, L. M. et Schönlieb, C.-B. (2021). Wasserstein GANs work because they fail (to approximate the Wasserstein distance). *arXiv preprint arXiv :2103.01678*.

[141] Sun, Y., Wohlberg, B. et Kamilov, U. S. (2019). An online plug-and-play algorithm for regularized image reconstruction. *IEEE Transactions on Computational Imaging*, 5(3):395–408.

[142] Sun, Y., Wu, Z., Xu, X., Wohlberg, B. et Kamilov, U. S. (2021). Scalable plug-and-play ADMM with convergence guarantees. *IEEE Transactions on Computational Imaging*, 7:849–863.

[143] Tartavel, G., Peyré, G. et Gousseau, Y. (2016). Wasserstein Loss for Image Synthesis and Restoration. *SIAM J. on Imaging Sciences*, 9(4):1726–1755.

[144] Teodoro, A. M., Bioucas-Dias, J. M. et Figueiredo, M. A. (2016). Image restoration and reconstruction using variable splitting and class-adapted image priors. *In IEEE International Conference on Image Processing (ICIP)*, pages 3518–3522. IEEE.

[145] Terris, M., Repetti, A., Pesquet, J.-C. et Wiaux, Y. (2020). Building firmly nonexpansive convolutional neural networks. *In IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8658–8662. IEEE.

[146] Themelis, A. et Patrinos, P. (2020). Douglas–Rachford Splitting and ADMM for Nonconvex Optimization : Tight Convergence Results. *SIAM Journal on Optimization*, 30(1):149–181.

[147] Tikhonov, A. N. (1963). On the solution of ill-posed problems and the method of regularization. *Doklady akademii nauk*, 151(3):501–504.

[148] Tolstikhin, I., Bousquet, O., Gelly, S. et Schölkopf, B. (2018). Wasserstein auto-encoders. *In Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*.

[149] Traonmilin, Y. et Aujol, J.-F. (2020). The basins of attraction of the global minimizers of the non-convex sparse spike estimation problem. *Inverse Problems*, 36(4):045003.

[150] Ulyanov, D., Lebedev, V., Vedaldi, A. et Lempitsky, V. S. (2016). Texture networks : Feed-forward synthesis of textures and stylized images. *In Proceedings of the 33nd International Conference on Machine Learning*, pages 1349–1357.

[151] Ulyanov, D., Vedaldi, A. et Lempitsky, V. (2017). Improved texture networks : Maximizing quality and diversity in feed-forward stylization and texture synthesis. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6924–6932.

[152] Ulyanov, D., Vedaldi, A. et Lempitsky, V. (2018). Deep image prior. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454.

[153] Vahdat, A. et Kautz, J. (2020). Nvae : A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33.

[154] Varma, M. et Zisserman, A. (2003). Texture classification : are filter banks necessary ? *In Proc. the IEEE CVPR*, volume 2.

[155] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. et Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[156] Venkatakrishnan, S. V., Bouman, C. A. et Wohlberg, B. (2013). Plug-and-play priors for model based reconstruction. *In 2013 IEEE Global Conference on Signal and Information Processing*, pages 945–948. IEEE.

[157] Wei, L. et Levoy, M. (2000). Fast texture synthesis using tree-structured vector quantization. *In Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488.

[158] Wei, L.-Y., Lefebvre, S., Kwatra, V. et Turk, G. (2009). State of the art in example-based texture synthesis. *In Eurographics 2009, State of the Art Report, EG-STAR*, pages 93–117. Eurographics Association.

[159] Wu, Y. N., Zhu, S. C. et Liu, X. (2000). Equivalence of Julesz ensembles and FRAME models. *International Journal of Computer Vision*, 38(3):247–265.

[160] Xu, L., Ren, J. S., Liu, C. et Jia, J. (2014). Deep convolutional neural network for image deconvolution. *Advances in neural information processing systems*, 27.

[161] Yu, G., Sapiro, G. et Mallat, S. (2012). Solving inverse problems with piecewise linear estimators : From Gaussian mixture models to structured sparsity. *IEEE Trans. Image Process.*, 21(5):2481–2499.

[162] Zhang, K., Li, Y., Zuo, W., Zhang, L., Van Gool, L. et Timofte, R. (2021). Plug-and-play image restoration with deep denoiser prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[163] Zhang, K., Zuo, W., Chen, Y., Meng, D. et Zhang, L. (2017a). Beyond a Gaussian denoiser : Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155.

[164] Zhang, K., Zuo, W., Gu, S. et Zhang, L. (2017b). Learning deep CNN denoiser prior for image restoration. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3929–3938.

[165] Zhang, K., Zuo, W. et Zhang, L. (2018). FFDNet : Toward a fast and flexible solution for CNN-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622.

[166] Zhang, S., Gao, Y., Jiao, Y., Liu, J., Wang, Y. et Yang, C. (2019). Wasserstein-Wasserstein auto-encoders. *arXiv preprint arXiv :1902.09323*.

[167] Zhu, S. C., Wu, Y. N. et Mumford, D. (1998). Filters, random fields and maximum entropy (FRAME) : towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126.

[168] Zoran, D. et Weiss, Y. (2011). From learning models of natural image patches to whole image restoration. *In 2011 International Conference on Computer Vision*, pages 479–486. IEEE.